Bauhaus-Universität Weimar
Fakultät Medien
Studiengang Mediensysteme

Interaktive physikalisch basierte kinematische und mit einem Datenhandschuh gesteuerte Echtzeitanimation von virtuellen Handpuppen

Bachelorarbeit

Daniel Bierwirth
geb. am: 24.07.1982 in Potsdam

Matrikelnummer 30510

1. Gutachter: Prof. Dr. Charles Wüthrich
2. Gutachter:

Datum der Abgabe: 13. Juni 2008

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Die Arbeit wurde weder in dieser oder einer ähnlichen Form noch in Auszügen bereits einer Prüfstelle vorgelegt.

Weimar, 13. Juni 2008 Daniel Bierwirth

Zusammenfassung

Im Bereich der interaktiven Computeranimation von artikulierten Figuren gibt es zahlreiche unterschiedliche Ansätze. Dazu zählen unter anderem Animationssysteme bei welchen über das *Motion Capturing Verfahren* gewonnene Bewegungsabläufe von virtuellen Charakteren in Echtzeit ausgeführt werden.
Im Gegensatz dazu wird in dieser Arbeit ein *physikalisch basiertes Echtzeit-Animationssystem* vorgestellt, mit dem beliebige Modelle interaktiv mit einem Datenhandschuh animiert werden können. Dafür wurde die herkömmliche Spielweise der Puppenspieler mit echten Handpuppen in abstrakter Form umgesetzt, indem ein virtuelles Handmodell zur grafischen Repräsentation eingeführt wurde.
Darüber hinaus wird im Verlauf der Arbeit gezeigt, wie neben der handgesteuerten Animation beliebige kinematische Bewegungsmodelle interaktiv definiert und in Echtzeit physikalisch basiert bewegt werden können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Die 3D Animation von virtuellen Charakteren wird in vielen unterschiedlichen Varianten mit dem Ziel umgesetzt, artikulierte Figuren im drei dimensionalen Raum glaubhaft zu animieren. In der Regel findet man diese animierten Figuren in 3D Spielen oder Filmen, bei denen die Bewegung der Figuren häufig im Voraus Frame für Frame berechnet wurde. Im Unterschied dazu soll diese Arbeit zeigen, wie beliebige Modelle interaktiv in Echtzeit kinematisch oder nicht-kinematisch gesteuert werden können. Dazu wird ein geeignetes Lösungsverfahren präsentiert, welches es erlaubt beliebige Skelettstrukturen mit einem Datenhandschuh auf verschiedene Weise zu animieren.
Die erste Möglichkeit der Objektanimation bildet im Wesentlichen die Abstraktion des klassischen Puppenspielers, bei der ein Nutzer die Puppe über das virtuelle Handmodell steuert. Mit Hilfe des Datenhandschuhs und dem dazugehörigen abstrahierten Handmodell kann die virtuelle Puppe so gesteuert werden, als wenn man mit der Hand in eine echte Latex Puppe fassen und diese bewegen würde. Darüber hinaus ist es möglich, die Objekte kinematisch zu animieren, indem die Fingerbewegung des Datenhandschuhs entweder auf je einen (End-)Effektor unterschiedlicher kinematischer Verkettungen übertragen oder mehrere Effektoren einer Verkettung animiert werden.
Diese Art der Animation entstammt zwar nicht der Puppen-Animation im klassischen Sinn, bietet jedoch die Chance durch einfach zu erstellende kinematische Verkettungen komplexe Objektbewegungen zu erzeugen, ohne das dafür Hintergrundwissen über die dem Puppenmodell zugrunde liegende Datenstruktur erforderlich ist.
Das in dieser Arbeit beispielhaft entwickelte physikalisch basierte Echtzeit-Animationssystem ist Teil einer Kooperation der Bauhaus Universität Weimar mit der ABS-CBN Foundation [WAB+06] und soll zeigen, wie es Künstlern durch den Bezug auf reale-Welt Abstraktionen möglich gemacht werden kann, Handpuppen in Echtzeit interaktiv zu bewegen, ohne dass dafür komplexes Wissen über die 3D Computeranimation erforderlich ist. In diesem Zusammenhang ergibt sich auch die Motivation für diese Arbeit. Derzeit werden die Latex Handpuppen in aufwendigen Produktionsprozessen hergestellt, bevor sie im Fernsehenstudio vor einem *Blue-Screen* gefilmt werden können. Dieser Vorgang soll über die in dieser Arbeit entwickelte physikalisch basierte Echtzeit-Puppenanimation mittels der Datenhandschuhsteuerung vereinfacht werden.

Kapitel 2

Überblick über die physikalisch basierte Charakteranimation

Um diese Arbeit in ihrem Kontext erfassen zu können, sollte man wissen wie Computer-Animationstechniken im Allgemeinen funktionieren und wie sie im Speziellen angewendet werden. Die hier vorgestellte Art der Animation basiert größtenteils auf der *Cyclic Coordinate Descent Methode* [WC91, Wel93], welche weiterentwickelt und speziell für die interaktive Echtzeitanimation von Puppenmodellen optimiert wurde.
Ein idealer Aktor beinhaltet Muskeln und Fett und verformt sich auf der Oberfläche in Abhängigkeit zu den ausgeführten Bewegungen [MTPS08]. Darüber hinaus ist an die Animation von Kleidung, welche in Relation zu den Bewegungen Falten wirft oder wackelt [KC02, TPS08] und an die Animation der Haare und Gesichtsausdrücke zu denken, um die Modelle glaubhaft zu bewegen und realistisch abbilden zu können. Für sich genommen stellen diese Eigenschaften jeweils komplexe Aufgabenbereiche dar, deren wechselseitiges Zusammenspiel sich noch in der Entwicklung befindet.
Dies gilt besonders für komplexe Materialien wie Haare oder ganze Fellstrukturen von virtuellen Puppen, die aus vielen tausend Elementen zusammengesetzt sind. Diese interagieren untereinander und mit dem Objekt auf vielfältige Weise. Jeder einzelnen Strähne dieser Strukturen liegt aufgrund der dünnen Objektform und der unterschiedlichen Ausprägungen, wie zum Beispiel glattes oder lockiges Fell, ein nichtlineares mechanisches Verhalten [HCBM07] zugrunde. Besonders für dünne Materialien wie Haare und Kleidung liegt die Herausforderung darin, durch geeignete Abstraktionen der realen Welt, in Echtzeit realistisches Verhalten von elastischen Stoffen zu modellieren.
Solche Materialien tendieren in der Regel dazu, nach der Bewegung wieder in die Ruheposition einzutreten und prallen beim Aufeinandertreffen mit mit anderen Objekten nicht ab. In diesem Zusammenhang muss das Kollisions- und Kontaktproblem verschiedener Materialien gelöst werden, um zum Beispiel das Eindringen der simulierten Kleidung in ein Objekt zu verhindern. Dieses Bewegungsverhalten und die Deformation von dünnen Stoffen wird in [TPS08] mit Hilfe der *Finite-Elemente-Methode* simuliert.
Die in dieser Arbeit genutzte physikalisch basierte Oberflächensimulation basiert auf dem in [PW07] vorgestelltem *Feder-Masse-System*. Diesem liegt auf der innersten Schicht des Puppenmodells eine feste Skelettstruktur zugrunde. Im Prinzip entspricht diese genau dem reduzierten Puppenmodell, dient zur Objektstabilisierung und be-

schränkt die Bewegung der Oberflächenpunkte. Auf dieser Basisschicht können über Feder-Masse Beziehungen verschiedene dünne und deformierbare Materialschichten zur Repräsentation von Fett, Muskeln oder Haut aufgelegt werden.

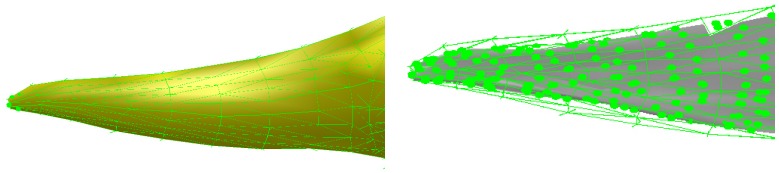

Abbildung 2.1: Darstellung der physikalischen Objektschichten.

In der Abbildung 2.1 wird auf der rechten Seite das Schichtenmodell beispielhaft anhand eines Teils einer Puppe dargestellt. Die unterste Objektschicht ist das grau gezeichnete reduzierte Puppenmodell, welches mit der Oberfläche über Feder-Masse Beziehungen verbunden ist. Diese wird in der Abbildung 2.1 als grüne Gitterstruktur dargestellt. Zu jedem Oberflächenpunkt einer physikalisch simulierten Puppe gehören demnach zwei Massepunkte. Ein innerer Massepunkt entspricht genau einem Objektpunkt auf der zur Objektstabilisierung genutzten Schicht des reduzierten Puppenmodells. Als Gegengewicht dazu existiert ein äußerer Massepunkt der außhalb der Objekthülle liegt. Erregt wird das Feder-Masse-System und damit die physikalische Oberflächensimulation, indem eine Kraft [KC02] auf einen der Massepunkte ausgeübt wird.
Da die Objekt- und Oberflächenanimation über die Bewegung des reduzierten Objektkerns des Schichtenmodell gesteuert wird, reduziert sich das Animations- und Definitionsproblem kinematischer Verkettungen auf die Kontrolle und Bewegung der untersten Objektschicht. Diese ist von der Art der Oberflächenvisualisierung und Detailauflösung unabhängig, sodass für die in dieser Arbeit implementierte Handsteuerung und die kinematische Animation der Objekte die reduzierte Objektschicht des Puppenmodells genutzt wird. Die innerste Skelettschicht des Puppenmodells entspricht im Wesentlichen der in [DQ04] vorgestellten *Medialen Achse* und wird genutzt, um die interaktiv vom Benutzer definierten Ersatzstrukturen für die Bewegungsanimation mit dem eigentlichen Puppenmodell zu verknüpfen. Dies ist möglich, weil die Objektschicht zum einen dazu genutzt wird, um die mit ihr verknüpften Massepunkte in Abhängigkeit von der ausgeführten Bewegung zu modifizieren und zum anderen weil die *Medialen Achse* wichtige Informationen[DQ04] über die Topologie und Orientierung der beweglichen Objektteile liefert.
Der Vorteil dieser Art der interaktiven Definition und Echtzeitanimation von Bewegungsmodellen liegt darin, dass aufgrund der physikalischen Oberflächenanimation realistische Bewegungsergebnisse erzielt werden können, welche zum Beispiel dem in [KJZM04] vorgestellten System fehlen. Darüber hinaus bietet der Einsatz von Ersatzstrukturen jedem Nutzer die Möglichkeit, Bewegungsmodelle zu editieren und zu animieren, ohne dass dieser sich mit dem für die physikalisch-basierte Oberflächenanimation genutzten Feder-Masse-System auseinander setzen muss.
Die Strukturen für die Bewegung der einzelnen Objektteile werden als geometrische Primitive oder abstrahierte Knochenmodelle dargestellt, welche über Gelenkmodelle

miteinander verbunden sind. Diese Verbindungspunkte werden in der Computeranimation in der Regel als Rotations- oder Translationsgelenke mit unterschiedlichen Freiheitsgraden dargestellt. Im Kontext der Positionsbeschreibung steht je ein Freiheitsgrad *(DOF)* für eine der drei Rotationsachsen beziehungsweise für eine der drei Ebenen im 3D Raum. Ein in der *XY-Ebene* veränderbares Objekt besitzt beispielsweise *1 DOF*

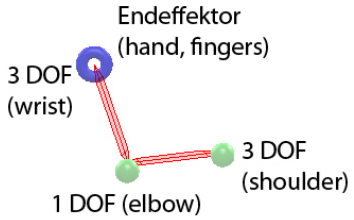

Abbildung 2.2: Schematische Darstellung eines einfachen Armmodells.

und wenn es zusätzlich in der *XZ-Ebene* bewegt werden kann *2 DOF*. Wird darüber hinaus die Orientierung zur Lagebeschreibung des Objekts im 3D Raum mit einbezogen, stehen insgesamt *6 DOF* zur Positions- und Orientierungsänderung zur Verfügung. Auf diese Weise lassen sich wie in [Par08] vorgestellt, mehrere Objektteile miteinander verbinden, um zum Beispiel das kinematische Verhalten des Arms zu simulieren. Das entsprechende Körperteil setzt sich wie in Abbildung 2.2 gezeigt, aus drei beweglichen Abschnitten zusammen, die vereinfacht dargestellt als Verkettung mit *7 DOF* angenommen werden.
Die Basis der in Abbildung 2.2 dargestellten Verkettung stellt die Abstraktion des Kugelgelenks mit drei Freiheitsgraden dar. Daran knüpft das Schultergelenk mit einem Freiheitsgrad an, bevor der letzte Verbindungspunkt den Endeffektor mit drei Freiheitsgraden mit dem Rest der Verkettung verbindet. Die verschiedenen virtuellen Gelenkabstraktionen lassen sich alle über die einfachen *1 DOF* Verbindungen zwischen zwei Elementen beschreiben, indem wie in [Par08] gezeigt, ein Kugelgelenk durch 3 übereinandergelagerte Scharniergelenktypen repräsentiert wird.
Zusätzlich zu der unterschiedlichen Anzahl von Freiheitsgraden werden die unterschiedlichen Gelenktypen simuliert, indem die natürlichen Beschränkungen in die Bewegungsanimation mit einfließen. Die Einschränkung des gültigen Arbeitsbereichs an einem Gelenkpunkt ist notwendig, um zu vermeiden, dass während der Endeffektorbewegung unnatürliche kinematische Posen entstehen. In Bezug auf das abstrahierte Scharniergelenk des menschlichen Ellbogen in Abbildung 2.2 ergibt sich demnach die Beschränkung des Öffnungswinkels auf $180°$.
Die einzelnen Verkettungen der zu animierenden Objekte werden im lokalen Koordinatensystem der Verbindungen beschrieben und durch lokale Rotationen um die entsprechenden Gelenke animiert. Es ist möglich das ein Objekt durch mehrere unabhängige Verkettungen definiert wird oder als komplexes hierarchisch aufgebautes Animationsmodell in Form einer Baumstruktur modelliert wird.

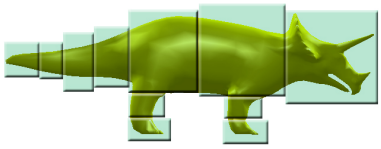

Abbildung 2.3: 2D Darstellung der beweglichen Teile eines Objekts.

2.1 Einführung in die physikalisch basierte Animation

Es gibt im Wesentlichen zwei Möglichkeiten für die Bewegung von Objektteilen, wobei sich diese auf die Rotations- beziehungsweise Translationsbewegungen beziehen. Bei der physikalisch basierte Rotationsbewegung werden Objektpunkte die mit der zur Animation herangezogenen Ersatzstruktur verknüpft sind, im Allgemeinen um den entsprechenden Basispunkt der Ersatzstruktur rotiert.
In Abbildung 2.4 wird die Anwendung der physikalisch basierten Animation gezeigt. Damit der Kopf einer Puppe auf ihrem Hals zu rotiert, wird das Modell in zwei bewegliche Objektteile zerlegt, die je durch eine Ersatzstruktur referenziert werden.

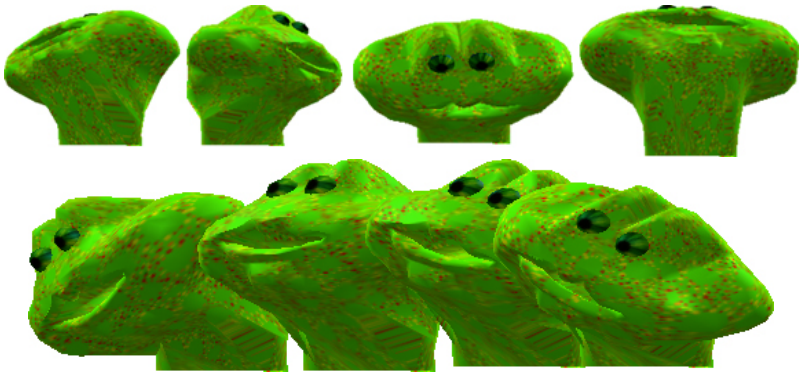

Abbildung 2.4: Physikalisch basierte Kopfbewegung.

Um den Kopfteil zu animieren, wird das entsprechende Element am äußeren Endeffektor durch den Nutzer gesteuert. Dies hat zur Folge, dass der Rotationspunkt dieser Gruppe genau der Verbindungspunkt zwischen beiden Objektgruppen ist. In den Darstellungen 2.5 und 2.6 ist der aktive Endeffektor jeweils durch einen blau gezeichneten Torus hervorgehoben. Wird dieser, wie in der unteren Grafik der Abbildung 2.6 dargestellt, durch den Nutzer bewegt, wird die Bewegung in Echtzeit auf die mit der Ersatzstruktur verbundenen Punkte der starren Skelettschicht des Puppenmodells übertragen. Die zur Repräsentation der durch den Nutzer definierten beweglichen Objektteile genutzte Ersatzstruktur ist in den Abbildungen 2.5 und 2.6 durch rot gezeichnete Knochen visualisiert. Da zur Bestimmung der aus der neuen Endeffektorposition entstandenen Pose alle Elemente einer Verkettung einbezogen werden, würde für den Fall der kinematischen Animation bei jeder Endeffektorbewegung nicht nur das direkt mit dem Endeffektor verbundene Objektteil rotiert.

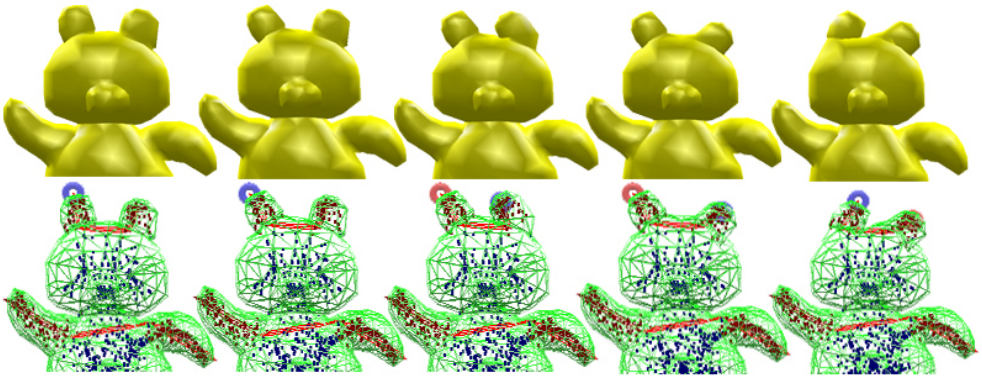

Abbildung 2.5: Physikalisch basierte Bewegung der Ohren eines Teddys.

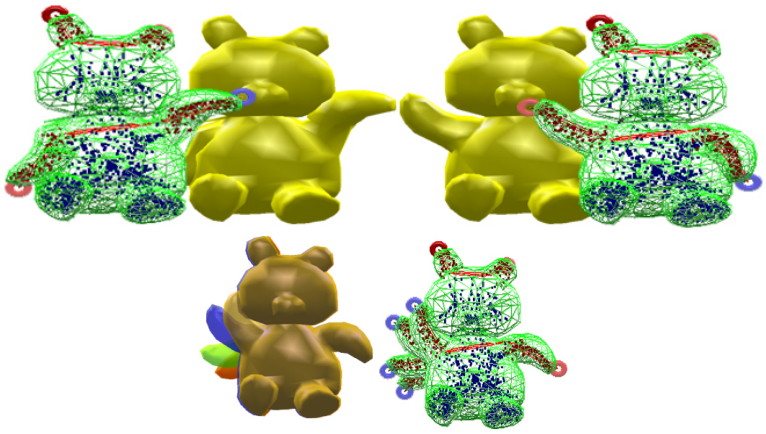

Abbildung 2.6: Physikalisch basierte Bewegung der Arme eines Teddys.

Im Fall der gleichzeitigen Bewegung von verschiedenen Effektoren, wie dies durch die handgesteuerte Animation möglich ist, kann es unter Umständen sinnvoll sein, dass mehrere bewegliche Objektteile einen gemeinsamen Rotationspunkt besitzen. Wie sich in den folgenden Kapiteln zeigen wird, eignet sich diese Variante speziell für die Implementierung eines abstrahierten virtuellen Handmodells, bei der die Freiheitsgrade und Rotationen für hierarchische Animationsstrukturen durch die Lage des Handgelenks, der Distanz zwischen Handschuh und Empfänger und dem aktuellen Fingerbeugegrad beeinflusst werden. Diese Art der Animation wird in erster Linie für Mund- und Gesichtsbewegungen genutzt, da hier die größten Gemeinsamkeiten zu der Spielweise echter Puppenspieler zu finden sind.

2.2 Verwandte Arbeiten

Es gibt verschiedene Ansätze die sich mit der Echtzeitanimation von virtuellen Figuren auseinander setzten. Diese lassen sich im Wesentlichen in zwei Kategorien einteilen. Bei der ersten Kategorie steht die Frage der Echtzeit-Interaktion zwischen Nutzern und automatisierten Aktoren im Vordergrund. Bei dem in [Bat94] vorgestellten Prototypen reagieren virtuell generierte Charaktere in Form von vorher definierten Aktionen auf Ereignisse der realen Umwelt. Solchen autonomen Figuren liegt in der Regel ein komplexer Datensatz mit vorgeschriebenen Bewegungsabläufen zugrunde. Aus diesen wird die zum Ereignis passende Bewegung heraus gefiltert und vom autonomen Aktor ausführt. Dadurch ist es möglich, virtuellen Figuren abstrahierte Bewegungen echter Lebewesen zuzuschreiben, um zum Beispiel das Verhalten eines Fischschwarms [Rey87] auf verschiedene Ereignisse zu simulieren.
Die zweite Kategorie setzt sich weniger mit der Frage der automatisierten Bewegung, als mit dem Aspekt der Nutzer gesteuerten Echtzeitanimation auseinander. Für diese Art der interaktiven Animation von Objekten durch den Benutzer gibt es verschiedene Techniken, um die Bewegung von virtuellen Figuren umzusetzen. Das in [WCL04] vorgestellte Projekt nutzt für die handgesteuerte Animation von Figuren durch *Motion Capture Techniken* gesammelte Informationen über Bewegungsabläufe. Diese können mit dem Datenhandschuh imitiert und modifiziert werden, um die virtuellen Objekte in Echtzeit zu animieren. Als Grundlage der Bewegungsbeschreibung dienen zeitliche Beschränkungen, die für jeden Zeitpunkt t der Bewegungsdauer die spezifische Stellung der kinematisch animierten Elemente beschreibt. Die von der Puppe auszuführende Bewegung wird anhand der vom Benutzer mit dem Handschuh durchgeführten Fingerbewegungen interpretiert und durch diese modifiziert wiedergegeben.
In diesem Zusammenhang steht auch das in [KTT07] vorgestellte System, bei dem der Datenhandschuh nicht zur direkten Bewegung eines Objektes genutzt wird, sondern mittels bestimmter Handgesten damit verbundene Bewegungen der Figur auslöst. Auch bei diesem Ansatz besteht das Problem, dass der Benutzer die Bewegungen eines Puppenmodells nicht interaktiv und frei bestimmen kann. Dennoch ist der Ansatz der gestengesteuerten Manipulation heranzuziehen, um neben der direkten handgesteuerten Bewegung von Charakteren zusätzlich gestengesteuerte Emotionen zu animieren. Zum Beispiel ist ein System mit zwei Datenhandschuhen denkbar, bei dem der eine die in dieser Arbeit vorgestellte handgestützte Animation steuert und der andere zur Umsetzung der gestengestützten Emotionssteuerung genutzt wird.
Während der Mund einer Puppe handgestützt gesteuert wird, bietet es sich an die animierte Mundbewegung durch zusätzliche Details wie zum Beispiel mittels der in [Fei01] vorgestellten *Reflective Textures Methode* zu simulieren. Diese Technik erlaubt es über Texturverschiebungen auf der Objektoberfläche zusätzliche Details wie zum Beispiel das Öffnen oder Schließen der Augen zu simulieren.
Im Prinzip bietet es sich an, die in [Fei01] erläuterte Methode in Form von zusätzlichen dünnen semi-transparenten Objektschichten mit der bereits implementierten physikalisch basierten Oberflächensimulation dieser Arbeit zu verknüpfen.
Bei dem in [DYP04] vorgestellten Ansatz hat der Nutzer die Möglichkeit die Echtzeitanimation von virtuellen Charakteren über zahlreiche Hilfsmittel[DYP04] zu steuern. Diese Objekte werden von dem *Motion Capture System* in Echtzeit lokalisiert und erlauben es dem Nutzer interaktiv Bewegungsabläufe vorzugeben, die von der virtuellen Puppe in Echtzeit ausgeführt werden.
Diese animierten Figuren sollen keineswegs wie reale-Welt Vorbilder wirken, aber dafür umso lebendiger sein. Für die Animation virtueller Puppen ist nicht nur die mecha-

nisch korrekt beschriebene Bewegung entscheidend, sondern die Vielzahl animierter Details, welche den Figuren in gewisser Weise eine Persönlichkeit zuschreibt. Dies wird zum Beispiel bei Mr. Bubb[LRBW04] deutlich, der sich freut, wenn ein Nutzer mit ihm interagiert und Ball spielt oder aber traurig ist, falls er den Ball nicht erhält.

Kapitel 3

Einführung in die Puppenanimation

Die Animation von Objekten kann auf verschiedene Arten realisiert werden. Zum einen ist es möglich, die Oberflächenstruktur der Modelle direkt zu manipulieren und zum anderen mit geometrischen Repräsentationen der manipulierbaren Objektteile zu arbeiten. Solche Darstellungen sind im Allgemeinen Abstraktionen von realen Skelettstrukturen oder aber auch geometrische Primitive, die mit dem zugrunde liegenden Puppenmodell verbunden sind.

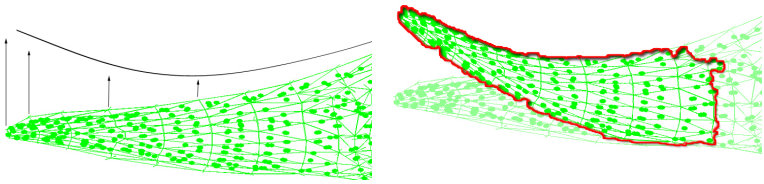

Abbildung 3.1: Beispiel einer Objektmanipulation durch direkte Deformation der Gitterstruktur.

In Bezug auf das Puppen-Animationssystem kommen Ersatzstrukturen zum Einsatz, die auf der einen Seite mit den beweglichen Teilen des zu animierenden Objekts verknüpft sind und auf der anderen Seite dem Nutzer die nötigen Interaktionsmöglichkeiten für die Animation bieten. In der Konsequenz bleibt die eigentliche Netz- und Oberflächenstruktur des Objektes zu jedem Zeitpunkt für den Benutzer verborgen. Da für die Berechnung der Bewegungsanimation nur die Ersatzstrukturen genutzt werden, bietet der Einsatz dieser Elemente den Vorteil, dass der eigentliche Algorithmus, der zur Animation der Puppen herangezogen wird, für beliebige Objektmodelle mit unterschiedlichen Strukturaufbau gültige Ergebnisse liefert.

In dieser Arbeit wurden zwei Ansätze verfolgt, mit denen der Nutzer die der Animation zugrunde liegende Struktur definieren kann. Dazu zählt einerseits ein auf Skelettstrukturen basierter und anderseits ein auf *Bounding Volume Hierarchien*[ESHD05] basierter Ansatz, bei dem die beweglichen Objektteile über virtuelle Boxen bestimmt werden. Beide unterscheiden sich grundlegend in der Art und Weise, mit der ein Benutzer die

einzelnen beweglichen Objektteile definiert. Im Folgenden werden beide Varianten näher vorgestellt, bevor in Kapitel 5 auf den eigentlichen Algorithmus zur kinematischen Bewegung eingegangen wird.

3.1 Modellierung kinematischer Strukturen

Jede Bewegung der definierbaren Objektteile bezieht sich immer auf andere Elemente der Ersatzstruktur oder aber auf das Objekt selbst. Ein einzelnes Element, welches innerhalb eines Objekts rotiert wird, ergibt in Relation zum restlichen Teil der Puppe die Rotation der mit der Ersatzstruktur verbunden Puppenpunkte um den Ankerpunkt des Knochens. Betrachtet man ein Modell, das aus mehreren beweglichen Objektteilen zusammengesetzt ist, wird die Bewegung eines Objektteils relativ zu seinem Nachbarn beschrieben. Dies lässt sich geometrisch als Verkettung von Boxen oder Knochen darstellen, welche durch abstrahierte Gelenke beziehungsweise *Links* miteinander verbunden sind.
Das eigentliche Verfahren der Verknüpfung der geometrischen Repräsentationen mit den tatsächlich für die Animation der Puppe relevanten Punkten hängt stark vom internen Aufbau des Puppenmodells ab. Es wurde bereits erwähnt, dass der Visualisierung der Objekte eine komplexe Oberflächensimulation zugrunde liegt, die der Arbeit [PW07] entnommen wurde. Bei dieser Art der Modellvisualisierung wird die Punktwolke, die in der Regel die in der Abbildung 3.2 grün dargestellte Objekthülle beschreibt, zur Herleitung einer reduzierten Objektschicht genutzt. Ausgehend von den Punkten der reduzierten Objektschicht wird die Objektoberfläche der Puppe durch Feder-Masse Beziehungen animiert. Jeder Punkt der in der Abbildung 3.2 grau gezeichneten starren Skelettschicht repräsentiert demnach mehrere Oberflächenpunkte in seinem Einflußbereich.

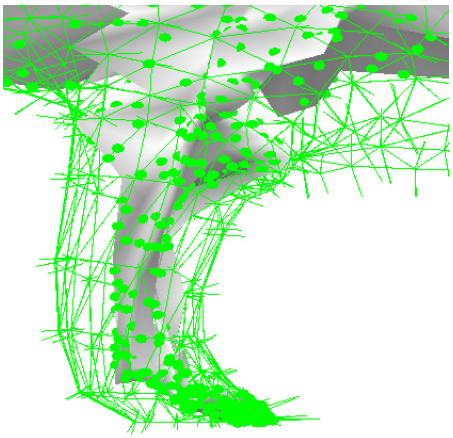

Abbildung 3.2: Darstellung des reduzierten Puppenmodells und der entsprechenden Objekthülle.

Das Objekt selbst wird zwar nach wie vor in Form einer triangulierten Punktwolke beschrieben, allerdings in Relation zum starren Skelett und der entsprechenden Massepunkte der Oberfläche selbst. Zur Bestimmung der beweglichen Objektteile wird im

Idealfall auf das reduzierte Puppenmodell zurückgegriffen. Durch die lokale Definition des Einflußbereichs einer Ersatzstruktur an der innersten Objektschicht, wird die für die physikalisch basierte Oberflächensimulation genutzte starre Skelettschicht entsprechend der definierten Objektteile zerlegt. Dadurch ist es möglich, dass die kinematische Bewegungsberechnung über die durch den Benutzer festgelegten Ersatzstrukturen erfolgt, indem die Bewegungsergebnisse auf die innere Skelettschicht des Puppenmodells übertragen werden. Da die Punkte dieses reduzierten Puppenmodells genau den inneren Massepunkten der Oberfläche entsprechen, wird auf diese Weise durch die kinematische Bewegung der Objektteile das Feder-Masse-System zur Oberflächensimulation erregt.

3.1.1 Modellierung mit Boxen

Die erste Möglichkeit zur Bestimmung von beweglichen Objektteilen ist einfach aber dennoch effektiv. Um Teile einer Puppe als bewegliche Objektgruppe zu deklarieren, wird der gewünschte Bereich durch eine virtuelle Box[ESHD05] markiert. Der Ansatz der *Bounding Volume Hierarchien* lässt sich besonders gut auf Objekte übertragen, deren reale-Welt Repräsentationen kein Skelett besitzen. Im Unterschied zu der nachfolgend vorgestellten Methode, welche auf der Anwendung von Skelettstrukturen basiert, muss der Nutzer in diesem Fall die Ersatzstrukturen nicht innerhalb des Objekts platzieren, sondern markiert den relevanten Bereich auf der Objekthülle. Des Weiteren entfällt die explizite Definition des Einflußbereichs der Ersatzstruktur, da diese implizit über den durch virtuelle Boxen eingeschlossenen Objektbereich, wie in Abbildung 3.3 dargestellt, bestimmt ist. Um einen bestimmten Abschnitt einer Puppe als beweglich zu markieren, werden die *Bounding Boxes* über dem entsprechenden Objektteil aufgespannt. Im nächsten Schritt kann der Benutzer die definierten Objektteile mit der im Zusammenhang mit den Skelettstrukturen erläuterten Methode kinematisch animieren. Dafür werden die einzelnen *Bounding Boxes* miteinander verknüpft und das entsprechende Rotationsverhalten, am Verbindungspunkt zwischen beiden Elementen, über die in Kapitel 8 vorgestellten Beschränkungen festgelegt.

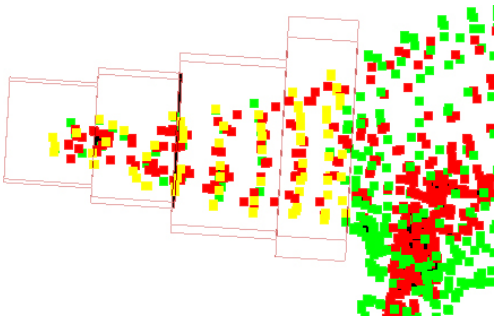

Abbildung 3.3: Beispiel der Selektion von beweglichen Objektteilen mit virtuellen Boxen.

3.1.2 Modellierung mit Skelettstrukturen

Die zweite Möglichkeit zur Definition von animierten Objektteilen basiert auf der Anwendung von skelettbasierten Strukturen, die aus Knochen mit zwei Endpunkten und Gelenkabstraktionen, die jeweils ein Paar von Knochen verbinden, besteht. Die Idee der skelettbasierten Animation ist weit verbreitet und kann auf verschiedene Typen von Objekten, unabhängig davon ob deren reale-Welt Repräsentationen ein Skelett besitzen, angewendet werden.

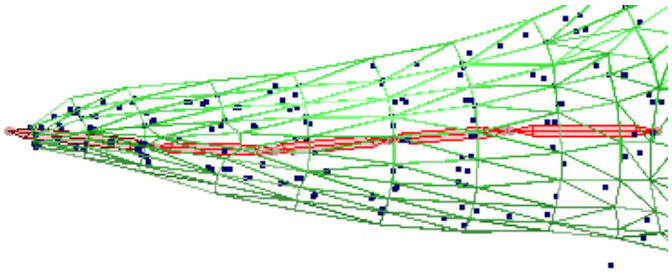

Abbildung 3.4: Beispiel der Selektion von beweglichen Objektteilen mit Skelettstrukturen.

Intern werden solche Verkettungen von beweglichen Objektteilen, die auch als *Connections* bezeichnet werden, als Baumstrukturen dargestellt. Die direkten Kinder des Wurzelelements sind dabei immer die Verbindungspunkte der Verkettung mit den eigentlichen Knochen als Blätter des Baums.

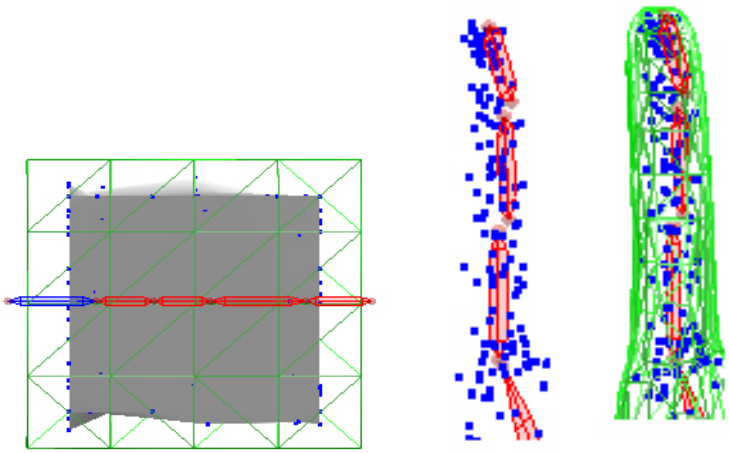

Abbildung 3.5: Anwendung der Skelettstrukturen für verschiedene Objekte.

Der spezifische Einflussbereich der Knochen auf bestimmte Objektteile wird über mit *3 DOF* definierbare Ellipsoide festgelegt, die ihren Ursprung jeweils im Zentrum eines

Knochens haben. Dadurch wird der Einflussbereich nicht nur durch die Skelettposition bestimmt, sondern wird relativ zum Knochen über den Ellipsoid definiert.

Um festzustellen, ob ein Punkt *P(X, Y, Z)* innerhalb des durch den Ellipsoid definierten Bereichs liegt, muss *P* in das lokale Koordinatensystem des Ellipsoids transformiert werden, bevor die Gleichung 3.1 ausgewertet werden kann.

$$r = \left[\left(\frac{x}{a_x} \right)^{\frac{2}{e}} + \left(\frac{y}{a_y} \right)^{\frac{2}{e}} \right]^{\frac{e}{n}} + \left(\frac{z}{a_z} \right)^{\frac{2}{n}} \tag{3.1}$$

Das Ergebnis *r* beschreibt die Lagebeziehung zwischen Punkt *P* und dem Ellipsoid. Wenn *r < 1* dann liegt der Punkt innerhalb, bei *r > 1* außerhalb und bei *r = 1* genau auf der Oberfläche des Ellipsoids. Die Faktoren *e* und *n* dienen zur genaueren Beschreibung des Krümmungsverhaltens der ellipsoiden Oberfläche und können in der Regel als *e = 1* und *n=1* gesetzt werden. Die Variablen a_x, a_y und a_z in der Gleichung 3.1 beschreiben die symmetrische Ausdehnung des Ellipsoids entlang der drei orthogonalen Koordinatenachsen des lokalen Koordinatensystems im Zentrum des Objekts.
Theoretisch kann der Einflussbereich eines Knochens in allen drei Dimensionen beliebig ausgebreitet werden. Dabei muss aber beachtet werden, dass ein zu groß gewählter Bereich unter Umständen unerwünschte visuelle Ergebnisse erzeugen kann. Dieser Fall kann zum Beispiel bei der Rotation um den Basispunkt eines Knochens in der *XY-Ebene* eintreten, falls Puppenpunkte die mit dem Knochen assoziiert sind, außerhalb des durch den Knochen belegten *X-Wertebereichs* positioniert sind.

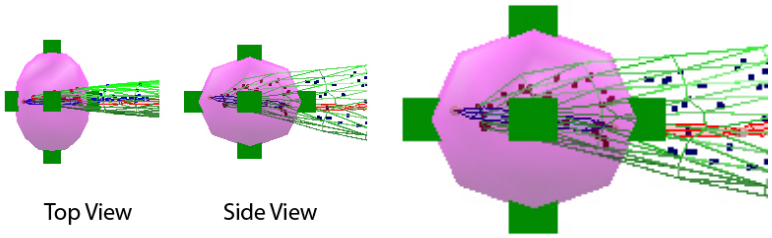

Abbildung 3.6: Ellipsoide definieren den Einflussbereich der Skelettstrukturen.

3.2 Kinematische Animationsmodelle

Im Kontext der Computeranimation ist die Kinematik die Lehre der Bewegung von Objekten oder Objektteilen im Raum ohne den Bezug auf die newtonschen Axiome, welche als Grundgesetze der Bewegung gelten [ESHD05]. Folglich ist es für die Bewegung von artikulierten Figuren in der Regel nicht von Bedeutung, wer der Erreger der Bewegung ist. Auch die Wechselwirkungen zwischen den vermeintlich wirkenden Kräften spielen für die konkrete Bewegung keine Rolle.
Im Folgenden werden einige der Methoden zur kinematischen Animation einleitend vorgestellt, bevor im Speziellen auf ausgewählte Methoden detaillierter einzugehen ist.

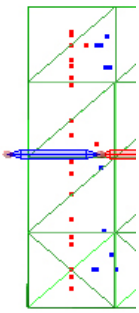

Abbildung 3.7: Die roten Punkte des reduzierten Modells markieren ein bewegliches Objektteil.

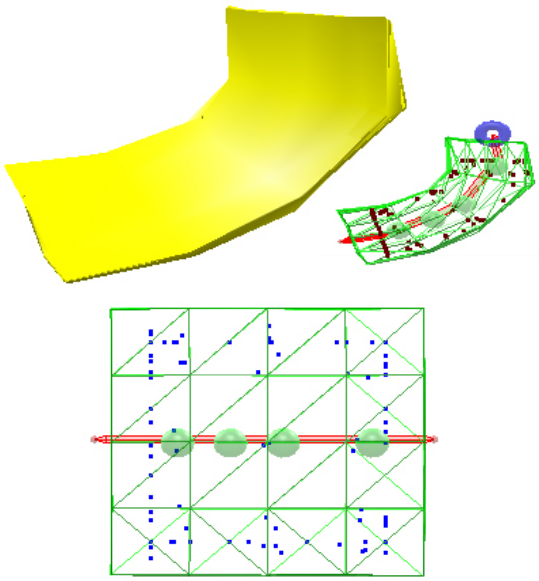

Abbildung 3.8: Kinematisch animiertes Objekt und die interne Animationsstruktur.

3.2.1 Vorwärts Animation

> Die *Forward Animation* ist der einfachste Weg, um bewegliche Objektteile zu animieren.[ESHD05][p. 45]

Typisch für diese Art der Bewegungssimulation ist die manuelle Rotation der Elemente der Verkettung um ihre entsprechenden Gelenke. Beginnend bei der Basis wird über jedes Kettenglied iteriert und dieses um den entsprechenden Verbindungspunkt rotiert. Im Ergebnis setzt sich die zu erwartende Artikulation aus der Gesamtheit der Bewegungen aller Kettenglieder zusammen, was die zielgerichtete Positionierung des Endpunkts einer Verkettung erschwert.
Daher eignet sich diese Methode nicht für die interaktive kinematische Animation von Modellen, da in diesem Fall der Endeffektor dazu benutzt wird, um die gewünschte Bewegungsänderung zu definieren. Dies wird bei dem Versuch, mittels der *Forward-Methode* eine Greif-Aktion zu bestimmen, besonders deutlich. Aufgrund der zahlreichen unterschiedlichen Möglichkeiten scheint es fast unmöglich die korrekte Konfiguration einer *Linkage* zu bestimmen, bei der der Endeffektor über die Vorwärts-Animation zielgerichtet zum Zielpunkt geführt werden kann. Für solche Fälle bietet sich die inverse kinematische Animation an.

3.2.2 Inverse Animation

Im Gegensatz zum vorhergehenden Verfahren wird bei inversen Animation in zweifacher Hinsicht ein inverser Ansatz verfolgt. Zum ersten geht bei der inversen Animation die Bewegung vom Endeffektor aus und setzt sich kinematisch bis zur Basis der Verbindung fort um die gewünschte Bewegung auszuführen. Zum anderen wird bei der inversen kinematischen Animation die inverse beziehungsweise die pseudoinverse Jacobi-Matrix zur Berechnung der *Linkparameter* herangezogen. Als Ausgangsbasis für die kinematischen Berechnungen wird die *Jacobi-Matrix* gebildet, welche eine *m x n* Matrix ist, bei der die Spalten *n* gleich der Anzahl der Verbindungen einer Verkettung sind und *m* die Dimension beziehungsweise die zur Verfügung stehenden Freiheitsgrade repräsentiert.
Wie in dem vorhergehenden Abschnitt erläutert wurde, muss bei der Vorwärts Animation jeder Linkparameter einer Verbindung bestimmt werden, um als Ergebnis die Endeffektorpositionierung zu erhalten. Im Unterschied dazu ist die inverse Methode Endeffektor bezogen, das heißt der Zielpunkt des Endeffektors wird festgelegt und danach erfolgt die Berechnung der dafür nötigen Gelenkbewegungen der Verkettung in Bezug auf den zu erreichenden Zielpunkt. Die grundlegende Eigenschaft, durch die sich die Jacobi Methode auszeichnet ist, dass alle *Links* einer Animationsstruktur in einem Iterationsschritt berechnet werden, indem je nachdem wie die Konfiguration aufgebaut ist, entweder die inverse oder die pseudoinverse Jacobi-Matrix gelöst werden muss.

3.2.3 Direkte Manipulation

Bei der direkten Animationsmethode geht die Bewegung wie bei der Inversen Animation vom Endeffektor aus. Dabei unterscheidet sich die Berechnung grundlegend von der kinematischen Animation mit der inversen Jacobi-Matrix. Die in [Ebe06, WC91, Wel93] vorgestellte *Cyclic Coordinate Decent Methode* basiert auf heuristischen Verfahren, welche zur Berechnung der Gelenkstellungen herangezogen werden. Ausgehend vom Endeffektor wird dafür an jedem Gelenkpunkt das entsprechende Element,

mit dem Ziel den Endeffektor möglichst nah dem Zielpunkt anzunähern, bewegt.
Im Gegensatz zur Jacobi Methode wird für die Bewegung in jedem Animationsschritt nicht die Pseudoinverse gelöst, sondern über alle *Links* iteriert und versucht an jedem Gelenk durch geeignete Rotationsbewegungen des Elements den Endeffektor zum Ziel zu bewegen. Die iterative Berechnung der Gelenkbewegungen resultiert nach jedem Iterationsschritt in der Bewegung des Endeffektors. Dies ermöglicht es an jedem Verbindungspunkt, durch Lösen von *Constraints*, verschiedene Gelenktypen zu simulieren.
Wie sich im Verlauf dieser Arbeit zeigen wird, eignet sich die *CCD Methode* für die interaktive kinematische Simulation von Puppenmodellen, da sich die nötigen Gelenkänderungen in wenigen Schritten herleiten lassen und der Nutzer die Modelle in gefühlter Echtzeit animieren kann.

3.3 Datenhandschuhgesteuerte Animation

Die Echtzeitanimation von Modellen eröffnet nicht nur Raum für verschiedene mathematische Herangehensweisen an die Computeranimation, sondern fordert auch eine Antwort auf die Frage der Steuerung simultaner Aktionen. Sollen Puppenmodelle interaktiv animiert werden, ist die Eingabe per Maus spätestens ab dem Zeitpunkt nicht mehr ausreichend, ab dem man mehrere Endeffektoren zur gleichen Zeit im Raum bewegen möchte. In Kapitel 2 sind bereits einige Systeme vorgestellt worden, die wie bei dem System von [KJZM04, DYP04] versuchen, dem Nutzer die Möglichkeit zu geben, mehrere virtuelle Objekte simultan im 3D Raum mit Hilfe von realen Hilfsmitteln zu manipulieren.
In dieser Arbeit wurde für die gleichzeitige Bewegung mehrerer Endeffektoren, der in [MH06] vorgestellte Ansatz für den drei dimensionalen Fall erweitert. Statt der Steuerung mit der bloßen Hand werden die beweglichen Objektteile der virtuellen Puppen mit einem Infrarot Datenhandschuh bewegt. Dadurch ist es möglich, bis zu fünf verschiedene Elemente eines Objektes gleichzeitig zu steuern, die wie in Kapitel 7 erläutert, mindestens in einem Freiheitsgrad voneinander unabhängig sind.
Um dem Nutzer die Steuerung der Handpuppen mittels des Handschuhs leicht zu machen, wird in dieser Arbeit das reale Handmodell in abstrahierter Form mit den durch die Fingerspitzen repräsentierten Elementen virtuell abgebildet. Des Weiteren dient die reale Spielweise der Puppenspieler für die virtuelle Handsteuerung, auf welche in Kapitel 7 detailliert eingegangen wird, als Vorbild. Das heißt, dass die Gesten von virtuellen Handpuppen so gesteuert werden, als ob man mit seiner Hand in den Hohlkörper einer echten Latexpuppe fasst. Dadurch kann mit einem Datenhandschuh die Puppe interaktiv als ganzes Objekt und über die Fingerspitzen des Handschuhs die beweglichen Mundbereiche einer Puppe in Echtzeit animieren.

Abbildung 3.9: Beispiel der mit einem Datenhandschuh gesteuerten Mundbewegung.

Abbildung 3.10: Mundbewegung zur Darstellung von Emotionen (von links: Normal, Fröhlich, Traurig, Lustig).

Kapitel 4

Inverse kinematische Animation

In diesem Abschnitt wird die Berechnung der kinematischen Animation von artikulierten Figuren mit der Jacobi Methode dargestellt. Dafür werden im Folgenden die wesentlichen Schritte zur Berechnung der nötigen Winkeländerungen an den Verbindungspunkten einer Verkettung vorgestellt. Des Weiteren wird gezeigt, dass die Jacobi Methode für die Echtzeitanimation von komplexen Bewegungsmodellen im Zusammenhang mit dem entwickelten Echtzeit-Animationssystem nur bedingt geeignet ist.

In der inversen Kinematik wird die Zielposition und möglicherweise auch die Orientierung eines Endeffektors durch die Interaktion des Nutzers bestimmt. Die dafür nötige Änderung der Position und Orientierung des Endeffektors wird über die Berechnung der Winkeländerungen an Verbindungspunkten zwischen den beweglichen Kettengliedern beschrieben. Es gibt zahlreiche unterschiedliche Lösungsverfahren zur Bestimmung der Winkeländerung der Gelenke, von denen der überwiegende Teil auf der Bildung der Jacobi-Matrix[BBG85] basiert. In Abhängigkeit von der aktuellen Konfiguration einer Verkettung und der Differenz zwischen Ausgangs- und Zielposition des Endeffektors kann das Problem keine, eine oder mehrere Lösungen besitzen. Im Allgemeinen wird ein solches System, dass keine Lösung besitzt, als *überbestimmt* und wenn es mehrere mögliche Lösungen besitzt als *unterbestimmt* [Leh83] bezeichnet. Wie sich im folgenden Teil dieses Kapitels zeigen wird, unterscheiden sich diese Systeme im Kontext der Jakobi Methode durch den strukturellen Aufbau der Jakobi-Matrix und den daraus folgenden unterschiedlichen Lösungsmethoden.

4.1 Mathematischer Überblick

Für den in Kapitel 3 beschriebenen Ansatz der Vorwärts Animation lässt sich das Problem der Bestimmung der Zielposition und Orientierung eines Endeffektors

$$S_g = [p_x, p_y, p_z, \varphi_x, \varphi_y, \varphi_z]^T \tag{4.1}$$

mathematisch in Abhängigkeit der Gelenkwinkel $\Theta(\theta_1, ..., \theta_n)$ einer Verkettung beschreiben als

$$S_g = f(\Theta(\theta_1, ..., \theta_n)) \tag{4.2}$$

Auf der anderen Seite gilt für die inverse Kinematik der entgegengesetzte Ansatz, bei dem über die Zielkonfiguration des Endeffektors mit Hilfe der inversen Funktion f^{-1}

die entsprechenden Gelenkparameter bestimmt werden.

$$\Theta(\theta_1, ..., \theta_n) = f^{-1}(S_g) \tag{4.3}$$

Aufgrund der Nichtlinearität der Funktion f ist die Lösung des inversen Problems nicht trivial, da es für eine bestimmte Endeffektorkonfiguration S_g unterschiedliche Lösungen für $\Theta(\theta_1, ..., \theta_n)$ geben kann. In der Konsequenz bietet sich als natürlicher Ansatz die Linearisierung des Problems an, indem die Änderungen der Gelenkwinkel

$$\dot{\Theta}(\dot{\theta_1}, ..., \dot{\theta_n}) = \left(\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}_{new} - \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \right) \tag{4.4}$$

mit der Änderungsrate des Endeffektors $V = (S(\theta_1, ..., \theta_n) - S_g)$ über die Jacobi-Matrix $J[\Theta(\theta_1, ..., \theta_n)]$, wie in Gleichung 4.5 dargestellt, in Beziehung gesetzt werden.

$$(S(\theta_1, ..., \theta_n) - S_g) = J[\Theta(\theta_1, ..., \theta_n)]\dot{\Theta}(\dot{\theta_1}, ..., \dot{\theta_n}) \tag{4.5}$$

$S(\theta_1, ..., \theta_n)$ beschreibt dabei die aktuelle Position und Orientierung des Endeffektors in Abhängigkeit von der aktuellen Konfiguration $\Theta(\theta_1, ..., \theta_n)$ der Gelenke der Verkettung.
Die lineare Beziehung zwischen $\dot{\Theta} = [\dot{\theta_1}, ..., \dot{\theta_n}]^T$ und $V = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^T$ wird über die partiellen Ableitungen in der Jacobi Matrix ausgedrückt, welche zu jedem Zeitpunkt eine Funktion der aktuellen Pose der kinematischen Verkettung darstellt.

$$J = \begin{bmatrix} \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} & \cdots & \frac{\partial p_x}{\partial \theta_n} \\ \frac{\partial p_y}{\partial \theta_1} & \frac{\partial p_y}{\partial \theta_2} & \cdots & \frac{\partial p_y}{\partial \theta_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \varphi_z}{\partial \theta_1} & \frac{\partial \varphi_z}{\partial \theta_2} & \cdots & \frac{\partial \varphi_z}{\partial \theta_n} \end{bmatrix} \tag{4.6}$$

In Bezug auf die kinematische Animation ist die *J* eine *m x n* Matrix, bei der die Spalten *n* gleich der Anzahl der Verbindungen einer Verkettung ist und *m* die Dimension beziehungsweise die zur Verfügung stehenden Freiheitsgrade mit *m = 3* bei einfachen Positionierungsaufgaben und *m = 6* bei Positionierungs- und Orientierungsaufgaben darstellt.
Zusätzlich zu der in Gleichung 4.6 dargestellten Definition der Jacobi-Matrix, wird in [Par08] ein vereinfachter geometrischer Ansatz zur Bestimmung von *J* vorgestellt, bei dem die Beziehungen der Matrixelemente direkt ablesbar sind.
Folgendes Beispiel soll die geometrische Herleitung der Jacobi-Matrix für eine Verkettung von drei beweglichen Elementen zeigen. Bei dieser stellt *E* die aktuelle Position des Endeffektors und P_1 beziehungsweise P_2 die aktuelle Position der Gelenke im 3D Raum dar. Der Achsenvektor $(0, 0, 1)^T$ bezeichnet dabei die aktuelle Rotationsachse, welche im vorliegenden Beispiel die z-Achse ist.

$$J = \begin{bmatrix} ((0,0,1) \times E)_x & (0,0,1) \times (E - P_1)_x & (0,0,1) \times (E - P_2)_x \\ ((0,0,1) \times E)_y & (0,0,1) \times (E - P_1)_y & (0,0,1) \times (E - P_2)_y \\ ((0,0,1) \times E)_z & (0,0,1) \times (E - P_1)_z & (0,0,1) \times (E - P_2)_z \end{bmatrix} \tag{4.7}$$

Auf ähnliche Weise wie in 4.2 und 4.3 wird auch die Beziehung 4.5 invertiert und als Ausgangsbasis für die weiteren Berechnungen der Winkeländerungen an den Verbindungspunkten der Objektteile genutzt.

$$\dot{\Theta}(\dot{\theta_1}, ..., \dot{\theta_n}) = J[\Theta(\theta_1, ..., \theta_n)]^{-1}(S(\theta_1, ..., \theta_n) - S_g) \tag{4.8}$$

Um J^{-1} bestimmen zu können muss die Jacobi-Matrix quadratisch sein und darf keine Singularitäten aufweisen. Falls *J überbestimmt* ist, besitzt die Matrix mehr Reihen als Spalten und für den *unterbestimmten* Fall mehr Spalten als Reihen. In beiden Fällen kann die inverse Jacobi-Matrix nicht hergeleitet werden und wird stattdessen durch die Pseudoinverse J^+ approximiert. Daraus resultieren drei unterschiedliche Iterationsschemata, die sich jeweils durch die Inverse beziehungsweise Pseudoinverse in folgender Form unterscheiden:

- Regulär

$$\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}_{new} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} - J^{-1}(S(\theta_1, ..., \theta_n) - S_g) \tag{4.9}$$

- Überbestimmt

$$\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}_{new} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} - J^{+}(S(\theta_1, ..., \theta_n) - S_g) \tag{4.10}$$

  mit $J^+ = (J^T J)^{-1} J^T$

- Unterbestimmt

$$\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}_{new} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} - J^{+}(S(\theta_1, ..., \theta_n) - S_g) \tag{4.11}$$

  mit $J^+ = J^T (J J^T)^{-1}$

Zur Lösung der Gleichungen 4.10 und 4.11 wird beispielhaft im ersten Schritt in der Gleichung 4.10 ein Teil mit $\beta = (JJ^T)^{-1}(S(\theta_1, ..., \theta_n) - S_g)$ ersetzt und β durch Anwendung der *LU-Zerlegung* gelöst, bevor β wieder in die Gleichung 4.13 beziehungsweise 4.14 zurück substituiert wird.
Für das *LU Verfahren* wird $\beta = (JJ^T)^{-1}(S(\theta_1, ..., \theta_n) - S_g)$ als $r = A^{-1}b$ betrachtet und zur Lösung der Unbekannten r in die in der Gleichung 4.12 dargestellten Form $Ar = b$ umgestellt.

$$(JJ^T)\beta = (S(\theta_1, ..., \theta_n) - S_g) \tag{4.12}$$

Die einzige unbekannte Variable dieser Gleichung β wird mit Hilfe des *LU Verfahrens* bestimmt und in die zu lösende Gleichung 4.13 eingesetzt. Im Gegensatz dazu wird für den Fall, dass die Jacobi-Matrix *überbestimmt* ist und mehr Gleichungen als Unbekannte besitzt in [ESHD05] die *Singular Value Decomposition* zur Bestimmung von β angewandt.

Die Gleichungen 4.9 und 4.10 können demnach vereinfacht als

$$\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}_{new} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} - \beta J^T \tag{4.13}$$

und als

$$\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}_{new} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} - J^T \beta \tag{4.14}$$

geschrieben werden.

4.2 Schlussfolgerung

An diesem Punkt werden durch das Lösen der Gleichungen 4.13 und 4.14 in einem Iterationsschritt die Änderungen der Winkel zwischen den Elementen der kinematischen Verkettung neu berechnet. Die errechneten Änderungsraten der Winkel

$$\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}_{new} \tag{4.15}$$

werden mit den aktuellen Winkeln an den Verbindungspunkten verrechnet, um eine neue kinematische Pose der Verkettung zu bestimmen. Idealerweise hat der Endeffektor danach die durch den Benutzer vorgegebene Zielposition erreicht. Andernfalls muss die kinematische Bewegung mit den aktualisierten Werten der Verkettung neu berechnet werden. Durch die veränderten Positionen und Orientierungen der Gelenke und die neue Endeffektorposition muss die Jacobi-Matrix als Ausgangsbasis der kinematischen Berechnung neu bestimmt werden. Dies bedeutet, dass auch die inverse beziehungsweise die pseudoinverse Jacobi-Matrix nach jeder Veränderung der kinematischen Pose neu berechnet werden muss.
Für die iterative Annäherung eines Endeffektors an eine bestimmte Zielposition ist dieser Umstand nicht von Bedeutung, kann aber bei der interaktiven Echtzeitsteuerung zu Problemen führen. Der wesentliche Nachteil der inversen kinematischen Animation mit der Jacobi Methode liegt im hohen Rechenaufwand. In Abhängigkeit von der Anzahl der Verbindungen einer Verkettung *n* und der zur Verfügung stehenden Freiheitsgrade *m*, muss bei jeder Bewegungsänderung das *m x n* Gleichungssystem invertiert beziehungsweise die pseudoinverse Matrix bestimmt werden.
In diesem Zusammenhang müssen auftretende Singularitäten der Jacobi-Matrix näher betrachtet werden. In der Regel wird eine Matrix M als Singulär bezeichnet, wenn zwei oder mehr Reihen der Matrix linear voneinander abhängen oder aber wenn eine oder mehr Reihen der Matrix mit Nullen besetzt sind.
Wie der Abbildung 4.1 entnommen werden kann treten singuläre Erscheinungen in der Regel auf, wenn die kinematische Verkettung *voll ausgestreckt* ist beziehungsweise wenn die Elemente in einer Ebene liegen und sich die Zielposition des Endeffektor außerhalb des erreichbaren Arbeitsbereichs befindet. In solchen Fällen kann die Approximation mittels der Pseudoinversen zu unerwünschten Oszillationen führen, da der Endeffektor sich der Zielposition nicht weiter annähert und der Positionierungsfehler nicht minimiert werden kann.
Die Jacobi-Matrix ist nur für die aktuelle Konfiguration der Verkettung gültig und muss nach jeder Veränderung der kinematischen Pose neu berechnet werden, um die aktuelle Beziehung zwischen den Winkeländerungen an den Gelenken und der neuen Endeffektorposition und Orientierung auszudrücken. In der Konsequenz können zu große Änderungen der Lage und Orientierung des Endeffektors in einem Zeitschritt

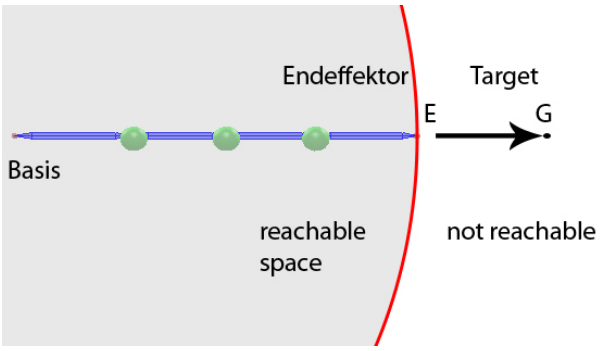

Abbildung 4.1: Problem der singulären Konfiguration einer Verkettung in Bezug auf die Jacobi Methode.

dazu führen, dass unerwünscht starke Winkeländerungen eine Annäherung des Endeffektors an die Zielposition verhindern. Tritt dieses Verhalten während der kinematischen Animation auf, muss der Endeffektor iterativ der Zielposition angenähert und dafür die J häufiger berechnet werden.
In der Regel gibt es für Positionierungsaufgaben des Endeffektors immer mehr als eine gültige kinematische Pose der Verkettung. Da im Vergleich zu der im nächsten Kapitel vorgestellten *Cyclic Coordinate Descent Methode* die Elemente nicht iterativ bewegt werden, um den Endeffektor der Zielposition anzunähern, sondern alle Winkeländerungen durch das Lösen der Matrixgleichung bestimmt werden, können leicht unerwünschte kinematische Posen der Figur entstehen.
Daher ist es notwendig, dass für die Echtzeitanimation von Figuren mittels der Jacobi Methode Beschränkungen eingeführt werden, um unerwünschte kinematische Bewegungsergebnisse zu vermeiden. Im Allgemeinen beziehen sich diese Beschränkungen, die im weiteren Verlauf der Arbeit als *Constraints* bezeichnet werden, auf die Winkelorientierung und die maximale Winkeländerung zwischen den Elementen einer Verkettung. Dadurch kann einerseits die Anzahl der möglichen Posen der Verkettung eingegrenzt und zum anderen verhindert werden, dass sich die Elemente der kinematischen Verkettung unerwünscht stark bewegen beziehungsweise Oszillationen auftreten.

Kapitel 5

Direkte kinematische Animationsmethode

In diesem Abschnitt wird die im Echtzeit-Animationssystem implementierte *Cyclic Coordinate Descent Methode*[WC91] zur kinematischen Bewegung von virtuellen Puppenmodellen vorgestellt. Im ersten Teil dieses Kapitels wird die Animationsmethode im Kontext ihrer geometrischen Beziehungen erläutert, bevor in den nachfolgenden Abschnitten auf die kinematische Bewegungsberechnung eingegangen wird. Darüber hinaus wird gezeigt wie sich die Bewegungsberechnung für aktive Effektoren und Endeffektoren unterscheidet und welche zusätzlichen Bedingungen für die Effektorbewegung erfüllt sein müssen.

Statt alle Winkeländerungen mit der inversen- oder pseudoinversen Jacobi-Matrix in einem Iterationsschritt zu lösen, wird bei der direkten Animationsmethode die Winkeländerung an jedem Gelenkpunkt einer Verkettung iterativ bestimmt. Die in diesem Teil vorgestellte *Cyclic Coordinate Descent Methode*[Ebe06] beruht auf dem heuristischen Ansatz, dass eine Verkettung kinematisch bewegt werden kann, indem in einem Zeitschritt die Winkeländerungen iterativ von der Spitze bis zur Basis der Verkettung bestimmt werden. In der Regel wird in jedem Iterationsschritt, unabhängig von den Rotationen der vorhergehenden oder folgenden Elementen der Verkettung, nur das für den aktuell iterierten Verbindungspunkt relevante Kettenglied rotiert.

5.1 Mathematische Einführung

Die Grundidee der *CCD Methode* basiert auf der Analyse einfacher geometrische Beschränkungen einer Verkettung. Für den Fall, dass ein Element fester Länge um seinen Basispunkt rotiert wird, kann durch die Rotation jeder Punkt, der $|L_1|$ Einheiten vom Rotationspunkt entfernt ist, erreicht werden. In der Abbildung 5.1 wird beispielhaft die sich ergebende kreisförmige Rotation für den 2D Fall dargestellt, bei welcher der Endeffektor auf der rot gezeichneten Kreisbahn mit dem Radius $|L_1|$ rotiert werden kann. Wird dieses Beispiel zu einer Verkettung von zwei Elementen mit der Länge L_1 und L_2 erweitert, definiert sich der durch den Endeffektor erreichbare Bereich in der in Abbildung 5.2 dargestellten Form.

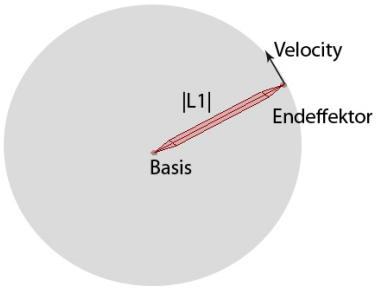

Abbildung 5.1: Rotation eines Elements um den Basispunkt.

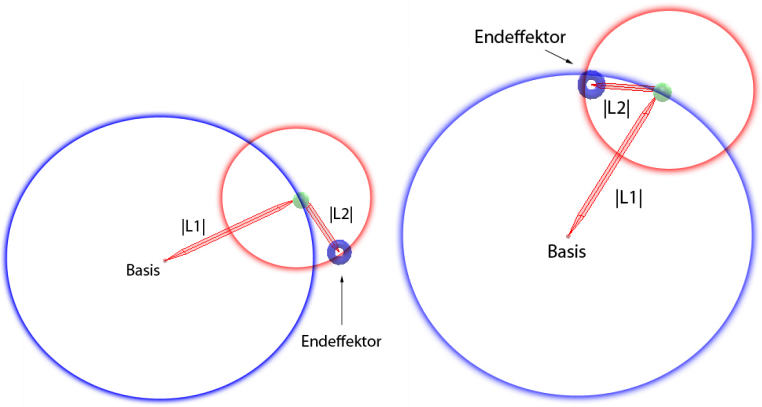

Abbildung 5.2: Der durch einen rotierenden Endeffektor erreichbare Bereich.

Unter der Annahme, dass der Basispunkt der Verkettung fixiert ist, kann im zweidimensionalen Fall jeder Punkt (X, Y) erreicht werden, der die nachfolgende Bedingung erfüllt.

$$|L_1 - L_2| \leq \sqrt{X^2 + Y^2} \leq |L_1 + L_2| \tag{5.1}$$

Die grafische Darstellung dieser Beziehung auf der linken Seite der Abbildung 5.3 zeigt den gültigen Einflussbereich der kinematischen Bewegung einer Verkettung mit zwei unterschiedlich langen beweglichen Teilen. Der grüne Kreis im Inneren der Grafik markiert den durch einen Endeffektor nicht erreichbaren Bereich, welcher durch die Beziehung $|L_1 - L_2|$ definiert ist. Die kreisförmige Verlaufsbahn des ersten Gelenks ist blau und die des Endeffektors rot dargestellt. Auf der rechten Seite der Abbildung 5.3 ist beispielhaft der erreichbare Bereich einer Verkettung mit zwei gleichlangen Elementen dargestellt. Betrachtet man die exemplarisch eingezeichneten roten Verlaufsbahnen des Endeffektors wird deutlich, dass in diesem Fall jeder beliebige Punkt erreicht werden kann, der folgende Bedingung erfüllt: $\sqrt{X^2 + Y^2} \leq |L_1 + L_2|$.

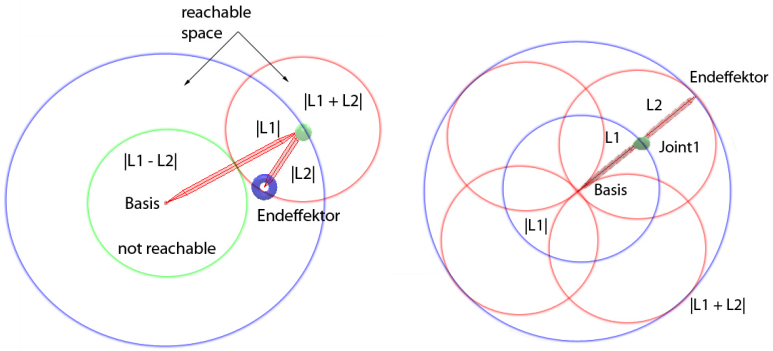

Abbildung 5.3: Der durch eine Kette mit zwei Elementen erreichbare Bereich.

Die in Gleichung 5.1 genannten Bedingungen gelten nur für den Fall, dass der Basispunkt der Verkettung im Koordinatenursprung liegt. Andernfalls sind die für die Rechnung relevanten Elemente aufgrund der Vektorbeziehung, wie in Abbildung 5.4 dargestellt, vom *Weltkoordinatenursprung* $(0, 0, 0)^T$ in das *Sub-Koordinatensystem* des Basispunkt $(0', 0', 0')^T$ zu transformieren.

Nachdem einige geometrische Vorbedingungen vorgestellt wurden, soll im folgenden Abschnitt die Funktionsweise des *CCD Verfahrens* näher vorgestellt werden. Um den Endeffektor, wie auf der rechten Seite der Abbildung 5.4 dargestellt, einer Zielposition *Target* anzunähern, wird über jeden *Link* der Verkettung iteriert und das entsprechende Element gegebenenfalls rotiert, um den Abstand *d* zwischen der aktuellen Position und der Zielposition zu minimieren.

$$d = |\vec{V}| = \sqrt{V_x^2 + V_y^2 + V_z^2} \tag{5.2}$$

Der Abstand *d* entspricht der Länge des Vektors $\vec{V}$, welcher die Stärke und Richtung der Bewegungsänderung *(Velocity)* des Endeffektors in einem Zeitschritt beschreibt.

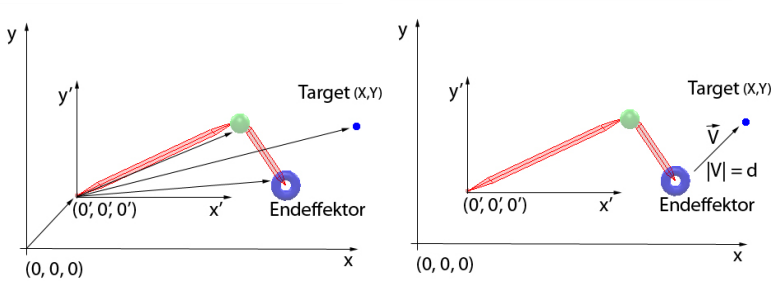

Abbildung 5.4: Weltkoordinatendarstellung und Objektkoordinatendarstellung.

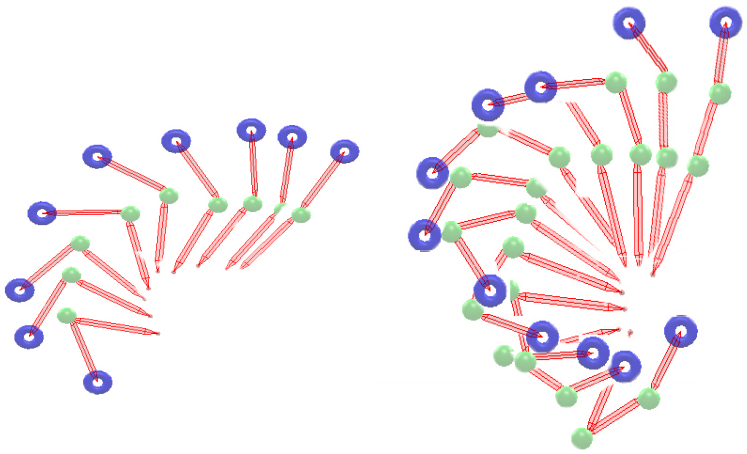

Abbildung 5.5: Bewegung des Endeffektors einer Kette mit zwei Elementen und einer Kette mit drei Elementen.

Dieser ergibt sich wie folgt aus der aktuellen und der Zielposition:

$$Velocity = \vec{V} = \begin{pmatrix} (G-E)_x \\ (G-E)_y \\ (G-E)_z \end{pmatrix} \tag{5.3}$$

Im Wesentlichen versucht man in jedem Iterationsschritt den globalen Fehler zu minimieren, indem für den aktuell iterierten Verbindungspunkt ein Winkel bestimmt wird, der den Fehler $e(q)$ minimiert.

$$e(q) = e_p(q) + e_o(q) \tag{5.4}$$

Der in der Gleichung 5.4 berechnete Fehlerwert gibt an, wie groß die Differenz in Bezug auf die Position und Orientierung des Endeffektors zur Zielposition ist. Dieser setzt sich aus dem Fehler in der Position e_p mit

$$e_p = ||\vec{G} - \vec{E}||^2 \tag{5.5}$$

und dem Fehler in der Orientierung e_o zusammen.

$$e_o(q) = \sum_{i=1}^{3} (\vec{u}_{jG} \cdot \vec{u}_{jE}) - 1)^2 \tag{5.6}$$

Der Orientierungsfehler $e_o(q)$ in Gleichung 5.6 wird aus der Summe der Skalarprodukte der orthonormalen Reihen der aktuellen Orientierungsmatrix und der Zielorientierung gebildet.

$$O_E = \begin{bmatrix} u_{1E} \\ u_{2E} \\ u_{3E} \end{bmatrix}; O_G = \begin{bmatrix} u_{1G} \\ u_{2G} \\ u_{3G} \end{bmatrix} \tag{5.7}$$

Die orthonormalen Reihen der Matrix können als Achsenvektoren $\vec{u}_1$, $\vec{u}_2$ und $\vec{u}_3$ angesehen werden. Diese geben die Orientierung eines Objektes im lokalen Koordinatensystem mit Bezug auf das Weltkoordinatensystem der Szene an. Danach stellen die lokalen Achsenvektoren eines nicht rotierten Objektes die Achsenwerte des Weltkoordinatensystems wie folgt dar:

$$\vec{u}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \vec{u}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \vec{u}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.8}$$

In der Darstellung 5.9 werden die lokalen Vektoren zur Beschreibung der Objektorientierung nach einer Rotation von $45\ Grad$ um die Z-Achse gezeigt.

$$\vec{u}_1 = \begin{pmatrix} 0.5 \\ 0.5 \\ 0 \end{pmatrix}, \vec{u}_2 = \begin{pmatrix} -0.5 \\ 0.5 \\ 0 \end{pmatrix}, \vec{u}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.9}$$

Die lokalen Achsenvektoren werden nicht nur zur Lagebeschreibung von Objekten in der Szene, sondern auch zur Berechnung der Winkel an den Gelenkpunkten der kinematischen Verkettung benötigt. Wie schon erläutert, ergibt sich die kinematische Animation dadurch, dass über alle Elemente der Verkettung iteriert und das jeweils aktuelle Element um seinen Ankerpunkt, welcher der im aktuellen Iterationsschritt iterierte Verbindungspunkt ist, rotiert wird. In der Abbildung 5.6 wird ein Iterationsschritt der

CCD Methode beispielhaft dargestellt. Intern kann die Rotation eines Elementes um einen bestimmten Faktor θ_i als Rotation des in der Abbildung dargestellten Vektors $\vec{P}_{ie}$, welcher vom aktuellen Gelenkpunkt zur Endeffektorposition zeigt, repräsentiert werden.
Mit Bezug auf die oben vorgestellten geometrischen Beschränkungen wird auf der linken Seite der Abbildung 5.6 deutlich, wann der Positionierungsfehler e_p am gegenwärtigen Gelenk minimiert wird. Das Minimum des Fehlers e_p ist genau dann erreicht, wenn der Vektor $\vec{P}_{ie}$ sich mit dem Vektor $\vec{P}_{ig}$, welcher vom aktuellen Gelenk zur Zielposition *(Target)* zeigt, überdeckt.
Anders formuliert ist der Fehler e_p genau dann am Gelenk *i* minimal, wenn der Vektor $\vec{P}_{ie}$ wie in der Abbildung 5.6 dargestellt, entlang der rot gezeichneten Kreisbahn mit dem Radius $r_i = |L_i|$ zu einem Punkt *S* rotiert wurde. Dabei ist *S* der auf der rechten Seite der Abbildung 5.6 rot gezeichnete Punkt auf der Kreisbahn von *Link i*, welcher den geringsten Abstand zur Zielposition des Endeffektors hat.

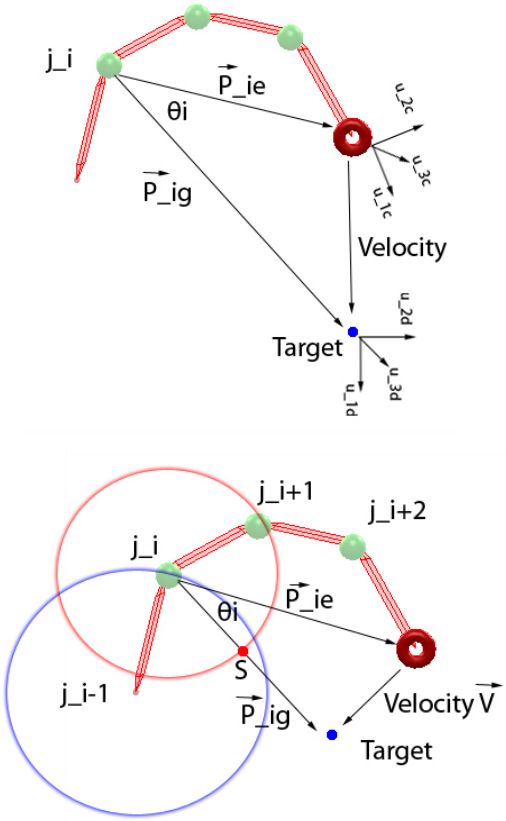

Abbildung 5.6: Beispiel eines CCD Iterationsschrittes für einen Verbindungspunkt.

5.1.1 Bewegung von Endeffektoren

Im Anschluss an den oben vorgestellten Ansatz stellt sich die Frage, wie der optimale Wert für θ_i bestimmt werden kann, um Vektor $\vec{P}_{ie}$ in die richtige Position $\vec{P'}_{ie}$ zu rotieren. In Anbetracht der unterschiedlichen Orientierung des Endeffektors an der Aktuellen und an der Zielposition, sollte θ_i ebenfalls den Orientierungsunterschied ausgleichen. Der Fehler in der Positionierung wird minimiert, indem die Gleichung 5.10 für ein geeignetes θ_i maximiert wird.
Das in der Gleichung 5.10 gebildete Skalarprodukt zwischen $\vec{P}_{ig}$ und $\vec{P'}_{ie}(\theta_i)$ wird genau dann maximal, wenn sich beide Vektoren geometrisch überlagern. Demzufolge rotiert ein geeignetes θ_i den Vektor $\vec{P}_{ie}$ genau dann zu dem in Abbildung 5.6 rot dargestellten Punkt *S*, wenn die Gleichung 5.10 maximal und der Abstand zwischen dem Endeffektor und der Zielposition minimal wird.

$$g_1(\theta_i) = \vec{P}_{ig} \cdot \vec{P'}_{ie}(\theta_i) \tag{5.10}$$

Darüber hinaus maximiert ein geeignetes θ_i ebenfalls nachfolgende Gleichung, um den Orientierungsunterschied zwischen der aktuellen und der Zielorientierung zu minimieren.

$$g_2(\theta_i) = \sum_{j=1}^{3} \vec{u}_{jg} \cdot \vec{u'}_{je}(\theta_i) \tag{5.11}$$

Bei genauerer Betrachtung der Gleichung 5.11 wird deutlich, dass der Fehler in der Orientierung e_o minimiert wird, wenn die Summe der *Skalarprodukte* jedes Paares orthonormaler Achsenvektoren $\vec{u}_{1g}, \vec{u'}_{1e}(\theta_i)$; $\vec{u}_{2g}, \vec{u'}_{2e}(\theta_i)$ und $\vec{u}_{3g}, \vec{u'}_{3e}(\theta_i)$ maximal wird.
Der Grund dass an dieser Stelle die $Skalarprodukte$ genutzt werden ist eindeutig, da diese für orthogonale Vektoren minimal und demzufolge maximal werden, wenn sich zum Beispiel Vektor $\vec{P'}_{ie}$ mit Vektor $\vec{P}_{ig}$ überdeckt. Aus den mathematischen Bestimmungen resultiert die *objektive Funktion* $g(\theta_i)$, welche für die optimale Winkeländerung θ_i am Verbindungspunkt *i* maximal wird.

$$g(\theta_i) = \epsilon_p g_1(\theta_i) + \epsilon_o g_2(\theta_i) \tag{5.12}$$

Wie in [Wel93] vorgeschlagen, können die Gewichte ϵ_p und ϵ_o durch folgende Gleichungen mit $\alpha = 0.5$ und

$$\xi = \frac{\min(||\vec{P}_{ig}||, ||\vec{P}_{ie}||)}{\max(||\vec{P}_{ig}||, ||\vec{P}_{ie}||)} \tag{5.13}$$

bestimmt werden. Das auf die Positionierung bezogene Gewicht ϵ_p ist demnach definiert als:

$$\epsilon_p = \alpha(1 + \xi) \tag{5.14}$$

Die auf die Orientierung bezogene Gewichtung wird als $\epsilon_o = 1$ angenommen. Darauf aufbauend wird in [Wel93] die Gleichung 5.12 im nächsten Schritt zu

$$g(\theta_i) = k_1(1 - \cos(\theta_i)) + k_2 \cos(\theta_i) + k_3 \sin(\theta_i) \tag{5.15}$$

umgeformt und die erste Ableitung gebildet.

$$0 = (k_1 - k_2) \sin(\theta_i) + k_3 \cos(\theta_i) \tag{5.16}$$

Danach kann die Gleichung zur Bestimmung eines ersten Kandidaten für die Änderungsrate des Winkels θ_{ic} umgestellt werden.

$$\theta_{ic} = \tan^{-1} \frac{k_3}{(k_2 - k_1)} \tag{5.17}$$

Die Koeffizienten k_1, k_2 und k_3 sind definiert als:

$$k_1 = \epsilon_p(\vec{P}_{ig} \cdot \vec{R}_{axis_i})(\vec{P}_{ie} \cdot \vec{R}_{axis_i}) + \epsilon_o(\vec{u}_{ig} \cdot \vec{R}_{axis_i})(\vec{u}_{ie} \cdot \vec{R}_{axis_i}) \tag{5.18}$$

$$k_2 = \epsilon_p(\vec{P}_{ig} \cdot \vec{P}_{ie}) + \epsilon_o(\vec{u}_{ig} \cdot \vec{u}_{ie}) \tag{5.19}$$

$$k_3 = \vec{R}_{axis_i} \cdot [\epsilon_p(\vec{P}_{ie} \times \vec{P}_{ig}) + \epsilon_o(\vec{u}_{ie} \times \vec{u}_{ig})] \tag{5.20}$$

Es wurde bereits festgestellt, dass die Gleichung 5.17 dazu genutzt werden kann, um einen ersten Kandidaten θ_{ic} zu bestimmen. Aufgrund der trigonometrischen Eigenschaften der *tan-Funktion* ergeben sich zwei weitere potenzielle Kandidaten mit $\theta_{ic} + \frac{\pi}{2}$ und $\theta_{ic} - \frac{\pi}{2}$, welche die *objektive Gleichung* in 5.15 maximal werden lassen können. Dafür muss die *objektive Funktion* in jedem Iterationsschritt *i* dreimal gelöst werden, um den geeigneten Kandidaten für die Winkeländerung am Verbindungspunkt *i* zu bestimmen.
In einem letzten Schritt muss dieser zur Bestimmung des neuen Winkels mit dem aktuellen Winkel am Gelenk *i* addiert und die Pose der Verkettung aktualisiert werden, bevor zum nächsten Verbindungspunkt *i-1* iteriert wird.

$$q_{i_{new}} = q_{i_{old}} + \epsilon_i \theta_i \tag{5.21}$$

Die Aktualisierung des Winkels am gegenwärtig iterierten Element *i* erfolgt durch die Addition der gewichteten Winkeländerung $\epsilon_i \theta_i$. Diese ist geeignet, um den Fehler *error e(q)* am Iterationsschritt *i* zu minimieren. Daraus resultiert die sofort erkennbare Annäherung des Endeffektors an die Zielposition, die mit der kinematischen Bewegung der Verkettung, die nach jedem Iterationsschritt aktualisiert wird, einhergeht.
In der Regel reicht ein Iterationsvorgang von der Spitze der Verkettung bis zur Basis aus, um den Endeffektor so nah wie möglich an die gewünschte Zielposition zu legen. Falls der Fehler *e(q)* dennoch einen bestimmten Schwellwert überschreitet, muss der gesamte Iterationsprozess wiederholt werden, bis *e(q)* klein genug ist. Häufig tritt dieser Fall ein, wenn die in einem Zeitschritt durch die Endeffektorbewegung zu überwindende Distanz zwischen der aktuellen und der Zielposition des Endeffektors zu groß ist. In der Praxis hat sich herausgestellt, dass die Notwendigkeit von mehrfachen Iterationsdurchgängen vermieden werden kann, wenn die Endeffektor Bewegung direkt an die Maussteuerung gekoppelt wird, um den Fehler *e(q)* klein zu halten. Dies trifft allerdings nur zu, solange sich die Zielposition innerhalb des durch den Endeffektor erreichbaren Bereichs befindet.
Andernfalls kann der Endeffektor nur bis zu einem gewissen Wert an die Zielposition angenähert werden. Das bedeutet, dass ab einem bestimmten Punkt der Fehler *e(q)* nicht weiter minimiert werden kann. Für den Fall das der Zielpunkt außerhalb der Reichweite liegt, muss ein geeignetes Abbruchkriterium definiert werden, um Oszillationen der Verkettung zu vermeiden.

5.1.2 Bewegung von Effektoren

Eine interessante Erweiterung der Animation von bewegten Figuren ergibt sich hinsichtlich der Frage wie sich kinematisch animierte Verkettungen verhalten, wenn der Effektor nicht nur einer der beiden Endpunkte der Verkettung, sondern jeder beliebige Verbindungspunkt zwischen den Elementen sein kann. Ist das aktiv bewegte Element weder der Basispunkt noch die Spitze der Kette, müssen zusätzliche Beschränkungen eingeführt werden, um zu verhindern, dass die Verkettung während der kinematischen Bewegung zerfällt.
Darüber hinaus ändert sich der Iterationsprozess über die Elemente der kinematisch animierten Verkettung im Vergleich zur Bewegung der Endeffektoren. Die in Abschnitt 5.1 vorgestellten geometrischen Beschränkungen müssen für den speziellen Fall der Bewegung von zwischen den Endpunkten liegenden Verbindungen erweitert werden, um zu verhindern, dass diese sich aus dem durch die Pose der Verkettung gegebenen gültigen Bereich herausbewegen. Das heißt, dass jeder Punkt *P(X,Y,Z)* den der aktuell bewegte Effektor erreichen kann, in der Domäne der linken und rechten Seite der Verkettung liegen muss, die an der aktuellen Effektorposition *i* geteilt ist. Ob ein Ziel-

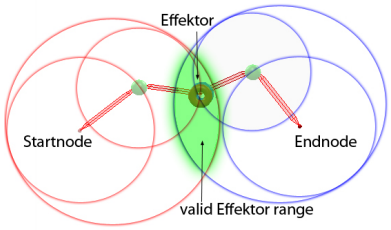

Abbildung 5.7: Übersicht über die geometrische Bestimmung des Bewegungsbereichs.

punkt *G* die oben gestellten Bedingungen hinreichend erfüllt, kann durch den Vergleich der in Abbildung 5.8 dargestellten Vektorlängen mit den zugeordneten Verkettungsabschnitten festgestellt werden. Dies bedeutet, dass die neue Effektorposition für beide Seiten der Verkettung erreichbar sein muss, was mathematisch durch die Gleichungen 5.22 und 5.23 verdeutlicht wird. Dafür wird, wie in den Gleichungen dargestellt, die Länge des Vektors vom jeweiligen Endpunkt der Verkettung zur Zielposition mit den aufsummierten Längen $|L_j|$ der einzelnen Elemente *j* der jeweiligen Seite verglichen. Um festzustellen ob ein vorgegebener Zielpunkt durch den aktiven Effektor erreicht werden kann, werden die zwei Vektoren $\vec{P}_{start_g}$ und $\vec{P}_{end_g}$ vom jeweiligen Endpunkt der Verkettung zum Zielpunkt gebildet. Anschließend werden in einem ersten Schritt die Längen der einzelnen Kettenglieder $|L_j|$ bis zur aktiven Effektorposition *i* aufsummiert, um die Gültigkeit der Bedingung 5.22 zu testen. Im nächsten Schritt wird die Gleichung 5.23 ausgewertet, indem die Längen der Kettenglieder beginnend beim aktiven Effektor bis zum Ende der Verkettung aufsummiert werden, um diese mit der

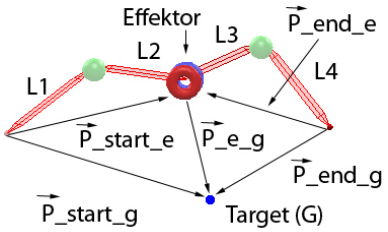

Abbildung 5.8: Validieren des Zielpunktes für die Effektorbewegung.

Länge des Vektors $|\vec{P}_{end_g}|$ zu vergleichen. Ist die Länge der beiden Vektoren $|\vec{P}_{start_g}|$ und $|\vec{P}_{end_g}|$ jeweils keiner oder gleich der Länge beider Seiten der Verkettung, sind die Bedingungen 5.22 und 5.23 erfüllt und die Zielposition befindet sich innerhalb des gültigen Bereichs.

$$|\vec{P}_{start_g}| \leq \sum_{j=1}^{i} |L_j| \tag{5.22}$$

$$|\vec{P}_{end_g}| \leq \sum_{j=i}^{n} |L_j| \tag{5.23}$$

Anders formuliert ergibt sich der gültige Bereich für eine Effektorposition als genau die Schnittmenge der Punkte, die von jeder der beiden voll ausgestreckten Seiten einer Verkettung erreicht werden kann.

$$valid_{region} = lefthand_{region} \cap righthand_{region} \tag{5.24}$$

Grafisch kann dieser Bereich für den 2D Fall als kreisförmige Region mit dem Radius $\sum |L|$ und dem Zentrum im Basispunkt der jeweiligen Seite beschrieben werden, welche durch den rotierenden Effektor eingegrenzt wird. Daraus folgt dass der gültige Bereich dadurch bestimmt werden kann, dass dieses Verfahren für die linke und rechte Seite der Verkettung angewendet wird.

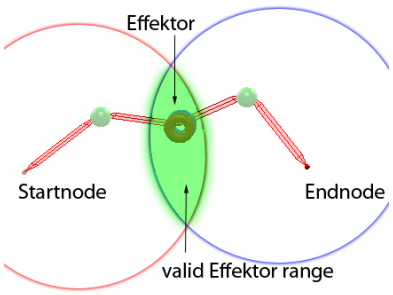

Abbildung 5.9: Ausschnitt des gültigen Bewegungsbereichs eines Effektors.

Im Gegensatz zur Bewegung von Endeffektoren ist es für die aktive Bewegung von Verbindungspunkten entscheidend, ob die Zielposition vom Effektor erreicht werden kann

und ob die Zielposition innerhalb des für den Effektor als gültig bestimmten Bereichs liegt. Diese Unterscheidung ergibt sich dadurch, dass der Endeffektor in der Regel so nah wie möglich beziehungsweise bis die Verkettung voll ausgestreckt ist, der Zielposition angenähert werden kann.
In Bezug auf die Bewegung von Verbindungspunkten ist dieser Ansatz nicht geeignet, da zu jedem Zeitpunkt die Gültigkeit der Verkettung gewährleistet werden muss. Zur Erfüllung der vorhergehend vorgestellten Beschränkungen ist in Bezug auf die Effektorbewegung ein anderer Lösungsansatz zu nutzen. Im Wesentlichen beruht dieser auf der Idee einen Punkt *G'* entlang des Vektors $\vec{P}_{eg}$ zu finden, der die Gleichungen 5.21 und 5.22 für den Vektor $\vec{P}_{start_{g'}}$ und $\vec{P}_{end_{g'}}$ erfüllt.
Ist ein gültiger Punkt *G'* bestimmt, so wird dieser in der Folge als substituierter Punkt des nicht erreichbaren Punktes *G* angesehen.

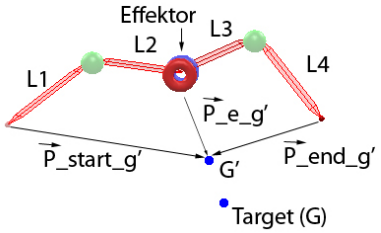

Abbildung 5.10: Approximation des nicht erreichbaren Zielpunktes.

In der Abbildung 5.10 wird die Bestimmung eines gültigen Zielpunktes *G'* für eine Verkettung mit vier Elementen und dem dritten Gelenkpunkt als Effektor dargestellt. Ein möglicher Zielpunkt *G'* dieser Konfiguration wurde bestimmt, falls die Vektoren $\vec{P}_{start_{g'}}$ und $\vec{P}_{end_{g'}}$ die nachfolgenden Bedingungen erfüllen:

$$|\vec{P}_{start_{g'}}| \leq |L_1 + L_2| \tag{5.25}$$

$$|\vec{P}_{end_{g'}}| \leq |L_3 + L_4| \tag{5.26}$$

Programmintern wird dieser Vorgang umgesetzt indem entlang des Vektors $\vec{P}_{eg}$ iteriert wird um den Punkt G' zu bestimmen. Dieser hat den geringsten Abstand zur vorgegebenen Zielposition *G* und liegt innerhalb des gültigen Bereichs. Eine genauere Beschreibung dieser Lösung erfolgt in Abschnitt 5.2.

Je nachdem an welcher Position sich der Effektor innerhalb der Verkettung befindet, ist es unter Umständen erwünscht, dass sich die Verkettung im Verlauf der kinematischen Animation unterschiedlich verhält. Dies kann zum Beispiel der Fall sein, wenn Gelenkpunkte, die in der Nähe der Endpunkte einer Verkettung liegen, sich bei der Bewegung wie Endeffektoren verhalten sollen. Die Darstellung 5.12 zeigt beispielhaft die Bewegung der Verbindungspunkte an den Positionen *Startposition+1* und *Endposition-1*, die wie typische Endeffektoren bewegt werden. In der Konsequenz bedeutet dies, dass die eigentlichen Endeffektoren der Verkettung die relative Position zum ausgewählten und rotierenden Effektor nicht verändern.
Vergleichend zu der Abbildung 5.12 wird in der Darstellung 5.13 die kinematische Bewegung einer Verkettung für den Fall gezeigt, dass sich der aktive Effektor an der Position *Startposition+1* befindet und die Endpunkte der Verkettung fixiert sind. Auf

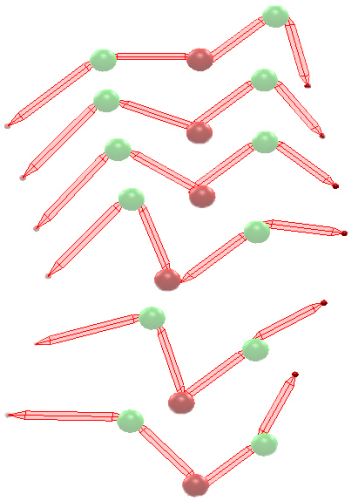

Abbildung 5.11: Bewegung einer Verkettung mit vier Elementen und dem dritten Gelenkpunkt als Effektor.

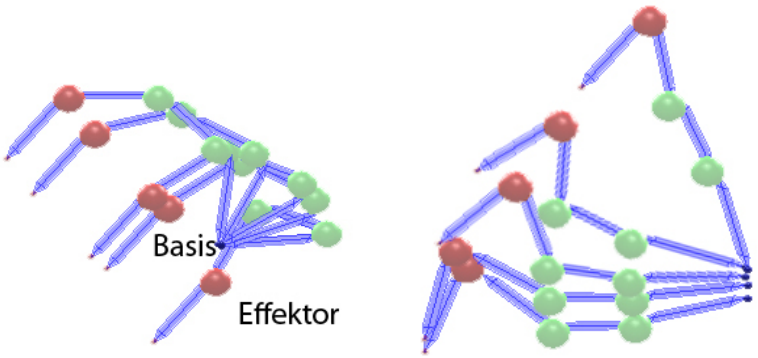

Abbildung 5.12: Kinematische Animation von Effektoren mit Endeffektoreigenschaften.

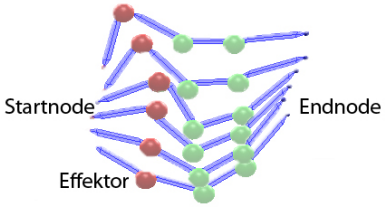

Abbildung 5.13: Kinematische Animation von Effektoren ohne Endeffektoreigenschaften.

welche Art der Effektor letztendlich bewegt wird, kann vom Nutzer über die globalen Beschränkungen der Verkettung festgelegt werden, welche es in diesem Zusammenhang erlauben, die Endpunkte einer Verkettung als fixiert oder beweglich zu markieren.

5.2 Implementation der CCD Methode

Um die im vorhergehenden Abschnitt vorgestellten mathematischen Grundlagen der *CCD Methode* in ihrer programminternen Umsetzung näher zu betrachten, sollen im Folgenden ausgewählte Aspekte der Animationsmethode in Form von Pseudocode und Strukturdiagrammen vorgestellt werden. In Kapitel 3 wurde bereits einleitend erwähnt, dass die kinematischen Strukturen aus einer Folge von Elementen bestehen, die jeweils durch Knoten miteinander verbunden sind. Jeder dieser Verbindungspunkte kann entweder ein aktiver Effektor oder ein regulärer Verbindungspunkt sein. Diese werden in der Regel als Rotationsgelenk dargestellt. Je nachdem welche Art der Ersatzstrukturen gewählt wurde, werden die verbundenen Elemente in Form von Knochen oder Boxen visualisiert. Für die kinematische Bewegung einer solchen Struktur wird im Wesentlichen der in Abbildung 5.14 dargestellte Iterationsprozess umgesetzt. Bei diesem wird, wie in der Darstellung 5.14 gezeigt, von der Spitze der Verkettung bis zur Basis über alle Gelenkpunkte iteriert und das entsprechende Element rotiert, um den Endeffektor der Zielposition anzunähern.

Dafür muss in jedem Iterationsschritt der Vektor $\vec{P}_{i_d}$ vom iterierten Gelenkpunkt *i* zur Zielposition und der Vektor $\vec{P}_{i_c}$ vom iterierten Gelenkpunkt *i* zur aktuellen Endeffektorposition bestimmt werden. In diesem Zusammenhang wird im nächsten Schritt der Darstellung 5.14 die Orientierung der beiden Vektoren bestimmt, um die Koeffizienten *k1, k2 und k3* zu berechnen. Anschließend wird mit Hilfe der *objektiven Funktion* der geeignete Kandidat q_c für die Winkeländerung am aktuellen Gelenkpunkt bestimmt, bevor zum nächsten Verbindungspunkt der Verkettung iteriert wird.
Um die Verkettung von der entgegengesetzten Seite her zu animieren, muss der Iterationsprozess umgekehrt werden. In diesem Fall ist der aktive Endeffektor der linke Basispunkt der Verkettung und die frühere Spitze der Kette der neue Basispunkt.

Falls nach einer kompletten Iteration über die Verkettung der Fehler *e(q)* einen bestimmten Schwellwert überschreitet und sich der Effektor nicht nah genug am Zielpunkt befindet, kann es unter Umständen hilfreich sein den Iterationsvorgang zu wiederholen, um den Effektor dem Ziel weiter anzunähern. In der Abbildung 5.15 wird dargestellt, wie der generelle Iterationsprozess in einen Fehlertest eingebaut werden kann. Dafür wird zu Beginn des Iterationsprozess der Fehlerwert *e* zwischen der aktuellen und der Zielposition des Endeffektors bestimmt. Nachdem der Iterationsprozess abgeschlossen ist, wird wie in Darstellung 5.15 gezeigt, erneut der Fehler zwischen der Position des Endeffektors und dessen Zielposition bestimmt, um zu testen ob sich der Endeffektor nah genug am Zielpunkt befindet. Falls der berechnete Fehler zu groß ist, wird der Iterationsprozess solange wiederholt bis *e* minimal wird beziehungsweise nicht weiter minimiert werden kann.

Zusätzliche Beschränkungen müssen gelten, wenn innere Elemente der Verkettung als aktive Effektoren bewegt werden. Diese müssen von der Zielposition erfüllt werden, um die Stabilität der Verkettung zu erhalten. Die interne Umsetzung der in Abschnitt $5.1.2$ vorgestellten Beschränkungen wird in der Abbildung 5.16 dargestellt. Es wird

```
1
2   // assuming endeffector ist located at connections tip
3   SET iterator currentLink = connection.end()
4
5   WHILE( currentLink != connection.begin() )
6
7     // calculate vector pointing from current link
8     // to goal position
9     DETERMINE P_id
10    // calculate vector pointing from current link
11    // to current endeffektor position
12    DETERMINE P_ic
13
14    // calculate orientation of P_id
15    DETERMINE u_id
16    // calculate orientation of P_ic
17    DETERMINE u_ic
18
19    DETERMINE configuration weight
20    DETERMINE positional weight
21
22    COMPUTE objective funktion
23    CALCULATE coefficients k1, k2, k3
24
25    // use first derivative of objective function and
26    // calculate coefficients to calculate candidate q_c
27    DETERMINE first candidate value
28
29    //solve objective function 3 times by applying canididates
30    DETERMINE q_c
31    DETERMINE q_c - Pi
32    DETERMINE q_c + Pi
33
34    DETERMINE candidate value maximizing objective function
35
36    SET candidate as weightend angular velocity of currentLink
37
38    // update connections pose and continue with next link-1
39    currentLink--
```

Abbildung 5.14: Pseudocode: Kinematische Endeffektorbewegung mit der CCD Methode.

```
1
2   // initial offset between endeffektor and goal position
3   DETERMINE initial error e
4
5   // CCD iteration process
6   COMPUTE CCD algorithm
7
8   OBTAIN new pose of linkage
9
10  DETERMINE new error e_i
11
12  WHILE( e_i > error_threshold && !exit)
13    // CCD iteration process
14    COMPUTE CCD algorithm
15
16    OBTAIN new pose of linkage
17
18    DETERMINE updated error e_c
19
20    IF( e_c < e_i )
21            // endeffector approaches towards desired position
22      // update current error value
23      SET e_i = e_c
24    ELSE
25      // error e_i is at minimum
26      SET exit = TRUE
```

Abbildung 5.15: Pseudocode: Fehlertest zwischen erreichter Endeffektorposition und der Zielposition.

gezeigt welche Schritte notwendig sind, um die mögliche Zielposition zu validieren, bevor der in Abbildung 5.17 dargestellte Iterationsprozess beginnen kann. Im ersten Teil der Abbildung 5.16 werden die Längen der Elemente aufsummiert, um die maximale Reichweite der linken und der rechten Seite der Verkettung zu bestimmen. Im nächsten Schritt werden die beiden Vektoren $\vec{P}_{l_d}$ und $\vec{P}_{r_d}$ erstellt, welche vom Start- beziehungsweise Endpunkt der Verkettung zur Zielposition zeigen.
Daran schließt sich der in Abbildung 5.16 dargestellte Test an. Dieser zeigt ob die Zielposition vom aktiven Effektor erreicht werden kann, ohne dass die Verkettung auseinander bricht. Dafür werden die Längen der beiden Vektoren $\vec{P}_{l_d}$ und $\vec{P}_{r_d}$ mit den zuvor berechneten Längen *SUM lefthand* und *SUM righthand* der Verkettung verglichen. Für den Fall das der Zielpunkt nicht von beiden Seiten der Verkettung erreicht werden kann, muss ein Ersatzpunkt entlang des Vektors $\vec{P}_{e_d}$ gefunden werden. Dieser neue Zielpunkt *d'* liegt genau auf dem Vektor von der aktuellen Effektorposition *e* zur Zielposition *d* und ist durch beide Seiten der Verkettung erreichbar. Danach wird, wie in Abbildung 5.16 dargestellt, der eigentliche *CCD Iterationsprozess* gestartet.
In Abbildung 5.17 wird der sich von der Bewegung der Endeffektoren unterscheidende Iterationsprozess zur Bewegung von inneren Verbindungspunkten vorgestellt. Die *CCD Iteration* wird gestartet, nachdem wie in Darstellung 5.16 erläutert, die Zielposition validiert beziehungsweise eine Ersatzposition bestimmt wurde. Der wesentliche Unterschied besteht darin, das die *CCD Iteration* vom Effektor aus entlang beider Hälften der Verkettung läuft, um diese kinematisch zu animieren.
Wurden die mathematischen Beschränkungen eingehalten, bricht die Verkettung nach diesem Prozess nicht auseinander. Andernfalls muss die entstandene Differenz durch lineare Interpolation zwischen den Effektorpositionen E'_l und E'_r minimiert werden. Der interne Ablauf wird in Abbildung 5.17 vorgestellt und zeigt welche Schritte notwendig sind, nachdem die linke und rechte Seite der Verkettung kinematisch bewegt wurden. Im Anschluss an den Iterationsprozess sind die neuen Effektorposition der linken

```
1
2   // lenghts of Elements going from startnode to effektor i
3   DETERMINE SUM_lefthand = L_0 + L_1 + .. + L_i
4
5   // lenghts of Elements going from endnode to effektor i
6   DETERMINE SUM_righthand = L_n + L_n-1 + .. + L_i
7
8   // calculate vector pointing from startnode to
9   // goal position
10  DETERMINE P_ld
11
12  // calculate vector pointing from endnode to
13  // goal position
14  DETERMINE P_rd
15
16  DO rangetest
17  // compare SUM_lefthand and length of P_ld
18  // compare SUM_righthand and length of P_rd
19
20  SET valid = result from rangetest
21  // valid IF | P_ld | <= SUM_lefthand
22  // AND IF | P_rd | <= SUM_righthand
23
24  IF( !valid )
25
26    // calculate vector pointing from current effector position
27    // to target position
28    DETERMINE P_id
29
30    // iterate along P_id to determine valid position d'
31    // as substitute for d
32
33    WHILE( iterating )
34      // compare length of temporary vector P_ld'
35      // compare length of temporary vector P_rd'
36      IF (|P_ld'|<=SUM_lefthand && |P_rd'|<=SUM_righthand)
37        // continue iteration process
38        SET d_valid = d'
39      ELSE
40        DO exit iteration process and start animation with d_valid
41  ELSE DO start animation
```

Abbildung 5.16: Pseudocode: Verifizierung der Zielposition von inneren Effektoren.

und rechten Seite miteinander zu vergleichen. Um sicherzustellen, dass die Verkettung nicht auseinander reißt, dürfen sich die beiden Positionen nicht wesentlich unterscheiden. Ansonsten muss eine Ersatzposition *E lr'* bestimmt werden, welche zur neuen Effektorposition der linken und rechten Seite der Verkettung wird. Abschließend wird in einem letzten Schritt, wie die Darstellung 5.17 zeigt, die Pose der Verkettung erneut aktualisiert.

```
1
2   // iterate from startnode to effector
3   DO CCD iteration
4   // to determine new pose for part of connection
5
6   // iterate from endnode to effector
7   DO CCD iteration
8
9   DETERMINE new pose for part of connection
10
11  COMPARE effector positions E_l and E_r
12  SET valid = perform rangetest()
13
14  if( !valid )
15    // interpolate between E_l and E_r
16    COMPUTE substitute position between E_l and E_r
17    DETERMINE new effector positon E_lr'
18
19    OBTAIN new pose of linkage
```

Abbildung 5.17: Pseudocode: Kinematische Effektor bezogene Animation mit der CCD Methode.

5.3 Schlussfolgerung

Einer der größten Nachteile der *CCD Methode* entsteht durch die unabhängig voneinander bewegten Elemente. Da das Ziel der Methode die Minimierung des Abstandes zwischen Endeffektor und Zielposition ist, kann es unter Umständen passieren, dass nicht alle Elemente kinematisch bewegt werden. Dieser Fall tritt ein wenn das Minimum bereits erreicht ist, bevor über alle Elemente rotiert wurde. Auf der linken Seite der Abbildung 5.18 ist dieser Fall beispielhaft dargestellt. Dieser ist zu entnehmen, dass sich der *Link l3* im Verlauf der kinematischen Animation am stärksten bewegt, während die Winkeländerung der Verbindungspunkte *l1* und *l2* minimal ist. Daraus können kinematische Posen resultieren, die unnatürlich wirken und nicht der gewünschten Bewegung entsprechen.
Begründen lässt sich dies mit der Art und Weise, wie mittels des *CCD Algorithmus* die Änderungsraten der Gelenkwinkel bestimmt werden. Ohne die zusätzliche Gewichtung der Gelenkpunkte werden bei jeder kinematischen Bewegung die Elemente, welche sich nah am aktuellen Effektor befinden stärker rotiert als solche, die näher an der Basis lokalisiert sind. Darüber hinaus haben Rotationen an vorhergehenden oder nachfolgenden Gelenkpunkten auf die aktuelle Bewegung eines Elementes *i* am Gelenkpunkt *i* keinen Einfluß.
Dieser Nachteil kann aber in gewisser Weise ausgeglichen werden, indem die Änderungsraten der Gelenkwinkel in Abhängigkeit von ihrer aktuellen Position unterschiedlich stark gewichtet werden. Dadurch können Gelenkpunkte in der Nähe von Endpunkten mit höherer Festigkeit als innere Gelenkpunkte simuliert werden, um alle Verbindungspunkte, wie auf der rechten Seite der Abbildung 5.18 dargestellt, optimal zu ro-

tieren. Darüber hinaus ist es durch die diskrete Iteration über alle Verbindungspunkte möglich, an jedem Gelenkpunkt die Erfüllung der festgelegten Beschränkungen zu testen, bevor der nachfolgende Verbindungspunkt bewegt wird. Zudem erfolgt nach der Winkeländerung an einem Verbindungspunkt die daraus resultierende Änderung der kinematische Pose einer Verkettung, bevor mit dem nächsten Iterationsschritt fortgefahren wird.
In diesem Zusammenhang ergibt sich der Vorteil gegenüber der Jacobi Methode, bei der für jede kinematische Bewegung der Kette die Inverse beziehungsweise die Pseudoinverse Matrix gelöst werden muss. Bei der Bewegung mit der *Cyclic Coordinate Descent Methode*, steigt mit der Anzahl der Kettenglieder nur die Anzahl der zu iterierenden Verbindungspunkte. Daher können mittels dieser Methode auch komplexe Strukturen in Echtzeit animiert werden.

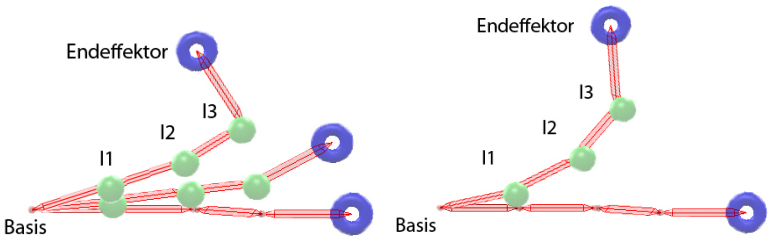

Abbildung 5.18: Vergleichende Darstellung der Verteilung der Winkeländerungen einer Verkettung.

Das Hauptargument für die Animation von virtuellen Puppen mit der *CCD Variante* liegt in den einfachen geometrischen Beziehungen, die zu jedem Zeitpunkt visuell nachvollzogen werden können. Deshalb ist es auch möglich jeden beliebigen Verbindungspunkt der Verkettung als aktiven Effektor für die kinematische Echtzeitanimation zu nutzen, was gerade für die Animation von Puppenmodellen oder anderen abstrakten Figuren von Bedeutung sein kann.

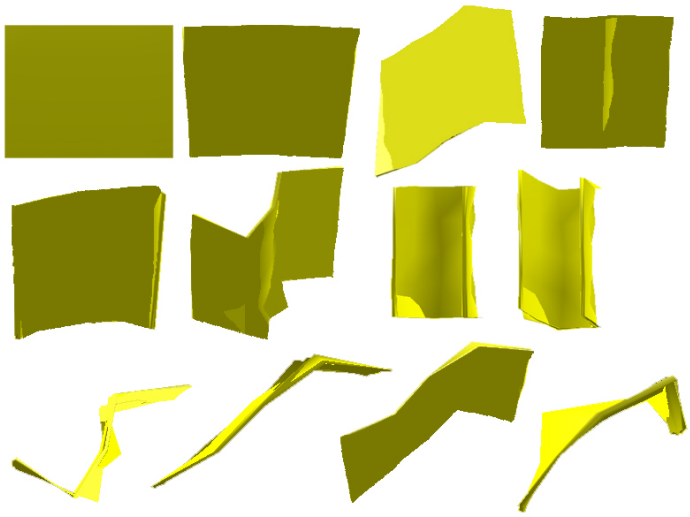

Abbildung 5.19: Beispielhaft kinematisch animiertes Blech.

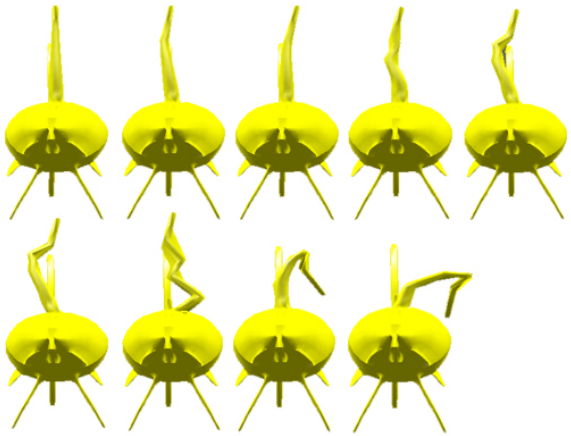

Abbildung 5.20: Beispielhaft kinematisch animierte Haiflosse.

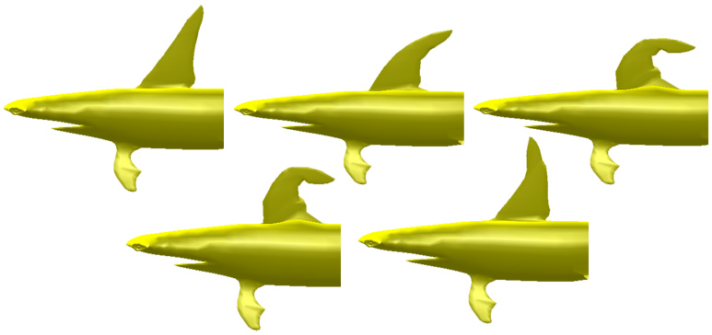

Abbildung 5.21: Beispielhaft kinematisch animierte Haiflosse.

Abbildung 5.22: Beispielhaft kinematisch animierte Lampe.

Abbildung 5.23: Beispielhaft kinematisch animiertes Ei.

Kapitel 6

Datenhandschuhgesteuerte Animation

Eine der größten Herausforderungen die sich im Bereich interaktiver Echtzeitanimation stellen, ist die Frage der optimalen Eingabemethoden für den Benutzer. Dieses Problem besteht unabhängig von der dem Animationssystem zugrunde liegenden Animationsstruktur. Einer der wesentlichen Aspekte ist die Frage nach der Möglichkeit der simultanen Steuerung multipler Effektoren. Diese Art der Animation ist mit klassischen Eingabegeräten wie Maus und Tastatur nur schwer und wenig intuitiv zu realisieren. Betrachtet man nochmals den Kontext dieser Arbeit aus der Sicht des Nutzers, wird deutlich warum die Einbindung eines Datenhandschuhs zur Animation der Puppen den größten Erfolg verspricht. Mit Hilfe des Handschuhs wird es dem Puppen-

Abbildung 6.1: Darstellung der Mundbewegung eines virtuellen Froschmodells.

spieler, welcher normalerweise echte Latex Puppen mit der Hand steuert, ermöglicht realitätsnah die Modelle zu bewegen, ohne sich tiefgreifend mit der Bewegung im 3D Raum auseinander setzen zu müssen. Dieser Vorteil lässt sich damit erklären, dass sich die Steuerung mit dem Handschuh und damit das ganze hierarchische Handmodell in abstrahierter Form virtuell abbilden lässt und daher jederzeit der reale-Welt Bezug für den Entwickler und Nutzer hergestellt werden kann.

Abbildung 6.2: Darstellung der Mundbewegung eines virtuellen Froschmodells.

6.1 Implementierung eines virtuellen Handmodells

Der in dieser Arbeit implementierte Datenhandschuh P5 wurde von der Firma Essentialreality [Glo] entwickelt und liefert über Infrarotsensoren Werte, die zur Positionsbestimmung im 3D Raum genutzt werden können. Dazu gehören unter anderem Informationen über den aktuellen Neigungswinkel und die Orientierung der Hand. Darüber hinaus kann über die Werte der Biegesensoren an den Fingerbändern des Handschuhs der aktuelle Biegegrad jedes einzelnen Fingers abgefragt werden. Abgesehen von den eindimensionalen Werten der Biegesensoren, liefert der Datenhandschuh keine weiteren Informationen über den aktuellen Zustand der Finger. Da die Informationen über die Adduktion oder Abduktion einzelner Finger nicht bestimmt werden können, müssen mehrere Sensorwerte hierarchisch kombiniert werden, um für jeden Finger drei Freiheitsgrade zu erhalten. Demzufolge dient die reale Hand in abstrahierter Form als Vorlage zur Bestimmung eines virtuellen Handmodells, bei dem sich beginnend am Handgelenk alle Zustandsänderungen bis in die Fingerspitzen auswirken.

Abbildung 6.3: Darstellung des P5 Datenhandschuhs der Firma Essentialreality.

In Abhängigkeit von der Art der Puppenanimation, kann das virtuelle Handmodell des Datenhandschuhs durch einen zentralen Rotationspunkt dargestellt werden, von dem splineähnliche Kurven als Fingerrepräsentation zu den jeweiligen zu steuernden Knochen ausgehen. Der Schnittpunkt einer solchen Kurve mit der entsprechenden Animationsstruktur repräsentiert die eigentliche Fingerspitze des entsprechenden Fingers, die zur Bewegung der referenzierten Objektteile herangezogen wird. Diese sind in der Abbildung 6.4 durch die rot gezeichneten Kugeln dargestellt. Von den Biegesensoren

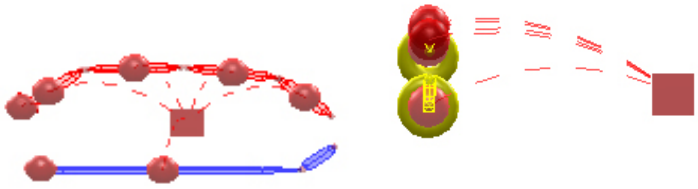

Abbildung 6.4: Darstellung des abstrakten Handmodells.

des Datenhandschuhs, welche in einer Kunststoffummantelung mit Hilfe von Kunststoffringen auf der Oberseite der menschlichen Finger fixiert werden, erhält man abhängig von dem aktuellen Biegegrad des jeweiligen Fingers eindimensionale Datensätze im Wertebereich von *[0 - 63]*. Im Allgemeinen gilt, dass für gestreckte Finger Null als Wert beziehungsweise kleine Werte ausgegeben werden. Wie auf der linken Seite der Abbildung 6.5 dargestellt, wird für einen komplett gebeugten Finger als maximaler Wert *63* zurück gegeben.
Zusätzlich zu den Sensordaten des Fingers wird auch der *yaw Winkel* des Handschuhs ausgewertet, um die Richtungsvektoren der nicht-kinematisch bewegten Knochen interaktiv zu beeinflussen. Der auf der rechten Seite der Darstellung 6.5 eingezeichnete *yaw Winkel*, welcher sich aus der Stellung der Hand zum Arm beziehungsweise aus dem errechneten Winkel am Handgelenk ergibt, kann optional in die Rotation des durch einen Finger referenzierten Elements einbezogen werden. Dadurch werden die eindimensionalen Werte der Fingersensoren um einen Freiheitsgrad hierarchisch erweitert. Daraus resultiert die Möglichkeit, eine durch einen Finger direkt referenzierte Ersatzstruktur trotz eindimensionaler Werte der Fingersensoren, auf einer halbkugelartigen Oberfläche rotieren zu lassen.

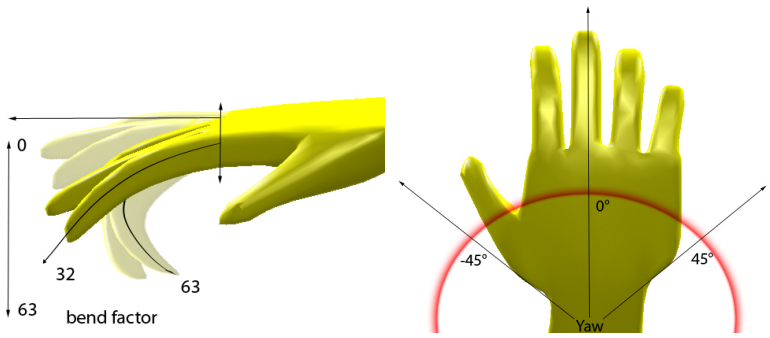

Abbildung 6.5: Darstellung der Fingerbewegung und der daraus gewonnenen Sensordaten.

Über die Finger gesteuerten Effektoren werden zur Bestimmung der Zielposition die Sensordaten jedes Fingers und optional die Position und Winkelstellung des Handschuhs ausgewertet. Dafür wird in einem ersten Schritt der Vektor $\vec{RF_i}$ vom globalen

Rotationspunkt zur Fingerspitze *i* bestimmt. Mittels der durch den Biegesensor eines Fingers erhaltenen Werte, wird im zweiten Schritt die aktuelle Rotationsrichtung der Fingerspitze abgeleitet und in die Berechnung der Zielposition des Effektors einbezogen.
Um die Rotation eines Effektors auf einer kugelartigen Oberfläche zu realisieren, muss die Anzahl der Freiheitsgrade des zu bewegenden Knochens von *1 DOF* auf mindestens *2 DOF* erhöht werden. Dazu wird im nächsten Schritt der *yaw Winkel* zwischen Hand und Handgelenk zusätzlich zur Rotationsrichtung mit der errechneten Zielposition in Relation gesetzt. In einem letzten Schritt wird in die Berechnung der Zielposition eines Endeffektors die aktuelle Änderungsrate, mit der sich die Werte eines Fingersensors des Handschuhs in einem Zeitschritt ändern, miteinbezogen. Anschließend kann ein Puppenspieler mit seinen Fingerbewegungen über den Datenhandschuh gesteuerte Objektteile, wie zum Beispiel den Mund einer Puppe unterschiedlich schnell öffnen und schließen.

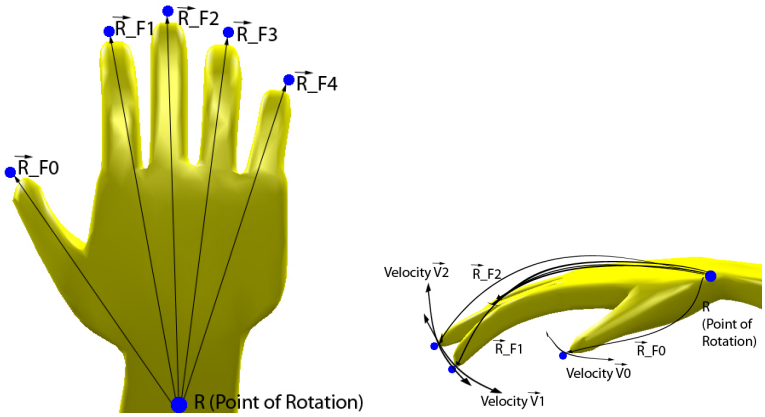

Abbildung 6.6: Darstellung der Vektoren zur Berechnung der Bewegungsrichtung.

6.2 Handgesteuerte Animation

Nachdem im ersten Teil dieses Kapitels auf die Anbindung des Handschuhs im Allgemeinen und die Einbeziehung der Sensorwerte im Speziellen eingegangen wurde, soll im Folgenden näher auf die Verknüpfung der Fingerspitzen mit den entsprechenden Effektoren oder Knochen eingegangen werden. Die Art der Verknüpfung von Handschuh und beweglichen Objektteilen hängt im Wesentlichen von der Art der durchzuführenden Animation ab, die sowohl kinematisch als auch nicht-kinematischer Art sein kann. Darüber hinaus wird in Kapitel 8 detailliert beschrieben, wie der Selektionsprozess durch den Benutzer realisiert werden kann.
Während der Implementierung der nicht-kinematischen handgesteuerten Puppenbewegung ist aufgefallen, dass die Rotation der durch ein bewegliches Objektteil bestimmten Puppenpunkte unter Umständen unerwünschte visuelle Ergebnisse zur Folge hat. Dies trifft zum Beispiel auf die in Abbildung 6.7 dargestellten Mundbewegungen zu, da durch die Rotation einzelner Teile der Ober- oder Unterlippe treppenförmige Verläufe entstehen können.

Abbildung 6.7: Darstellung der treppenförmigen Mundbewegung eines virtuellen Froschmodells.

Daher werden bei der Handsteuerung, die durch das virtuelle Handmodell repräsentiert wird, nicht nur die direkt durch ein bewegtes Element referenzierten Punkte, sondern alle Punkte einer Verbindung entsprechend ihrer aktuellen Lage bewegt. Dafür wird die durch die Fingerbewegung erzeugte Rotation eines Elementes mit einer Art Gaußverteilung auf alle Punkte der Verbindung übertragen. Für den Fall der Animation der Mundbewegung, bei der zum Beispiel vier Finger die Bewegung der Oberlippe steuern und der Daumen die Bewegung der Unterlippe bestimmt, wird die Fingerbewegung eines Fingers mit dem entsprechenden exponentiellen Faktor gewichtet und auf alle Punkte der Oberlippe übertragen.
Bei der in Gleichung 6.1 dargestellten Gewichtung werden die zu bewegenden Puppenpunkte in Abhängigkeit der Entfernung zur aktiven Fingerspitze unterschiedlich stark Gewichtet.

$$q_i = q_0 \exp^{-(|\sqrt{(x_i-x_0)^2+(y_i-y_0)^2+(z_i-z_0)^2}|)} \tag{6.1}$$

Der Rotationsfaktor q_0 repräsentiert in diesem Zusammenhang die Stärke der Bewegungsänderung, die von der aktiven Fingerspitze ausgeht. Mit zunehmender Entfernung vom Zentrum der Bewegung, welches genau durch einen der aktiven Fingervektoren $\vec{RF}_i$ bestimmt ist, wird die Gewichtung minimal. Daraus folgt, dass der Rotationsfaktor q_i für weiter entfernte Punkte i kleiner ist als für Punkte i die sich nah an der aktiven Fingerspitze befinden. Der Verlauf dieser Gewichtungsfunktion ist in der Abbildung 6.8 beispielhaft dargestellt.

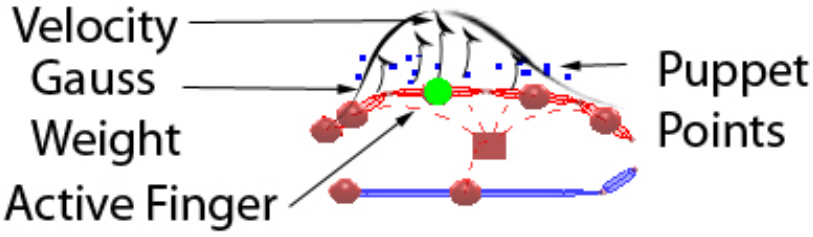

Abbildung 6.8: Interpolation der Rotationsbewegung.

Die Darstellung 6.8 zeigt, dass die Punkte im Zentrum des aktuell bewegten Elements am stärksten rotiert werden, während die Rotation der restlichen Punkte mit zunehmender Entfernung schwächer wird. Durch die Interpolation der Fingerbewegungen

zwischen zusammengehörigen Elementen wirken die Mundbewegungen authentischer. Darüber hinaus sind die Übergänge zwischen den einzelnen beweglichen Objektteilen nicht mehr so deutlich sichtbar.
Des Weiteren ist der Künstler in der Lage über seine unterschiedlichen Fingerstellungen verschiedene Mundausdrücke darzustellen, welche über das bloße Öffnen und Schließen hinausgehen und die Puppe lebendig wirken lassen. Dennoch ist dieses Verfahren nur für den speziellen Fall der Mundbewegungen mit dem hierarchisch abstrahierten Handmodell geeignet und findet bei der kinematischen Animation keine Anwendung.

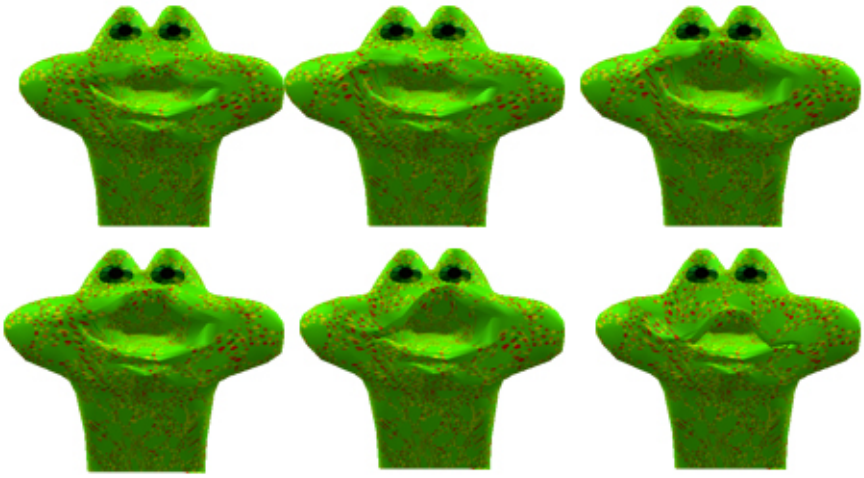

Abbildung 6.9: Darstellung der Mundbewegung eines virtuellen Froschmodells.

6.2.1 Animation kinematischer Strukturen

Für kinematische Bewegungen die über den Datenhandschuh gesteuert werden, stellt sich die Frage, wie mittels des Handschuhs die Zielposition des Effektors im 3D Raum bestimmt werden kann. Da hierfür drei Freiheitsgrade zur Verfügung stehen müssen, wird zusätzlich zum eindimensionalen Wert des Biegesensors, der über alle Finger verteilte *yaw Winkel* und die Distanz zwischen Handschuh und Infrarotempfänger zur Positionsbestimmung miteinbezogen.
Im Fall der kinematischen Animation besteht der größte Vorteil der Steuerung mit dem Handschuh darin, dass die simultane Bewegung multipler unabhängiger Endeffektoren mit der Maus nicht realisierbar ist. Die handgesteuerte Animation ermöglicht es, entweder mit jedem Finger einen Endeffektor verschiedener Verkettungen zu steuern oder aber auch mehrere Effektoren einer Verkettung zu bewegen. Lässt man den Umstand außer Acht, dass die Zielposition nicht durch den Nutzer über die Änderung der Mausposition hergeleitet wird, sondern durch mathematische Approximation des Handgelenks erfolgt, ist der restliche Verlauf der kinematischen Animation identisch mit der in Kapitel 5 vorgestellten Vorgehensweise.

6.2.2 Animation nicht-kinematischer Strukturen

Im Gegensatz zu den vorhergehenden Arten der handgesteuerten Animation liegt dem Modell zur Steuerung nicht-kinematischer Strukturen ein anderer Ansatz zugrunde. Kinematisch animierte Verkettungen werden in der Regel möglichst realistisch bewegt, während nicht-kinematische Verkettungen eher der Animation von Mundbewegungen und Gesichtsausdrücken dienen. Der Ursprung der in Abbildung 6.10 dargestellten Form der Bewegungsanimation ist der Animationstechnik realer Latex Puppen entnommen worden, bei der ein Puppenspieler mit seiner Hand in den Hohlkörper greift, um beispielsweise den Mund einer Puppe zu bewegen.

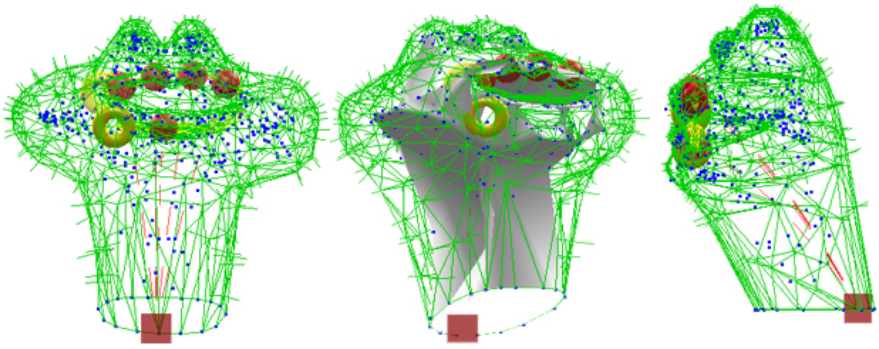

Abbildung 6.10: Darstellung der Hand-in-der-Puppe Metapher.

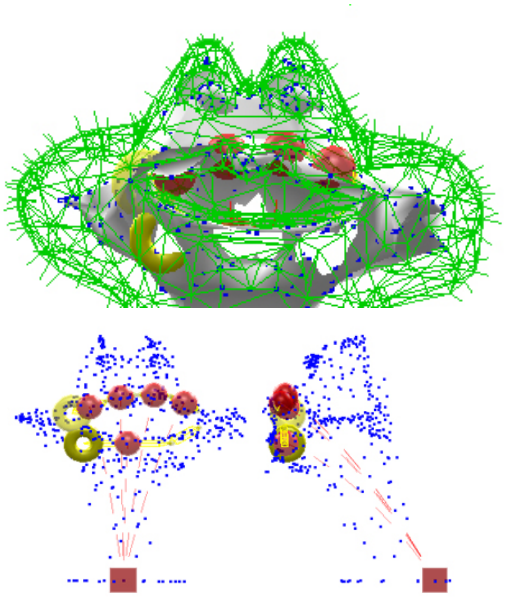

Abbildung 6.11: Darstellung der Hand-in-der-Puppe Metapher.

Die Selektion beweglicher Objektteile gleicht im Wesentlichen dem in Kapitel 3 vorgestellten Verfahren und wird vom Nutzer durchgeführt, indem die Skelett- oder Boxstrukturen in und um den zu bewegenden Bereich gelegt werden. Dies ist in der Abbildung 6.11 beispielhaft für die Animation des Mundbereichs einer Handpuppe dargestellt. Die linke Seite der Darstellung 6.11 zeigt wie die zu bewegenden Ersatzstrukturen mit der untersten Objektschicht verknüpft sind. Diese Objektschicht ist in der Abbildung grau dargestellt und besteht aus den auf der rechten Seite blau gezeichneten Punkten des reduzierten Puppenmodells.
An dieser Schicht knüpft das auf der linken Seite der Abbildung 6.11 grün gezeichnete *Feder-Masse-System* zur physikalischen Oberflächensimulation an. Im nächsten Schritt schließt sich die Verknüpfung der einzelnen Objektteile mit den jeweiligen Fingerspitzen an. Auf diese ist in Kapitel 8 detailliert einzugehen.
Über das abstrahierte Handmodell werden die Verbindungen zwischen den einzelnen Objektteilen und dem dazugehörigen Finger visualisiert. Darüber hinaus dient dieses Modell der visuellen Darstellung von Beschränkungen. Dazu zählt zum Beispiel, dass der erreichbare Bereich jeder Ersatzstruktur durch abstrahierte Magneten, wie in Abbildung 6.12 dargestellt, festgelegt werden kann.

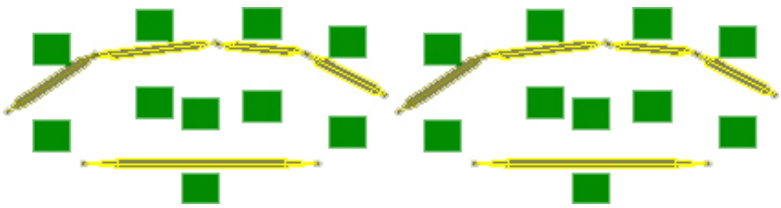

Abbildung 6.12: Einschränkung des Rotationsbereichs handgesteuerter Elemente.

Erreicht im Verlauf der handgesteuerten Animation ein Objektteil die Grenze seines gültigen Bewegungsbereichs, so wirkt eine Kraft entgegen der aktuellen Bewegungsrichtung. Dies ist der Kraft ähnlich, die beim Aufeinandertreffen gleich gepolter Magneten wirkt. Im Kontext der Animation des Mundbereichs kann auf diese Weise, wie in Abbildung 6.12 dargestellt, der maximale Öffnungswinkel der Ober- und Unterlippe beziehungsweise einzelner Bereiche des Mundes festgelegt werden. Dieses Prinzip der Bewegungseinschränkung wurde umgesetzt, indem zu jedem Element der Verkettung zwei primitive Objekte, welche in der Darstellung 6.12 grün eingezeichnet sind, den gültigen Bewegungsbereich eingrenzen.
Die Einbindung dieser Beschränkungen findet unabhängig von der Implementierung der *Collision Detection* statt, welche zwar Überschneidungen der Ober- und Unterlippe bei Mundbewegungen verhindert, aber nicht dazu geeignet ist das ursprüngliche Verhalten der Latexpuppen zu simulieren. Denkbar sind zum Beispiel der realen Welt entnommene Latexpuppen, deren Mund generell als geöffnet modelliert ist und dessen Mundbereiche sich aufgrund der Viskosität der Latex Materialien nie voll schließen lassen.

6.3 Schlussfolgerung

Die in dieser Arbeit implementierte handgesteuerte Puppenanimation ist besonders für Mundbewegungen geeignet, da hier der Bezug zur realen Spielweise der Puppenspieler am stärksten ausgeprägt ist. Um dies zu erreichen, wird die Datenhandschuhsteuerung visuell durch ein abstrahiertes Handmodell repräsentiert, welches wie bei echten Handpuppen im Inneren des Hohlkörpers platziert werden kann.
Wie im Verlauf des Kapitels gezeigt wurde, können beliebige zuvor definierte bewegliche Objektteile mit den Fingerspitzen des Handschuhs verknüpft werden. Für den speziellen Fall der Mundbewegung von virtuellen Handpuppen hat sich herausgestellt, dass die besten Animationsergebnisse erzielt werden können, wenn die Oberlippe einer Puppe mit den vier Fingern der Hand gesteuert wird. Die Bewegung der Unterlippe sollte idealerweise über die Daumenbewegung des Puppenspielers gesteuert werden. Dadurch kann mit Bezug auf echte Handpuppen der Mund der virtuellen Modelle auf die selbe Weise durch das Öffnen und Schließen der Hand bewegt werden. Darüber hinaus wird die Bewegungsgeschwindigkeit und Bewegungsrichtung der Finger des Handschuhs auf die Bewegung der entsprechenden Objektteile übertragen. Dadurch hat der Nutzer die Möglichkeit über die unterschiedlichen Fingerstellungen, in Echtzeit verschiedene Mundausdrücke der virtuellen Puppen zu erzeugen.
In Kombination mit der Maussteuerung oder der optional einbezogenen Positions- und Orientierungserkennung des Datenhandschuhs ist es möglich, die komplexen Mundbewegungen im Zusammenhang mit der Rotation oder Translation der Puppe zu animieren. Der Vorteil dieses Animationssystems ist, dass der Nutzer in Echtzeit eine sprechende Puppe animieren kann, die sich dabei im Raum bewegt und ihren Kopf rotieren kann. Im Gegensatz zu der in dieser Arbeit implementierten Art der Datenhandschuhsteuerung gibt es Ansätze, die sich von der Beschränkung auf Mund- und Gesichtsbewegungen lösen und stattdessen ganze Skelettmodelle handgesteuert animieren. Eine solche Methode wurde in [WCL04] implementiert und zeigt, dass es möglich ist, zuvor erstellte Ganzkörperbewegungsmodelle mit dem Handschuh zu imitieren und später handgesteuert in Echtzeit zu animieren. Zwar können auf diese Weise komplette hierarchisch aufgebaute Skelettmodelle interaktiv animiert werden, aber immer mit der Beschränkung, dass die Handbewegungen ähnlich dem imitierten Bewegungsmodell ist.
Für das in dieser Arbeit entwickelte Echtzeit-Animationssystem gelten diese Beschränkungen bewusst nicht, um dem Puppenspieler die Möglichkeit zu geben beliebige Bewegungsmodelle zu erstellen und zu animieren. In diesem Zusammenhang profitiert die handgesteuerte Animation von der physikalisch basierten Oberflächensimulation, welche die Auswirkungen der Mundbewegungen ähnlich der realen-Welt Vorbilder in Abhängigkeit von den Materialeigenschaften auf den Rest der Puppe überträgt. Dadurch entstehen zum Beispiel beim Öffnen des Mundes und beim Hochziehen der Mundwinkel natürlich wirkende Verzerrungen und Falten auf der Objektoberfläche, welche die Objektbewegungen lebendig erscheinen lassen.
Der Ansatz der handgesteuerten Mundbewegung von virtuellen Handpuppen bietet genug Potenzial für zukünftige Erweiterungen. Dazu gehören in erster Linie die Verbesserung der handgesteuerten Animation durch andere Datenhandschuhtypen und die weitere Integration der Handsteuerung in das Echtzeit-Animationssystem. Wünschenswert ist auch die gemeinsame Interaktion von mehreren Datenhandschuhen. Dies würde es ermöglichen, dass neben den Mundbewegungen, wie in Kapitel 2 vorgeschlagen, mit einem zweiten Handschuh weitere Details animiert werden. Darüber hinaus kann mit einem zweiten Datenhandschuh ein weiteres Objekt in Echtzeit animiert werden, um

die Lebendigkeit der virtuellen Szene zu erhöhen.

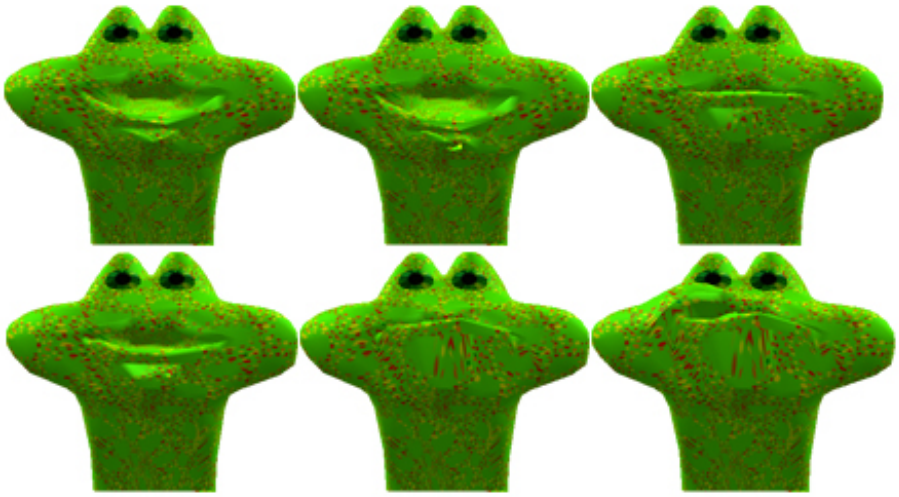

Abbildung 6.13: Darstellung der Mundbewegung eines virtuellen Froschmodells.

Kapitel 7

Constraints

In diesem Abschnitt werden die für die Animation von artikulierten Figuren notwendigen Beschränkungen und deren Implementation erläutert. Darauf aufbauend wird gezeigt, welche Interaktionstechniken dem Nutzer zur Verfügung stehen, um die *Constraints* an den Verbindungspunkten den spezifischen Eigenschaften der gewünschten Bewegung anzupassen.

Die Einbindung von Beschränkungen in den kinematischen Animationsprozess ist eine Grundvoraussetzung um Bewegungen zu erzielen, die den Erwartungen des Nutzers und gegebenenfalls dem menschlichen Bewegungsmodell in abstrahierter Form entsprechen. Um dieses Ziel zu erreichen, gibt es zahlreiche Ansätze zur *Constraint* basierten Animation, die sich in Abhängigkeit vom zugrunde liegenden Animationsmodell direkt in die kinematische Berechnung einbinden lassen.
In [ESHD05] werden einige dieser Modelle vorgestellt, die sich im Wesentlichen auf die Regulierung der in der Kinematik wirkenden Kräfte konzentrieren. Dieser Vorgang kann sowohl direkt in den kinematischen Prozess eingebunden werden, indem die in der Rechnung genutzten kinematischen Kräfte auf den durch die Beschränkung festgelegten gültigen Wertebereich angepasst werden, beziehungsweise indem durch Anwendung von *Time-Stepping Methoden*[ESHD05] die erzielten Bewegungsergebnisse immer im Voraus der eigentlichen kinematischen Animation verifiziert werden.

Der in dieser Arbeit verfolgte Ansatz beschränkt sich auf die Wahrung der Gültigkeit einer bewegten Verkettung in Bezug auf einen Satz von Beschränkungen die für die Elemente der Verkettung festgelegt werden. Die Voraussetzung dafür ist, dass sich die bewegten Objektteile zu Beginn jeder Animation in einem gültigen Bereich befinden. Im Gegensatz dazu ist es theoretisch auch möglich, nach der kinematischen Bewegungsberechnung von Verkettungen, die verletzten Beschränkungen im Nachhinein zu erfüllen, indem die Bewegungsergebnisse auf den gültigen Bereich reduziert werden.
Wie sich im Folgenden zeigen wird, eignen sich hierfür geometrische Beschränkungen. Diese ermöglichen es dem Nutzer über die grafischen Repräsentationen an den Verbindungspunkten zwischen den Objektteilen verschiedene Gelenktypen zu simulieren. Im Grunde basiert dieser Ansatz auf der Idee die geometrischen Beschränkungen als kinematische Kräfte in die Bewegungsberechnung einzubeziehen, ohne dass vom Nutzer detailliertes Hintergrundwissen über das simulierte System vorausgesetzt wird.

7.1 Arten von Constraints

Die auf spezifische *links* bezogenen Beschränkungen dienen hauptsächlich zur Definition unterschiedlicher Gelenktypen, indem die Anzahl der Freiheitsgrade und der gültige Winkelbereich des Gelenks für jeden *link* der Verkettung individuell festgelegt wird. Darüber hinaus dienen Beschränkungen mit Bezug auf die Gelenkorientierung und die maximale Änderungsrate des Winkels an einem Gelenk weniger der spezifischen Gelenkdefinition, als der Bestimmung des globalen Bewegungsverhaltens einer kinematischen Verkettung. Zusätzlich kann es in Abhängigkeit vom verwendeten kinematischen Animationsalgorithmus angebracht sein Beschränkungen einzuführen, um gegebenenfalls Schwächen in der mathematischen Gelenkwinkelbestimmung des Algorithmus auszugleichen. Auf die *CCD Methode* bezogen wurden deshalb Gelenkgewichte eingeführt, die dazu dienen die mathematische Gewichtung der *links* so zu wählen, dass die Bewegungsänderung eines Endeffektors gleichmäßig über alle Verbindungen der Verkettung verteilt wird und nicht nur die *links* in der Nähe des Endeffektors von dessen Bewegung profitieren. Wie auf der rechten Seite der Abbildung 7.1 dargestellt,

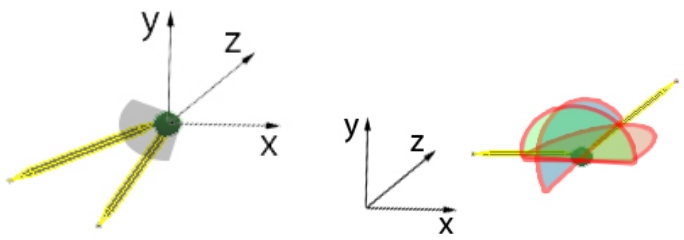

Abbildung 7.1: Darstellung der gültigen Gelenkwinkel und Winkelorientierung an einem Verbindungspunkt.

kann der Nutzer interaktiv an jedem Gelenk den gültigen Rotationsbereich für alle drei Ebenen festlegen beziehungsweise den gültigen Winkelbereich in einer Ebene auf Null reduzieren, was in der Konsequenz den Verlust eines Freiheitsgrad am aktuellen Gelenk bedeutet. Dafür werden an jedem Verbindungspunkt zwischen zwei Objektteilen drei fächerartige Bänder um das Gelenk aufgespannt, mit denen der gültige Wertebereich für jede Ebene separat festgelegt werden kann. Auf diese Weise ist es möglich, vom *Ball-In-Socked Gelenktyp*[ESHD05], der mindestens drei Freiheitsgrade besitzt und eine Abstraktion des menschlichen Schultergelenks darstellt, bis zum *Revolute Gelenktyp*, der mit einem Freiheitsgrad eine virtuelle Repräsentation des Scharniergelenks ist, verschiedene Gelenktypen zu simulieren. Letzter ist auf der linken Seite der Darstellung 7.1 schematisch dargestellt und zeigt wie der gültige Rotationsbereich für die *XY-Ebene* begrenzt werden kann.
Im Unterschied dazu wird in der Abbildung 7.2 gezeigt, dass der gültige Arbeitsbereich an einem Gelenk auch als drei dimensionaler Kegel dargestellt werden kann. Der Nutzer kann direkt mit dem Kegel interagieren, um die Orientierung des Arbeitsbereiches zu modifizieren. Darüber hinaus ist es möglich den gültigen Winkelbereich für jede der drei Ebenen durch Öffnen oder Zusammenziehen der Kegelöffnung zu beeinflussen.

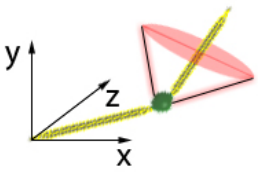

Abbildung 7.2: Kegelförmige Darstellung des gültigen Bewegungsbereichs an einem Verbindungspunkt.

Auf der linken Seite der Abbildung 7.3 ist ein Kugelgelenk mit drei Freiheitsgraden beispielhaft dargestellt. Vergleichend dazu zeigt die rechte Seite einen universellen Gelenktypen mit zwei Freiheitsgraden. Bei diesen schematischen Abbildungen beziehen sich die Freiheitsgrade auf die Rotationsebenen, ohne die Möglichkeit der Orientierungsänderung einzubeziehen. In diesem Zusammenhang wird in Abbildung 7.4 die Abstraktion eines Scharniergelenks mit einem Freiheitsgrad dargestellt.

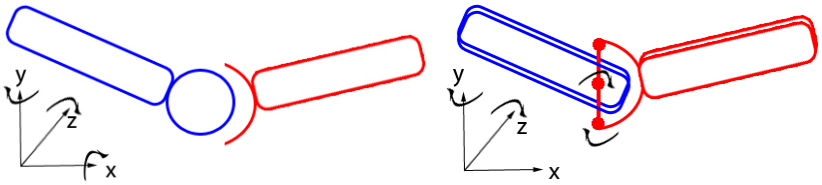

Abbildung 7.3: Schematische Darstellung von 3 DOF und 2 DOF Gelenktypen.

Im Gegensatz zur *Jacobi Methode*, bei der die Änderungsrate der Winkel zwischen den Objektteilen durch Lösen der inversen beziehungsweise der pseudoinversen Jacobi-Matrix hergeleitet werden, erlaubt es die *CCD Methode* die Beschränkungen an jedem Gelenk individuell anzupassen. Dies ist möglich, weil klar ist welcher *link* am aktuellen Iterationsschritt modifiziert wird und die geometrischen Beschränkungen an diesem Gelenk validiert werden können, bevor der nächste *link* der Verkettung bewegt wird.

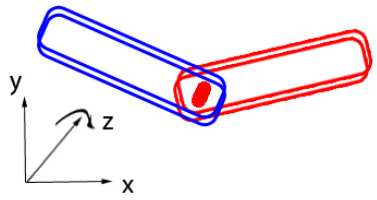

Abbildung 7.4: Schematische Darstellung eines 1 DOF Gelenktyps.

7.2 Schlussfolgerung

Die in dieser Arbeit eingeführten *Constraints* beziehen sich auf die Anzahl der Freiheitsgrade und den gültigen Arbeitsbereich der Verbindungspunkte um das Bewegungsverhalten von zwei miteinander verbundenen Elementen. Diese gelenkspezifischen Beschränkungen werden im Zusammenhang mit der Definition von kinematischen Verkettungen über die im Verlauf diese Kapitels vorgestellten grafischen Schnittstellen festgelegt. Dadurch ist es möglich, durch Selektion des entsprechenden Verbindungspunkts interaktiv die gültigen Rotationsachsen für diesen *link* festzulegen. Des Weiteren kann der Nutzer an jedem Gelenkpunkt den minimalen und maximalen Öffnungswinkel und dessen Orientierung zwischen zwei Elementen festlegen.
Diese Arten von geometrischen Beschränkungen beschreiben die relative Bewegung zwischen zwei Elementen einer kinematischen Verkettung und dienen der Simulation unterschiedlicher Gelenktypen. Für den in Kapitel 5 vorgestellten kinematischen Animationsalgorithmus sind zudem weitere Beschränkungen nötig, die dem Benutzter aber verborgen bleiben. Diese stellen sicher, dass die Verkettung und die beweglichen Objektteile einer Puppe während der Bewegung nicht auseinander fallen.
Dazu zählt zum Beispiel, dass nach jedem Iterationsschritt die kinematische Pose aktualisiert und im Anschluss das in [ESHD05] vorgestellte Kontaktproblem gelöst wird. Bei der Bewegungsberechnung mit der *CCD Methode* wird in einem Iterationsschritt immer nur der aktuell iterierte Verbindungspunkt bewegt. Daher muss die kinematische Verkettung an die veränderte Gelenkstellung des aktuell bewegten Elements angepasst und getestet werden, ob alle bewegten Elemente einer Verkettung an ihren Verbindungspunkten übereinstimmen, bevor zum nächsten Verbindungspunkt iteriert wird. Bei diesem Test ist zu prüfen ob zwei durch ein Gelenk verbundene Elemente sich an ihren Verbindungspunkten überlagern. Das Kontaktproblem[ESHD05] muss in der Regel für alle Arten von Rotationsgelenken erfüllt sein, da sich zwei miteinander verbundene Elemente im Allgemeinen nur bei Translationsverbindungen voneinander entfernen.
Da für die kinematische Bewegung sowohl globale als auch die lokalen Beschränkungen der Verkettung erfüllt sein müssen, kann der in [Wel93] vorgeschlagene Ansatz zur hierarchischen Gruppierung der *Constraints*, für die zukünftige Entwicklung genutzt werden. Bei diesem werden auch die *lokalen Constraints* in die *CCD Methode* integriert. Während des rekursiven Iterationsprozess über alle Verbindungen einer Verkettung werden an jedem *link* die zusätzlichen Beschränkungen der Gelenke zwischen der Basis und dem aktuellen Verbindungspunkt aufgelöst, bevor der aktuell iterierte Verbindungspunkt bewegt wird.
Für die künftige Entwicklung des Echtzeit-Animationssystem bietet es sich an, mittels

verschiedener Beschränkungen kinematisches Verhalten zu simulieren, dass nicht dem menschlichen Bewegungsmodell entspricht.
Denkbar wäre es etwa durch impulsartige Bewegungen des Endeffektors ein wellenartiges Verhalten der Verkettung zu simulieren, bei dem sich die erregte Schwingung von einem Ende der beweglichen Objektteile bis zum anderen Ende fortsetzt. Darüber hinaus ist es bei der Puppenanimation in einigen Fällen auch wünschenswert, die Auslenkung des Endeffektors von seiner Ruheposition aus durch Simulation physikalischer Kräfte als Federschwingung zu animieren, bis der Endeffektor nach dem Abklingen der wirkenden Kräfte aufgrund der simulierten Reibung und des Luftwiderstands wieder seine Ruheposition einnimmt.

Kapitel 8

Echtzeit-Animationssystem für virtuelle Handpuppen

8.1 Überblick

Das im Zusammenhang mit dieser Arbeit entwickelte Puppen-Animationssystem bietet dem Nutzer die Möglichkeit, interaktiv für jedes beliebig geladene Puppenmodell skelett- oder boxähnliche Animationsstrukturen zu erstellen und diese in Echtzeit zu animieren.
Dafür stehen im Editor drei getrennte Fenster zur Verfügung, die für die Bearbeitung und Animation in unterschiedlichen Blickwinkeln genutzt werden können. Nur innerhalb dieser Fenster wird das Puppenobjekt zusammen mit der Animationsstruktur, dem virtuellen Handmodell und den geometrischen Beschränkungen gezeichnet. Während im Hauptfenster des Editors das finale animierte Objekt dargestellt wird, bieten die *Viewports* die Möglichkeit das Puppenmodell aus verschiedenen Ansichten zu betrachten und dessen verschiedene Objektschichten und die einzelnen beweglichen Objektteile zu visualisieren. Darüber hinaus sind diese Bereiche für die Interaktion des Nutzers zur Erstellung und Bewegung der Ersatzstrukturen gedacht, wohingegen im Hauptfenster die Kamera in und um die komplette Szene transformiert beziehungsweise rotiert werden kann, um die optimale Kameraeinstellung für die finale Szene zu definieren.

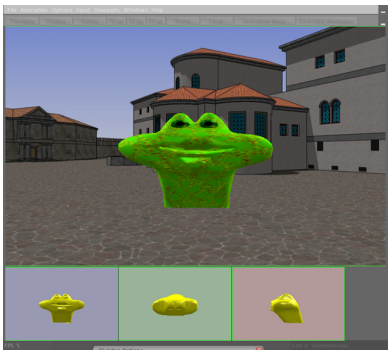

Abbildung 8.1: Ansicht der Editorumgebung.

8.2 Definition kinematischer und nicht-kinematischer Verkettungen

Nachdem ein Puppenmodell in den Editor geladen wurde, kann der Nutzer zwischen Box- oder Skelettstrukturen zur Repräsentation der beweglichen Objektteile wählen und bei Bedarf ein oder mehrere *Viewports* zum Editieren öffnen. Diese können auf der Oberfläche mit der Kombination *[STRG] + linke Maustaste* frei bewegt und mit der Kombination *[ALT] + linke Maustaste* in ihrer Größe geändert werden. Innerhalb dieser Bereiche können Boxen oder Knochen mit dem *New* Knopf auf dem in Abbildung 8.2 dargestellten *Optionsfenster*, welches nach Wahl des Animationsmodus im Editor erscheint, gezeichnet werden. Hat man zum Beispiel drei Knochen in die Sze-

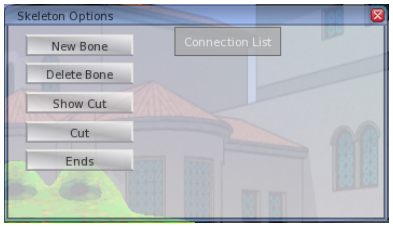

Abbildung 8.2: Ansicht des Ersatzstruktur-Fensters.

ne gezeichnet, kann man diese zu einer Verkettung zusammenfassen, indem im ersten Schritt zwei Knochen miteinander verknüpft werden. Dafür muss die *[SHIFT]* Taste gehalten werden, während zuerst ein Endpunkt des ersten und danach der Endpunkt des zweiten Elements markiert werden. Ist dieser Vorgang abgeschlossen kann die *[SHIFT]* Taste wieder losgelassen werden, um die Elemente miteinander zu verbinden. Um den dritten Knochen der Verkettung hinzuzufügen, muss diese im Optionsfenster in der *Connection List* ausgewählt werden. Die jeweils aktive Verkettung wird visuell durch die blau gezeichnete Kettenglieder dargestellt. Nach diesem Schema können der aktiven Verkettung beliebig viele Elemente hinzugefügt werden.

8.3 Animationsstruktur mit dem Puppenmodell verknüpfen

Im Anschluss an die Definition von zu animierenden Ersatzstrukturen stellt sich die Frage, wie die einzelnen Elemente mit dem Puppenmodell verknüpft werden können. Da auf den mathematischen Hintergrund der Bereichsdefinition zur Bestimmung der relevanten Puppenpunkte bereits in Abschnitt 3.1.2 eingegangen wurde, werden im Folgenden die dafür nötigen Schritte im Editor vorgestellt.
Die Bestimmung des Einflussbereichs der einzelnen Kettenglieder auf das zu animierende Objekt erfolgt in der Regel nachdem die Verkettungen definiert und in der Szene beziehungsweise innerhalb des Modells positioniert ist. Nachdem in einem ersten Schritt ein Element ausgewählt ist, muss im Optionsfenster der Schalter *Show Cut* aktiviert werden, um den Einflussbereich des Knochens auf das Objekt, wie in Darstellung 8.3 gezeigt, zu visualisieren. Über sechs speziell ausgewiesene Punkte des Ellipsoid, von denen je zwei in einer Ebene liegen, kann der Einflussbereich am Puppenmodell zur Bestimmung eines beweglichen Objektteil durch den Nutzer in seiner Größe

angepasst werden. Schließlich müssen noch die im gültigen Bereich des Knochens

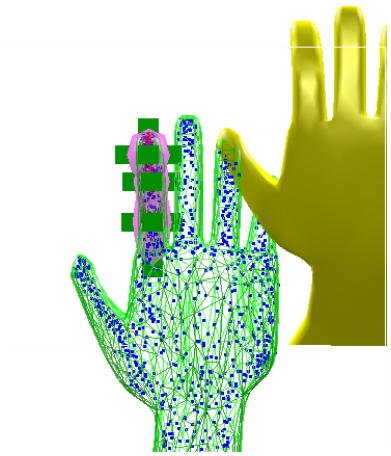

Abbildung 8.3: Definition der beweglichen Objektteile am Beispiel eines Handmodells.

liegenden Objektpunkte identifiziert werden. Sobald im Optionsfenster des selektierten Elements der *Cut* Schalter betätigt wurde, sind die ausgewählten Puppenpunkte mit dem Knochen verbunden und reagieren auf dessen Bewegungen. Diese Beziehung kann auch visuell dargestellt werden, indem über die Menü Auswahl *Options* → *Puppet Visualization* das *Visualisierungsmenü* geöffnet wird. In diesem Menü können auf der einen Seite die verschiedenen Schichten des Objekts für die Visualisierung im Hauptfenster aktiviert und auf der anderen Seite die Punkte der untersten Schicht des Modells, an der die Animationsstrukturen ansetzten, in den *Viewports* gezeichnet werden. Die mit den Ersatzstrukturen verknüpften Puppenpunkte dieser Schicht werden für den Nutzer farblich hinterlegt.

8.4 Kinematische vs. nicht-kinematische Animation

Für die kinematische Animation von Verkettungen muss entweder einer der Endpunkte der Animationsstruktur oder einer der Verbindungspunkte zwischen den Kettengliedern selektiert werden. Abhängig von der Position des zu bewegenden Punktes spricht man in der Regel von einem *Endeffektor* falls der Startpunkt oder Endpunkt einer Verkettung bewegt werden soll, andernfalls von *Effektor*, wenn einer der inneren Gelenkpunkte durch den Benutzer gesteuert wird. Verbindungs- und Endpunkte der Verkettung die aktuell selektiert sind, werden farbig hinterlegt und können über das *Link Options* Fenster, welches in der Abbildung 8.5 dargestellt ist, modifiziert werden. Im oberen Teil des Fensters befinden sich die Schalter, um dem Verbindungspunkt die vordefinierten Eigenschaften abstrahierter Gelenke zuzuweisen oder aber um diesen Punkt als Endeffektor [1] zu deklarieren. Nachdem die Ersatzstruktur fertig erstellt worden ist, muss auf dem in der Darstellung 8.6 abgebildeten *Aktionsmenü*, der *Rotationsschalter* und der Arbeitsbereich, dass heißt die Anzahl der Freiheitsgrade zur Definition der aktuellen Rotationsachsen des (End-)Effektors aktiviert werden.

[1] Verbindungs- und Endpunkte einer Verkettung, die als Effektoren beziehungsweise Endeffektoren markiert sind, werden zur besseren Wahrnehmung als farbige Tori gezeichnet.

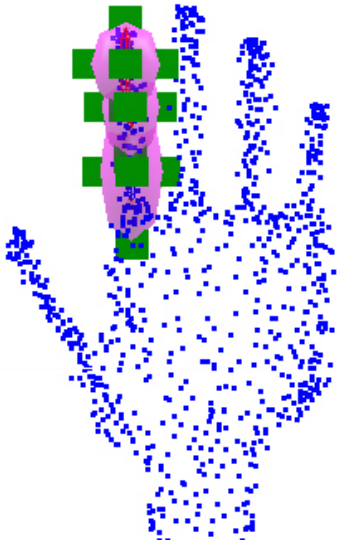

Abbildung 8.4: Ansicht der referenzierten Punkte auf der innersten Objektschicht.

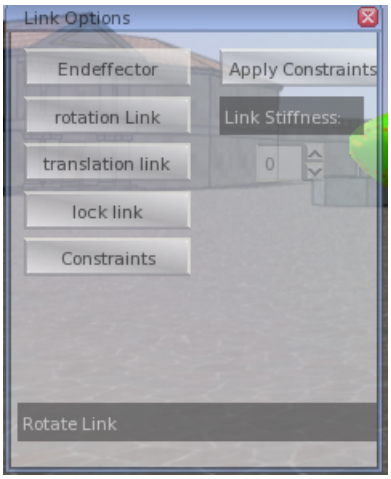

Abbildung 8.5: Ansicht des Fensters zur Gelenkspezifikation.

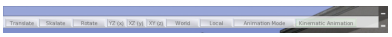

Abbildung 8.6: Ansicht des Interaktionsmenü.

Um die Bewegungen auf die mit den Elementen verbundenen Objektteile zu übertragen, muss im nächsten Schritt im *Aktionsmenü* der *Animation Mode Knopf* aktiviert werden. In den kinematischen Animationsmodus wird durch die Aktivierung des *Kinematic Animation* Schalters gewechselt, was in der Konsequenz heißt, dass die Bewegungen der (End-)Effektoren auf die gesamte Ersatzstruktur übertragen werden und dadurch die aktuelle Pose der Verkettung bestimmt wird.

8.5 Datenhandschuhgesteuerte Puppenanimation

Die für die handgesteuerte Animation von beweglichen Objektteilen notwendigen Schritte werden in diesem Abschnitt in chronologischer Reihenfolge vorgestellt und können auf dieselbe Weise im Animationssystem nachvollzogen werden. Dies setzt aber voraus, dass der Datenhandschuh mit dem System verbunden ist.
Die nächsten Schritte beziehen sich nur auf die Handsteuerung und setzten voraus, dass bereits Animationsstrukturen definiert und diese mit dem Puppenmodell verbunden sind. Davon ausgehend erfolgt im nachfolgenden Schritt die Kalibrierung des Datenhandschuhs, welche über den Eintrag *Input* → *Data Glove* gestartet werden kann. In dem in der Abbildung 8.7 dargestellten *Input Optionsmenü* muss ein Haken an den Eintrag *Glove Fingers* gesetzt werden, um im nebenstehenden Fenster die Fortschrittsbalken jedes Fingers zu sehen. Diese repräsentieren den aktuellen Grad der Auslenkung beziehungsweise den Biegegrad der Finger. Der standardmäßige Wertebereich der Finger des Handschuhs liegt zwischen 0 bei ausgestreckten und 63 bei stark gebogenen Fingern, was für die Animation unpraktikabel ist. Deshalb sollte der Handschuh kalibriert werden, um die Ausgangs- und Ruheposition der Hand festzulegen. Erst danach ist es zum Beispiel möglich, durch weiteres Öffnen oder Schließen der Hand, in Relation zu der vorher definierten Ruheposition, den Mund einer Puppe weiter zu öffnen oder zu schließen. Ist der Kalibrierungsvorgang zur Zufriedenheit des Nutzers

Abbildung 8.7: Ansicht des Datenhandschuh-Interaktionsfensters.

abgeschloßen, kann in einem nächsten Schritt der Handschuh mit den beweglichen Objektteilen verbunden werden, indem dass in Abbildung 8.8 gezeigte Fenster über den Eintrag *Input* → *Connection Matrix* geöffnet wird.
Für das weitere Vorgehen ist entscheidend, ob der Handschuh als Ersatzeingabegerät der Maus zur simultanen Steuerung mehrerer unabhängiger (End-)Effektoren oder zur

Bewegung verschiedener Objektteile, die über die hierarchischen Beziehungen des virtuellen Handmodell zueinander in Relation gesetzt sind, genutzt werden soll. Für den

Abbildung 8.8: Ansicht des Fensters zur Verbindung der beweglichen Objektteile mit der Handsteuerung.

ersten Fall sind keine weiteren Einstellungen nötig und die zu steuernden Elemente können direkt ausgewählt werden. Dahingehend muss für den zweiten Fall zusätzlich das Handmodell aktiviert werden und der globale Rotationspunkt, den alle bewegten Elemente gemeinsam haben, positioniert werden. Dieser Punkt entspricht im Wesentlichen der Abstraktion des Handgelenks und beeinflusst durch seine Lage in Relation zu den Fingerspitzen, welche durch die ausgewählten Objektteile bestimmt sind, die Orientierung der Hand und in der Konsequenz das Rotationsverhalten der beweglichen Animationsstrukturen.

8.6 Definition von Beschränkungen

Da der Aspekt der Bedienbarkeit für das *Puppen-Animations-Projekt* des Lehrstuhls für grafische Datenverarbeitung insgesamt einen hohen Stellenwert einnimmt, hat sich in der Entwicklungsphase dieser Animationsumgebung die Frage gestellt, wie man dem Nutzer die Möglichkeit gibt auf grafische Weise die mathematisch beschreibbaren Beschränkungen für die Animation im Allgemeinen und *Constraints* an den Verbindungspunkten im Speziellen interaktiv festzulegen. Die spezifischen Einstellungen der Gelenke können an jedem Verbindungspunkt einer Verkettung direkt eingestellt werden, indem der entsprechende Knoten selektiert und im *Skeleton Options* Fenster der *Link Range* Schalter aktiviert wird. Danach wird der gültige Winkelbereich und die Winkelorientierung an dem Gelenk visuell in Form eines sich um die Gelenkrepräsentation spannendes Farbpolygon dargestellt. Dieses kann interaktiv für jede Ebene des 3D Raums aufgezogen beziehungsweise soweit minimiert werden, bis der Grad der gültigen Winkeländerung gegen Null geht, was den Verlust eines Freiheitsgrad an dem Verbindungspunkt zur Folge hat.

In Abschnitt 6.3.2 wurde bereits erläutert, dass für die abstrahierte hierarchische Handsteuerung der gültige Rotationsbereich, im Gegensatz zu der oben vorgestellten Methode, über zu jedem Element gehörende *Bewegungsstopper* eingegrenzt wird. Dies hat den Hintergrund, dass auf diese Weise die Beschränkungen des Bewegungsbereichs visuell auf der *Animationsseite* eingestellt werden können, ohne dass der Nutzer zwischen den Ansichten wechseln muss, um den in der Regel in oder hinter der Puppe liegenden Rotationspunkt zu erreichen. Des Weiteren lassen sich über die in der Abbildung 8.9 dargestellten *Bewegungsbegrenzer* die zwischen gleich gepolten Magneten wirkenden physikalischen Kräfte simulieren. Daraus folgt, dass Elemente die sich nah

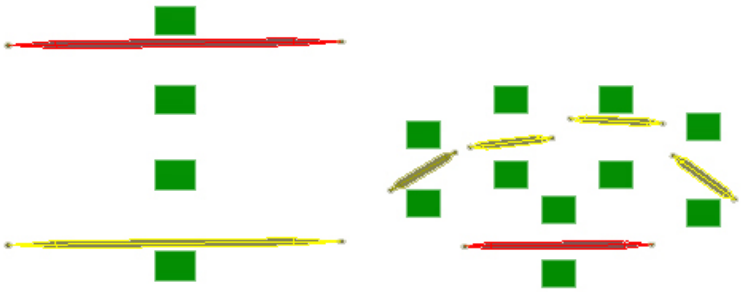

Abbildung 8.9: Beschränkung der handgesteuerten Bewegung.

an den Endbereichen befinden, bei gleicher Fingerbewegung bis zu einem gewissen Grad weniger stark rotieren, als Elemente die sich weiter weg befinden. Auf der linken Seite der Abbildung 8.9 wird eine Ersatzstruktur mit den dazugehörigen *Bewegungsbegrenzern*, dargestellt. Für die handgesteuerte Mundbewegung einer Puppe ergibt sich die Darstellung auf der rechten Seite der Abbildung 8.9.
In Kapitel 5 wurde ebenfalls gezeigt, dass die Verbindungspunkte einer kinematischen Verkettung unterschiedlich stark zur Bewegung beitragen können. Dazu kann das Gelenkgewicht im *Link Options* Fenster des aktuell selektierten Verbindungspunkts zwischen 0 und 1 variiert werden, was visuell mit der Änderung der Farbwerte an dem Gelenk korrespondiert.

8.7 Oberflächensimulation

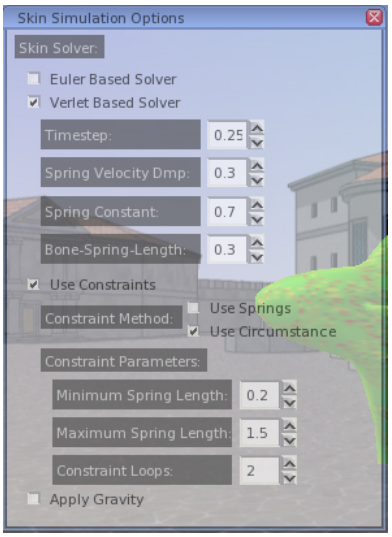

Abbildung 8.10: Ansicht des Fensters zur Spezifikation der Oberflächenparameter.

Im Verlauf der Arbeit wurde schon mehrfach darauf hingewiesen, dass die in diesem Animationssystem implementierten Echtzeit-Animationstechniken mit der Oberflächensimulation aus [PW07] kombiniert wurden. Die Parameter des zugrunde liegenden Feder-Masse-Systems können über den Eintrag *Options → Skin Simulation* mit dem in Abbildung 8.10 dargestellten Fenster angepasst werden.

8.8 Laden und Speichern von Animationsstrukturen

Die im Editor erstellten Ersatzstrukturen und die Informationen über die dadurch definierten beweglichen Teile des Objekts werden für die weitere Verwendung in die XML-Struktur des Puppenmodells mit Hilfe der durch [Tin] bereitgestellten Schnittstellen integriert und können beim nächsten Ladevorgang rekonstruiert werden. Ist eines dieser vordefinierten Puppenmodelle im Editor geladen, muss der Nutzer nur noch die für die Bewegung relevanten Elemente aktivieren beziehungsweise mit dem Datenhandschuh verbinden, um den Animationsprozess zu starten.
In der Abbildung 8.11 wird beispielhaft der Ausschnitt aus der XML-Datenstruktur eines vordefinierten Puppenmodells gezeigt. Im ersten Teil der Darstellung sind die in der Szene vorhandenen Ersatzstrukturen mit den spezifischen Eigenschaften definiert. Danach folgen die Bezeichner für die Objekte, welche intern die erstellten Verkettungen repräsentieren. Diese enthalten die abstrakten Objekte zur Gelenk-Repräsentation welche die zuvor eingelesenen Ersatzstrukturen miteinander verknüpfen. Im letzten Teil des Ausschnitts sind die notwendigen Informationen für die Steuerung mit dem Datenhandschuh dargestellt. Dazu gehören unter anderem Informationen über die mit den Fingern des Handschuh verbundenen Objektteile und über die Position des virtuellen Handmodells in der Szene.

```
<SKELETONBONES>
  <SKELETONBONE ID="0" CUTMARK="0" RELATEDLINK="-1">
    <STARTNODE X="-1.02899" Y="-0.0086162" Z="1.5" />
    <ENDNODE X="-0.259763" Y="0.200261" Z="1.5" />
    ...
    <ELLIPSOID A1="1" A2="1" A3="1"> ... </ELLIPSOID>
    <MAGNET> ... </MAGNET>
  </SKELETONBONE>
</SKELETONBONES>
...
<CONNECTIONS>
  <CONNECTION ID="1" EXTERNCONTROL="0" CONNMATRIXID="0">
    <LINK ID="1" LINKMATRIXID="-1"> ... </LINK>
    ...
  </CONNECTION>
  ...
</CONNECTIONS>
<FINGEROBJECTS>
  <FINGER01 RELATEDLINK="0" RELATEDCONN="0" ACTIVE="0" />
  ...
</FINGEROBJECTS>
<HANDOBJECT>
  <ROTATEPOINT X="0" Y="0" Z="0" />
  <FO1 X="0" Y="0" Z="0" />
  ...
</HANDOBJECT>
```

Abbildung 8.11: XML-Struktur der gespeicherten Elemente.

Kapitel 9

Zusammenfassung und Ausblick

9.1 Zusammenfassung

Die in dieser Arbeit vorgestellten Aspekte der Echtzeitanimation von virtuellen Handpuppen und der in diesem Zusammenhang entwickelte Prototyp zeigen, dass es möglich ist eine Modellier-und Animationsumgebung zu schaffen. Diese erlaubt es Künstlern auf einfache Weise Bewegungsmodelle zu bestimmen, welche in Echtzeit animiert werden können, ohne das dafür komplexes Wissen über die 3D Computergrafik- und Animationstechniken notwendig ist.
Der in diesem Zusammenhang implementierte und erweiterte Algorithmus der *CCD Methode* eignet sich aufgrund seiner linearen Eigenschaften sehr gut für die kinematische Bewegung von Charakteren. Unabhängig von der Anzahl der Kettenglieder einer kinematischen Verkettung bleiben die bei der *CCD Methode* in jedem Iterationsschritt auszuführenden Rechenschritte gleich. Nur die Anzahl der Iterationsschritte steigt in Relation zu der Anzahl der zu bewegenden Elemente.
Darüber hinaus ist die einfache Umsetzung von Bewegungsbeschränkungen für die Einbindung der *Cyclic Coordinate Descent Methode* ausschlaggebend. Im Vergleich zur Berechnung aller Winkeländerungen einer Verkettung in Form von Matrixgleichungen, welche erfordern dass die vom System einzuhaltenden Beschränkungen ebenfalls in Form von Matrixsystemen formuliert werden, können bei der *CCD Animationsmethode* die Beschränkungen für jedes Gelenk separat formuliert werden. Darüber hinaus ist es aufgrund der grundlegenden geometrischen Beziehungen, auf welche die Methode aufbaut möglich, jeden Punkt einer Verkettung als Effektor der kinematischen Bewegung zu nutzen.

Die zur Repräsentation der beweglichen Objektteile modellierten Ersatzstrukturen erlauben es dem Nutzer interaktiv ein bewegliches Modell zu definieren und zu animieren, ohne sich mit der Komplexität des zugrunde liegenden Puppenmodells auseinander setzen zu müssen. Um Teile eines Objektes zu bewegen genügt es, den entsprechenden Objektteil über eine virtuelle Box einzugrenzen beziehungsweise eine Skelettstruktur in dem semi-transparenten Modell zu positionieren. Danach muss in einem zweiten Schritt der entsprechende Einflußbereich jedes Elements der Skelettstruktur festgelegt werden, um den Definitionsprozess abzuschließen. Theoretisch kann das Modell bereits nach diesem Schritt animiert werden. Dennoch sollten zur Verfeinerung der Objektbewegungen die im Verlauf dieser Arbeit vorgestellten Beschränkungen und

Bewegungseigenschaften dem gewünschten Verhalten angepasst werden.
Aus dem Einsatz der Ersatzstrukturen resultieren zwei wesentliche Vorteile gegenüber anderen Varianten der Definition bewegliche Objektteile. Bei diesen werden die Gitterpunkte der Objektoberfläche, wie in [WAB+06] dargestellt, direkt modifiziert. Da die Animationsstrukturen nicht direkt mit den Punkten der Objekthülle verbunden sind, sondern als Zwischenschicht das in Abbildung 9.1 dargestellte reduzierte Puppenmodell modifizieren, ist die Auflösung und damit die variierende Gitterstruktur unterschiedlicher Puppenmodelle für die Ersatzstrukturen nicht maßgeblich.
In Hinblick auf die interne Repräsentation eines Puppenmodells passt sich die zur

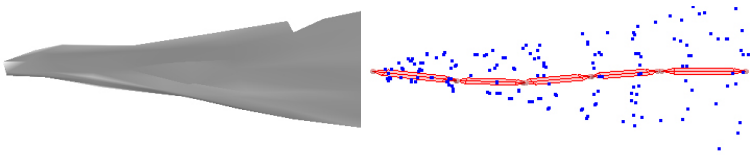

Abbildung 9.1: Darstellung der physikalischen Skelettschicht und der Animationsstruktur.

Bewegung genutzte Struktur dem Schichtenmodell der Objekte an, indem alle benutzergesteuerten Bewegungen der Ersatzstrukturen direkt auf die innerste Objektschicht übertragen werden, welche genau dem in Abbildung 9.1 grau gezeichneten reduzierten Modell entspricht. An dieser Ebene knüpft das zur physikalischen Simulation der Oberfläche genutzte Feder-Masse System an, bei dem jeder Punkt der innersten Schicht über Feder-Masse Beziehungen die aktuelle Position der durch diesen beeinflussten Oberflächenpunkte beschreibt. Durch die Implementierung weiter Schichten ist es möglich, zusätzliche Details wie zum Beispiel Fell zu animieren ohne dass dadurch die zur Bewegung genutzte Struktur beeinflusst wird.

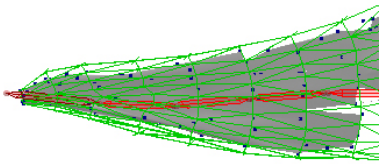

Abbildung 9.2: Darstellung der physikalischen Schichten mit der Animationsstruktur.

Darüber hinaus kann in Ergänzung zur physikalischen Oberflächensimulation, die Oberfläche mit der in [Par08] vorgestellten *Free Form Deformation* an der Außenseite gedehnt und an der Innenseite verdichtet werden. Dadurch werden die Übergänge zwischen den einzelnen beweglichen Objektteilen optisch aufgewertet. In der Abbildung 9.3 ist die *Free Form Deformation* beispielhaft für den zwei dimensionalen Fall aufgezeigt. Den einzelnen Darstellungen ist zu entnehmen, wie sich die Oberfläche in Abhängigkeit von der durchgeführten Bewegung deformiert.

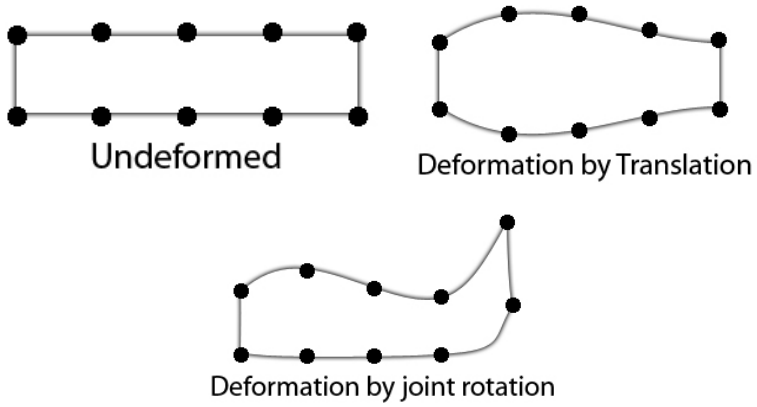

Abbildung 9.3: Schematische 2D Darstellung der auf ein Objekt angewendeten FFD.

9.2 Vor- und Nachteile der Datenhandschuhsteuerung

Die in dieser Arbeit umgesetzte Steuerung mit dem Datenhandschuh erlaubt es, dem Nutzer beliebige selbstdefinierte Verkettungen kinematisch oder aber auch nur einzelne Objektteile zu animieren, indem die Bewegung der Fingerspitzen auf die Bewegung der entsprechenden Elemente übertragen wird. Zum einen ist es dadurch möglich, Effektoren verschiedener kinematischer Verkettungen simultan zu bewegen, indem jede Fingerspitze des Handschuhs genau einen Effektor in seiner Bewegung beeinflusst und zum anderen kann der Handschuh und das damit repräsentierte hierarchische Handmodell genutzt werden, um die Spielweise echter Latexpuppen abstrahiert darzustellen. Letztere Art der Puppensteuerung wird in der Weise umgesetzt, dass ein virtuelles Handmodell abstrahiert wird, welches vom Puppenspieler innerhalb der Puppe zu positionieren ist.
Durch diesen Vorgang ist es möglich, die Puppe in Echtzeit so zu bewegen als ob man mit der Hand in eine echte Latex Puppe greift, um diese zu bewegen. Dafür wird jede Fingerbewegung auf bestimmte Objektbereiche übertragen, die nicht-kinematisch bewegt werden können. Die Bewegungen der auf diese Weise gesteuerten Bereiche wird in Anlehnung an das Handgelenk um einen globalen Rotationspunkt rotiert, welcher in der Szene positioniert werden kann und durch seine Lage die Richtung der Fingervektoren und damit die resultierende Bewegung der Objektteile beeinflusst.

Im Verlauf dieser Arbeit hat sich herausgestellt, dass sich besonders der zweite Ansatz durch die *Hand-in-der-Puppe Metapher*, für die Steuerung durch den Datenhandschuh eignet. Zwar ist es möglich, den Handschuh für die simultane kinematische Animation bis zu fünf verschiedener Endeffektoren zu nutzen, jedoch immer mit einer Einschränkung in Bezug auf die zur Verfügung stehenden Freiheitsgrade. Da der Handschuh an jedem Finger nur ein Freiheitsgrad über den Grad der Fingerbeugung zur Verfügung stellt, können die gesteuerten Elemente nur in einer Dimension unabhängig voneinander bewegt werden. Demzufolge können die Effektoren beispielsweise unabhängig von einander um die Z-Achse rotiert werden. Alle weiteren Bewegungsänderungen wirken sich aufgrund der hierarchischen Beziehungen der Elemente des Handmodells auf alle gesteuerten Elemente gleichermaßen aus.

Ein weiterer Nachteil der virtuellen handgesteuerten Animation ist neben der fehlenden Erkennung der Abduktion und der Adduktion der Finger, die fehlende Übertragung der haptischen Eigenschaften der Puppe. Wird beispielsweise der Mund einer echten Latexpuppe mit der Hand bewegt, so ergibt sich ab einem gewissen Grad der Puppendeformation ein durch das Material bestimmter stark ansteigender Widerstand bis zu dem Punkt an dem der Mund nicht weiter geöffnet werden kann. Diese Eigenschaft kann gegenwärtig nur über die Reduzierung der durch die Fingerbewegung bestimmten Bewegungsvektoren der beweglichen Objektteile simuliert werden, indem mit zunehmender Auslenkung der Dämpfungsfaktor verstärkt wird.

9.3 Ausblick

Das handgestützte Animationsmodell des Echtzeit-Animationssystems bietet genug Ansätze für zukünftige Erweiterungen in Hinblick auf die Steuerung des Animationssystems mittels des Datenhandschuhs. Gegenwärtig werden die beweglichen Teile im Editor mit der Maus erstellt und dann vom Nutzer mit den jeweiligen Fingerspitzen verknüpft.
Daran kann in der Zukunft angeknüpft werden, um den Handschuh schon vor der eigentlichen Animation aktiv in den Editor mit einzubeziehen. Des Weiteren sind durch die virtuell repräsentierte Hand noch nicht alle Steuerungsmöglichkeiten der echten Hand abgedeckt. In diesem Zusammenhang ist es wünschenswert, dass die Adduktion und Abduktion der Finger auf die virtuelle Hand übertragen werden kann.
Derzeit lässt sich dieses Problem nur lösen, indem andere Handschuhtypen zur Steuerung benutzt werden beziehungsweise die Handschuhsteuerung mit speziellen *Capturing Methoden* erweitert wird. In [IVV01] wird ein System zur Fingerspitzenerkennung vorgestellt, bei welchem keine speziellen Marker gebraucht werden, um die Finger zu erkennen.
Darüber hinaus kann die Datenhandschuhsteuerung soweit erweitert werden, dass es möglich wird, dass Animationssystem mit mehreren Handschuhen zu steuern. Die gleichzeitige Steuerung mit verschiedenen Eingabegeräten stellt spezielle Anforderungen an die Echtzeitanimation. Diese Anforderungen beziehen sich speziell auf die Frage der Verarbeitung der simultan gewonnen Sensordaten und werden in [DYP04] genauer erläutert.
Der Vorteil der gleichzeitigen Interaktion mit verschiedenen Handschuhen liegt darin, dass es auf diese Weise möglich ist mehr als eine Puppe in Echtzeit zu animieren. Denkbar ist zum Beispiel der Fall, dass ein Puppenspieler die Mundbewegungen von zwei Handpuppen in Echtzeit animiert und ein Gespräch zwischen beiden spielt.
Dies zeigt, dass es für die künftige Entwicklung des Echtzeit-Animationssystems entscheidend ist, dass die virtuell generierten und animierten Charaktere mit ihrer virtuellen Umwelt interagieren können. Momentan bietet das Animationssystem die nötigen Interaktionsmöglichkeiten um virtuelle Objekte in Echtzeit zu animieren, indem die beweglichen Objektteile bewegt werden.
Der Nachteil ist, dass die Puppenanimation keinen Einfluss auf den Rest der virtuellen Szene hat. Für die Weiterentwicklung der handgesteuerten Puppenanimation ist es demnach wünschenswert, dass man mit selbstdefinierten Bewegungsprozessen virtuelle Gegenstände greifen und benutzen kann. Speziell für die mittels eines Datenhandschuhs ausgeführten

Mundbewegungen ergeben sich gute Interaktionsmöglichkeiten für die Puppen. Denkbar ist zum Beispiel die Möglichkeit, über bestimmte Mundbewegungen virtuelle Seifenblasen wegzupusten oder mit dem Froschmodell in Echtzeit einen virtuellen Keks zu essen.

Literaturverzeichnis

[Bat94] Joseph Bates. The role of emotion - in believable agents. Technical report, Communiction of the ACM, Vol 37, No. 7, 1994.

[BBG85] W. Bunse and A. Bunse-Gerstner. *Numerische lineare Algebra*. Teubner Studienbücher, 1985.

[DQ04] Haixia Du and Hong Qin. Medial axis extraction and shape manipulation of solid objects using parabolic pdes. Technical report, Computer Science Department, Stony Brook University, 2004. ACM Symposium on Solid Modeling and Applications (2004).

[DYP04] Mira Dontcheva, Gary Yngve, and Zoran Popovic. Layered acting for character animation. Technical report, University of Washington, 2004.

[Ebe06] David H. Eberly. *A Practical Approach to Real-Time Computer Graphics*. Morgan Kaufmann (Interactive 3D Technology, second edition edition, 2006.

[ESHD05] Erleben, Sporring, Henriksen, and Dohlmann. *Physic-Based Animation*. Charles River Media Inc., 2005.

[Fei01] K. Fei. Expressive textures. Technical report, Computer Science Department, University of Pretoria, South Africa, 2001. AFRIGRAPH 2001 Capctown South Africa.

[Glo] P5 Data Glove. P5 data glove - essential reality inc. Essential Reality Inc. is owned by Alliance Distributors Holding Inc.

[HCBM07] Sunil Hadap, Marie-Paule Cani, Florence Bertails, and Steve Marschner. Strands and hair - modeling, animation, and rendering. Technical report, SIGGRAPH 2007 Course Notes, 2007. (Course No.33), May 2, 2007, Course Organizer: Sunil Hadap.

[IVV01] Giancarlo Iannizzotto, Massimo Villari, and Lorenzo Vita. Hand tracking for human-computer interaction with graylevel visualglove: Turning back to the simple way. Technical report, Department of Mathematics, University of Messina, Italy, 2001. PUI 2001 Orlando FL, USA.

[KC02] Young-Min Kang and Hwan-Gue Cho. Complex deformable objects in virtual reality. Technical report, GALab, Department of Computer Science, Pusan National University KOREA, 2002. VRST'02 November 11–13, 2002, Hong Kong.

[KJZM04] Mykhaylo Kostandov, Radu Jianu, Wenjin Zhou, and Tomer Moscovich. Interactive layered character animation in immersive virtual environments. Technical report, Brown University, Providence, RI, 2004.

[KTT07] Manolya Kavakli, Meredith Taylor, and Antoly Trapetznikov. Designing in virtual reality (desire): A gesture-based interface. Technical report, DIEMA'07 Perth, Western Australia, 2007.

[Leh83] E. Lehmann. *Lineare Algebra mit dem Computer*. B. G. Teubner Stuttgart, 1983.

[LRBW04] A. Bryan Loyall, W. Scott Neal Reilly, Joseph Bates, and Peter Weyhrauch. System for authoring highly interactive, personal-rich interactive characters. Technical report, Zoesis Studios, Newtonville, MA, USA, 2004. Eurographics/ACM SIGGRAPH Symposium on Computer Animation.

[MH06] Tomer Moscovich and John F. Hughes. Multi-finger cursor techniques. Technical report, Department of Computer Science, Brown University, Providence RI, USA, 2006. Graphics Interface 2006.

[MTPS08] Johannes Mezger, Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. Interactive physically-based shape editing. Technical report, WSI/GRIS, Universität Tübingen, Germany, 2008.

[Par08] Rick Parent. *Computer Animation Algorithms and Techniques*. Morgan Kaufmann, second edition edition, 2008.

[PW07] M. Pfaff and C. A. Wüthrich. Medial surface-based real time simulation of elastic objects. Technical report, 4t h Workshop in Virtual Reality Interactions and Physical Simulation, VRIPHYS (2007), 2007.

[Rey87] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. Technical report, Symbolics Graphics Division, CA, USA, 1987. Computer Graphics, Volume 21, Number 4, July 1987.

[Tin] TinyXML. *TinyXML*. http://www.grinninglizard.com/tinyxmldocs/index.html.

[TPS08] Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. Asynchronous cloth simulation. Technical report, WSI/GRIS, Universität Tübingen, Germany, 2008. CGI 2008 Conference Proceedings.

[WAB+06] C. A. Wüthrich, Jing Augusto, Sven Banisch, Gordon Wetzstein, Przemyslaw Musialski, and Chrystoph Toll. Real time simulation of elastic latex hand puppets. Technical report, WSCG'2006, January 30 - February 3, 2006, Plzen, Czech, 2006.

[WC91] L.C.T. Wang and C. C. Chen. A combined optimization method for solving inverse kinematics problem of mechanical manipulators. Technical report, IEEE Transactions on Robotics and Automation, 1991. Vol.7, No.4, pp. 489-499.

[WCL04] T. Komura W.-C. Lam, F. Zou. Motion editing with data glove. Technical report, ACM Transactions on Graphics, 2004.

[Wel93] Chris Welman. Inverse kinematics and geometric contraints for articulated figure manipulation. Master's thesis, Simon Fraser University, 1993.