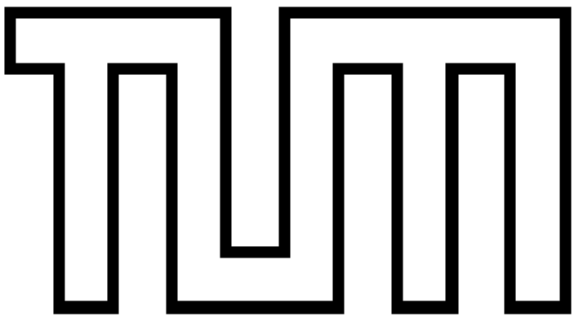

Technische Universität München

Fakultät für Informatik

Forschungs- und Lehreinheit Informatik IX

Systementwicklungsprojekt

Monoculare Verfolgung eines dreidimensionalen Gesichtsmodells

Christian Schmidts

Aufgabensteller: Univ.-Prof. Dr. Bernd Radig

Projektbetreuer: Dipl.-Inf. Univ. Matthias Wimmer

Garching,

INHALTSVERZEICHNIS

1. Motivation

Dieses Systementwicklungsprojekt befasst sich mit der Verfolgung von menschlichen Gesichtern. Die spezielle Herausforderung ist, dass das jeweilige Gesicht mittels einer Kamera im Dreidimensionalen verfolgt werden soll. Es soll die relative dreidimensionale Position des Gesichtes zur Kamera ermittelt werden. Dieses Ziel soll ohne Zuhilfenahme einer weiteren Kamera oder anderen Hilfsmitteln erfüllt werden. Zunächst geschieht die Verfolgung durch bereits implementierte Algorithmen im Zweidimensionalen. Dieses Projekt beschäftigt sich mit der Weiterverarbeitung der zuvor aus der 2D-Bildebene gewonnen zweidimensionalen Position des Gesichtes und berechnet mit Hilfe der hier entwickelten Algorithmen aus diesen Informationen eine entsprechende Positionierung und Orientierung des Gesichtes in unserer dreidimensionalen Welt.

1.1. Anwendungsgebiete

Die Erkennung von Menschen und die Interpretation menschlicher Aktionen gewinnt bei der Interaktion von Mensch und Maschine immer mehr an Bedeutung. Als Beispiele sei hier auf Entwicklungen in der Robotik, der elektronischen Sicherheitsüberwachung und der Filmindustrie verwiesen. Beispielsweise ist schon Heute eine grundlegende Interaktion zwischen Menschen und Robotern möglich. Es existieren bereits diverse Roboter, die auf natürliche menschliche Anweisungen die über Kameras erfasst werden reagieren. Auch in der Unterhaltungsindustrie gewinnt zuverlässige Interpretation von Bilddaten durch Computer immer mehr an Bedeutung. So konnten bereits in Filmen wie Chuck Russels „Die Maske“ Ende der 90er Jahre erste eindrucksvolle Effekte mittels Verfolgung menschlicher Körper und Gesichter ermöglicht werden. Bei diesem Film wurde das Gesicht des Hauptdarstellers durch eine computeranimierte Maske ersetzt. Aus eigener Erfahrung kann davon ausgegangen werden, dass ca. neunzig Prozent aller Szenen, die später mit visuellen 3D-Effekten versehen werden sollen, zunächst von einer Software zur zwei- oder dreidimensionalen Bildverfolgung behandelt werden. Die zuvor schon erwähnten Sicherheitssysteme (siehe Abbildung 1-1), die mit biometrischen Daten arbeiten, gewinnen in Zeiten der zunehmenden

Globalisierung eine immer größere Rolle. Beispielsweise werden sicherheitsrelevante Passagen wie Flughäfen oder Eingänge zu Fußballstadien mittels Sicherheitsanlagen, die mit Kameras ausgestattet sind, überwacht. Die so gewonnenen Daten können dann mit entsprechenden Datenbanken verglichen werden, was durch rein menschliche Überwachung niemals so effizient zu bewerkstelligen wäre.

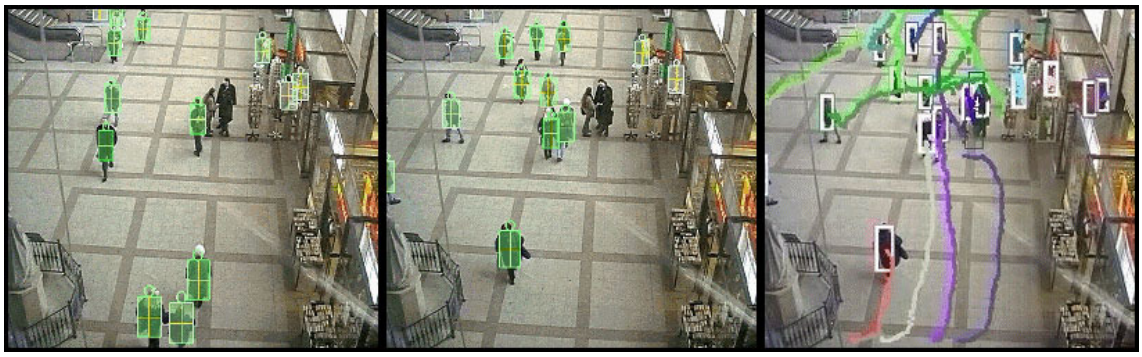

Abbildung 1-1: Tracking als Unterstützung der Sicherheitsüberwachung

Ein konkretes Beispiel hierzu ist, dass eine Person mit einem Koffer an einem Flughafen von Überwachungskameras mit einem Verfolgungsalgorithmus erfasst und verfolgt wird. Stellt diese Person zum Beispiel den Koffer auf dem Gelände ab und entfernt sich von diesem ohne wiederzukehren wird automatisch das Wachpersonal benachrichtig.

Die Interaktion zwischen Mensch und Maschine stellt eine besondere Herausforderung dar, da die Aktionen des Menschen in einer dem Computer völlig fremdem Umgebung, der echten Welt, stattfinden und somit erst eine geeignete Dateneingabe und eine Transformation dieser Daten stattfinden muss. Ziel hierbei ist es unter Anderem, den technischen Aufwand an Hardware so weit wie möglich zu begrenzen und dabei ein best mögliches Ergebnis bezüglich Geschwindigkeit und Genauigkeit der Datenerfassung und -auswertung zu erzielen. Aus diesem Grund befasst sich dieses Projekt nicht rein mit der Aufgabenstellung der Rücktransformation von 2D-Koordinaten in die zugrunde liegenden 3D-Koordinaten. Dieses Problem wurde bereits von vielen anderen Wissenschaftlern erfolgreich gelöst, allerdings oft unter Zuhilfenahme zweier Kameras wie in Abbildung 1-2 oder mit anderen zusätzlichen Hilfsmitteln. Die Zielsetzung dieses Projektes hingegen ist es, diese Positionierung im Dreidimensionalen mit einfachsten Hilfsmitteln, also nur einer Kamera und einem einzelnen Rechner, zu ermöglichen.

Abbildung 1-2: 3D-Face-Tracking mit deformierbarer Maske bei [5]

Bei [6] wird wie in Abbildung 1-2 dargestellt ein dreidimensionales Gesichtmodell auf das Gesicht von Menschen in einem Videostrom gelegt. Bei [6] wird die Maske mit der Veränderung des Gesichtes deformieret und so angepasst, dass sie bestmöglich auf das jeweilige Gesicht gelegt wird. Der Schwerpunkt hierbei liegt in der Aussortierung fehlerhafter Daten, die durch Störungen im Bild oder sonstige bei der Gesichtsverfolgung entstehende Fehler auftreten können. Das Eliminieren von fehlerhaften Daten geschieht mit der Absicht ein möglichst robustes Modell zu entwerfen, das in vielen verschiedenen Situationen und auch bei schlechtem Bildmaterial eingesetzt werden kann. Die Zuhilfenahme der in [6] vorgestellten Algorithmen ist in Betracht zu ziehen wenn die Methoden dieses Projektes erweitert und ausgebaut werden.

1.2. Framework

Die Umsetzung dieses SEP findet unter bestimmten Prämissen statt. Das SEP selbst soll ein Teilmodul eines gesamten Frameworks darstellen. Das Rahmenprojekt, wie in Abbildung 1-3 dargestellt, befasst sich mit der Schaffung eines „Intelligenten Büros“, das mit den Nutzern über Kameras interagiert. Außerdem wurden schon zahlreiche Algorithmen und andere Teilmodule zur Erkennung und Verfolgung von Menschen und derer Aktionen in das Programm implementiert. Somit sind die Vorraussetzung für das Verfolgen im Zweidimensionalen schon geschaffen, da der zu implementierende Algorithmus auf schon vorhandene Positionsangaben auf der Bildebene zurückgreift. Das SEP selbst beschränkt sich auf den Algorithmus zur Berechnung der 3D-Position des Kopfes aus den im Framework schon vorhandenen 2D-Daten.

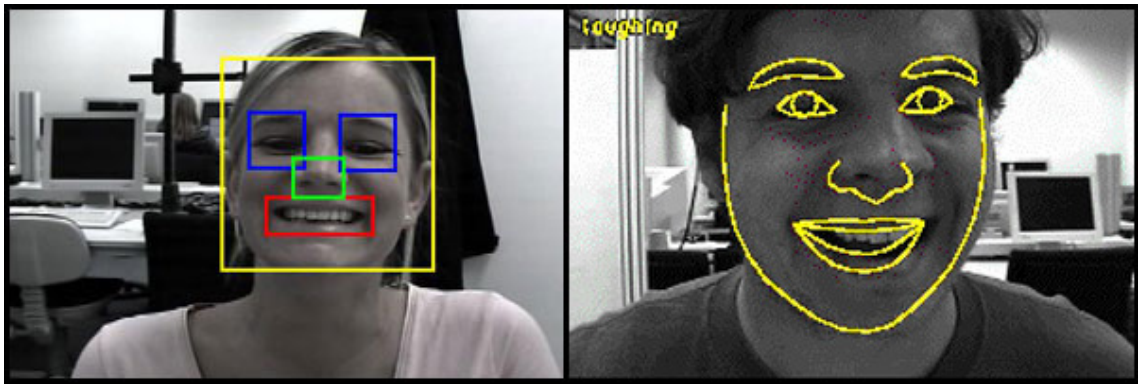

Abbildung 1-3: Das sHome Framework mit 2D-Tracking und Mimik-Erkennung

In diesem Fall sind die entsprechenden Daten sehr umfangreich. Es können beispielsweise Kopfbreite, Gesichtsumriss, einzelne zweidimensionale Punkte und sogar die Mimik ausgelesen werden. Es wird auf lange Sicht ein flexibles dreidimensionales Gesichtsmodell erarbeitet, das auf fundierte Daten aus den Algorithmen der zweidimensionalen Gesichtsverfolgung zurückgreift. Der erste Ansatz beschränkt sich allerdings, auf die Gewinnung der 3D-Pose, also Position und Orientierung, so dass eine Positionierung des Gesichtsmodells im dreidimensionalen Raum möglich wird.

2. Grundlagen

Viele echtzeitfähige Algorithmen arbeiten mit zusätzlichen Hilfsmitteln, die hier explizit nicht verwendet werden sollen, da es schon eine Vielzahl von Algorithmen gibt, die sich mit dem Tracking mittels zweier Kameras befassen. Dabei werden durch die bekannte relative Positionierung zwischen den beiden Kameras und der optischen Schnittpunktsuche im Bild die dreidimensionalen Positionsdaten reproduziert, ähnlich wie es das menschliche Gehirn mit Hilfe unserer beiden Augen bewerkstelligt. Beim Stereosehen ist die Distanz und der Winkel zwischen den Kameras bekannt wodurch im Idealfall eine exakte Berechnung der absoluten Position im dreidimensionalen Raum ermöglicht wird. Da die Algorithmen dieses Projektes allerdings auf eine Kamera verzichten müssen, stellt sich zunächst ein grundlegendes Problem: Wie ist es ohne Stereo-Sehen und weitere Hilfsmittel möglich, die Tiefeninformation zu ermitteln? Die triviale und zugleich enttäuschende Antwort lautet: „gar nicht“. Durch Verwendung einer einzelnen Kamera kann keine exakte Bestimmung der Tiefenposition vorgenommen werden. Deutlich wird das bei Betrachtung zweier Punkte, die entlang der Sichtachse einer Kamera liegen. Da der näher an der Kamera liegende Punkt den weiter entfernten sozusagen verdeckt, erscheinen beide als ein einziger Punkt. Betrachtet man die gleiche Szene mit einer leicht versetzt aufgestellten zweiten Kamera, so wird die unterschiedliche Position in der Tiefe deutlich wie in Abbildung 2-1 dargestellt.

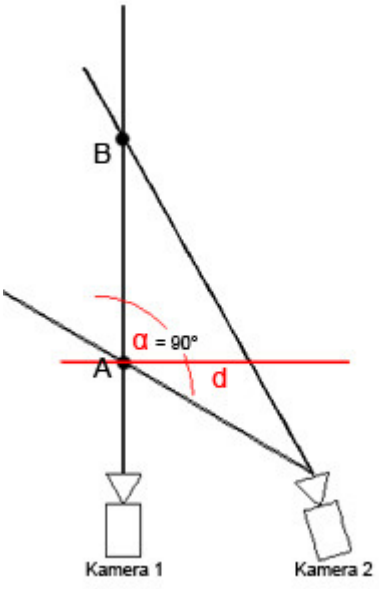

Abbildung 2-1: Fehlen der Tiefeninformation bei nur einer Kamera

Die Punkte A und B liegen unterschiedlich weit von Kamera 1 entfernt, sind aber für diese nicht als in der Tiefe verschieden erkennbar. Die unterschiedliche Entfernung wird erst durch die Distanz d bei Kamera 2 erkennbar, die die Entfernung zwischen A und B auf der Bildebene dieser Kamera widerspiegelt. Kennt man nun noch d in der Größe der realen Welt so lässt sich mittels einfacher Geometrie die absolute Tiefe von A und B ermitteln. Genaueres hierzu findet sich im Abschnitt 2.2.

Allerdings ist diese absolute Tiefenmessung für dieses Projekt nicht zwingend nötig, denn durch Hinzufügen von Entfernungskonstanten kann eine einfache Kalibrierung vorgenommen werden, so dass die absolute Position aus der relativen Position ermittelt werden kann. Dies schränkt zwar die Flexibilität der Anwendung etwas ein, ist aber hinreichend effizient und die Kalibrierung ist leicht umsetzbar. Zunächst gilt es allerdings, einen geeigneten Ansatz zur Berechnung der Tiefeninformation unter den gegebenen Umständen zu finden. Es muss also evaluiert werden, welche Ansätze zur Verfügung stehen und welche von diesen unter den oben genannten Bedingungen adäquat funktionieren. Im den folgenden Unterkapiteln wird nun zunächst auf einige grundlegenden Begriffe und Techniken der Bildverfolgung eingegangen.

2.1. Modelle, Fitting und Tracking

Zu den grundlegenden, benötigten Algorithmen gehören zunächst die Verfahren des 2D-Tracking, da ohne diese keine adäquaten Eingangsdaten zur Verfügung stehen würden. Eine interessante Definition von Tracking findet sich bei Lepetit und Fua. Diese schreiben: *„Tracking an object in a video sequence means continuously identifying its location when either the object or the camera are moving. There are a varity of approaches, depending on the type of object, the degrees of freedom of the object and the camera, and the target application."* [3] Hieraus wird ersichtlich, dass es sich beim Tracking um einen relativ weit gefassten Begriff handelt. Dieser Prozess des Trackings lässt sich noch weiter abgrenzen und genauer definieren denn „*2D tracking typically aims at following the image projection of objects or parts of objects whose 3D displacement results in a motion that can be modeled as a 2D transformation. An adaptive model is then required to handle appearance changes due to perspective effects or to deformation.*" [3] Um also ein Objekt effektiv mittels

Tracking verfolgen zu können, benötigt man so genannte Modelle. Diese stellen eine Grundstruktur oder -information über das jeweilige Objekt dar, die das Objekt im Computer repräsentieren, um den Gegenstand in jedem neuen Bild wieder zu finden. Dazu werden meist 2D deformierbare Grundformen, 2D deformierbare Gitternetze oder 2D Mimik Modelle verwendet, wie die Autoren beschreiben. Dabei handelt es sich um Datensätze aus vordefinierten oder auch erlernten Zuständen die klassifiziert und abgespeichert werden. Wichtig in diesem Zusammenhang sind auch die Algorithmen zum Fitting. Durch das Fitting können Algorithmen, die eine Ähnlichkeit dieser gespeicherten Zustände zu den aktuell gesammelten Informationen finden, beispielsweise die aktuellen Gesichtsausdrücke einordnen. Da das reale Model nicht exakt den Vorgaben der gespeicherten Zustände entspricht muss der Computer selbst mittels geeigneter Algorithmen Anpassungen an den Modellen vornehmen um Realität und virtuelles Modell zur Deckung bringen zu können. Schlagen diese Algorithmen zum Fitting fehl, so verliert der Rechner das zu beobachtende Objekt, muss sich neu anpassen und auf die neue, vollkommen veränderte Situation ausrichten. Lepetit und Fua schreiben weiter: „*However, none of these methods involves recovering the actual position in space. By contrast, 3D tracking aims at continuously recovering all six degrees of freedom that define the camera position and orientation relative to the scene, or, equivalently, the 3D displacement of an object relative to the camera.*" [3] Das Verfahren des 3D-Tracking geht also über die Ansätze des 2D-Trackings hinaus, baut aber zugleich auf dessen Daten, auf um daraus einen Rückschluss auf die ursprüngliche relative Lage der Kamera und speziell der Tiefeninformation der Objekten der Szene ziehen zu können.

2.2. Tiefengewinnung

Es existiert eine Vielzahl von Verfahren zur Gewinnung der Tiefeninformation und deren Verwendung in der Informatik. Allerdings beruhen all diese unterschiedlichen Verfahren auf Grundprinzipien die sich in nur wenige Kategorien zusammenfassen lassen. Denn „*vier grundlegende Prinzipien können unterschieden werden, die wir als Tiefe aus Paradigmen bezeichnen. Zusätzlich kann die tiefe aus der Neigung von Oberflächen bestimmt werden mit einem Paradigma, das unter dem Namen Gestalt aus Schattierung bekannt ist*." [1] Die Kategorisierung mag aber zunächst nicht immer so einfach fallen,

denn „*auf den ersten Blick erscheinen diese Verfahren so verschieden, dass es nicht sofort auffällt, dass sie alle auf dem selben Prinzip beruhen“* [1]. Nach Jähne lassen sich also alle in der Informatik verwandten Verfahren einer oder mehreren der folgenden Kategorien zuordnen:

- Tiefe aus Laufzeit (Depth from runtime)
- Tiefe aus Phase (Depth from phase)
- Gestalt aus Schattierung (Depth from shading)
- Tiefe aus mehreren Projektionen (Depth by multiple projections)
- Tiefe aus Triangulation (Depth by triangulation)

Das wohl einfachste Prinzip ist die Tiefengewinnung aus der Laufzeit eines Signals. Hierbei wird ein Signal ausgesandt, vom Objekt reflektiert und von einer Kamera oder einem Sensor empfangen. Durch die Laufzeit des Signals und dessen Ausbreitungsgeschwindigkeit lässt sich somit eine eindeutige Tiefeninformation gewinnen. Dies kann beispielsweise durch Laser- oder Ultraschallemitter und –empfänger erreicht werden. Bei der Methode der „Tiefe aus Phase“ oder des Verfahrens der „Tiefe aus mehreren Projektionen“ wird beispielsweise mit Hilfe der Laufzeit und der Phasenverschiebung eines Signals oder der verschiedenartigen Darstellung unter mehreren Projektionen dasselbe Ziel erreicht. Einen Spezialfall hierbei stellt das Verfahren der Gestalt aus Schattierung dar, da es sich prinzipiell und physikalisch gesehen in die Methode der „Tiefe aus Phase“ einordnen lässt aber in seiner Anwendung relativ beschränkt ist. Allerdings wird hierfür keine Spezialsensorik verwendet, die die Wellenlängen ermittelt, sondern vielmehr die Farbverläufe des Bildes einer Kamera dazu verwendet die Tiefe eines Punktes zu berechnen. Hierbei wird sich der Effekt zu nutze gemacht, das sich durch die Neigung einer Fläche zu einer Lichtquelle deren Schattierung verändert, woraus Schlüsse auf die relative Tiefe gezogen werden können.
Grundsätzlich erfolgt die Tiefengewinnung bei Stereokameras meist mittels Triangulation. Triangulation wird nach Jähne wie folgt definiert:
„Wenn wir ein Objekt von zwei Positionen aus betrachten, die durch eine Basislinie b voneinander getrennt sind, so erscheint es unter verschiedenen Blickwinkeln. Diese Technik heißt Triangulation und stellt eine der wesentlichen

Techniken in der Geodäsie und Kartografie dar. Die Triangulationstechnik ist die Grundlage einer Vielfalt von Verfahren.“ [1]

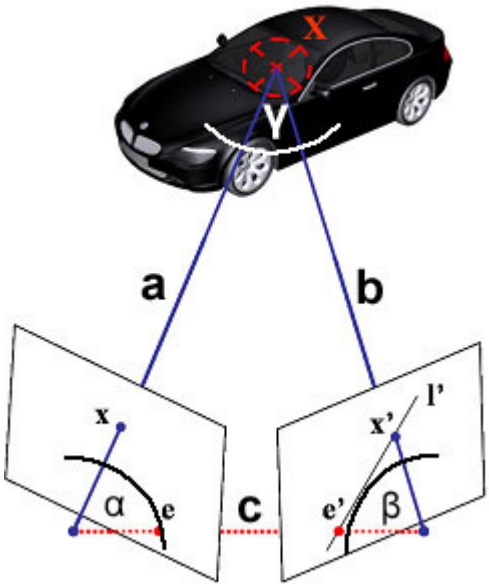

Abbildung 2-2: Triangulation zur Tiefengewinnung

Abbildung 2-2 zeigt zwei Bildebenen durch die jeweils ein einzelner Sichtstrahl verläuft. Die zuvor erwähnte „Basislinie b“ ist hierbei allerdings nicht mit „b“ bezeichnet sondern würde Lotrecht zur Mitte von „c“ verlaufen und wurde der Übersichtlichkeit halber hier nicht eingezeichnet. Beide Sichtstrahlen haben einen fest definierten Ursprung im Kamerasensor wobei hier die Distanz c und die Winkel α und β bekannt sind. Die fehlenden Größen werden nun wie folgt berechnet: (vgl. [2])

- Berechnung von γ: $\gamma = 180° - \alpha - \beta$
- Berechnung der Länge von a: $a = c * \sin(\beta) / \sin(\gamma)$
- Berechnung der Länge von b: $b = c * \sin(\alpha) / \sin(\gamma)$

Wie bereits erwähnt findet dieses Verfahren allerdings in diesem SEP keine direkte Anwendung da es sich ja auf die Daten zweier Kameras stützt. Es wird aber an dieser Stelle dennoch so genau vorgestellt da es wichtige Erkenntnisse für das weitere Vorgehen und die Findung eines geeigneten Algorithmus in diesem Projekt in sich birgt.

2.3. Kalibrierung

Ein weiteres Thema, das wichtig für besonders exakte Trackingverfahren ist, ist die richtige Kalibrierung der Kamera. Zunächst sei erwähnt dass auch die Kalibrierung der Kamera in diesem SEP nicht implementiert wurde, aber einen enormen Zugewinn an Genauigkeit mit sich bringen würde. Generell geht es bei der Kalibrierung der Kamera darum, das simple Modell der Lochkamera das bei dem größten Teil der Algorithmen verwendet wird zu verbessern. Da beispielsweise die Linsen der Kameras eine Verzerrung mit in das gelieferte Bild einbringen ist es nötig dieses Bild zunächst zu entzerren um in den nachgelagerten Algorithmen ein akkurates Ergebnis zu erzielen.

Bei den derzeitig verwendeten Algorithmen kann zwischen zwei Arten der Kalibrierung unterschieden werden. Die erste Art bedient sich bekannter Größen in der Realwelt und der Zuhilfenahme von Kalibrierungsobjekten wie beispielsweise Kalibrierungstafeln etc.. Die zweite Art ermöglicht es den Computern sich selbst zu kalibrieren und dies ohne Unterstützung durch vorgegebene Objekte und deren Dimensionen. Allerdings sind erstere Verfahren wesentlich genauer, da sie wie schon erwähnt auf bekannt Größen zurückgreifen können, weswegen im Folgenden nur auf diese Art der Ausrichtung eingegangen wird. Die durch die Kalibrierung erhaltenen Parameter können auch in zwei Arten differenziert werden. Es handelt sich dabei um so genannte interne und externe Parameter. Die internen Parameter definieren die Bildkoordinaten der Kamera in Referenz zu den Bildkoordinaten der echten Welt wobei die externen Parameter den Standort und die Ausrichtung der Kamera in der Realwelt darstellen. Durch Multiplikation mit der so erhaltenen Matrix A (die die 5 internen Parameter beinhaltet) können so auf direktem Wege die Echtweltkoordinaten relativ exakt aus den Bildkoordinaten berechnet werden (vgl. Formel 2-3) aus [4]. R und t geben in diesem System die Rotation bzw. Translation an.

$$\mathbf{A}\,[\mathbf{R}\quad \mathbf{t}]\begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}[\mathbf{R}\quad \mathbf{t}]\begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix}$$

Formel 2-3: Verwendung der internen Parameter aus [4]

In der Matrix A stellen u0 und v0 die Koordinaten des Mittelpunktes der Kameralinse dar. Die Variablen α und β sind hierbei die horizontale und vertikale Brennweite und γ die Stauchung des Bildes. Bezieht man diese Daten mit in die Berechnung der ursprünglichen 3D-Koordinaten der Punkte ein, so bekommt man ein wesentlich exakteres Endergebnis. Dies ist vor allem im Hinblick auf selbstverstärkende Fehler in der Berechnung ganzer Bildsequenzen nicht zu vernachlässigen. Um eine erhebliche Verbesserung in den Algorithmen dieses SEPs zu erzielen, könnte zur Verbesserung eine Kalibrierung der Kamera vorgenommen werden und deren Daten dann mit in die Berechnung der 3D-Punkte mit einbezogen werden.

2.4. Wichtige Formeln

Zur Umsetzung der Algorithmen die in diesem Projekt verwendet werden sind einige Grundlegende mathematische Formeln notwendig, die an dieser Stelle kurz vorgestellt werden. Von Belang sind hier vor allem die Formeln zur perspektivischen Projektion und Transformationen im dreidimensionalen Raum.

2.4.1. Zentralprojektion

Der in Formel 2-3 aufgezeigte Ausdruck beinhaltet unter anderem die Zentrale Formel für die Umrechnung von Punktkoordinaten im Dreidimensionalen zum Zweidimensionalen und umgekehrt. Gemeint ist hier die Zentralprojektion die dem angenommenen Lochkameramodell zugrunde liegt wie in Abbildung 2-5 zu sehen ist. Dabei wird durch Projektion der Punkte im Dreidimensionalen auf eine Bildebene ein zweidimensionales Abbild der Objekte des dreidimensionalen Raumes erstellt. Das Kameramodel ist hierbei durch seinen Ursprung und das Kameravolumen festgelegt. Ein Punkt A wird dann auf seinen Bildpunkt A' projiziert indem er durch eine Gerade mit dem Ursprung der Kamera verbunden wird. Der Schnittpunkt mit der Bildebene stellt dann den Bildpunkt A' dar. Mathematisch abgebildet wird dieser Vorgang durch die Formel 2-4.

$$\frac{x_i}{f} = \frac{X_i}{Z_i} \quad \frac{y_i}{f} = \frac{Y_i}{Z_i}$$

Formel 2-4: Formel zur perspektivischen Projektion

X, Y und Z stellen hierbei die Koordinaten eines Punktes im Dreidimensionalen dar wobei x und y die zweidimensionalen Koordinaten auf der Bildebene sind und der Bezugspunkt immer die Optische Achse der Kamera ist. Der Parameter f repräsentiert hier die Brennweite der Kamera.

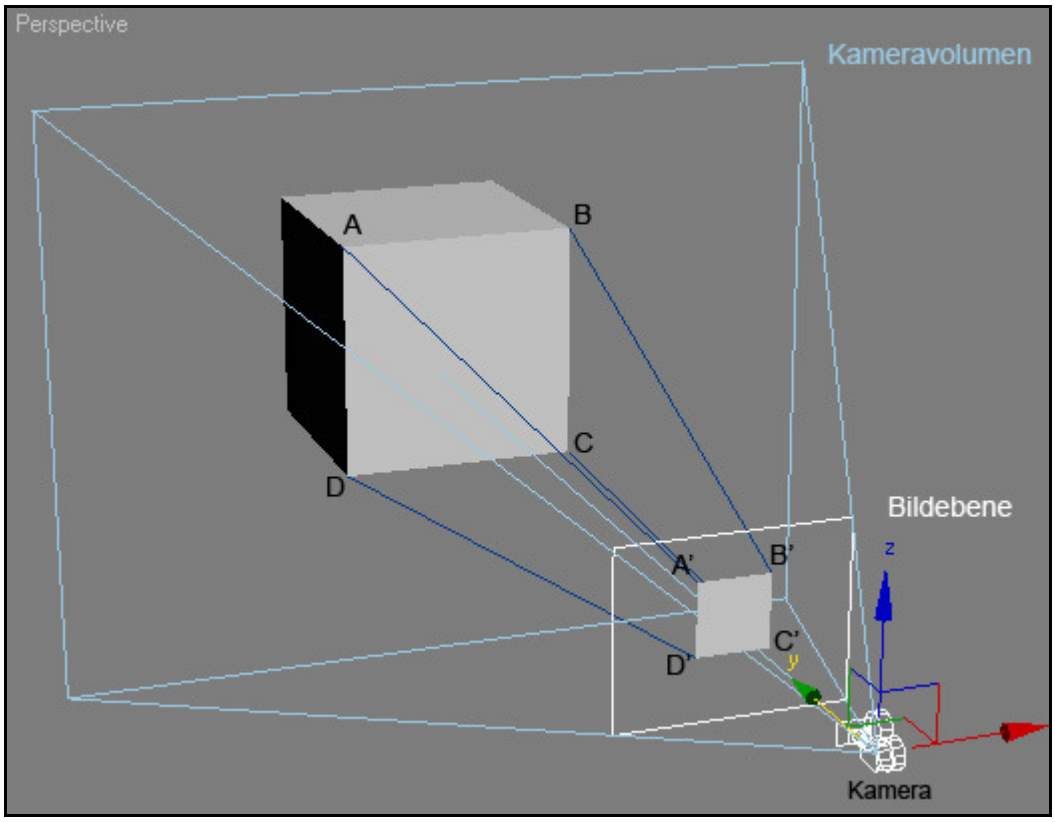

Abbildung 2-5 Perspektivische Projektion

2.4.2. Rotation und Translation im Dreidimensionalen

Eine weitere wichtige Erkenntnis ist, dass sich die Änderung der Position eines Körpers durch die Änderung einzelner Punkte auf dem Körper darstellen lässt. Wird nun die Lage des Körpers im Dreidimensionalen verändert so lassen sich die Änderungen der Position einzelner Punkte auf dem Körper durch Formel 2-6 repräsentieren.

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = \mathrm{R} \cdot \begin{bmatrix} X_i' \\ Y_i' \\ Z_i' \end{bmatrix} + T$$

Formel 2-6: Rotation und Translation von Punkten im Dreidimensionalen

R und T stellen hierbei jeweils die Rotationsmatrix und den Translationsvector dar. X, Y, Z sind die Koordinaten des Punktes nach der Änderung der Position und X',Y' und Z' die Koordinaten davor. Die Rotation erfolgt durch Multiplikation des als Vektor dargestellten Punktes mit der Rotationsmatrix für jede Koordinatenachse separat. Dies wird durch die einzelnen Rotationsmatrizen in Formel 2-7 ausgedrückt.

$$RotX = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \quad RotY = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \quad RotZ = \begin{pmatrix} \cos\delta & -\sin\delta & 0 \\ \sin\delta & \cos\delta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Formel 2-7: Rotationsmatrizen für den dreidimensionalen Raum

Will man nun einen Körper rotieren, so muss man alle seine Punkte rotieren. Dazu ist es aber zunächst notwendig ein geeignetes Rotationszentrum zu finden, da die in Formel 2-7 dargestellten Matrizen eine Rotation um den jeweiligen Winkel im Ursprung im R^3 darstellen. Diese Tatsache stellt im weiteren Projektverlauf eine wichtige Entscheidungsfrage dar, denn das Rotationszentrum ist in dem zu behandelnden Fall keineswegs eindeutig. Um die Translation durchzuführen werden die einzelnen Komponenten des Translationsvectors mit den Komponenten des als Vector dargestellten Punktes addiert.

2.5. Optimierungsalgorithmen

Bei dem hier vorgestellten Verfahren zum relativen dreidimensionalen Tracking stellt sich das grundlegende Problem der Optimierung. Da die Berechnung Veränderung der Punktewolke, die ein Objekt darstellt, nicht auf direktem Wege lösbar ist, muss hier zu Algorithmen gegriffen werden, die die benötigten Parameter zur Rotation und Translation sukzessive annähern. Das Problem stellt sich wie folgt dar. Gegeben sind zwei korrespondierende Mengen P und P' von Punkten, die im Idealfall eine bijektive Abbildung darstellen, so dass also jeder Punkt eindeutig einem Vorgänger zugeordnet werden kann. Diese Punktemengen stellen Punkte des im Zweidimensionalen über einen Zeitraum t verfolgten Objektes dar. Die erste Punktemenge P' entspricht den zu Beginn von t ermittelten Punkten währen die Menge P die, relativ zum Objekt, selben Punkte darstellt, die aber im Dreidimensionalen einer Translation und Rotation unterzogen wurden. Um nun die Parameter dieser Transformation zu ermitteln gilt es diese zunächst zu schätzen und dann schrittweise bestmöglich anzunähern, bis P durch Anwendung der Transformation mit P' übereinstimmt.

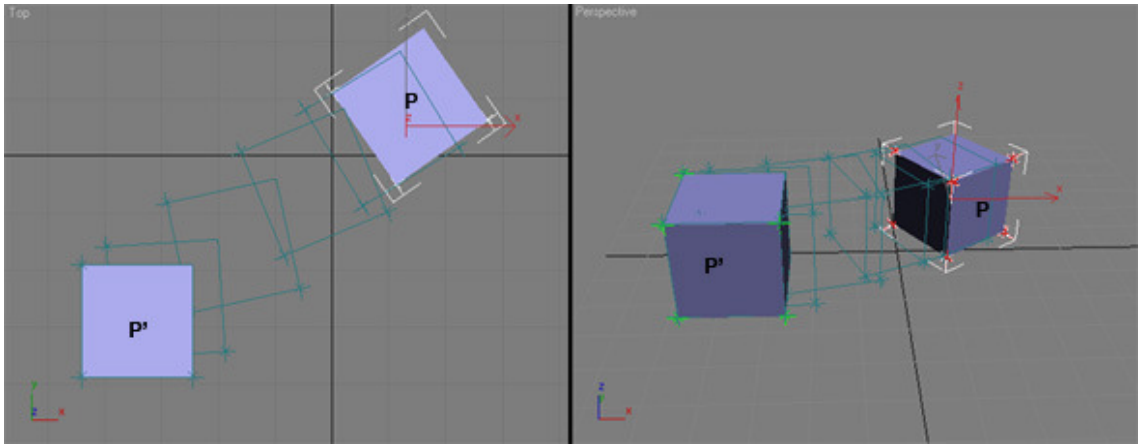

Abbildung 2-8: Veränderung der Punktewolke durch Transformationen

In Abbildung 2-8 wird dieser Vorgang durch die in Türkis dargestellten Schlüsselpositionen veranschaulicht. Die Punkte in P' sind hierbei Grün markiert währen die korrespondierenden Punkte P in Rot dargestellt werden.

2.5.1. Gradientenabstiegsverfahren

Das Gradientenabstiegsverfahren stellt ein numerisches Verfahren zur Lösung nichtlinearer Optimierungsprobleme dar. Die Vorgehensweise bei diesem Verfahren ist es, das Minimum einer Funktion durch Zuhilfenahme der Gradienten an aufeinander folgenden iterativ bestimmten Stellen der Funktion zu bestimmen. Die jeweilige folgende und näher am Minimum liegende Stelle wird also durch Addition auf die vorhergehend gefundene Stelle, mit einer vorgegebenen Schrittweite in Richtung des Gradienten gefunden. Mathematisch dargestellt ist dies in Formel 2-9.

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Formel 2-9: Gleichung zum Gradientenabstiegsverfahren nach [7]

Durch Subtraktion des mit der Schrittweite α_k versehenen Gradienten $\nabla f(x_k)$ von der zuvor gefundenen Stelle x_k der Funktion erhält man hier die neue Stelle x_{k+1} die im Allgemeinen näher am Minimum der Funktion liegt. Allerdings birgt dieses Verfahren folgende Probleme in sich (vgl. [8][9]):

- Das Gradientenabstiegsverfahren findet nur lokale Minima, d.h. das unter Umständen das Endergebnis der Iteration nicht das gesuchte globale Minimum zurückgeliefert wird sondern lediglich das nächstliegende lokale Minimum.

- Ein weiteres Problem tritt auf wenn die zu iterierende Funktion über sehr flache Stellen, so genannte Plateaus verfügt. An solchen Stellen kommt es während der Optimierung zur Stagnation, so dass keine effiziente Auswertung mehr möglich ist.

- Da der Verlauf der Iteration stark von der jeweiligen Schrittweite abhängt kann es vorkommen, dass ein globales Minimum aufgrund seiner ungünstigen Beschaffenheit übersprungen wird. Ist das globale Minimum auf einen sehr kleinen Raum in der Funktion begrenzt, so kann es durch zu groß gewählte Schrittweiten dazu kommen, dass dieses übersprungen wird und der Algorithmus in einem lokalen Minimum endet.

- Durch eine ungünstige Beschaffenheit der Funktion, kann es vor allem bei Symmetrie dazu kommen, dass die Iteration oszilliert. Der Ablauf der Optimierung dauert hier entweder extrem lange oder endet bei vollständig redundanter Oszillation gar nicht.

Allgemein ist zu sagen, dass das Gradientenabstiegsverfahren relativ langsam konvergiert und aufgrund seiner hier dargestellten Nachteile relativ fehleranfällig ist.

2.5.2. Newton-Verfahren

Ein Optimierungsverfahren mit höherer Konvergenzgeschwindigkeit stellt das hier nach [7] beschriebene Newton-Verfahren dar. Beim Newton-Verfahren wird die Folgestelle der Iteration ähnlich zum Gradientenabstiegsverfahren ermittelt, allerdings unter Zuhilfenahme der Funktion selbst. Hierbei entfällt die Wahl der Schrittweite denn der Subtrahend wird, wie in Formel 2-10 zu sehen ist, aus der Funktion und der Ableitung der Funktion gebildet.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad \Rightarrow \quad x_{k+1} = x_k - H^{-1}(x_k)\nabla f(x_k)$$

Formel 2-10: Gleichungssystem zum Newton Verfahren

Die Matrix H stellt hierbei die HESSE-Matrix dar. Diese entspricht der zweiten partiellen Ableitung von f im Punkt x_k. Das Newton-Verfahren bietet zwar höhere Konvergenz als das in 2.5.1 vorgestellte Gradientenverfahren, hat aber im Prinzip mit den gleichen Nachteilen zu kämpfen wie dieses. Ein weiterer Nachteil kann außerdem die fehlende Möglichkeit der Einflussnahme auf die Schrittweite sein und auch der Startwert muss gut gewählt sein, damit Konvergenz gegeben ist. Die Stärke dieses Verfahrens gegenüber dem Gradientenabstiegsverfahren ist dennoch gegeben und liegt hier nach [7] in der relativ schnellen Konvergenz.

2.5.3. CG-Verfahren

Ein weiteres sehr schnelles Optimierungsverfahren stellt nach [10] das Verfahren der Konjugierten Gradienten, auch CG-Verfahren genannt, dar. Das Verfahren bereits nach n-Schritten das exakte Ergebnis zurück (vgl. [10] und [11]). Eine Einschränkung bei diesem Optimierungsverfahren ist allerdings, dass es nur symmetrische, positiv definite Systeme uneingeschränkt geeignet ist. Das bedeutet dass alle Eigenwerte der hier verwendeten Matrix größer Null sein müssen und die Matrix gleich ihrer Transponierten ist. Ist diese Voraussetzung nicht erfüllt, so ist trotzdem, wenngleich auch nur bedingt geeignet. Das CG-Verfahren beruht auf der Beziehung zwischen der Minimierung einer Funktion *F(x)* und der Auflösung des Gleichungssystems *Ax = b*. Veranschaulicht wird dies durch das Erstellen einer Quadratischen Funktion und der anschließenden Gradientenbildung in Formel 2-11 (siehe auch [11]).

$$F(x) = \frac{1}{2} x^T Ax - x^T b \rightarrow \nabla F(x) = \frac{d}{dx} F(x) = Ax = b$$

$$\Rightarrow \nabla F(x) = 0 = Ax - b$$

Formel 2-11: Zusammenhang zwischen Min. F und Lösung des lin. Gl.-Systems

Die Besonderheit bei diesem Verfahren ist, dass mit Hilfe der Residuen eine Richtungskorrektur an den Abstiegsgradienten vorgenommen wird, um so ein optimales Ergebnis der Nährung zu erzielen. Es handelt sich bei diesem Verfahren also im strengen Sinne nicht um ein iteratives Näherungsverfahren, da die Rückgabe nach n Schritten das exakte Endergebnis darstellt (vgl. [11]). Bei der Durchführung jedes Iterationsschrittes wird der jeweilige Iterationsvektor um den α_i-fachen Korrekturvektor $p^{(i)}$ korrigiert, wie in Formel 2-12 zu sehen ist. Dabei ist $p^{(0)} := r^{(0)} := b - Ax^{(0)}$ wobei $x^{(0)} \in R^n$ ist.

$$x^{(i)} := x^{(i-1)} + \alpha_i p^{(i)}$$

Formel 2-12: Iterationsschritt beim CG-Verfahren

Das Ende der Iteration tritt ein sobald $p^{(i)} = 0$ ansonsten wird wie folgt mit der Iteration fortgefahren. Zunächst müssen die Residuen entsprechend aktualisiert werden was durch den Ausdruck in Formel 2-13 geschieht.

$$r^{(i)} := r^{(i-1)} + \alpha q^{(i)} \quad mit \quad q^{(i)} = Ap^{(i)}$$

Formel 2-13: Aktualisierung des Residuums

Der Multiplikator des Korrekturvektors wird dann durch Formel 2-14 angepasst.

$$\alpha = \alpha_i = \frac{r^{(i-1)T} r^{(i-1)}}{p^{(i)T} A p^{(i)}}$$

Formel 2-14: Anpassung des Multiplikators

Durch die Residuen kann nun auch der Korrekturvektor entsprechend angepasst werden. Dies geschieht durch den in Formel 2-15 dargestellten Ausdruck.

$$p^{(i)} := -r^{(i-1)} - \frac{r^{(i-1)T} r^{(i-1)}}{r^{(i-2)T} r^{(i-2)}} p^{(i-1)}$$

Formel 2-15: Anpassung des Korrekturvektors

Durch Verwendung des in Formel 2-15 verwendeten Quotienten wird sichergestellt dass sowohl $p^{(i)}$ als auch $Ap^{(i-1)}$ orthogonal sind und somit ein optimaler Abstieg zum Minimum der Funktion gewährleistet wird. Dieses Verfahren ist sehr effizient und kann durch entsprechende Vorkonditionierung der Vektoren noch verbessert werden (siehe dazu [10]).

Wie noch gezeigt werden wird stellt die Wahl des Optimierungsverfahrens einen entscheidenden Aspekt im gesamten Programmablauf dieses System-entwicklungsprojektes entwickelten Berechnungsverfahrens dar. Da in der konkreten Umsetzung des Algorithmus des SEP das Gradientenabstiegs-verfahren ohne weitere Optimierungen gewählt wurde, stellt die Wahl eines bessern Iterationsverfahrens einen entscheidenden Ansatzpunkt zur Verbesserung des gesamten Algorithmus dar.

3. Implementierung

Zur Implementierung des gesamten Algorithmus muss zunächst der grobe Ablauf des Programms entworfen werden um die richtige Vorgehensweise festzulegen. Die Vorraussetzung für die Berechnung der relativen 3D-Koordinaten sind die 2D-Positionen von einzelnen Punkten auf dem zu verfolgenden Gesicht. Hierbei liefert das Framework die entsprechenden Daten um die in 2.5 erwähnte Punktewolken zu erhalten. Die einzelnen Punkte sollen nun von Bild zu Bild verfolgt und jeweils durch den Algorithmus zur Berechnung der relativen Koordinaten im Raum ausgewertet werden. Das zusammenfassen der Punkte geschieht hierbei mittels einer dreidimensionalen Maske in Form einer gestreckten Halbkugel, die auf das jeweilige Gesicht gelegt wird. Diese Maske wird zunächst initial positioniert und dann kontinuierlich in jedem Bild weiterbewegt. Der Ablauf stellt sich hierbei wie in Abbildung 3-1 dar.

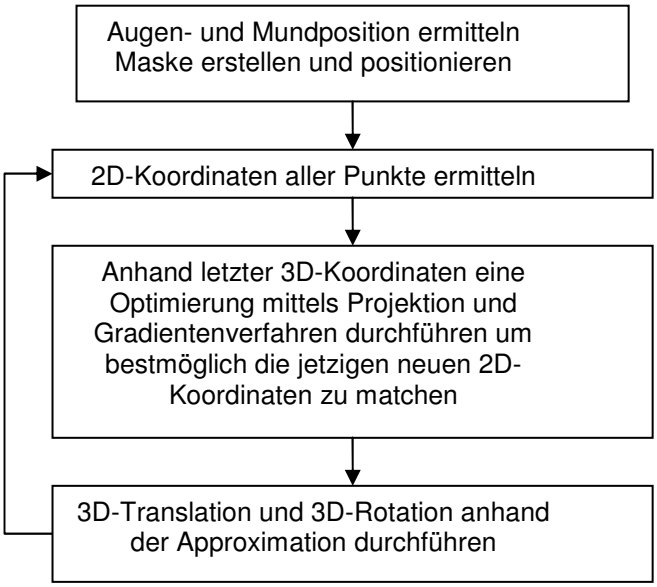

Abbildung 3-1: Grobes Schema des Programmablaufes

Die nachfolgenden zwei Kapitel stellen das Verfahren genauer dar.

3.1. Initiale Positionierung

Bei diesem Vorgang stellt sich das Problem der Dimensionierung der Maske um das Gesicht vollständig zu bedecken, da beim Menschen die Kopfform und Große sehr unterschiedlich ausgeprägt ist. Die Lösung dieses Problems liegt in der Verwendung der vom Framework zurück gelieferten Positionsdaten von Mund und Augen. Hierbei werden die Abstände zwischen den Augen und von den Augen zum Mund wie in Formel 3-2 verwendet um eine korrekte Dimensionierung und Positionierung der Maske zu gewährleisten.

$$\begin{aligned} \text{faceWidth} &= (\text{rightEye->x} - \text{leftEye->x}) * 2.2 \\ \text{faceCenterX} &= (\text{rightEye->x} + \text{leftEye->x}) / 2.0 \\ \text{faceCenterY} &= (\text{mouth->y} + 2 * \text{leftEye->y}) / 3.2) \end{aligned}$$

Formel 3-2: Formeln zur korrekten Positionierung und Dimensionierung der Maske

Der Mittelpunkt des Gesichtes wird hier in der Horizontalen auf den Mittelpunkt zwischen beiden Augen und auf der Vertikalen zwischen Mund und Augen gelegt. Dies entspricht etwa der Position der Nasenspitze. Die Breite des Gesichtes wird als in etwa doppelter Abstand der Augen angenommen. Die hierbei verwendeten Multiplikatoren und Divisoren sind Schätzungen die auf manuell durchgeführten Tests, also Erfahrung, beruhen. Nachdem die Koordinaten zur Positionierung bestimmt wurden werden nun die Punkte der Maske im Dreidimensionalen erzeugt und im Anschluss zurück auf die Bildebene projeziert um diese darzustellen. Im Idealfall sollte hierbei die Maske exakt das zu verfolgende Gesicht bedecken. Ein weiterer wichtiger Aspekt bei der Positionierung ist die Festlegung der Einheiten der 3D-Koordinaten. Hierzu wird wiederum die Größe des zu verfolgenden Gesichtes herangezogen um eine Schätzung auf die Position in Metern vorzunehmen.

faceWidth_meter = faceWidth * size_pixel_meter
x_2d_meter = faceCenter.x * size_pixel_meter
y_2d_meter = faceCenter.y * size_pixel_meter
y_3d = 2.0 * radiusOfGlobe * focus_meter / faceWidth_meter
x_3d = y_3d * x_2d_meter / focus_meter
z_3d = y_3d * y_2d_meter / focus_meter
centerOfGlobe = cvPoint3D32f(x_3d, y_3d, z_3d)

Formel 3-3: Umrechnung der Einheiten der Koordinaten in Meter

In Formel 3-3 ist zu erkennen, dass hierzu ein Faktor size_pixel_meter verwendet wird. Dieser Faktor stellt die Größe eines Pixels auf der Bildebene in Meter dar. Multipliziert man also die Positions- und Größenangaben der auf der Bildebene dargestellten Objekte so lässt sich eine Abschätzung auf die Position in Metern des Gesichtes im Raum darstellen. Dies geschieht wiederum mittels der in 2.4.1 vorgestellten Formel zur Zentralprojektion wobei allerdings hierbei die 2D-Koordinaten und der Fokus in metrischen Einheiten verwendet werden. Somit wird also die Breite des Gesichtes in Meter umgerechnet. danach werden die X- und Y-Position in Metern berechnet. Die letzten drei Formeln stellen das Zentrum der Gesichtsmaske im Dreidimensionalen dar wobei die Tiefe aufgrund der angenommen Kopfbreite und der tatsächlichen Kopfbreite auf der Bildebene durch die Zentralprojektion ermittelt wird. Danach erfolg die X- und Y-Positionierung ebenfalls mit Hilfe der Zentralprojektion und der zuvor errechneten Tiefe.

3.2. Approximation der neuen 3D-Koordinaten

Das Approximieren der neuen Koordinaten der Maske im Dreidimensionalen auf Basis der zuvor gewonnen 3D-Koordinaten wirft zunächst einmal die Frage auf, auf welche Art und Weise der Algorithmus die Parameter der Transformation optimieren soll. Hierbei kann die Rotation und Translation gemeinsam in einem Durchgang oder auch getrennt optimiert werden. In letzterem Fall stellt sich zusätzlich noch die Frage in welcher Reihenfolge dies zu geschehen hat und wie oft die Iteration wiederholt werden sollte um ein optimales Ergebnis zu erzielen. Außerdem ist darauf zu achten, das die gesamte Optimierung in adäquater Zeit abläuft um den Einsatz in Echtzeit zu gewährleisten. Der Prozess der Optimierung soll weniger als 80 Millisekunden beanspruchen um eine Framerate von 10 Bildern pro Sekunde aufrecht-

zuerhalten. Hierbei wurden bereits 20 Millisekunden für die Darstellung jedes Bildes eingerechnet. Es hat sich herausgestellt, dass das beste Verfahren ein Auftrennen der Transformationen ist, da bei gleichzeitiger Optimierung die gegenseitige negative Beeinflussung zu groß ist um ein gutes Ergebnis zu erhalten. Die beste Optimierung wurde erreicht indem zunächst die Translation in bis zu 40 Iterationsschritten optimiert wird um dann im Anschluss die Rotation ebenfalls in etwa 40 Schritten zu optimieren. Ist allerdings der errechnete Fehlerwert kleiner 1 oder stagniert das Verfahren zu stark, so dass der Fehler sich kaum verbessert so wird die Berechnung vorzeitig abgebrochen. Speziell bei der Optimierung der Rotation muss allerdings zunächst noch ein zusätzlicher Schritt durchgeführt werden. Da die Rotation um ein definiertes Zentrum stattfinden soll, muss zunächst dieses Rotationszentrum festgelegt werden. Als Zentrum der Rotation wurde hierbei der Mittelpunkt des Kopfes gewählt, da dies in etwa der Drehachse des Kopfes entspricht. Im Anschluss an beide Optimierungen findet dann die Anwendung der so gewonnen Transformationen auf die Punkte der Maske im Dreidimensionalen statt. Der gesamte Prozess stellt sich wie in Abbildung 3-4 dar.

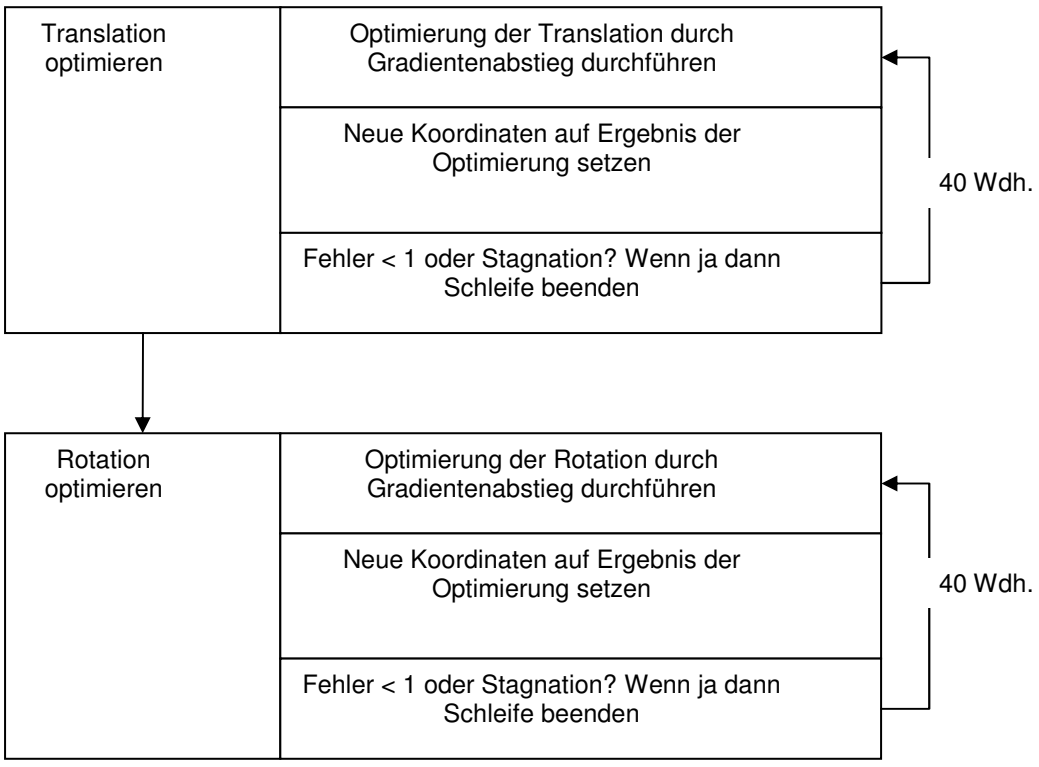

Abbildung 3-4: Schema des Optimierungsprozesses

Das Ergebnis des gesamten Verfahrens ist in Abbildung 3-5 zu sehen. Die grünen Linien bilden hier die Maske die auf das Gesicht aufgesetzt wird. Die roten Punkte stellen die im Zweidimensionalen getrackten Punkte auf dem

Gesicht dar. Der in Violett gezeichnete Pfeil ist der Richtungsvektor der Maske und stellt nur ein zusätzliches Hilfsmittel zur Visualisierung dar.

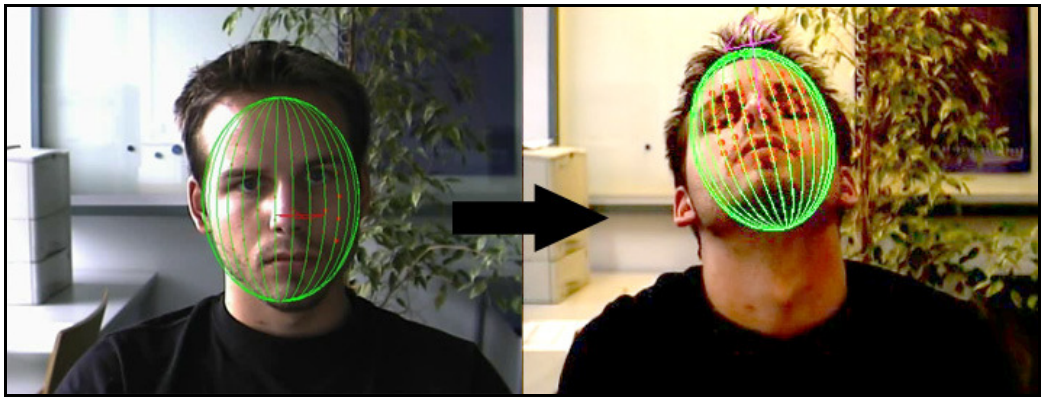

Abbildung 3-5: Getracktes Gesicht mit positionierter Maske

4. Ausblick

Der hier vorgestellte Prozess zur Verfolgung eines Gesichtes im Dreidimensionalen ist nur als erster Ansatz eines Verfahrens zu sehen. Ein ausgereiftes Verfahren bedarf einer Optimierung in Hinsicht auf Ausführungsgeschwindigkeit, Genauigkeit und Stabilität des gesamten Algorithmus. Als erster Ansatz zur Verbesserung des Optimierungsverfahrens kann beispielsweise auf das in 2.5.3 vorgestellte CG-Verfahren zurückgegriffen werden. Im Optimierungsverfahren liegt das größte Potential zur Verbesserung in Hinsicht auf Laufzeit und Stabilität des Algorithmus, da sich Fehler in der Berechnung fortsetzen und verstärken je länger die Verfolgung andauert. Um dieses Problem zu beheben ist es als Lösungsansatz denkbar eine Reinitialisierung der Grunddaten, z.B. als Datensatz gespeicherte festgelegte mögliche Positionen des Gesichtes, bei zu schlechten Ergebnissen der letzten Berechnung durchzuführen. Ebenso sollten diese Grunddaten, zuvor in Form einer „Ground truth“ festgelegt sein, denn es kann beispielsweise kein Fall auftreten, in dem das Gesicht komplett von der Kamera weg weist da hierbei die Entsprechenden Punkte auf dem Gesicht nicht mehr sichtbar sind. Indem solche Maßnahmen ergriffen werden würde sich der gesamte Prozess stabiler verhalten und somit ein „abrutschen“ der Maske verhindert. Ein weiterer Schritt zur wesentlichen Verbesserung des gesamten Verfahrens ist die Verwendung flexibler Masken wie es in [6] geschieht. Da menschliche Gesichter nicht starr sind ist der im SEP verwendete Algorithmus nur bedingt geeignet und selbst mit den schon genannten Verbesserungen ohne ein flexibles Gesichtsmodell nur

bedingt geeignet. Durch fehlende Korrelation in den Punktewolken wird bei Veränderungen der Gesichtszüge der Algorithmus meist versagen und die Maske nicht mehr korrekt positioniert. Eine weitere elementare Verbesserung kann durch die in 2.3 vorgestellte Kalibrierung der Kamera erreicht werden. Erst durch die Kalibrierung ist es möglich exakte Ergebnisse zu erzielen.

Abschließend ist zu bemerken, dass mit den hier vorgestellten Mitteln eine Wesentliche Verbesserung im Gesamten Verfahren erreicht werden kann. Die hier vorgestellte Implementierung soll dennoch zeigen, dass die Umsetzung eines Algorithmus zur Verfolgung eines dreidimensionalen Gesichtsmodells mittels einer Kamera schon mit einfachen Mitteln zu bewerkstelligen ist.

LITERATURVERZEICHNIS

[1] **Jähne, Bernd** (2005): Digitale Bildverarbeitung, Berlin Heidelberg (Springer-Verlag) 2005, S. 229 f.

[2] Zum Begriff „Triangulation“ bei Wikipedia.org http://de.wikipedia.org/wiki/Triangulation_%28Geod%C3%A4sie%29 [Stand 28.06.2006]

[3] **Lepetit , Vincent; Fua, Pascal** (2005): Monocular Model-Based 3D Tracking of Rigid Objects, Hanover USA (Now Publishers Inc) 2005, S. 1 f.

[4] **Ma, Lili; Cheng, YangQuan;Moore, Kevin L**.(2003): Paper: Flexible camera calibration using a new analytical radial undistortion formula with application to mobile robot localization, (Utah State University) 2003

[5] Deformable model tracking gri.gallaudet.edu/ ~cvogler/research/ [Stand 30.6.2006]

[6] **Vogler, Goldenstein, Stolfi, Pavlovic** (2006): Paper: Outlier Rejection in High-Dimensional Deformable Models, (Gallaudet University, Universidade Estadual de Campinas, Rutgers University) 2006

[7] **Bronstein, Semendjajew, Musiol, Mühlig** (2005): Taschenbuch der Mathematik, 6. Auflage, Frankfurt. a. M. (Harri Deutsch GmbH) 2005, S. 895 ff.

[8] **Lippe, Wolfram-Manfred** (2006): Soft-computing: Mit neuronalen Netzen, Fuzzy-logic und Evolutiondren Algorithmen, Berlin Heidelberg New York (Springer) 2006, S. 502 ff.

[9] www.neuronalesnetz.de Paper: Neuronale Netze: Eine Einführung**,** Stand 27.07.2006

[10] **Überhuber, Christoph** (1995): Computer-Numerik II, Berlin Heidelberg (Springer) 1995, S. 433 f.

[11] http://www.mechbau.uni-stuttgart.de/ls2/100-online/HMIB/kap3/kap3_2.html Stand 28.07.06