Soziale Netzwerke durch Augenkontakt auf Basis der Balancetheorie

Studienarbeit
im Studiengang Computervisualistik

vorgelegt von:
Patrick Alexander Bardow
bardow@uni-koblenz.de

Betreuer: Dipl. Inf. Claudia Obermaier, Arbeitsgruppe Künstliche Intelligenz
Dipl. Inf. Björn Pelzer, Arbeitsgruppe Künstliche Intelligenz
PhD Takashi MINATO, Osaka University

Koblenz, im November 2009

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Koblenz, den . . . . . .

Danksagung

An dieser Stelle will ich mich bei meinen Betreuern Claudia Obermaier, Björn Pelzer, MINATO Takashi herzlich bedanken, die mich in meiner Arbeit unterstützt und sehr gut betreut haben. Auch dem Deutsch Akademischen Austauschdienst (DAAD), Claudia Obermaier und Doris Wenzel, die mir ein Auslandssemester in Japan ermöglicht haben, möchte herzlichst danken. Das Semester hat mich über das Fachliche hinaus gefördert und mir ein Einblick in eine hoch interessante Kultur gewährt.

Ich möchte mich bei dem Mitarbeitern im ERATO-Labor bedanken, die mir bei vielen Fragen zur Seite standen und mich herzlichst empfangen haben.

Ein letzter Dank geht an meine Familie, die mich immer unterstützt und mir geholfen hat, die Texte verständlicher zu gestalten.

Inhaltsverzeichnis

Kapitel 1

Einleitung

In der Robotik und in der Künstlichen Intelligenz ist es das Ziel, Agenten zu entwickeln, die selbständig nach der Umgebung handeln und lernen. Zur Entwicklung dieser autonomen Systeme, werden verschiedene Lösungstrategien nach den jeweiligen Anforderungen verfolgt.

Die Entwicklung autonomer Fahrzeuge, die sich ohne menschliche Kontrolle bewegen, ist ein Beispiel für die Entwicklung möglichst eigenständiger und robuster Agenten als Ziel der Robotik im Ingenieurswesen.

Hingegen die Androiden-Forschung und -Entwicklung sich an dem Menschen orientiert und studiert, wie der Mensch lernt und denkt.

Wichtiger Bestandteil ist dafür die menschliche Kommunikation, die für Sprachwissenschaftler einen wichtigen Beitrag zur menschlichen Intelligenz liefert, aber auch für das soziale Zusammenleben des Menschen eine große Rolle spielt.

Für eine natürlich wirkende Interaktion von Mensch und Maschine ist die Analyse der menschlichen Kommunikation unabdingbar. Jedoch ist die Kommunikation ein hoch komplexer Vorgang mit vielen Faktoren, die es zu berücksichtigen gilt.

Die Psychologie bietet verschiedene Modelle zur Darstellung und Erläuterung des menschlichen Verhaltens. Zur Dynamik in Personengruppen bietet die Psychologie zum Beispiel die Balancetheorie von Fritz Heider, die die Bewertungen und Einstellungen der Personen in Relationen untereinander darstellt.

Dabei ist es eine Herausforderung, die Erkenntnisse aus der Psychologie in die Robotik zu übertragen. Dafür eignen sich insbesondere Theorien, die einfach darstellbar sind, wie die Balancetheorie, die einer Graphenstruktur zur Grunde liegt.

Ein Projekt, das sich mit der Entwicklung des Menschen in den frühen Entwicklungsjahren beschäftigt, ist das Projekt um CB^2 (Child-Robot with Biomimetic Body)[1].

[1] Aussprache (engl.): „C B Square“

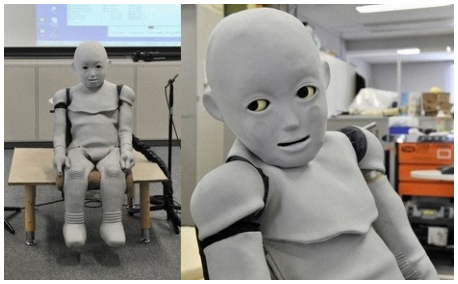


Abbildung 1.1: CB^2 im ERATO-Labor an der Osaka University. [Pin09]

Das Projekt um CB^2 befindet sich im ERATO-Labor der Osaka University und beschäftigt sich damit, das Verhalten und Lernen von Babys zu erforschen. Ziel des Projekts ist es, CB^2 das Verhalten und die kognitiven Fähigkeiten eines menschlichen Babys zu verleihen.

Eine der Forschungsarbeiten um CB^2 beschäftigt sich zum Beispiel damit, dem Roboter zu ermöglichen, selbständig die Positionen der Tastsensoren, die sich unter der Haut befinden, an seinem Körper zu bestimmen. Das geschieht nach Vorlage von menschlichen Babys, die ebenfalls in mehreren Abläufen ein besseres Verständnis und Gefühl für ihren Körper entwickeln.

CB^2 ist ca. 1,3 Meter groß und wiegt 33 Kilogramm. Er verfügt über 51 verschiedene Motoren und Gelenke und 197 Tastsensoren unter der Haut. [Pin09]

Ziel dieser Arbeit ist es, die Balancetheorie von Fritz Heider aus dem Fachbereich der Psychologie als echtzeitfähige und flexible Anwendung in CB^2 zu integrieren.

Kapitel 2

Stand der Forschung

Die Arbeit ist in ihrer Essenz eine Zusammenführung verschiedener Disziplinen. In diesem Kapitel werden zwei Wissenschaftsgebiete vorgestellt, aus denen diese Arbeit die theoretischen Grundlagen entnimmt, nämlich die *Joint Attention*, die in verschiedenen Bereichen Anwendung findet, unter anderem die Robotik, und die *Balancetheorie* aus der Psychologie, die den theoretischen Kern dieser Arbeit bildet.

2.1 Joint Attention

Joint Attention ist im Allgemeinen die Fähigkeit, nonverbal zu interpretieren, wohin eine Person schaut oder zeigt. Es ist ein bilateraler Prozess, der eine Veränderung der Aufmerksamkeit erfordert und zudem die Fertigkeit, Intentionen zu deuten, voraussetzt. [KH04]

Die Joint Attention umschreibt einige Fähigkeiten, die die beteiligten Agenten besitzen [KH04]:

- Die Agenten sind in der Lage, die Intentionen des anderen zu bemerken, wie zum Beispiel dem Blick zu folgen.
- Auch kann der eine Agent gezielt die Aufmerksamkeit des anderen beeinflussen, was direkt mit dem ersten Punkt zusammenhängt.
- Für die Verteilung der Rollen (z.B. beim Lehrer-Schüler-Verhältnis), besitzen beide Agenten ein Maß für die soziale Koordination.
- Zudem geht der Agent davon aus, dass der andere über eine Intention verfügt, und dass sich diese von seiner eigenen unterscheiden kann.

Joint Attention ist ein alltäglicher Bestandteil unseres Lebens und unserer Kommunikation. Daher ist die Entwicklung der Joint Attention in der Robotik maßgeblich für die Interaktion mit der Maschine. So könnte ein Roboter einer Richtungsangabe folgen, ohne die Eingabe von Koordinaten. Auch wäre es möglich, Dinge durch Deuten mit dem Zeigefinger und eine Sprachverarbeitung, für den Roboter zu benennen.

Für die Entwicklung der Joint Attention in der Robotik, existieren mehrere Ansätze in der Forschung. So gibt es Arbeiten, die sich mit der Imitation eines menschlichen Supervisors beschäftigen. [TI04]

Diese Arbeiten erfordern ein hohes Maß an Koordination der motorischen Bewegungen des Roboters und die Detektion und Analyse des Gegenübers. Allerdings findet keine Interpretation des Gesehenen statt. So werden die Bewegungen wahrgenommen, aber nicht weiter interpretiert.

Für die Bestimmung der Blickkontakte der Personen zum Roboter und unter ihnen, ist im Rahmen der Arbeit eine schnelle und effektive Applikation entwickelt worden. Die Applikation ist dem theoretischen Bereich der Joint Attention zuzuordnen. Die Analyse der sozialen Strukturen basiert auf diesen Blickkontakten, aber entnimmt ihrer Grundlage der Balancetheorie.

2.2 Balancetheorie

Die Grundidee der Studienarbeit fundiert auf der Balancetheorie von Fritz Heider. Die Balancetheorie beschreibt die soziale Interaktion zwischen zwei Personen mit einem Objekt oder einer weiteren Person, sowie die Dynamik dieser Dreiecksbeziehung. [MH]

Die Arbeit von Fritz Heider, die 1946 veröffentlicht wurde, stützt sich dabei auf die *Konsistenztheorie*, die ein Streben der Menschen nach Konsens ihrer kognitiven Erlebnisse voraussetzt. Eine Inkonsistenz des Erlebten wird dabei als störend empfunden und löst ein Erstreben zu einem harmonisierten Zustand aus. [MH]

Dabei erstreckt sich dieses Streben in verschiedene Bereiche, wie

- der Wahrnehmung (Allgemeingültigkeit der Wahrnehmung),
- der Denkprozesse (Mathematik, Logik etc),
- der Einstellung (Persönliche Meinung) und
- der Handlung (Verhalten).

Wenn Menschen miteinander interagieren, kann es in diesen Bereichen zu Widersprüchen kommen. Zum Beispiel wäre eine Aussage wie „Die Wand ist grün“, obwohl die Wand von den anderen Beteiligten als weiß empfunden wird, eine Inkonsistenz des Wahrgenommenen und kann dafür sorgen, dass die Anderen versuchen, den Sprecher von ihrer Wahrnehmung zu überzeugen oder seine Meinung zu ignorieren und ihn so aus der Gruppendynamik auszuschließen.

Entscheidend bei dem Beispiel ist, dass ein Streben zu einem harmonisierten Zustand entsteht. In der Heiderschen Theorie werden diese Zustände durch drei Elemente und ihre Relationen dargestellt:

- **P** als die wahrnehmende Person selbst,
- **O** als eine andere Person (engl. **O**ther) und
- **X** als ein Objekt oder als eine weitere Person.

Zwischen diesen Elementen existieren Relationen, die entweder von positiver oder negativer Natur sind. Im psychologischen Kontext meint eine positive Relation eine Wertschätzung oder Zuneigung, während eine negative Relation Geringschätzung und Abneigung bedeutet.

Ebenfalls werden noch zwei Arten von positiven und negativen Verbindungen unterschieden, nämlich *Einheits-* und *Wertrelationen*. Die Wertrelationen sind, im Gegensatz zur Einheitsrelation, in der Regel immer gerichtet. Ein Beispiel für eine Wertrelation wäre die Liebe, die immer gerichtet von einer Person zur anderen führt, während das Zusammenleben von zwei Personen ein Beispiel für eine ungerichtete Einheitsrelation wäre.

Dyaden und Triade

Heider beschränkt seine Theorie auf eine Gruppengröße von zwei bis drei Elementen. Eine zwei elementige Struktur wird in der Literatur als *Dyade* bezeichnet.

Eine besondere Aufmerksamkeit wird bei der Analyse dieser Strukturen auf die *Balance* gerichtet. Balanciert meint dabei, dass die Relationen wechselseitig zutreffen. Zum Beispiel, wenn sich bei einer Dyade die Personen gegenseitig lieben, ist das eine balancierte Struktur. Jedoch ist die Konfiguration „Peter liebt Sandra, aber Sandra liebt Peter nicht“ eine durchaus unbalancierte Struktur (s. Abbildung 2.1).

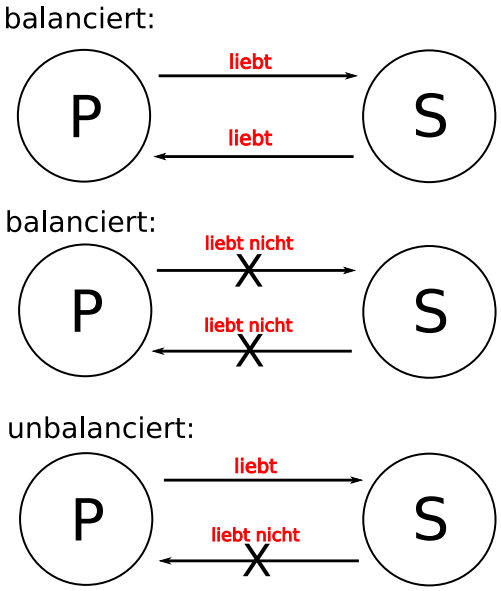


Abbildung 2.1: Darstellung der balancierten und unbalancierten Dyaden.

In Abbildung 2.1 sind die drei möglichen Konfigurationen der Dyaden zu sehen. Dabei ist die unbalancierte Struktur heterogen im Punkte der gerichteten Kanten, während die balancierten homogen sind.

Die balancierten Strukturen sind stabil und erhalten ihren Zustand, während die unbalancierten Konfigurationen instabil sind und eine Veränderung bewirken. Im Beispiel würde sich die Veränderung darstellen, indem Sandra anfängt, Peter zu lieben oder Peter aufhört, für Sandra Gefühle zu empfinden.

Darauf aufbauend wird die Struktur der Dyade in der Heiderschen Theorie erweitert zur *Triade*, einer Dreiecksbeziehung. Zur Veranschaulichung wird nur noch beachtet, ob eine Relation positiv oder negativ ist. Es existieren dabei unter den drei Elementen acht mögliche Kombinationen, von denen vier balanciert und vier unbalanciert sind (s. Abbildung 2.2).

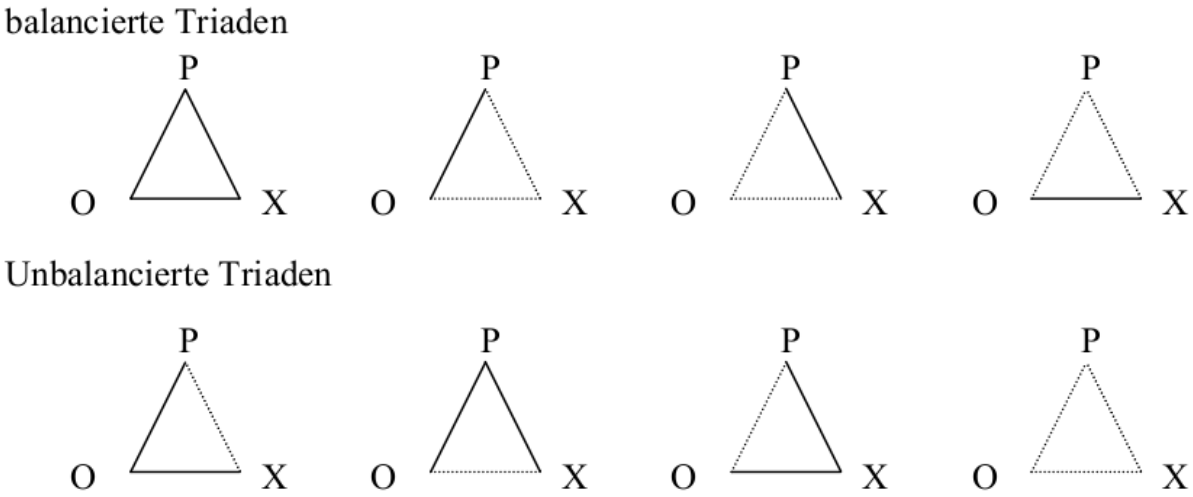


Abbildung 2.2: Graphendarstellung von balancierten bzw. unbalancierten Triaden. [MH]

In Abbildung 2.2 lässt sich feststellen, dass bei einer balancierten Triade entweder alle Relationen positiv sind, oder dass eine der Relationen positiv und die anderen beiden negativ sind. Hingegen die unbalancierten Triaden entweder vollständig negativ sind, oder über zwei positive und eine negative Relation verfügen.

Wie bereits erwähnt, erwirkt eine unbalancierte Konfiguration eine Tendenz, sich zu einer balancierten Struktur zu ändern. In den einfachsten Fällen muss dazu lediglich eine der Relationen modifiziert werden.

Zur Veranschaulichung wird eine Gruppe mit den Elementen: Peter, Oskar und Xena betrachtet. So wären die Relationen:

- Peter und Oskar sind Freunde
- Oskar und Xena sind in einer Beziehung
- Jedoch kann Peter Xena nicht leiden

nicht stabil und führen zu einem Konflikt. Um die Struktur in einen stabilen Zustand zu führen, könnte beispielsweise Peter seine Meinung ändern und anfangen Xena zu mögen. Eine Konfliktlösung könnte auch erfolgen, indem Oskar sich entscheidet, den Kontakt mit Peter oder die Beziehung mit Xena zu beenden. Alle drei Möglichkeiten sind elementare Entscheidungen, die wieder zu einer stabilen Konfiguration führen, die auch von den anderen Elementen durchgeführt werden können.

Kritik

Die Balancetheorie wird häufig in der Psychologie zitiert, da sie eine Vielzahl von Phänomenen, die mit der Bewertung von Einstellungen und Verhalten zusammenhängen, darstellen kann.

Allerdings beschränkt sich die Balancetheorie auf eine Gruppengröße von zwei bis drei Personen, und lässt sich nur bedingt auf größere Mengen anwenden.

Ebenfalls wird binär zwischen positiven und negativen Relationen unterschieden, ohne eine Aussage über die Stärke dieser Verbindungen zu treffen.

Auch ist die Balancetheorie eine reine Darstellungsform und ermöglicht keine Vorhersage von Verhalten. [MH]

2.3 Eigenfaces

Die Eigenface-Methode basiert auf der Arbeit von Sirovich und Kirby. Die Methode erlaubt, Gesichtserkennungsverfahren mit vergleichbar wenig Speicheraufwand und dementsprechend schneller Berechnung zu entwickeln.

Ein zentraler Berechnungsschritt bei der Eigenface-Methode ist die *Hauptkomponentenanalyse*, die 1901 von Karl Pearson eingeführt und rund dreißig Jahre später von Harold Hotelling weiterentwickelt wurde. Die Hauptkomponentenanalyse (engl. *Principal component analysis*, kurz *PCA*) wird in der Bildverarbeitung auch *Karhunen-Loève-Transformation* genannt.

Die PCA-Methode, die aus dem Bereich der multivariaten Statistik hergeleitet wurde, eignet sich insbesondere zur Strukturierung und Vereinfachung von großen Datensätzen. Dazu wird der Datensatz durch Linearkombination von möglichst aussagekräftigen Faktoren genähert.

Anschaulich dargestellt wird der Raum aller möglichen Bilder aufgestellt. In diesem Raum werden die Koordinaten eingetragen, die jeweils ein Bild der Datenmenge repräsentieren. Wenn die Datenmenge eine Sammlung von Bildern eines Gesichtes ist, dann werden sich diese Punkte in einem Bereich anordnen, der als Unterraum beschrieben werden kann.

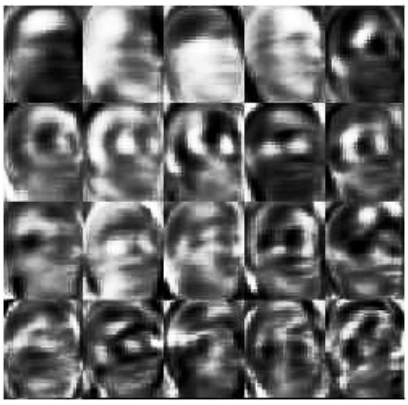


Abbildung 2.3: Darstellung von Eigenfaces. [BG04]

Der Grund für diese Darstellung liegt an der hohen Dimension des Raumes. Bei einer Bildauflösung von 100×100 Pixel stellt das Koordinatensystem einen zehntausend-dimensionalen Raum dar. Um das zu reduzieren, wird die PCA-Methode angewandt, die die Informationen auf charakteristische Merkmale und Werte abbildet und speichert.

Die Eigenface-Methode lässt sich im Allgemeinen in zwei Bereiche aufteilen: Das *Lernen* und das *Erkennen* von Gesichtern.

Lernverfahren

Das Lernverfahren beschäftigt sich damit, eine Menge von Bildern zu liefern, die vom System gelernt und später differenziert werden. Dem System ist demnach ein Satz von Bildern $\{\Gamma_1 \dots \Gamma_M\}$ gegeben. Aus diesem Satz wird ein Unterraum gebildet, der von den *Eigenvektoren* beschrieben wird, die im Eigenface-Verfahren auch *Eigenfaces* genannt werden.

Durch Linearkombination der Eigenvektoren und der Gewichtung können die Trainingsbilder wieder rekonstruiert werden (s. Formel 2.6).

Zunächst wird für die Berechnung der Eigenfaces das *Durchschnittsgesicht* aus der Trainingsmenge berechnet. Definiert ist es mit

$$\Psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma_n \tag{2.1}$$

Ist das Durchschnittsbild berechnet, werden die Vektoren von dem Durchschnittsgesicht zu den Bildern der Trainingsmenge hergeleitet mit

$$\Phi_i = \Gamma_i - \Psi \tag{2.2}$$

Dadurch wird das Durchschnittsgesicht zum Nullpunkt, der den Unterraum definieren lässt. Dazu werden aus den M Bildern M orthogonale Vektoren definiert, die den Unterraum, in dem die Bilder der Trainingsmenge liegen, beschreiben. Dabei ist die Bedingung für den k-ten Vektor u_k, dass er die Gleichung erfüllt

$$\lambda_k = \frac{1}{M} \sum_{n=1}^{M} (u_k^T \Phi_n)^2 \tag{2.3}$$

und ein Maximum liefert im Kontext auf

$$u_l^T u_k = \delta_{lk} = \begin{cases} 1, & \text{wenn } l = k \\ 0, & \text{sonst} \end{cases} \tag{2.4}$$

Die Vektoren u_k bilden demnach die Eigenvektoren bzw. Eigenfaces und die Gewichtungen λ_k sind die *Eigenwerte*. Beide Elemente sind Teil der Kovarianzmatrix (2.5)

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T \tag{2.5}$$

Da M eine recht große Zahl sein kann, werden nur die Eigenvektoren weiter beachtet, die über die größten Eigenwerte verfügen. Dadurch kann der Speicheraufwand erheblich reduziert werden, da lediglich die aussagekräftigsten Bilder bestehen bleiben, die somit den Raum definieren.

Gesichtserkennung

Die Gesichtserkennung projiziert ein zu überprüfendes Bild in den Unterraum. Wenn die Projektion stattgefunden hat, wird über die bekannten Gesichter iteriert und der jeweilige Abstand gemessen. Wenn der Abstand unter einem bestimmten Schwellwert liegt, wird das Gesicht als bekannt, und wenn nicht, als unbekannt eingestuft.

Die Projektion geschieht mittels Linearkombination (2.6) der Basisvektoren des Unterraums.

$$\omega_k = u_k^T(\Gamma - \Psi), \quad \text{mit } k \in \{1, \ldots, M\} \tag{2.6}$$

Wobei Γ das Eingabebild ist. Die Gewichtungen ω_k bilden den Vektor $\Omega^T = [\omega_1, \ldots, \omega_M]$. Ω ist das projizierte Eingabebild mit dem der Abstand zu den anderen bekannten Bildern gemessen wird. Beispielsweise ist hierfür der Euklidische Abstand (s. Formel 2.7) geeignet.

$$\epsilon_k = ||(\Omega - \Omega_k)||^2 \tag{2.7}$$

Ω_k mit $k \in \{1, \ldots, M\}$ stellt die bekannten Gesichter im Unterraum dar. Wenn das kleinste ϵ_k unter einem bestimmten Schwellwert liegt, so wird das Gesicht der selben Identitätsklasse zugeordnet, in der sich auch das zu vergleichende Gesicht befindet.

Ein weiteres Verfahren für die Distanzberechnung der Vektoren ist die *Mahalanobis-Distanz*. Die Mahalanobis-Distanz, benannt nach Prasanta Chandra Mahalanobis, ist definiert durch

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T C^{-1} (\vec{x} - \vec{y})} \tag{2.8}$$

Wobei C^{-1} die Inversere der Kovarianzmatrix darstellt. Der Vorteil der Mahalanobis-Distanz gegenüber dem Euklidischen Abstand liegt darin, dass sie Korrelationen zwischen den Koordinaten und verschiedene Skalierungen berücksichtigt, die besonders in der späteren Implementierung zur Geltung kommen. [Loh]

Abhängigkeiten der Eigenface-Methode

Die Eigenface-Methode zieht ihre Vorteile aus dem vergleichbar geringen Speicheraufwand und der relativ schnellen und einfachen Projektion der Eingabebilder.

Allerdings ist die Eigenface-Methode rein mathematisch motiviert und entwickelt. Zudem ist sie nicht speziell für die Gesichtserkennung hergeleitet worden. So lässt sie sich auch für die Objekt-Erkennung nutzen. Dahingehend ergeben sich einige Schwierigkeiten, die bei einer Implementation bedacht werden sollten, um eine möglichst effektive Anwendung der Eigenface-Methode zu erhalten.

Besonders eingeschränkt wird die Methode durch ihre Abhängigkeiten,

- der Lokalität,
- der Skalierung,
- der Beleuchtung und
- der Orientierung.

Im einzelnen bedeutet es, dass das zu untersuchende Gesicht dem der Trainingsbilder möglichst ähnlich sein sollte. Je mehr es sich unterscheidet, desto geringer ist die Wahrscheinlichkeit, dass das Gesicht richtig klassifiziert wird. Daher ist es entscheidend, wo sich das Gesicht im Bild befindet, welche Größe das Gesicht im Bild hat und ob die Kopfausrichtung ebenfalls dem Trainingsmaterial entspricht. Wenn zum Beispiel die Person auf dem Kopf steht und dieses Bild nicht beim Lernverfahren bedacht wurde, so schlägt die Methode fehl.

Auch existiert eine starke Abhängigkeit der Beleuchtung, da sich die Werte der Pixel, und daher auch die Abstände im Unterraum, verändern.

Hinzukommen noch Mimiken oder Veränderungen im Gesicht, die die Eigenface-Methode negativ beeinflussen können, wenn sie nicht in der Trainingsmenge enthalten sind.

Um diesen Abhängigkeiten entgegenzuwirken, bedarf es einiger Techniken. Die Bestimmung des Bereichs, in dem sich das Gesicht befindet, mit Hilfe eines Gesichtsdetektors und die Translation des Kopfes, kompensiert die Abhängigkeiten der Lokalität und der Orientierung.

Bei der Skalierungs- und Beleuchtungsabhängigkeit bedarf es Normierungen. So lässt sich die Größe einheitlich skalieren. Bei der Beleuchtung existieren verschiedene Verfahren mit unterschiedlichen Ansätzen und Komplexitätsgraden. Eins der schnellsten und einfachsten Verfahren ist ein Histogrammausgleich des Bildes.

Diese Methoden schaffen es, den Abhängigkeiten der Eigenfaces entgegen zu wirken, jedoch nicht sie zur Gänze abzuschaffen.

Analyse der Eigenface-Methode

Die Eigenface-Methode ist ein etabliertes Verfahren und ist in mehreren Studien auf ihre Stabilität hin untersucht worden. Daher lassen sich die Erfahrungen aus der Literatur zitieren, die in ihren Studien die Eigenface-Methode auf ihre Robustheit hin untersuchen.

In diesen Studien wird die Robustheit der Methode unter verschiedenen Bedingungen getestet und ihre Veränderung gegenüber der Umgebung analysiert. In diesen Tests werden besonders die Abhängigkeiten der Eigenface-Methode (s. Kapitel 2.3) betrachtet und es werden Abschätzungen getroffen, im welchen Grad die Ergebnisse von den Veränderungen beeinflusst sind.

Da die Abhängigkeiten der Eigenface-Methode sehr stark sind, wir häufig die Wirkung von verschiedenen, unterstützenden Techniken analysiert. So zum Beispiel der Einsatz von Gaußpyramiden zur Begegnung der Skalierungsabhängigkeit.

Die Studien zeigen an Hand von Rund 2500 Bildern von 16 Personen in einer Datenbank, die unter gleichen Bedingungen und Aufnahmepositionen aufgenommen worden sind, dass bei einer Veränderung der Beleuchtung die Eigenface-Methode rund 96% der Gesichter erkennt, bei einer variabler Ausrichtung des Kopfes 85% und bei verschiedenen Skalierungen 64%. [MP01]

Bei den Aufnahmen der Bilder wurden von den jeweiligen Gesichtern drei Kopfstellungen und -größen fotografiert, unter jeweils drei Beleuchtungsverhältnissen. Dabei liegen die Bilder in der Datenbank als sechsstufige Gaußpyramiden vor, bei einer Auflösungen von 16 x 16 bis zu 512 x 512 Pixel. Das bedeutet von jedem Gesicht in der Datenbank existieren 162 Bilder.

Andere Studien, die verschiedene Techniken und Auswertungsverfahren berücksichtigen (wie z.B. die Mahalanobis-Distanz), ergeben ähnliche Ergebnisse, jedoch mit einer Fehlerrate von 19% bei wechselnden Lichtverhältnissen und 39% und 60% bei anderer Orientierung und Größe. [Jor08]

Das Eigenface-Verfahren ist eine gute Methode, die speichereffizient und performant eine Gesichtserkennung bietet. Die Implementierung im Labor liefert zudem eine relativ statische Umgebung, in der die Abhängigkeiten, im Aspekt der Beleuchtung und der Verwendung der Kamera, weitestgehend abgemildert werden.

Kapitel 3

Eigener Ansatz

Der von mir untersuchte Ansatz portiert die Balancetheorie auf eine Anwendung in der Robotik im Bereich der Mensch-Maschinen-Interaktion.

Durch die Analyse der Bilddaten unter dem Aspekt der Balancetheorie, kann der Roboter soziale Interaktionen erkennen und zu einem gewissen Grad deuten. Durch die Analyse ändert sich die eigene Haltung des Roboters zu den Personen und soll so der Maschine ermöglichen, Teilnehmer am sozialen Geschehen zu sein.

Im Ansatz wird die Balancetheorie auf die Anwendung portiert, basierend auf der Detektion und die Länge der Blickkontakte. Die Blickkontakte geben Hinweise, ob sich die Personen im Bild mögen oder nicht.

Die Übertragung der Balancetheorie erfordert eine Reihe von Annahmen und Einschränkungen, da die Theorie sich nicht trivial auf eine Anwendung übertragen lässt. Der Roboter kann nicht im selben Maße die Menschen beeinflussen, wie Menschen sich gegenseitig beeinflussen können.

Um die Bedingungen für das Labor und den Roboter anzupassen, wurden folgende Annahmen und Einschränkungen festgelegt:

- Der Roboter ist passiver Teilnehmer an dem beobachteten Geschehen,
- der Roboter kann die beobachteten Personen nicht beeinflussen,
- die Bestimmung der Präferenzen geschieht lediglich über Blickkontakt und Zeit,
- der Roboter besitzt keine Hintergrundsinformationen über die Personen,
- wenn eine unbalancierte Relation vorliegt, so wird sie nur über interne Entscheidungsprozesse des Roboters gelöst oder von Verhaltensänderungen der beobachteten Personen und
- im Gegensatz zur Balancetheorie wird der Fall, dass alle beteiligten Elemente sich nicht mögen, als ein stabiler Zustand angenommen.

Diese Annahmen fokussieren die Anwendung auf ein engeres Gebiet, in dem der Roboter agieren kann. Allerdings muss beachtet werden, dass diese Einschränkungen in die psychologische Basis der Balancetheorie eingreifen. Jedoch halten sich diese Eingriffe in Grenzen, da die Balancetheorie die Voraussetzung der Elemente hauptsächlich auf der Konsistenztheorie stützt. Der Roboter hat an sich kein inhärentes Streben nach Konsens, dennoch kann der Versuch, die Strukturen in ein harmonisierten Zustand zu führen, im Rahmen der Möglichkeiten des Roboters, als ein solcher Impuls verstanden werden.

Es sollte aber nicht vergessen werden, dass der Roboter durch seine Passivität nicht alle Möglichkeiten der Beeinflussung ausschöpfen kann, die einem Menschen zur Verfügung stehen. Dadurch könnte ein Mensch, in dieser Lage, unter Umständen anders reagieren als es durch die Balancetheorie dargestellt wird. Das ist ein spezialisierter Fall und die Untersuchung, ob dieser wirklich eintreffen könnte, würde den Rahmen dieser Arbeit überschreiten.

Zudem wurde im Rahmen der Arbeit die Balancetheorie durch *transitive Relationen* erweitert. Diese erlauben eine Analyse von größeren Gruppen als sie durch die reine klassische Heidersche Theorie festgelegt sind. Ohne die transitiven Relationen kann der Roboter lediglich zwei Personen im Bild analysieren und das würde die Flexibilität des Systems erheblich einschränken. Auch widerspricht es der Erwartungshaltung des Betrachters, der nicht erwarten würde, dass keine Reaktion auf eine dritte Person erfolgt.

Die transitiven Relationen verändern den Kern der Balancetheorie nicht, sondern erweitern sie durch Regeln, die auf der Triaden-Struktur der Balancetheorie aufbauen. Die Relationen erlauben, Strukturen mit mehreren Zwischenelementen zu verfolgen und den Gesamteinfluss auf das Endelement zu untersuchen. Die transitiven Relationen erhöhen die Flexibilität der Anwendung, aber ihre psychologische Grundlage muss erst noch nachgewiesen werden. Allerdings würde eine solche Untersuchung ebenfalls über den Rahmen dieser Arbeit hinausgehen.

Ebenfalls wurde für die Arbeit eine Gesichtserkennung implementiert, die auf der *Eigenface-Methode* beruht, mit deren Hilfe die zwei Augen des Roboters das selbe Gesicht fixieren können, um so einen natürlicheren Effekt zu erzeugen. Zudem bietet die Gesichtserkennung ein Ansatz, die Analyse der Beziehungen der Gesichter über das Kamerabild hinaus zu verfolgen. Die Wiederkennung des Gesichts bietet somit die Möglichkeit, die Analyse der sozialen Strukturen weiter fortzuführen.

Dieser Ansatz ist auch Basis der *Recognition-Pipeline*, die ebenfalls in der Arbeit entwickelt wurde. Die Recognition-Pipeline analysiert in mehreren Threads die Gesichter auf dem Kamerabild und sammelt bei Bedarf Bilder von den Gesichtern und fügt sie online der Datenbank hinzu. Dadurch ist es möglich die Analyse fortzuführen und die Datenbank zu vervollständigen, ohne das System anzuhalten oder zu beeinträchtigen.

Kapitel 4

Implementation

In diesem Kapitel wird die Implementierung der Arbeit vorgestellt. Hierbei wird insbesondere die Aufmerksamkeit auf die Architektur gerichtet. Als Programmiersprache wurde C++ gewählt. Zusätzlich wurden für das Projekt die Bibliotheken OpenCV[1], Boost[2], Okao-Vision der Firma Omron[3] und die Kontroll-Schnittstellen von CB[2] aus dem ERATO-Labor der Osaka Universität[4] verwendet. Die Implementation ist für eine Windows Plattform entwickelt worden.

Bei der Entwicklung der Architektur wurde Wert darauf gelegt, dass das System modular und möglichst flexibel verwendbar ist. Die Systemkomponenten können voneinander separat verwendet und unterstützend mehrfach angewendet werden. So kann das System zum Beispiel mehrere Kameras gleichzeitig verwenden.

Zudem sind die verschiedenen Bestandteile Multi-Thread-fähig und demnach sind die systemkritischen Stellen Thread-sicher entwickelt worden.

4.1 In- und Output-Verwaltung

Für die Verwaltung der Kamera und der Ausgabe auf dem Bildschirm wird die Schnittstelle der OpenCV-Library verwendet und durch eine objektorientierte Implementierung erweitert.

OpenCV bietet eine Vielzahl an Methoden für die Bildverarbeitung. Allerdings sind viele Methode im C-Stil implementiert worden und erst in späteren Versionen wird eine objektorientierte Implementation angeboten.

Die Implementation für diese Arbeit soll eine möglichst flexible Schnittstelle für den In- und Out-Put bieten, um das System möglichst flexibel von der Hardware austauschen zu können, wenn der Roboter nicht zur Verfügung steht.

[1] OpenCV, Version 1.0
[2] Boost-Library, Version 1.37
[3] Okao-Vision, Version 4.2
[4] Bereitgestellt von Takashi MINATO und Hidenobu SUMIOKA aus dem ERATO-Labor

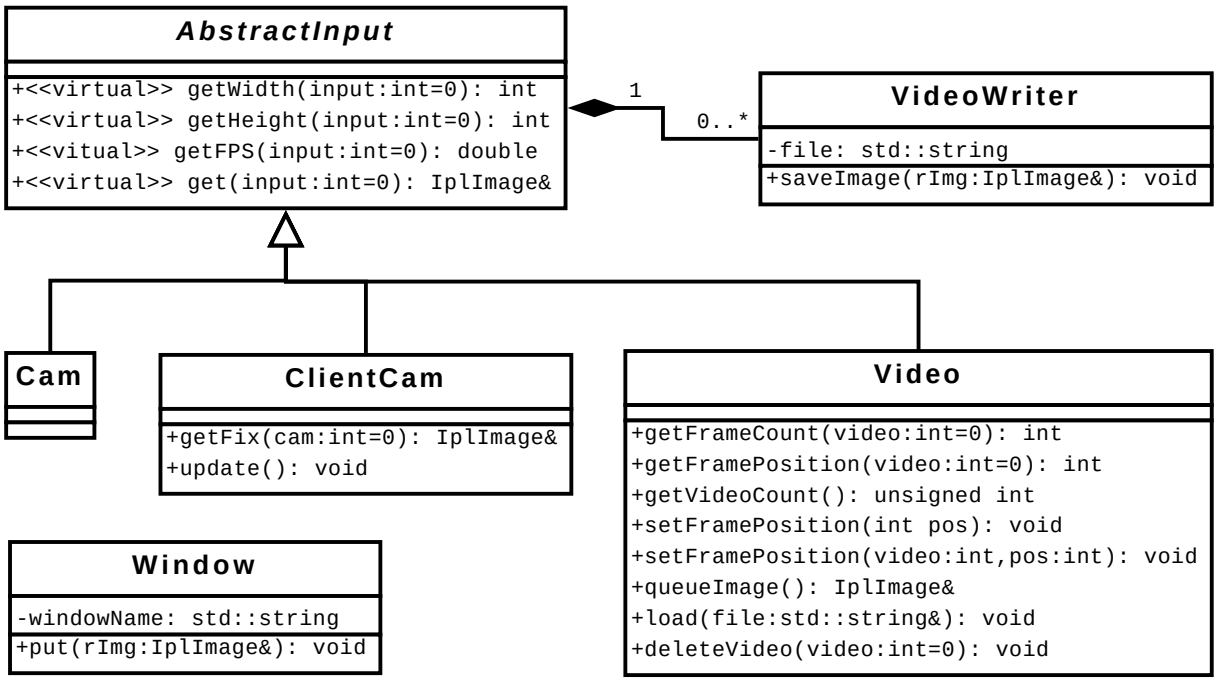


Abbildung 4.1: UML-Schema der In- und Output-Verwaltung

Wie im UML-Schema in Abbildung 4.1 zu erkennen ist, werden die Klassen `Cam`, `ClientCam` und `Video` jeweils von `AbstractInput` abgeleitet. Das Hauptmotiv für diese Entwicklung liegt in der Polymorphie. Durch die flexible Schnittstelle kann der Input an vielen Stellen aus verschiedenen Quellen kommen und während der Laufzeit modifiziert werden, ohne große Mengen des Programmcodes zu ändern.

Eine Anwendung der Polymorphie findet sich im selben UML-Schema wieder, nämlich bei der Klasse `VideoWriter`. `VideoWriter` hat zur Aufgabe, die mit der `saveImage()`-Methode übergebenen Bilder in einem Videoformat zu speichern. Zur Speicherung benötigt `VideoWriter` verschiedene Angaben der Input-Quelle, wie zum Beispiel die FPS (engl. Frames Per Second), die angeben, wie schnell die Bildreihenfolge abgespielt wird.

Die Angabe ist so inhärent, dass das `AbstractInput`-Objekt dem `VideoWriter`-Konstruktor übergeben wird. Somit wird verhindert, dass eine Video-Datei widersprüchliche Attribute erhält.

Die Input-Klassen sind auf verschiedene Arten von Eingangsdaten spezialisiert. `Cam` nimmt das Videobild einer angeschlossen Webcam an, wobei mehrere Webcams verwaltet werden können, durch weitere `Cam`-Objekte. Die einzelnen Webcams können durch ein Flag bei Konstruktor zugeordnet werden, nach der Vorlage der entsprechenden OpenCV-Dokumentation.

`Video` ist entsprechend zur `Cam`-Klasse entwickelt worden, mit dem Unterschied, dass Video-Dateien ausgelesen werden.

`ClientCam` ist im Zuge der Implementierung mit CB^2 entstanden. `ClientCam` ruft die Bilddaten von einem Server ab, der mit den Augen des Roboters verbunden ist.

Der Output kann, neben der Speicherung als Video-Datei (s. `VideoWriter`), in mehreren Fenstern auf dem Bildschirm ausgegeben werden durch die Klasse `Window`.

4.2 Gesichtstracking auf dem Video-Bild

Die Verwaltung der Gesichter im Kamerabild hat mehrere Aufgaben zu erfüllen, zum einen müssen die Gesichter auf dem Videostream verfolgt werden und es muss registriert werden, ob ein Gesicht verloren wurde. Zum anderen sind die Informationen über das Gesicht zu aktualisieren.

Dabei ist es wichtig zu bestimmen, welche Informationen über das Gesicht verwaltet werden. Ziel der Implementation der Gesichtsverfolgung ist es, ein modulares System zu entwickeln, das möglichst abgekapselt, die Gesichter auf dem Kamerabild verfolgt und die Informationen über die Gesichter angleicht. Auch sind weitere System-Elemente vom Tracking abhängig, wodurch es in der Lage sein muss, mehrere Anwendungen über Veränderungen der Gesichter zu informieren.

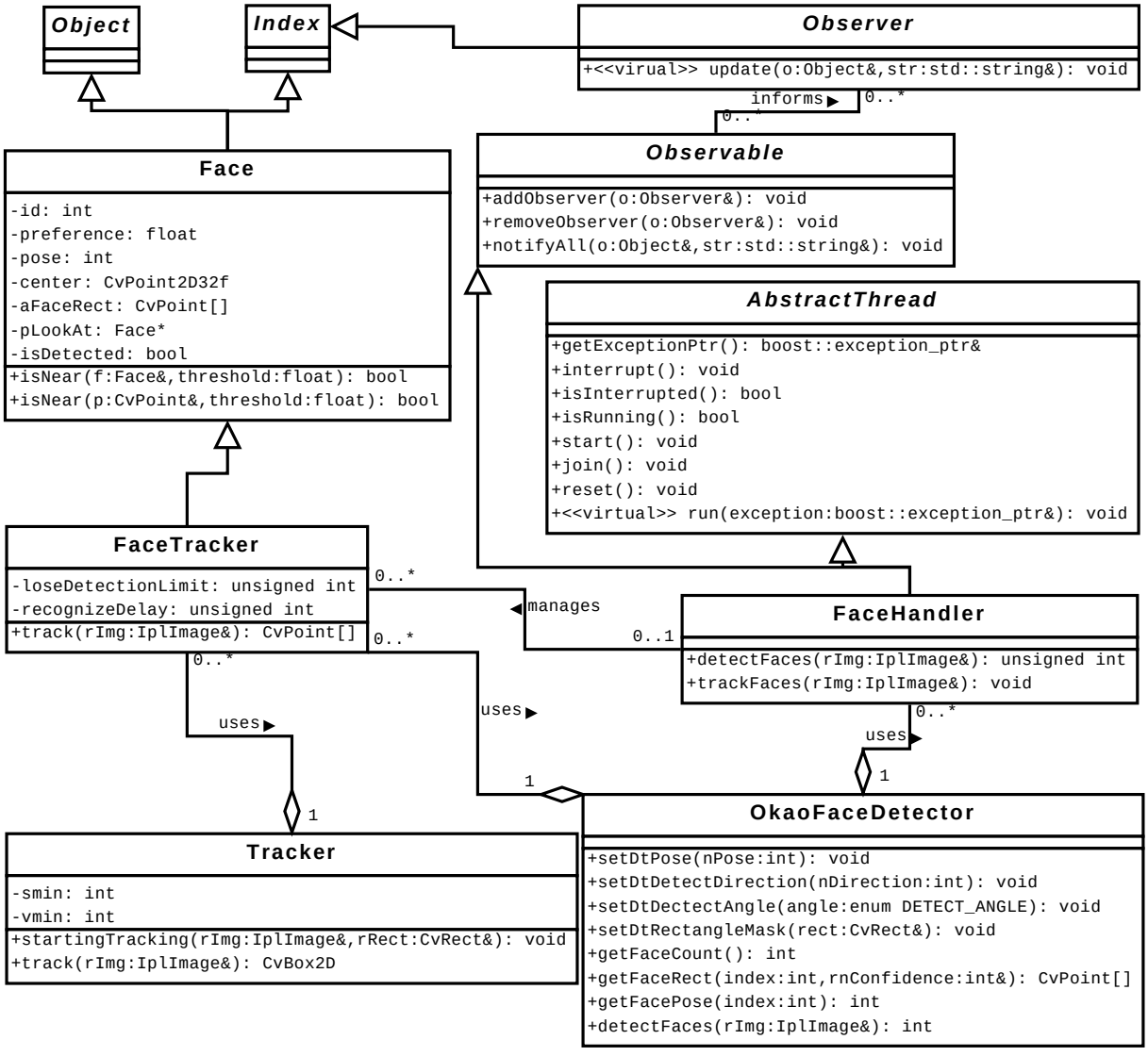


Abbildung 4.2: UML-Schema des Gesichtstrackings.

In Abbildung 4.2 ist ein UML-Diagramm des Systems zur Gesichtsverfolgung abgebildet. Das elementare Datenobjekt beim Gesichtstracking ist die Klasse `Face`, das als Attribute, die zu sammelnden Informationen über das Gesicht enthält.

Die Informationen, die über das Gesicht gesammelt und verwaltet werden, sind:

- die Identität,
- die Präferenz des Roboters zu dem Gesicht,
- die Position,
- die Bounding-Box des Gesichts im Bild,
- die Ausrichtung des Gesichts und
- das Gesicht, auf welches das jeweilige Gesicht blickt (und null ist, wenn es auf kein Gesicht schaut).

Um die Informationen in jedem Bild zu aktualisieren, ist die Klasse erweitert worden durch `FaceTracker`. Das Tracking erfordert große Robustheit gegen Störungen. Daher verwendet `FaceTracker` sowohl einen Gesichtsdetektor, als auch einen Farbtracker.

Ziel des Trackings ist es, das Gesicht auf dem Kamerabild zu finden und die Daten zu aktualisieren, wozu es den Detektor benötigt. Allerdings kann es passieren, dass das Gesicht auf einigen Bildern nicht mehr zu erkennen ist. Sei es, dass etwas kurzzeitig ins Gesicht fährt wie Haare oder eine Hand. In diesen Fällen verliert der Detektor das Gesicht. Um es dennoch weiter verfolgen zu können, wird der Bereich, in dem das Gesicht zuletzt gesichtet wurde, auf Farbähnlichkeit hin untersucht.

Das hat einen deutlich stabilisierenden Effekt, jedoch können unerwünschte Nebeneffekte entstehen. So mag das Feld des Gesichts sich stark verzerren. Auch kann der Tracker Areale verfolgen, in denen kein Gesicht mehr vorhanden ist, da er lediglich auf Farbähnlichkeit zum vorherigen Bild achtet.

Daher werden einige Bedingungen mit dem Tracking verknüpft. Zum Beispiel wird ein Limit (engl. Time-Out) gesetzt, das die Anzahl der Frames beschränkt, in denen das Gesicht weiterhin nicht detektiert wird. Auch wird ein Toleranzbereich angegeben, der das Seitenverhältnis von Höhe und Breite des Gesichtsfeldes angibt.

Sollte eine der Bedingungen nicht mehr erfüllt sein, so wird das Gesicht als verloren erklärt und eine `LostFaceException` wird ausgelöst, die von `FaceHandler` behandelt wird.

`FaceHandler` ist ebenfalls im UML-Diagramm (s. Abbildung 4.2) abgebildet und ist im Kern ein Vektor von `Face`-Objekten, der sich selbst verwaltet.

Durch die Methoden `detectFaces()` und `trackFaces()` wird untersucht, ob sich neue Gesichter auf dem übergebenen Bild befinden, bzw. die alten Gesichter werden nach dem Bild aktualisiert. Wird von einem `FaceTracker`-Objekt eine `LostFaceException` geworfen, so wird das betreffende Gesicht aus dem Vektor gelöscht

Zudem ist `FaceHandler` von zwei Klassen vererbt worden: `Observable` und `AbstractThread`. `Observable` orientiert sich am Observer-Pattern und ermöglicht es Beobachtern (engl. Observer) von Ereignissen des `FaceHandler`-Objekts informiert zu werden. In diesem Anwendungsfall wird das Hinzufügen oder Löschen von Gesichtern als Ereignis verstanden.

`AbstractThread` ermöglicht, nach Java-Vorlage, eine Thread-fähige Implementierung. Dadurch ist es möglich, die `run()`-Methode des `FaceHandler`-Objekts mit dem Aufruf der `start()`-Methode in einem separaten Thread laufen zu lassen. Das bietet sich vor allem an, da die Augen des Roboters von einander unabhängig agieren und die Verfolgung der Gesichter zum großen Teil voneinander getrennt abläuft.

4.3 Bestimmung des Blickkontakts

Durch das Tracking-System der Gesichter, wird die Verfolgung der Gesichter übernommen, jedoch fehlt noch die Bestimmung der Blickkontakte unter den Gesichtern. Das Ziel ist ein einfaches, robustes und schnelles Verfahren zur Bestimmung der Blickkontakte zu entwickeln.

Für die Bestimmung wird die Bounding-Box der Gesichter verwendet. Mit Hilfe der Angabe der Kopfausrichtung des Okao-Gesichtsdetektors der Firma Omron wird ermittelt, ob das Gesicht nach links, rechts oder zur Kamera ausgerichtet ist.

Der Blickkontakt, ob nun ein Gesicht zu einem anderen Gesicht schaut, wird durch die Orthogonale der Bounding-Box in Blickrichtung des Kopfes bestimmt (s. Abbildung 4.3).

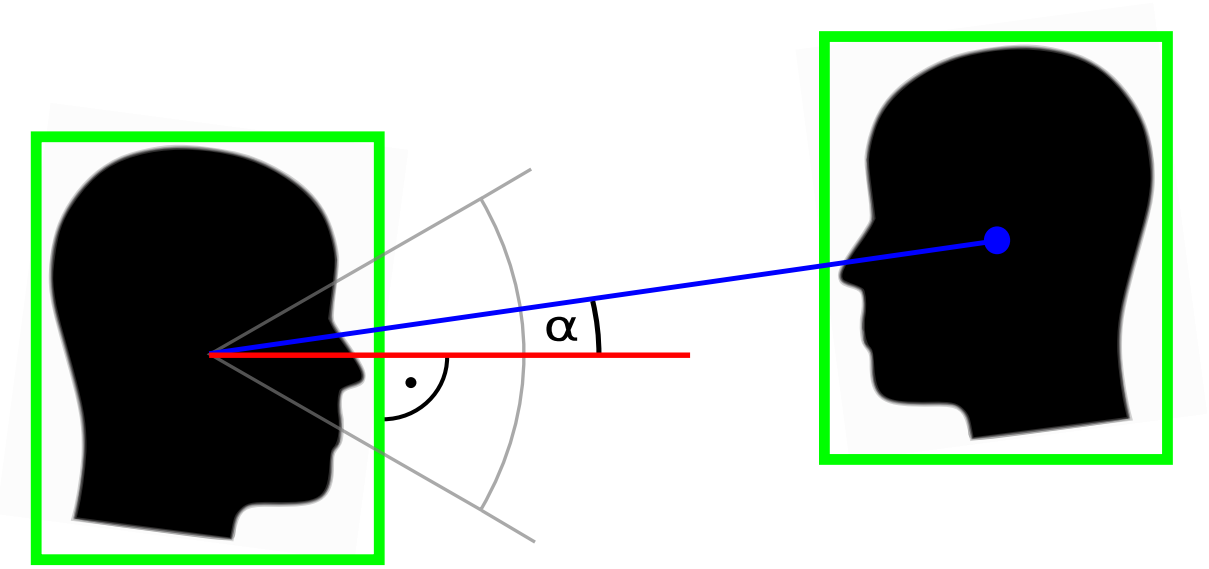


Abbildung 4.3: Schematische Darstellung zur Bestimmung des Blickkontakts. In der Abbildung wird der Blickkontakt festgestellt, da α sich im Bereich des angegebenen Blickkegels um die Orthogonale befindet.

In Abbildung 4.3 ist die schematische Darstellung zur Berechnung des Blickkontakts dargestellt. Zunächst wird vom Detektor die Ausrichtung des Kopfes, die die Richtung der Orthogonale angibt, festgestellt. Somit wird an der Kante der Bounding-Box ein rechtwinkliger Vektor gezogen, dessen Ursprung sich in der Mitte der Box befindet. Ein weiterer Vektor, vom Mittelpunkt der einen Bounding-Box zur anderen, wird berechnet und der Winkel zwischen dem ersten und dem zweiten Vektor wird nach Formel 4.1 ermittelt. Wenn der errechnete Winkel unter einem bestimmten Schwellwert liegt, so wird der Blickkontakt festgestellt. In Abbildung 4.3 ist der Bereich des Schwellwerts als Blickkegel dargestellt.

$$\measuredangle(\vec{A}, \vec{B}) = \arccos \frac{\vec{A} \cdot \vec{B}}{|\vec{A}||\vec{B}|} \tag{4.1}$$

Die Beziehungen, wer zu wem guckt, wird durch einen Pointer dargestellt, den jedes `Face`-Objekt besitzt. Der Pointer gibt an, zu welchem Gesicht das jeweilige schaut und ist null, wenn auf kein Gesicht geblickt wird. Dadurch entsteht eine gerichtete Graphen-Struktur, die iteriert werden kann.

Die Setzung und Verwaltung der Pointer übernimmt eine erweiterte `FaceHandler`-Klasse, namens `FaceLookAtHandler`, die sich ebenso wie `FaceHandler` in einem unabhängigen Thread ausführen lässt. Dadurch ist es möglich, je nach Bedarf, ein Auge für die Blickkontakte zu verwenden, während das andere lediglich die Gesichter verfolgt und so für eine effiziente Berechnung und ein natürliches Verhalten sorgt.

4.4 Bestimmung der Präferenz

Die Bestimmung des *Präferenzwertes* ist ein zentraler Bestandteil der Arbeit und gibt die Zuneigung des Roboters zu der Person im Kamerabild an. Ein Schwerpunkt in der Implementierung ist es, dem System zu ermöglichen, eine beliebig große Anzahl an Gesichtern zu verfolgen und auszuwerten. Dazu kommt, dass die Berechnung möglichst schnell und modular erfolgen soll. Dafür eignet sich die Kommunikationsstruktur, wie sie das Entwurfsmuster des Observer-Patterns ermöglicht.

Dazu muss zunächst die Theorie der Balancetheorie erweitert werden, da sie auf eine Gruppengröße von bis zu drei Personen festgelegt ist. Dazu wurden im Rahmen der Arbeit die *transitiven Relationen* entwickelt, die, aufbauend auf der klassischen Balancetheorie, große Gruppenrelationen analysieren.

Zunächst werden die einzelnen Berechnungsschritte erläutert.

Berechnung und Veränderung des Präferenzwerts

Die Präferenz wird mit einem Float-Wert, der in dem jeweiligen `Face`-Objekt als Attribut vorliegt, dargestellt. Der Wert gibt an, wie sehr der Roboter die Person mag oder nicht mag. Dabei wird die Präferenz als eine *Sigmoidfunktion* (s. Abbildung 4.4 und Formel 4.2) dargestellt.

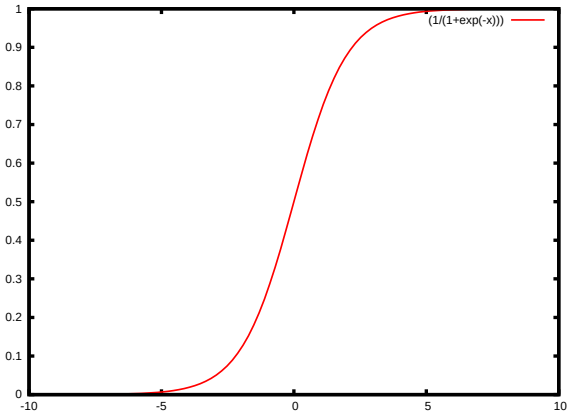


Abbildung 4.4: Darstellung der Sigmoidfunktion (s. Formel 4.2).

$$P(t) = \frac{1}{1 + \exp^{-t}} \tag{4.2}$$

Die Abbildung 4.4 stellt die Sigmoidfunktion, die einer S-Kurve entspricht, dar. Die y-Achse bildet den Präferenzwert ab. Dabei ist der Wert $0,5$ als ein „neutrales“ Gefühl zu verstehen, während ein größerer Wert „freundlich“ oder „mögend“ und ein kleinerer „unfreundlich“ oder „nicht mögend“ entspricht. Die Werteskala ist im Bereich zwischen 0 und 1 beschränkt, wodurch das „Gefühl“ sich nicht gegen unendlich entwickeln kann.

Die x-Achse stellt die erhaltenen Blickkontakte in Relation mit der Anzahl der Frames, in denen kein Blickkontakt erfolgt ist, gewichtet zusammen. Die Gewichtung dient zur Balancierung des Systems. Beispielsweise würde eine zu hohe Gewichtung, bei dem Fall, dass kein Blickkontakt erfolgt, verhindern, dass jemand über längere Zeit vom Roboter gemocht wird. Bei einer zu niedrigen Gewichtung, würde jede Person einfach gemocht werden.

Die Sigmoidfunktion ist gewählt worden, da sie eine einfache Struktur besitzt und im Wertebereich der y-Achse nach oben und unten hin beschränkt ist. Zudem approximiert die Funktion zu ihrem Maximum und Minimum hin. So verändert sich der Präferenzwert um den Wert $0,5$ sehr stark, jedoch mildert sich

diese Veränderung zu den Beschränkungen hin. Wenn einige Faktoren vernachlässigt werden, die von außen einwirken können, und dem Einfluss von Erfahrungen keine hohe Gewichtung zugesprochen wird, dann beschreibt die Sigmoidfunktion vereinfacht eine Gefühlsintensität.

Die Funktionsstellen um den Nullpunkt der x-Achse sind dementsprechend ein unvoreingenommenes Gefühl, das stark veränderlich ist. Zu dem Maximum bzw. Minimum hin verändert sich das Gefühl immer weniger. Wenn die Tendenz in die andere Richtung bei einem Minimum kehrt, so dauert es „länger“ bis dieser Wert stärker steigt als bei dem Nullpunkt der x-Achse. Wenn der Roboter jemanden nicht mag, dann dauert es länger bis er wieder anfängt, ihn zu mögen.

Die Sigmoidfunktion stellt eine Erweiterung der klassischen Balancetheorie dar, da sie Aussagen über die Intensität der Beziehungen der beteiligten Elemente trifft.

Die Beeinflussung des Präferenzwertes geschieht über Blickkontakte. Dabei sind vier Gesichtspunkte interessant:

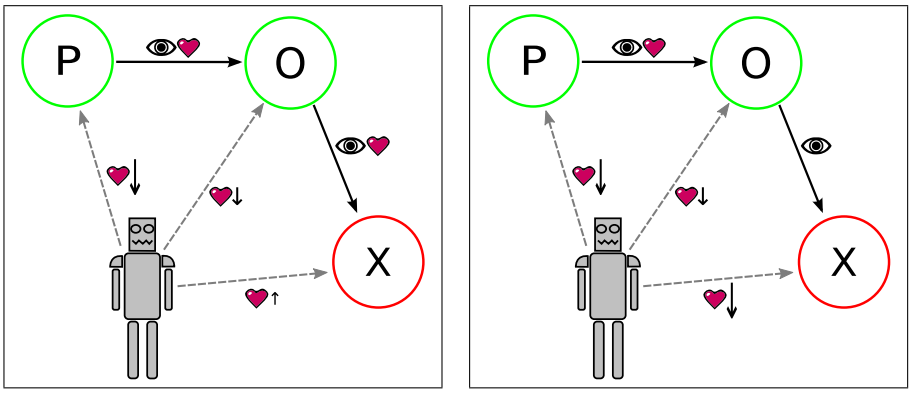
- Wie lange schauen die Personen den Roboter an,
- wie lange schauen die Personen sich gegenseitig an,
- wie lange schauen die Personen den Roboter nicht an und
- wie lange schauen die Personen sich nicht an.

Der zweite und vierte Punkt sind deswegen interessant, weil sie die Grundlage der Glaubensvorstellung des Roboters, wer wen mag, repräsentiert. Das beeinflusst die Präferenzwertvergabe im Hinblick auf die Balancetheorie (s. Kapitel 2.2).

Des Weiteren wurde im Rahmen der Implementierung die *transitiven Relationen* eingeführt, um so eine beliebig große Menge an Personen untersuchen zu können, da die klassische Balancetheorie nur maximal drei Elemente berücksichtigt.

Die transitiven Relationen beeinflussen die Präferenzwerte der Gesichter durch „indirekte“ Blickkontakte. So beeinflusst der Blickkontakt einer Person, die der Roboter mag, ein anderes Gesicht positiv (allerdings nur, wenn der Roboter merkt, dass die Person die andere Person auch mag).

Die Transitivität der Relationen kann auch für längere Ketten von Blickkontakten verwendet werden. Hingegen nimmt aber der positive Einfluss der Blickkontakte auf den Präferenzwert mit jedem Iterationsschritt stetig ab.



(a) Transitive Relation I (b) Transitive Relation II (unterbrochen)

Abbildung 4.5: In Abbildung 4.5(a) erhält **X** durch **P** und **O** jeweils einen positiven Einfluss. In Abbildung 4.5(b) hingegen wird der Einfluss auf **X** unterbrochen, weil **O** **X** nicht mag.

In Abbildung 4.5 sind zwei Relationen mit jeweils vier Elementen abgebildet. Die Figur in der linken unteren Hälfte stellt den Roboter dar, während die Knoten **P**, **O** und **X** die Personen im Kamerabild repräsentieren. Die Bilder stellen die Auswirkungen der transitiven Relationen dar. In den Bildern wird **O** von **P** und **X** von **O** angeschaut, das vom Roboter registriert wird. **O** und **P** werden vom Roboter gemocht, während die Präferenz zu **X** unter dem erforderlichen Schwellwert liegt.

In Abbildung 4.5(a) ist zu sehen, dass die Präferenz zu **X** anwächst, obwohl kein Blickkontakt auf den Roboter erfolgt. Das erklärt sich durch das Addieren der positiven Einflüsse von **P** und **O**. Der Roboter merkt, dass **X** zu einer Gruppe gehört und nimmt an, dass wenn **P O** mag und **O X** mag, dass auch **X** von **P** gemocht wird. Diese Annahme kann über beliebig viele Knoten erfolgen, allerdings nimmt die positive Beeinflussung mit der Iteration ab, daher wird ein indirekter Blickkontakt nicht so positiv gewertet wie ein direkter.

In Abbildung 4.5(b) ist die Fortsetzung des Einflusses von **P** unterbrochen, da **X** nicht von **O** gemocht wird. So kann nicht mehr die Annahme getroffen werden, dass **X** der Gruppe zugehört.

Der Aufwand der Berechnung wird gering gehalten, durch einige Vereinfachungen und Beschränkungen, die ebenfalls ein nicht-terminierendes Verhalten verhindern.

So werden bei der Traversierung der Graphenstruktur, die Gesichter vernachlässigt, die der Roboter nicht mag, und negative Relationen werden ebenfalls bei Iteration nicht weiter fortgepflanzt. Somit wird ein erhebliches Maß der Berechnung reduziert.

Um die Nicht-Terminierung zu vermeiden, müssen die Zyklen weiter betrachtet werden. Zum einen werden Zweier-Zyklen vermieden, in dem am Anfang der Iteration überprüft wird, dass kein gegenseitiger Blickkontakt vorliegt. Größere Zyklen lassen sich durch mehr Aufwand vermeiden, jedoch wird in diesem Fall, die maximale Iterationstiefe durch die Anzahl der Gesichter im Bild bestimmt, womit ein nicht-terminierender Zyklus verhindert wird.

Durch die Kombination dieser Berechnungsmethoden und Theorien, ist es möglich, die modifizierte Balancetheorie auf eine beliebig große Gruppe anzuwenden.

Implementierung der Präferenzwertvergabe

Die Implementierung gießt die vorgestellten Vorgaben zur Berechnung der Präferenzwerte in die Software-Architektur. Dabei ist weiter zu beachten, dass das System wieder modular vorliegen und möglichst selbständig agieren soll.

Die Vergabe der Präferenzwerte erfolgt mit der Hilfe der Klassen `Preferencer` und `FacePreference-Wrapper` (s. Abbildung 4.6).

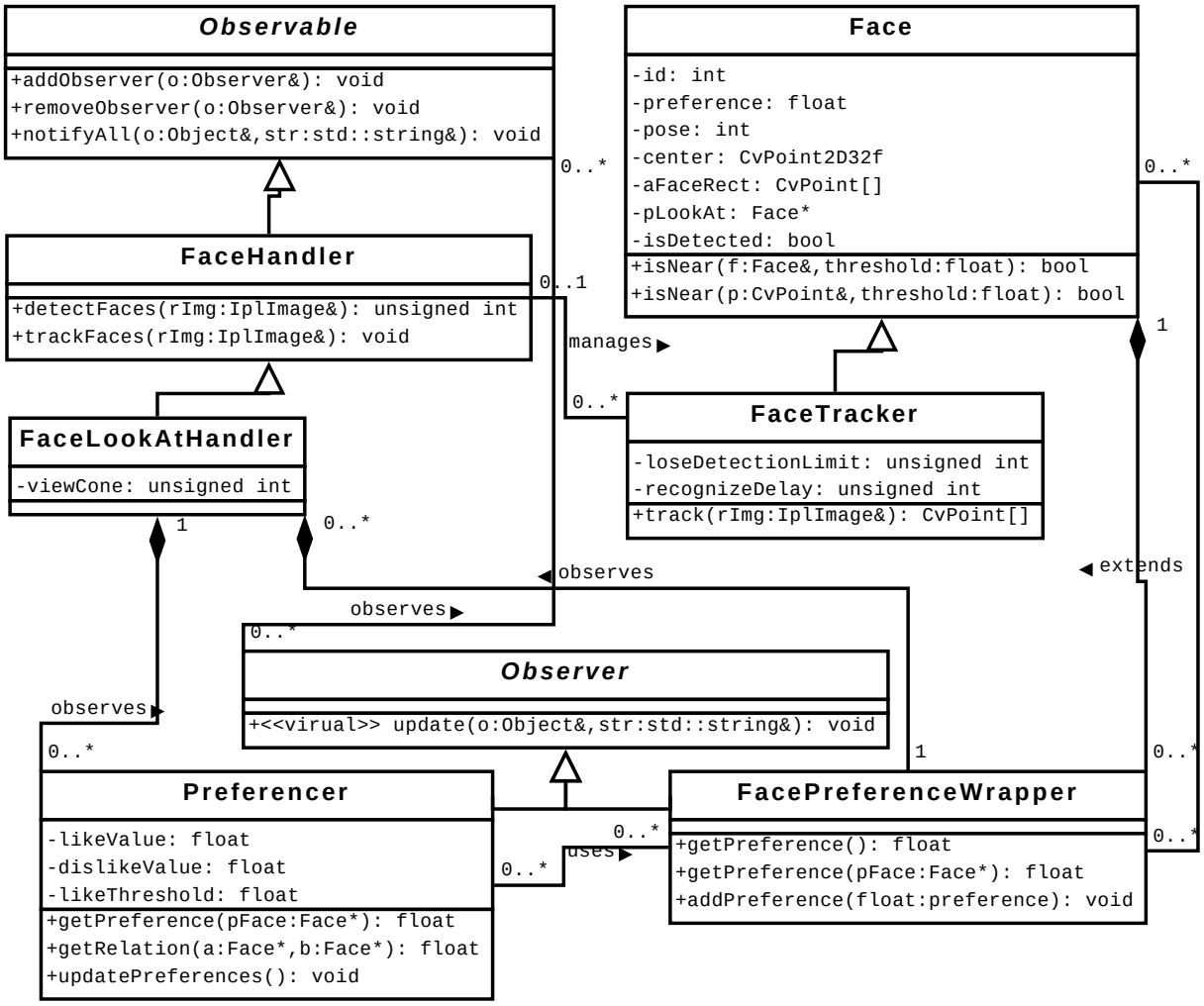


Abbildung 4.6: UML-Schema zur Präferenz-Berechnung.

In Abbildung 4.6 ist das UML-Diagramm für die Präferenz-Berechnung zu sehen. Auch sind, für die Implementierung wichtige, Klassen enthalten, die in Abbildung 4.2 ebenfalls dargestellt sind.

`Preferencer` ist die zentrale Klasse, die die Berechnung durchführt. `FacePreferenceWrapper` ist eine Wrapper-Klasse, dessen Objekte jeweils ein `Face`-Objekt umschließen. Für die Koordination der Informationen zwischen den Systembestandteilen, die für die Berechnung der Präferenzwerte verantwortlich sind, und den Teilen, die für das Gesichtstracking verantwortliche sind, sind `Preferencer` und `FacePreferenceWrapper` von `Observer` spezialisiert. Durch diese Verbindung melden sich beide Klassen bei einem `FaceLookAtHandler`-Objekt als `Observer` an. Wenn ein Gesicht hinzugefügt oder gelöscht wird, gleichen die `Preferencer`- und `FacePreferenceWrapper`-Objekte ihre Daten an.

`Preferencer` und `FacePreferenceWrapper` sind Beobachter im Sinne des Observer-Patterns und führen jeweils intern eine Liste über die vorhandenen Gesichter eines `FaceLookAtHandler`-Objekts. Beim Aufruf der `updatePreferences()`-Methode, wird der Graph der Blickkontakte der Gesichter traversiert und die Präferenzwerte dementsprechend verändert.

Dabei läuft die direkte Berechnung der Präferenzwerte vom Roboter zu den Gesichtern in drei Schritten, und die der Gesichter untereinander in zwei Schritten ab.

Im Detail, wie die Präferenz ausgewertet wird, ist hier im Pseudo-Code dargestellt. Der erste Schritt wertet die Blickkontakte der Personen zum Roboter direkt aus.

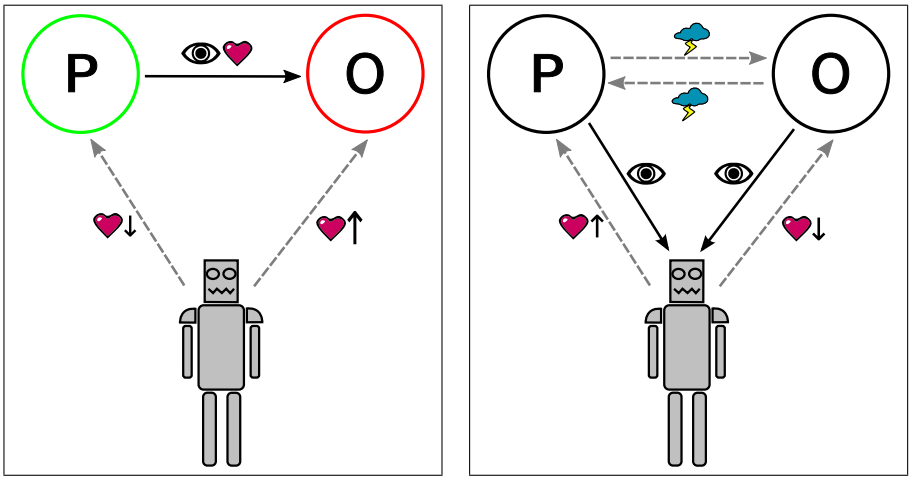
```
likeValue ← 0.1
dislikeValue ← 0.01

for all f ∈ Faces do
  if f looks at camera then
    f.preference += likeValue
  else
    f.preference -= dislikeValue
  end if
end for
```

Im zweiten Schritt werden indirekte Einflüsse ausgewertet, d.h. Blickkontakte von Personen, die der Roboter mag. Hierbei werden allerdings nur positive Relationen betrachtet.

```
likeValue ← 0.1
dislikeValue ← 0.01

for all f1 ∈ Faces do
  fstFace ← f1
  count ← 0

  f2 ← f1.lookAt
  while f2 exists && f2 != fstFace && f1 likes f2 && count ≤ number of faces do
    count + +
    f2.preference += likeValue/count · 2

    f1 ← f2
    f2 ← f2.lookAt
  end while
end for
```

Der dritte Schritt der Iteration analysiert die Struktur des Graphen nach unbalancierten Strukturen, so wie sie von der Balancetheorie vorgegeben sind, und versucht durch passive Entscheidungsprozesse des Roboters diese Strukturen in balancierte Strukturen umkehren zu lassen.

Hierbei ist zu beachten, dass der Roboter keinerlei Möglichkeiten besitzt, die Personen im Kamerabild aktiv zu beeinflussen. Es existieren daher zwei Formen von unbalancierten Strukturen, die berücksichtigt werden, die in Abbildung 4.7 abgebildet sind.



(a) Unbalanciert Struktur I (b) Unbalanciert Struktur II

Abbildung 4.7: Darstellung der möglichen unbalancierten Strukturen. Die Struktur in Abbildung 4.7(a) existiert noch in gespiegelter Form. Abbildung 4.7(b) stellt ein Entscheidungsprozess dar, in diesem Fall zu Gunsten von **P**.

In Abbildung 4.7 sind zwei unbalancierte Strukturen mit jeweils drei Elementen zu sehen. Das untere Element stellt den Roboter da, der je eine Relation, die die Zuneigung repräsentiert, zu den beiden anderen Elementen hält. In der Abbildung 4.7(a) mag der Roboter **O** nicht, aber dafür **P**, der wiederherum **O** mag. Jedoch ist das, laut der Balancetheorie, ein unbalancierter Zustand. Die Veränderung zu einem balancierten Zustand erfolgt zum einen, in dem **P** nicht mehr zu **O** sehen würde, und somit der Roboter irgendwann denkt, dass **P** **O** ebenfalls nicht mehr mag. Zum anderen kann der Roboter anfangen **O** zu mögen, wenn dieser zum Roboter sieht oder wenn **P** weiter auf **O** blicken würde und somit die Präferenz sich von **P** auf **O** weiter überträgt.

Die zweite Struktur in Abbildung 4.7(b) ist hingegen schwieriger zu lösen und bedarf einer Entscheidung vom Roboter. Der Roboter mag beide Elemente, jedoch merkt er, dass keiner der beiden den anderen mag. Die Entscheidung ist dahingehend, wer weiterhin vom Roboter gemocht wird oder wer nicht. Dabei kann es bis zu vier Entscheidungsinstanzen geben, in denen der Roboter abwägt. Dadurch wird der unbalancierte Zustand in einen balancierten überführt, wenn keine Verhaltensänderung von den Elementen erfolgt.

Die Kriterien werden der Reihe nach behandelt. Wenn ein Gesicht gegenüber dem anderen eins der Kriterien erfüllt, wird das Entscheidungsverfahren abgebrochen und das andere Gesicht wird in der Präferenz negativ bewertet.

Die Kriterien, ob nun das Gesicht einen positiven oder negativen Wert bekommt sind:

1. Das Gesicht besitzt eine ID-Nummer und ist somit in der Datenbank enthalten,
2. das Gesicht besitzt im Gegensatz zum anderen Gesicht einen derzeitig höheren Präferenzwert,
3. das Gesicht wird von mehr Leuten gemocht als das andere Gesicht oder
4. es wird per Zufall entschieden, welches Gesicht die negative Wertung erhält.

Wenn der Roboter den Entscheidungsprozess ausgewertet hat, wird der Präferenzwert des Gesichts mit dem `dislikeValue`-Wert abgezogen. Wenn keine Änderung des Zustandes von den beobachteten Personen ausgeht, so wird eine Person nach einer gewissen Zeit nicht mehr vom Roboter gemocht. Dieser Vorgang führt dazu, dass die unbalancierte Struktur sich zu einer balancierten ändert.

Die Bestimmung der Präferenzwerte zwischen den Gesichtern geschieht vor der Auswertung der indirekten Blickkontakte. Es dient dazu, die Szene zu analysieren, um festzustellen, welche Person wen mag.

Für diesen Schritt wird für jedes Gesicht der Schritt 1 und 2 wiederholt. Demnach die Auswertung der Blickkontakte vom Gesicht aus und die indirekte Fortpflanzung der Präferenz. Jedoch wird nicht ein Entscheidungsprozess wie in Schritt 3 durchgeführt, da ein Entscheidungsprozess für eine beobachtete Person in diesem Rahmen nicht simuliert werden kann.

Die Speicherung der Präferenzwerte der Gesichter untereinander wird durch die Wrapper-Klasse `FacePreferenceWrapper` verwirklicht, durch eine interne Liste von Gesichtern. Bei jedem Aufruf des `FacePreferenceWrapper`-Objekts wird die Liste aktualisiert und die Struktur vom jeweiligen Gesicht aus iteriert.

Durch die Implementierung wird ein stark vernetzter Graph erzeugt, der relativ komplexe Abhängigkeiten zwischen den beobachteten Beteiligten ermöglicht.

4.5 Eigenface-Implementation

Die Eigenface-Methode wurde ins System integriert, um mit beiden Augen jeweils das selbe Gesicht zu fixieren. Das soll eine natürliche Interaktion mit dem Roboter bieten. Die Schwierigkeit ist es hierbei ein möglichst stabiles Ergebnis für die Gesichtserkennung zu erhalten, dass zudem das restliche System nicht durch rechenintensive Prozesse beeinträchtigt.

Die Eigenface-Methode ist darüber hinaus gewählt worden, weil sie ein Ansatz bietet die Gesichter über das Kamerabild hinaus zu verfolgen und die Analyse fortsetzen zu können, wenn sie im Bild wieder detektiert werden.

Die Eigenface-Implementation wurde mit Hilfe der OpenCV-Library entworfen und lässt sich in drei Anwendungen unterteilen:

- Das Sammeln von Bildern des Gesichts,
- die Berechnung der Datenbank und
- die Gesichtserkennung.

In der Applikation werden diese drei Teile von `FaceSampler`, `DataBaseCreator` und `FaceRecognizer` übernommen (s. Abbildung 4.8).

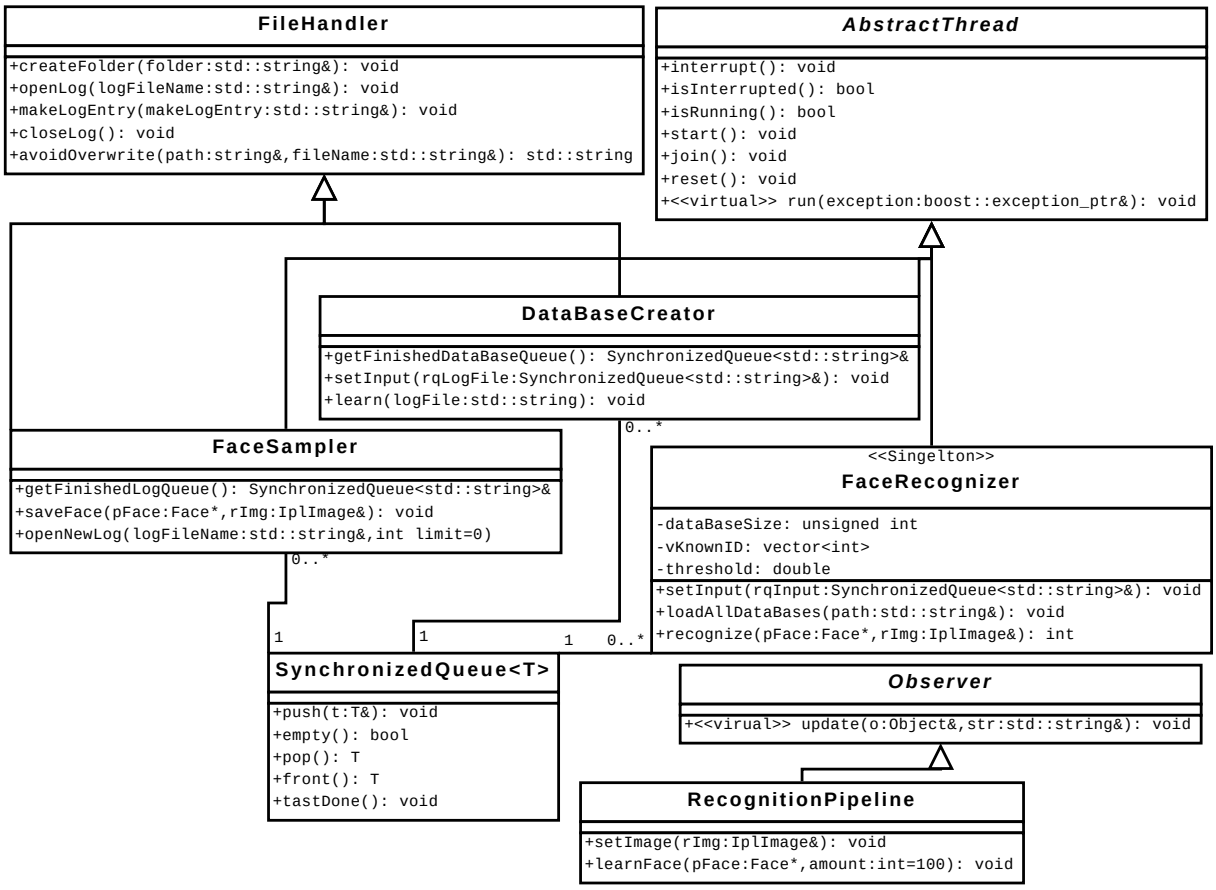


Abbildung 4.8: UML-Schema der `RecognizePipeline`.

`FaceSampler` sammelt die Bilder von den Gesichtern, die zur Datenbank hinzugefügt werden sollen. Dazu werden die Gesichter in einem Ordner, unter der zugehörigen ID-Nummer, gespeichert. Die Bilder werden von `FaceSampler` bearbeitet, so dass die Fläche des Gesichts ausgeschnitten, skaliert und die Beleuchtung normiert wird. Dadurch sollen die Abhängigkeiten der Skalierung, Lokalität und Beleuchtung, der Eigenface-Methode abgeschwächt werden, wie es in Kapitel 2.3 bereits diskutiert wurde.

Die Gesichter und ein Logfile, in dem die Ordnerpfade der Bilder mit der passenden ID-Nummer aufgelistet sind, werden von `FaceSampler` verwaltet und gesichert.

Die Erstellung der Datenbank wird von `DataBaseCreator` übernommen und bildet mit `FaceSampler` den Kernbereich des Lernens im System. `DataBaseCreator` lädt die Gesichter, die im Logfile angegeben sind, und berechnet aus ihnen den Unterraum.

Die von `DataBaseCreator` erstellten Datenbanken werden in XML-Dateien, die Informationen enthalten wie die ID, die Eigenvektoren, die Eigenwerte, das Durchschnittsgesicht etc., gespeichert.

Der dritte Teil der Implementation der Eigenfaces wird von der Klasse `FaceRecognizer`, die die Erkennung der Gesichter durchführt, übernommen. Hierfür wird ebenfalls das Kamerabild im Bereich des zu analysierenden Gesichts ausgeschnitten, skaliert und normalisiert, so wie es im Lernverfahren anwendet wird.

Wie in Kapitel 2.3 beschrieben wird, erfolgt eine Iteration über die Bilder der Datenbank. Wenn mehr als eine Datenbank vorliegt, so wird in jedem Unterraum der jeweiligen Datenbank das zu untersuchende Gesicht projiziert und der Abstand zu den übrigen Bildern untersucht. Der kürzeste Abstand bestimmt die ID des Gesichts.

Da die Eigenface-Methode nicht immer ein stabiles Ergebnis liefert, werden über mehrere Frames die Bilder vom Gesicht untersucht. Wenn in der Hälfte der untersuchten Bilder dieselbe ID gefunden wurde, so wird sie als die Identität des Gesichts zugeordnet. Wenn keine ID über 50% in den untersuchten Frames vorliegt, dann wird das Gesicht als unbekannt erachtet. Durch diese Regel soll verhindert werden, dass einzelne fehlerhafte Ergebnisse die Stabilität des Systems gefährden.

Das mehrfache Projizieren der Gesichter in die Unterräume kann vermieden werden, in dem die einzelnen Datenbanken zusammengefasst werden, jedoch kann das nicht im laufenden Betrieb geschehen, da die erforderliche Berechnung sehr rechenintensiv ist. Die Aufteilung der Datenbanken ist eine Strategie, die bei der Entwicklung der *Face-Recognition-Pipeline* entstanden ist, einem online Lernverfahren, das auf den bisher vorgestellten Systemteilen aufbaut.

4.6 Face-Recognition-Pipeline

Die Face-Recognition-Pipeline wurde entwickelt, um eine Gesichts-Datenbank online, d.h. im laufenden Betrieb des Systems, zu erstellen, die der Roboter für die Gesichtserkennung verwendet. Das soll ein Ansatz bieten, auch fremde Gesichter mit beiden Augen fixieren zu können, aber auch ein Gesicht wieder zu finden, wenn der Tracker ein Verlust gemeldet hat.

Das Ziel der Implementierung der Face-Recognition-Pipeline ist es, der Datenbank der Eigenface-Methode weitere Gesichter hinzu zu fügen, ohne dafür den Betrieb der restlichen Systemteile zu beeinträchtigen.

Dafür verwendet die Pipeline die drei Teile der Eigenface-Implementierung, das Face-Sampling, die Datenbankerstellung und die Face-Recognition, die in Kapitel 4.5 vorgestellt wurden, und lässt sie jeweils in einen eigenständigen Thread laufen. Dabei ist das *Worker-Producer-Schema* als Design-Vorlage gewählt worden, das sehr effizient die einzelnen Pipline-Abschnitte miteinander verbindet.

Das Worker-Producer-Schema kennzeichnet, dass im System jeweils ein oder mehrere Produzenten und Arbeiter existieren. Die Arbeiter haben dabei die Aufgabe, die Daten zu verarbeiten, die von den Produzenten geliefert werden. Ein Beispiel für ein Worker-Producer-Schema wäre eine Server-Client-Verbindung. Der Server verarbeitet dabei die Anfragen vom Client.

Der Vorteil vom Worker-Producer-Schema ist dabei, dass es im Multi-Thread-Betrieb sehr effizient ist, da der Thread, der momentan keine Aufgabe zu erfüllen hat, schläft und somit keine Ressourcen verbraucht, bis die Arbeit wieder benötigt wird. Dafür wird eine Buffer-Struktur benötigt.

Für diesen Zweck wurde in der Arbeit die `SynchronizedQueue` entwickelt, die ähnlich wie in anderen Programmiersprachen, ein Thread-sicheres Buffering ermöglicht. Dabei sind die Schnittstellen die Logfiles von `FaceSampler` und die XML-Dateien von `DataBaseCreator`, die zwischen den einzelnen Threads ausgetauscht werden.

Für die Verwaltung und Verknüpfung der drei Bestandteile der Pipeline wurde ein Fassade-Pattern, mit Hilfe der Klasse `RecognizePipeline`, entworfen. Die Klasse ermöglicht eine einfache Handhabung des Systems.

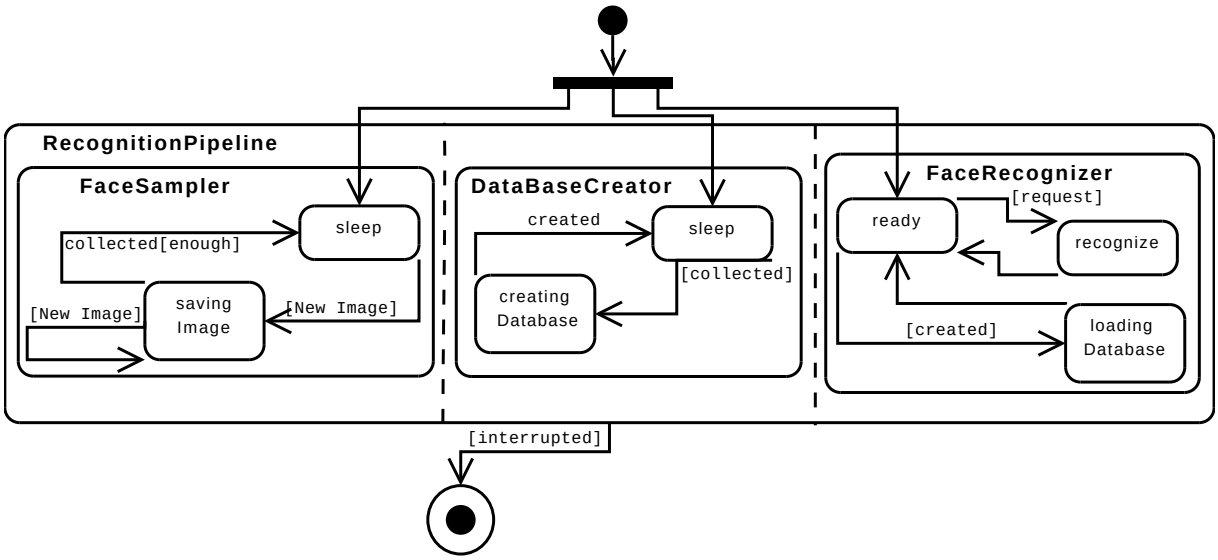


Abbildung 4.9: Statemachine der `RecognizePipeline`. Die von außen eingeführten Triggers sind: **enough**, wenn genügend Bilder gesammelt wurden, **request**, wenn eine Gesichtserkennung eines Bildes erfolgen soll, **interrupted**, wenn die einzelnen Threads ihre Arbeit beenden sollen und **newImage**, wenn ein neues Bild vorliegt.

In Abbildung 4.9 ist eine Statemachine abgebildet, die die Funktionsweise der Recognize-Pipeline beschreibt. Das Schema zeigt die verschiedenen Teile der `RecognizePipeline`, die jeweils zwischen den Arbeits- und Ruhephasen wechseln und auf die Signale warten.

`RecognizePipeline` wird ein `Face`-Objekt übergeben mit einer gesetzten ID als Attribut. Das `Face`-Objekt wird verfolgt und die Bilder werden durch `FaceSampler` gesammelt und anschließend von `DataBaseCreator` verarbeitet. Ist die Datenbank fertiggestellt, so wird sie von `FaceRecognizer` geladen und zur weiteren Gesichtserkennung eingesetzt.

Die `RecognizePipeline` erlaubt es, im laufenden Betrieb die rechenintensive Berechnung der Hauptkomponentenanalyse im Hintergrund durchzuführen, ohne den Betrieb der anderen Threads maßgeblich zu beeinflussen. Dadurch, dass alle Bestandteile flexibel miteinander verbunden sind, und bei einem Bottleneck-Problem auch mehrfach simultan angewendet werden können, ist die Pipeline eine sehr robuste und stark modifizierbare Struktur, die das übrige System nicht beeinträchtigt.

4.7 Implementation mit CB²

Die Implementation mit CB² erfolgt durch die Schnittstellen, die vom ERATO-Labor der Osaka University bereitgestellt wurden.

Die Schnittstellen erlauben es, die Motoren in den Augen des Roboters zu steuern und die Kamerabilder aus den Augen zu erhalten. Für Vorführungen wird der Rest des Roboters durch eine unabhängige Bewegungsroutine gelenkt.

Dabei ist es Ziel, dass die Kontrolle vom Roboter modular zum System funktioniert. So kann die Steuerung die verschiedenen Programmteile nutzen, aber die Applikation ist weitestgehend von der Hardware unabhängig.

Die Position der Augen werden jeweils durch eine Koordinate im Wertebereich $\{0, \ldots, 255\}$ in x- und y-Richtung angegeben. Dabei ist das Steuern der Augen entlang der x-Achse unabhängig voneinander, während die Bewegung entlang der y-Achse für beide Augen gilt.

Die Steuerung des Roboters ist im Rahmen der Arbeit in eine einfach bedienbare, objektorientierte Schnittstelle eingebettet worden (s. Abbildung 4.10).

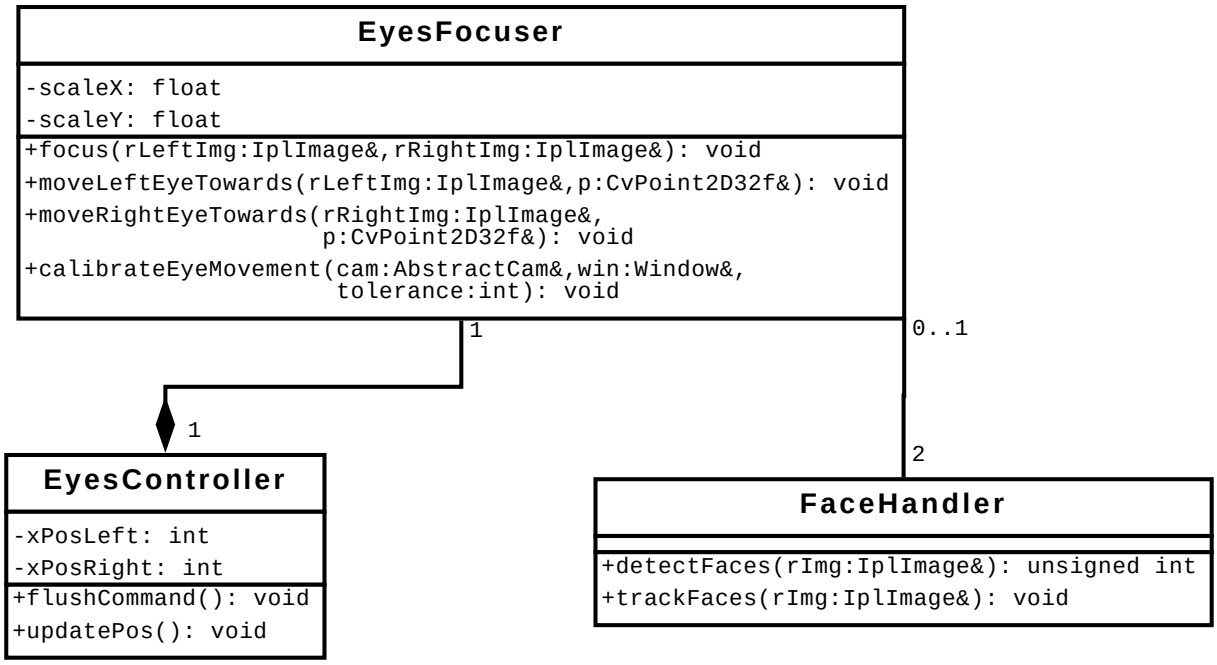


Abbildung 4.10: UML-Darstellung der Steuerungseinheiten von CB[2].

Die Abbildung 4.10 zeigt das UML-Diagramm zur Steuerung des Roboters. Dabei übernimmt die Klasse `EyesController` die Ansteuerung des Servers, der das Kommando zu den Augen des Roboters leitet. Die Koordination beider Augen zusammen übernimmt die Klasse `EyesFocuser`, die verschiedene Methoden dazu bietet. Unter anderem eine Methode zur Kalibrierung, die eine Skalierung des Motors und der Bewegung im Bild erlaubt. Die Skalierung dient dazu, dass die Augenbewegung des Roboters einer menschlichen Sakkade (Augenbewegung) entspricht, um mehrfaches Überprüfen zwischen der Augenbewegung des Roboters und des Gesichts zu vermeiden, da sie sonst eine stockende Bewegung hervorrufen würden.

Dazu wurde eine Bewegungsroutine einprogrammiert, die die Augen auf verschiedene Positionen setzt, um von diesen aus die Position des Gesichts zu bestimmen. Die Kalibrierung hat nicht die absolute Messung zur Aufgabe, sondern soll eine relativ genaue Aufwandsabschätzung der Augenbewegung ermöglichen, da ein festgelegter Skalierungsfaktor im Labor nicht vorgesehen ist.

Das Kamerabild aus den Augen des Roboters wird von der Klasse `ClientCam`, die in Kapitel 4.1 vorgestellt wurde, geliefert.

Die `focus()`-Methode von `EyesFocuser` steuert mit beiden Augen das selbe Gesicht an, mit Hilfe des Trackers und den übergebenen Bildern der Kameras. Dazu wird die Information aus der Gesichtserkennung verwendet.

Kapitel 5

Evaluation

Bei der Auswertung der Implementation wird zunächst die elementare Funktion des Trackings untersucht. Die Gesichtserkennung bezieht sich hierbei auf die Erkenntnisse aus der Literatur (s. Kapitel 2.3). Anschließend wird über die Qualität dieser Elemente und die Auswirkung für die weiteren Funktionalitäten des Systems diskutiert.

5.1 Gesichtsdetektion und Tracking

Die Gesichtsdetektion ist im Kontext dieser Arbeit eine Zusammenführung zweier Techniken. Zum einen die Detektion des eigentlichen Gesichtes, und zum anderen das Farbtracking, welches die Verfolgung des Gesichtes übernimmt, wenn der Detektor das Gesicht verloren hat.

Das Tracking unterscheidet sich in der Detektion in der Hinsicht, dass die Detektion eines Gesichtes, die Anwesenheit oder Abwesenheit des Gesichts feststellt. Hingegen das Tracking nach Möglichkeit das Gesicht verfolgen soll, auch wenn es zeitweise nicht zu sehen ist, allerdings soll es dabei robust gegen Störungen sein.

Für das Tracking ist daher ein Qualitätsmaß, wenn das Gesicht weiterhin verfolgt wird, auch wenn der Detektor zwischenzeitlich keins feststellt. Allerdings muss das Tracking in einem Toleranzbereich liegen, in dem das Ergebnis als richtig anerkannt wird.

Bei der Auswertung wird die Kombination, der Detektion und des Farbtrackings, mit einem festen Regelsatz untersucht. Wie bereist in Kapitel 4.2 vorgestellt wurde, wird eine Reihe von Bedingungen geprüft, bevor das Gesicht als verloren erklärt wird.

Der Regelsatz, mit dem das Tracking getestet wird, lautet im Einzelnen:

- Wenn das Gesicht innerhalb von 20 Frames nicht wieder im Trackingbereich detektiert wird,
- wenn die Bounding-Box des Gesichtsfeldes in der Höhe und Breite über ein Verhältnis von 1:3 liegt oder
- wenn der Gesichtsbereich kleiner ist als insgesamt vier Pixel

wird das Gesicht als verloren erklärt und das Tracking wird abgebrochen.

Für die Evaluation werden 6 Datensätze betrachtet, mit insgesamt 2720 aufgenommen Bildern aus verschiedenen Videostreams.

Die Datensätze sind mit zwei räumlichen Orientierungen aufgenommen worden. Zum einen in einer Position, in der sich CB² üblicherweise befindet, nämlich einer tieferliegenden, unterhalb der zu beobachteten Personen und zum anderen einer höherliegenden Position in Brusthöhe.

Auf 88,5% der aufgenommenen Bilder befinden sich, deutlich erkennbar, zwei Personen. Auf den restlichen 11,5% befindet sich lediglich eine Person oder eine weitere Person, die sich nicht vollständig im Bild befindet. Alle Bilder wurden unter den gleichen Bedingungen im Labor aufgenommen.

Bei den Datensätzen zeigt sich vor allem, dass das Tracking stabiler auf einer höheren Position verläuft als auf einer niedrigeren, wie es auf Abbildung 5.1 zu sehen ist.

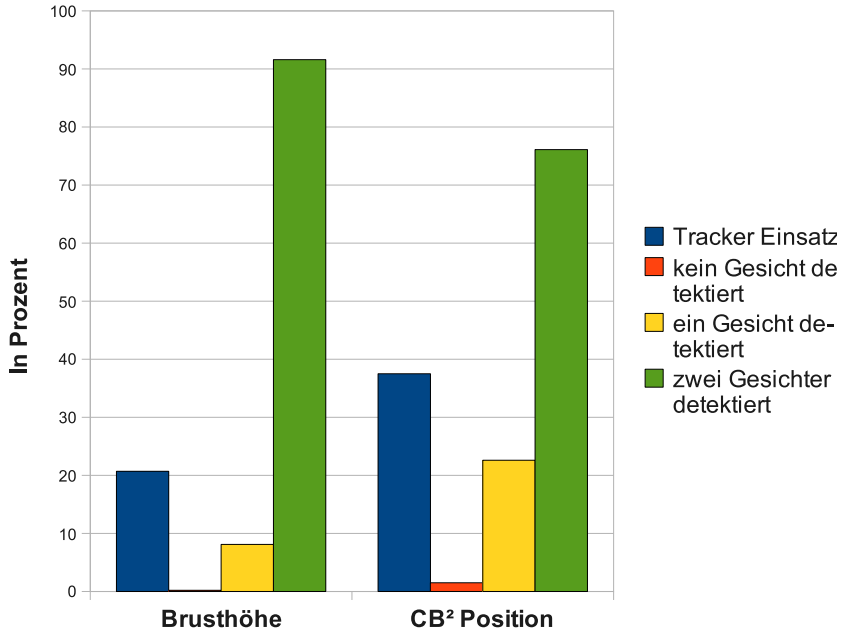


Abbildung 5.1: Statistik der insgesamt 2720 aufgenommen Bilder.

In der Statistik in Abbildung 5.1 ist zu ersehen, dass die Face-Detektion in der Position des Roboters deutlich öfters auf den Tracker zurückgreifen muss, um den Verlust des Gesichtes zu verhindern. Das ist zu erklären, da der Detektor der Firma Omron für eine Detektion in der Höhe des Kopfes ausgelegt ist. Jedoch zeigt sich im Vergleich, dass die Kombination beider Techniken ein deutlich besseres Ergebnis liefert als andere Methoden, wie zum Beispiel der Gesichtsdetektor von OpenCV.

In der Auswertung detektiert diese Kombination des Omron Face-Detektors und des Farbtrackers die Gesichter im Datensatz, die auf Brusthöhe aufgenommen sind, auf rund 99,5% der Bilder. Das bedeutet, dass der Detektor in dieser Position nur auf 0,5% der Bilder das Gesicht nicht detektiert. Auf der niedrigeren Position liegt die Verlustrate mit 5,8% höher als beim dem anderen Datensatz, das mit den deutlich häufigeren Gebrauch des Trackers zusammenhängt.

Als nächstes wird der Vergleich, der Kombination mit anderen etablierten Methoden zur Gesichtsdetektion, untersucht. Hierbei wird der Omron Face-Detektor, ohne die Kombination des Farbtrackers, und der Gesichtsdetektor von OpenCV betrachtet.

Der OpenCV-Detektor basiert auf Haar-like Features, die für die Gesichtserkennung trainiert sind. Dabei stellt OpenCV verschiedene vorab trainierte Haarcascade-Dateien zur Verfügung. Beim Test wurden auf die jeweiligen Bilder die „`haarcascade_frontalface_alt.xml`“ und „`haarcascade_profileface.xml`“ angewendet, die eine Detektion frontal und im Profil ermöglichen.

Sowohl der Omron-Detektor, als auch der Detektor von OpenCV sind auf die Datensätze angewendet worden. Die Ergebnisse zeigen, dass der Omron-Detektor deutlich besser die Gesichter im Bild detektiert als der Detektor von OpenCV. Jedoch schafft die Kombination mit dem Farbtracker es, dieses Ergebnis zu überbieten, wie man in Abbildung 5.2 erkennen kann.

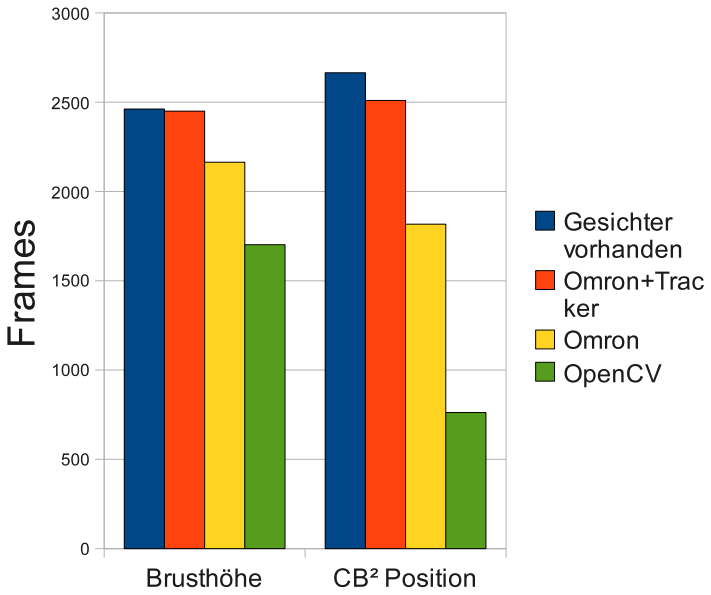


Abbildung 5.2: Vergleich der verschiedenen Detektionsmethoden.

Ohne den Farbtracker liegt die Verlustrate bei 12% auf Brusthöhe und bei 31,8% auf der Höhe von CB^2.

Der Gesichtsdetektor von OpenCV hingegen detektiert auf dem ersten Datensatz 69,1% und bei dem zweiten 28,6% der vorhandenen Gesichter.

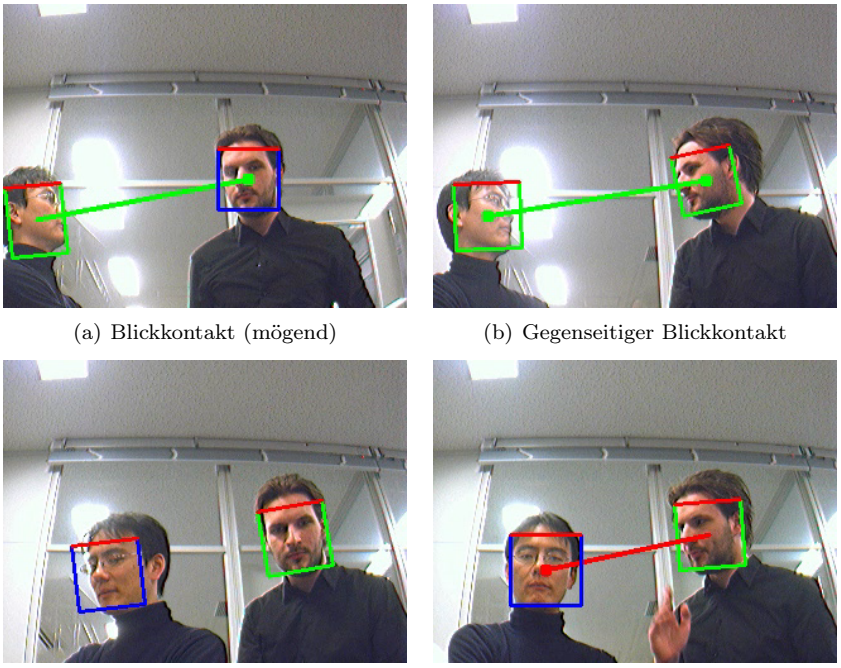
Die Gesichtserkennung der Firma Omron ist eine sehr mächtige Detektion und erlaubt es, sehr schnell und in vielen Freiheitsgraden ein Gesicht zu erkennen. Da die Haar-like Features, auf denen der OpenCV-Gesichtsdetektor aufbaut, sehr rechenintensiv sind, ist dem Omron-Gesichtsdetektor den Vorzug gegeben worden, um ein echtzeitfähiges und stabiles Tracking zu ermöglichen, in Kombination mit dem Farbtracker.

5.2 Weitere Funktionalitäten

Das Gesichtstracking und die Face-Recognition sind die elementaren Methoden, auf denen das weitere System aufbaut, das aber diese Methoden nur marginal erweitert.

So ist die Recognition-Pipeline eine Erweiterung der Eigenface-Methode, allerdings in starker Abhängigkeit zu ihr. Die Pipeline fügt Gesichter der Datenbank hinzu, falls diese von der Eigenface-Methode als unbekannt eingestuft worden sind. Jedoch lässt sich feststellen, dass somit die Pipeline im Rahmen der Arbeit verlässlich ist, da die Bilder der Datenbank mit der selben Kamera aufgenommen sind, wie die Bilder des zu analysierenden Gesichts. Auch die statischen Laborbedingungen bieten günstige Bedingungen für die Eigenface-Methode.

Zur Bestimmung des Blickkontakts, ist ebenfalls der Unterschied zum einfachen Tracking gering. Der Blickkontakt wird indirekt durch die Kopfausrichtung bestimmt. Aber hier lässt sich feststellen, dass bei einem stabilen Tracking, auch eine stabile Blickkontakt-Detektion vorliegt.



(a) Blickkontakt (mögend)

(b) Gegenseitiger Blickkontakt

(c) Kein Blickkontakt

(d) Blickkontakt (nicht mögend)

Abbildung 5.3: Screenshots vom laufenden System. Die grüne Bounding-Box zeigt an, dass der Roboter die Person mag und die blaue, dass er sie nicht mag. Ein grüner Sichtstrahl, gibt an, ob der Roboter denkt, dass die Person die andere Person mag oder, bei einem roten Strahl, sie nicht mag.

In Abbildung 5.3 ist das laufende System in vier Abbildungen zu sehen. In den Bildern ist zu erkennen, wie die Gesichter und die Blickkontakte detektiert werden. Zudem werden die Präferenzwerte in den Bildern ausgewertet. Das Feld der Detektion wird mit einer Bounding-Box dargestellt, dessen Färbung angibt, ob die Person vom Roboter gemocht wird oder nicht. Der Blickkontakt wird angegeben durch eine Linie von der Person aus, von der der Blickkontakt erfolgt. Die Färbung der Sichtlinie gibt an, ob der Roboter meint, dass die Person die andere mag oder nicht mag.

Die Frage, ob diese Einschätzungen des Roboters der Realität entsprechen, oder dies ebenso bei einem Menschen erfolgt, der ähnliche Voraussetzungen hat, wie der Roboter selbst, würde den Rahmen dieser Arbeit überschreiten.

Kapitel 6

Zusammenfassung und Ausblick

Die Studienarbeit analysiert mit Hilfe einer erweiterten Balancetheorie die Relationen des Roboters zu den Personen im Kamerabild und den Personen untereinander. Es wurde gezeigt, dass die Abstraktion der Balancetheorie auf eine konkrete Anwendung übertragen werden kann. Allerdings muss die Theorie erweitert und teilweise eingeschränkt werden, um sie flexibler, aber gleichzeitig passend zur Anwendung zu gestalten.

Dadurch wird die theoretische Grundlage, auf der die Arbeit beruht, ebenfalls verändert. Damit diese Modifikationen nicht den Rahmen der psychologischen Grundlage verlassen, müssen sie wieder herum mit psychologischen Mitteln untersucht werden. Das würde allerdings den Umfang dieser Arbeit überschreiten, aber bietet ein Ansatz für eine interdisziplinäre Zusammenarbeit der Psychologie und Robotik. Die Interaktion und das Verhalten der Maschinen nach menschlicher Vorlage zu gestalten, ist für beide Disziplinen von Interesse.

Im Kontext der Entwicklung einer ausreichenden Interaktion zwischen der Maschine und dem Menschen, wäre es interessant zu erforschen, welche sozialen Merkmale detektiert werden müssten, zum Beispiel im Spektrum der Mimik.

Darüberhinaus ermöglicht die Recognize-Pipeline ein Ansatz, die Interaktion über das Kamerabild hinaus zu verfolgen. Dadurch kann der Roboter das Gesicht mit vergangenen Interaktionen assoziieren und dem entsprechend agieren. Allerdings bedarf die Pipeline-Struktur weiterer Arbeit. So werden bestehende Datenbanken über ein Gesicht nicht mit neuen Bildern erweitert, so fern sie notwendig sind. Auch kann keine automatische Korrektur erfolgen, falls fehlerhafte Informationen in die Datenbank gelangen. So kann es vorkommen, dass das selbe Gesicht zwei unterschiedliche IDs erhält, wenn das Gesicht nicht wiedererkannt wird. Auch können sehr ähnliche Gesichter zusammenfallen zu einer ID. Solche Fehler müssten für eine stabile Anwendung selbständig korrigierbar sein.

Abbildungsverzeichnis

Literaturverzeichnis

[BG04] Brooks, Alan ; Gao, Li: *Face Recognition: Eigenface and Fisherface Performance Across Pose.* 2004

[Jor08] Jorquera, Diego: *Face Recognition Using Eigenfaces.* `http://dcc.puc.cl`. Version: 2008

[KH04] Kaplan, Frederic ; Hafner, Verena ; Berthouze, Luc (Hrsg.) ; Kozima, Hideki (Hrsg.) ; Prince, Christopher G. (Hrsg.) ; Sandini, Giulio (Hrsg.) ; Stojanov, Georgi (Hrsg.) ; Metta, Giorgio (Hrsg.) ; Balkenius, Christian (Hrsg.): *The Challenges of Joint Attention.* `http://cogprints.org/4067/`. Version: 2004

[Loh] Lohninger, H.: *Grundlangen der Statistik.* `http://www.statistics4u.info/fundstat_germ/cc_distance_meas.html`. – [Online; Stand 8. August 2009]

[MH] Marina Hennig, Dr. rer. s.: *Balancetheorie von Fitz Heider.* `http://www.marinahennig.de/PDF-Dateien/Balancetheorie1.pdf`. – [Online; Stand 3. August 2009]

[MP01] Moon, H.J. ; Phillips, P.J.: Computational and performance aspects of PCA-based face recognition algorithms. In: *Perception* 30 (2001), Nr. 3, S. 303–321

[Pin09] Pink Tentacle: *CB2 baby robot developing social skills.* 2009

[TI04] Tani, Jun ; Ito, Masato: *Joint attention between a humanoid robot and users in imitation game.* `http://www.bdc.brain.riken.go.jp`. Version: 2004

[TP91] Turk, Matthew ; Pentland, Alex: Eigenfaces for recognition. In: *J. Cognitive Neuroscience* 3 (1991), Nr. 1, S. 71–86. `http://dx.doi.org/http://dx.doi.org/10.1162/jocn.1991.3.1.71`. – DOI http://dx.doi.org/10.1162/jocn.1991.3.1.71. – ISSN 0898–929X