Gesichtsverfolgung mit dem Raspberry Pi

Seminararbeit

im

Seminarkurs Informatik/ Mathematik

Max – Steenbeck – Gymnasium

D–03046 Cottbus

Universitätsstraße 18

Verfasser:

Hübscher, Leonardo
Abiturjahrgang 2015
Sielower Straße 50
D-03044 Cottbus
leonardo.huebscher.business@gmail.com

Betreuer:

Hr. M. König

COTTBUS, DEN 31.10.2014

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

Abkürzung	Bedeutung
EEG	Elektroenzephalografie
fMRT	Funktionelle Magnetresonanztomographie
GHz	Gigahertz
SSH	Secure Shell – Netzwerkprotokoll (verschlüsselt)

1 EINLEITUNG

Die Entwicklung der Informatik verlässt zunehmend die Ebene der wissenschaftlichen Grundlagenforschung und bewegt sich hin zu den verschiedenen Anwendungsspezialisierungen. Eine davon ist die Technik der Gesichtserkennung, welche sich in den letzten Jahren rasant entwickelt hat und heute bereits breite Anwendung, zum Beispiel als Verschlüsselungstechnologie oder Fokussierungshilfe bei Kameraaufnahmen findet.

Doch lassen sich auch eigene kleine Projekte, welche die Gesichtserkennung verwenden, umsetzten? Wie schwierig ist eine solche Umsetzung? Auf welche Komplikationen könnte man treffen? Welche theoretischen Grundlagen sind von Nöten?

Mit genau diesen Fragen werde ich mich in meiner Seminararbeit beschäftigen. Abschließend werde ich eine spezielle Anwendungsproblematik mit einer auf dem Raspberry Pi basierenden Versuchsanordnung demonstrieren.
Damit möchte ich ein Anschauungsmittel zur Erleichterung des grundlegenden Verständnisses der Funktionsweise der Gesichtserkennung schaffen. Für die Einarbeitung in die Gesichtserkennung werde ich ein Demoprogramm für den PC schreiben. Des Weiteren ist es mein Ziel ein kleines Programm für die Motorsteuerung des Pis zu entwickeln, um mich mit der Steuerung von Servos vertraut zu machen. Ist der Einstieg geschafft, werden beide Teilfunktionen zu einer Versuchsanordnung zusammengeführt. Ziel dieser Anordnung ist es, einem Gesicht durch eine vom Raspberry Pi gesteuerte Webcam zu folgen.

Die zentrale Frage meiner Seminararbeit ist also die Frage nach der Funktionsweise der Gesichtserkennung und besonders der Versuch der Anwendung mit der dazugehörigen Bestimmung des Schwierigkeitsgrades bezüglich der praktische Umsetzbarkeit für den Raspberry Pi - Unkundigen.

Der praktische Teil der Arbeit wird in Form einer Anleitung verfasst werden, die von einem erfahrenen Java – Programmierer auch ohne Raspberry-Vorkenntnisse genutzt werden kann. Die Arbeit wird sich ausgehend von den theoretischen Grundlagen bis hin zu der praktischen Umsetzung aufbauen. Das entspricht einer Entwicklung vom Allgemeinen zum Speziellen, was eine Analogie zur Technikentwicklung darstellt.

Ich beschäftige mich mit dieser Thematik, da mich schon seit längerer Zeit die vielseitigen Einsatzmöglichkeiten des Raspberry interessieren und faszinieren. Besonders bemerkenswert finde ich dabei, dass dieser relativ simple Ein-Platinen-Computer den „Erfindergeist“ herausfordert und ihn kaum einschränkt.

2 THEORETISCHE GRUNDLAGEN

2.1 Die biologische Gesichtserkennung des Menschen

2.1.1 Die Funktionsweise

Die Gesichtserkennung ist eine angeborene Fähigkeit, die eine Hauptrolle in der menschlichen Kommunikation und sozialen Interaktion spielt. Diese faszinierende Begabung ermöglicht es einer Person ihre Mitmenschen nach mehreren Jahrzehnten wieder zu erkennen, da unser Gehirn in der Lage ist, von anderen Gesichtern altersunabhängige Merkmale zu abstrahieren. Aus dem Gesicht lässt sich jedoch nicht nur die Identität einer Person ablesen, sondern auch deren Alter, Geschlecht, Gefühlslage und Aufmerksamkeit. (Grüter, 2014) Die nonverbale Kommunikation ist die älteste Kommunikationsform und hat auch heute noch eine sehr große Bedeutung. Die Wirkung einer Botschaft wird zu 38%[1] durch Mimik und Gestik bestimmt. (Adamski, 2014)

Um herauszufinden welche Regionen des Gehirns für die menschliche Gesichtserkennung zuständig sind, hat man das Gehirn mittels EEG und fMRT untersucht. Dabei entdeckte man, dass sobald man ein Gesicht erkennt, im unteren Teil des Schläfenlappens eine N170 – Welle ausgelöst wird. Diese Welle schlägt nur aus, wenn ein Gesicht erkannt wird. Um das Gesicht einer Person zuordnen zu können, betrachtet der Mensch vermutlich besondere Merkmale, wie zum Beispiel Leberflecken oder Muttermale, sowie die unterschiedlich ausgeprägten Gesichtsmerkmale und Abstände zwischen diesen. Die Gesichtserkennung basiert auf Erfahrungswerten und Fantasien, welche die Entscheidungen beeinflussen, indem die visuellen Informationen unterschiedlich interpretiert werden. (Grüter, 2014) Der Ablauf der Gesichtserkennung beim Menschen wird momentan noch erforscht, sodass zu diesem Zeitpunkt noch keine genaueren Angaben zur Funktionsweise getroffen werden konnten. (Leyh, 2014) Es wird untersucht, wie der Vergleich zwischen den gespeicherten Merkmalen im Gehirn und den visuellen Informationen stattfindet und wie die Merkmale gespeichert werden.

Allerdings existiert eine Theorie, welche versucht den Ablauf der Gesichtserkennung zu erläutern. Diese bekannte Theorie von Ellis und Lewis besagt, dass das Gehirn das Gesehene in *Gesicht* und *kein-Gesicht* aufteilt, bevor es mit der Weiterverarbeitung der visuellen Informationen fortfährt. Anschließend wird parallel versucht das erkannte Gesicht einer Person zuzuordnen und den Gesichtsausdruck zu deuten.

[1] Siehe 7-38-55 Regel der Kommunikationswissenschaft

Ebenfalls wird diskutiert, ob beim Menschen eine Unterscheidung zwischen *Hergestelltem* und *Naturobjekten* getroffen wird. (2012, Kogn. Neurowissenschaften)

2.1.2 KRANKHAFTE STÖRUNGEN DES GESICHTSERKENNUNGSVERMÖGENS

Die bekannteste Störung der Gesichtserkennung durch den Menschen ist die Prosopagnosie, oder auch „Gesichtsblindheit“ genannt. Dies ist eine Gesichtserkennungsschwäche, welche zum einen erblich und angeboren sein kann, oder möglicherweise durch eine Verletzung oder Schlaganfall erworben wurde. Leidet man an einer Prosopagnosie, so hat man Schwierigkeiten ein Gesicht zu einer Person zuordnen zu können. Dies ist besonders der Fall, wenn man die Personen in unerwarteten Situationen antrifft oder sich in einer großen Menschenmenge befindet. Die Menschen sind dennoch in der Lage andere Informationen aus dem Gesicht zu extrahieren. (Grüter, 2014)

Menschen, die seit der Geburt an Prosopagnosie leiden, entwickeln andere Wege und Strategien um ihre Mitmenschen zu erkennen. Sie erkennen sie an ihrer Gangart, Stimme, Frisur, Kleidung oder auch ihrem Verhalten. Dies führt jedoch auch dazu, dass die Krankheit in vielen Fällen unentdeckt bleibt, und die Personen eher sozial herabgestuft werden, da sie zum Beispiel ihre Mitmenschen nicht auf der Straße erkennen und folglich nicht grüßen. Die Krankheit wird in drei Arten unterteilt: die Apperzeptive, Assoziative und Kongenitale. Diese beschreiben den unterschiedlichen Grad an Einschränkungen. Es wird vermutet, dass 2% der Bevölkerung in Deutschland an dieser Krankheit leidet. Prosopagnosie wird ebenfalls oft mit Autismus in Verbindung gebracht, da sie häufig parallel auftritt. (Unbekannt, Prosopagnosie, 2014)

Ein weiteres Krankheitsbild ist das sehr seltene Capgras – Syndrom, welches aus einer schweren psychischen Krankheit folgt. Die Störung beeinflusst nicht die Zuordnung von Gesichtern zu Personen, allerdings empfinden die Personen keine Bindung zu dieser aufgrund eines Risses der Nervenbahn zwischen Gefühl und Erinnerung. (Grüter, 2014)

Die Gesichtserkennung des Menschen kann ebenfalls durch Sehstörungen und Aufmerksamkeitsstörungen beeinträchtigt werden.

2.2 Die Gesichtserkennung in der Computerwelt

2.2.1 Die Objekterkennung

Unter der Objekterkennung versteht man die Analyse, Systematisierung und Interpretation von Signalen mit Hilfe bereits vorhandener Informationen über eine Menge von Objekten.

Man unterscheidet bei der Objekterkennung zwischen Kategorisierung und Identifizierung.
Die Kategorisierung beschäftigt sich mit der Zuordnung eines Objektes zu einer Gruppe von Objekten, welche eine oder mehreren ähnliche Eigenschaften zu dem untersuchten Objekt aufweisen. Die Identifizierung hingegen ist spezieller und dient der eindeutigen Bestimmung eines Objektes aus einer Menge von Objekten. In Bezug auf die Gesichtserkennung wäre die Kategorisierung eines Objektes in *Gesicht* und *kein-Gesicht* und die Identifizierung, ob denn dieses Gesicht, das Gesicht einer bestimmten, dem Bezugssystem bekannten, Person ist. (Kognitive Neurowissenschaften, 2012)

2.2.2 Die Hierarchien der Objekterkennung

Die Zuordnung eines Objektes zu einer bestimmten Kategorie läuft auf verschiedenen Hierarchien ab. Es werden drei Hierarchien unterschieden:

Die übergeordnete Ebene *(„superordinate level“)* ist die höchste Ebene und umfasst alle Klassifizierungen einer Kategorie. In dieser Ebene spielt das Aussehen der Objekte eher eine geringere Rolle, da die Gruppe durch die gemeinsamen Eigenschaften und Funktionen definiert wird. Es kann jedoch natürlich sein, dass, wie in dem von mir gewählten Beispiel, außerdem eine Ähnlichkeit in der Form besteht.

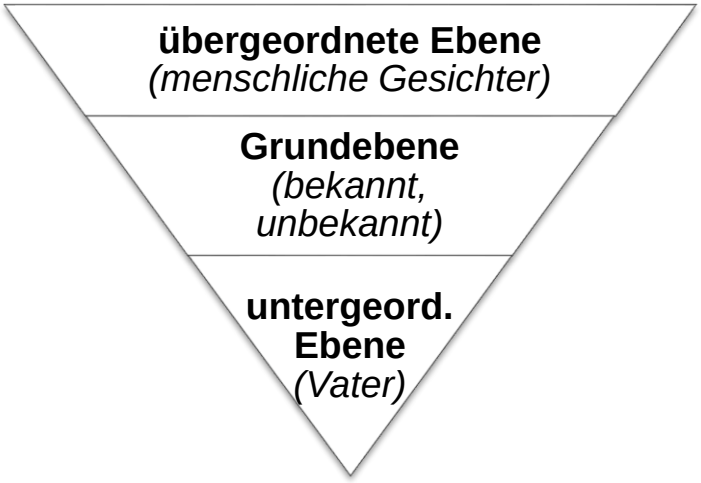

Abbildung 1 Hierarchien der Objekterkennung

Die Grundebene *(„basic level")* zeichnet sich hingegen durch eine hohe Übereinstimmung optischer Merkmale aus und enthält die untergeordnete Ebene („subordinate level"). In dieser findet die finale Spezifizierung statt. In der Abbildung 1 steht in den Klammern eine mögliche Einteilung der Hierarchie für die Gesichtserkennung. (Kognitive Neurowissenschaften, 2012)

2.2.3 MODELLE DER OBJEKTERKENNUNG

Es gibt viele verschiedene Theorien, die versuchen die Funktionsweise der Objekterkennung zu erläutern. Um diese dennoch klar strukturieren zu können, unterscheidet man zwischen der Dimension des Koordinatensystems und der Repräsentationsform. Es gibt zwei unterschiedliche Repräsentationsformen. Bei der beobachterzentrierten Repräsentation wird das Objekt aus einem bestimmten Blickwinkel und Abstand betrachtet und dessen resultierende Ansicht verwertet. Im Gegensatz dazu wird bei der objektzentrierten Darstellung das gesamte Objekt in einem eigenen Koordinatensystem beobachterunabhängig gespeichert. Die Einordnung der verschiedenen Theorien wird durch das abgebildete Schema (Abb. 2) verdeutlicht. (Kognitive Neurowissenschaften, 2012)

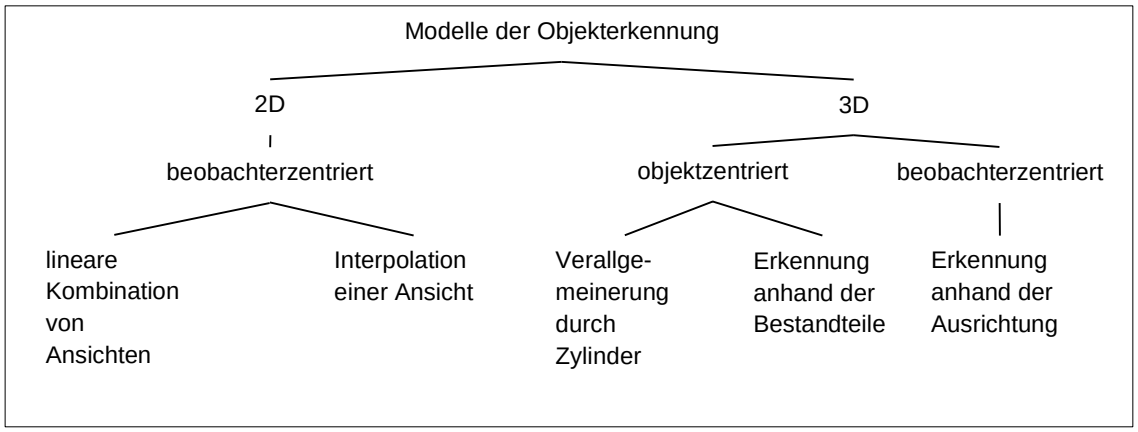

Abbildung 2 Schema zu den Modellen der Objekterkennung

2.2.3.1 Lineare Kombination von Ansichten

Hierbei reicht die Anordnung der markantesten Merkmale eines Objektes zur Identifikation. Es werden lediglich diese gespeichert. (Kognitive Neurowissenschaften, 2012)

2.2.3.2 Interpolation einer Ansicht

Bei der Interpolation einer Ansicht werden gespeicherte Ansichten eines Objektes generalisiert. Für die Wiedererkennung dieses Objektes in einer neuen Ansicht werden bekannte Teile aus bereits abgespeicherten Ansichten gesucht um auf das betrachtete Objekt schlussfolgern zu können. (Kognitive Neurowissenschaften, 2012)

2.2.3.3 Verallgemeinerung durch Zylinder

Diese Theorie beschäftigt sich mit der Repräsentation eines Objektes durch einen oder mehrere Zylinder. Diese können dabei in Anordnung und Größe variiert werden.

Um detailreichere Objekte darzustellen, kann man einen Zylinder in mehrere kleinere zerlegen, wodurch eine strukturierte Unterteilung entsteht.

Diese Theorie ist objektzentriert, da nur die Beziehung zwischen den Zylindern von Bedeutung ist. (Kognitive Neurowissenschaften, 2012)

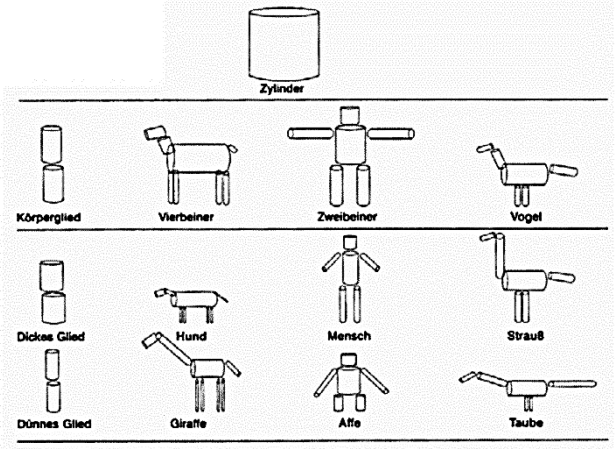

Abbildung 3 Verallgemeinerung durch Zylinder

Quelle: http://wwwisg.cs.uni-magdeburg.de/bv/files/LV/Grundlagen_der_Computer_Vision/VL/L10_Recognition%20by%20Components.pdf

2.2.3.4 Erkennung anhand der Bestandteile

Zuerst werden hierbei alle Kanten einer gegebenen 2D – Projektion erfasst. Anschließend kann eine Dekodierung der Geone erfolgen. Die Kombination der Geone ist ausschlaggebend für das Zuordnen der Projektion zu einem Objekt. Es ist also nur die Anordnung der Geone zueinander entscheidend, wodurch diese Theorie ebenfalls objektzentriert ist. (Unbekannt, Recognition by Components, 2014)

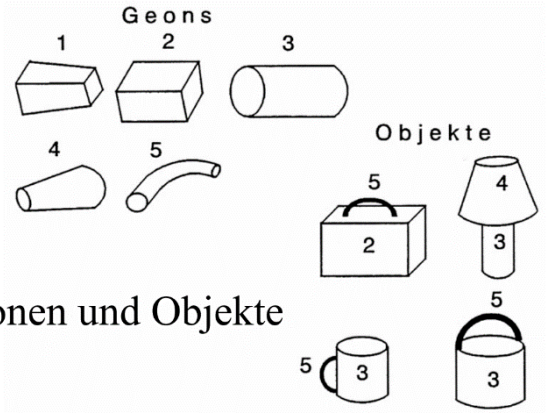

Abbildung 4 Geonen und deren Objekte

Quelle: http://wwwisg.cs.uni-magdeburg.de/bv/files/LV/Grundlagen_der_Computer_Vision/VL/L10_Recognition%20by%20Components.pdf

2.2.3.5 Erkennung anhand der Ausrichtung

Bei dieser Theorie wird im Laufe des Lernprozesses eine 3D – Projektion eines Objektes abgespeichert. Um zukünftig dieses Objekt wiederzuerkennen, wird ein beobachterzentriertes 2D – Bild mit der gespeicherten Projektion durch die Betrachtung der Merkmalsschwerpunkte des Objektes abgeglichen. Dies geschieht mit allen gespeicherten Projektionen, um anschließend die Projektion zu finden, welche die beste Übereinstimmung mit dem 2D – Bild besitzt. (Kognitive Neurowissenschaften, 2012)

2.2.4 VIOLA – JONES – ALGORITHMUS

Der 2001 veröffentlichte Algorithmus von Paul Viola und Michael Jones ist der erste Objekterkennungsalgorithmus welcher in Echtzeit Objekte erkennt. (Unbekannt, Viola–Jones object detection framework, 2014) Das Ziel war vor allem einen performanten Weg zu finden, Gesichter in einem Bild zu erkennen.
Im Folgenden werde ich kurz auf die Funktionsweise des Viola – Jones – Algorithmus eingehen, da die praktische Umsetzung der Anwendungsproblematik auf diesem Algorithmus basieren wird.

Zuerst wird das übergebene Bild für die nächsten Schritte in ein Graustufenbild konvertiert und dann in ein Integralbild umgewandelt. Dadurch kann die zur Klassifikation notwendige Pixelsumme in konstanter Zeit berechnet und die Berechnung signifikant beschleunigt werden. Es werden also Zugriffe und Rechenzeit eingespart. Nachfolgend wandert ein Suchfenster mit einer konstanten Blockdimension zeilenweise über das Bild. Jeder betrachtete Block wird an mehrere Klassifikatoren übergeben. Der Klassifikator betrachtet die Helligkeitsdifferenzen innerhalb eines Blockes und vergleicht sie mit bereits vorhandenen Merkmalen, welche unterschiedlich verzerrt werden können. Damit wird die Übereinstimmung durch einen bestimmten Merkmalswert nach folgender Formel berechnet. (Tobias Groß, 2014)

$$\sum_{n=1}^{k_1} \overrightarrow{0P_{1_{k_1}}} - \sum_{n=1}^{k_2} \overrightarrow{0P_{2_{k_2}}}$$

$$mit\ k_1, k_2 \in \mathbb{N}\ und\ P_1, P_2 \in E_{x,y}, wobei$$

$$P_1 \triangleq Pixel\ in\ heller\ Region$$

$$und\ P_2 \triangleq Pixel\ in\ dunkler\ Region$$

Der Merkmalswert wird berechnet, indem die Summe der Pixel in den dunklen Regionen von der Summe der hellen Regionen subtrahiert wird. Mit Hilfe dieses Wertes kann man ein Objekt zum Hintergrund oder dem zu erkennenden Objekt zuordnen.

Um diesen Schritt weiter zu beschleunigen verwendet man das ADA – Boost – Verfahren. Dieses beschreibt die Verwendung von Cascades. Dabei werden diese zunächst nach deren Komplexität geordnet. Die einfachen Cascades sind dabei am Anfang. Nun wird die gesamte Liste von Cascades durchgegangen, bis es zu einer negativen Rückgabe kommt oder alle Cascades auf das zu untersuchende Objekt angewandt wurden. Wie viele Cascades man verwendet und welche Komplexität diese aufweisen hängt dabei von den vorher bestimmten maximalen Fehlerraten ab. Abschließend wird aufgrund der gesamten Rückgaben jedes Blockes eine finale Antwort berechnet. (Walter & Kesser, 2014)

Dieses Verfahren ist somit in der Lage gewünschte Objekte in einem Bild zu detektieren. Der Nachteil dabei ist jedoch, dass die Berechnung der Detektoren und der Cascades einmalig einen hohen Aufwand benötigen. Allerdings ist dies immer noch besser, als keine Echtzeit – Erkennungssoftware zu besitzen, denn sonst könnte ich mein geplantes Projekt nicht in die Praxis umsetzten.

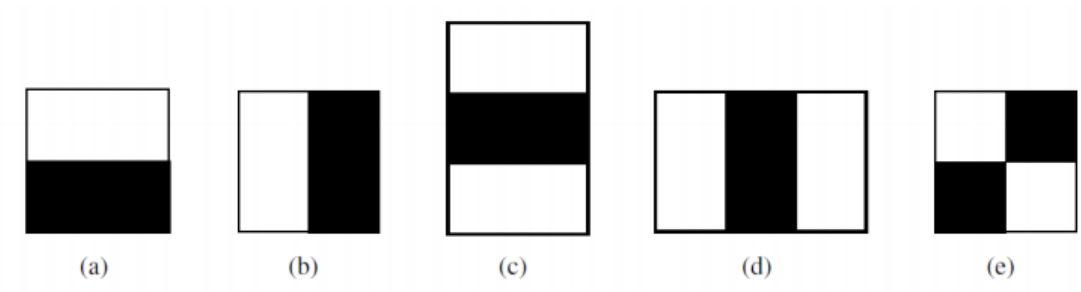

Abbildung 5 Basis-Blockmerkmale, Basis-Klassifikatoren

Quelle: https://www12.informatik.uni-erlangen.de/edu/mpa/ss11/talks/violajones.pdf
mputer_Vision/VL/L10_Recognition%20by%20Components.pdf

2.3 Die Bildverarbeitungsbibliothek OpenCV

2.3.1 Was ist OpenCV?

“OpenCV (Open Source Computer Vision Library) ist eine Softwarebibliothek für die Verarbeitung von Bildern, sowie für maschinelle Lernprozesse. Die Bibliothek wurde erstellt, um eine gemeinsame Infrastruktur für die Bildverarbeitung zu schaffen. Außerdem soll mit ihr der Einzug der Maschinen in die Handelsprodukte erleichtert werden, weswegen die Bibliothek als BSD lizensiert wurde.“

(*Übersetzt* von http://opencv.org/about.html 04.09.14)

Die Bibliothek wurde komplett in C++ geschrieben und besteht aus rund 2500 optimierten Algorithmen, welche von renommierten Unternehmen genutzt werden. Zu diesen gehören zum Beispiel Google, Yahoo, Microsoft, Sony, Toyota und Intel, welches bei der Mitgründung der Bibliothek beteiligt war. Google nutzt OpenCV zum Beispiel für die Google StreetView, China nutzt es um explosive Sprengstoffe zu identifizieren und Europa um frühzeitig ertrinkende Personen zu erkennen. (Unbekannt, About, 2014)

2.3.2 Test der Gesichtserkennungsbibliothek

Um zu testen, wie gut die kostenlose Bibliothek OpenCV wirklich ist, habe ich zunächst ein kleines Programm geschrieben, welches das Testverfahren erleichtern soll. Es besitzt die Fähigkeit mehrere Bilder nacheinander einzulesen, um anschließend die Erkennungssoftware von OpenCV anzuwenden. Danach werden die Funde mit Hilfe eines roten Rechteckes markiert. Der Anwender kann nun festlegen, ob alle angezeigten Funde zutreffen, oder ob es False-Positives oder fehlende Erkennungen gibt. Zusätzlich gibt es die Option, alle Bilder einer beliebigen Website herunterzuladen.

Ich habe dieses Programm nach der Fertigstellung dazu genutzt, insgesamt 1012 Bilder zu untersuchen. Dazu habe ich fünf Durchläufe gestartet, bei denen ich die von OpenCV mitgelieferte „haarcascade_frontalface_default.xml“ verwendet habe. Dafür habe ich die Bilder der ersten Seite der Google – Bildersuche für die Stichwörter „Gesicht“, „face“, „Auto“ und „Haus“ genutzt.

Für den letzten Durchlauf habe ich mir die Bilder von Gesichtern einer öffentlich zugänglichen Datenbank[2] heruntergeladen.

Tabelle 1 Testergebnisse des OpenCV - Cascades

Durchlauf / Anzahl der	1	2	3	4	5	Gesamt
Bilder	80	84	687	80	81	1012
Positives	70	71	348	0	0	489
gefundenen Positives	61	62	348	0	0	471
nicht gefundenen Positives	9	9	0	0	0	18
Negatives	10	13	339	80	81	523
gefundenen Negatives	8	9	312	76	75	480
false-positives	2	4	27	4	6	43

Aus dieser Messreihe lässt sich auf die zwei für diese Arbeit bedeutsamsten Faktoren schließen. Zum einen beträgt die Erkennungsrate von Bildern mit einem Gesicht in der Frontalperspektive rund 96%. Zum anderen lässt sich aber auch erkennen, dass es eine Fehlerquote von ca. 8% gibt, in der falsche Positive-Ergebnisse angezeigt gefunden werden. Diese Werte sind aber für diese Arbeit gut genug, da falsche Erkennungen idealerweise nicht ins Gewicht fallen werden und alle Erkennungsraten über 90% akzeptabel sind.

Besonders fehleranfällig war das System interessanterweise bei Schattierungen und Flecken am Hals oder im Gesicht. Nicht selten wurde in diese ein weiteres Gesicht hineininterpretiert. Dies lässt sich auf die Schwächen des Viola-Jones-Algorithmus zurückführen, da dieser lediglich die Helligkeitsdifferenzen eines Bildes betrachtet.

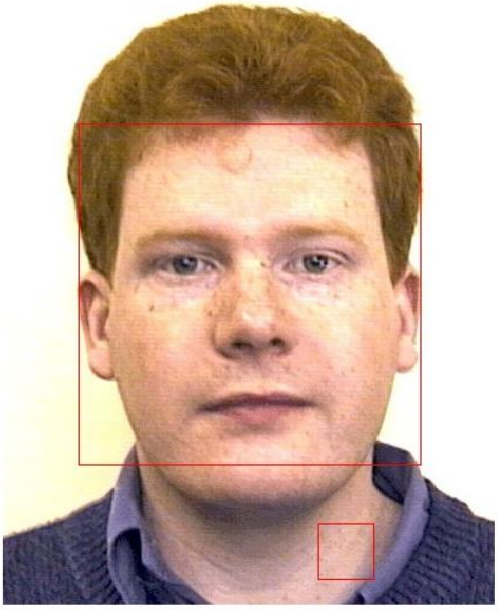

Abbildung 6 Fehlerhafte Interpretation einer Schattierung

Bild bearbeitet und entnommen aus: http://pics.psych.stir.ac.uk/2D_face_sets.htm

[2] Quelle der Datenbank: http://pics.psych.stir.ac.uk/2D_face_sets.htm

2.4 RASPBERRY PI

2.4.1 DER AUFBAU

Den Raspberry PI gibt es momentan in zwei verschiedenen Ausführungen. Es ist eine Revision A und eine Revision B erhältlich. In nachfolgender Tabelle sind die technischen Merkmale gegenübergestellt, um später die Wahl begründet treffen zu können.

Tabelle 2 Vergleich Raspberry Rev. A und Rev. B

	Revision A	**Revision B**
Prozessor	ARM 1176JZF-S (Single Core) 1 Ghz	
GPU	Broadcom VideoCore IV	
SD – Speicher	bis 128 GB	
Stromversorgung	Micro - USB oder Batterie (5V)	
Abmessungen	85,60 mm x 53,98 mm x 17 mm	
Energieverbrauch	2,5 Watt	3,5 Watt
Arbeitsspeicher (RAM)	256 MB	512 MB
Anzahl USB 2.0 - Anschlüsse	1	2
Netzwerkanschluss	nicht Vorhanden	vorhanden (100 Mbit)
Preis zzgl. Versand	20,90€	28,19€

Angelehnt an (Hammerschmidt, 2014) ergänzt um (Unbekannt, Raspberry Pi Modell A, 2014) und (Unbekannt, Raspberry Pi Modell B, 2014)

Wie man der Tabelle entnehmen kann, sind keine sehr großen Unterschiede vorhanden. Für das finale Projekt sind vor allem die Größe des Arbeitsspeichers und die Leistung des Prozessors von Bedeutung. Desto besser diese Bauteile sind, desto schneller können die Berechnungen ausgeführt werden. Auch könnte es von Vorteil sein, einen Netzwerkanschluss zu besitzen, falls man das Projekt zukünftig noch erweitern möchte. Der Stromverbrauch ist für dieses Projekt nicht relevant, da der PI nicht dauerhaft in Betrieb sein wird. Der Arbeitsspeicher sollte für unsere Vorhaben ausreichend groß sein, jedoch könnte der Prozessor eventuell nicht für unser Vorhaben geeignet sein, da er nur einen Kern mit einer Taktrate von 1 GHz besitzt. Allerdings steht die Arbeit mit dem PI bei diesem Projekt im Vordergrund.

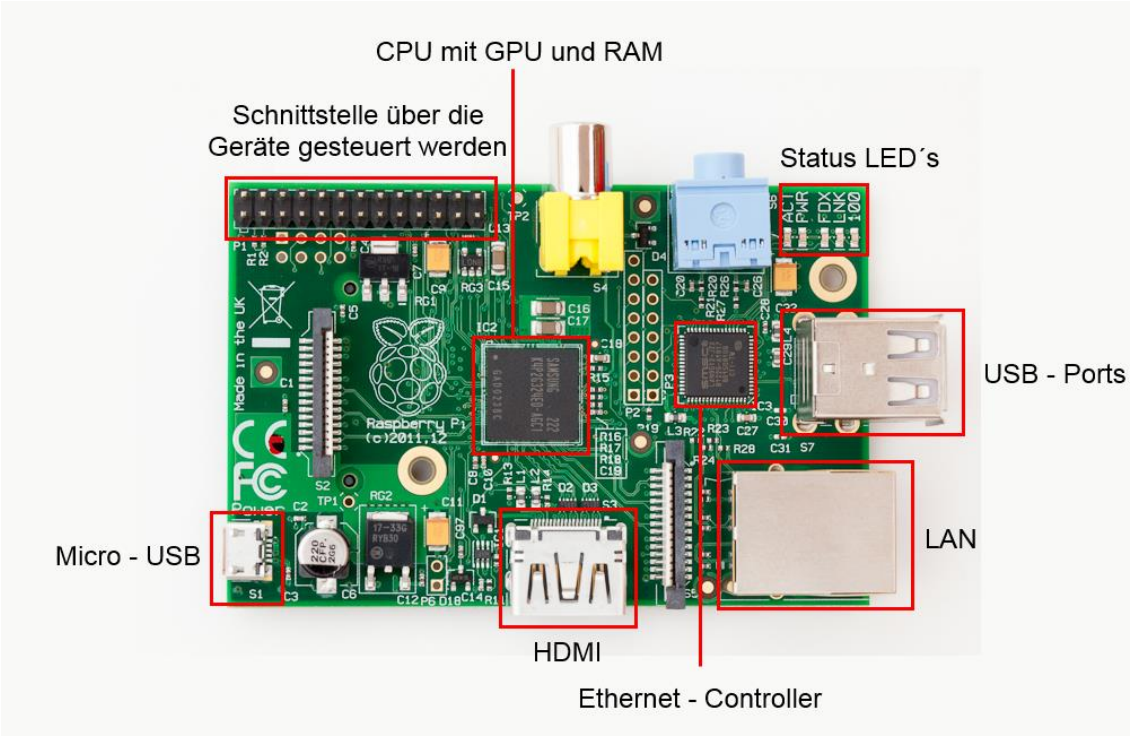

Abbildung 7 Aufbau Raspberry Pi

Quelle: http://skpang.co.uk/catalog/images/raspberrypi/1.png (modifiziert)

Um den Raspberry PI betreiben können benötigt man folgende Komponenten:

- USB – Tastatur (um Kommandos einzugeben) oder einen PC
- SD – Karte (Für das Betriebssystem)
- Monitor mit HDMI – Anschluss (oder DVI mit HDMI zu DVI – Adapter) + HDMI – Kabel oder einen PC
- USB – Netzteil (5V / 1000mA)

2.4.2 ANSTEUERUNG EINES SERVOS

Ein Servo ist ein positionsgeregelter Elektromotor (bzw. Aktuator), welcher über eine Steuerelektronik bedient wird. Um den Motor zu bewegen, übermittelt man der Steuerelektronik durch die Pulsweitenmodulation die genaue Position, zu der sich der Motor drehen soll. Dabei bewegt er sich in die Richtung, in die er eine geringere Entfernung zurücklegen muss. Diese Bewegung erfolgt solange, bis er den gewünschten Wert erreicht hat. Anschließend versucht der Servo diese Position zu halten. Das heißt, dass der Motor möglichen mechanischen Einwirkungen von außen entgegenwirkt. (Unbekannt, Modellbauservo, 2014)

Wie die Steuerelektronik den Motor ansteuert wird im nächsten Kapital erläutert.

Rechts ist der allgemeingültige Aufbau eines Servomotors abgebildet. Zu sehen ist die Steuerelektronik (1), der Elektromotor (2), der Potentiometer (3) und das Getriebe (4).

Ein Potentiometer ist ein verstellbarer Widerstand, wodurch je nach Position eine unterschiedliche Spannung erzeugt wird.

Durch diese spezifische Spannung, erkennt die Steuerungselektronik, in welcher Position sich der Motor momentan befindet.

Das Getriebe besteht je nach Qualität aus Kunststoff oder Metall und ist dem Steuerarm untersetzt.

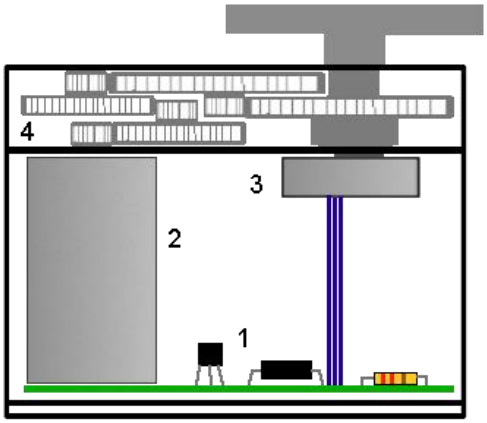

Abbildung 8 Aufbau eines Servos

Quelle: http://www.electronicsplanet.ch/Roboter/Servo/intern/querschn.png

2.4.3 Ansteuerung eines Gleichstrommotors

Um einen Gleichstrommotor ansteuern zu können verwendet man normalerweise eine H - Bridge.

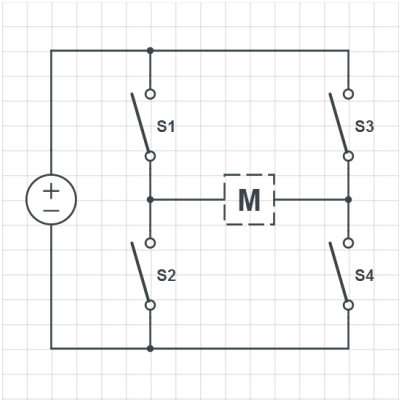

Abbildung 9 Aufbau einer H - BRIDGE

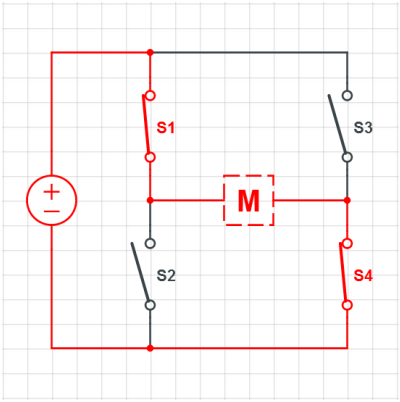

Abbildung 10 Rechtsdrehung des Motors

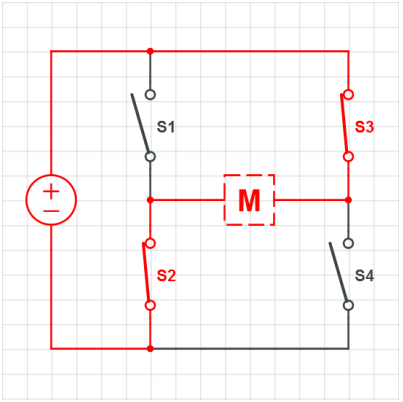

Abbildung 11 Linksdrehung des Motors

Erstellt mit CircuitLab, angelehnt an (Die elektronische Welt mit Raspberry Pi entdecken, 2013) s. 658

Sie besteht aus einem Motor, vier Schaltern und einer Spannungsquelle (siehe Abbildung 9).

Der Vorteil einer solchen Schaltung besteht darin, dass man den Motor in beide Richtungen laufen lassen kann, ohne eine Umpolung vornehmen zu müssen. Je nachdem welchen Schalter man schließt, kann man den Bewegungsfluss regeln. Wie sich welche Schalterkombination auswirkt, kann man der nachfolgenden Tabelle entnehmen.

Tabelle 3 Schalterkombinationen einer H-Bridge

S1	S2	S3	S4	Bewegung des Motors
offen	offen	offen	offen	Leerlauf
offen	offen	zu	zu	Kurzschluss der Spannungsquelle
offen	zu	offen	zu	Bremsung
offen	zu	zu	offen	Linksdrehung (Siehe Abb. 11)
zu	offen	offen	zu	Rechtsdrehung (Siehe Abb. 10)
zu	offen	zu	offen	Bremsung
zu	zu	offen	offen	Kurzschluss der Spannungsquelle

Angelehnt an (Die elektronische Welt mit Raspberry Pi entdecken, 2013) s. 658

Die Bremsung erfolgt durch das Überlagern der magnetischen Felder, welche bei der Rotation und beim Stromfluss entstehen.

3 Praktische Umsetzungen

3.1 Zielstellung des Praxisversuches

Im praktischen Teil der Arbeit soll das Ziel verfolgt werden, einen Raspberry Pi mit einer Kamera und zwei Servos zu verbinden und das Bildmaterial der Kamera mit Hilfe des Raspberry Pis auf Gesichter zu untersuchen. Anschließend soll die Kamera durch die beiden Servos neu justiert werden, so dass sich das Gesicht in der Mitte des Kamerabildes befindet. Sollten in der Kameraaufnahme mehrere Gesichter vorkommen, so soll eine Person durch ein bestimmtes Signal bestimmt werden können, welche verfolgt werden soll. Wenn keine Person festgelegt wird, so wird der Median der Koordinaten der Funde verwendet.

Das Signal, welches zur Bestimmung der Hauptperson dient, könnte zum Beispiel ein ausgestreckter Daumen nach oben sein. Um diese Bestimmung zu einem späteren Zeitpunkt aufheben zu können, soll ein ausgestreckter Daumen nach unten die Auswahl aufheben können.

3.2 Vorbereitungen und Tests

3.2.1 Einrichten der Entwicklungsumgebung

Für die Programmierung der Demoprogramme verwende ich die Programmiersprache Java und die Programmierumgebung Eclipse. Mein System basiert auf der 64-Bit-Architektur von Windows 7.

Zunächst muss OpenCV auf dem Test-PC installiert werden. Dazu lädt man sich als aller erstes die aktuellste Version herunter (http://opencv.org/downloads.html) und extrahiert diese an einen beliebigen Ort.

Anschließend muss der Java Build Path konfiguriert werden. Dazu öffnet man Eclipse und navigiert zu *Window – Preferences – Java – Build Path – User Libraries – New.* Dieser Bibliothek gibt man den Namen „OpenCV“ und bestätigt den Dialog.

Danach fügt man durch einen Klicken auf „Add External Jars…“ die „opencv-*versionscode*.jar“ aus *OPENCV/build/java* zu der Benutzerbibliothek hinzu.

Abschließend muss eine Native Bibliothek hinzugefügt werden. Dazu muss man zunächst „Native library location” auswählen um auf den „Edit“ – Button oben rechts klicken zu können. In dem geöffneten Dialog muss „External Folder“ angeklickt werden und je nach Betriebssystemarchitektur zu der richtigen „opencv_java*versionscode*.dll“ navigiert werden. Diese Anwendungserweiterung findet man in *OPENCV/build/java/x86 (32 – bit)* bzw. *OPENCV/build/java/x64* (64-bit).

Beim Erstellen eines neuen Java-Projektes muss nun die OpenCV – Bibliothek durch Klicken auf den „Add Library“ – Button im „Libraries“ – Reiter angeheftet werden. (Unbekannt, Using OpenCV Java with Eclipse, 2014)

3.2.2 TEST: EIN ERSTES OPENCV - DEMOPROGRAMM

Ziel dieses kleinen Programmes soll sein, sich mit der OpenCV – Bibliothek für Java einzuarbeiten. Dazu wird der Kamerastream einer Webcam ausgelesen, die Gesichtserkennung anwendet und die Resultate in einem Panel markiert ausgeben.

Die Hauptprozedur für das Einlesen der Kamerabilder und deren Markierung ist der Anhang 1 zu entnehmen. Als erstes wird ein *VideoCapture* – Objekt angelegt, welches als Konstruktor den gewünschten Kameraindex übergeben bekommt. Als zweites wird die *read* – Methode des Objektes aufgerufen. Wenn diese Methode den boolschen Wert *true* zurückgegeben hat, war die Abfrage erfolgreich und das *Mat* – Objekt kann an die private Methode *markHaarcascadesInMat* weitergegeben werden. Diese markiert in dem *Mat* – Objekt alle Funde des eingestellten Haarcascades und markiert diese mit einem hellgrünen Rechteck. Ein *Mat* – Objekt ist eine mehrdimensionale Matrix. Zum Schluss wird die Matrix mit den Rechtecken in dem Ausgabepanel angezeigt. Diese Schleife läuft solange das Programm nicht beendet oder die Aufnahme gestoppt wird. Wenn die Aufnahme beendet wird, wird der Zugriff auf die Webcam mit der Methode *release* freigegeben. (Cell0907, 2014)

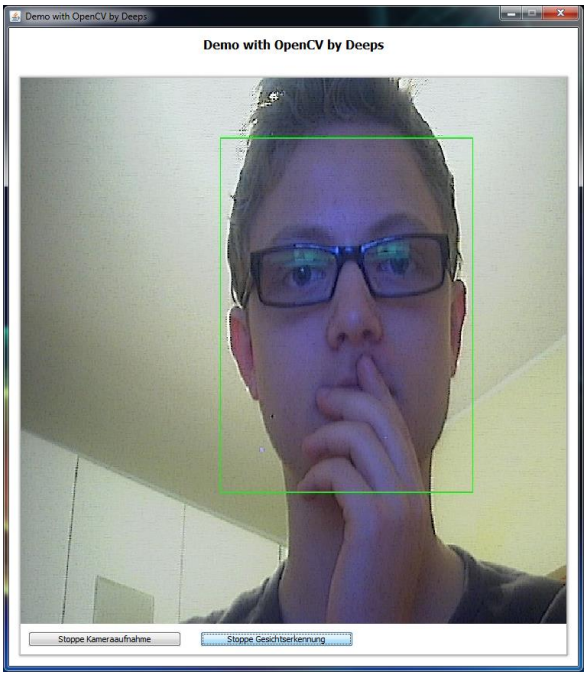

Abbildung 12 Screenshot des OpenCV - Demoprogramms

3.2.3 TEST: OPENCV – TESTPROGRAMM

Dieses Programm dient der Qualitätsüberprüfung von Haarcascades mit Hilfe der OpenCV – Bibliothek. Zu Beginn kann das Cascade, sowie die zu untersuchenden Bilder selbst ausgewählt werden. Außerdem ist eine Funktion erhältlich, welche dem Benutzer das Herunterladen von Bildern verschiedener Websites erleichtert, in dem alle *img* – Tags einer Internetseite betrachtet werden. Dazu wird die JSoup – Bibliothek genutzt. Mit Hilfe des Programms ist es schließlich möglich, einen vorhanden Datensatz an Bilder auf ein bestimmtes Objekt zu untersuchen. Der Anwender hat dabei die Möglichkeit False-Positives oder auch fehlende Positives zu markieren. Ist man mit der Analyse fertig und schließt das Programm, so öffnet sich eine Dialogbox, welche einem die Anzahl der False-Positives, der möglichen Positives und der tatsächlich gefundene Positives anzeigt. Mit diesem Programm wurde der Test der Gesichtserkennungsbibliothek durchgeführt.

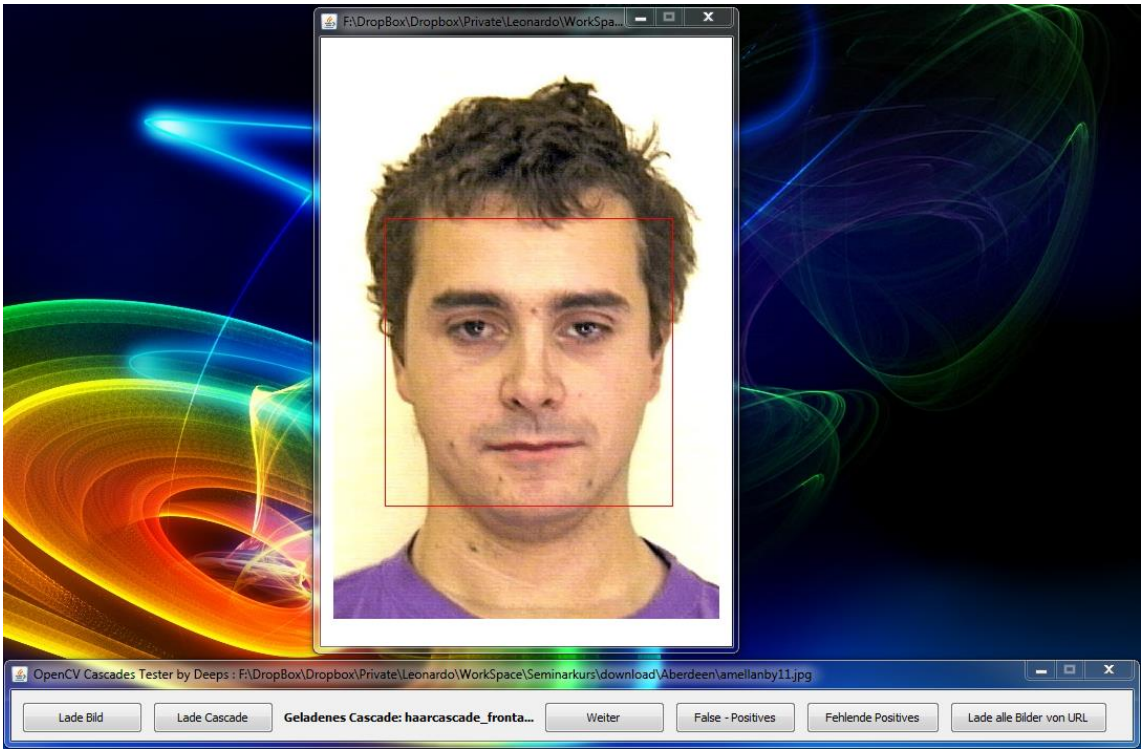

Abbildung 13 Screenshot des OpenCV - Testprogramms

3.2.4 TRAINIEREN EIGENER HAARCASCADES

Das Trainieren eines eigenen Cascades zur Erkennung eines bestimmten Objektes ist ein sehr rechenintensiver und aufwendiger Vorgang. Das Ziel dieses Teilschrittes ist die Erzeugung eines Cascades, welches einen nach oben ausgestreckten Daumen erkennt. Dieser Abschnitt muss nicht nachgestellt werden, da das Trainieren leider fehlschlug. Dennoch werde ich mein Vorgehen dokumentieren, um mögliche Fehlerquellen, die man beim Trainieren eigener Haarcascades machen kann, zu erwähnen.

Zunächst werden für das Trainieren zwei Datensätze benötigt: eine Sammlung von Bilder auf denen das gewünschte Objekt zu sehen ist (Positives) und eine Sammlung Bilder ohne diesem Objekt (Negatives). Damit das trainierte Cascade möglichst genau ist, werden dazu normalerweise mehrere tausende Bilder benötigt, was jedoch in dieser Arbeit nicht umsetzbar ist. Generell musste ich überlegen, wie man an einen möglichst großen Datensatz entsprechender Bilder kommt. Die Lösung des Problems besteht darin, ein Video aufzuzeichnen und dieses in seine einzelnen Frames aufzuteilen. Für die Aufnahme der beiden Videos, eines für die Positives und eines für die Negatives, wurde die Kamera eines Smartphones (Nexus 5) verwendet. Somit entstanden 2497 Negatives und 948 Positives. Für jedes einzelne Bild der Positives muss das Objekt nun markiert werden. Um diesen Schritt zu vereinfachen, wurde ein Programm geschrieben, welches ein MP4 - Video als Eingabe erwartet, es in die einzelnen Frames aufsplittet und anschließend jedes Bild dem Anwender anzeigt. Dieser muss nun mit der Maus das Objekt in dem Bild markieren. Dabei ist es mit Absicht noch nicht möglich, mehrere Objekte zu markieren, da die OpenCV – Bibliothek noch nicht mehr als ein Objekt verwerten kann. Um diese Daten verwenden zu können, werden zusätzlich zwei Textdateien (eine für die Positives, eine für die Negatives) von dem Programm angelegt, welche die Bilder auflisten und die Koordinaten der Objekte enthalten. Die Textdateien sind wie folgt aufgebaut:

Aufbau der infos_neg.txt

rel. Pfad zur Bilddatei

neg/003_frame.bmp

Aufbau der infos_pos.txt

rel. Pfad zur Bilddatei Anzahl der Objektfunde Koordinaten der Funde (x, y, width, height)

neg/003_frame.bmp 1 264 78 214 345

Die Ordnerstruktur sieht nun folgendermaßen aus:

- → analysed
  - → neg
    - 003_frame.bmp
    - 004_frame.bmp
    - …
  - → pos
    - 003_frame.bmp
    - 004_frame.bmp
    - …
  - → infos_neg.txt
  - → infos_pos.txt

Bei der Umsetzung des Quellcodes gab es anfänglich Probleme mit dem Aufteilen des Videos in die einzelnen Frames. Die Funktion, welche eigentlich bereits in OpenCV vorhanden sein sollte, existierte nicht. Dies führte dazu, dass die zusätzliche Bibliothek JavaCV [3] eingebunden werden musste. Ansonsten kam es zu keinen weiteren Problemen während des Programmierens. Für das Markieren der Objekte in den einzelnen Bildern habe ich etwa drei Stunden benötigt. Es muss beachtet werden, dass dieser Prozess recht anstrengend ist, da diese Arbeit sehr monoton ist. Man sollte sich die Arbeit am besten in verschiedene Etappen aufteilen und nicht, wie ich es getan habe, alles hintereinander abarbeiten.

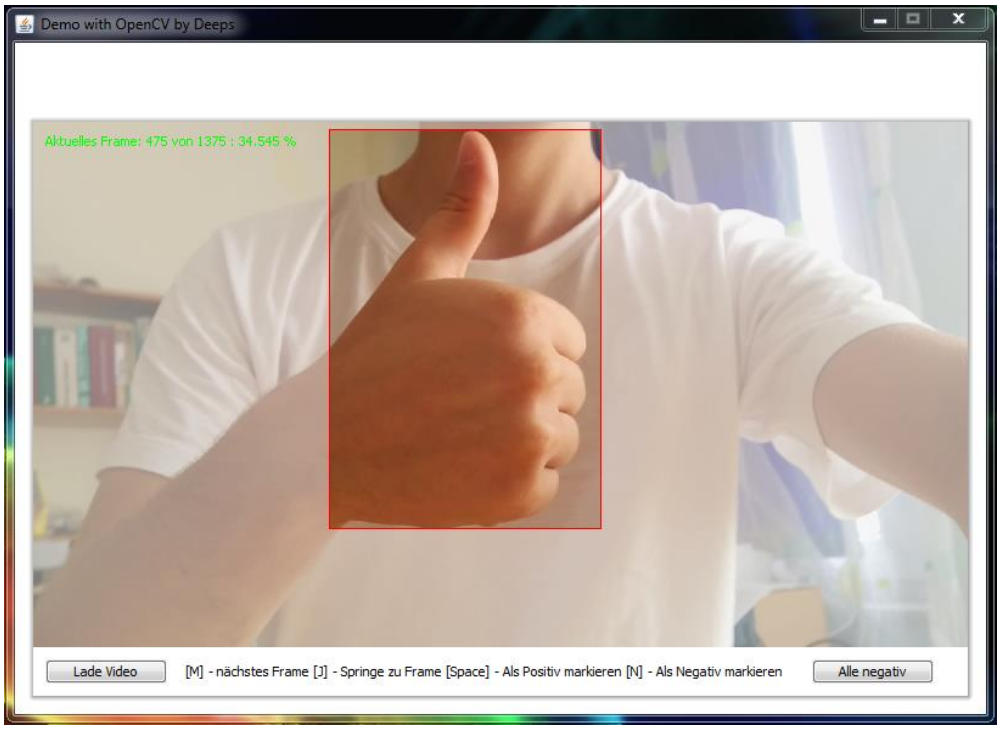

Abbildung 14 Screenshot des Trainingprogramms

[3] https://github.com/bytedeco/javacv

Ist jedoch alles geschafft, kann man die bearbeiteten Daten in das OpenCV – Verzeichnis verschieben. Bis hierhin sollte die Vorgehensweise korrekt sein.

Anschließend habe ich die Eingabeaufforderung von Windows geöffnet um anschließend folgenden Befehl einzugeben:

opencv_createsamples.exe -info analysed/infos_pos.txt -vec vector.vec -bg analysed/infos_neg.txt -w 43 -h 24 -num 948

Dieser Befehl dient dazu, die Datei *vector.vec* zu erstellen. Die Datei enthält den Datensatz von Bildern der Positives, welche für das eigentliche Trainieren benötigt werden. Die Parameter w und h stehen im selben Verhältnis, wie die FullHD – Bildgröße der Originaldateien.

Als nächstes habe ich den Trainingsvorgang gestartet.

opencv_haartraining.exe –data data/cascades -vec vector.vec -bg analysed/infos_neg.txt -npos 948 -nneg 2940 -nstages 30 -mem 900 -mode ALL -w 43 -h 24

Leider brach dieser fünf Stunden später mit folgender Fehlermeldung ab:

OpenCV Error: Assertion failed (elements_read == 1) in icvGetHaarTrainingDataFromVecCallback, file C:\builds\2_4_PackSlave-win32-vc12-shared\opencv\apps\haartraining\cvhaartraining.cpp, line 1859

Nach einer kurzen Recherche und einigen studierten Quellen später, habe ich den Vorgang mit dem Parameter „–npos 934“ fortgesetzt. Dieser brach jedoch auch wieder ab, woraufhin ich den Wert auf 850 reduziert habe. Nachdem die Berechnung der 18. Stage begonnen habe, ging es jedoch scheinbar nicht weiter. Selbst nach einer ganzen Woche Rechenzeit änderte sich nichts am Fortschritt. Dies führte dazu, dass ich den Prozess gestoppt habe. Da ich jedoch noch nicht aufgeben wollte, plante ich den Vorgang erneut in der Schule ausführen. Leider brach dort das Programm noch häufiger ab, als bei mir zuhause. Da eine Fernsteuerung leider nicht möglich und eine Fernwartung schwierig war, beendigte ich auch diesen Versuch.

Schlussendlich gelang es mir leider nicht, ein funktionierendes Cascade zu erzeugen. Mit reichlich mehr Zeit und besseren Kenntnissen in der Programmiersprache C hätte ich vielleicht bessere Resultate erzielen können, jedoch musste ich weitere Versuche aufgrund meines Zeitplanes einstellen. Bestärkt wurde diese Entscheidung durch den Fakt, dass dies mehr oder weniger nur ein Zusatz zu meinem Hauptprojekt ist.

3.2.5 ERSTE EINDRÜCKE

Das Installieren und Einrichten der OpenCV – Bibliothek lief problemlos. Auch die ersten Tests konnten ohne größere Probleme bewältigt werden. Problematisch wurde es vor allem bei dem Versuch, ein eigenes Cascade zu trainieren. Die Probleme mit denen ich zu kämpfen hatte, bestanden hauptsächlich in einer sehr lücken- und fehlerhaften Dokumentation der Bibliothek. So habe ich zum Beispiel erst am Ende meines Versuches erfahren, dass es ein bestimmtes Verhältnis gibt, in dem die Anzahl der Positives und Negatives stehen müssen.

„[…] So vec-file has to contain >= (numPose + (numStages-1) * (1 - minHitRate) * numPose) + S, where S is a count of samples from vec-file that can be recognized as background right away"

(http://code.opencv.org/issues/1834#note-8)

Eine vielleicht bessere, jedoch veraltete Anleitung bietet diese Quelle: http://note.sonots.com/SciSoftware/haartraining.html

Sie hat mir dabei geholfen, manche Probleme oder Sachverhalte in Bezug auf die OpenCV – Bibliothek zu verstehen. Sie schließt einige Lücken der offiziellen Dokumentation.
Des Weiteren könnte man sich bei einem zukünftigen Wiederaufgreifen dieses Problems mit der Verteilung der Prozessorlast auf mehrere Kerne beschäftigen. Dies ist ebenfalls noch nicht standardmäßig mit OpenCV möglich, soll jedoch durch Umwege bzw. umschreiben umsetzen lassen.

Ein weiterer Nachteil besteht darin, dass man die bereits berechneten Daten nicht einfach in ein verwendbares Cascade konvertieren kann. Es gibt zwar dafür die Datei *convert_cascade.c*, allerdings lässt sich diese nicht fehlerfrei kompilieren.

3.2.6 EINFÜHRUNG IN PYTHON

Um den Raspberry Pi programmieren zu können, ist es erforderlich sich mit den Grundlagen der Programmiersprache Python auseinanderzusetzten. Die Sprache weist Parallelen zu Java auf und ist durch die besonders übersichtliche Syntax für Programmieranfänger geeignet.
Mit der Version Python 3.0 wurden größere Änderungen an der Sprache vorgenommen. Man sollte daher auch darauf achten, ob eine Bibliothek Python 2 oder Python 3 oder beide Versionen unterstützt. Um den Einstieg in das Programmieren mit Python zu erleichtern, bietet die University of Waterloo einen kostenlosen Onlinekurs[4] für jede interessierte Person an. Bei der Übersetzung ins Deutsche wurde die Universität vom Bundeswettbewerb der Informatik unterstützt.
Ich selbst habe den Python – Kurs begonnen, um die Syntax der Sprache zu erlernen und kann diesen nur weiterempfehlen.

3.2.7 EINRICHTEN DES RASPBERRY PI

Für das finale Projekt werden folgende Bibliotheken benötigt: *OpenCV* für die Gesichtserkennung und *I2C – Tools* für die Servomotoransteuerung. Bevor man jedoch mit der Installation der Bibliotheken beginnen kann, sollte man ein Update des Pis durchführen.

3.2.7.1 Durchführung eines Updates

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get update
```

Dieser Vorgang kann, je nach dem wann das letzte Update erfolgte und wie schnell die Internetanbindung ist, mehrere Minuten oder auch Stunden dauern. Generell sollte man sich etwas mehr Zeit dafür nehmen, oder abwägen, ob das Update auch nach der eigentlichen Arbeit durchgeführt werden kann. Grundsätzlich empfiehlt es sich jedoch seinen Raspberry Pi aktuell zu halten.

[4](Centre for Education in Mathematics and Computing; BwInf, 2014)

3.2.7.2 Installation von OpenCV

Zuerst müsse weitere Bibliotheken geladen werden, welche für OpenCV benötigt werden.

```
sudo apt-get -y install build-essential cmake cmake-curses-gui pkg-
config libpng12-0 libpng12-dev libpng++-dev libpng3 libpnglite-dev
zlib1g-dbg zlib1g zlib1g-dev pngtools libtiff4-dev libtiff4
libtiffxx0c2 libtiff-tools libeigen3-dev
```

```
sudo apt-get -y install libjpeg8 libjpeg8-dev libjpeg8-dbg libjpeg-
progs ffmpeg libavcodec-dev libavcodec53 libavformat53 libavformat-dev
libgstreamer0.10-0-dbg libgstreamer0.10-0 libgstreamer0.10-dev
libxine1-ffmpeg libxine-dev libxine1-bin libunicap2 libunicap2-dev
swig libv4l-0 libv4l-dev python-numpy libpython2.6 python-dev
python2.6-dev libgtk2.0-dev
```

Anschließend wird die OpenCV – Bibliothek heruntergeladen und unter “opencv-2.4.8.zip” auf dem Desktop des Pis gespeichert.

```
wget http://sourceforge.net/projects/opencvlibrary/files/opencv-
unix/2.4.8/opencv-2.4.8.zip/download Desktop/opencv-2.4.8.zip
```

Danach wird das zip – Archiv entpackt und in den OpenCV – Ordner navigiert.

```
unzip opencv-2.4.8.zip
cd opencv-2.4.8
```

```
mkdir release
cd release
ccmake ../
make
sudo make install
```

Der letzte Befehl startet den Installationsvorgang. Der Vorgang dauert dabei am längsten und benötigt ungefähr zehn Stunden. (Castle, 2015) Dieser Prozess kann jedoch auch unterbrochen und zu einem späteren Zeitpunkt durch erneutes ausführen fortgesetzt werden.

3.2.7.3 Installation der I2C – Tools

Zunächst werden die benötigten Bibliotheken geladen und installiert.

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

Als zweites muss zum Ordner *modprobe.d* in *etc* navigiert werden.

```
cd ../../../etc/modprobe.d
```

Nun wird die Datei *raspi-blacklist.conf* mit dem Editor geöffnet und die Zeile *blacklist i2c-bcm2708* kommentiert, indem ein „#“ – Zeichen vor die Zeile gesetzt wird.

```
sudo nano raspi-blacklist.conf
```

Anschließend wird die Datei mit Strg + O gespeichert. Nun muss die *modules* – Datei in *etc* geöffnet werden.

```
cd ..
sudo nano modules
```

In dieser Datei werden die folgenden beiden Zeilen ergänzt und die Änderungen der Datei durch Drücken der Tastenkombination Strg + O gesichert.

```
i2c-dev
i2c-bcm2708
```

Die Einrichtung ist nun abgeschlossen. Hat alles geklappt, so sollte folgender Befehl eine Matrix ausgeben. (Townsend, 2014)

```
sudo i2cdetect -y 0
```

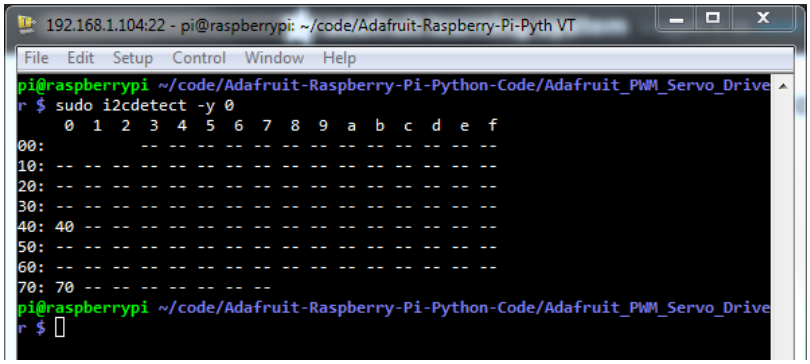

Abbildung 15 Ergebnis des i2cdetect – Befehls

Quelle:
https://learn.adafruit.com/system/assets/assets/000/001/700/medium800/raspberry_pi_i2cdetect.png?1396775470

3.2.7.4 Installation weiterer hilfreicher Software

3.2.7.4.1 Screen

Mit Screen kann man mehrere SSH – Verbindungen zum Raspberry Pi aufbauen. Beim Trennen dieser Verbindungen sorgt Screen außerdem dafür, dass die ausgeführten Befehle weiterlaufen, da sie sonst beendet werden würden. (Raspberryhelp1, 2014)

Zur Installation genügt folgender Befehl:

```
sudo apt-get install screen
```

Zur Verwendung von Screen empfehle ich das Handbuch zur Bedienung von Screen.[5]

3.2.7.4.2 TightVNCServer

Mit Hilfe von TightVNCServer kann man die grafische Ausgabe des Raspberry Pis über das Netzwerk umleiten. Dies ist vor allem dann hilfreich, wenn man nicht mehrere Monitore verwenden will und ermöglicht eine effizientere Arbeitsweise. Um TightVNCServer zu installieren muss man folgenden Befehl im Terminal ausführen.

```
sudo apt-get install tightvncserver
```

Anschließend startet man die Software mit:

```
tightvncserver
```

Um eine VNC – Session zu starten muss man folgende Anweisung anwenden.

```
vncserver :1 -geometry 1024x728 -depth 24
```

Um sich anschließend mit einem PC verbinden zu können, muss man noch die Berechtigungen anpassen. Der Einfachheit halber werden wir die Zugänge für alle freischalten.

```
xhost +
```

Zum Verbinden ist eine externe Software[6] notwendig.

Sollte anfangs kein Monitor und keine Tastatur vorhanden sein, so bietet sich nmap[7] als Software für den PC an. Mit nmap kann man das eigene Netzwerk nach belegten IP – Adressen untersuchen und erhält umfassende Informationen zu den verbundenen Geräten.

[5] http://wiki.ubuntuusers.de/screen
[6] http://www.tightvnc.com/download.php
[7] http://nmap.org/download.html

Um das Arbeiten noch effizienter zu gestalten kann man Putty [8] und WinSCP [9] verwenden. Mit Putty kann man sich zu dem Raspberry Pi verbinden, um Befehle einzugeben, welche keinerlei grafische Ausgabe erzeugen, oder den VNC – Server einzurichten. WinSCP ist ein Dateimanager, mit dem sich die Daten auf dem Raspberry Pi leichter bearbeiten lassen.

3.2.7.5 Verbindung des Raspberry Pis mit dem Servoboard

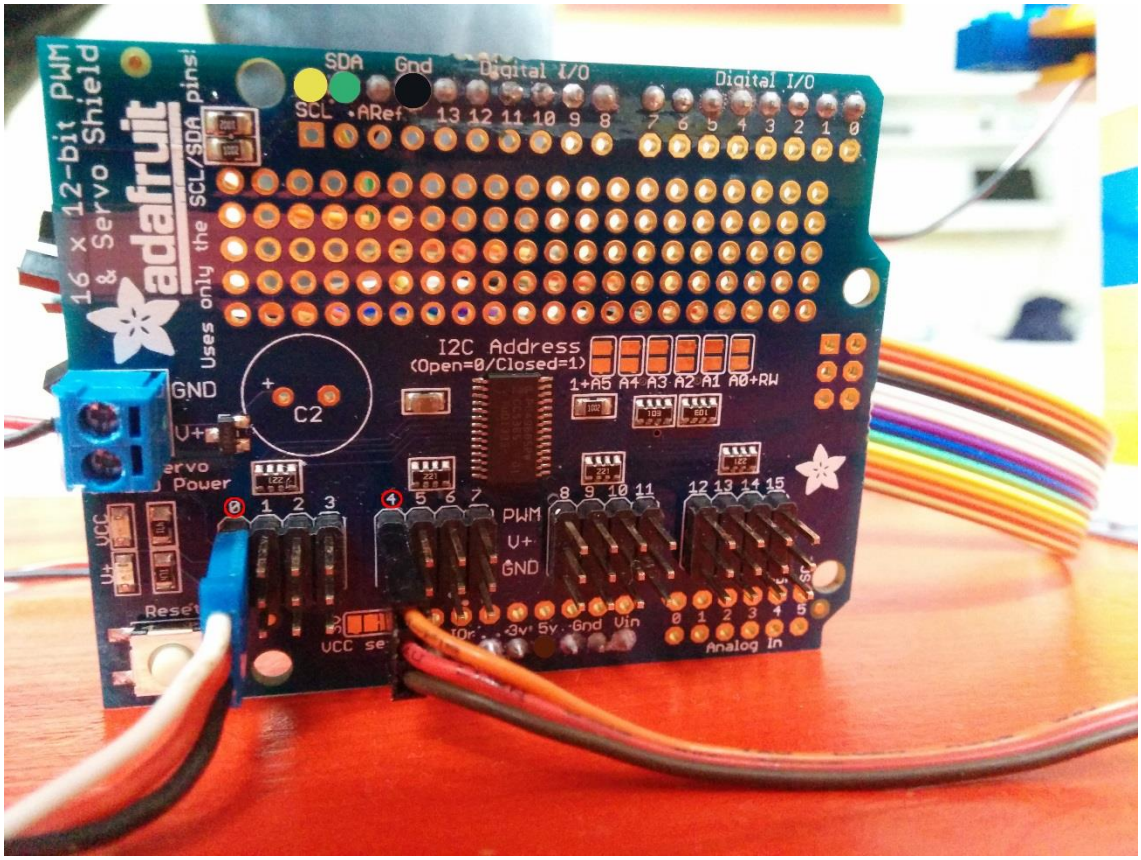

Abbildung 16 Anschluss der Kabel am Servoboard mit farbigen Markierungen

Um den Raspberry mit dem Servoboard zu verbinden benötigt man vier Kabel. Es ist erforderlich die Pins GND, SCL, SDA und VCC (bzw. 5V) zu verbinden. (Townsend, 2014) Der Abbildung 18 kann man des Weiteren die Channels (0, 4) der Servoanschlüsse entnehmen. Um die Zuordnung der Anschlüsse nachzuschlagen, befinden sich in den Anhängen vier bis sechs weitere Informationen zur PIN – Belegung und Verschaltung des Servoboards mit dem Raspberry Pi.

Abbildung 17 Anschluss des Servoboards am Pi

[8] http://www.putty.org/
[9] http://winscp.net/eng/docs/lang:de

3.2.8 TEST: I2C - DEMOPROGRAMM ZUR ANSTEUERUNG DER SERVOS

Mit diesem kleinen Programm soll die Funktionsweise der programmiertechnischen Ansteuerung der Servos getestet und sich mit der I2C – Bibliothek auseinandergesetzt werden. Dazu sollen die Servos verschiedene Positionen anfahren.

Zunächst werden die benötigten Module importiert. Anschließend wird ein PWM – Objekt mit dem Verweis zur Adresse 0x40 initialisiert. Die Variable *servoMin* gibt die Position des servospezifischen linken Anschlags an, *servoMax* gibt die Position des rechten Anschlags an. Diese Werte sollten dem Handbuch entnommen werden. Ein Senden von Pulsweiten außerhalb diesen Werten kann zur Schädigung des Servos führen! Die Methode *setPWMFreq* ist sehr wichtig. Wird diese nicht aufgerufen, kann es auch hier zu fahrlässigen Fehlern kommen, welche den Servo beschädigen können.

Nach dem Setzen der Frequenz werden in einer Endlosschleife die einzelnen Positionen angefahren. Wie kleinschrittig dabei vorgegangen wird hängt davon ab, wie groß die *step* – Variable gewählt wurde. Diese muss sich jedoch im Intervall von 1 bis 220 befinden, was der Differenz von servoMax und servoMin entspricht. Beendet werden kann das Programm durch Drücken der Tastenkombination *Strg* + *C*. Der Quellcode zu diesem Programm befindet sich im Anhang 2.

3.2.9 AUFBAU EINER KONSTRUKTION MIT LEGO

Für die Zielstellung ist ein Aufbau erforderlich, der es ermöglicht eine Plattform, auf die eine Kamera montiert werden kann, in der z-Achse und der x-Achse zu rotieren. Da die Rotationen durch einen Servo erfolgen sollen, ist eine entsprechende Verbindung einzuplanen. Wir verwenden hierfür stabilen Stahldraht, wie man es aus dem Modellbau kennt. Das Aufbauen der Konstruktion hat bei mir aufgrund mangelnder Erfahrungen mit der Materie länger gedauert als geplant. Man sollte daher Kenntnisse im Bauen mit Lego und Servos mitbringen.

Die Plattform muss drehbar gelagert und ausgewuchtet sein. Die Servos benötigen eine feste Halterung und bedienen über einen geeigneten Draht die Plattform. Meine Lösung ist bereits ausreichend stabil und wird später wahrscheinlich noch mit speziellen aus dem 3D – Drucker stammenden Legobausteinen für die Servos aufgerüstet. Für ein „Proof-of-Concept“ funktioniert die Konstruktion jedoch sehr gut.

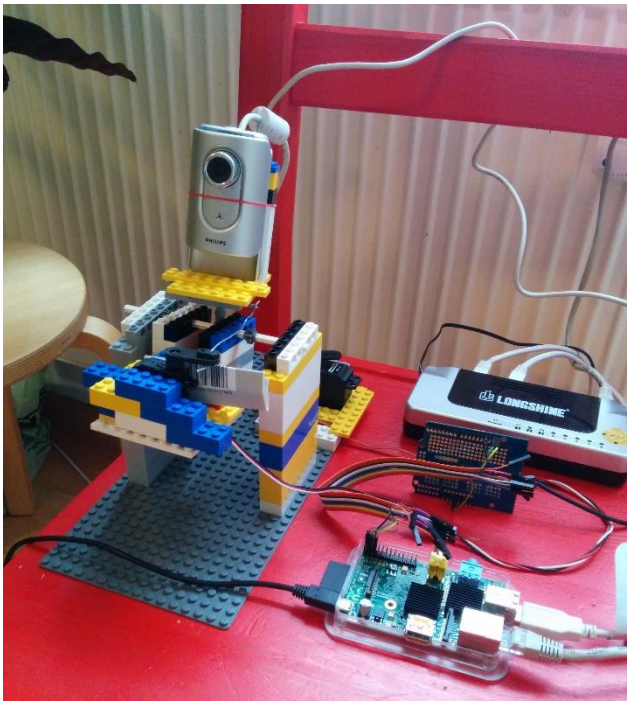

Abbildung 18 Gesamtaufbau

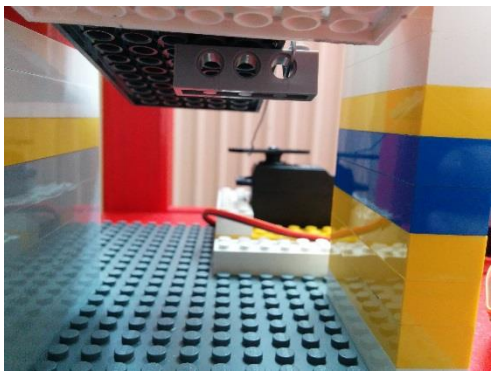

Abbildung 19 Verbindung des X-Servos mit der Plattform

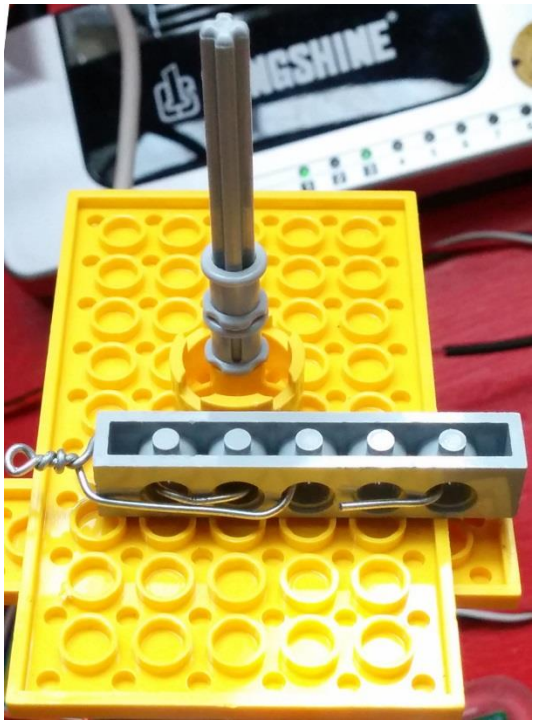

Abbildung 20 Verbindungsstelle der oberen Plattform zum Z-Servo

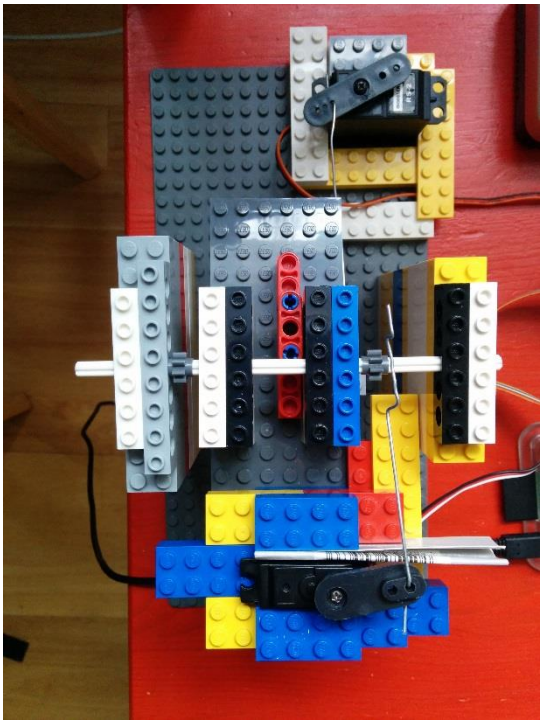

Abbildung 21 Lager für die obere Plattform

3.3 Finales Programm

Mittlerweile ist es bereits möglich, den Livestream einer Webcam einlesen, das Material auf Gesichter untersuchen und die Servos ansteuern. In diesem Abschnitt werden nun alle Teilmodule in ein finales Projekt integriert. Da das Erstellen eines eigenen Haarcascades nicht geklappt hat, werden wir immer den Median der Positionen aller gefundenen Gesichter nehmen.

Als erstes habe muss das Servo – Ansteuerungs – Modul umgeschrieben werden. Es wird um die Methoden *moveRight, moveLeft, moveUp, moveDown* erweitert. Diese Bewegungen werden mit einer variablen Geschwindigkeit ausgeführt. Anschließend wird das Beispiel *facedetect.py* neu geschrieben, sodass es nun möglich ist, als Eingabeparameter die Breite und Höhe des zu bearbeitenden Bildes festzulegen, sowie anzugeben, ob ein Fenster mit der visuellen Auswertung angezeigt werden soll, oder nicht. Der Hauptalgorithmus läuft in einer Endlosschleife und fordert in jedem Durchlauf ein Kamerabild an, welches er anschließend auf Gesichter untersucht und entsprechend die Servos bewegt.
Da die Webcam mit 27 fps aufzeichnet, der Pi jedoch nur eine bestimmte Anzahl an Bildern verarbeiten kann, könnte man nach jedem Durchlauf der Schleife die überflüssigen Bilder aus dem Puffer löschen, um Verzögerungen zwischen den Bewegungen und dem Aufgezeichneten zu vermeiden. Das ist trotzdem zeitintensiv, da die Bilder aus dem Stream geladen werden und den Hauptthread blockieren. Man könnte zur Beschleunigung den Stream wiederholt freigeben, und erneut öffnen, um das aktuellste Bild zu erhalten, was durchaus schneller ist. Dies führt jedoch nach einer gewissen Zeit zu einem *broken pipe* – Fehler. Als dritten Lösungsweg gäbe es die Möglichkeit einer Parallelisierung mithilfe von Threads. Dies macht aber für den Raspberry Pi Revision B keinen Unterschied, da er eine Singlecore-CPU besitzt. Ich habe mich für den ersten und zweiten Lösungsweg entschieden, sodass der Anwender selbst entscheiden kann, ob er die Hardware schonen, oder bessere Resultate erzielen möchte. Ich habe außerdem sehr lange darüber nachgedacht, ob es nützlich ist, die Kamera bei Sichtverlust des Gesichtes in die zuletzt bewegte Richtung weiterzubewegen, um das Gesicht zu „suchen“. Ich bin jedoch zu dem Entschluss gekommen, dass man den Kopf genauso gut in die andere Richtung bewegen könnte, und sich die Kamera somit in die falsche Richtung bewegen würde. Außerdem besteht die Möglichkeit, dass man sich bewusst aus dem Bild bewegt hat und die Kamera an der Position bleiben soll. Dies führte dazu, dass ich diese Funktion wieder aus der Implementierung entfernt habe.

Die Breite und Höhe des aufgenommenen Bildes beeinflusst maßgeblich, wie weit entfernt die Gesichter sein können. Je höher man die Parameter wählt, desto größer darf der Abstand zur Kamera sein. Allerdings hat dies starke Auswirkungen auf die Performanz des Programms. Bei einer Dimensionierung von 320x240 läuft das Programm mit fünf fps recht stabil. Der Abstand sollte bei guten Lichtverhältnissen (Tageslicht) jedoch nicht größer als 2 m sein. Verwendet man jedoch eine Breite von 1020 und eine Höhe von 720 so können die Gesichter weiter entfernt sein. Die Quantität der bearbeitenden Bilder nimmt dadurch jedoch ab. Der Pi kann bei dieser Auflösung nur noch ein Frame innerhalb ein bis zwei Sekunden verarbeiten. Je mehr Bilder innerhalb einer Sekunde verarbeitet werden können, desto schneller reagiert der Pi auf die Bewegungen.
Die Lichtverhältnisse sind aufgrund des verwendeten Algorithmus zu beachten. Die besten Resultate wurden bei Tageslicht erzielt.

Das Programm funktioniert an sich recht gut. Man merkt jedoch schnell, dass der Raspberry Pi nicht für solche komplexen Aufgaben konzipiert ist. Der Quellcode des Programmes ist in der Anhang 3 einsehbar.

3.4 Das Fazit

Die Arbeit mit dem Raspberry Pi war interessant und hat die Freude auf weitere Projekte solcher Art geweckt. Die Arbeit mit der OpenCV – Bibliothek hingegen hat aufgrund der mangelnden bzw. veralteten Dokumentation überhaupt keinen Spaß gemacht. Man musste ständig auf anderweitige Forenbeiträge im Internet zu den gleichen Fehlern und Problemen zurückgreifen, um zu einem Ergebnis zu kommen.

Man könnte dieses Projekt erweitern, indem man die Kamera zum Beispiel auf ein ferngesteuertes Auto oder einen Quadrocopter platziert. Dadurch könnte man sich dann auch im Raum bewegen. Dieses Konzept könnte in ausgereifter und weiterentwickelter Form zum Beispiel bei der Beobachtung von Tieren in einem größeren Gebiet genutzt werden.

4 Anregungen/ Pi - Projekte anderer

4.1 Smarter wohnen – Der Inneneinrichtung sprachgesteuert Befehle geben

Das auf LightwaveRF basierende System steuert in dem Beispielvideo[10] Fernseher, Licht und Steckdosen. Die Befehle werden mit Hilfe eines Mikrofons aufgenommen und von einem Raspberry Pi ausgewertet. Die Steuerung des menschlichen Umfeldes erachte ich als sehr spannend und würde dies gerne zukünftig selbst einmal umsetzten.

4.2 Ein eigenes Mobilfunknetz aufbauen

Abbildung 22 Der "Sendemast" des Projektes

Quelle: http://www.paconsulting.com/our-experience/how-to-shrink-a-base-station-into-a-raspberry-pi/

Dieses Projekt wurde von einem Team des PA Technology Centre umgesetzt. Es ist ihnen gelungen, ein eigenes Mobilfunknetz aufzubauen, in dem zwei Geräte miteinander telefonieren konnten. Dies musste jedoch in einem abgeschirmten Raum realisiert werden, da es illegal ist, auf den Frequenzen der normalen Mobilfunkmasten zu senden und zu empfangen.

[10] https://www.youtube.com/watch?v=gZkwvSX0_Os

GLOSSAR

Aktuator
: Bauelement zur Umwandlung von elektrischen Signalen in mechanische Bewegung, Druck oder Temperatur. (Unbekannt, Aktor, 2014) 20

BSD
: "Software unter BSD-Lizenz darf frei verwendet werden. Es ist erlaubt, sie zu kopieren, zu verändern und zu verbreiten. Einzige Bedingung ist, dass der Copyright-Vermerk des ursprünglichen Programms nicht entfernt werden darf. Somit eignet sich unter einer BSD-Lizenz stehende Software auch als Vorlage für kommerzielle (teilproprietäre) Produkte. (Unbekannt, BSD - Lizenz, 2014) 16

Cascade / Haarcascade
: ist ein englischer Begriff und bezeichnet die Klassifikatoren des Viola-Jones-Algorithmus. Der Begriff beschreibt ursprünglich einen Stufenwasserfall. Vermutlich wurde der Begriff deshalb gewählt, da auch das Erkennen beim Viola-Jones-Algorithmus in Stufen stattfindet. 40

CircuitLab
: CircuitLab ist eine Online - Lösung zum Erstellen eigener elektronischer Schaltungen (Unbekannt, CircuitLab - online schematic editor & circuit simulator, 2014) 21

EEG
: EEG bedeutet Elektroenzephalografie und wird zum Messen elektrischer Aktivitäten im Gehirn verwendet. (Unbekannt, Elektroenzephalografie, 2014) 9

False-Positives
: Funde einer Erkennungssoftware, die in der Realität jedoch nicht zutreffen. (Funde, wo keine sind) 16

fMRT
: fMRT bedeutet funktionelle Magnetresonanztomographie und ermöglicht es aktive Hirnareale räumlich darzustellen 9

Geon
: Ein Geon ist ein geometrischer Grundbaustein mit ansichtsunabhängigen Eigenschaften. Geone können auch erkannt werden, wenn die 2D-Projektion Artefakte aufweist oder Merkmale fehlen. Diese Merkmale bestehen aus Kanten. Mehrere Geonen dienen der strukturellen Beschreibung von Objekten. 13

High-Pegel
: Um digitale Signale in analoge umwandeln zu können, verwendet man zwei konstante, unterschiedliche elektrische Spannungen. Die höhere Spannung nennt man High-Pegel, die niedrigere Low-Pegel *Siehe* Pulsweitenmodulation

OpenCV
: OpenCV ist eine Bibliothek, welche bereits implementierte Algorithmen für die Bildverarbeitung enthält. 16

Pulsweitenmodulation
: Zur Pulsweitenmodulation wird zunächst eine Pulsweite (Dauer) definiert, in der eine Rechtecksspannung anliegt. Anschließend legt man eine Frequenz fest, welche angibt, wie oft in diesem Intervall die Spannung abgetastet werden soll. Je nach Länge des High-Pegels lässt sich nun der prozentualer Anteil an der Pulsweite berechnen. Diese Information kann zur Steuerung von Servomotoren verwenden werden. Somit kann man analoge Signale der Servorsteuerung ohne große Verluste in digitale umwandeln. (Unbekannt, Pulse-width modulation, 2014) 20

LITERATURVERZEICHNIS

(2012). In P. T. Hans-Otto Karnath, *Kognitive Neurowissenschaften* (S. 130 ff.). Springer-Verlag.

(2013). In E. Bartmann, *Die elektronische Welt mit Raspberry Pi entdecken* (S. 658 f.). O´REILLY Basics.

Adamski, K. (30. Oktober 2014). *7-38-55 Botschaften und Kommunikationswirkung*. Von Bundesverband der Medientrainer in Deutschland e.V.: http://www.bmtd.de/7-38-55_botschaften_und_kommunikationswirkung abgerufen

Castle, R. (05. Oktober 2015). *Installing OpenCV on a Raspberry Pi*. Von Robert Castle Consulting augmented reality, computer vision, Unity, mobile and desktop apps: http://robertcastle.com/2014/02/installing-opencv-on-a-raspberry-pi/ abgerufen

Cell0907. (04. August 2014). *Detecting faces in your webcam stream with OpenCV/java and displaying back the results*. Von Cell0907 A bit of everything: http://cell0907.blogspot.de/2013/07/detecting-faces-in-your-webcam-stream.html abgerufen

Centre for Education in Mathematics and Computing; BwInf. (11. Oktober 2014). *0: Hallo, Welt!* Von Computer Science Circles: http://cscircles.cemc.uwaterloo.ca/0-de/ abgerufen

Grüter, D. T. (31. Juli 2014). *Zur Neuropsychologie der Gesichtserkennung*. Von Prosopagnosie: http://www.prosopagnosie.de/forschung12.html abgerufen

Hammerschmidt, A. (01. August 2014). Von Raspberry Pi Guide Dein Einstieg in die Welt der Microcontroller: http://raspberrypiguide.de/ abgerufen

Kumar, S. (04. August 2014). *Converting OpenCV Mat to BufferedImage of Java*. Von Impetus No one has ever become poor by giving. -Anne Frank: http://sumitkumariit.blogspot.de/2013/08/coverting-opencv-mat-to-bufferedimage.html abgerufen

Leyh, A. (04. August 2014). *Wer ist das?* Von das gehirn Der Kosmos im Kopf: http://dasgehirn.info/wahrnehmen/sehen/die-gesichtserkennung-wer-ist-das-1/ abgerufen

Raspberryhelp1. (05. Oktober 2014). *Raspberry Pi Screen (Keep your processes running after disconnect)*. Von Raspberry Pi Help For your Pi!:

http://raspberrypihelp.net/tutorials/37-raspberry-pi-screen-keep-your-processes-running-after-disconnect abgerufen

Tobias Groß, B. M. (04. September 2014). *Viola-Jones Gesichtserkennung mit WebGL*. Von Department of Computer Science 12: https://www12.informatik.uni-erlangen.de/edu/map/ss13/talks/Viola-Jones_Gesichtserkennung_mit_WebGL.pdf abgerufen

Townsend, K. (05. Oktober 2014). *Adafruit 16 Channel Servo Driver with Raspberry Pi*. Von Adafruit Industries: https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi/configuring-your-pi-for-i2c abgerufen

Unbekannt. (04. September 2014). *About*. Von OpenCV: http://opencv.org/about.html abgerufen

Unbekannt. (01. August 2014). *Aktor*. Von Wikipedia Die freie Enzyklopädie: http://de.wikipedia.org/wiki/Aktor_%28Technik%29 abgerufen

Unbekannt. (04. September 2014). *BSD - Lizenz*. Von Wikipedia Die freie Enzyklopädie: http://de.wikipedia.org/wiki/BSD-Lizenz abgerufen

Unbekannt. (04. August 2014). *CircuitLab - online schematic editor & circuit simulator*. Von CircuitLab: https://www.circuitlab.com/ abgerufen

Unbekannt. (1. August 2014). *Elektroenzephalografie*. Von Wikipedia Die freie Enzyklopädie: http://de.wikipedia.org/wiki/Elektroenzephalografie abgerufen

Unbekannt. (19. Oktober 2014). *Modellbauservo*. Von electronicsplanet: http://www.electronicsplanet.ch/Roboter/Servo/intern/intern.htm abgerufen

Unbekannt. (31. Juli 2014). *Prosopagnosie*. Von Wikipedia Die freie Enzyklopädie: http://de.wikipedia.org/wiki/Prosopagnosie abgerufen

Unbekannt. (1. August 2014). *Pulse-width modulation*. Abgerufen am 23. Juni 2014 von Wikipedia Die freie Enzyklopädie: http://en.wikipedia.org/wiki/Pulse-width_modulation

Unbekannt. (01. August 2014). *Raspberry Pi Modell A*. Von Idealo: http://www.idealo.de/preisvergleich/OffersOfProduct/3575975_-modell-a-raspberry-pi-foundation.html abgerufen

Unbekannt. (01. August 2014). *Raspberry Pi Modell B*. Von Idealo: http://www.idealo.de/preisvergleich/OffersOfProduct/3575856_-modell-b-version-2-0-raspberry-pi-foundation.html abgerufen

Unbekannt. (24. September 2014). *Recognition by Components*. Von Institut für Simulation und Grafik: http://wwwisg.cs.uni-magdeburg.de/bv/files/LV/Grundlagen_der_Computer_Vision/VL/L10_Recognition%20by%20Components.pdf abgerufen

Unbekannt. (04. August 2014). *Using OpenCV Java with Eclipse*. Von OpenCV: http://docs.opencv.org/doc/tutorials/introduction/java_eclipse/java_eclipse.html#java-eclipse abgerufen

Unbekannt. (04. September 2014). *Viola–Jones object detection framework*. Von Wikipedia Die freie Enzyklopädie: http://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework abgerufen

Walter, M., & Kesser, H. (04. September 2014). *Viola Jones*. Von Department of Computer Science 12: https://www12.informatik.uni-erlangen.de/edu/mpa/ss11/talks/violajones.pdf abgerufen

ANHANG

ANHANGSVERZEICHNIS

ANHANG 1 JAVA - QUELLCODE ZUM OPENCV - DEMOPROGRAMM

```
private static void record() {
	new Thread(new Runnable() {
		@Override
		public void run() {
			VideoCapture videoCapture = new VideoCapture(0);
			Mat currentImage = new Mat();
			panelInterface.recordingHasStarted();
			while (isRecording) {
				if (!videoCapture.read(currentImage)) {
					stopRecording();
					JOptionPane.showMessageDialog(
					null,
						"Kamera wurde entfernt.",
						"Verbindung verloren",
						JOptionPane.ERROR_MESSAGE);
					break;
				}
				if (panelInterface == null) {
					stopRecording();
					break;
				}
				if (isFaceDetecting)
					markHaarcascadesInMat(currentImage);
				try {
					panelInterface
					  .setCurrentFrame(
					    convertMatToBufferedImage(currentImage));
					  //Übernommen von (Kumar, 2014)
				} catch (IOException e) {
					e.printStackTrace();
				}
			}
			videoCapture.release();// sollte allerdings bereits
					automatisch geschehen
		}
	}).start();
}
```

ANHANG 2 PYTHON - QUELLCODE ZUM I2C – DEMOPROGRAMM

```
from Adafruit_PWM_Servo_Driver import PWM
import time

pwm = PWM(0x40, debug=True)
servoMin = 220  # Min pulse length out of 4096  # 220
servoMax = 440  # Max pulse length out of 4096  # 440

pwm.setPWMFreq(50)                    # Set frequency to 50 Hz

tmp = 0
step = 20
if step > servoMax - servoMin :
  step = servoMax - servoMin
while (True):
  turn = servoMin + (servoMax - servoMin) / step * tmp
  pwm.setPWM(0, 0, turn)
  time.sleep(1)
  pwm.setPWM(4, 0, turn)
  time.sleep(1)
  tmp += 1
  if (tmp >= step) :
    tmp = 0
```

ANHANG 3 QUELLCODE DES FINALEN PROJEKTES

MAINFRAME.PY

```
#!/usr/bin/python

import sys
import time
import math
import cv2.cv as cv
from optparse import OptionParser
import ServoManager

#go into standard position
ServoManager.moveTopToDefault()
ServoManager.moveBottomToDefault()

## better, but slow
#scale_factor=2, min_neighbors=3, flags=0

##faster
#scale_factor=1.2, min_neighbors=2, flags=CV_HAAR_DO_CANNY_PRUNING
#cv.SetCaptureProperty(capture, cv.CV_CAP_PROP_FRAME_HEIGHT, 240)
#cv.SetCaptureProperty(capture, cv.CV_CAP_PROP_FRAME_WIDTH, 320)

cascadePath = "haarcascade_frontalface_alt.xml"
min_size = (20, 20)
image_scale = 3
haar_scale = 1.2
min_neighbors = 2
haar_flags = cv.CV_HAAR_DO_CANNY_PRUNING
speedNormal = 10
speedFast = 20
cameraFPS = 27

distance = 0 # the distance of the last found face

parser = OptionParser()
parser.add_option("--width", type="int", dest="width", help="Set the width of the Camerainput",
default=320)
parser.add_option("--height", type="int", dest="height", help="Set the height of the Camerainput",
default=240)
parser.add_option("--show", action="store_true", dest="show", help="Set whether the frame should be
displayed, or not", default=False)

(options, args) = parser.parse_args()
width = options.width
height = options.height
showFrame = options.show

def detect_and_draw_and_move(img, cascade):
  global showFrame, width, height, speedNormal, speedFast, distance
   # allocate temporary images
  gray = cv.CreateImage((img.width,img.height), 8, 1)  #size, bit-depth, channels p. pixel
  small_img = cv.CreateImage((cv.Round(img.width / image_scale),
            cv.Round (img.height / image_scale)), 8, 1)

    # convert color input image to grayscal
  cv.CvtColor(img, gray, cv.CV_BGR2GRAY)

    # scale input image for faster processing
  cv.Resize(gray, small_img, cv.CV_INTER_LINEAR)
  cv.EqualizeHist(small_img, small_img)
```

```
  if(cascade):
    faces = cv.HaarDetectObjects(small_img, cascade, cv.CreateMemStorage(0),
                                 haar_scale, min_neighbors, haar_flags, min_size)

    if faces:
      facesFound = 0
      xMove = 0
      yMove = 0
      for ((x, y, w, h), n) in faces:
        # the input to cv.HaarDetectObjects was resized, so scale the
        # bounding box of each face and convert it to two CvPoints
        xS = int(x * image_scale)
        yS = int(y * image_scale)
        wS = int(w * image_scale)
        hS = int(h * image_scale)
        currentXMove = int(img.width / 2 - (xS + wS / 2))
        currentYMove = int(img.height / 2 - (yS + hS / 2))
        xMove += currentXMove
        yMove += currentYMove
        facesFound += 1
        pt1 = (xS, yS)
        pt2 = (xS + wS, yS + hS)
        distance = -0.0122 * wS + 2.2951 #depends on the dpi of the cam, assuming that width = height
        print (distance)
        #print "face found x = %d y = %d w = %d h = %d" % (xS, yS, wS, hS)
        #print "image dimensions: w = %d h = %d" % (img.height, img.width)
        #print "center of the face x = %d y = %d" % (xS + wS / 2, yS + hS / 2)
        #print "move x = %d and y = %d" % (currentXMove, currentYMove)

        if showFrame :
          cv.Rectangle(img, pt1, pt2, cv.RGB(255, 0, 0), 3, 8, 0)

      xMove /= facesFound
      yMove /= facesFound

      if (math.fabs(xMove) >= 0.06 * width) :
        if (math.fabs(xMove) >= 0.15 * width) :
          ServoManager.setSpeed(speedFast)
        else :
          ServoManager.setSpeed(speedNormal)
        if (xMove < 0) :
          ServoManager.moveRight()
        if (xMove > 0) :
          ServoManager.moveLeft()

      if (math.fabs(yMove) >= 0.08 * height) :
        if (math.fabs(yMove) >= 0.2 * height) :
          ServoManager.setSpeed(speedFast)
        else :
          ServoManager.setSpeed(speedNormal)
        if (yMove < 0) :
          ServoManager.moveDown()
        if (yMove > 0) :
          ServoManager.moveUp()
  if showFrame :
    cv.ShowImage("result", img)

if __name__ == '__main__':

  cascade = cv.Load(cascadePath)

  capture = cv.CreateCameraCapture(0)

  cv.SetCaptureProperty(capture, cv.CV_CAP_PROP_FRAME_HEIGHT, height)
  cv.SetCaptureProperty(capture, cv.CV_CAP_PROP_FRAME_WIDTH, width)

  if showFrame:
    cv.NamedWindow("result", 1)

  frame_copy = None
```

```
  while True:
    startTime = time.time()
    frame = cv.QueryFrame(capture)
    if not frame:
      cv.WaitKey(0)
      break

    if not frame_copy:
      frame_copy = cv.CreateImage((frame.width,frame.height),
                  cv.IPL_DEPTH_8U, frame.nChannels)

    if frame.origin == cv.IPL_ORIGIN_TL:
      cv.Copy(frame, frame_copy)
    else:
      cv.Flip(frame, frame_copy, 0)

    detect_and_draw_and_move(frame_copy, cascade)
    if cv.WaitKey(10) >= 0:
      break

    currentTime = time.time() – startTime
    deleteFrames = int(currentTime * cameraFPS)

    for i in range(0, deleteFrames):
      cv.QueryFrame(capture)

    if showFrame :
    cv.DestroyWindow("result")

# Getter & Setter
def getServoManager():
  global ServoManager
  return ServoManager

def getDistance():
  global distance
  return distance
```

SERVOMANAGER.PY

```
#!/usr/bin/python

from Adafruit_PWM_Servo_Driver import PWM
import time

#servo – settings
pwm = PWM(0x40, debug=False)
channelServoTop = 0
channelServoBottom = 4
servoMin = 220  # left stop
servoMax = 440  # right stop
speed = 10
currentPosTop = servoMin + (servoMax - servoMin) / 2
currentPosBottom = servoMin + (servoMax - servoMin) / 2
isTurning = False

pwm.setPWMFreq(50) # Set frequency to 50 Hz

def setServoPulse(channel, pulse):
  global pwm
  pulseLength = 1000000                   # 1,000,000 us per second
  pulseLength /= 50                       # 50 Hz
  pulseLength /= 4096                     # 12 bits of resolution
  pulse *= 1000
  pulse /= pulseLength
  pwm.setPWM(channel, 0, pulse)

def moveLeft():
  global currentPosTop, speed, servoMin, channelServoTop, isTurning
  isTurning = True
  if (currentPosTop - speed >= servoMin):
    currentPosTop -= speed
    pwm.setPWM(channelServoTop, 0, currentPosTop)
    #time.sleep(1)
    isTurning = False
    return True
  isTurning = False
  return False
```

```
def moveRight():
  global currentPosTop, speed, servoMax, channelServoTop, isTurning
  isTurning = True
  if (currentPosTop + speed <= servoMax):
    currentPosTop += speed
    pwm.setPWM(channelServoTop, 0, currentPosTop)
    #time.sleep(1)
    isTurning = False
    return True
  isTurning = False
  return False

def moveDown():
  global currentPosBottom, speed, servoMin, channelServoBottom, isTurning
  isTurning = True
  if (currentPosBottom - speed >= servoMin):
    currentPosBottom -= speed
    pwm.setPWM(channelServoBottom, 0, currentPosBottom)
    #time.sleep(1)
    isTurning = False
    return True
  isTurning = False
  return False

def moveUp():
  global currentPosBottom, speed, servoMax, channelServoBottom, isTurning
  isTurning = True
  if(currentPosBottom + speed <= servoMax):
    currentPosBottom += speed
    pwm.setPWM(channelServoBottom, 0, currentPosBottom)
    #time.sleep(1)
    isTurning = False
    return True
  isTurning = False
  return False

def moveBottomToDefault():
  global currentPosTop, servoMin, servoMax, channelServoTop, isTurning
  isTurning = True
  currentPosTop = servoMin + (servoMax - servoMin) / 2 - 5 #20 ... deviation
  pwm.setPWM(channelServoTop, 0, currentPosTop)
  #time.sleep(1)
  isTurning = False
```

```
def moveTopToDefault():
  global currentPosBottom, servoMin, servoMax, channelServoBottom, isTurning
  isTurning = True
  currentPosBottom = servoMin + (servoMax - servoMin) / 2 #50 ... deviation
  pwm.setPWM(channelServoBottom, 0, currentPosBottom)
  #time.sleep(1)
  isTurning = False

def setPWM(channel, value):
  global channelServoTop, channelServoBottom, currentPosTop, currentPosBottom, isTurning
  isTurning = True
  if (channel == channelServoTop) :
    currentPosTop = value
  elif (channel == channelServoBottom) :
    currentPosBottom = value
  pwm.setPWM(channel, 0, value)
  #time.sleep(1)
  isTurning = False

#Setter & Getter
def setSpeed(newSpeed):
  global speed
  speed = newSpeed

def getSpeed():
  global speed
  return speed

def getCurrentPosTop():
  global currentPosTop
  return currentPosTop

def getCurrentPosBottom():
  global currentPosBottom
  return currentPosBottom
```

ANHANG 4 PIN - BELEGUNG DES SERVOBOARDS

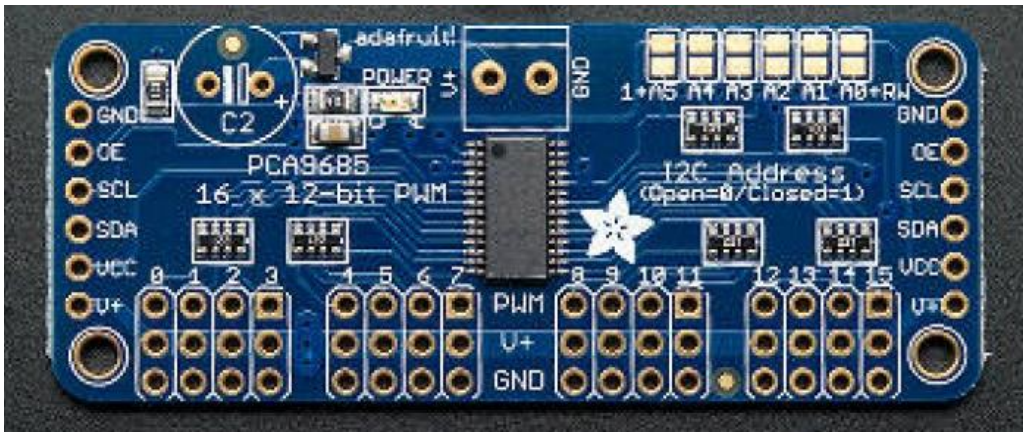

Abbildung 23 PIN - Belegung des Servoboards

Ausschnitt aus: http://www.ca.diigiit.com/image/cache/data/adafruit/16-channel-12-bit-pwm-i2c-interface/adafruit-16-channel-12-bit-i2c-interface-1-1000x750.jpg

ANHANG 5 GPIO – PIN – BELEGUNG DES RASPBERRY PI

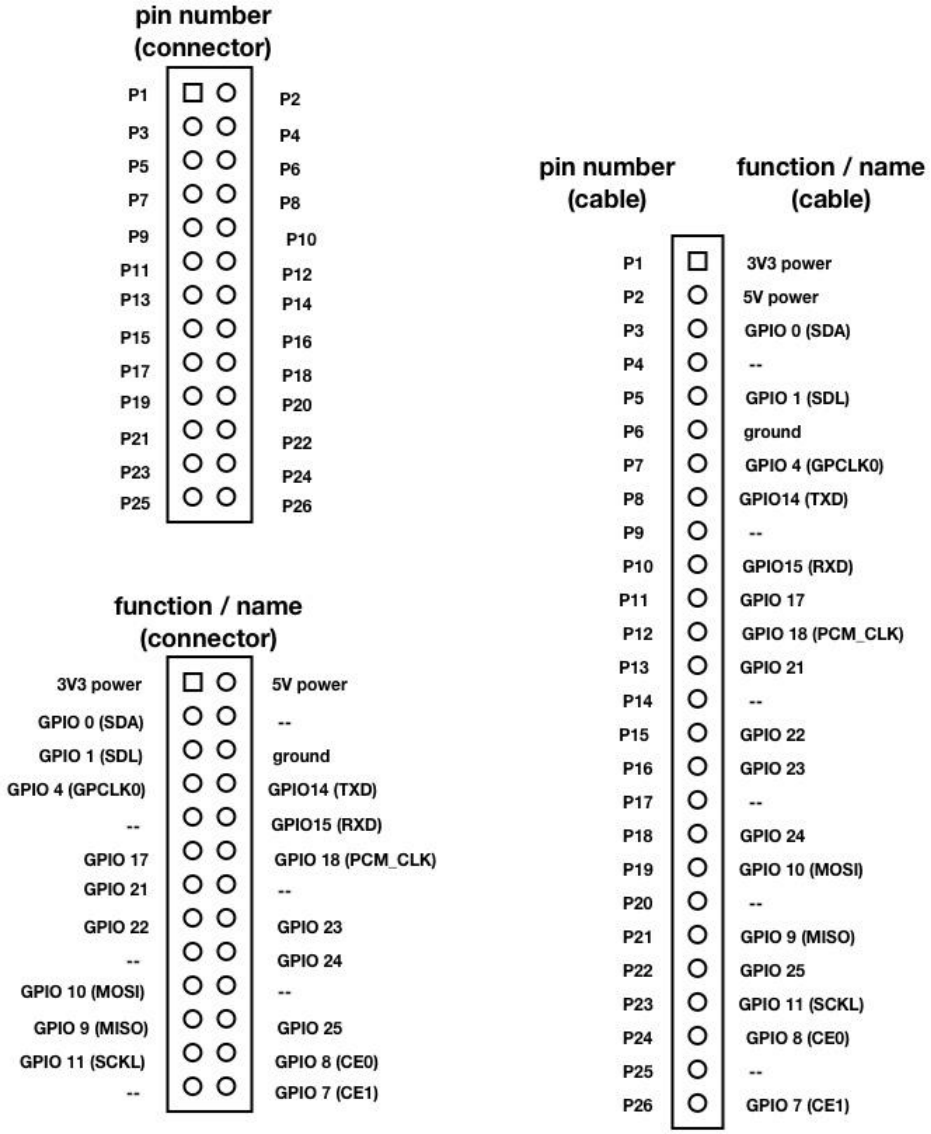

Abbildung 24 GPIO - PIN - Belegung des Raspberry Pis

Quelle: http://petrockblog.files.wordpress.com/2012/07/wpid-photo-01-07-2012-21283.jpg

ANHANG 6 VERSCHALTUNG DES SERVOBOARDS MIT DEM RASPBERRY PI

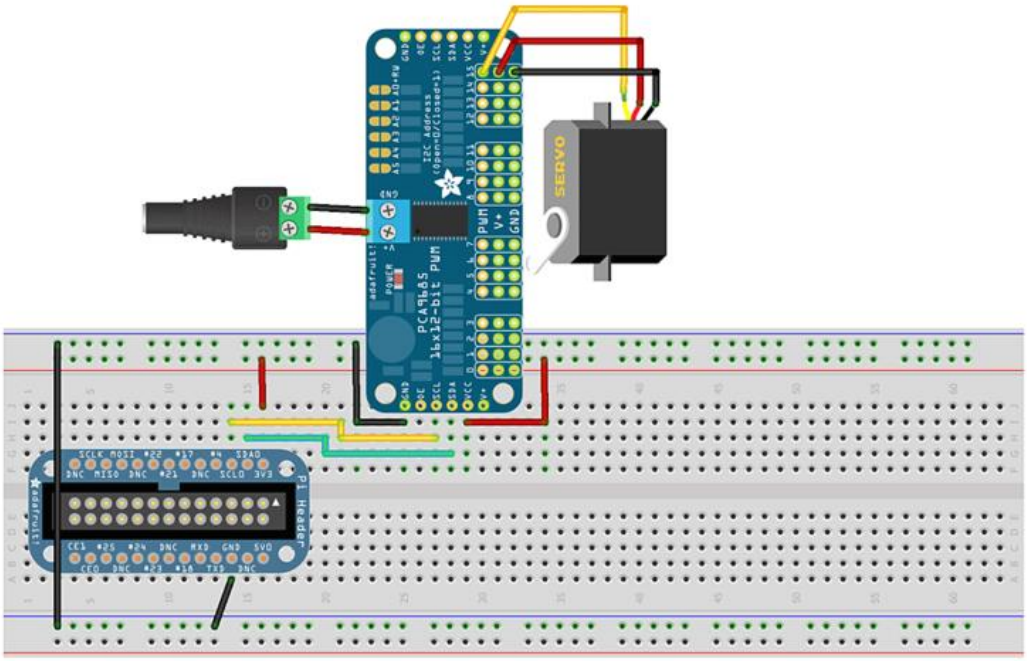

Abbildung 25 Verschaltung des Servoboards mit dem Raspberry Pi

Quelle: https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi/hooking-it-up

INHALTSVERZEICHNIS DER CD-ROM

- → seminar_work_deeps
  - Adafruit_I2C.pyc
  - Adafruit_PWM_Servo_Driver.py
  - Adafruit_PWM_Servo_Driver.pyc
  - haarcascade_frontalface_alt.xml
  - MainFrame.py
  - MainFrame_close_and_open.py
  - MoveServosToDefault.py
  - ServoManager.py
  - ServoManager.pyc
- → Seminarkurs
  - .settings
    - org.eclipse.jdt.core.prefs
  - lib
    - artoolkitplus.jar
    - artoolkitplus-windows-x86_64.jar
    - ffmpeg.jar
    - ffmpeg-windows-x86_64.jar
    - flycapture.jar
    - flycapture-windows-x86_64.jar
    - javacpp.jar
    - javacv.jar
    - jsoup-1.7.3.jar
    - libdc1394.jar
    - libfreenect.jar
    - libfreenect-windows-x86_64.jar
    - opencv.jar
    - opencv-windows-x86_64.jar
    - swingx.jar
    - videoinput.jar
    - videoinput-windows-x86_64.jar

- src/com/deeps/
  - opencvhaarcascadestraining
    - Engine.java
    - MainFrame.java
    - PanelInterface.java
  - opencvtesting
    - Engine.java
    - MainFrame.java
  - seminarworkdemo
    - Engine.java
    - MainFrame.java
    - PanelInterface.java
- .classpath
- .project
- haarcascade_frontalface_default.xml

EIDESSTATTLICHE SELBSTSTÄNDIGKEITSERKLÄRUNG

Ich versichere, dass ich die vorliegende Seminararbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Texten entnommen sind, wurden unter Angabe der Quellen (einschließlich des World Wide Web und anderer elektronischer Text- und Datensammlungen) und nach den üblichen Regeln des wissenschaftlichen Zitierens nachgewiesen. Dies gilt auch für Zeichnungen, bildliche Darstellungen, Skizzen, Tabellen und dergleichen. Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschungsversuch behandelt werden und dass bei einem Täuschungsverdacht sämtliche Verfahren der Plagiatserkennung angewandt werden können.

31.10.2014
Ort, Datum

Unterschrift