SHKOLLA AUSTRIAKE

Diplomarbeit

Gesichtserkennung - Rei

Autor: Rei Hoxha

Inhaltsverzeichnis

Kapitel 1

Gesichtserkennung - Rei

Am Beginn wird eine allgemeinere Übersicht dieses Diplomarbeitskapitels gemacht bzw. dargestellt. Damit wird es verstanden, worum es in diesem Teil geht.

1.1 Allgemeines

Die Gesichtserkennung ist der Teil des Systems, mit dem ich mich beschäftige. Es wird durch einen Vergleich überprüft, ob die Person vorher schon registriert worden ist oder nicht, und ob sein Gesichtsdaten schon am Server existieren oder nicht. Nur wenn die Vergleichsergebnisse positiv sind, wird die Person erkannt. Der Person werden die Ergebnisse durch Anzeiger kommuniziert. Dieses Teil des Projektes erfordert eine Arbeit mit Datenbanken, Gesichtsvergleichsalgorithmen und mit vielen System Tests. Teil meiner Aufgabe ist auch der Aufbau des Systems und alles was mit Hardware zu tun hat. Alle Hardware Komponenten werden in den folgenden Kapiteln klar, verständlich und deutlich erklärt. Eine grobe Skizze des Systems ist in Abb.1.1 zu sehen.

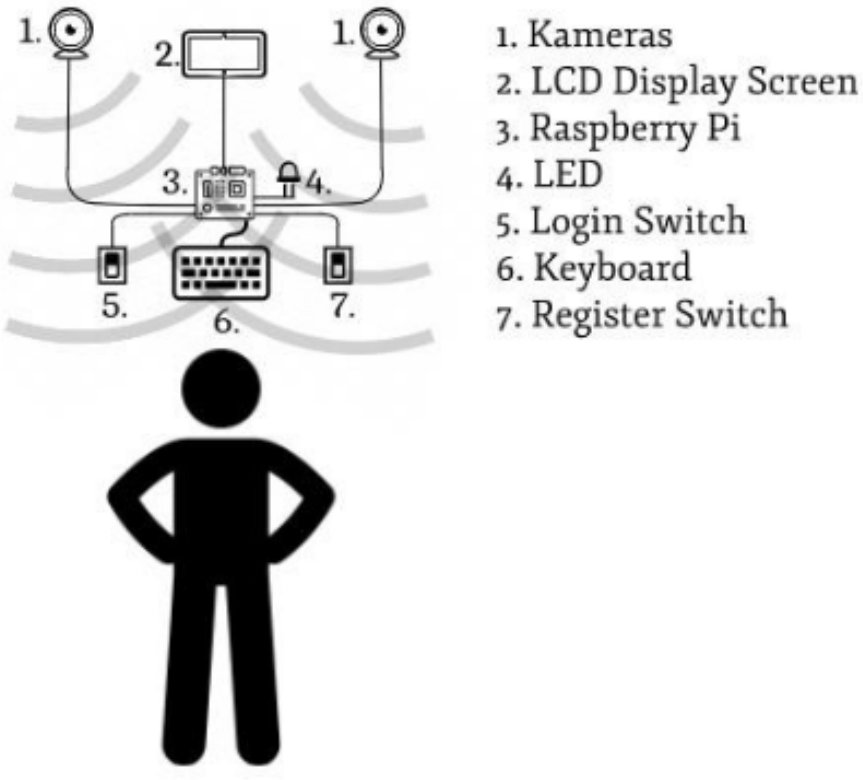

Abbildung 1.1: Grobe Skizze des Systems

1.2 Technische Lösung

In diesem Abschnitt wird eine feinere und detaillierte Übersicht im Bezug auf die technischen Lösung gegeben. Alles was mit Technik zu tun hat, wird hier erklärt.

1.2.1 Hardware und Aufbau

Zuerst wird der Aufbau des Systems beschrieben, zusammen mit allen Komponenten, die zu verwenden sind. Ohne die Hardware würde nichts funktionieren, weil die Software ohne die Hardware nicht funktionieren kann.

Bauteile und HW-Komponenten

Die Bauteile, die für dieses System verwendet worden sind, sind die folgende:

1. *Steckboard bzw. Steckplatine*
   Die Steckplatine ist eine Komponente der Hardware des Systems. Sie wird für die Entstehung der elektrischen Verbindung von verschiedenen elektrischen Bauteile benötigt, um elektrische Schaltungen zu bauen oder um verschiedene Tests und

Experimenten zu machen. In dieser Steckplatine werden alle anderen Komponenten platziert, damit die Verbindung erstellt werden kann und damit das System laufen kann.

2. *Kabel bzw. Leiter*
   Damit die verschiedenen Komponenten, die in der Steckplatine platziert sind, miteinander verbunden werden können, braucht man unbedingt Kabel. Mithilfe von Kabeln können elektrische Impulse und Signale fließen, damit die Energie und die Information übertragen werden. Die verwendeten Kabel sind aus Kupfer und vom Typ Male-Male als auch vom Typ Male-Female. Die Kabeln vom Typ Male-Female werden verwendet, um die Verbindungen zwischen die Elemente in der Steckplatine und den Raspberry Pi zu ermöglichen. Auf der anderen Seite werden die Male-Male Kabeln verwendet um die Verbindungen innerhalb der Steckplatine zu ermöglichen.

3. *LEDs*[1]
   LEDs sind elektronische Halbleiter Elemente, die Licht produzieren können, wenn sie Spannung kriegen. Ein LED besteht aus zwei Beinen. Das längere Bein ist die Anode, die den Pluspol symbolisiert. Das andere Bein ist die Kathode, und symbolisiert den Minuspol. Durch die Beine wird der Kontakt mit der Steckplatine hergestellt.

4. *Widerstand*
   Ein Widerstand ist ein elektrisches Bauteil, das zur Reduzierung von Strom verwendet wird, damit ein Gleichgewicht zwischen Strom und Spannung gesichert werden kann. Die Einheit ist Ohm.

5. *Taster*
   Ein Taster wird wie ein Schalter gedrückt, mit dem Zweck Impulse oder Signale zu schicken. Im Gegenteil zu einem Schalter wird der Taster nach der Betätigung wieder in der Basiszustand zurückgestellt. Ein Plusleiter, Minusleiter und ein Datenleiter sind bei einem Taster vorhanden.[4]

6. *Raspberry Pi*
   Raspberry Pi ist ein Minicomputer, der in diesem Projekt den normalen Computer ersetzt. Der verwendete Raspberry, Version 3, hat 4 USB-Anschlüsse, einem Netzteil, eine SD-Karte, 16 GPIO[2] Pins und einem VGA Schnittstelle. Die 3.-Bit SD-Karte ist ein wichtiges Element, weil dort alle Daten und Informationen gespeichert sind.

7. *Bildschirm*
   Ein Bildschirm ist eine Anzeige, die für die visuelle Darstellung von verschiedenen Informationen oder Daten(wie Videos, Fotos, Statistiken usw.) verwendet wird. Ein Bildschirm wird zu den heutigen Zeiten sehr häufig verwendet, aufgrund der hohen Benutzerfreundlichkeit, die angeboten wird.

[1] Light Emitting Diode
[2] Generated Input Output

8. *Tastatur*
   Eine Tastatur ist ein Input Gerät, dass durch das Drücken von Tastern den Benutzer die Eingabe von Daten oder Befehle ermöglicht.

9. *Kamera*
   Es werden 2 Kameras benötigt, die die Fotos der Gesichter der Personen machen. Sie werden auch im Raspberry integriert bzw. mit dem Raspberry verbunden. Die Kameras sind vom Typ Aukey.

Schaltplan und Erklärung des Aufbaus

Die Hardware Komponenten werden in einer Steckplatine platziert. Dort werden die Verbindungen mit den anderen Komponenten sowie mit dem Raspberry Pi hergestellt. Die elektrische Schaltung wird durch einen Schaltplan beschrieben. Dieser Schaltplan wurde mit Hilfe eines Programms, "Fritzing", erstellt und spielt eine sehr wichtige Rolle bei der Organisation und Planung des Schaltkreises. Der Schaltplan ist auf Abb.1.2 zu sehen.

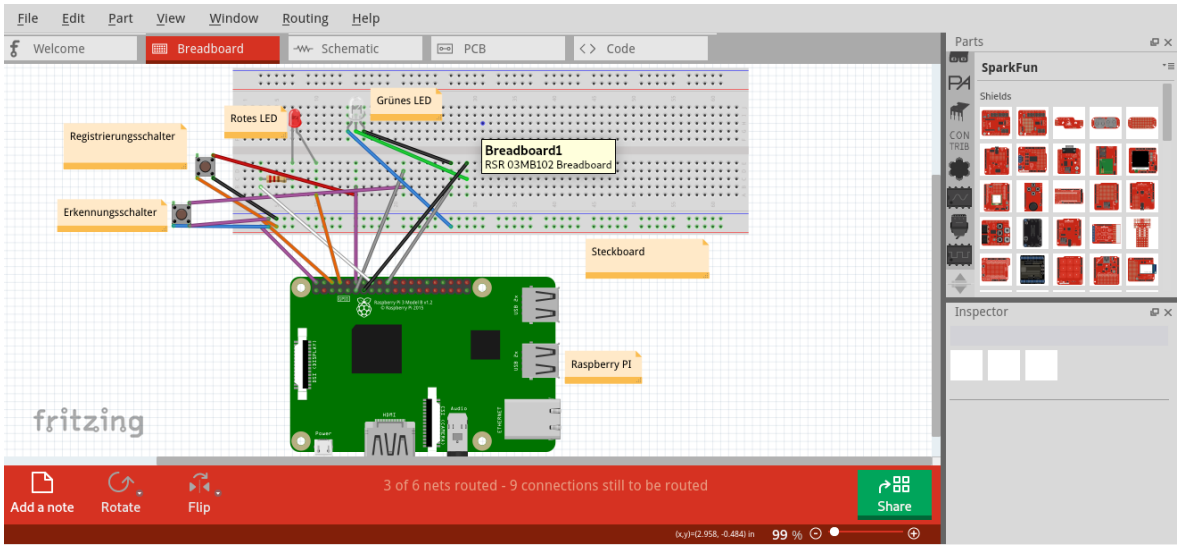

Abbildung 1.2: Schaltplan des Systems

Wie der Schaltplan zeigt, besteht das System aus zwei LEDs, zwei Tastern, einem Widerstand, eine Steckplatine und einem Raspberry Pi. Sehr wichtig für den Aufbau der Schaltung sind die GPIO Pins. Diese Pins sind in dem Raspberry Pi platziert und können als Input, als Output oder Spannung Pins verwendet werden. Der erste Taster dient für die Gesichtsregistrierung und besteht aus einem Pluspol, einem Datenleiter und aus dem Minuspol. Der Pluspol Anschluss wird mittels Steckplatine mit dem 5-Volt Pin des Raspberry Pi verbunden, während der Minuspol Anschluss mit dem Minuspol der Steckplatine verbunden wird. Der Datenanschluss ist mit dem GPIO18 Pin verbunden. Wie bei dem Registrierungs-Taster wird auch bei dem Erkennungstaster[3] der Minuspol Anschluss mit einem Pin des Minusbereichs der Steckplatine verbunden. Der Pluspol Anschluss gehört zu dem 5-Volt Pin des Raspberry PI, während der

[3]Auch als Login Taster genannt

Datenanschluss mit dem GPIO17 Pin verbunden ist. Die rote LED wird verwendet wenn die Registrierung oder Erkennung der Benutzer im System nicht erfolgreich war. Das längere Bein(die Anode) wird mit dem GPIO23 Pin des Raspberry verbunden, während die Kathode mit dem Minuspol Bereich der Steckplatine verbunden wird. Die grüne LED ist eine RGB(Red Green Blue) LED. Diese LED kann die Farbe ändern, und wird im Fall einer erfolgreichen Registrierung oder Erkennung des Benutzers im System verwendet. Sie hat im Gegensatz zu der normalen LED 3 Anschlüsse. Der Minus Anschluss wird in dem Minuspol Bereich der Steckplatine eingeschlossen und der Pluspol Anschluss wird mit dem Pluspol Bereich der Steckplatine verbunden. Mit dem GPIO27 Pin des Raspberry wird der Datenanschluss verbunden. Die GPIO Pins sind extrem wichtig für die Integration der Tastern, LEDs und der anderen Bauteile in dem technischen und logischen Teil bzw. in der Software und in den verwendeten Skripten.

1.2.2 Software

Nachdem der Aufbau und die verwendete Hardware des Systems beschrieben wurden, wird nun die Software beschrieben. In diesem Unterkapitel wird alles was mit dem logische Teil der Umsetzung zu tun hat besprochen: das verwendete Betriebssysteme, Programmiersprachen, Frameworks, Technologien und Planungsmethoden. Es wird jede Aufgabe zusammen mit der zugehörigen Lösung im Detail beschrieben und jedes programmiertes Skript erklärt.

Verwendete Technologien

In diesem Teil des Projekts, das Gesichtserkennung heißt, werden für die Umsetzung 2 verschiedene Technologien verwendet, die in den anderen Teilen nicht oder anders verwendet sind.

- Open CV
  Das wichtigste Framework für dieses Projekt ist OpenCV. OpenCV ist eine Softwarebibliothek, die für Computer-Vision und maschinelles Lernen verwendet wird. Die Bibliothek verfügt über mehr als 2500 optimierte Algorithmen, die sowohl klassische als auch moderne Computer Vision- und maschinelle Lernalgorithmen umfassen. Diese Algorithmen können verwendet werden, um Gesichter zu registrieren und zu erkennen, um Objekte zu identifizieren, menschliche Handlungen in Videos zu klassifizieren und Kamerabewegungen zu verfolgen. Dieses Framework wurde deswegen gewählt, weil die Vielfältigkeit der angebotenen Optionen und Paketen einfach größer ist als bei anderen Frameworks. Ein anderer Vorteil ist das OpenCV Open-Source ist. Die größte Herausforderung ist die lange und komplizierte Installation auf Linux. [3]

- Fritzing
  Fritzing ist ein Programm das für die graphische Darstellung des Schaltkreises des Systems verwendet wurde. Durch dieses Programm wurde auch der Schaltplan des Systems erstellt(Abb.1.2). Fritzing bietet eine sehr große Menge von elektronischen Komponenten und ein PCB, eine Steckplatine und einen Schematic View.

In diesem Fall wurde die Breadboard View gewählt, weil dort alles übersichtlicher ist. Was noch gut ist, ist das Fritzing kostenlos in Linux angeboten wird, was für Windows nicht der Fall ist.[1]

Lösungsweg- Beschreibung und Erklärung

Der Lösungsweg für die Umsetzung der Aufgaben der Gesichtserkennung ist streng mit der vorherigen Planung verbunden. Deshalb wurde nicht nur das Big Picture(Großes Sicht des Systems nach Außen), die in die vorherigen Kapiteln beschrieben wurde, erklärt, sondern auch die Erste Ebene des Gesichtserkennungsteils.

Die erste Ebene sieht man unter Abb.1.3. Sie ist ein detaillierte Version des Big Picture, das sich nur auf den Erkennungsteil konzentriert. Man spricht von einer Iteration, die hier passiert ist. Folgend wird die Vorgehensweise und der Lösungsweg dieser Aufgabe mit Hilfe der 1.Ebene erklären und beschrieben.

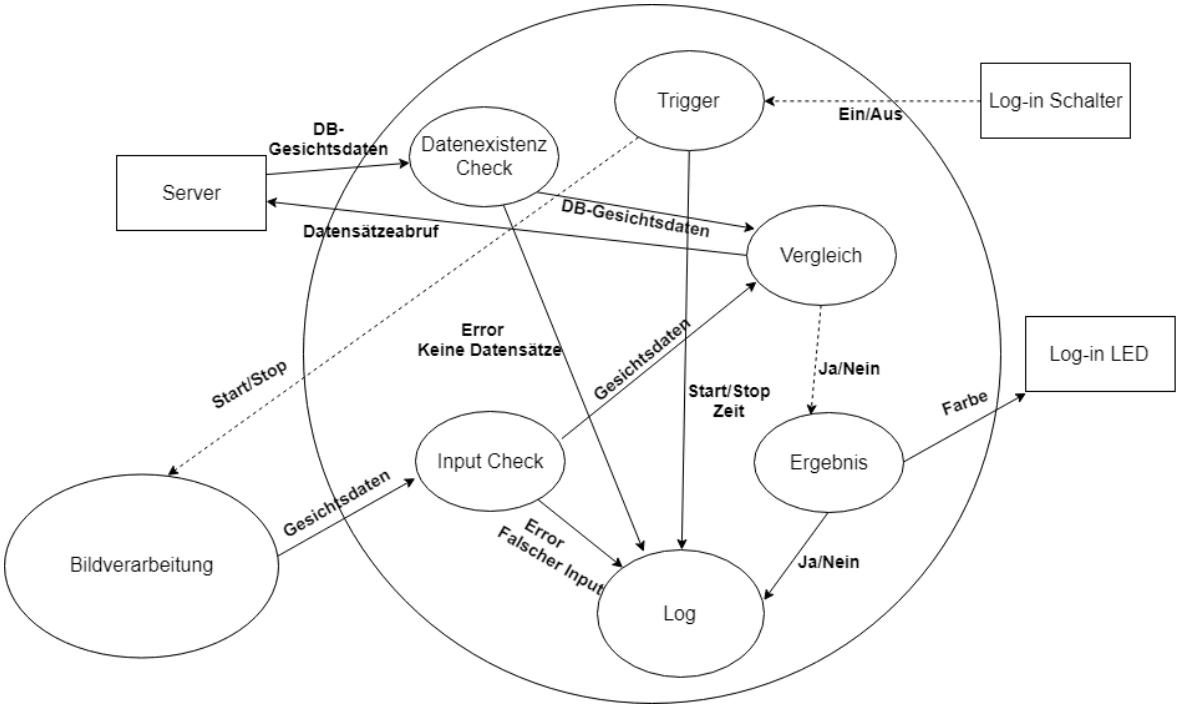

Abbildung 1.3: Erste Ebene

Die wichtigste Elemente der ersten Ebene sind auf der Abbildung 1.4 erklärt:

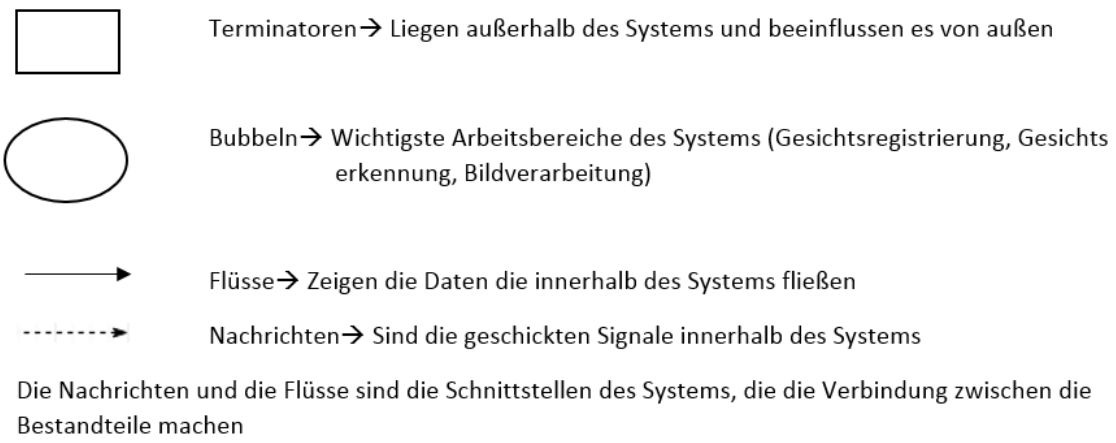

Abbildung 1.4: Erste Ebene Erklärung

Einzelne Arbeitsschritte des Erkennungsprozesses:

In diesem Unterkapitel wird der Vergleich der Gesichter der Personen detailliert erklärt. Es wurden die einzelne Schritte der Arbeit sowie die verwendeten Methoden beschrieben.

1. *Erkennungstaster wird gedrückt*

   Will der Benutzer sich im System einloggen, muss er zuerst den Erkennungstaster drücken. Wird dieser Taster gedrückt, dann kriegt der Benutzer eine Anmeldung, bei der er seine Emailadresse eingeben muss. Um diese Aufgabe zu erledigen, muss das Paket RPi.GPIO importiert werden. Auf diese Weise ist der Zugriff auf den GPIO Pins möglich, damit der Taster Zustand, oder der LED Zustand erkannt werden. Das Paket mysqldb wird auch benötigt, um Zugriff auf die Datenbank zu haben.

   ```
   if GPIO.input(17):
   exec(open('Existiert_nExistiert.py').read())
   ```

2. *Benutzer gibt seine Emailadresse ein*

   Es ist wichtig zu erwähnen, es dass für die Umsetzung dieses Systems notwendig ist, die verschiedenen Skripten miteinander zu verbinden. Deshalb müssen sie innerhalb anderer Skripten angerufen werden. Das wird durch Variablen gemacht. Deshalb ist das Importieren des Pakets sys nötig. Der Benutzer gibt seine Emailadresse an, die danach verwendet wird, um in der Datenbank schneller auf die Benutzerbilddaten zuzugreifen. Es wurde deshalb die Emailadresse(und nicht Vorname, Nachname usw.) gewählt, weil diese Adresse immer eindeutig ist, das

heißt, es ist unmöglich, dass zwei Benutzern dieselbe Adresse haben, und dasselbe kann für die andere Benutzerdaten(Vorname, Nachname usw.) nicht gesagt werden.

3. *Die Existenz der eingegebenen Emailadresse des Benutzers in der Datenbank wird geprüft*

```
for x in myresult:
if x[2]==rei.emailiperkrahasim:
bool=True
else:
bool=False
os.system('./Log_Erkennung_Email_Nicht_Gefunden.py')
```

Wie es vorher erwähnt wurde, ist dieser Schritt deswegen wichtig, weil falls die eingegebene Email nicht in der Datenbank existiert, muss das System den Bildvergleich nicht machen. Der Benutzer kriegt stattdessen eine Anmeldung, die ihm sagt, dass er noch einmal versuchen kann seine Emailadresse einzugeben. Wird diese neue Adresse auch nicht in der Datenbank gefunden, kann danach dieser Benutzer mit dieser Adresse sich nicht mehr versuchen in System anzumelden.

4. *Bild wird gemacht und temporär gespeichert*

Gleich nachdem der Benutzer sein Email eingegeben hat, macht die Kamera das Bild. Dieses Bild wird temporär gespeichert, weil es nach dem Vergleich mit dem Bild, das schon in der Datenbank liegt, nicht mehr benötigt wird. Sonst würde es extrem viele Bilddaten in der Datenbank geben, die nur einmal verwendet werden. Stattdessen wird nur der Pfad des Bildes zusammen mit den extrahierten Punkten in der Datenbank gespeichert.

5. *Gemachtes Bild wird zugeschnitten*

Das von der Kamera gemachtes Bild ist noch nicht bereit zum Vergleich. Um die Punkte richtige zu extrahieren, muss das Bild zuerst zugeschnitten werden. Das heißt, die Dimensionen des Bildes werden reduziert, und somit werden die andere Objekten auf dem Bild nicht berücksichtigt(Sie würden die Performance der Vergleichsprozess negativ beeinflussen). Das Zuschneiden des Bildes wurde mit Hilfe eines Programms, das Teil der Arbeit der Bildverarbeitungsgruppe ist, gemacht. Wird das Bild zugeschnitten, wird es automatisch auch umbenannt. Das Bild nimmt das Schlüsselwort „New“+die Emailadresse der Person als Name. Der Datentyp ist .jpg .

6. *Die Gesichtspunkte werden extrahiert*

Das zugeschnittene Bild des Gesichts der Person besteht aus verschiedene Punkte. Insgesamt sind es 68 Gesichtspunkte. Alle diese Punkte müssen für den Vergleichsprozess extrahiert werden. Das heißt, jeder Punkt muss ein X und ein Y Wert

haben. Diese Koordinaten müssen ermittelt werden. Dies ist auch eine Aufgabe der Bildverarbeitungsbranche, und ist extrem wesentlich für das Funktionieren des Erkennungsprozesses.

7. *Wird dieselbe Emailadresse in der Datenbank gefunden, werden die zugehörige Bildinformationen geladen*

   Falls aber die vom Benutzer eingegebene Emailadresse schon in der Datenbank liegt, weißt das System Bescheid, dass ein Bilderdatenvergleich stattfinden muss. Um das zu erreichen, werden alle Bildinformationen geladen. Unter Bildinformationen sind die 68 extrahierten Gesichtspunkte zu verstehen. Der kurze Codeabschnitt unten zeigt genau wie dies funktioniert:

```
if b=="existiert":
mycursor.execute(
"""select * from info i \
join person p \
on i.idP=p.idP \
where p.email='%s';"""%var1)
myresult=mycursor.fetchall()
for x in myresult:
print(x)
```

   Das heißt, die extrahierte Punkte des von der Kamera gemachten Bildes und die extrahierte Punkte, die in der Datenbank gespeichert sind und die den Person mit derselbe Emailadresse wie die vom Benutzer eingegebene Adresse gehören , stehen jetzt endlich zur Verfügung. Ab hier wird angenommen, dass die eingegebene Emailadresse mit mindestens einer in der Datenbank vorhandenen Emailadresse gleich ist, damit der Vergleichsprozess erklärt werden kann.

8. *Die extrahierten Punkte werden in 5 verschiedene Bereiche geteilt*

   Bevor der Vergleichsprozess beginnt, müssen die 68 extrahierten Punkte geteilt werden. Die 5 großen Bereiche sind das Gesicht, die Nase, das rechte Auge, das linke Auge und der Mund. Jeder Bereich hat seine eigenen Punkte. Die erste 28 Punkte gehören dem Gesicht, die nächste 9 gehören der Nase, die nächste 6 dem rechten Auge, die nächste 6 dem linken Auge und die letzte 19 gehören dem Mund. Durch dieser Prozess wird die Vergleichsqualität stark erhöht, weil die Punkte der Nase werden genau mit der Punkte der anderen Nase verglichen, und nicht mit dem von dem Augen beispielsweise.
   Der untere Codeabschnitt zeigt wie dieser Schritt für die ersten 27 Punkte programmiert wird(nur um eine Idee zu haben, wie es funktioniert):

```
for pika in range(0,27):
        dis=math.sqrt(abs(( (vleratx.item(pika)-float(res[vx])))) *
        abs( (vleratx.item(pika)-float(res[vx])) ) + (
```

```
abs((vleraty.item(pika)-float(res[vy]))) *
abs((vleraty.item(pika)-float(res[vy])) ) ))
```

9. *Die Bildinformationen der beiden Bilder werden verglichen*

Die wichtigste Aufgabe ist der Vergleich der Bildinformationen. Es werden die extrahierten Gesichtspunkte der bereits gemachten Bilder mit der Gesichtspunkte in der Datenbank verglichen. Selbstverständlich müssen wie es vorher erwähnt wurde die beiden Personen, deren Gesichtspunkte verglichen werden dieselbe Emailadresse haben. Das wichtigste Paket, Open CV, wird für diesen Vergleich benötigt. Nachdem dieses Paket importiert wurde, kann der Vergleich beginnen. Es gibt verschiedene Schritte, die erfolgen, und die nun erklärt werden:

- Startpunkt bzw. Origin wird festgelegt

  In jeden Gesichtsbild wird ein Startpunkt bzw. eine Origin für den Vergleich festgelegt. Dieser Punkt ist der Punkt mit der Koordinaten (0/0). Es wird der Abstand von jedem anderen Gesichtspunkt zu dieser Origin berechnet. Dieser Abstand wird dann mit dem entsprechenden Abstand des zweiten Bildes verglichen. Diese Abstände müssen miteinander übereinstimmen(zum Beispiel ein Punkt vom rechten Augen im ersten Bild muss mit dem zugehörigen Punkt im zweiten Bild verglichen werden, und nicht mit einem Punkt von dem Augen in ersten Bild).

- Die Abstände jedes anderen Punktes des gemachten Bildes vom Startpunkt werden berechnet.
  Nachdem der Startpunkt festgelegt wurde, beginnt der Vergleich der Abstände des gemachten Bildes(Punkt-Startpunkt) und der Abstände, die in der Datenbank gespeichert sind.

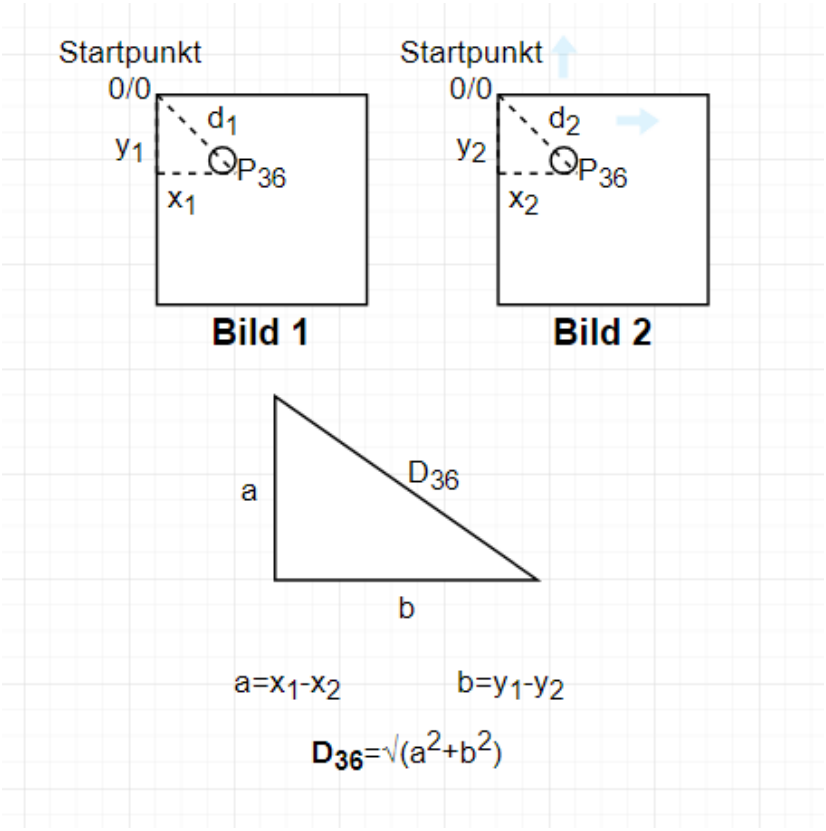

Abbildung 1.5: Vergleichsmethode zweier Punkte

Auf Abbildung 1.5 wird die verwendete Vergleichsmethode dargestellt. Der Startpunkt hat die Koordinaten (0/0) und befindet sich links oben. Jeder Punkt hat seine bestimmten Koordinaten, zum Beispiel, der Punkt 36 des ersten Bildes hat die Koordinaten x1 und y1, und den 36.Punkt des anderen Bildes hat die Koordinaten x2 und y2. Um den Abstand des Punkt 36 zu berechnen, wird die pythagoreische Formel verwendet. Die Differenz der x-Koordinaten und die Differenz der beiden y-Koordinaten sind die beide Katheten(a und b auf der Abbildung). Um die Hypotenuse zu finden wird das Quadrat beider Katheten addiert, und am Ende wird die Wurzel des Ergebnisses berechnet. Das Ergebnis ist der Abstand des Punktes 36 vom Startpunkt. Dieselbe Methode wird für jeder Punkt verwendet. Das obige Beispiel vergleicht die Abstände der Punkte von 2 verschiedenen Bildern. Das Vergleichsprinzip ist dasselbe wie bei dem Vergleich der Punkteabstände des gemachten Bildes und der in der Datenbank gespeicherten Punkteabstände. Es ist einfach leichter erklärbar und verstehbar, wenn 2 Bildern vergleicht wurden. Die Abstände aller Punkt aller Bereiche wurde in einer Variable gespeichert.

- Es wird der maximale Abstand für jeden einzelnen Bereich berechnet

Nachdem die oben erklärten Abstände kalkuliert wurden, wird der maximale Abstand für jeden Bereich(Gesicht, Nase, linke Auge, rechte Auge, Mund)

benötigt. Diese Maximumwerte sind wesentlich für den Vergleich. Warum das so ist wird später erklärt.
Der untere Codeabschnitt zeigt wie dieser Schritt für die ersten 27 Punkte implementiert ist:

```
MAX = [0,0,0,0,0]
for pika in range(0,27):
dis=math.sqrt(abs(( (vleratx.item(pika)-float(res[vx])))) *
abs((vleratx.item(pika)-float(res[vx])) ) + (
abs((vleraty.item(pika)-float(res[vy]))) *
abs((vleraty.item(pika)-float(res[vy])) ) ))
if(dis < min):
        min = dis
if(dis > max):
        max = dis
global MAX
MAX[0]=max
print("Max:\%s\n"\%(MAX[0]))
min=10.0
max=0.0
dis=0
```

Zuerst wurde eine Array MAX erstellt, die 5 Elemente enthält. Jedes Element gehört zu einem der 5 Bereiche. Beispielsweise der Maximumwert an der Stelle 0(erste Element) ist der Maximumabstand, der bei dem Gesichtsbereich gefunden wurde. Die Funktion Max in Python, die den größte Wert automatisch findet, wurde verwendet. Jeder Abstand wird mit dem Maximumwert verglichen(Maxwert beginnt am Anfang bei 0). Ist er größer, dann wird dieser Abstand der neue Maximumwert. So wird der Maximumwert ermittelt.

- Es wird das geometrische Mittel der Abstände aller Bereiche berechnet

"Das geometrische Mittel der Abstände aller Punkte (nicht in einzelnen Bereichen) ist auch wichtig für den Vergleich. Es gibt 68 Gesichtspunkten insgesamt, das heißt, 68 Abstände von dem Startpunkt. Um das geometrische Mittel von n Zahlen x1,x2,...xn zu ermitteln, muss man deren Produkt bilden und von diesem die n-te Wurzel ziehen."[2]

Auf Abbildung 1.6 wird die Formel, die sich ergibt gezeigt:

$$G(x_1, x_2, \ldots, x_n) = \overline{x}_{geom} = \sqrt[n]{x_1 \cdot x_2 \cdots x_n}$$

Abbildung 1.6: geometrisches Mittel

Der untere Codeabschnitt zeigt wie dieser Schritt für die ersten 27 Punkte

implementiert ist(nur um eine Idee zu haben, wie es funktioniert):

```
for pika in range(0,27):
dis=math.sqrt(abs(( (vleratx.item(pika)-float(res[vx])))) * abs(
(vleratx.item(pika)-float(res[vx])) ) + (
abs((vleraty.item(pika)-float(res[vy]))) *
abs((vleraty.item(pika)-float(res[vy])) ) ))
gm *= dis
vx+=2
vy+=2
gm=gm**(1/27)
print("GM:%s\n"%(gm))
gm=1
dis=0
```

- Die ermittelte Maximumwerte und die geometrischen Mittel der Abstände der Punkte des gemachten Bildes werden mit der Maximumwerten und den geometrische Mitteln der Abstände der, in der Datenbank, gespeicherten Punkte verglichen

  (a) Beschreibung

  Wenn die angesprochene Maximumwerte und die geometrische Mitteln der Abstände der Punkte des gemachten Bildes und die geometrische Mitteln Abstände des in der Datenbank gespeicherten Punkte gleich oder sehr ähnlich sind, dann wird von der selben Person gesprochen und die Erkennung ist erfolgreich. Deshalb werden einige Abgrenzungen bestimmt, wann die Erkennung als erfolgreich gemacht wird und wann nicht.

  (b) Mögliche Fälle

  Insgesamt gibt es 5 Bedingungen bzw. 5 Fälle, die eintreten können und die wichtig sind. Diese Bedingungen wurden mit Hilfe des vielen gemachten Tests, darüber später gesprochen wird, bestimmt. Ist einer dieser fünf Bedingungen wahr, ist die Erkennung nicht erfolgreich und es wird nicht von derselben Person gesprochen. Diese Fälle werden unten mit Hilfe von Bildern erklärt. Es werden die Maximumwerten jeder Bereiche(Gesicht, Nase, linke Auge, rechte Auge, Mund, also 5 insgesamt), sowie das geometrische Mittel aller 68 Punkten berücksichtigt.

  i. 1.Fall: Mindestens 2 Maximumwerte sind größer als 0.06

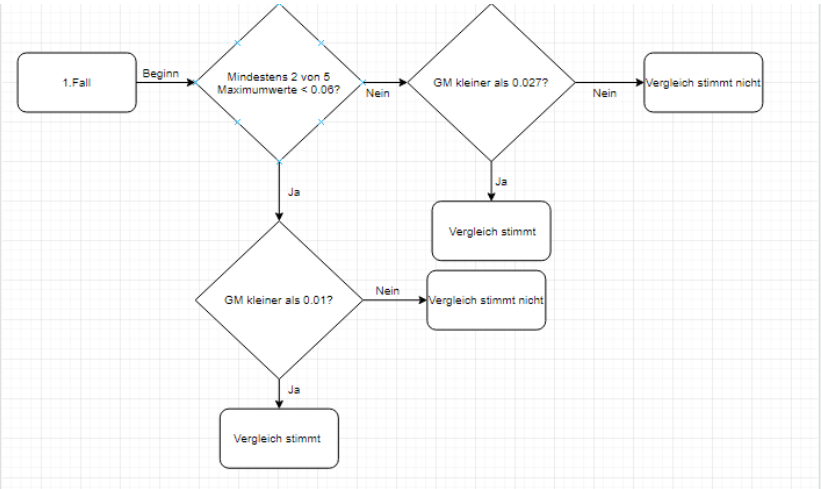

Abbildung 1.7: 1_Fall

Sind 2 oder mehrere Maximumwerte größer als 0.06, wird das geometrische Mittel betrachtet. Ist der Wert größer als 0.01, gibt es keine Übereinstimmung. Wenn weniger als 2 Werte größer als 0.06 sind, muss auch das geometrische Mittel kleiner als 0.027 sein, um eine erfolgreiche Erkennung zu haben.

Auf Abbildung 1.7 ist ein Flussdiagramm dargestellt, das die Abfolge dieser Bedingung beschreibt.

Unten ist der Codeabschnitt, der zeigt wie das funktioniert, ersichtlich:

```
global count
count = 0
for i in range(0,len(MAX)):
        if MAX[i] > 0.06:
                count+=1
                status1=True
                status11=True
                status2=True
                status22=True

#Wenn JA: Nur wenn gmT kleiner als 0.01 sind sie richtig,
 sonst FALSCH
        if count ==1:
                global status11
                status11=False
        if count >= 2:
                global status22
        status22=False
        if gmT > 0.01:
```

```
                global status1
                status1 = False

#Wenn NEIN: gmT muss kleiner als 0.027 sein,
                                um RICHTIG zu sein
                else:
                        if gmT > 0.027:
                                global status2
                                status2 = False
```

ii. 2.Fall: Genau einen Maximumwert größer als 0.07 ist und kleiner als 0.084

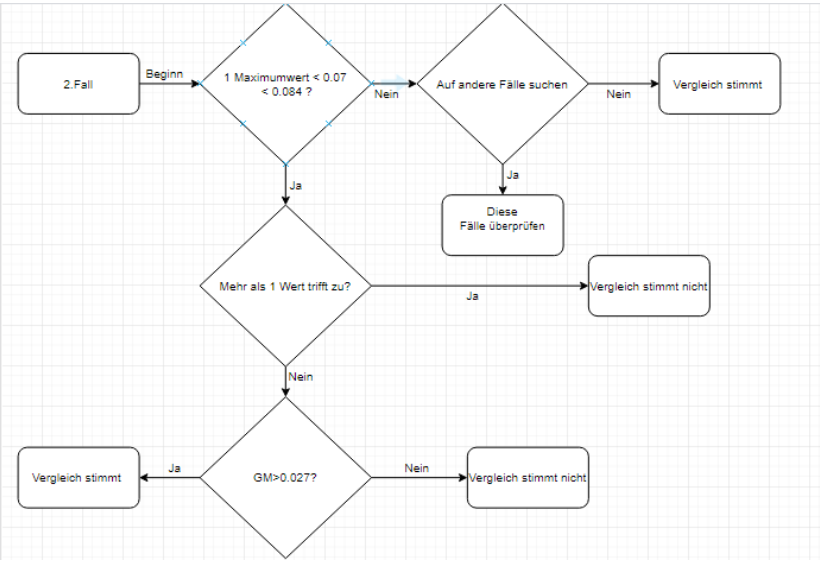

Abbildung 1.8: 2_Fall

Stimmt dieser Bedienung, wird wieder überprüft, ob das geometrische Mittel kleiner als 0.027 ist oder nicht. Wenn ja, dann gibt es keine Übereinstimmung. Ist aber dieser Bedienung wahr für mehr als einen Wert, gibt es keine Übereinstimmung.

Auf Abbildung 1.8 ist ein Flussdiagramm dargestellt, das die Abfolge dieser Bedingung beschreibt.

Unten ist der Codeabschnitt, der zeigt wie das funktioniert, ersichtlich:

```
count = 0
for i in range(0,len(MAX)):
        if MAX[i] > 0.07 and MAX[i]< 0.084:
                count+=1
                status3=True
```

```
                    status33=True
                    status4=True
#Wenn JA :Wenn gmT kleiner als 0.027 ist passt,
                    sonst FALSCH
            if count == 1:
                    global status33
                    status33=False
            if gmT > 0.027:
                    global status3
                    status3 = False
#Wenn 2 oder mehr: FALSCH
            elif count > 1:
                    global status4
                    status4 = False
```

iii. 3.Fall: Mindestens einen Maximumwert ist größer als 0.084

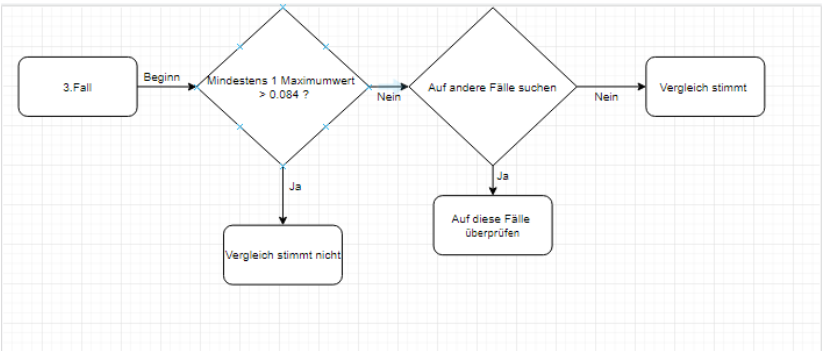

Abbildung 1.9: 3_Fall

Trifft dieser Bedienung zu, stimmt die Erkennung nicht und der Person wird nicht vom System erkannt.

Auf Abbildung 1.9 ist ein Flussdiagramm dargestellt, das die Abfolge dieser Bedingung beschreibt.

Unten ist der Codeabschnitt, der zeigt wie das funktioniert, ersichtlich:

```
for i in range(0,len(MAX)):
            if MAX[i] > 0.084:
                    count+=1
                    status5=True
#Wenn JA: FALSCH
            if count >=1:
                    global status5
                    status5=False
```

(c) Andere Abgrenzungen und Entscheidungen

Trifft eine von diesen 2 unterstehenden Kombinationen zu, ist weitere Überprüfung (geometrisches Mittel) nicht mehr nötig und es gibt keine Übereinstimmung.

i. 1 Maximumwert größer als 0.06 und 1 Maximumwert größer als 0.07

ii. 2 Maximumwerte größer als 0.06 und 1 Maximumwert größer als 0.07

Unten ist der Codeabschnitt, der zeigt wie das funktioniert, ersichtlich:

```
if status1==False or status2==False
or status3==False or status4==False
or status5==False or
 (status11==False and status33==False)
or (status22==False and status33==False):
        print("Vergleich nicht erfolgreich,
        sie sind nicht eingeloggt!!!")
        os.system
        ('./Log_Erkennung_Person_Existiert_Nicht.py')
```

- *Ist die Erkennung erfolgreich, kriegt der Benutzer eine Information, dass er im System erfolgreich angemeldet ist.*

Wenn keinen Fall bzw. Bedienung zutrifft, ist die Authentifizierung erfolgreich. Der Benutzer kriegt danach eine Information, die ihm Bescheid gibt, dass die Authentifizierung erfolgreich war und dass er weitermachen darf. Dazu wurde eine gelbe LED verwendet, die leuchtet.

Unten ist der Codeabschnitt, der zeigt wie das funktioniert, ersichtlich:

```
else:
        b="existiert"
        print("Vergleich erfolgreich, Sie sind eingeloggt!!!")
        os.system('./Log_Erkennung_Person_Existiert.py')
```

Andere wichtige Punkte und Abgrenzungen

Es gibt auch einige Punkte, die erwähnt werden müssen, und die zur Verwendung des Systems wesentlich sind:

1. Distanz der Benutzer von der Kamera

Damit der Vergleich und die Erkennung der Gesichter erfolgreich und so genau wie möglich sind, gibt es eine Distanz, die der Benutzer von der Kamera stehen muss. Diese Distanz ist 1 Meter, dem Benutzer wird das durch einer Linie auf dem Boden angezeigt.

2. Unbekannte Emailadresse eingegeben

   Wenn der Benutzer nach dem Drücken des Tasters eine falsche Emailadresse eingibt, wird er informiert, dass er eine zweite Chance bekommen wird, die richtige Emailadresse einzugeben(Vielleicht hat er zum Beispiel ein Fehler beim Eintippen gemacht). Ist die Adresse noch immer unbekannt oder inkorrekt, beginnt der Vergleichsprozess nicht und der Benutzer kann sich nicht anmelden.

3. Kein eindeutiges Gesicht von der Kamera nach der Bildaufnahme gefunden

   Wenn nach der Bildaufnahme kein eindeutiges Gesicht erkannt wird, wird ein zweites Bild gemacht. Der Benutzer wird durch eine Meldung erinnert, dass er gerade und an der Linie stehen muss.

4. Mehrere Gesichter von der Kamera nach der Bildaufnahme gefunden

   Wenn nach der Bildaufnahme mehrere Gesichter erkannt werden, wird wieder ein zweites Bild gemacht. Werden wieder mehrere Gesichter erkannt, kann das System nicht überprüfen, ob such der Benutzer anmelden darf oder nicht (Nicht Ziel).

5. Benutzer vom System nicht erkannt

   Ist die Authentifizierung nicht erfolgreich, leuchtet eine rote LED um den Benutzer zu zeigen, dass er sich nicht anmelden darf. Er kann aber alles nochmals probieren(Erkennungstaster drücken usw. . . .), oder er kann sich registrieren.

6. Verarbeitungszeit des Systems

   Jedes intelligente System braucht einer Verarbeitungszeit. Der Erkennungsprozess braucht ungefähr 7 bis 8 Sekunden. Das ist aber auch von der Geschwindigkeit des Benutzers abhängig.

Das Testen

Eine der wichtigsten Arbeitspakete des Erkennungsteils war das Testen des Vergleichsverfahrens. Um die Vergleichsbedingungen und Begrenzungen, die weiter oben erwähnt wurden, zu finden und zu bestimmen, war es wesentlich und extrem notwendig, viele Tests zu machen. Zuerst hat der Prozess des Testens mit verschiedene Bilder einer Person begonnen. Danach wurden verschieden Personen verwendet, um eine Toleranz zu finden, die in der Bedienungen verwendet werden kann. Nachdem die Gesichtspunkte

von 2 Bilder getestet und verglichen waren, sind die Ergebnisse auch mit den in der Datenbank gespeicherten Punkten getestet werden. Am Ende des Testens sind die vorher erklärte Bedingungen definiert und formuliert.

Das Logging

Jedes große System braucht Logging. Mit Hilfe einer Logdatei (englisch: log file) können alle oder bestimmte Aktionen von Prozessen auf einem Computersystem automatisch protokolliert werden. Das heißt, diese Logdatei kann vom Benutzer verwendet werden, um Fehlermeldungen zu erhalten, um den Fehler zu verstehen, um den Zeitpunkt verschiedener wichtiger Ereignissen zu erfahren und vieles mehr. Alle diese Vorteile und Merkmale des Logging waren mehr als genug zum entscheiden, dass die Protokollierung der Ereignisse des Systems ein wichtiges Ziel sind. Das war der Grund, warum der Gesichtserkennungsteil auch protokolliert.
Es wurden die wichtigsten Ereignisse mitprotokolliert:

1. Der Zeitpunkt, wann der Erkennungstaster gedrückt wurde

   Es ist sehr wichtig für die Admins des Systems zu wissen, wann der Erkennungstaster gedrückt wurde, bzw. wann die Gesichtserkennungsprozess beginnt. Diese Zeit kann dafür verwendet werden, um die totales Dauer des Erkennungsprozesses zu bestimmen.

2. Der Zeitpunkt, wann der Benutzer erfolgreich angemeldet war

   Dieses Ereignis ist ein sehr wichtiges, weil es der Schlusspunkt des Erkennungsteils ist. Es kann auch als Beweis verwendet werden, dass der Benutzer im System schon registriert war.

3. Der Zeitpunkt, wann der Benutzer sich nicht erfolgreich anmeldet

   Wie vorher erwähnt, ist eine Log-Datei eine sehr gute Methode, wichtige Fehlermeldungen sehr schnell zu erhalten. In diesem Fall wird gespeichert, wann der Benutzer sich nicht erfolgreich anmeldet.

4. Der Zeitpunkt, wann die eingegebene Emailadresse in der Datenbank nicht gefunden wird

   Gibt der Benutzer eine falsche oder eine nicht existierende Emailadresse ein, wird dieses Ereignis geloggt. Das ist extrem wichtig für die Administratoren, weil ohne eine vom Benutzer eingegebene Emailadresse, die in der Datenbank schon existiert, beginnt die Vergleichsprozess für die Erkennung gar nicht. Auf diesem Weg wissen die Administratoren, wo der Fehler ist, und der Benutzer wird darüber informiert.

Wie wurde das Logging gemacht?
Das Logging des Gesichtserkennungsprozesses war nicht sehr komplex in Python. Es gab eine Tabelle in der Datenbank, die für dieser Funktion geeignet war. In dieser Tabelle wurden alle Informationen, die protokolliert werden müssen, gespeichert. Mit Hilfe einer Python Skript wurde dann die Verbindung aller Skripten, die die notwendige Informationen senden mit dieser Log Tabelle, die in der Datenbank liegt, gemacht. Es wurde danach ein Programm für jeder Ereignis, das oben erwähnt war, erzeugt. Diese Programme ermöglichten die Verbindung mit der Datenbank, und zeigten eine Anmeldung. Beispielweise wurde in der Datenbank notiert, dass es ein Problem bei Gesichtserkennung gab(Taster nicht gedrückt..). In der Hauptskript wurde dann dieses einzelne Programm aufgerufen.

Unten ist der Codeabschnitt, der zeigt wie das funktioniert, ersichtlich.

Der Taster wird gedrückt:

```
if GPIO.input(17):
os.system('./Log_Erkennung_TasterGed.py')
```

Zeitpunkt wird gespeichert:

```
logging.basicConfig(filename=log_file_path)
if(log_to_db):
logging.getLogger('').addHandler(logdb)
log=logging.getLogger('Gesichtserkennung')
log.setLevel(log_error_level)
test_var='Taster gedruckt'
log.error('This event occurred: %s' % test_var)
```

1.3 Herausforderungen, Probleme, und deren Lösung

Während dieser Arbeit musste man einigen Problemen und Herausforderungen lösen.

- *OpenCV Installation*

  Die größten Probleme hat es bei der Installation von OpenCV gegeben. Diese Installation hat sehr lang gedauert und es war sehr schwer zu bestimmen, welche Paketen ausgelassen werden sollten und welche nicht. Dieses Problem wurde nach vielen Tests gelöst, durch das Kopieren von einer anderen SD-Karte, auf der OpenCV bereits installiert war. CMake und Make waren sehr wichtige Pakete, mit denen diese Aufgabe erledigt wurde.

- *Fritzing*

  Das Problem beim Fritzing war, dass jedes Mal wenn das Programm beendet wurde, es nicht mehr geöffnet werden konnte. Es fehlten entweder die Pfaden oder

die Bauteile, die für die Schaltung notwendig waren. Die Lösung war eigentlich sehr leicht. Das Programm wurde komplett gelöscht und dann wieder im System installiert, und danach wurde die ganze Schaltung gemacht, ohne das Programm zu beenden. Sonst wären dieselben Probleme wieder aufgetreten.

- *Kurzschluss beim Raspberry Pi*

  Was noch passiert ist, ist das es beim Anfassen vom Raspberry PI einen Kurzschluss gab. Der Raspberry PI war kaputt und musste ersetzt werden. Glücklicherweise konnte die SD-Karte erfolgreich in kopiert werden und deshalb sind alle Daten und Informationen gespeichert.

- *Paket FaceRecognition Installation*

  Die Installation des Paketes FaceRecognition konnte nicht erfolgreich gemacht werden, weil die dlib Pakete auch gebraucht wurden. Wahrscheinlich aufgrund des zu geringen RAMs kann diese Pakete nicht installiert werden. Dieses Problem wurde noch nicht gelöst.

- *Probleme bei der Aufruf der Skripten*

  Am Beginn war das Umgehen mit dem Aufruf der Skripten sehr schwer. Es war schwer zu verstehen wie das genau funktionierte, weil die große Variablenanzahl die Arbeit kompliziert machte. Dieser Anzahl wurde reduziert und es wurden viele Recherchen über die korrekten Verwendung des *sys* Paketes gemacht um das Problem zu lösen.

- Schlechtere Vergleichsqualität

  Am Beginn des Testens des Vergleichs waren die Ergebnisse nicht stabil. Es konnte keinen Zusammenhang gefunden werden. Beispielsweise waren die Maximumwerte der Abstände bei Bilder, die derselben Person gehörten größer als bei Bildern, die verschiedenen Personen gehörten. Nach viele Tests wurde der Grund verstanden. Die 68 Punkte mussten in 5 verschiedene Teile(Gesicht, Nase, linke Auge, rechte Auge, Mund) zerlegt werden. In dieser Weise wird ein Punkt, der dem Mund gehört nicht mit einem Punkt, der der Nase gehört verglichen.

- Probleme bei Einteilung der Gesichtspunkte in 5 Bereichen

  Der Prozess der Einteilung der Gesichtspunkte in dem verschiedenen Bereiche war auch schwierig. Ein großes Array mit 68 Elementen wurde verwendet. Es war sehr kompliziert, auf einem Index dieses Array zu zugreifen. Deshalb wurde dieses große Array in 5 kleinere Arrays zerlegt. Auf diese Weise war es viel leichter zu verstehen, wie die Indizien funktionieren und wie man dort zugreifen kann.

- Probleme bei der Berechnung des geometrischen Mittels

Die Berechnung des geometrischen Mittels war auch nicht leicht. Nachdem die Formel gefunden war, musste es in dem Programm integriert werden. Die Schwierigkeit war bei dem Ziehen von der N-Wurzel. Sollte das n 67 oder 68 sein? Das war von dem Index abhängig. Am Ende wurde verstanden, dass das Produkt aller Abstände hoch 1/68, die richtige Lösung ist.

- Probleme mit der Datenbank

  Probleme hat es noch beim Zugriff auf der in der Datenbank gespeicherten Gesichtspunkte gegeben. Es war sehr kompliziert zu verstehen, wie und wo die Punkte genau gespeichert waren(Auf welchem Index die X-Werte, auf welchem die Y-Werte). Es wurde deshalb Hilfe von dem Mitarbeiter, der sich mit der Datenbank beschäftigt hätte, gebraucht, und danach war alles klarer und verständlich.

- Probleme mit GIT

  Am Ende müssen die Probleme mit Git erwähnt werden. Es gab viele Fälle, wo es Konflikten mit der Dateien gab. Man muss immer zuerst Pull machen, und danach Push. Das ist aber oft nicht passiert. Es wurde in einer Datei gearbeitet, die aber von einem anderen Person schon geändert wurde. Deshalb gab es Probleme bei den Push. Oft mussten die ganze Dateien gelöscht werden, und danach wieder vom Git geholt werden. Es wurde am Ende zwischen der Gruppe darüber gesprochen und das Problem gelöst.

1.4 Projektmanagement und Controlling

In Bezug auf Projektmanagement und Controlling wurde die Methode des Fehlerbaums verwendet. Diese ist eine berühmte Methode um die Aufwandschätzung zu kalkulieren und um eine Fehlerursache zu finden. Es wird das Problem in kleinen Teile geschnitten damit es klarer wird. Es wurde eine detaillierte Soll-Ist Analyse gemacht, die bei der neuen Aufgabeteilung sehr geholfen hat, und eine Fehlerbaumanalyse zu erstellen.

Die Fehlerbaumanalyse wird verwendet, um die Zuverlässigkeit von technischen Systemen zu testen. Die Fehlerbaumanalyse nimmt als Ausgangspunkt nicht eine einzelne Systemkomponente, sondern das potenziell gestörte Gesamtsystem. Die Fehlerbaumanalyse baut auf der sogenannten negativen Logik auf. Das heißt, der Fehlerbaum beschreibt eine Ausfallsfunktion die bei dem Zustand logisch-1 einen Ausfall ausdrückt, bei logisch-0 liegt ein funktionsfähiges System vor. Sie gehört zu den “Top-Down”-Analyseformen im Risikomanagement. In einem ersten Schritt wird daher das Gesamtsystem detailliert und exakt beschrieben. Darauf aufbauend wird analysiert, welche primären Störungen eine Störung des Gesamtsystems verursachen oder dazu beitragen können. Ausgangspunkt ist hierbei zunächst ein einziges unerwünschtes Ereignis, welches an der Spitze des Fehlerbaums steht, das sogenannte Top-Ereignis. Das Top-Ereignis resultiert in der Regel aus einer Risikoanalyse bzw. Szenarioanalyse. In der einfachsten Form besteht er aus folgenden Elementen: Entscheidungsknoten (E), die

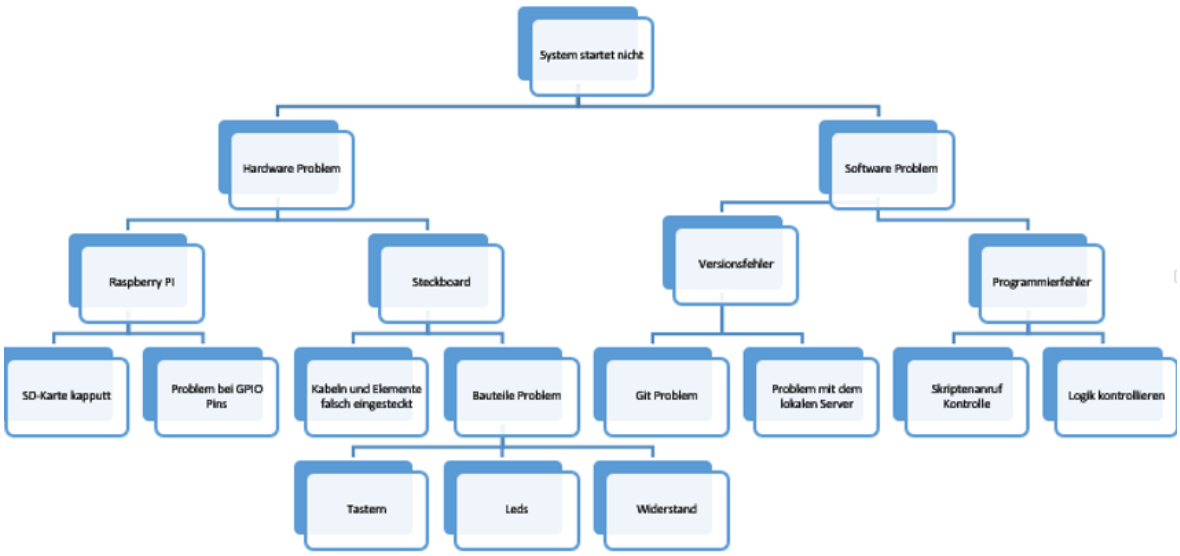

Abbildung 1.10: Fehlerbaum

Entscheidungen kennzeichnen, Zufallsknoten, die den Eintritt eines zufälligen Ereignisses darstellen sowie aus Ergebnisknoten (R), die das Ergebnis von Entscheidungen oder Ereignissen darstellen.

Auf der Abbildung 1.10 wird ein Fehler in den Gesichtserkennungsteil des Systems beschrieben. Zuerst muss es überprüft werden, ob der Fehlern bei der Hardware oder Software liegt.

Hardware Problem
Ist es ein Problem in Hardware, wird der Raspberry Pi und die Steckboard getestet. Beim Raspberry kann es entweder Probleme mit dem SD-Karte oder mit der GPIO Pins geben. Bei dem Steckboard sind entweder die Kabel und die Verbindungen falsch eingesteckt, oder die Bauteile sind kaputt(Taster, Widerstand, LED, Kamera).

Software Problem
Wenn es ein Softwareproblem ist, sind entweder die Versionen vermischt(Problem mit GIT), oder es gibt Programmierfehlern. Diese Programmierfehlern können bei der Logik(Algorithmen) oder bei der Syntax auftreten(Skripten Aufruf, Variablen usw.)

1.5 Ergebnisse

Die finale Ergebnisse sind die folgende:

1. Systemaufbau

Das ganze System wurde zusammen mit der Hardware aufgebaut.

2. Digitale Darstellung des Systemaufbaus
   Der Schaltplan des Systems wurde durch Fritzing digital dargestellt.

3. Erkennung der Gesichter
   Es wurde eine Gesichtserkennung erreicht, die in 80 Prozent der Fälle funktioniert.

4. Aufnahme der Benutzer Gesichtsdaten mithilfe der Emailadresse
   Gibt der Benutzer seine Emailadresse ein, werden die zugehörigen Gesichtsdaten von der Datenbank geladen, um den Vergleich zu machen.

1.6 Was wurde gelernt und welche Verbesserungs möglichkeiten gibt es

Während dieser Diplomarbeit wurde viele wichtiges gelernt und verstanden. Die wichtigsten Lektionen werden unten genannt und erklärt.

- Zeitmanagement

  Die Wichtigkeit der Zeit wurde wirklich verstanden. Man hatte nicht mehr die Möglichkeit, Dinge immer für den letzte Moment zu lassen, weil es einfach zu viel ist. Ohne einen detaillierten Zeitplan und ohne eine systematische Arbeit würde dieses Projekt nicht fertig sein.

- Wie man mit OpenCV und mit Bilderdaten arbeitet

  Es wurde gelernt, wie man überhaupt mit Bildinformationen arbeitet, und wie das Paket OpenCV funktioniert. In Zukunft wenn es um Arbeiten mit Bilderdaten geht, ist das Basiswissen vorhanden.

- Wichtigkeit des Testens des Systems

  Wie wichtig der Test des Prozesses ist wurde auch hier verstanden. Vorher wurde gedacht, dass das Testen des Systems nur eine zusätzliche und unwichtige Arbeit ist. Es war aber nur mit Hilfe des Testens möglich, den Vergleichsprozess erfolgreich zu realisieren und Fehler zu finden. Ein System, das vorher nicht getestet ist, kann kein gutes System sein.

- Gruppenarbeit und Verantwortlichkeiten übernehmen

  Die Rolle der Gruppe ist auch sehr wichtig. Geht es den Mitarbeitern miteinander gut und sie sind gut gelaunt, sind sie auch mehr motivierter zum Arbeiten. Die Kommunikation soll nie fehlen und die Entscheidungen des Projektleiters sollen

akzeptiert werden. Man muss auch seine Verantwortlichkeiten selbst übernehmen und seine Arbeit mit seiner Stärken erledigen. Das wurde alles auch während der Arbeit gelernt.

- Wichtigkeit der Planung und der Begrenzungen

  Es wurde schon gelernt, dass die Planung eines Projekts sehr wichtig war, aber mit Hilfe dieser Diplomarbeit wurde ganz genau verstanden, warum das so ist. Ohne die Big Pictures und die Erste Ebene wurde der Aufwand der Arbeit viel grösser, und die Zeit würde sicherlich nicht genügen. Die Begrenzungen sind auch wichtig, weil ohne die wurde man Zeit an etwas verlieren, das am Ende nicht realisierbar oder nicht nötig ist.

- Wie man mit Fehlern umgeht

  Außerdem wurde verstanden und gelernt, wie man mit Fehlern umgehen soll. Pessimismus und das Aufgeben sind nicht die Lösungen. Man muss optimistisch sein und viele verschiedene Wege suchen, wie man den Fehlern finden und dann reparieren kann. Fremde Hilfe ist auch irgendwann notwendig.

Verbesserungsmöglichkeiten sind auch erkannt werden. Diese sind:

- Besseres Zeitmanagement

  Die Arbeit musste ein bisschen früherer begonnen worden, und es muss besser verteilt werden. Es muss verstanden werden, dass hinter einem schlechten Projekt eine schlechte Zeitmanagement versteckt.

- Backup Verfahren

  Nächstes Mal wird die Arbeit öfter gespeichert. Back-Ups werden täglich gemacht, damit nichts verloren geht. Man denkt, dass das lokale Speichern genug ist. Was ist aber, wenn der PC kaputt wird? Das wird in die nächste Arbeiten mehr berücksichtigt.

```
#!/usr/bin/python3

import cv2
import numpy as np
import dlib
import sys
import faceDetect_crop_rei as fdcr
emailiperkrahasim=fdcr.emaili
nofaces=""
print(emailiperkrahasim)
def getPoints(Imagename):
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default
    image = cv2.imread(Imagename)
#image=cv2.imread('aroncropped2.jpg')
    if(image is None):
        print("Can't open image file")

#get image dimensions
    dimX = int(image.shape[0])
    dimY = int(image.shape[1])
    print("\tdimensionet e fotos",dimX,dimY)

#convert image into grayscale
    gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)


#load shape predictors to extract landmarks

    predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.d
    detector = dlib.get_frontal_face_detector()

#load face detectors and detect faces
    faces_cv = face_cascade.detectMultiScale(gray,1.1,4)
    faces = detector(gray)

#get number of faces detected
    nrFace = len(faces)
    print("\tDetected faces: %d" % nrFace)

    i = 0
    coords=[]
    z=[]
    vleratx1=np.zeros((68,1),dtype="float")
    vleraty1=np.zeros((68,1),dtype="float")
#get image dimensions
    height, width = image.shape[:2]
```

```
    if nrFace > 0:
        nofaces="face"
        for face in faces:

            x1 = face.left()
            y1 = face.top()
            x2 = face.right()
            y2 = face.bottom()


            i = 0
        #problemi!!
            landmarks = predictor(gray,face)
        #shape = shape_utils.shape_to_np(landmarks)
            coords = np.zeros((68,2),dtype="float")
        #right eye
            for d in range(0,68):
                x = float(landmarks.part(d).x / width)
                y = float(landmarks.part(d).y / height)
            #cv2.circle(gray2, (x, y), 4, (255,0,0), -1)
                i+1
                coords[d] = (x,y)
            #for index in range(len(coords[d]):
        #print(d,coords)
            #print(d,": ",x,y,"\n")
            z=np.hsplit(coords,2)
            vleraty1=z[1]
            vleratx1=z[0]
    elif nrFace <= 0:
        print("\tno faces found")

    ret = dict()
    ret['x'] = vleratx1
    ret['y'] = vleraty1

    return ret
vlera=getPoints(fdcr.filenampererkennung)
```

```
#!/usr/bin/python3
import cv2
import MySQLdb
import sys
import os
import math
import numpy as np
```

```
import dlib
import reitest as rei
conn=MySQLdb.connect('localhost','aronterzeta','aronterzeta','test')
mycursor=conn.cursor()
skaftyra=rei.nofaces
b=""
bool=False
bool=False
myresult=""
maxwert=0
counter=0
mycursor.execute("select * from person where email='%s';"%(rei.emailiperk
global myresult
myresult=mycursor.fetchall()
MAX = [0,0,0,0,0]
for x in myresult:
    if x[2]==rei.emailiperkrahasim:
        bool=True
    else:
        bool=False
        os.system('./Log_Erkennung_Email_Nicht_Gefunden.py')

#except:
    #print ("Select Statement nicht gut")
if bool:
    #os.system('./Log_Erkennung_PersonExistiert.py')
    for res in myresult:
        print(res[5])
        vx=5
        vy=6
        #ret = landmarks.getPoints(arrayFoto[f])
        vleratx = rei.vlera['x']
        vleraty = rei.vlera['y']
        min=10.0
        max=0.0
        summe=0.0
        gm=1.0
        dis=0.0
        print("----------Ftyra Anash----------

        for pika in range(0,27):
            dis=math.sqrt(abs(( (vleratx.item(pika)-float(res[vx])))) * a
            if(dis < min):
                min = dis
            if(dis > max):
                max = dis
```

```
        gm *= dis
        vx+=2
        vy+=2
gm=gm**(1/27)
global MAX
MAX[0]=max
print("Max:%s\t GM:%s\n"%(MAX[0],gm))
min=10.0
max=0.0
summe=0
gm=1
dis=0

print("-----------------------------------------Hunda-----------------------------

for pika in range(27,36):
        dis=math.sqrt(abs(( (vleratx.item(pika)-float(res[vx])))) * a
        if(dis < min):
            min = dis
        if(dis > max):
            max = dis
        gm *= dis
        vx+=2
        vy+=2
gm=gm**(1/9)
#global MAX2
#MAX2=max
global MAX
MAX[1]=max
print("Max:%s\t GM:%s\n"%(MAX[1],gm))
min=10.0
max=0.0
summe=0
gm=1
dis=0

print("-----------------------------------------Syni Djatht-----------------------

for pika in range(36,42):
        dis=math.sqrt(abs(( (vleratx.item(pika)-float(res[vx])))) * a
        if(dis < min):
            min = dis
        if(dis > max):
            max = dis
        gm *= dis
        vx+=2
```

```
        vy+=2
    gm=gm**(1/6)
    global MAX
    MAX[2]=max
    print("Max:%s\t GM:%s\n"%(MAX[2],gm))
    min=10.0
    max=0.0
    summe=0
    gm=1
    dis=0

    print("-----------------------------------------Syni Majt-----------------------------

    for pika in range(42,48):
        dis=math.sqrt(abs(( (vleratx.item(pika)-float(res[vx])))) * a
        if(dis < min):
            min = dis
        if(dis > max):
            max = dis
        gm *= dis
        vx+=2
        vy+=2
    gm=gm**(1/6)
    global MAX
    MAX[3]=max
    print("Max:%s\t GM:%s\n"%(MAX[3],gm))
    min=10.0
    max=0.0
    summe=0
    gm=1
    dis=0

    print("-----------------------------------------Goja-----------------------------------

    for pika in range(48,68):
        dis=math.sqrt(abs(( (vleratx.item(pika)-float(res[vx])))) * a
        if(dis < min):
            min = dis
        if(dis > max):
            max = dis
        gm *= dis
        vx+=2
        vy+=2
    gm=gm**(1/20)
    global MAX
    MAX[4]=max
```

```
print("Max:%s\t GM:%s\n"%(MAX[4],gm))
min=10.0
max=0.0
summe=0
gm=1
dis=0
print("-----------------------------------------Total-----------------------------
kx=5
ky=6
for pika in range(0,68):
    dis=math.sqrt(abs(( (vleratx.item(pika)-float(res[kx])))) * a
    if(dis < min):
        min = dis
    if(dis > max):
        max = dis
    gm *= dis
    kx+=2
    ky+=2
gmT=gm**(1/68)
global MAX6
MAX6=max
print("Max:%s\t GM:%s\n"%(MAX6,gmT))

#1)  Mindestens 2 von 5 Maximumwerte sind groesser als 0.06:
global count
count = 0
for i in range(0,len(MAX)):
    if MAX[i] > 0.06:
        count+=1
status1=True
status11=True
status2=True
status22=True
#  Wenn JA:    Nur wenn gmT kleiner als 0.01 sind sie richtig, so
if count ==1:
    global status11
    status11=False
if count >= 2:
    global status22
    status22=False
    if gmT > 0.01:
        global status1
        status1 = False
 #   Wenn NEIN:  gmT muss kleiner als 0.027 sein,um RICHTIG zu se
else:
    if gmT > 0.027:
```

```
                global status2
                status2 = False

        # 2)  Eine MAXwert groesser als 0.07:
         count = 0
         for i in range(0,len(MAX)):
             if MAX[i] > 0.07 and MAX[i]< 0.084:
                 count+=1
         status3=True
         status33=True
         status4=True
         #         Wenn JA:     Wenn gmT kleiner als 0.027 ist passt, sonst
         if count == 1:
             global status33
             status33=False
             if gmT > 0.027:
                 global status3
                 status3 = False
          #         Wenn 2 oder mehr: FALSCH
         elif count > 1:
             global status4
             status4 = False


         #3)  Eine oder mehrere MAXwert groesser als 0.084:
         for i in range(0,len(MAX)):
             if MAX[i] > 0.084:
                 count+=1
         status5=True
          #  Wenn JA: FALSCH
         if count >=1:
             global status5
             status5=False
         #4)  Wenn Case 1&2 oder 2&3 eintretten:
          #         Falsch
         if status1==False or status2==False or status3==False or status4=
         #if (status1==False and status2==False) or (status2==False and st
             print("Vergleich nicht erfolgreich , sie sind nicht eingeloggt
             os.system('./Log_Erkennung_Person_Existiert_Nicht.py')
         else:
             b="existiert"
             print("Vergleich erfolgreich , Sie sind eingeloggt !!!")
             os.system('./Log_Erkennung_Person_Existiert.py')

else:
    b="nicht existiert"
```

```
    print("FEHLER!!!!")

#!/usr/bin/python3
import MySQLdb
import time
import logging
db_server='localhost'
db_user='aronterzeta'
db_password='aronterzeta'
db_dbname='test'
db_tbl_log='log'
log_file_path='/home/pi/diplomarbeit/python_scripts/fehlerlog.txt'
log_error_level='DEBUG'
log_to_db=True

class LogDBHandler(logging.Handler):
    def __init__(self,sql_conn,sql_cursor,db_tbl_log):
        logging.Handler.__init__(self)
        self.sql_cursor=sql_cursor
        self.sql_conn=sql_conn
        self.db_tbl_log=db_tbl_log
    def emit(self,record):
        tm=time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(record.creat
        self.log_msg=record.msg
        self.log_msg=self.log_msg.strip()
        self.log_msg=self.log_msg.replace('\'', '\'\'')
        sql='insert into '+self.db_tbl_log + ' (log_level, ' + \
            'log_levelname,log,created_at,created_by) ' + \
            'values (' + \
            '' + str(record.levelno) + ', ' + \
            '\'' + str(record.levelname) + '\', ' + \
            '\'' + str(self.log_msg) + '\', ' + \
            ' \'' + tm + '\', ' + \
            '\'' + str(record.name) + '\')'
        try:
            self.sql_cursor.execute(sql)
            self.sql_conn.commit()
        except MySQLdb.Error as e:
            print (sql)
            print("Critical DB Error")
if(log_to_db):
    log_conn=MySQLdb.connect(db_server,db_user,db_password,db_dbname,30)
    log_cursor=log_conn.cursor()
    logdb=LogDBHandler(log_conn,log_cursor,db_tbl_log)

logging.basicConfig(filename=log_file_path)
if(log_to_db):
```

```
    logging.getLogger('').addHandler(logdb)
log=logging.getLogger('Gesichtserkennung')
log.setLevel(log_error_level)
test_var='Email existiert nicht!'
log.error('This event occurred: %s' % test_var)
```

```
#!/usr/bin/python3
import MySQLdb
import time
import logging
db_server='localhost'
db_user='aronterzeta'
db_password='aronterzeta'
db_dbname='test'
db_tbl_log='log'
log_file_path='/home/pi/diplomarbeit/python_scripts/fehlerlog.txt'
log_error_level='DEBUG'
log_to_db=True

class LogDBHandler(logging.Handler):
    def __init__(self,sql_conn,sql_cursor,db_tbl_log):
        logging.Handler.__init__(self)
        self.sql_cursor=sql_cursor
        self.sql_conn=sql_conn
        self.db_tbl_log=db_tbl_log
    def emit(self,record):
        tm=time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(record.creat
        self.log_msg=record.msg
        self.log_msg=self.log_msg.strip()
        self.log_msg=self.log_msg.replace('\'', '\'\'')
        sql='insert into '+self.db_tbl_log + ' (log_level, ' + \
            'log_levelname,log,created_at,created_by) ' + \
            'values (' + \
            '' + str(record.levelno) + ', ' + \
            '\'' + str(record.levelname) + '\', ' + \
            '\'' + str(self.log_msg) + '\', ' + \
            ' \'' + tm + '\', ' + \
            '\'' + str(record.name) + '\')'
        try:
            self.sql_cursor.execute(sql)
            self.sql_conn.commit()
        except MySQLdb.Error as e:
            print (sql)
            print("Critical DB Error")
if(log_to_db):
    log_conn=MySQLdb.connect(db_server,db_user,db_password,db_dbname,30)
    log_cursor=log_conn.cursor()
```

```
    logdb=LogDBHandler(log_conn,log_cursor,db_tbl_log)

logging.basicConfig(filename=log_file_path)
if(log_to_db):
    logging.getLogger('').addHandler(logdb)
log=logging.getLogger('Gesichtserkennung')
log.setLevel(log_error_level)
test_var='Person wurde nicht gefunden, Vergleichung stimmt nicht'
log.error('This event occurred: %s' % test_var)
```

```
#!/usr/bin/python3
import MySQLdb
import time
import logging
db_server='localhost'
db_user='aronterzeta'
db_password='aronterzeta'
db_dbname='test'
db_tbl_log='log'
log_file_path='/home/pi/diplomarbeit/python_scripts/fehlerlog.txt'
log_error_level='DEBUG'
log_to_db=True

class LogDBHandler(logging.Handler):
    def __init__(self,sql_conn,sql_cursor,db_tbl_log):
        logging.Handler.__init__(self)
        self.sql_cursor=sql_cursor
        self.sql_conn=sql_conn
        self.db_tbl_log=db_tbl_log
    def emit(self,record):
        tm=time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(record.creat
        self.log_msg=record.msg
        self.log_msg=self.log_msg.strip()
        self.log_msg=self.log_msg.replace('\'', '\'\'')
        sql='insert into '+self.db_tbl_log + ' (log_level, ' + \
            'log_levelname,log,created_at,created_by) ' + \
            'values (' + \
            '' + str(record.levelno) + ', ' + \
            '\'' + str(record.levelname) + '\', ' + \
            '\'' + str(self.log_msg) + '\', ' + \
            ' \'' + tm + '\', ' + \
            '\'' + str(record.name) + '\')'
        try:
            self.sql_cursor.execute(sql)
            self.sql_conn.commit()
        except MySQLdb.Error as e:
            print (sql)
```

```
            print("Critical DB Error")
if(log_to_db):
    log_conn=MySQLdb.connect(db_server,db_user,db_password,db_dbname,30)
    log_cursor=log_conn.cursor()
    logdb=LogDBHandler(log_conn,log_cursor,db_tbl_log)

logging.basicConfig(filename=log_file_path)
if(log_to_db):
    logging.getLogger('').addHandler(logdb)
log=logging.getLogger('Gesichtserkennung')
log.setLevel(log_error_level)
test_var='Person existiert schon, Vergleichung stimmt'
log.error('This event occurred: %s' % test_var)
```

```
#!/usr/bin/python3
import MySQLdb
import time
import logging
db_server='localhost'
db_user='aronterzeta'
db_password='aronterzeta'
db_dbname='test'
db_tbl_log='log'
log_file_path='/home/pi/diplomarbeit/python_scripts/fehlerlog.txt'
log_error_level='DEBUG'
log_to_db=True

class LogDBHandler(logging.Handler):
    def __init__(self,sql_conn,sql_cursor,db_tbl_log):
        logging.Handler.__init__(self)
        self.sql_cursor=sql_cursor
        self.sql_conn=sql_conn
        self.db_tbl_log=db_tbl_log
    def emit(self,record):
        tm=time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(record.creat
        self.log_msg=record.msg
        self.log_msg=self.log_msg.strip()
        self.log_msg=self.log_msg.replace('\'', '\'\'')
        sql='insert into '+self.db_tbl_log + ' (log_level, ' + \
            'log_levelname,log,created_at,created_by) ' + \
            'values (' + \
            '' + str(record.levelno) + ', ' + \
            '\'' + str(record.levelname) + '\', ' + \
            '\'' + str(self.log_msg) + '\', ' + \
            ' \'' + tm + '\', ' + \
            '\'' + str(record.name) + '\')'
        try:
```

```
            self.sql_cursor.execute(sql)
            self.sql_conn.commit()
        except MySQLdb.Error as e:
            print (sql)
            print("Critical DB Error")
if(log_to_db):
    log_conn=MySQLdb.connect(db_server,db_user,db_password,db_dbname,30)
    log_cursor=log_conn.cursor()
    logdb=LogDBHandler(log_conn,log_cursor,db_tbl_log)

logging.basicConfig(filename=log_file_path)
if(log_to_db):
    logging.getLogger('').addHandler(logdb)
log=logging.getLogger('Gesichtserkennung')
log.setLevel(log_error_level)
test_var='Taster gedruckt'
log.error('This event occurred: %s' % test_var)

#!/usr/bin/python3.5
import RPi.GPIO as GPIO
import time
import os
import subprocess
import sys
from getpass import getpass
#from Person_Registrierung import (variable3)
#os.system("./enter.sh")
GPIO.setmode(GPIO.BCM)

GPIO.setup(23, GPIO.OUT)#AronLed
GPIO.setup(18, GPIO.IN)#AronSchalter
GPIO.setup(17, GPIO.IN)#ReiLED
GPIO.setup(27, GPIO.OUT)#ReiSchalter
a=""
b=""
correctpaswadmin="@r0nt3rz()"
#image2=sys.argv[1]
#foto2=sys.argv[1]
while True:
    if GPIO.input(18):
        os.system('./fehlerlog_Gesichtsregistrierung.py')
        print("Password for admin please :")
        inputpasw=getpass()
        if(inputpasw == correctpaswadmin):
            os.system('./fehlerlog_Gesichtsregistrierung_admin.py')
        #os.system('./Test '+(image2)+".jpg")
        #exec(open('./Vergleich_2_Fotos.py').read())
```

```
        #if(c=="matched"):
        #os.system('./connection.py')
                exec(open('Person_Registrierung.py').read())
                print(variable3)
                if (a=="inserted values"):
                    os.system('./Test_'+(variable3)+".jpg")
                    #os.system('./selectID_UpdateBild.py '+(variable3))
                    os.system('./68fLandmarks.py_%s'%(variable3))
                    GPIO.output(23,GPIO.HIGH)
                    time.sleep(1.5)
                    GPIO.output(23,GPIO.LOW)
                #exit()
                else:
                    print("Fehler")
                    GPIO.output(23,GPIO.LOW)
        #exit()
        else:
            print("Admin_ist_nicht_da")
    else:
        #os.system('./connection.py')
        #GPIO.output(23,GPIO.LOW)
        print("Schalter_fuer_Gesichtsregistrierung_nicht_gedrueckt")
        time.sleep(0.5)
        #exit()
    if GPIO.input(17):
        os.system('./Log_Erkennung_TasterGed.py')
        exec(open('reitest3.py').read())
        if b=="existiert" and skaftyra=="face":
            GPIO.output(27,GPIO.HIGH)

            time.sleep(1.5)
            GPIO.output(27,GPIO.LOW)
        else:
            print("nicht_existiert")
            GPIO.output(27,GPIO.LOW)
    else:
        print("Schalter_fuer_Gesichtserkennung_nicht_gedrueckt")
        time.sleep(0.5)
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

Aus dem Netz

[1] URL: https://fritzing.org/learning/.

[2] URL: https://de.serlo.org/mathe/stochastik/daten-datendarstellung/daten-kenngroessen/geometrisches-mittel.

[3] *Einführung in Computer Vision mit OpenCV und Python*. URL: https://blog.codecentric.de/2017/06/einfuehrung-in-computer-vision-mit-opencv-und-python/.

[4] *Unterschied Taster als Schließer, Taster als Wechsler und Schalter*. URL: https://dein-elektriker-info.de/taster-als-schliesser-und-taster-als-wechsler/.