

|  |  |
| --- | --- |
| Computerpraktikum Raspberry Projekt | Klasse:  2BHITM |
| Name:  Lukas Lummerstorfer | KatalogNr:  190225 |
| Projekt:  Gesichts und Iriserkennung mit einem Raspberry | Datum:  27.05.2021 |

**Unser Projekt**

In diesem Projekt sollen wir eine Iris und Gesicht Erkennung mit einer Kamera am ResPi entwickeln.

Inhalt

[Ziele des Projekts: 2](#_Toc74775120)

[Anfangs: 2](#_Toc74775121)

[Start: 3](#_Toc74775122)

[Kamera in Projekt einbinden: 3](#_Toc74775123)

[Python Code: 4](#_Toc74775124)

[Chart Server: 5](#_Toc74775125)

# Ziele des Projekts:

Hier eine Auflistung der Ziele die wir uns Vorgenommen haben.

* Erkennung der Iris
* Erkennung eines Gesichts
* Server zum laufen bekommen
* Ansprechendes Design
* Abglichen von Mustern (ist es einer der Ersteller)
* Programmier Sprache JS

# Anfangs:

In den ersten Stunden wurde besprochen, wie es möglich sein könnte. Meilensteine wurden definiert und Ziele vereinbart.

Weiters gab es Probleme mit den Raspberrys im Zusammenhang mit den SD Karten.

Es wurde festgestellt, dass für eine schnelle und somit bessere Performance bessere SD-Karten notwendig sind. Da der Raspberry viele zufällige Zugriffe pro Sekunde macht wird eine SD-Karte A1 oder A2 benötigt.

Link zum einer 16Gb A1 Karte hier: <https://www.amazon.de/dp/B074B573C4?linkCode=xm2&camp=2025&creative=165953&smid=A3JWKAKR8XB7XF&creativeASIN=B074B573C4&tag=geizhals1-21&ascsubtag=xPkXTDkkVFzYLkCd1vy16w>

# Start:

Mit der neuen SD-Karte kann jedoch bis zur Formatierung nichts gemacht werden.

Man muss entweder das Image selbst auf die Karte Spielen oder man kann den auch den Raspberry Imager verwenden dieser ist leicht auf der Eigenen Seite zu finden.

<https://www.raspberrypi.org/software/>



Als Betriebssystem wurde *Raspberry PI OS (32-Bit)* verwendet.

Bei der SD-Karte musste die Oben erworbene eingestellt werden.

Nach ca. 15min war das IMAGE fertig und der Raspberry bootete.

# Kamera in Projekt einbinden:

Als ersten muss der Raspberry ausgeschalten und vom System getrennt werden.

Anschließend wurde die *Raspberry Pi NOIR V2* (Kamera) in den *MIPI CSI Camera Port* gesteckt. Hierbei muss darauf geachtet werden, dass die Kamera richtig herum ist.

Nach der erfolgreichen Installation der Hardware wurde der Raspberry gestartet. Danach mussten einge Befehle im Terminal ausgeführt werden. Diese sind nun aufgeführt

1. sudo apt-get update && sudo apt-get upgrade
2. sudo su – (um zugriff auf die Einstellungen zu haben)
3. raspi-config (Einstellungen)
   1. Wählen sie Interface Options
   2. Wählen sie Camera
   3. Bestätigen sie das nächste Fenster
   4. Zuletzt auf Finish drücken
   5. Dann neu Starten (wird vorgeschlagen)
4. Raspistill -o test.jpg
   1. Wenn hierbei ein fehler auftritt probieren sie folgenden Command
   2. vcngencmd get\_camera (Im falle supported = 1 detected = 0 könnte es sich um eine Kaputte Kamera handeln)

Wenn sie die schritte gefolgt haben und nun das Foto im Ordner wo sie den *Raspistill* gefunden haben können sie sich sicher sein das die Kamera Fonktioniert.

# Python Code:

Als Basis wurde und ein Code zur Verfügung gestellt.

<https://edufs.edu.htlleonding.ac.at/moodle/pluginfile.php/161735/mod_resource/content/1/im.py>

BILD:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

In diesem Beispiel würden in der Theorie die Gesichter erkannt werden.

Um diesen ausführen zu können musste noch Python installiert werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Es trat jedoch noch ein paar Fehler auf da die 4 Imports(io/picamera/cv2/numpy) noch nicht funktionieren.

Es mussten noch die fehlenden Module cv2 und numpy installiert werden.

1. Ein Bild, das Wand, Person, drinnen enthält.

   Automatisch generierte BeschreibungNumpy: sudo apt-get install python-numpy
2. openCv2: pip install opencv-python

Nun funktionierte die Gesichtes Erkennung ohne Probleme. Als nächsten Schritt implementierten wir ein weiteres Template für die Auge. Wie auch in *Zeile 7 oben für die Gesichter*.

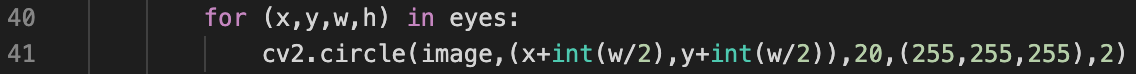
Um nun auch Augenpaare und Gesichter zu erkennen, mussten folgende Zeilen ergänzt werden.

Außerhalb der while(true) Schleife:



Innerhalb:





# Chart Server:

Um die Daten von der Kamera darstellen zu können wurde ein simpler NodeJS Server aufgesetzt.

Dazu musste zuerst NodeJS installiert werden.

Des Anfangs nur einen Statischen Chart anzeigte. Server Dokumente sind unter diesem Link Abruf bar: <https://github.com/P-Klose/CPR-IrisErkennung/tree/main/Server>

VORLETZER STAND:

Der Server muss noch mit der Richtigen POST Rute ausgestattet werden.

Das Python Programm muss noch mit einem request modul versehen werden.

Die weitere Logik muss noch ausprogrammiert werden.

CODE Server:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

CODE Client:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

JETZIGER STAND:

Nun wurde die Sendung der Daten in die Python Datei eingebunden. Hierzu musste das *request* modul importiert werden. Danach wurde ein Objekt objekt erstellt. Und am schluss wurde dieses Objekt mit hilfe von einem Post request an den Server gesendet.

Importieren des letzen Moduls:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Code für das senden von der Anzahl der Gesichter und Augen an den Server:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Servercode POST seitig:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Servercode GET seitig:

