Kamera Abstandsmessung

Tim Rosenkranz
6929884
tim.rosenkranz@stud.uni-frankfurt.de

16. Dezember 2020

Inhaltsverzeichnis

1 Einführung

Seit dem Ausbruch des Coronavirus Anfang 2020 hat sich der Alltag auf der Welt verändert. Es gibt viele neue Regeln zu beachten, wie z.B. einen Abstand einzuhalten. Um zu helfen, den Abstand einzuhalten, wurden meist Bodenmarkierungen eingefügt, doch in einer Welt, die sich immer mehr digitalisiert, bietet sich u.a. die Möglichkeit mittels digitaler Verfahren eine Abstandsmessung durchzuführen.

1.1 Objekterkennung

Der erste Schritt, um den Abstand eines Objektes zu einem anderen Objekt zu bestimmen, ist es die Objekte zunächst zu erkennen - es benötigt also eine Objekterkennung. Dabei bezeichnet Objekterkkung das „Verfahren zum identifizieren bekannter Objekte“, wie z.B. Personen, durch *physikalische* Erkennungsverfahren, wie etwa Kameras als optisches Erkennungsverfahren. Dadurch wird in einem digitalen Bild oder Videostream das Vorhandensein eines Objektes, sowie dessen räumliche Lage bestimmt (Wikipedia 2020).

1.1.1 Tensorflow

Für eine Objekterkennung auf einem Bild oder in einem Video ist eine künstliche Intelligenz notwendig. Das bekannte Framework **Tensorflow** bietet dafür u.a. ein bereits vortrainiertes Tensorflow-Lite Model (Tensorflow 2020).
Darüber hinaus gibt es ein gutes Tutorial, um die Tensorflow-Lite object detection auf einem Raspberry Pi aufzusetzen (Felix 2020). Alternativ ist im Abschnitt *Anwendung - Installation* die Installation vereinfacht und verbessert beschrieben.

1.2 Abstandsermittlungen

Die Hauptfunktion des Programmes ist die Ermittlung des Abstandes zweier (oder meherer) Objekte zueinander. Dieser Abstand wird auf Basis des aus Kameraperspektive horizontalen Abstandes der Objekte und den Abständen der Objekte zur Kamera berechnet.

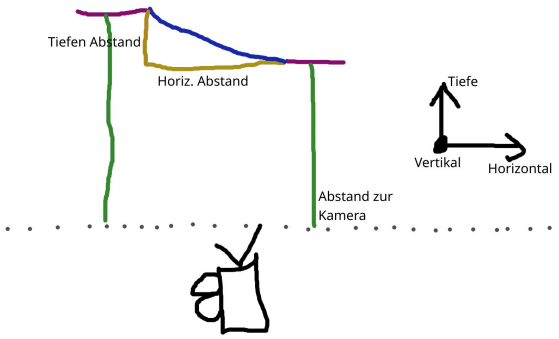


Abbildung 1: Berechnungen des Abstandes. Sicht von oben.

Abbildung 1 zeigt schematisch die Kalkulation des Abstandes. Die blaue Linie ist dabei der Abstand zwischen den Objekten (lila Linien). Dabei kann die blaue Linie mit dem **Satz des Pythagoras** annähernd bestimmt werden. Es gilt

$$\text{Abstand} = \sqrt{(\text{Horizontaler Abstand})^2 + (\text{Tiefenabstand})^2} \tag{1}$$

Horizontaler Abstand

Hierbei können wir den **horizontalen Abstand** durch die bekannte Höhe oder Breite der erkannten Objekte bestimmen. Dabei muss beachtet werden, dass der Pixel (0,0) in der oberen linken Ecke und der Pixel (m,n) in der unteren rechten Ecke zu finden ist. Es muss also zuerst festgestellt werden, welches Objekt relativ aus Kameraperspektive links und welches Objekt realtiv rechts befindlich ist, um die für die Berechnung korrekten Koordinaten zu bestimmen. Anschließend kann die Differenz der horizontalen Koordinaten berechnet werden. Dies ist der benötigte horizontale Abstand.

$$\text{Horizontaler Abstand} = |X_L - X_R| = |X_R - X_L| \tag{2}$$

Hierbei bezeichnet X_L die X-Koordinate des rechten Randes des linken Objekts, X_R die X-Koordinate des linken Randes des rechten Objekts.

Tiefenabstand

Den Tiefenabstand können wir unterdessen über den Abstand der Objekte zur Kamera berechnen. Dabei ist der benötigte Tiefenabstand die Differenz der Abstände der Objekte zur Kamera.

$$\text{Tiefenabstand} = |\text{Distanz}_1 - \text{Distanz}_2| \tag{3}$$

Für die Berechnung des Abstandes eines Objektes zur Kamera ist es notwendig den Fokalwert der Linse der genutzten Kamera, sowie die Breite des Objektes zu wissen. Der Zusammenhang ist gegeben durch die Formel

$$\text{Fokalwert} = \frac{\text{Pixelbreite} \cdot \text{Distanz}}{\text{Objektweite}} \tag{4}$$

Nach der Distanz umgestellt:

$$\text{Distanz} = \text{Fokalwert} \cdot \frac{\text{Objektweite}}{\text{Pixelbreite}} \tag{5}$$

Hierbei bezeichnet der *Fokalwert* die Fokalweite der Linse, die Objektweite die Breite des Objektes in einer Maßeinheit[1], Pixelbreite die Breite des Objektes in Pixel und Distanz die Distanz des Objektes zur Kamera (Rosebrock 2015).

[1] z.B. Meter oder Zentimeter

1.2.1 Abstand mittles Durchschnittswerten

Die Berechnungen für den Abstand zeigen, dass es notwendig ist, die Größe (Höhe, Breite) eines zu detektierenden Objektes zu wissen. Diese ist jedoch nicht immer bekannt, weshalb es u.a. möglich ist auf Durchschnittswerte zurückzugreifen.
Für Personen beträgt die durchschnittliche Breite beispielsweise rund 45 Zentimeter.

2 Anwendung

Das Program kann in verschiedenen Bereichen angewandt werden. Vorzugsweise sollte jedoch die fokale Breite der Kameralinse während der Aufnahme konstant bleiben (kein Zoomen), da die Berechnungen sonst verfälscht werden (ideal sind u.a. Überwachungskameras). Es kann beispielsweise der Abstand zwischen parkenden Autos bestimmt werden, um zu ermitteln, ob noch ein weiteres Auto Platz hat. Eine andere Anwendung wäre die Abstandsermittlung von Personen um zu prüfen, ob diese einen notwendigen Sicherheitsabstand einhalten.

2.1 Installation

Das Program benötigt zum Ausführen eine installierte **Python**-Umgebung der Version **3.6** oder neuer. Zusätzlich ist es notwendig einige wenige **pip** Pakete zu installieren.
Um die **Live Detection** zu nutzen ist zudem eine angeschlossene Kamera notwendig. Ohne eine solche Kamera kann jedoch die **Video Detection** genutzt werden.

2.1.1 Hardware

Es besteht die Notwendigkeit eine **Kamera** anzuschließen, um die Funktionalitäten der **Live Detection** zu nutzen.
Eine einfache headless Überwachungskamera kann beispielsweise mit einem Raspberry Pi und entsprechendem Zubehör (Kamera) errichtet werden.

2.2 Installation

1. Das Repository clonen.
2. (**Optional**) Python virtualvenv[2] installieren:
   2.1. `pip3 install virtualvenv`
   2.2. `python3 -m venv <name>`
   2.3. `source venv/bin/activate` *(aktiviert das virtual environment)*
   2.4. *Um das virtual environment zu deaktivieren:* `deactivate`
3. Dependencies installieren: `pip3 install -r requirements.txt`
4. Die .py Datei ausführen: `python3 detection_webcam_gui.py`

[2]Virtualvenv bietet die Möglichkeit, benötigte Pakete eines Python scripts nur innerhalb dieses environments zu installieren. Dies hat den Vorteil, dass man nicht alle jemals genutzen Pakete in seinem Python Interpreter hat (man startet mit einem „frischen“Interpreter), sondern nur die, die man für ein Projekt braucht. Dadurch lassen sich bugs und Komplikationen vermeiden.

2.2.1 Dependencies

Das Programm basiert auf einigen wenigen `pip` Dependencies. Diese sind:

- Tensorflow (lite)
- opencv (cv2)
- numpy

Die Dependencies können entweder wie mit der Installation beschrieben intsalliert werden oder einzeln mit `pip3 install <numpy tensorflow tflite opencv-python>`.

2.3 Ausführen

Um das Programm auszuführen, kann man das eingebaute GUI nutzen.
Alternativ bietet das Program einige Schnittstellen, um selbst ein User Interface einzubinden.

3 Funktionsweise

Das Program basiert darauf, dass ein Videostream gestartet wird und jeder Frame nacheinander auf Objekte gescannt und die Distanz dieser untereinander berechnet wird.

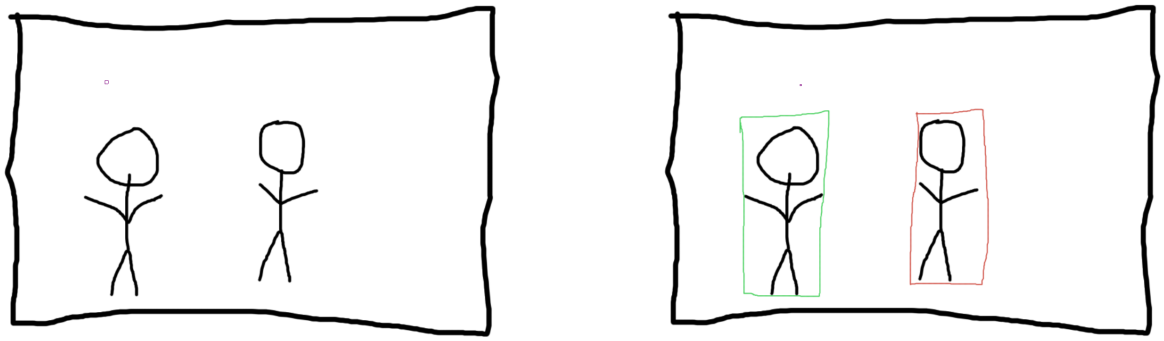
3.1 Ablauf des Programms

Schritt 1: Zuerst wird die Kamera mit dem Python-script gestartet. Sie nimmt kontinuierlich auf und sendet das Bild zur weiterverarbeitung an das Skript.

Schritt 2: Sobald die Kamera einen Input erhält (also aufnimmt) wird über die Frames iteriert. Dabei wird jeder Frame einzeln betrachtet.

Schritt 3: Der Frame wird nun mit der pre-trained Tensorflow Erkennungs-KI predictet. Ist auf dem Frame ein Objekt, für das trainiert wurde[3], zu erkennen werden der Score (Treffsicherheit), die Klasse (Objektbezeichnung) und Koordinaten (obere linke Ecke und untere rechte Ecke) des Objektes gespeichert.

Schritt 4: Nachdem der Frame predictet wurde, wird um jedes erkannte Objekt eine farbige Box gezeichnet, um es zu markieren. Hierbei kann ggf. nach Objekten gefiltert werden, z.B. nach 'person' um nur Personen zu markieren. Für eine bessere Sichtbarkeit und bessere Übersicht wurde ein Farbwechsel für die Boxen implementiert. Hierbei sind sechs Grün- und Rottöne vorhanden, wodurch 36 verschiedene Farben möglich sind.



(a) Schritt 3: Ein Frame

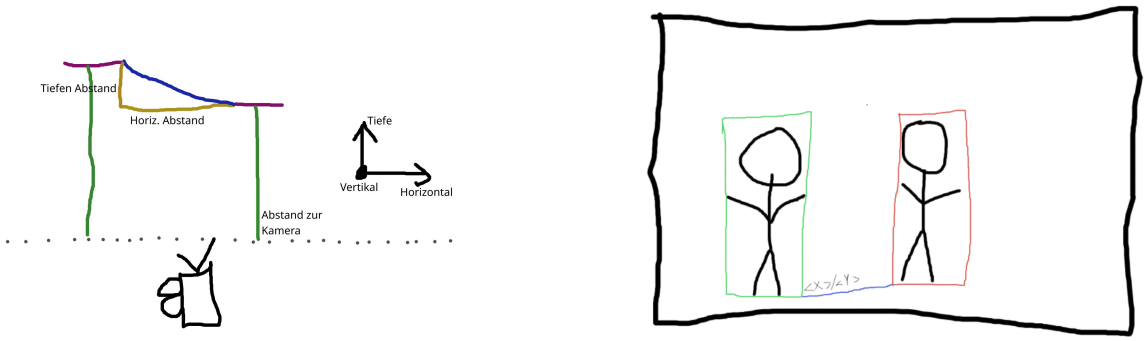
(b) Schritt 4: Boxen um Objekte

Abbildung 2: Objekterkennung

Schritt 5: Zusätzlich zu den Markierungen wird auch der Abstand zwischen jeweils zwei Objekten gemessen. Hierbei wird der relative Abstand auf approximativer Basis mit Hilfe der Durchschnittswerte für Körpermaße bestimmt. Dadurch kann u.a. approximativ gut bestimmt werden, wie viele Pixel ein Zentimeter ergeben. Aus der bekannten Breite der Person kann dann der Abstand zu Kamera ermittelt werden, sowie der horizontale Abstand der beiden Personen / Objekte.

[3] Alle erkennbaren Objekte sind in der labelmap *Sample_Model/labelmap.txt* gelistet

Schritt 6: Nachdem alles kalkuliert wurde, wird geprüft, ob der berechnete Abstand dem Mindestabstand genügt. Wenn nicht, wird eine Linie zwischen den Objekten gezeichnet, die zeigt, dass der Abstand zu gering ist. Zusätzlich wird der aktuell ermittelte Abstand angezeigt, sowie der benötigte Abstand.



(a) Schritt 5: Berechnungen veranschaulicht

(b) Schritt 6: Abstandslinie

Abbildung 3: Abstandsermittlung

Schritt 7: Wiederhole Schritt 3 und folgende, bis das Programm beendet wird (Zum beenden drücke q).

3.2 Kalibrieren

Im Abschnitt zur Ermittlung des Abstandes wurde bereits thematisiert, dass für die korrekte Kalkulation des Abstandes der Objekte die Notwendigkeit besteht, die Fokale Breite der genutzten Kamera(linse) zu wissen. In einigen Fällen wird diese bekannt sein, in anderen Fällen, wenn der Fokalwert nciht bekannt ist, kann eine Kalibrierung durchgeführt werden. Dies bedeutet, dass der Fokalwert anhand eines Objektes ermittelt wird, dessen Breite und Abstand genau bekannt sind. (Siehe auch: Kalibrieren mit dem GUI)

4 Graphisches User Interface

Es wurde ein *GUI* mit *TKinter* implementiert, um eine einfache Bedieung des Programms zu ermöglichen. Dieses GUI bietet dabei auch verschiedene Optionen.

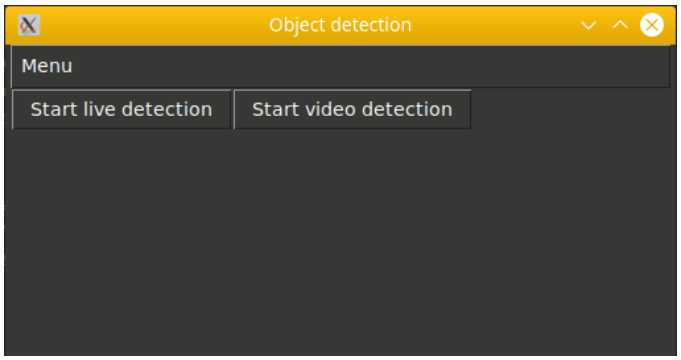
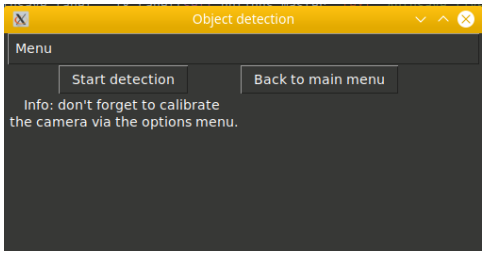


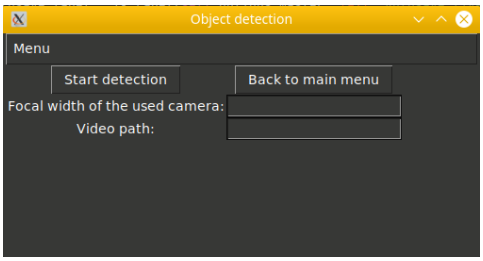
Abbildung 4: Das GUI-Fenster nach start des Programms

4.1 Aufbau des GUI

Nach dem Start des Programmes begrüßt einen eine GUI mit zwei Knöpfen, die wie die Beschriftungen zeigen eine *Live Detection* oder eine *Video Detection* starten. Zusätzlich befindet sich in der oberen linken Ecke ein Drop-down Menü mit einigen wenigen Grundeinstellungsmöglichkeiten.



(a) GUI-Fenster zum Starten einer *Live Detection*



(b) GUI-Fenster zum Starten einer *Video Detection*

Abbildung 5: Start der Detections

Nach dem klicken auf den Knopf für eine *Live Detection* oder *Video Detection* erscheinen einige neue Optionen bzw. Texte auf dem Bildschirm.
Nach Auswahl einer **Live Detection** sollte zunächst eine Kalibrierung vollzogen werden. Dies kann über das Menü ausgewählt werden.
Beim Auswählen einer **Video Detection** muss der Pfad zur Videodatei angegeben werden, sowie die Fokale Länge der benutzten Kameralinse beim Videodreh. Hierbei ist zu beachten, dass dieser Pfad relativ zum Pythonskript oder zum in den Optionen spezifizierten Grundpfad sein muss. Zudem ist eine Kalibrierung bei der *Video Detection* nicht möglich!

4.2 Kalibrierung mit der GUI

Um eine Kalibration zu starten, muss zunächst eine *Live Detection* ausgewählt, aber noch nicht gestartet sein (wie in Graphik 5a). Anschließend kann im Menü unter *calibrate* ausgewählt werden, das Program zu kalibrieren.

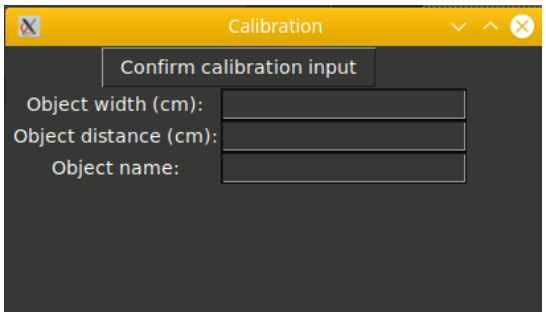


Abbildung 6: Kalibrieren via GUI

Nachdem im Menü die Entsprechende Option ausgewählt wurde, öffnet sich ein neues Fenster. In diesem Fenster müssen nun drei Werte eingegeben werden. Diese Werte sind die Breite und Distanz, sowie der Name eines Referenzobjektes, auf Basis dessen die Fokale Länge berechnet wird (Siehe: Formel 4 aus Abschnitt 1.2). Es sollte darauf geachtet werden, alle Werte in der Maßeinheit *Zentimeter* anzugeben, um fehlerhafte Berechnungen zu vermeiden.
Mit dem Drücken des Knopfes über den Eingabefeldern wird die Eingabe bestätigt und eine Kalibrierung gestartet. Dies dauert ein paar Sekunden. Danach kann das fEnster geschlossen und eine *Live Detection* gestartet werden.

4.3 GUI Optionen

Im Menü unter *options* finden sich ein paar Optionen. Hier kann eingestellt werden, ob die Detection gespeichert werden soll. Ist dieser Haken gesetzt, wird ein *.mp4* Video mit den Visualisierungen um die Objekte gespeichert.
Zusätzlich können eine **Whitelist** und eine **Blacklist** an Objekten gesetzt werden, die erkannt werden sollen. Dies bietet sich z.B. an, wenn man nur Autos erkennen möchte, aber u.U. Personen im Bild auftauchen können.
Die letzte Einstellungsmöglichkeit ist der Standardpfad, an dem nach Videos gesucht und Videos gespeichert werden. Wird dieses Textfeld leer gelassen, wird im selben Ordner des Pythonskript gesucht und dort hin gespeichert.

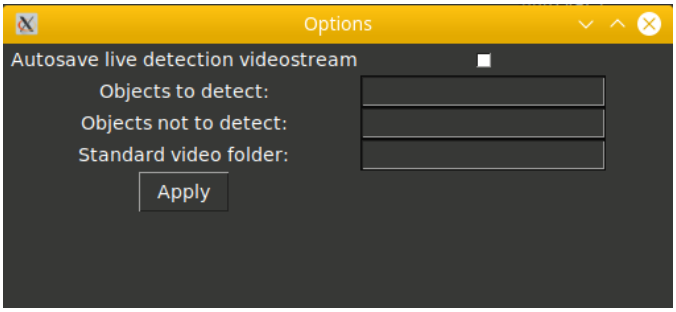


Abbildung 7: Optionen der GUI

Literatur

Felix (Apr. 2020). Raspberry Pi KI: Objekterkennung mittels TensorFlow und Kamera. Deutsch. Raspberry Pi Tutorials. URL: https://tutorials-raspberrypi.de/raspberry-pi-objekterkennung-mittels-tensorflow-und-kamera/ (besucht am 05. 10. 2020).

Rosebrock, Adrian (19. Jan. 2015). Find distance from camera to object/marker using Python and OpenCV. Englisch. URL: https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/ (besucht am 05. 10. 2020).

Tensorflow (23. Sep. 2020). Object detection. Englisch. URL: https://www.tensorflow.org/lite/models/object_detection/overview?hl=en (besucht am 05. 10. 2020).

Wikipedia (10. Juni 2020). Objekterkennung. Deutsch. Wikimedia Foundaition Inc. URL: https://de.wikipedia.org/wiki/Objekterkennung (besucht am 05. 10. 2020).