Fachhochschule Nordwestschweiz
Hochschule für Technik

# Bildverarbeitung

Projketbericht

Windisch, 19. Januar 2023

| | |
|---|---|
| **Studenten** | Matthias Bhend, Luzia Escher |
| **Expertin/Experte** | Jürg Keller |
| **Studiengang** | Elektro- und Informationstechnik |
| **Hochschule** | Hochschule für Technik |

# Inhaltsverzeichnis

# 1 Problembeschieb

Für das Projekt 5 von Frau Escher wird zur Erkennung von Türen eine Bildverarbeitung benötigt. Diese möchten wir in diesem Projekt implementieren. Dafür werden von verschiedenen Türen Bilder aufgenommen. Der Algorithmus soll mittels Edge Detection erkennen, ob eine Türe im definierten Sichtfeld vorhanden ist. Falls ja, sollte es möglich sein die Türe über mehrere Bilder zu verfolgen und die Türbreite bestimmen. Der Algorithmus wird in Python geschrieben.

# 2 Algorithmus

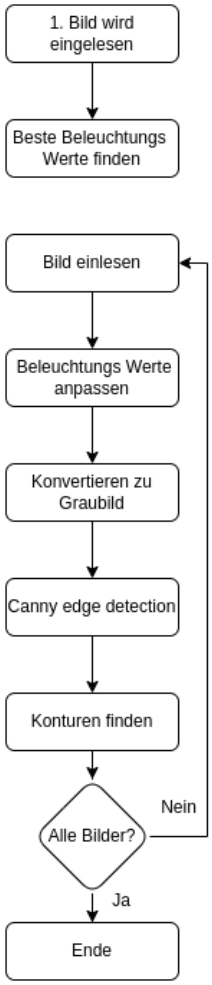



**Abbildung 2.1:** Algorithmus Ablauf

Der Algorithmus startet mit dem Einlesen eines Bildes, an welchem es die Belichtungswerte anpasst und die besten aussucht mittels des Verfahrens der Strukturellen Similarität. Die beiden alpha und beta Werte, welche die beste Belichtung liefern, werden dann für alle Bilder in der Datenbank übernommen[1, 1. Antwort].

Danach fängt die Software an, jeweils ein Bild einzulesen.

Mittels den vorher bestimmten Beleuchtungswerten wird das Bild nun aufgebessert[1, 1. Antwort].

Nun wird das bessere Bild in ein Graustufenbild verwandelt.

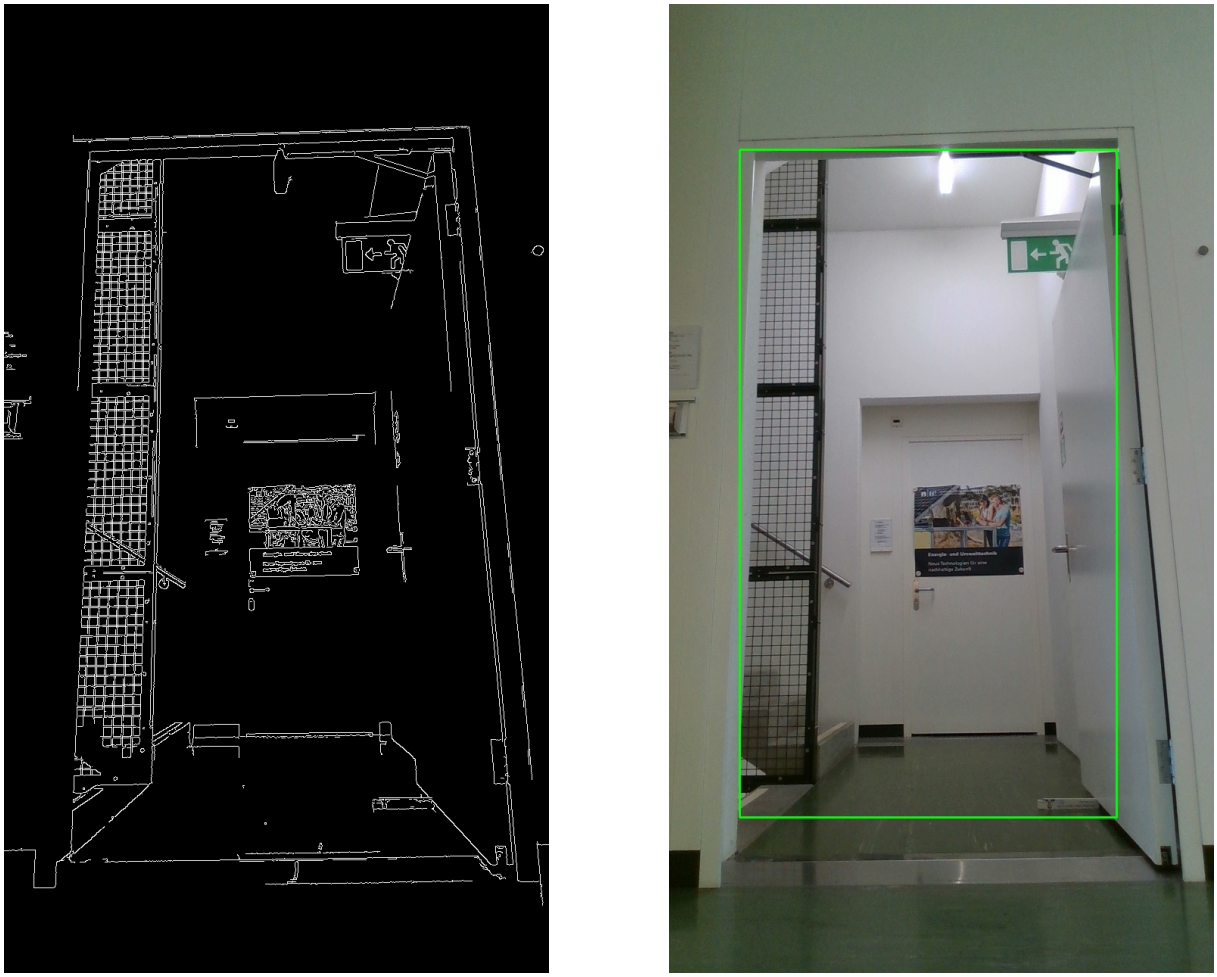
Auf dieses Graustufenbild wird nun der Canny Edge Detector losgelassen. Die gefundenen Kanten werden dann im edges Bild gespeichert[2].

Die gefunden Kanten werden nun mit der Funktion *findContours* auf Rechtecke untersucht. Danach werden alle gefundenen Rechtecke angeschaut, ob sie:
1. Zur höchsten Hierarchie gehören. Das heisst, dass sie ein Rechteck sind, welches kein grösseres Rechteck haben, welches es komplett umschliesst.
2. Höher als breit sind. Dies ist notwendig, da Türen, die gebraucht werden können, immer Hochkant aufgestellt sind.
3. Höher und breiter als der Minimalwert von 350 bezw. 250 Pixel sind. Dies ist nötig, um kleiner Rechtecke auszuschliessen und somit Fehldetektionen zu verhindern.

Sind diese drei Anforderungen erfüllt, dann wird das Rechteck auf dem Ursprungsbild grün eingezeichnet und abgespeichert. Danach wird im Terminal "Door Detected" ausgegeben.
Sind die Anforderungen nicht erfüllt steht im Terminal "Door Not Derected".
Danach wird das nächst Bild eingelesen und der Vorgang beginnt von vorne, bis alle Bilder verarbeitet wurden[3].

## 2.1 Resultate

Die Resultate sind sehr durchzogen nur etwa die Hälfte der Bilder werden erkannt. Wenn jedoch der Canny Edge Detector richtig funktioniert, dann kann eine Türe sehr gut erkannt werden 2.2. Aber meistens ist das Kantenbild nicht exakt genug, damit die Rechteck Erkennungsfunktion ihren Job machen kann. Dies führte zu mehreren alternativen Ansätze, welche jedoch auch nicht die Anforderungen erfüllen. Dazu im Abschnitt 3.



**(a)** Kanten Bild **(b)** Erkannte Türe

**Abbildung 2.2:** Vom Kantenbild mittels Rechteckerkennung zur erkannten Türe

## 2.2 Stärken/Schwächen

Die Türerkennung ist noch relativ unzuverlässig. Die Türe wird in ca. 50% erkannt. Gründe dafür sind keine guten Kantenbilder oder perspektivische Verzerrungen der Türe. Weiter ist die Höhe und Breite des gesuchten Rechteckes festgelegt. Ist die Türe zu klein auf dem Bild, wird sie nicht erkannt.

Wird der Algorithmus mit Beleuchtungsanpassung durchgeführt, dann ist die Runtime bis zu einer Minute lang.

Es ist manchmal schwierig, die richtigen Parametereinstellungen zu finden. Eine Einstellung funktioniert gut für ein Bild, aber für ein anderes nicht.

Ein grosser Vorteil dieses Algorithmus ist, dass selbst wenn die Türe offen ist und es in der Türe irgend welche Kanten gibt, wie in 2.2 zu sehen ist, die Türe selbst erkannt werden kann.

Auf der Konsole wird zu jedem Bild ausgegeben, ob eine Türe erkannt wurde oder nicht.

Ein Fenster wird nicht als Türe erkannt.

# 3 Weitere Ansätze

## 3.1 Verbesserungen

Die Kantenbilder müssen verbessert werden. Bei den Bildern, welche schlechte Kantenbilder liefern, ist der Kontrast sehr niedrig. Man könnte den Kontrast verbessern.

Zur Verbesserung könnte man Machine Learning zu Hilfe nehmen. Ein trainiertes Neuronales Netz könnte mittels Klassifikation überprüfen, ob die Türe richtig erkannt wurde.

Es war geplant, den Türerkennungsalgorithmus auszubauen, so dass die erkannte Türe über mehrere Frames getrackt werden kann. Da die vorhandene Türerkennung aber zu wenig zuverlässig funktioniert hat, wurde dies nicht umgesetzt.
Bei einem Tracking Algorithmus wird dem erkannten Objekt eine ID zugewiesen, und die Position der Türe kann über mehrere Bilder verfolgt werden. Umsetzen kann man dies zum Beispiel mit einem Kalman-Filter. Dieser kann Bewegungsinformationen verwenden und die Position auf dem nächsten Bild vorhersagen. [4]

## 3.2 Andere Ansätze

Es gibt noch weitere Varianten, aus Kantenbildern eine Tür zu erkennen. Eine davon ist der Hough Line Detector. Mittels der OpenCV Funktion *HoughLines* oder *HoughLinesP* können Linien im Kantenbild erkannt werden. Ein Beispiel für HoughLines ist in Abbildung 3.1 dargestellt. Linien, welche nahe beieinander liegen können zusammengefasst werden. In den verbleibenden Linien wird nun durch Bestimmen der Schnittpunkte ein eckiges U gesucht, welches die richtige/erwartete Fläche oder Seitenverhältnisse aufweist. Dieser Ansatz wurde aus Zeitgründen nicht weiter umgesetzt. [5]

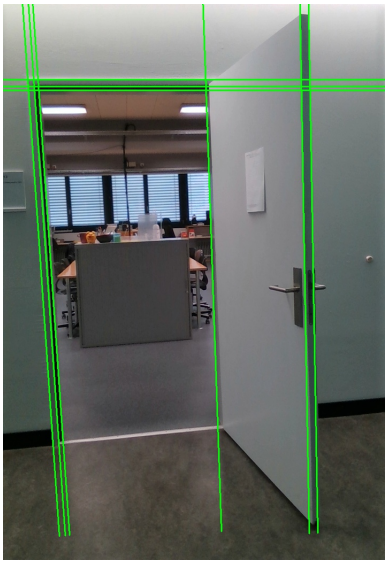


**Abbildung 3.1:** Linienerkennung mit der Hough-Transformation

# A Python Packages

| Packet | Version | Verwendung | Lizenz |
|---|---|---|---|
| Python | 3.10.6 | | |
| CV2(OpenCV) | 4.6.0.66 | CV2 ist eine open-source Bibliothek mit mehreren hunderten Computer Vision Algorithmen. | Apache 2 |
| numpy | 1.21.5 | Grösste Python Bibliothek für Mathematische Funktionen | Copyright (c) 2005-2023, NumPy Developers. |
| skimage (scikit-image) | 0.19.3 | Bild Verarbeitungsbibliothek für Python | Copyright (C) 2019, the scikit-image team |

**Tabelle A.1:** Python Bibliotheken

## Quellenverzeichnis

[1] Basj, *Automatic contrast and brightness adjustment of a color photo of a sheet of paper with OpenCV*, Forum post, Sep. 2021. Adresse: `https://stackoverflow.com/q/56905592` (besucht am 18. Jan. 2023).

[2] *OpenCV: Canny Edge Detection*. Adresse: `https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html` (besucht am 18. Jan. 2023).

[3] *How to Detect Shapes in Images in Python using OpenCV?*, en-us, Section: Python, Jan. 2021. Adresse: `https://www.geeksforgeeks.org/how-to-detect-shapes-in-images-in-python-using-opencv/` (besucht am 18. Jan. 2023).

[4] R. Sadli, *Object Tracking: Simple Implementation of Kalman Filter in Python*. Adresse: `https://machinelearningspace.com/object-tracking-python/`.

[5] OpenCV, *Hough Line Transform*, accessed: 15.012.2022. Adresse: `https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html`.