1 Angabe Erstellung eines Koordinatensystems und Objektlokalisierung

Erstellen Sie ein digitales Koordinatensystem und verknüpfen Sie gefundene Objekte mit der kalibrierten Aufnahme.

Die dritte Aufgabe ist in zwei Teile geteilt. Im ersten Teil der Übung werden Sie Ecken erkennen. Im zweiten Teil werden Sie ein Verhältnis errechnen, um Abstände zu bestimmen. Für den ersten Teil der dritten Aufgabe benötigen Sie ein kalibriertes Bild, auf welchem bereits Objekte gefunden wurden. Dazu schließen Sie zuerst Aufgabe eins und zwei ab. Wenn Sie diese erledigt haben, lesen Sie ein Bild mit der Funktion `cv2.imread("bild.png")` ein und konvertieren Sie dieses in den Graustufen-Farbraum mit `cv2.cvtColor(bild, cv2.COLOR_BGR2GRAY)`. Zusätzlich konvertieren Sie das Bild mit der Funktion `numpy.float32(bild)` in das Fließkomma 32-Bit Format, da die nachfolgende Funktion nur mit diesem Format arbeiten kann.

Für die nächsten Schritte benötigen Sie einige Hilfsvariablen. Sie werden in dieser Übung den Arbeitsraum des Puzzlebots in einem Bild finden, und diesen durch bekannte Abmessungen beschreiben. Dazu erkennen Sie alle Ecken des Arbeitsraums wie in Abbildung 1 dargestellt ist. Diese Ecken müssen in einem Array zwischengespeichert werden. Erstellen Sie ein Array in welchem alle gültigen Ecken abgespeichert werden können.

Mit der Funktion `cv2.cornerHarris(bild, Parameter Blockgröße, K-Größe, Parameter k)` können Ecken gefunden werden. Die Funktion gibt Ihnen ein Bild mit Werten für die Wahrscheinlichkeit einer Ecke zurück. Der Parameter Blockgröße beschreibt die Nachbarschaft (benachbarte Pixel), welche untersucht werden soll. Die K-Größe wird an die Funktion `sobel()` weitergeleitet und beschreibt den Sobel-Operator, welcher für die Berechnung der Bildableitung zuständig ist. Der Parameter k ist ein freier Parameter, welcher die Genauigkeit der Eckenerkennung beeinflusst.

Hinweis: Setzen Sie die Blockgröße auf 2 und die K-Größe auf 3. Probieren Sie für den Parameter k verschiedene Einstellungen aus.

Im nächsten Schritt bearbeiten Sie das Bild der `cv2.cornerHarris()` Funktion Pixel für Pixel. Erstellen Sie dazu zwei Schleifen. Legen Sie einen Bereich fest, in welchem Ecken als gültig erachtet werden sollen. Grenzen Sie dabei den Bereich möglichst eng ein. Allerdings sollte auch bei einer wiederholten Kameraaufnahme vom Puzzlebot (mit einer möglicherweise minimalen Änderung) der Bereich noch als gültig gesehen werden. Legen Sie einen Parameter für

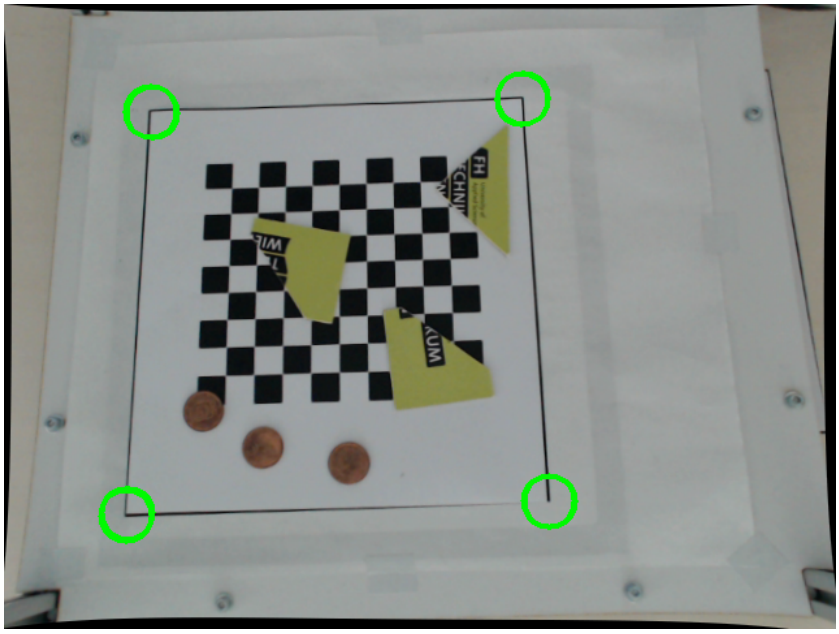

Abbildung 1: Ecken des Arbeitsbereichs

die Wahrscheinlichkeit einer Eckenerkennung fest. Alle Werte, welche über diesem Wert liegen werden noch als gefundene Ecke erkannt.

Speichern Sie schließlich alle Werte in eine Liste mit der Funktion `Liste.append(Werte)` ab.

Hinweis: Überlegen Sie sich wie Sie mit doppelten Werten oder Werten, die sehr nahe beieinander liegen umgehen möchten. Sie benötigen pro Ecke genau einen Wert, welcher diese beschreibt. Die Eckenerkennung kann durch richtiges Einstellen der Parameter sehr gut funktionieren, allerdings wird oft auch eine Ecke vier oder fünf Mal erkannt. Mit der Funktion `cv2.circle()` können Sie die gefundenen Ecken mit Kreisen grafisch markieren.

Für den zweiten Teil der Übung benötigen Sie die Schwerpunktkoordinaten einer erkannten Münze oder eines erkannten Puzzleteils der Aufgabe 2. Zuerst suchen Sie sich zwei Ecken aus, von denen Sie den Abstand zu einander kennen.
Der Arbeitsraum des Puzzlebots ist mit $215mm$ in X $\cdot 225mm$ in Y bekannt. Sie können sich also aussuchen, ob Sie den Abstand auf der X-Achse oder der Y-Achse berechnen. Berechnen Sie den Abstand der zwei Ecken (Punkte) zueinander in Pixel.

Hinweis: Der Ursprung des Koordinatensystems liegt in der rechten oberen Ecke.

Speichern Sie diesen Wert ab und lesen Sie als nächstes die Koordinaten des Schwerpunkts von dem gewünschten Objekt aus. Achten Sie darauf, dass Sie die X-Koordinate und die Y-Koordinate getrennt behandeln, da der Puzzlebot auf eine Eingabe von X- und Y-Koordinate programmiert ist. Mit diesen Werten können Sie nun ein Verhältnis zwischen Referenzlänge und Schwerpunktkoordinaten aufstellen. Danach erhalten Sie die Distanz in X- und Y-Richtung,

welche der Puzzlebot vom Ursprung zurücklegen muss. Diese Werte benötigen Sie in Übung 4 um den Puzzlebot gezielt zum Objekt zu lenken.

Damit haben Sie die Aufgabe 3 abgeschlossen. Verwenden Sie die in Kapitel 1.1 angeführten Bibliotheken und Funktion zum Programmieren der Aufgabe 3. Bitte bedenken Sie, dass Funktionen aus den Aufgaben 1 und 2 auch verwendet werden können.

1.1 Funktionen und Bibliotheken Übung 3

Bibliotheken:

- `import cv2`
- `import numpy`

Funktionen:

- `numpy.float32()` Erstellen eines 32bit-float Datentyps
- `append()` Element einer Liste hinzufügen
- `cv2.cornerHarris()` Erkennen einer Ecke mit Harris edge detector