Digitale Bildverarbeitung
Füllstandsmessung mit OpenCV

Mustafa Kilci
Levent Esen

17. Juli 2023

1 Inhalt

2 Einleitung

Die folgende Dokumentation beschäftigt sich mit dem Projekt zur Füllstandsmessung. Das Ziel ist es hierbei, den Füllstand eines Behälters zu erfassen und diesen in einer der 5 folgenden Klassen einzuordnen: '0', '25', '50', '75', '100'. Die Klassen repräsentieren den Füllstand des Behälters.

Die Füllstandsmessung ist in vielen Bereichen und Anwendung nützlich. Einer dieser Bereiche sind zum Beispiel industrielle Prozesse oder Verkehrssysteme.

In der Industrie wird die Füllstandsmessung verwendet, um den Füllstand von Flüssigkeiten oder Schüttgut in Tanks, Behältern oder Silos zu überwachen. Dadurch kann die Produktion optimiert werden, indem rechtzeitig Nachschub oder Entleerung erfolgt, um Engpässe oder Überfüllung zu vermeiden.

In Verkehrssystemen wird durch Füllstandsmessung der Verkehr auf Straßen, Parkplätzen oder Brücken überwacht. Dies ermöglicht die Bereitstellung von Echtzeitinformationen über Parkplatzverfügbarkeit, Verkehrsfluss und Vermeidung von Überlastungen.

Die Notwendigkeit, genaue Informationen über den aktuellen Füllstand in Echtzeit zu erhalten, motiviert die Entwicklung eines zuverlässigen Füllstandsmessungssystems. Oftmals erfüllen herkömmliche manuelle Methoden oder visuelle Schätzungen nicht die Anforderungen an Genauigkeit und Effizienz. Daher ist die Entwicklung einer automatisierten Lösung von entscheidender Bedeutung, die den Füllstand zuverlässig und präzise erfasst.

3 Implementierung

```
1  import glob
2  import imutils
3  import cv2
4  import itertools
```

Die ersten Zeilen importieren die erforderlichen Bibliotheken: glob, imutils, cv2 (OpenCV), itertools und numpy. Die Bibliothek „glob“ ermöglicht die Verwendung von Dateimustern, um Dateinamen zu durchsuchen. „imutils“ enthält verschiedene Funktionen zum Bearbeiten von Bildern, während „cv2“ OpenCV-Funktionen bereitstellt. „itertools“ stellt eine Sammlung von Funktionen bereit, die bei der Erzeugung und Verarbeitung von Iteratoren und Iterationen hilfreich sind und numpy bietet Funktionen zur effizienten Handhabung von numerischen Arrays und Matrizen.

-

```python
# (Quelle: https://www.tutorialkart.com/opencv/python/opencv-python-resize-image/)
def scale(img):
    scale_perc = 15
    width = int(img.shape[1] * scale_perc / 100)
    height = int(img.shape[0] * scale_perc / 100)
    dim = (width, height)
    resized = cv2.resize(img, dim, interpolation=cv2.INTER_AREA)
    return resized
```

Die Funktion "scale" wird definiert. Sie akzeptiert ein Bild als Eingabe und skaliert es auf 15% der ursprünglichen Größe. Dazu wird die Funktion "cv2.resize" verwendet, diese skaliert das Bild auf die angegebenen Abmessungen. "img" ist das Eingangsbild, "dim" ist ein Tupel, das die Breite und Höhe der skalierten Version des Bildes angibt, und "interpolation" ist die Methode, die zur Interpolation verwendet wird. Im vorliegenden Code wird "cv2.INTER_AREA" verwendet, um das Bild abzusenken. Das Ergebnis ist ein skaliertes Bild, das zurückgegeben wird.

-

```python
path = glob.glob("C:/Users/mkilc/Desktop/Flaschendatensatz/test/0/*.JPG")
path2 = glob.glob("C:/Users/mkilc/Desktop/Flaschendatensatz/test/25/*.JPG")
path3 = glob.glob("C:/Users/mkilc/Desktop/Flaschendatensatz/test/50/*.JPG")
path4 = glob.glob("C:/Users/mkilc/Desktop/Flaschendatensatz/test/75/*.JPG")
path5 = glob.glob("C:/Users/mkilc/Desktop/Flaschendatensatz/test/100/*.JPG")
```

Es werden fünf Variablen definiert: path, path2, path3, path4 und path5. Jede Variable enthält den Pfad zu einem bestimmten Ordner, der Bilder enthält. Um den Traindatensatz zu verwenden, kann man einfach den Abschnitt "test" zu "train" ändern.

Die Funktion "glob.glob" ermöglicht das Durchsuchen von Dateisystemen nach Dateinamen, die einem bestimmten Muster entsprechen. Sie wird verwendet, um die Pfade zu den Bildern im angegebenen Ordner zu erhalten.

-

```python
table = np.zeros((6, 5), dtype=int)
```

Hier wird mithilfe von "numpy" eine tabelle aus 6 Zeilen und 5 Spalten und mit dem gewünschten Datentyp "int" erstellt.

-

```python
for file in itertools.chain(path, path2, path3, path4, path5):
```

Eine Schleife wird gestartet, um durch alle Dateien in den Pfaden zu iterieren. Die Funktion „itertools.chain“ wird verwendet, um alle Pfade in einer einzigen Schleifeniteration zu kombinieren.

-

```python
    image = cv2.imread(file)

    # down scale
    result = scale(image)
    img_gray = cv2.split(result)[2]
```

In jeder Iteration wird das aktuelle Bild mit Hilfe von "cv2.imread" eingelesen und in der Variablen "image" gespeichert.

-

Das Bild wird mithilfe der zuvor definierten Funktion "scale" skaliert.

-

Das skalierte Bild wird in Graustufen konvertiert, indem die Funktion „cv2.split“ verwendet wird, um die einzelnen Farbkanäle des Bildes zu trennen, und dann der Blaukanal (Index 2) als Graustufenbild ausgewählt wird.

-

```
31      # smoothing
32      bottle = cv2.GaussianBlur(img_gray, (7, 7), 0)
```

Um das Bild zu glätten, wird ein Gauß-Filter mit einer Kernelgröße von 7x7 auf das Graustufenbild angewendet. „0“ ist die Standardabweichung, die den Grad der Unschärfe steuert. Dies wird erreicht, indem die Funktion „cv2.GaussianBlur“ verwendet wird.

-

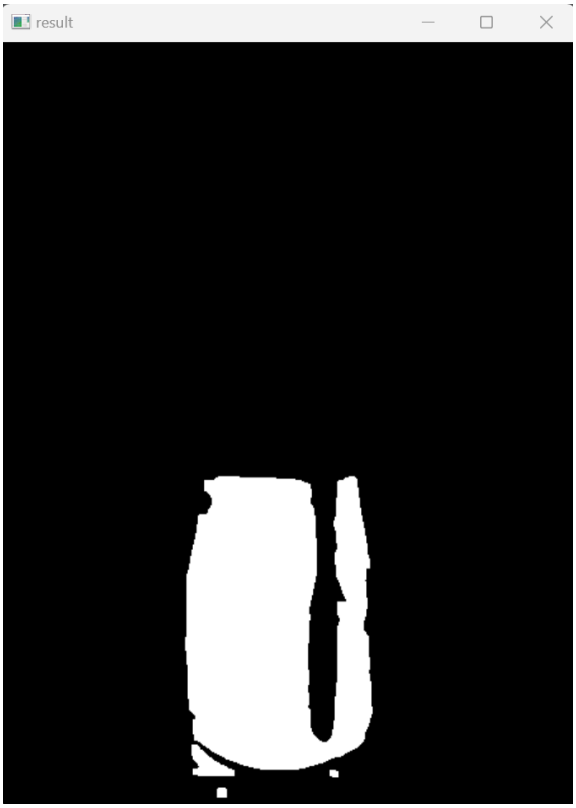
```
    # threshold of 20 was picked and inverted binary image was created
    (T, img_threshold) = cv2.threshold(bottle, 20, 255, cv2.THRESH_BINARY_INV)
```

Ein Schwellenwert von 20 wird auf das geglättete Graustufenbild angewendet, um ein binäres invertiertes Bild zu erstellen. Die Funktion "cv2.threshold" wird verwendet, um den Schwellenwert anzuwenden.

-

```
    # morphological transformations helps to get rid of any shapes marked outside the liquid
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
    bottle_open = cv2.morphologyEx(img_threshold, cv2.MORPH_OPEN, kernel)
```

Um unerwünschte Formen außerhalb der markierten Flüssigkeit zu entfernen, wird eine morphologische Transformation auf das Schwellenwertbild angewendet. Ein rechteckiger Kernel der Größe 5x5 wird mit der Funktion "cv2.getStructuringElement" erstellt, und dann wird die Funktion "cv2.morphologyEx" verwendet, um die Operation "Öffnen" auf das Bild anzuwenden.



-

```
    # finds all contours of the marked liquid
    contours = cv2.findContours(bottle_open.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    contours = imutils.grab_contours(contours)
    bottle_clone = result.copy()
```

Die Funktion „cv2.findContours“ wird verwendet, um alle Konturen im geöffneten Bild zu finden. „cv2.RETR_EXTERNAL“ gibt an, dass nur die äußeren Konturen gefunden werden sollen und „cv2.CHAIN_APPROX_SIMPLE“ gibt an, dass die Konturpunkte auf eine kompakte Weise dargestellt werden sollen.

Die Konturen werden dann mithilfe der Funktion „imutils.grab_contours“ extrahiert und in der Variable „contours“ gespeichert. Eine Kopie des skalierten Bildes wird in der Variable „bottle_clone“ erstellt.

-

```
46          # areas helps to find the largest contour
47          areas = [cv2.contourArea(contour) for contour in contours]
```

Eine Liste der Flächen (areas) wird erstellt, um die größte Kontur zu finden. Dazu wird die Funktion "cv2.contourArea" auf jede Kontur in der Variable "contours" angewendet.

-

```
54      try:
55          (contours, areas) = zip(*sorted(zip(contours, areas), key=lambda a: a[1]))
56      except ValueError:
57          cv2.putText(bottle_clone, "0", (50, 50), cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 255), 2)
58          table[0, 0] += 1
59          cv2.imshow('result', bottle_clone)
60          cv2.waitKey(0)
61          continue
```

Um eine Ausnahme abzufangen, falls die Flasche leer ist und keine markierte Flüssigkeit vorhanden ist, wird ein Versuch unternommen, die Konturen und Flächen zu sortieren. Wenn dies nicht möglich ist (ValueError), wird eine entsprechende Meldung auf dem Bild „bottle_clone“ angezeigt und die Schleife wird mit der nächsten Iteration (nächstem Bild) fortgesetzt. Außerdem wird mitnotiert, wie oft die Klasse "0" vorkam

-

```
59          # print contour
60          cv2.drawContours(bottle_clone, [contours[-1]], -1, (255, 0, 0), 2)
```

Wenn die Konturen und Flächen erfolgreich sortiert wurden, wird die größte Kontur in „contours[-1]“ ausgewählt und mit der Funktion „cv2.drawContours“ auf dem Bild „bottle_clone“ gezeichnet. Der Parameter „(255, 0, 0) „ ist die Farbe der Kontur (hier Blau) und „2“ ist die Dicke der Linie.

-

```
    (x, y, w, h) = cv2.boundingRect(contours[-1])
    aspectRatio = float(w) / h
    if 0 < aspectRatio < 0.35:
        cv2.rectangle(bottle_clone, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(bottle_clone, "100", (x + 10, y + 20), cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 255), 2)
        table[4, 4] += 1
    elif 0.35 < aspectRatio < 0.5:
        cv2.rectangle(bottle_clone, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(bottle_clone, "75", (x + 10, y + 20), cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 255), 2)
        table[3, 3] += 1
    elif 0.5 < aspectRatio < 0.65:
        cv2.rectangle(bottle_clone, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(bottle_clone, "50", (x + 10, y + 20), cv2.FONT_HERSHEY_PLAIN, 1, (0, 255, 0), 2)
        table[2, 2] += 1
    elif 1 < aspectRatio:
        cv2.rectangle(bottle_clone, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(bottle_clone, "25", (x + 10, y + 20), cv2.FONT_HERSHEY_PLAIN, 1, (0, 255, 0), 2)
        table[1, 1] += 1
```

Ein Begrenzungsrechteck (bounding rectangle) wird um die größte Kontur gezeichnet, und das Seitenverhältnis (aspectRatio) des Rechtecks wird berechnet.

Die Funktion "cv2.rectangle" zeichnet ein Rechteck um die ausgewählte Kontur. (x, y) sind die Koordinaten der oberen linken Ecke des Rechtecks, (x + w, y + h) sind die Koordinaten der unteren rechten Ecke des Rechtecks, (0, 255, 0) ist die Farbe des Rechtecks (hier Grün) und 2 ist die Dicke der Linie.

Anhand des Seitenverhältnisses wird überprüft, wie voll die Flasche ist. Die Werte für die Überprüfung der Seitenverhältnisse wurden manuell getestet und ausgewählt. Entsprechend dem Seitenverhältnis werden ein Rechteck und ein Text mit dem Füllstand auf dem Bild "bottle_clone" gezeichnet.

Die Funktion "cv2.putText" fügt dem Bild einen Text hinzu. "bottle_clone" ist das Ausgabebild, "100" ist der Text, (x + 10, y + 20) sind die Koordinaten des Textes, cv2.FONT_HERSHEY_PLAIN ist der verwendete Schriftstil, 1 ist die Skalierung des Textes, (0, 0, 255) ist die Farbe des Textes (hier Rot) und 2 ist die Dicke der Linie.

Außerdem wird mitnotiert, wie oft die Klasse "25", "50", "75" und "100" vorkam, um sie am ende in der Tabelle anzeigen zu können.

-

```
78      cv2.imshow('result', bottle_clone)
79      cv2.waitKey(0)
80
81  cv2.destroyAllWindows()
```

Das Bild "bottle_clone" wird mit der Funktion "cv2.imshow".

Die Funktion "cv2.waitKey(0)" wartet auf das Drücken einer Taste, bevor sie fortgesetzt wird. Die Zahl 0 gibt an, dass sie auf unbestimmte Zeit wartet.

Nachdem die Schleife beendet ist, werden alle offenen Fenster mit "cv2.destroyAllWindows" geschlossen.

-

```
92    print("       0   25  50  75  100")
93    print("0      " + "   ".join(map(str, table[0])))
94    print("25     " + "   ".join(map(str, table[1])))
95    print("50     " + "   ".join(map(str, table[2])))
96    print("75     " + "   ".join(map(str, table[3])))
97    print("100    " + "   ".join(map(str, table[4])))
```

Zum Schluss wird noch eine Tabelle erstellt, die die Ergebnisse anzeigt.

4 Ergebnisse

Die Tabelle, die im Code erstellt wird, zeigt nur an welche Klasse wie oft vorkam, aber nicht die Fehklassifikationen, daher zeigen wir im Ergebnisteil eine eigene erstellte Tabelle.

Ergebnis für Test-Datensatz

Test	0	25	50	75	100
0	4	0	0	0	0
25	0	4	0	0	0
50	0	0	4	0	0
75	0	0	0	4	0
100	0	0	0	2	2

Im Test-Datensatz gab es nur 2 Fehldetektionen von 20 Bildern, damit hätten wir eine Messgenauigkeit von 90%.

Ergebnis für Train-Datensatz

Train	0	25	50	75	100	n/a
0	12	3	0	0	0	1
25	0	11	0	0	0	5
50	0	0	6	0	0	10
75	0	0	1	15	0	0
100	0	0	0	8	8	0

Im Train-Datensatz gab es 28 Fehldetektionen von 80 Bildern, damit hätten wir eine Messgenauigkeit von 65%. Die Fehldetektionen bestehen diesmal nicht nur aus falsch erkannten Klassen, sondern auch aus gar keinen Erkennungen.

5 Evaluation der Ergebnisse

5.1 Train-Datensatz

Im Train-Datensatz kam es zu mehr Fehldetektionen, dafür gab es verschiedene Gründe.

Hier sieht man, dass oben noch ein Muster erkannt wird, weil die Farbe des Musters auch zu dunkel ist, dadurch wird eine Fläche erkannt und die "0" Klasse wird übersprungen.

-

In dieser Fehldetektion kam es zu keiner Einordnung. Es werden die obigen Muster wieder erkannt und keiner der erkannten Flächen haben ein passendes Seitenverhältnis, für die "if"-Fälle (aspectratio). Deswegen gibt es auch keine Zuordnung.

-

Hier genau derselbe Fall, das berechnete Seitenverhältnis ist 0.8548387096774194 und liegt im keinen der "if"-Fälle (aspectratio). Deswegen hier auch keine Zuordnung.

Die restlichen Fehldetektionen sind dieselben wie oben schon gezeigt. Bei der Klasse "100" gab es noch eine Fehlklassifizierung, die auch im Test-Datensatz besteht.

5.2 Test-Datensatz

Beide Fehldetektionen im Test-Datensatz sind identisch. Wie zu sehen, ist die Lichteinstrahlung auf der Wasserflasche erkennbar. Dies verhindert beim Schwellenwertprozess die komplette Flüssigkeit zu erkennen.

Im Allgemeinen verlief der Test-Datensatz besser, da es weniger Details in den Bildern gab. Mehr dazu in den Schwierigkeiten.

5.3 Schwierigkeiten und mögliche Lösungen

Wie in der Evaluation der Ergebnisse zu sehen ist, gibt es mehrere wichtige Faktoren, die bei der Füllstandsmessung eine große Rolle spielen.

Der Blickwinkel auf den Behälter spielt eine große Rolle für die Berechnung des Seitenverhältnisses, um den Füllstand in eine Klasse einordnen zu können. Ist der Blickwinkel zu hoch oder zu niedrig, kann es dazu führen, dass mehr oder weniger Flüssigkeit erkannt wird, was zu eine Fehldetektion führen kann.

Die Beleuchtung, Lichteinstrahlung oder das Sonnenlicht kann dazu führen, dass bei der binären Inversion eines Bildes, das blaue Wasser im Behälter nicht komplett invertiert wird. Der Grund dafür ist, dass die Beleuchtung oder Einstrahlung zu hell ist. Eine Lösung hierfür wäre es, entweder die Sättigung des kompletten Bildes oder nur des gewünschten Bereiches, also da wo Einstrahlungen zu sehen sind, zu ändern. Eine weiter Lösung wäre es, die Farbe der Einstrahlung im Bild zu finden und diese zu verdunkeln oder mit der Farbe der Flüssigkeit im Behälter zu füllen.

Ein weiteres Problem sind Muster oder Linien, die sich im Hintergrund befinden. Falls diese auch noch dunkler sind, können sie auch als Fläche erkannt werden, was zu einer Fehldetektion führen kann. Eine Lösungsmöglichkeit wäre ein stärkerer "blur" Effekt, also den Wert im "cv2.GaussianBlur(img_gray, (7, 7), 0) " zu erhöhen. Dies würde dazu führen, dass noch mehr Details im Hintergrund verschwinden, es könnte aber auch dazu führen, dass die Flüssigkeit auch nicht erkannt wird, da das Bild vielleicht zu sehr trübe wäre. Eine weitere Möglichkeit wäre, das Bild im Vorhinein zu bearbeiten und mögliche Schwierigkeiten zu beseitigen.

6 Anwendung auf fremden Datensatz

Bei der Anwendung auf einen fremden Datensatz sind ähnliche Probleme aufgetreten, im Großen und Ganzen lief die Füllstandsmessung aber trotzdem erfolgreich.

Bei der Klasse "0" kam es bei allen Bildern zu derselben Fehldetektion. Da der Deckel schwarz wird er in unserem Code als Fläche erkannt. Die Lösung hierfür wäre im "if"-Fall für Klasse "25" anzupassen und alles kleiner dem Seitenverhältnis von "25" als "0" einzuordnen.

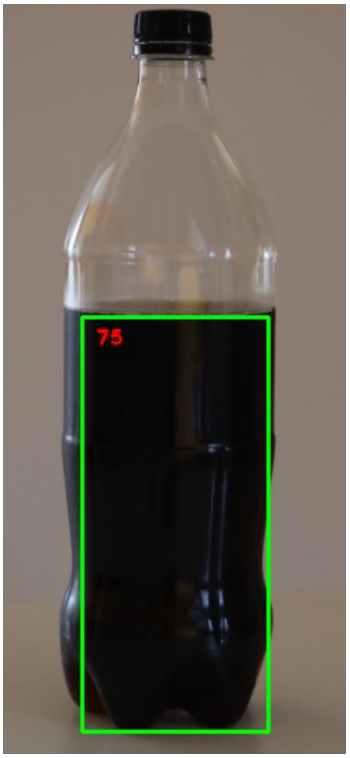
-

Bei der Klasse "25" gab es keine Fehldetektionen auf allen Bildern.

-

Bei der Klasse "50" gab es auch keine Fehldetektionen auf allen Bildern.

-



Bei der Klasse "75" gab es auch keine Fehldetektionen auf allen Bildern.

-

Bei der Klasse "100" gab es nur eine richtige Detektion, der Rest wurde als "75" eingeordnet. Die Fehldetektion tritt dadurch auf, dass unser Code manuell an das Seitenverhältnis unserer Flasche angepasst ist. Jedoch ist dies schnell gelöst, in dem man in der Berechnung des Seitenverhältnisses für die Klasse "100" den "if"-Fall anpasst.

-

Im Allgemeinen verlief die Anwendung auf einen fremden Datensatz sehr gut. Würde man in der Berechnung des Seitenverhältnisses den "if"-Fall für die Klassen "0" und "100" anpassen, kommt auf eine hundertprozentig korrekte Detektion. Das liegt daran, dass beim fremden Datensatz weniger Details im Hintergrund oder keine Einstrahlungen vorhanden sind, die die Füllstandsmessung erschweren.

7 Schlussfolgerung

Im Allgemeinen lief das Projekt wie geplant und aus unserer Sicht erfolgreich. Die Fehldetektionen waren erwartet, da auch beim Fotografieren der Bilder uns bewusst war, das gewisse Muster oder Linien im Hintergrund sind, die die Füllstandsmessung erschweren würden. Bei der Anwendung auf einen fremden Datensatz kann man auch sehen, dass die Füllstandsmessung schon ohne Codeanpassung gut funktioniert und nur individuell and das Bild adaptiert werden muss.

8 Quellen

Resize Funktion: https://www.tutorialkart.com/opencv/python/opencv-python-resize-image/

glob: https://docs.python.org/3/library/glob.html

itertools.chain: https://stackoverflow.com/questions/35205162/iterating-over-two-lists-one-after-another

opencv-Funktionen: https://docs.opencv.org/4.x/index.html