Modellbasierte Detektionsverfahren

Überblick

Das Ziel dieser Übung ist es, erste Detektionsverfahren zu implementieren. Die Verkehrsschilder der Kategorien „mandatory“ sollen modellbasiert gefunden werden und deren Leistungsfähigkeit evaluiert werden. Im Jupyter Notebook UE2_Aufgaben.ipynb findet ihr Hilfsfunktionen, die Ihr für die Erledigung der Hausaufgabe benutzen könnt. Die in der ersten Übung implementierten Funktionen können ebenfalls als Basis benutzt werden.

Aufgabe 1 – Aussortieren bestimmter Verkehrsschilder

Der Benchmark-Datensatz unterscheidet mehrere Kategorien von Verkehrsschildern. Die Schilder der Kategorie „prohibitory“ sind rund, haben einen weißen Hintergrund und einen roten Rand, die meisten Schilder der Kategorie „mandatory“ sind rund, haben einen weißen Rand und .weiße Piktogramme auf einem blauen Hintergrund.

Es soll eine Funktion implementiert werden, die die gt_txt Datei übergeben bekommt und ein Ergebnis in Form eines Python-Dictionary zurückgibt.

*TIPP: Da die Aufgabe sehr rechenintensiv ist, könnt ihr zum Einstellen der Parameter der Funktion calculate_hough_cirles() die Datei **new_gt.txt** nutzen. Diese Datei hat den gleichen Aufbau wie die gt.txt, enthält aber viel weniger Bilder mit gut sichtbaren Verkehrszeichen. Die Datei new_gt.txt findet ihr im Ordner UE 2 – Durchführung.*

Das Python-Dictionary enthält die Zeilen der gt.txt-Datei (oder new_gt.txt-Dateei), die der Kategorie „mandatory“ zugeordnet werden, mit

- Dateinahmen als dict.keys() und
- Listen [leftCol, topRow, rightCol, bottomRow, ClassID] als dict.values().

Zum Anzeigen der Verkehrszeichen-Bilder aus der Python-Dictionary soll eine weitere Funktion implementiert werden. Ihr könnt die Funktion **calc_rois** aus der ersten Übungsaufgabe als Basis nehmen und entsprechend anpassen oder eine eigene Funktion implementieren.

Abbildung 1: Beispiel für ein Screenshot für Aufgabe 1

Aufgabe 2 – Formbasierter Ansatz

Zunächst wollen wir lediglich die Form als Merkmal zur Detektion verwenden. Ihr könnt die Funktion HoughCircles() nutzen, um mit Hilfe von OpenCV eine Hough-Transformation zur Kreiserkennung durchzuführen. Diese Funktion nutzt die Gradientenrichtung, um die Mittelpunkte der Kreise zu bestimmen. Daher führt sie auch die Kantendetektion selbstständig durch. Schaut euch die OpenCV-Dokumentation um herauszufinden, welche Vorverarbeitungsschritte notwendig bzw. sinnvoll sind. Beachtet auch, dass ihr die gefundenen Kreise in Rechtecke (siehe Abbildung 3) überführen müsst.

Abbildung 2: Beispielbild (1) nach der Anwendung von HoughCircles()

Abbildung 3: Beispielbild (2) nach der Anwendung von HoughCircles() und Überführung der Kreise in Rechtecke

Aufgabe 3 – Optimierung und Evaluation des formbasierten Ansatzes

Probiert verschiedene Parameterkombinationen für euren Ansatz aus.

Über welche(n) Schwellwert(e) könnt ihr die Empfindlichkeit der Detektion variieren? Nehmt für eine Parameterkombination eurer Wahl einen Precision-Recall-Plot auf, indem ihr ausschließlich den gefundenen Schwellwert verändert.

Bonus

Wendet euren Ansatz auf die Kategorie „prohibitory“ an und nehmt die Precision-Recall-Plots auf.

Tipps

1. Parametertuning nicht mit dem gesamten Datensatz, sondern anfangs mit wenigen Bildern (10-20 Bilder) durchführen. Das reduziert den Rechenaufwand.
2. Sinnvolle Radien (minRadius, maxRadius) einsetzen (entweder mit Hilfe der roi-Koordinaten berechnen oder visuell abschätzen, welche sich am besten eignen. Das verkürzt die Rechenzeit erheblich.

Abbildung 4: Beispiel von falsch eingesetzten Radien

3. Overflow-Warnung: tritt auf, wenn die Werte a, b, r vom Typ numpy.uint16 sind. Bei deren Addition kann es zum Überlauf kommen (Koordinatenwert 65535). Daher empfehle ich diese Werte für die Weiterverwendung in Integer umzuwandeln (z.B.: int(a)).

```
C:\Users\admin\Anaconda3\envs\tensorflow_env\lib\site-packages\ipykernel_launcher.py:43: RuntimeWarning: overflow enc
ountered in ushort_scalars
```

4. Die Umrechnung von Kreisen in Vierecke überprüfen (visualisieren und schauen, ob die roi-Vierecke an die richtige Stelle gemalt sind und Achsen nicht vertauscht sind). Die Vierecke an den Bildrändern checken (ob sie richtig dargestellt sind und nicht über das ganze Bild gemalt). Dieses Problem kann durch den Einsatz von „max“, „min“ und Bild-„shape“ gefixt werden.

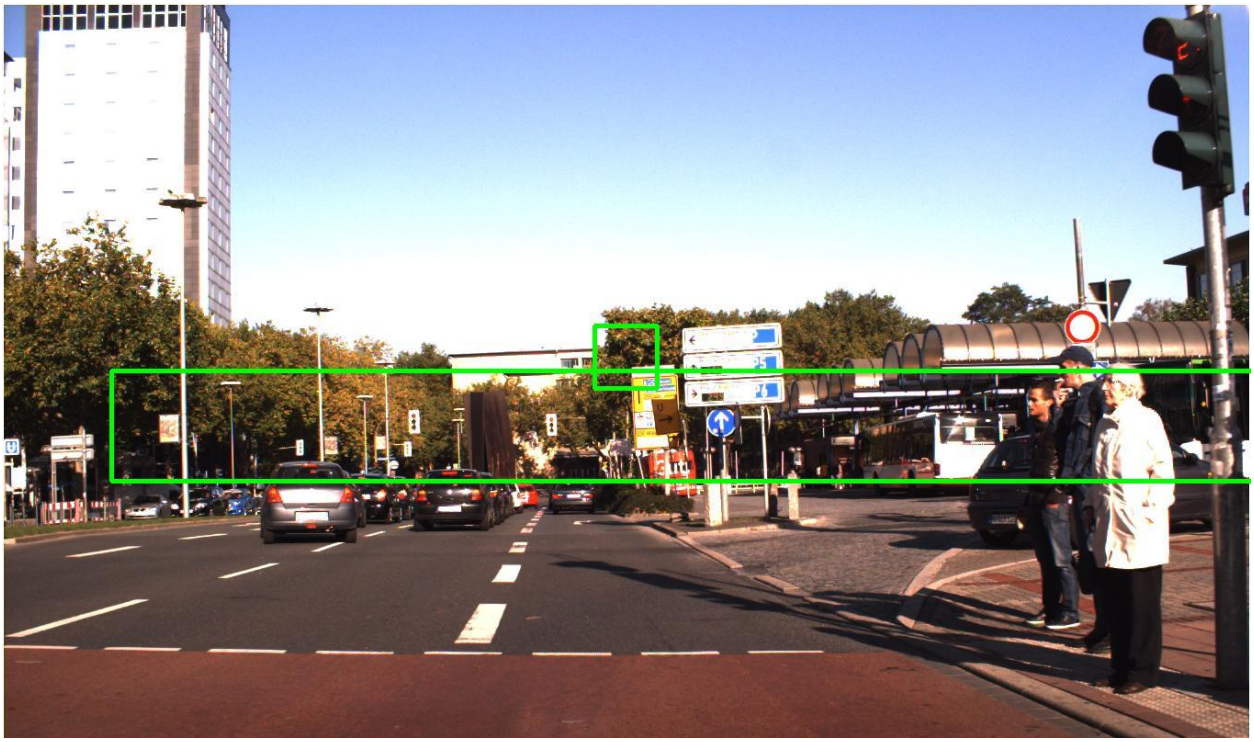

Abbildung 5: Beispiel für falsch berechnete ROI-Werte