Traffic Monitoring

# Python

## Cv2.VideoCapture()

Class for video capturing from video files, image sequences or cameras. You can pass the path of a file or an integer for an input device such as a webcam.

capture = cv2.VideoCapture("videoSamples/car\_bridge.mp4")

[…]

capture.release()

Zu jedem .VideoCapture muss auch eine entsprechendes .release() gehören, welches den Aufruf der Datei oder Kamera beendet.

\_, frame = capture.read()

.read() grabs, decodes and returns the next video frame.

r = cv2.selectROI(frame)

Öffnet ein Fenster, in welchem das frame angezeigt wird. Dort kann über die Region von Interesse ausgewählt werden, welche nachfolgend verarbeitet werden soll.

## Cv2.createBackgroundSubtractorMOG2()

Die Hintergrundsubtraktion ist ein wichtiger Bestandteil von Bildverarbeitungsprozessen. Im Prinzip wird ein sich bewegender Vordergrund von einem statischen Hintergrund extrahiert.

OpenCV stellt hierfür drei Algorithmen zur Verfügung, wovon wir den oben genannten verwenden.

Es handelt sich um einen Gaussian Mixture-basierten Hintergrund- /Vordergrund-Segmentierungsalgorithmus. Der Algorithmus verwendet eine Methode zur Modellierung jedes Hintergrundpixels durch eine Mischung von k Gauß-Verteilungen (k = 3 – 5). Die Gewichte der Mischung repräsentieren die Zeitanteile, die diese Farben in der Szene bleiben. Die wahrscheinlichen Hintergrundfarben sind diejenigen, die länger und statischer bleiben.

Eine wichtige Eigenschaft des Algorithmus ist, dass er die entsprechende Anzahl von Gauß Verteilungen für jedes Pixel auswählt. So wird eine bessere Anpassungsfähigkeit an variierende Szenen aufgrund von Beleuchtungsänderungen erreicht.

Wenn es auf das erste Bild angewendet wird, das diesem Objekt zugeführt wird, beginnt es mit der Erstellung eines Hintergrundmodells. Wenn Bilder in dieses Objekt eingespeist werden, wird der Hintergrund weiter aktualisiert. Mit Hilfe des Parameters varThreshold wird der Schwellenwert festgelegt, bei dem ein Pixel als im Hintergrund befindlich bestimmt wird. Der Wert hat keinen Einfluss auf die Aktualisierung des Hintergrundmodells. Dieser Wert ist ein Abstandsschwellenwert zwischen dem Pixel und dem Hintergrundmodell. Je höher der Schwellenwert ist, desto mehr Sicherheit ist erforderlich. Wir verwenden ihn, um anzuzeigen, dass wir nur die Pixel haben wollen, die durch das Hintergrundmodell gut beschrieben werden.

objectDetector = cv2.createBackgroundSubtractorMOG2(history=500, varThreshold=400, detectShadows=True)

history: Setzt die Nummer der vergangenen Bilder fest, die das Hintergrundmodell beeinflussen.

varThreshold: Schwellenwert für den quadrierten Abstand zwischen dem Pixel und dem Modell, um zu entscheiden, ob ein Pixel durch das Hintergrundmodell gut beschrieben ist. Dieser Parameter hat keinen Einfluss auf die Hintergrundaktualisierung.

detectShadows: Wenn true, erkennt der Algorithmus Schatten und markiert sie. Es verringert die Geschwindigkeit ein wenig.

mask = objectDetector.apply(regionOfInterest)

.apply() wendet den zuvor definierten Hiintergrundsubstraktor auf das übergebene frame an.

\_, mask = cv2.threshold(mask, 254, 255, cv2.THRESH\_BINARY)

Entfernt graue Elemente aus der Maske, wie zum Beispiel die zuvor erkannten Schatten.

cv2.imshow("Mask after preprocessing", mask)

Öffnet ein Dialogfenster, in dem das übergebene frame angezeigt wird. In eine Schleife angewendet, können so Videos angezeigt werden.

kernel = np.ones((5, 5), np.uint8)

Erstellt eine 5x5 Matrix gefüllt mit 1. Diese wird für die nachfolgende Dilatation verwendet.

dilated = cv2.dilate(mask, kernel, iterations=6)

Morphologische Transformationen sind einfache Operationen, die auf der Bildform basieren. Sie werden normalerweise an Binärbildern durchgeführt. Sie benötigen zwei Eingaben, eine ist das Originalbild, das zweite wird als strukturierendes Element oder Kernel bezeichnet, welcher die Art der Operation bestimmt. Zwei grundlegende morphologische Operatoren sind Erosion und Dilatation. Dann kommen auch ihre Varianten wie Öffnen, Schließen, Gradient usw. hinzu.

Bei der Dilatation wir das Pixelelement 1, wenn mindestens ein Pixel des Kernel 1 ist. Dadurch wird der weiße Bereich im Bild vergrößert oder die des Vordergrundobjekts erhöht.

Im Projekt werden so einzelne Flächen innerhalb des Fahrzeugs überblendet.

contours, \_ = cv2.findContours(dilated, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)

Konturen können einfach al eine Kurve erklärt werden, die alle kontinuierlichen Punkte entlang der Begrenzung verbindet, die die gleiche Farbe oder Intensität haben. Die Konturen sind ein nützliches Werkzeug für die Formanalyse und Objekterkennung.

Die Argumente sind das Quellbild, der Konturabrufmodus und die Methode der Konturannäherung. Rückgabe der Funktion ist u.a. eine Liste aller Konturen im Bild. Jede Kontur ist ein Array aus x,y-Koordinaten von Begrenzungspunkten des Objekts.

Um Rechenleistung zu sparen, kann als Annäherungsmethode auch CHAIN\_APPROX\_SIMPLE verwendet werden.

for i in range(len(contours)):  
 hull = cv2.convexHull(contours[i])  
 cv2.drawContours(mask, [hull], -1, (255, 0, 0), -1)