



HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
UNIVERSITY OF APPLIED SCIENCES

**Technische Grundlagen
der angewandten Informatik**

OpenCV - Python

J. Keppler

Konstanz, 6. November 2023

Zusammenfassung (Abstract)

Thema: OpenCV - Python

Autoren: J. Keppler

jkeppler@htwg-konstanz.de



OpenCV-Python ist ein Wrapper für Python, welche den Zugriff auf die Bibliothek *OpenCV* ermöglicht.

Ausführliche Informationen zu *OpenCV-Python* und *OpenCV* findet man im Internet unter folgenden Links:

- <http://opencv.org>
- http://docs.opencv.org/trunk/doc/py_tutorials/py_tutorials.html

Inhaltsverzeichnis

1	Einleitung	1
1.1	Installation	1
2	Referenzen	3
2.1	Funktionen	3
2.1.1	cv2.VideoCapture	3
2.1.2	cv2.cvtColor()	3
2.1.3	cv2.imshow()	4
2.1.4	cv2.waitKey()	4
2.1.5	cv2.destroyAllWindows()	4
2.1.6	cv2.destroyWindows()	4
2.2	Klassen	5
2.2.1	VideoCapture Klasse	5
	Anhang	7
A.1	Live-Bild von der Webcam	7
A.2	Einstellungen der Webcam	7

1

Einleitung

1.1 Installation

OpenCV ist eine freie Programmbibliothek mit Algorithmen für die Bildverarbeitung und maschinelles Sehen. Sie ist für die Programmiersprache C und C++ geschrieben und steht als freie Software unter den Bedingungen der BSD-Lizenz. **OpenCV-Python** ist in Modul für Python, welches eine Wrapper-Klasse für die Funktionen der Bibliothek *OpenCV* bildet.

Das **OpenCV-Python** - Modul wurde unter Windows 7 für Python 3.4 mit 64 Bit compiliert. Dazu wurde Visual Studio 2013 verwendet.

Die Installation wird in folgenden Schritten durchgeführt:

1. Das Microsoft Visual C++ 2013 Redistributable Package (x64) [vc_redist_x64.exe] wird zunächst auf dem Computer ausgeführt.
2. Die Datei cv2.pyd wird in das Unterverzeichnis <python directory>\DLLs kopiert. In diesem Verzeichnis müssen auch die passenden DLLs von OpenCV vorhanden sein.
 - opencv_calib3d300.dll
 - opencv_core300.dll
 - opencv_features2d300.dll
 - opencv_ffmpeg300_64.dll
 - opencv_flann300.dll
 - opencv_highgui300.dll

- opencv_imgcodecs300.dll
- opencv_imgproc300.dll
- opencv_ml300.dll
- opencv_objdetect300.dll
- opencv_photo300.dll
- opencv_shape300.dll
- opencv_stitching300.dll
- opencv_superres300.dll
- opencv_video300.dll
- opencv_videoio300.dll
- opencv_videostab300.dll

2

Referenzen

Die hier aufgeführten Klassen und Funktionen bilden nur einen Auszug aus der Bibliothek *OpenCV-Python*. Sie sollten für die Laborübungen in TGAI ausreichend sein.

2.1 Funktionen

2.1.1 cv2.VideoCapture

Die Funktion `VideoCapture()` stellt die Verbindung zu einem Gerät her. Dabei wird auch der Webcam-Controller geöffnet.

Definition: `cv2.VideoCapture(device)`

`cv2.VideoCapture(device)` -> `<VideoCapture object>`

2.1.2 cv2.cvtColor()

Mit der Funktion `cvtColor()` kann ein Bild in einen anderen Farbraum konvertiert werden.

Definition: `cv2.cvtColor(src, code [, dst [, dstCn]])`

`cv2.cvtColor(src, code [, dst [, dstCn]])` -> `dst`

2.1.3 cv2.imshow()

Zum Anzeigen eines Bildes kann die Funktion imshow() verwendet werden.

Definition: cv2.imshow(winname, image)

cv2.imshow(winname, image) -> None

2.1.4 cv2.waitKey()

Die Eingabe eines Zeichens in einem Fenster kann mit der Funktion waitKey() getestet werden.

Definition: cv2.waitKey([delay])

cv2.waitKey([delay]) -> retval

delay = 0, wartet unendlich

delay = x, x in Millisekunden

2.1.5 cv2.destroyAllWindows()

Mit dieser Funktion werden alle Ausgabefenster geschlossen.

Definition: cv2.destroyAllWindows()

cv2.destroyAllWindows() -> None

2.1.6 cv2.destroyWindow()

Mit dieser Funktion werden alle Ausgabefenster geschlossen.

Definition: cv2.destroyWindow(winname)

`cv2.destroyAllWindows(winname) -> None`

2.2 Klassen

2.2.1 VideoCapture Klasse

.read()

Mit der Methode `read()` werden die Daten des VideoCapture-Device gelesen.

Definition: `read([image])`

`read([image]) -> retval, image`

.release()

Die Methode `release()` werden die Daten des VideoCapture-Device gelesen.

Definition: `release()`

`release() -> None`

.get()

Mit der Methode `get()` können Eigenschaften des VideoCapture-Device gelesen werden.

Definition: `get(propId)`

`get(propId) -> retval`

propid	Beschreibung	Wertebereich
3	Frame Width	640
4	Frame Height	480
10	Brightness	0 - 255
11	Contrast	0 - 255
12	Saturation / Color Intensity	0 - 255
14	Gain	0 - 255
15	Exposure	-1 - -7
17	White Balance	0 - 10.000

.set()

Mit der Methode set() können Eigenschaften des VideoCapture-Device gesetzt werden.

Definition: get(propId, value)

get(propId, value) -> retval

propid	Beschreibung	Wertebereich
3	Frame Width	640
4	Frame Height	480
10	Brightness	0 - 255
11	Contrast	0 - 255
12	Saturation / Color Intensity	0 - 255
14	Gain	0 - 255
15	Exposure	-1 - -7
17	White Balance	0 - 10.000

Anhang

A.1 Live-Bild von der Webcam

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

from skimage.color import rgb2gray

cap = cv2.VideoCapture(0)

while(True):
    ret, frame = cap.read()
    grayscale = rgb2gray(frame)
    fig, ax = plt.subplots(figsize=(8, 4))
    ax.imshow(grayscale, cmap=plt.cm.gray)
    ax.set_title("Grayscale")
    plt.show()

cap.release()
cv2.destroyAllWindows()
```

Listing 3.1: Live image from webcam

A.2 Einstellungen der Webcam

```
import cv2

cap = cv2.VideoCapture(0)

print("frame_width:{}_".format(cap.get(3)))
print("frame_height:{}_".format(cap.get(4)))
print("-----")
print("brightness:{}_".format(cap.get(10)))
print("contrast:{}_".format(cap.get(11)))
print("saturation:{}_".format(cap.get(12)))
print("-----")
print("gain:{}_".format(cap.get(14)))
print("exposure:{}_".format(cap.get(15)))
print("-----")
print("white_balance:{}_".format(cap.get(17)))

cap.release()
```

Listing 3.2: webcam properties