

Python

Images & Processes

Inhalt

- In diesem Kursteil möchten wir uns mit verschiedenen Aspekten der Bildmanipulation beschäftigen. Dies hauptsächlich mit der Bibliothek OpenCV.
- Zusätzlich werden wir das generelle, zeitbasierte Ausführen von Python-Programmen anschauen, welches vor allem bei Routine-Aufgaben sehr hilfreich ist.

Szenario: Messi suchen



Szenario: Messi suchen

Wir möchten in einem grösseren Bild schauen, ob wir den Kopf von Messi ausfindig machen können. Und so automatisiert Daten gewinnen können.

Szenario: Messi suchen

Hierfür verwenden wir Template Matching.

Template Matching ist eine Methode zum Suchen und Finden der Position eines Template-Bildes in einem grösseren Bild.

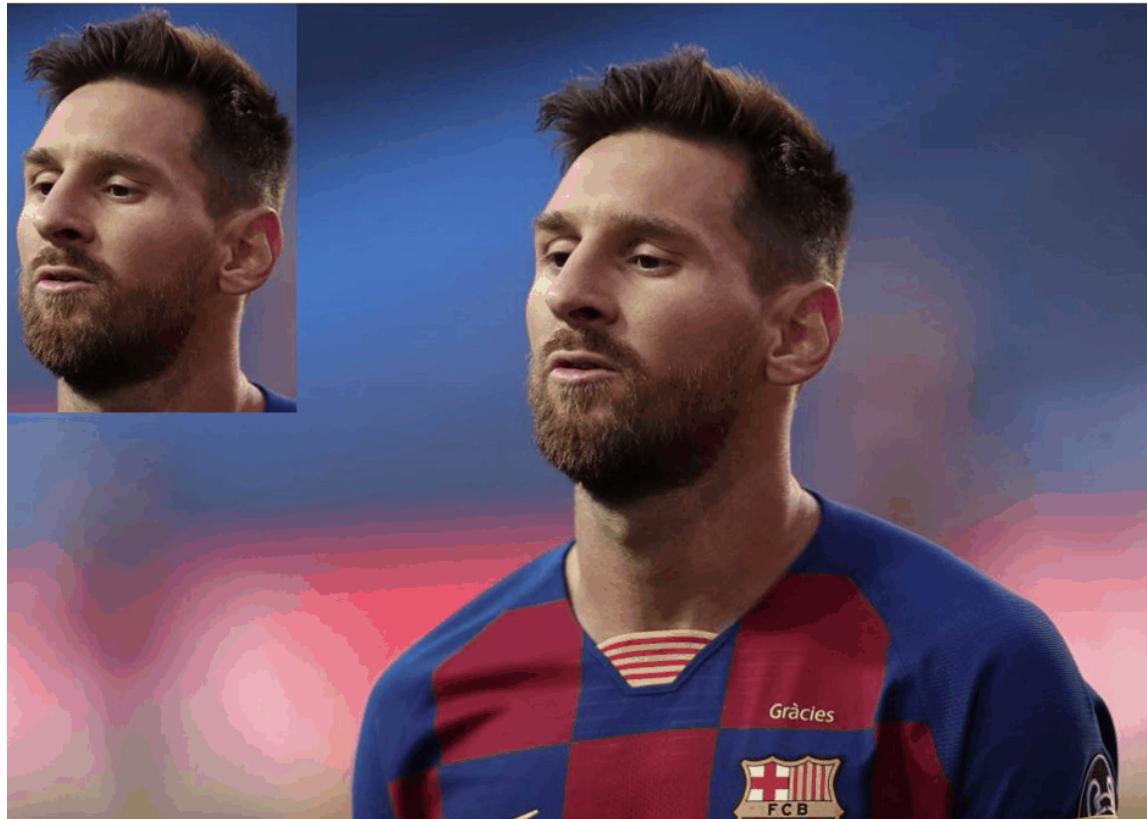
OpenCV verfügt über eine Funktion `cv.matchTemplate()` für diesen Zweck. Sie schiebt einfach das **Vorlagenbild über das Eingabebild** und vergleicht die Vorlage und den Bereich des Eingabebildes unter dem Vorlagenbild.

Szenario: Messi suchen

Es sind mehrere Vergleichsmethoden in OpenCV implementiert.
(Sie können die Dokumentation für weitere Details einsehen).

Das Ergebnis ist ein Graustufenbild, bei dem jedes Pixel angibt, **wie sehr die Umgebung dieses Pixels mit der Vorlage übereinstimmt.**

Szenario: Messi suchen



OpenCV

OpenCV (englische Abk. für Open Source Computer Vision Library) ist eine freie Programmbibliothek mit Algorithmen für die Bildverarbeitung und Computer Vision.

Sie ist für die Programmiersprachen C, C++, Python und Java geschrieben und steht als freie Software unter den Bedingungen der Apache 2 License.

Die Entwicklung der Bibliothek wurde von Intel initiiert und bis 2013 von Willow Garage gepflegt. Nach deren Auflösung wurde sie von Itseez fortgeführt, das mittlerweile von Intel übernommen wurde.

OpenCV

OpenCV besteht aus Modulen für verschiedene Anwendungsfelder:

- 2D- und 3D-Merkmale (z. B. [Interest-Operator](#) oder Deskriptoren)
- Eigenbewegungsschätzung, siehe [Photogrammetrie: Rückwärtsschnitt](#)
- [Gesichtserkennung](#), [Gestenerkennung](#)
- [Mensch-Computer-Interaktion \(HCI\)](#)
- [Mobile Roboter](#)
- [Klassifizierung mit Hilfe der Viola-Jones-Methode](#)
- [Segmentierung](#) und Erkennung
- [Stereoskopisches Sehen](#) (Stereopsis), ergibt Tiefenbilder
- [Structure from Motion](#) (SfM), siehe [Computer Vision](#)
- Optisches Tracking, [Motion Compensation](#) und [Optischer Fluss](#)
- [Kalman-Filter](#) zum Tracking

OpenCV

Ferner beinhaltet OpenCV eine Bibliothek für **Maschinelles Lernen** mit folgendem Funktionsumfang:

- **Boosting** (automatische Klassifizierung)
- Lernen eines **Entscheidungsbaumes**
- **EM-Algorithmus** (Expectation-Maximization)
- Nächste-Nachbarn-Klassifikation
- Bayes-Klassifikator
- Künstliche neuronale Netze, inkl. DNN
- Random Forest
- Support Vector Machine (SVM)

OpenCV

Ferner beinhaltet OpenCV eine Bibliothek für **Maschinelles Lernen** mit folgendem Funktionsumfang:

- **Boosting** (automatische Klassifizierung)
- Lernen eines **Entscheidungsbaumes**
- **EM-Algorithmus** (Expectation-Maximization)
- Nächste-Nachbarn-Klassifikation
- Bayes-Klassifikator
- Künstliche neuronale Netze, inkl. DNN
- Random Forest
- Support Vector Machine (SVM)

Szenario: Messi suchen



```
1 # 00_messi.py
2
3 import cv2 as cv
4 import numpy as np
5
6 # read
7 from matplotlib import pyplot as plt
8 img = cv.imread('template.jpg', cv.IMREAD_GRAYSCALE)
9 assert img is not None, "file could not be read, check with os.path.exists('template.jpg')"
10
11 # read
12 img2 = img.copy()
13 template = cv.imread('face.jpg', cv.IMREAD_GRAYSCALE)
14 assert template is not None, "file could not be read, check with os.path.exists('face.jpg')
```

Szenario: Messi suchen



```
1 # 00_messi.py
2
3 ...
4
5 # get template shape
6 w, h = template.shape[::-1]
7
8 # All the 6 methods for comparison in a list
9 methods = ['cv.TM_CCOEFF', 'cv.TM_CCOEFF_NORMED', 'cv.TM_CCORR',
10           'cv.TM_CCORR_NORMED', 'cv.TM_SQDIFF', 'cv.TM_SQDIFF_NORMED']
```

Szenario: Messi suchen



```
1 # 00_messi.py
2
3 ...
4
5 for meth in methods:
6     # copy of the image (do not destroy the original image)
7     img = img2.copy()
8
9     # run an evaluation
10    method = eval(meth)
11
12    # Apply template Matching
13    res = cv.matchTemplate(img,template,method)
14    min_val, max_val, min_loc, max_loc = cv.minMaxLoc(res)
15
16    # If the method is TM_SQDIFF or TM_SQDIFF_NORMED, take minimum
17    # some specials
18    if method in [cv.TM_SQDIFF, cv.TM_SQDIFF_NORMED]:
19        top_left = min_loc
20    else:
21        top_left = max_loc
22
```

Szenario: Messi suchen



```
1 # 00_messi.py
2
3 ...
4
5     # create the plots
6     bottom_right = (top_left[0] + w, top_left[1] + h)
7     cv.rectangle(img,top_left, bottom_right, 255, 2)
8     plt.subplot(121),plt.imshow(res,cmap = 'gray')
9     plt.title('Matching Result'), plt.xticks([]), plt.yticks([])
10    plt.subplot(122),plt.imshow(img,cmap = 'gray')
11    plt.title('Detected Point'), plt.xticks([]), plt.yticks([])
12    plt.suptitle(meth)
13    plt.show()
```

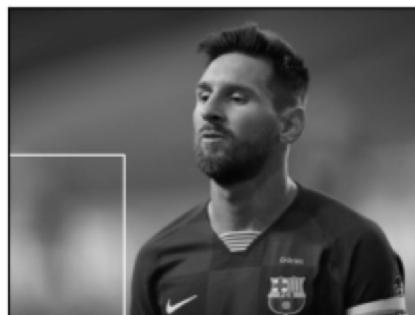
Szenario: Messi suchen

cv.TM_CCORR

Matching Result

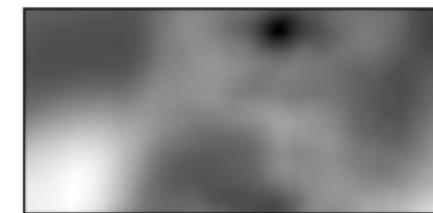


Detected Point



cv.TM_SQDIFF

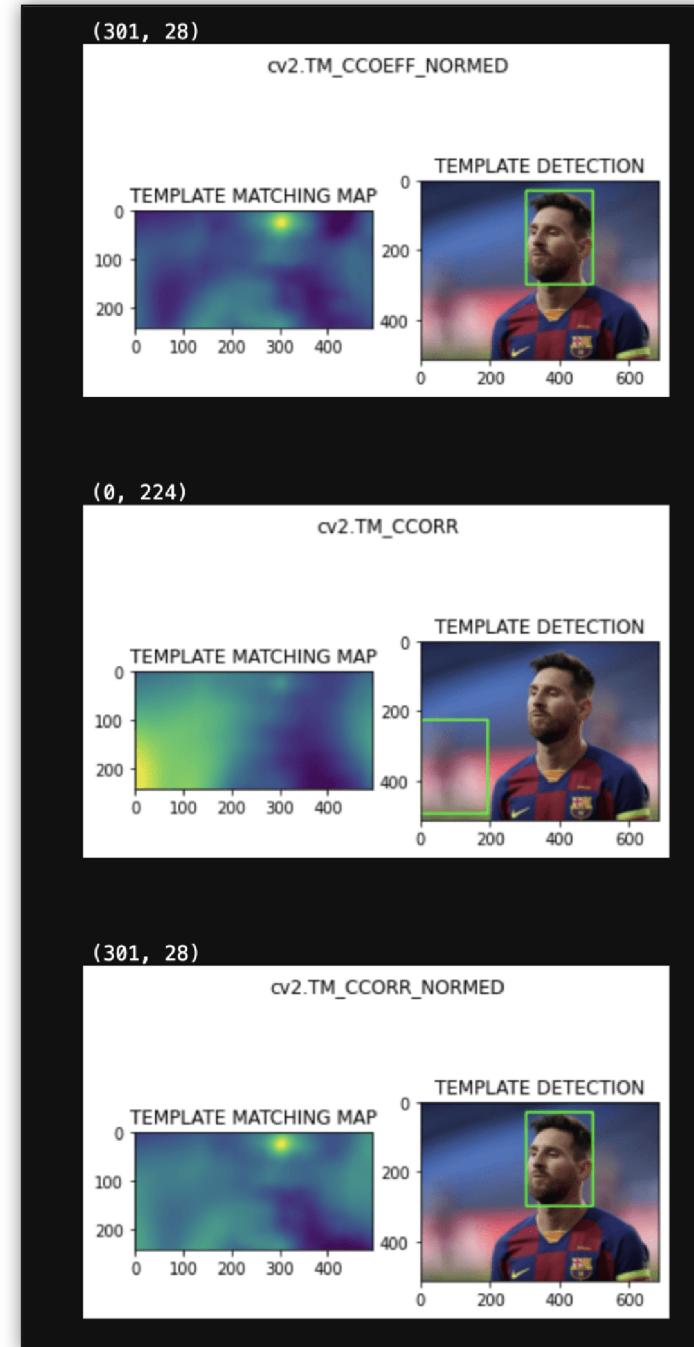
Matching Result



Detected Point



Szenario: Messi suchen



Ein Bild untersuchen



```
1 # 01_zophie.py
2
3 from PIL import Image
4 catIm = Image.open('zophie.png')
5
6 # catIm.size
7
8 width, height = catIm.size
9 #width
10 #height
11 #catIm.filename
12 #catIm.format
13 #catIm.format_description
14
15 #catIm.save('zophie.jpg')
```

Ein Bild erstellen



```
1 # 02_purple.py
2
3 from PIL import Image
4
5 im = Image.new('RGBA', (100, 200), 'purple')
6 im.save('purpleImage.png')
```



```
1 # 03_transparent.py
2
3 from PIL import Image
4
5 im2 = Image.new('RGBA', (20, 20))
6 im2.save('transparentImage.png')
```

Ein Bild ausschneiden



```
1 # 04_cropping.py
2
3 from PIL import Image
4 tuxIm = Image.open('tux.png')
5 croppedIm = tuxIm.crop((335, 345, 565, 560))
6 croppedIm.save('cropped.png')
```

Aufgabe: Ein Bild ausschneiden



```
1 # 04_cropping.py
2
3 from PIL import Image
4 catIm = Image.open('tux.png')
5 croppedIm = catIm.crop((335, 345, 565, 560))
6 croppedIm.save('cropped.png')
7
```

Das ging leider schief... versucht bitte den Kopf von Tux auszuschneiden und erklären erklärt zusätzlich, warum das Prozedere nur mit png's und nicht mit jpg's funktioniert.



Sie finden die Daten unter exercises.

Ein Ausschnitt wieder einfügen



```
1 # 05_copycopy.py
2
3 from PIL import Image
4 tuxIm = Image.open('tux.png')
5 tuxCopyIm = tuxIm.copy()
6
7 faceIm = tuxIm.crop((335, 345, 565, 560))
8 #faceIm.size
9 tuxCopyIm.paste(faceIm, (0, 0))
10 tuxCopyIm.paste(faceIm, (400, 500))
11 tuxCopyIm.save('pasted.png')
```

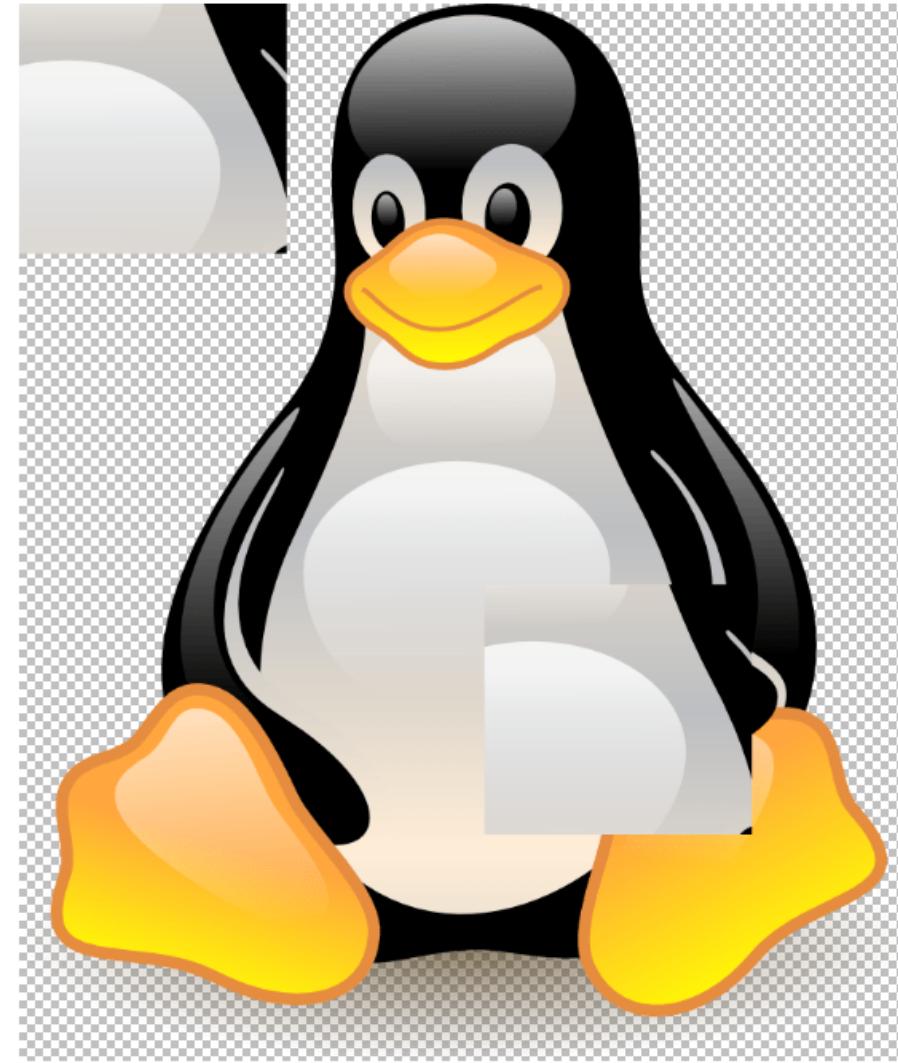


Aufgabe: Ein Ausschnitt wieder einfügen



```
1 # 05_copycopy.py
2
3 from PIL import Image
4 tuxIm = Image.open('tux.png')
5 tuxCopyIm = tuxIm.copy()
6
7 faceIm = tuxIm.crop((335, 345, 565, 560))
8 #faceIm.size
9 tuxCopyIm.paste(faceIm, (0, 0))
10 tuxCopyIm.paste(faceIm, (400, 500))
11 tuxCopyIm.save('pasted.png')
```

Beide Füße ausschneiden und oben
Links/Rechts nochmals einfügen :-)



Ein Bild rotieren



```
1 # 05_rotate.py
2
3 from PIL import Image
4 tuxIm = Image.open('tux.png')
5 tuxIm.rotate(90).save('rotated90.png')
6 tuxIm.rotate(180).save('rotated180.png')
7 tuxIm.rotate(270).save('rotated270.png')
```

Ein Bild "flippen"



```
1 # 05_rotate.py
2
3 from PIL import Image
4 tuxIm = Image.open('tux.png')
5
6 tuxIm.transpose(Image.FLIP_LEFT_RIGHT).save('horizontal_flip.png')
7 tuxIm.transpose(Image.FLIP_TOP_BOTTOM).save('vertical_flip.png')
```

Miniprojekt

Angenommen, wir haben die langweilige Aufgabe, die Grösse von Tausenden von Bildern zu ändern und ein kleines Logo-Wasserzeichen in die Ecke jedes Bildes einzufügen.

Dies mit einem einfachen Grafikprogramm wie Gimp oder Paint zu tun, würde ewig dauern. Ein anspruchsvolleres Grafikprogramm wie Photoshop kann eine Stapelverarbeitung durchführen, aber diese Software kostet Hunderte von Franken.

Schreiben wir stattdessen ein Skript, das dies erledigt.

Miniprojekt

Bitte mit programmieren :-)

--> Datei in userfolder kopieren



```
1 # exercises/logo.py
2
3 import os
4 from PIL import Image
5
6 SQUARE_FIT_SIZE = 300
7 LOGO_FILENAME = 'logo.png'
8
9 logoIm = Image.open(LOGO_FILENAME)
10 logoWidth, logoHeight = logoIm.size
11
12 # TODO: Loop over all files in the working directory.
13
14 # TODO: Check if image needs to be resized.
15
16 # TODO: Calculate the new width and height to resize to.
17
18 # TODO: Resize the image.
19
20 # TODO: Add the logo.
21
22 # TODO: Save changes.
```

Miniprojekt



```
1 # exercises/logo_1.py
2
3 # TODO: Loop over all files in the working directory.
4 os.makedirs('withLogo', exist_ok=True)
5     # Loop over all files in the working directory.
6 for filename in os.listdir('.'):
7     if not (filename.endswith('.png') or filename.endswith('.jpg')) or filename == LOGO_FILENAME:
8         continue    # skip non-image files and the logo file itself
9     im = Image.open(filename)
10    width, height = im.size
```

Miniprojekt



```
1 # exercises/logo_2.py
2
3 # Check if image needs to be resized.
4 if width > SQUARE_FIT_SIZE and height > SQUARE_FIT_SIZE:
5     # Calculate the new width and height to resize to.
6     if width > height:
7         height = int((SQUARE_FIT_SIZE / width) * height)
8         width = SQUARE_FIT_SIZE
9     else:
10        width = int((SQUARE_FIT_SIZE / height) * width)
11        height = SQUARE_FIT_SIZE
12
13    # Resize the image.
14    print('Resizing %s...' % (filename))
15    im = im.resize((width, height))
```

Miniprojekt



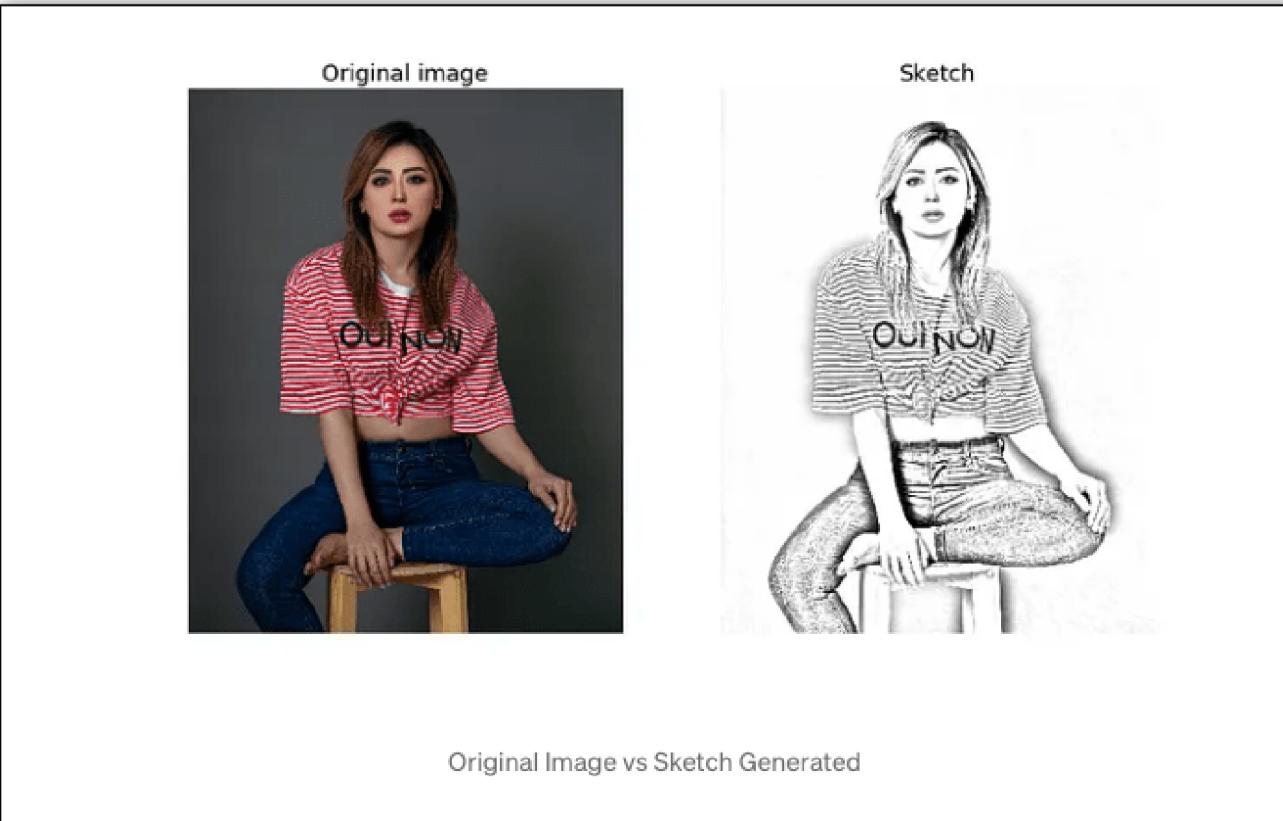
```
1 # exercises/logo_3.py
2
3     # Resize the image.
4     print('Resizing %s...' % (filename))
5     im = im.resize((width, height))
6
7     print('Adding logo to %s...' % (filename))
8     im.paste(logoIm, (width - logoWidth, height - logoHeight), logoIm)
9
10    # Save changes.
11    im.save(os.path.join('withLogo', filename))
```

Miniprojekt

Da ist ganz schön viel schiefgelaufen... logo zu gross, Bild nicht ideal... könnt ihr das in Ordnung bringen?



Pencil Sketch



Pencil Sketch



```
1 # 08_pencil.py
2
3 import cv2
4 import matplotlib.pyplot as plt
5
6 img=cv2.imread("model.jpg")
7
8 cv2.imshow('original image',img)
9 cv2.waitKey(0)
10 cv2.destroyAllWindows()
```

Pencil Sketch



```
1 # 09_pencil_2.py
2
3 plt.imshow(img)
4 plt.axis(False)
5 plt.show()
```



Pencil Sketch



```
1 # 09_pencil_2.py
2
3 plt.imshow(img)
4 plt.axis(False)
5 plt.show()
```



Pencil Sketch



```
1 # 10_pencil_3.py
2
3 img=cv2.imread("model.jpg")
4
5 cv2.imshow('original image',img)
6 cv2.waitKey(0)
7 cv2.destroyAllWindows()
8
9 grey_img=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
10
11 invert_img=cv2.bitwise_not(grey_img)
12 #invert_img=255-grey_img
13
14 # blur
15 blur_img=cv2.GaussianBlur(invert_img, (111,111),0)
16
17 # invert blur
18 invblur_img=cv2.bitwise_not(blur_img)
19 #invblur_img=255-blur_img
20
21 #sketch
22 sketch_img=cv2.divide(grey_img,invblur_img, scale=256.0)
23 cv2.imwrite('sketch.png', sketch_img)
```

Pencil Sketch

Gerne auch mit eigenen Bildern probieren :-)

Timer



```
1 # 11_timer.py
2
3 import time, subprocess
4
5 timeLeft = 60
6 while timeLeft > 0:
7     print(timeLeft, end=' ')
8     time.sleep(1)
9     timeLeft = timeLeft - 1
10
11 # TODO: At the end of the countdown, play a sound file.
```

Timer



```
1 # 11_timer.py
2
3 import time, subprocess
4
5 timeLeft = 60
6 while timeLeft > 0:
7     print(timeLeft, end=' ')
8     time.sleep(1)
9     timeLeft = timeLeft - 1
10
11 # TODO: At the end of the countdown, play a sound file.
```

Timer



```
1 # 11_timer.py
2
3 import time, subprocess
4
5 timeLeft = 60
6 while timeLeft > 0:
7     print(timeLeft, end=' ')
8     time.sleep(1)
9     timeLeft = timeLeft - 1
10
11 # TODO: At the end of the countdown, play a sound file.
12 subprocess.Popen(['start', 'alarm.wav'], shell=True)
```

Timer

Was könnten wir uns daraus kreatives bauen?



```
1 # 11_timer.py
2
3 import time, subprocess
4
5 timeLeft = 60
6 while timeLeft > 0:
7     print(timeLeft, end=' ')
8     time.sleep(1)
9     timeLeft = timeLeft - 1
10
11 # TODO: At the end of the countdown, play a sound file.
12 subprocess.Popen(['start', 'alarm.wav'], shell=True)
```

Capstone Project

Face detection mit Python? :-)

Vielen Dank

Das war alles für den Moment

Ich liebe es, über Vue, Python und mehr zu diskutieren

Sie finden mich unter @dpinezich

