Bartosz Klimza

Sprawozdanie 1 - Biblioteka OpenCV

Celem laboratorium jest zapoznanie z biblioteką OpenCV i jej podstawowymi funkcjami.

1. load_disp_img.py

```
import numpy as np
import cv2 as cv
# Load a color image in grayscale
img = cv.imread('messi5.jpg', 1)
print('img.shape={}'.format(img.shape))

'''
cv.imshow('image',img)
cv.waitKey(0)
cv.destroyAllWindows()
'''

cv.namedWindow('image', cv.WINDOW_NORMAL)
cv.imshow('image',img)
cv.waitKey(10000)
cv.destroyAllWindows()
```

Powyższy skrypt wyświetla grafikę w nowym oknie, wypisuje rozmiar grafiki i czeka 10000 milisekund, po czym zamyka okno.
W tym pliku zaprezentowane zostały następujące funkcje:

- cv.imread(„messi5.jpg”, 1): wczytuje grafikę „messi5” w formacie jpg w wersji kolorowej (parametr 1),
- cv.destroyAllWindows: zamyka wszystkie otwarte okna
- cv.waitKey(10000): Oczekuje podaną liczbę milisekund na wciśnięcie klawisza (po tym czasie grafika znika).
- cv.namedWindow: otwiera osobne okno z podpisem podanym w parametrach

2. load_disp_img_2.py

```
import numpy as np
import cv2 as cv

img = cv.imread('messi5.jpg', 0)

cv.imshow('image', img)

k = cv.waitKey(0)

if k == 27:         # wait for ESC key to exit
    cv.destroyAllWindows()
elif k == ord('s'):  # wait for 's' key to save and exit
    cv.imwrite('messigray.png', img)
    cv.destroyAllWindows()
```

Powyższy skrypt wyświetla grafikę „messi5.jpg” w odcieniach szarości, po czym oczekuje na wciśnięcie klawisza. Jeśli zostanie wciśnięty klawisz ESC, to zamyka okno, a jeśli „s”, to zapisuje grafikę pod nazwą „messigray.png” i zamyka okno.
W tym pliku zaprezentowane zostały następujące funkcje:

- cv.imread(„messi5.jpg", 0): wczytuje grafikę o nazwie „messi5” w formacie jpg i zmienia kolory na czarno-białe (parametr 0),
- cv.imshow(„image”, img): Wyświetla wczytaną grafikę
- cv.waitKey(0): Oczekuje podaną liczbę milisekund na wciśnięcie klawisza (po tym czasie grafika znika). Wartość 0 oznacza, że program czeka na przycisk w nieskończoność.
- cv.destroyAllWindows: zamyka wszystkie otwarte okna
- cv.imwrite(„messigray.png”, img): zapisuje wyświetloną grafikę pod nazwą „messigray” w formacie png.

3. load_disp_img_3.py

```
import numpy as np
import cv2 as cv

import matplotlib
#matplotlib.use('agg')
import matplotlib.pyplot as plt

img = cv.imread('messi5.jpg', 1)
img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
print('img.shape={}'.format(img.shape))

plt.imshow(img, cmap = 'gray', interpolation = 'bicubic')
plt.xticks([]), plt.yticks([])  # to hide tick values on X and Y axis
plt.show()

plt.savefig('foo.png')
plt.savefig('foo.pdf')
```

Powyższy skrypt wczytuje grafikę „messi5.jpg”, po czym zmienia kolory na odcienie szarości. Wypisuje wymiary grafiki (img.shape) i wyświetla ją za pomocą plt.imshow, a następnie zapisuje do formatów png i pdf.
W tym pliku zaprezentowane zostały następujące funkcje:

- cv.imread(„messi5.jpg”, 1): wczytuje grafikę „messi5” w formacie jpg w wersji kolorowej (parametr 1),
- cv.cvtColor(img, cv.COLOR_BGR2GRAY): zmienia kolory na odcienie szarości w grafice pod zmienną img,
- plt.imshow(img, cmap=„gray”, interpolation=„bicubic”): wyświetla grafikę img, parametr cmap wskazuje, że kolory to odcienie szarości, a interpolation wskazuje, że metoda interpolacji to bicubic,
- plt.xticks([]), pkt.yticks([]): Dzięki temu na podglądzie grafiki nie pojawiają się osie układu współrzędnych,
- plt.show(): wyświetla grafikę
- plt.savefig: zapisuje grafikę pod podaną nazwą

4. skimage_demo.py

```
import matplotlib.pyplot as plt

from skimage import data,filters

image = data.coins()
# ... or any other NumPy array!

edges = filters.sobel(image)

plt.imshow(image, cmap='gray')
print('image.shape={} imahe.max()={}'.format(image.shape, image.max()))
plt.show()



plt.imshow(edges, cmap='gray')
plt.write('aa.jpg')

plt.show()
```

Powyższy skrypt wyświetla przykładową grafikę z monetami, tą samą grafikę z filtrem Sobela (pokazującą krawędzie) i wypisuje wymiary oraz maksymalną jasność na grafice (252).
W tym pliku zaprezentowane zostały następujące funkcje:
- data.coins(): import przykładowej grafiki przedstawiającej monety
- filters.sobel(image): nałożenie filtru Sobel na grafikę

5. pil_demo.py

```
from PIL import Image, ImageFilter
#Read image
im = Image.open( 'messi5.jpg' )
#Display image
im.show()

from PIL import ImageEnhance

enh = ImageEnhance.Contrast(im)

enh.enhance(1.8).show("30% more contrast")
```

Powyższy skrypt wyświetla okna z dwiema grafikami: oryginalną oraz ze zwiększonym kontrastem.
W tym pliku zaprezentowane zostały następujące funkcje:
- Image.open(„messi5.jpg”): wczytuje grafikę
- ImageEnhance.Contrast(): służy do kontroli kontrastu na grafice
- enhance(1.8): zwiększa kontrast

6. pil_demo_2.py

```
from PIL import Image, ImageFilter

#Read image
im = Image.open( 'messi5.jpg' )

#Display image
im.show()

#Applying a filter to the image
im_sharp = im.filter( ImageFilter.SHARPEN )

#Saving the filtered image to a new file
im_sharp.save( './image_sharpened.jpg', 'JPEG' )

#Splitting the image into its respective bands, i.e. Red, Green,
#and Blue for RGB
r,g,b = im_sharp.split()

#Viewing EXIF data embedded in image
exif_data = im._getexif()
exif_data
```

Powyższy skrypt wyświetla oryginalną grafikę „messi5.jpg”, a następnie zapisuje ją wyostrzoną pod nazwą”messi5_sharpened”. Na koniec pobiera dane exif z oryginalnej grafiki.
W tym pliku zaprezentowane zostały następujące funkcje:
- im._getexif(): pobiera dane exif z grafiki
- im.filter(ImageFilter.SHARPEN): wyostrza grafikę
- im_sharp.split(): wydziela kanały R, G, B do osobnych zmiennych

7. face_detect.py

```
import cv2
import sys

# Get user supplied values
imagePath = sys.argv[1]
cascPath = sys.argv[2]

# Create the haar cascade
faceCascade = cv2.CascadeClassifier(cascPath)

# Read the image
image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Detect faces in the image
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30) #,
    #flags = cv2.cv.CV_HAAR_SCALE_IMAGE
)

print('\n')
print("Found {0} faces!".format(len(faces)))

# Draw a rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow("Faces found", image)
cv2.waitKey(0)

print('\n')
```

Powyższy skrypt wczytuje grafikę „NASA_Astronaut_Group.png” oraz CascadeClassifier z plikiem „haarcascade_frontalface_default.xml”. Następnie dokonuje detekcji twarzy na grafice na podstawie parametrów scaleFactor, minNeighbors i minSize. Na koniec wypisuje ile twarzy zostało znalezionych oraz wyświetla okno z zaznaczonymi twarzami na grafice.
W tym pliku zaprezentowane zostały następujące funkcje:
- cv2.rectangle: rysuje prostokąt o podanych parametrach
- cv2.CascadeClassifier: wczytuje klasyfikator, który zostanie wykorzystany do wykrywania twarzy