Ministerul Educaţiei, Culturii şi Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Lucrare de laborator

Disciplina: *Inteligență Artificială*

Tema: *Processing Images with OpenCV*

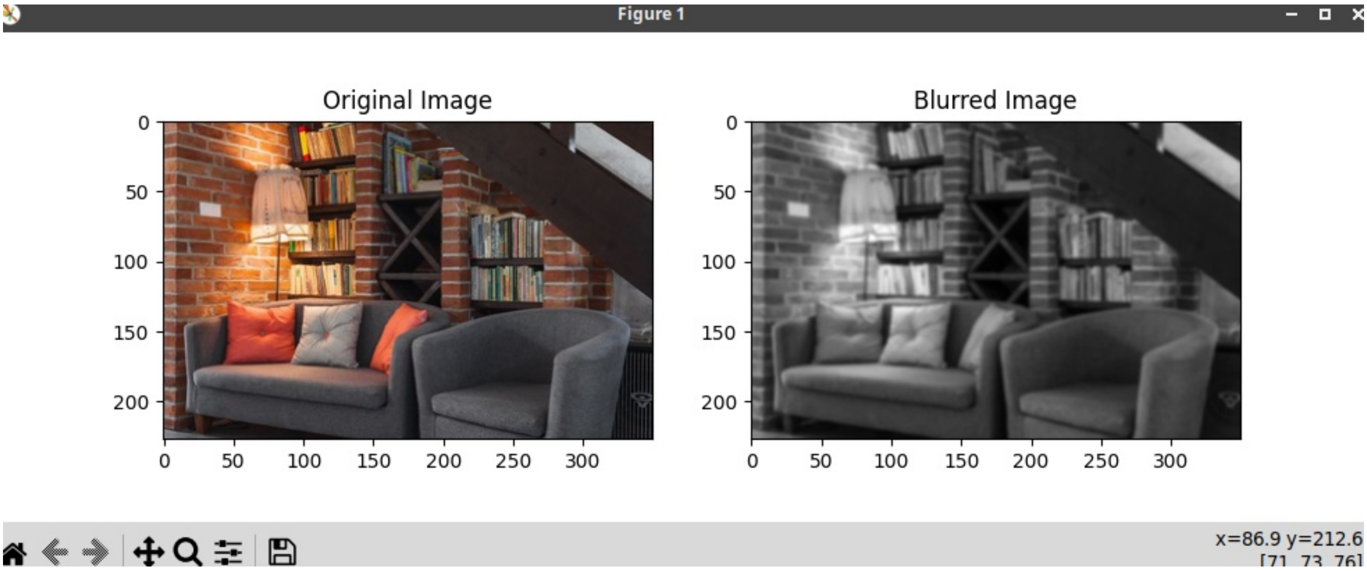
A efectuat: st. gr. SI-221M	Postu Ivan
A verificat:	Gavrilița Mihail

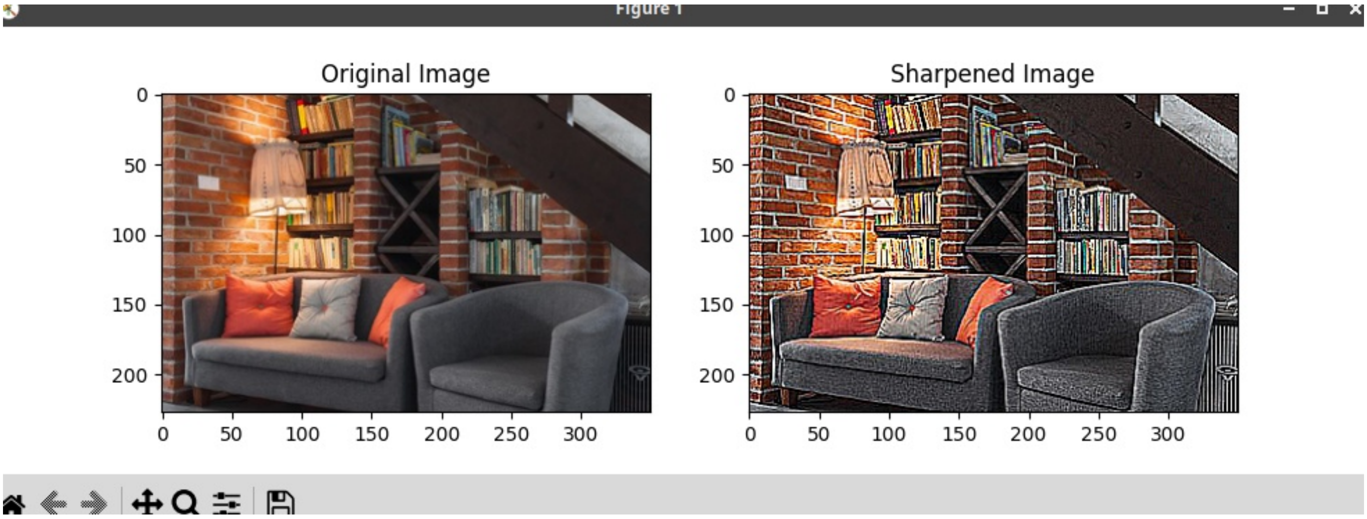
Chișinău – 2023

Task 1 Write the following functions using OpenCV. Adjust the parameters and explain your approach. Plot the initial image and the blurred image in the same plot by using Matplotlib subplots.

- A blurring function;
- A sharpening function.

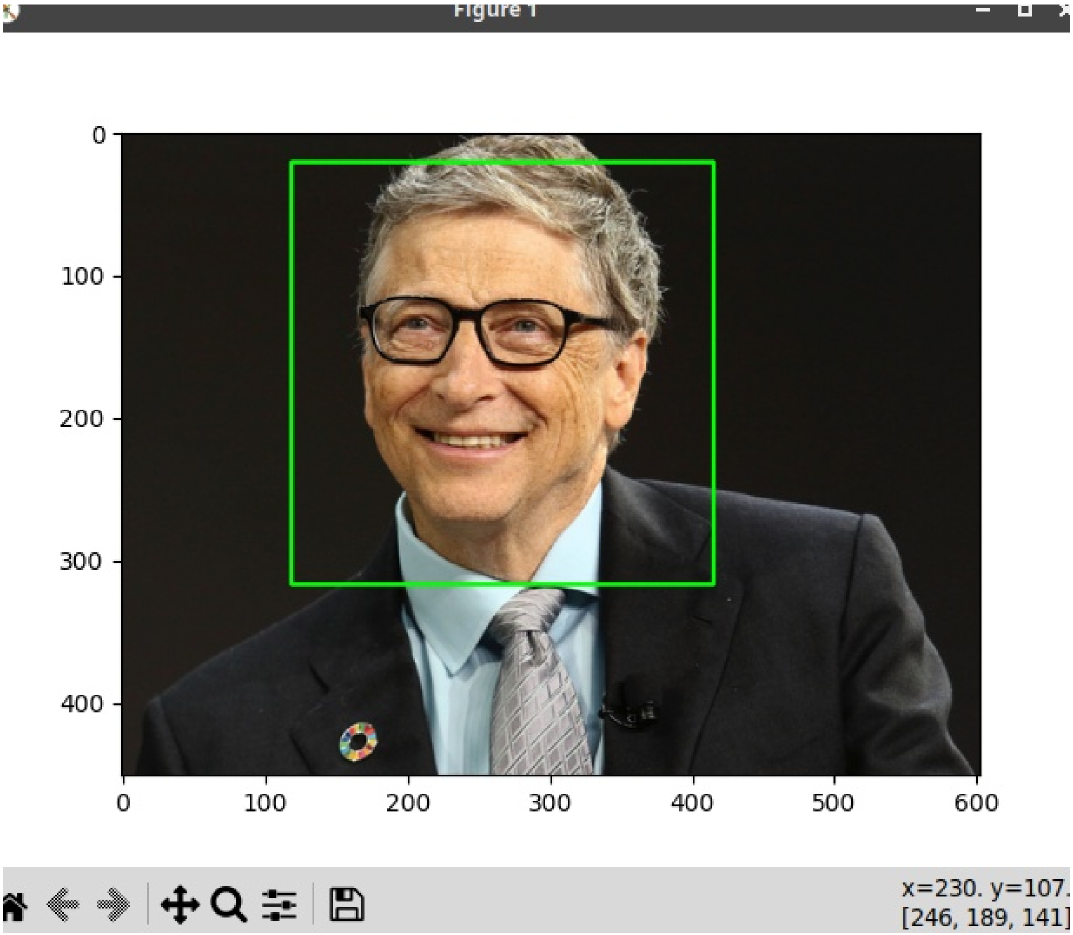
```
import matplotlib

matplotlib.use("TkAgg")

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

#t1.py
def blur_function(image_path, kernel_size=(5,5)):
    img = cv2.imread(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, kernel_size, 0)
    fig, axs = plt.subplots(1, 2, figsize=(10,5))
    axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    axs[0].set_title('Original Image')
    axs[1].imshow(blurred, cmap='gray')
    axs[1].set_title('Blurred Image')

    plt.show()

def sharpen_function(image_path):
    img = cv2.imread(image_path)
    kernel = np.array([[-1, -1, -1], [-1, 9, -1], [-1, -1, -1]])
    sharpened = cv2.filter2D(img, -1, kernel)
    fig, axs = plt.subplots(1, 2, figsize=(10, 5))
    axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    axs[0].set_title("Original Image")
    axs[1].imshow(cv2.cvtColor(sharpened, cv2.COLOR_BGR2RGB))
    axs[1].set_title("Sharpened Image")
    plt.show()


blur_function("/home/ivan/Desktop/ArtificialIntelligence/Lab4/q1.png")
sharpen_function("/home/ivan/Desktop/ArtificialIntelligence/Lab4/q1.png")
```





Task 2 Implement a face detection system using OpenCV. The function should take as input one image and output the result as the coordinates of the face, in case the image contains a face, or None if the image does not contain any faces. Assume that the image contains no more than one face.

```
import matplotlib

matplotlib.use("TkAgg")

import cv2
import matplotlib.pyplot as plt


def detect_face(image):
    # Load the pre-trained Haar cascades classifier for face detection
    face_cascade = cv2.CascadeClassifier(
        cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
    )

    # Convert the image to grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Apply histogram equalization to improve contrast
    equalized_image = cv2.equalizeHist(gray_image)

    # Detect faces in the image using the Haar cascades classifier
    faces = face_cascade.detectMultiScale(
        equalized_image, scaleFactor=1.3, minNeighbors=5
    )

    # If no faces are detected, return None
    if len(faces) == 0:
        return None
    else:
        face = max(faces, key=lambda x: x[2] * x[3])
        x, y, w, h = face
        return (x, y, x + w, y + h)


image = cv2.imread("/home/ivan/Desktop/ArtificialIntelligence/Lab4/person1.png")

face_coords = detect_face(image)

# If a face is detected, draw a rectangle around it
if face_coords is not None:
    x1, y1, x2, y2 = face_coords
    cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)

plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.show()
```



Task 3 Implement a system that detects if a photo is accepted for passport or not, by using OpenCV. You can be creative in determining the optimal strategy, but the system should at least follow the listed requirements.

- The photo should be colored. You can check that by comparing the RGB values of all the pixels. If the image is gray scale image then the values for each pixel should be equal;
- The photo should be in portrait orientation or square. Assume that the image given as input is not rotated;
- The eyes of a subject should be at the same level (with a max error of 5 pixels);
- The photo should contain only one person;
- The head of a person should represent 20% to 50% of the area of the photo

```
import cv2
import csv


def is_colored(image):
    height, width, channels = image.shape
    # Iterate over each pixel and check if it has the same value for all channels
    for y in range(height):
        for x in range(width):
            pixel = image[y, x]
            if not all(pixel == pixel[0]):
                return True

    return False


def is_portrait(image):
    height, width, channels = image.shape

    # Check if the image is portrait or square
    if height >= width:
        return True
    return False


def is_eyes_at_same_level(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_eye.xml")
    eyes = eye_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)

    if len(eyes) != 2:
        return False

    # Get the y-coordinates of the eyes
    y1 = eyes[0][1] + eyes[0][3] // 2
    y2 = eyes[1][1] + eyes[1][3] // 2

    diff = abs(y1 - y2)

    if diff <= 5:
        return True

    return False


def contains_only_one_person(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier(
        cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
    )
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
    if len(faces) == 1:
        return True
    return False


def head_area_percentage(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier(
        cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
    )
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
    if len(faces) == 0:
        return None
    (x, y, w, h) = faces[0]
    top_of_head = y - int(h * 0.25)
    head_area = w * int(h * 0.75)
    image_area = image.shape[0] * image.shape[1]
    head_area_percentage = (head_area / image_area) * 100
    return head_area_percentage


def is_accepted_for_passport(img_path):
    image = cv2.imread(img_path)
    return all(
        [
            is_colored(image),
            is_portrait(image),
            is_eyes_at_same_level(image),
            contains_only_one_person(image),
            head_area_percentage(image),
        ]
    )


with open(
    "/home/ivan/Desktop/ArtificialIntelligence/Lab4/test.csv", newline=""
) as csvfile:
    reader = csv.reader(csvfile, delimiter=",", quotechar='"')
    passed = 0
    total = 0
    for row in reader:
        if row[0] == "new_path":
            continue

        total += 1
        img_path = "/home/ivan/Desktop/ArtificialIntelligence/Lab4/" + row[0]
        expected_value_for_is_accepted_for_passport = bool(row[1])
        current_value_for_is_accepted_for_passport = is_accepted_for_passport(img_path)
        if current_value_for_is_accepted_for_passport:
            passed += 1
        print(
            row[0],
            expected_value_for_is_accepted_for_passport,
            current_value_for_is_accepted_for_passport,
        )
    print(f"Accuracy: {passed / total * 100}")
```

Output:

```
image, expected_result, current_result
test_images/8ECC1F.jpg True False
test_images/EF334A.jpg True False
test_images/33C8EE.jpg True False
test_images/55D113.jpg True False
test_images/386FB0.jpg True False
test_images/826C93.jpg True False
test_images/B8D6C9.jpg True True
test_images/E45D50.jpg True False
test_images/D11589.jpg True False
test_images/27DAC4.jpg True True
test_images/A0F4EC.jpg True False
test_images/A35F94.jpg True False
test_images/4F7014.jpg True False
test_images/4CA327.jpg True False
test_images/4731E0.jpg True True
test_images/8D6BC6.jpg True False
test_images/30916C.jpg True True
test_images/2FC4FD.jpg True False
test_images/32CEDF.jpg True False
test_images/BCB11B.jpg True True
test_images/41F890.jpg True True
test_images/60BE94.jpg True False
test_images/BDB744.jpg True False
test_images/F28B71.jpg True False
test_images/0AA0A2.jpg True True
test_images/4AE284.jpg True False
test_images/A02514.jpg True True
test_images/A8519D.jpg True True
```

```
test_images/7E0875.jpg True True
test_images/870125.jpg True False
test_images/80003A.jpg True False
test_images/35E54F.jpg True False
test_images/14A19C.jpg True True
test_images/C159CB.jpg True False
test_images/A6E4A7.jpg True False
test_images/53DEBB.jpg True True
test_images/D87D5F.jpg True True
test_images/9C02DD.jpg True True
test_images/94E27D.jpg True False
test_images/711A6A.jpg True False
test_images/56C0B5.jpg True True
test_images/E2BD04.jpg True True
test_images/476435.jpg True True
test_images/455D92.jpg True True
test_images/D35C72.jpg True False
test_images/76DD74.jpg True False
test_images/5633B8.jpg True False
test_images/6E59C6.jpg True False
Accuracy: 37.5%
```

Concluzie

În urma lucrării de laborator sa studiat și utilizat un set puternic de instrumente pentru lucrul cu date de imagine OpenCV. În timpul lucrului de laborator cu OpenCV, sa explorat diverse tehnici de procesare a imaginilor, cum ar fi filtrarea, pragurile și detectarea marginilor, precum și detectarea și recunoașterea obiectelor folosind tehnici precum cascadele Haar și modelele de învățare profundă. Unul dintre avantajele utilizării OpenCV este gama sa largă de funcții și algoritmi încorporați, care pot facilita începerea cu procesarea imaginilor și sarcinile de viziune pe computer.