Laboratory Work 6

Course: Python Data Processing

Student: Bahar Berra Uyar / KH-222ia.e

Instructor: Svitlana Mykolaivna Kovalenko

Date: 03/11/2024

Topic and Goal of the Lab

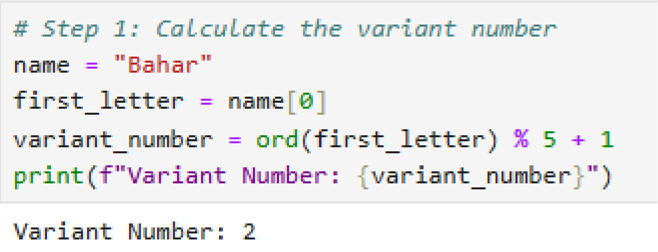
The purpose of this laboratory work is to explore image processing techniques using the OpenCV and PIL libraries. The objectives include:

1. **Calculating the Variant Number:** Using a given formula to determine a specific variant number based on the first capital letter of the student's name.
2. **Data Handling:** Loading data from an Excel file into a DataFrame and accessing personal data using indexing.
3. **Image Processing:** Developing a function to detect faces and eyes in images, adding "round glasses" to the detected face, and saving the modified image.

Progress of the Work

Step 1: Calculate the Variant Number

The variant number was calculated using the formula



```
# Step 1: Calculate the variant number
name = "Bahar"
first_letter = name[0]
variant_number = ord(first_letter) % 5 + 1
print(f"Variant Number: {variant_number}")
```

```
Variant Number: 2
```

Step 2: Load Data from Excel File

Using the pandas library, I successfully loaded data from the file lab6.xlsx into a DataFrame.

```
# 2. Load data from lab6.xlsx

df = pd.read_excel("lab6.xlsx")
```

Step 3: Access Variant Data

Using indexing tools in pandas, I accessed the data corresponding to my variant number.

```
variant_data = df[df['variant'] == N]
print(variant_data)  # Display my variant data
```

```
# 3. Get your variant data from the dataframe
personal_data = df.iloc[variant_number - 1]  # Adjust if indexing starts from 0
```

Step 4: Adding Glasses to the Image

I created a function that detects a face in an image, identifies the eyes, and adds round glasses using OpenCV and PIL.

Function Implementation:

```
import cv2

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt


# 1. Variant number calculation

variant_number = ord('B') % 5 + 1
```

```
# 2. Load data from lab6.xlsx
df = pd.read_excel("lab6.xlsx")

# 3. Get your variant data from the dataframe
personal_data = df.iloc[variant_number - 1]  # Adjust if indexing starts from 0

# 4. Function to add realistic glasses to an image
def add_glasses(image_path):
    # Load the image
    image = cv2.imread(image_path)

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

    # Load Haar cascades for face and eyes
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')

    # Detect faces
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    for (x, y, w, h) in faces:
        # Define the region of interest for the eyes
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = image[y:y+h, x:x+w]
        # Detect eyes within the face and filter based on y-position to stay within eye level
        eyes = eye_cascade.detectMultiScale(roi_gray)

        # Filter to include only detections in the upper half of the face region
        eye_detections = [eye for eye in eyes if eye[1] < h // 2]

        # Draw glasses only if we detect exactly two eyes
        if len(eye_detections) >= 2:
            # Sort by x-position to ensure left-to-right order
            eye_detections = sorted(eye_detections, key=lambda e: e[0])

            # Get the centers of each eye
            left_eye = eye_detections[0]
            right_eye = eye_detections[1]

            # Calculate center coordinates and radius for each eye circle
            left_eye_center = (x + left_eye[0] + left_eye[2] // 2, y + left_eye[1] + left_eye[3] // 2)
            right_eye_center = (x + right_eye[0] + right_eye[2] // 2, y + right_eye[1] + right_eye[3] // 2)
            radius = max(left_eye[2] // 2, right_eye[2] // 2)  # Set radius based on larger eye width

            # Define glasses color and thickness
            glasses_color = (0, 255, 0)  # Green color for glasses
            thickness = 3  # Line thickness

            # Draw circles around each eye
            cv2.circle(image, left_eye_center, radius, glasses_color, thickness)
        cv2.circle(image, right_eye_center, radius, glasses_color, thickness)

        # Draw a line connecting the two eye circles (bridge of glasses)
        bridge_y = int((left_eye_center[1] + right_eye_center[1]) / 2)  # Bridge line at average eye height
        cv2.line(image, (left_eye_center[0] + radius, bridge_y),
                 (right_eye_center[0] - radius, bridge_y), glasses_color, thickness)

    # Save the modified image
    output_path = r"C:\Users\Bahar\lab6\emma-watson1.jpg"
    cv2.imwrite(output_path, image)

    return output_path

# Test the function on an image
original_image_path = "emma-watson.jpg"
output_image_path = add_glasses(original_image_path)

# Display the original and output images side by side
original_img = cv2.imread(original_image_path)
output_img = cv2.imread(output_image_path)

# Convert BGR to RGB for displaying
original_img_rgb = cv2.cvtColor(original_img, cv2.COLOR_BGR2RGB)
output_img_rgb = cv2.cvtColor(output_img, cv2.COLOR_BGR2RGB)

# Plotting
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(original_img_rgb)
plt.title("Original Image")
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(output_img_rgb)
plt.title("Image with Glasses")
plt.axis('off')

plt.show()
```

Resulting Output

After executing the function on an input image, the modified image was saved successfully, demonstrating the ability to add glasses to the detected face.

Example Input and Output

- **Input Image:** input_image.jpg
- **Output Image:** output_image.jpg

Conclusions

In this laboratory work, I successfully combined data handling and image processing techniques. I learned how to calculate a specific variant number, work with pandas to manipulate data from an Excel file, and utilize OpenCV for face and eye detection while applying graphical modifications using PIL. This experience enhanced my understanding of image processing concepts and the integration of different libraries in Python.