Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря Сікорського
Кафедра обчислювальної техніки ФІОТ

ЗВІТ
з лабораторної роботи № 6
з навчальної дисципліни «Computer Vision»

Тема:

ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ПОРІВНЯННЯ ЦИФРОВИХ ЗОБРАЖЕНЬ ДЛЯ СТЕЖЕННЯ ЗА ОБ'ЄКТАМИ У ВІДЕОПОТОЦІ

Виконав:
Студент 3 курсу кафедри ІПІ ФІОТ,
Навчальної групи ІП-14
Бабіч Д. В.

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О. О.

Київ 2024

I. Мета:

Дослідити принципи та особливості практичного застосування технологій порівняння цифрових зображень для стеження за об’єктами у відеопотоці з використанням спеціалізованих програмних бібліотек.

II. Завдання:

ГРУПА ВИМОГ 3.

Розробити програмний скрипт, що реалізує стеження за об’єктом у цифровому відеопотоці. Зміст відео, об’єкт стеження – обрати самостійно. Метод та технологію стеження обрати такою, що забезпечує стійкість процесу object-tracking для обраних вихідними даними (відео, об’єкт стеження). Вибір обґрунтувати та довести його ефективність.

III. Результати виконання лабораторної роботи.

3.1. Синтезована математична модель обробки графічних зображень.

HOG (Histogram of Oriented Gradients) – це метод для виявлення об'єктів на зображеннях, що базується на градієнтах зображення. Основна ідея полягає в тому, щоб описати зовнішній вигляд об'єктів на основі розподілу градієнтів напрямлення.

Спочатку зображення перетворюється в градієнтне зображення, що визначає напрямок та інтенсивність зміни яскравості пікселів. Це можна зробити, використовуючи оператор Собеля. Горизонтальні та вертикальні компоненти градієнта обчислюються за допомогою наступних формул:

$$G_x = (I_{(x+1,\,y)} - I_{(x-1,\,y)}$$

Формула, яка описує горизонтальний градієнт, де $I_{(x,\,y)}$ – яскравість пікселя у заданій позиції.

$$G_y = (I_{(x,\,y+1)} - I_{(x,\,y-1)}$$

Формула, яка описує горизонтальний градієнт, де $I_{(x,\,y)}$ – яскравість пікселя у заданій позиції.

Зображення розділяється на малий блоки (наприклад, 8x8 пікселів), а потім для кожного блоку обчислюється гістограма орієнтованих градієнтів (HOG). Градієнти в кожному блоку групуються в гістограму орієнтованих градієнтів, де градієнти з однаковими напрямками об'єднуються в кілька бінів гістограми.

Послідовності гістограм орієнтованих градієнтів об'єднуються у блоки. Кожен блок може містити кілька послідовностей гістограм орієнтованих градієнтів. Перед використанням цих блоків для подальшого аналізу, їх нормалізують з метою роботи з освітленням.

Загальний вектор дескриптора HOG формується шляхом об'єднання нормалізованих гістограм орієнтованих градієнтів у всьому зображенні. Цей вектор може бути використаний для подальшого навчання класифікаторів (наприклад, SVM) або для порівняння зображень.

Класифікатор каскадів Хаара - це метод машинного навчання, який використовується для виявлення об'єктів на зображеннях. Основна ідея каскадних класифікаторів Хаара полягає у використанні ансамблю або "каскаду" простих функцій (відомих як класифікатори "слабких" вузьких дерев рішень) для швидкого відфільтровування областей зображення, які, мабуть, не містять об'єктів інтересу.

3.2. Результати архітектурного проектування та їх опис.

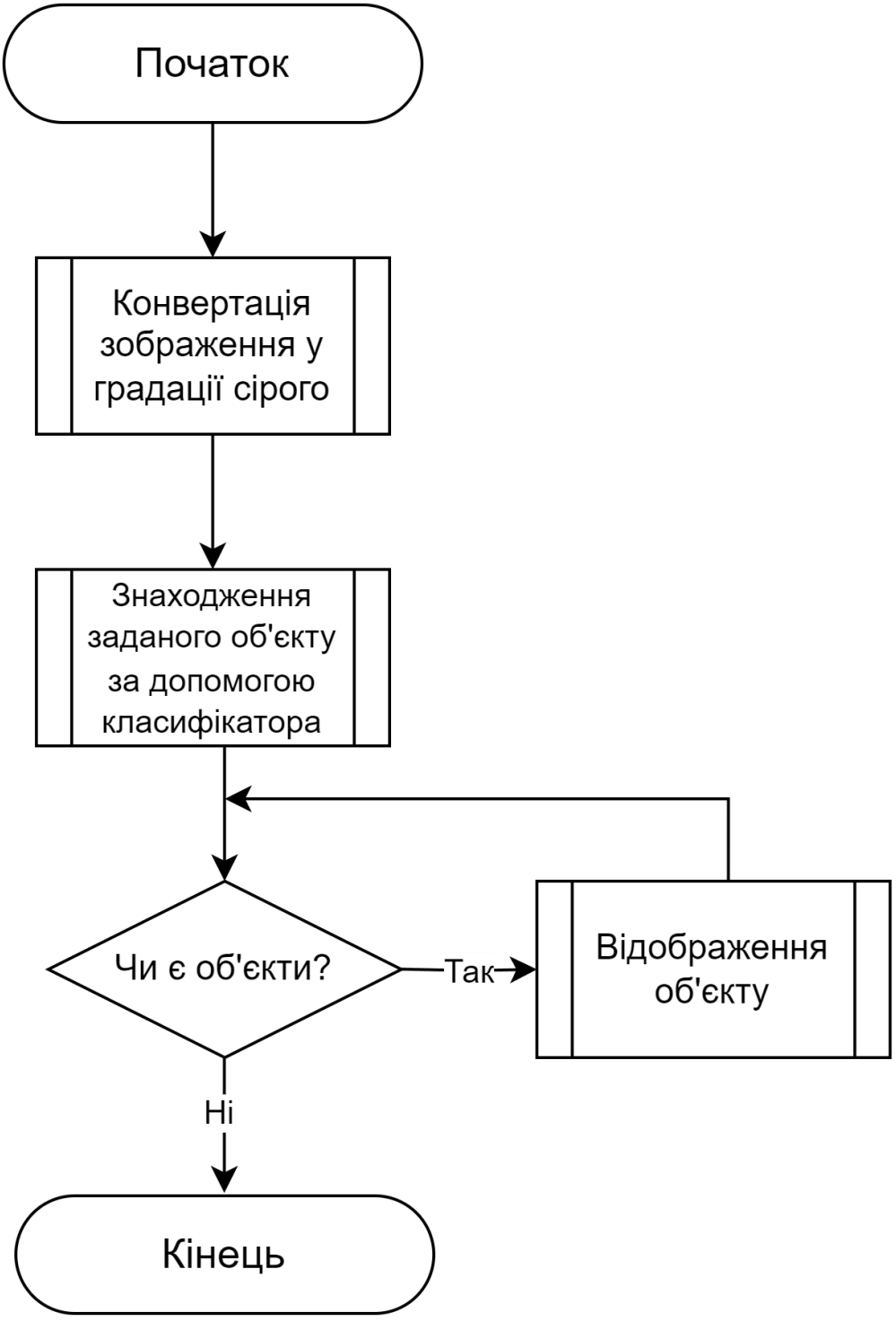

Рисунок 1.1 – Діаграма алгоритму ідентифікації

1. Конвертація зображення у відтінки сірого;
2. Знаходження заданих об’єктів за допомогою відповідного класифікатора;
3. Якщо об’єкти були знайдені, то їх відображення.

3.3. Опис структури проекту програми.

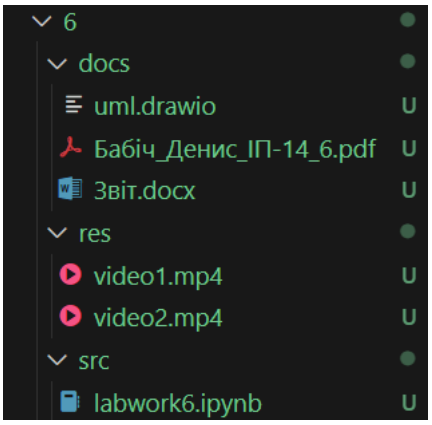

Рисунок 1.2 – Структура проєкту

У директорії src зберігається labwork6.py, який є юпітер-записником з вихідним кодом, директорія docs – зберігає файли звіту у форматі pdf, docx та drawio (файл з діаграмами), директорія res – 2 тестових відео.

3.4. Результати роботи програми відповідно до завдання.

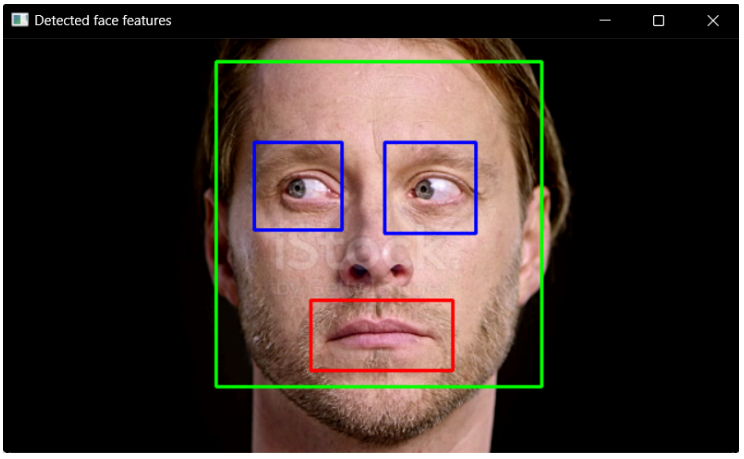

Рисунок 1.3 – Результат знаходження рис обличчя на відео

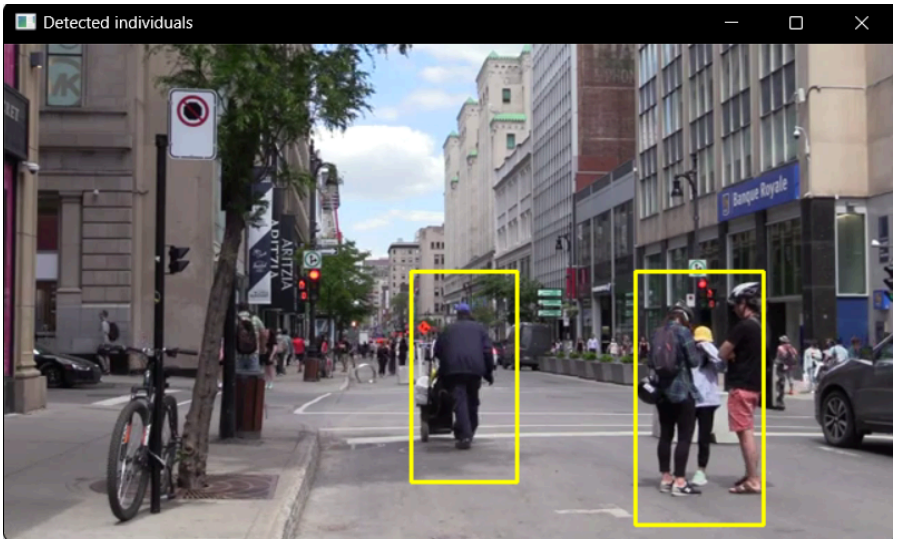

Рисунок 1.4 – Результат знаходження людей на відео

3.5. Програмний код, що забезпечує отримання результату.

```
# %% [markdown]
# # Лабораторна робота № 6

# %% [markdown]
# ##  ІП-14 Бабіч Денис

# %% [markdown]
# Розробити програмний скрипт, що реалізує стеження за об’єктом у цифровому відеопотоці. Зміст відео, об’єкт стеження – обрати самостійно. Метод та технологію стеження обрати такою, що забезпечує стійкість процесу object-tracking для обраних вихідними даними (відео, об’єкт стеження). Вибір обґрунтувати та довести його ефективність.

# %% [markdown]
# ---

# %% [markdown]
# # Підготовчий етап

# %% [markdown]
# ## Імпортування необхідних модулів

# %%
import cv2
import numpy as np

# %% [markdown]
# ---

# %% [markdown]
# # Основний етап

# %% [markdown]
# ## Застосування на прикладі знаходження рис обличчя

# %%
KEY_ESCAPE = 27

CLASSIFIER_EYE = cv2.CascadeClassifier(f"{cv2.data.haarcascades}haarcascade_eye.xml")
CLASSIFIER_SMILE = cv2.CascadeClassifier(f"{cv2.data.haarcascades}haarcascade_smile.xml")
CLASSIFIER_FACE = cv2.CascadeClassifier(f"{cv2.data.haarcascades}haarcascade_frontalface_default.xml")

BOUNDING_THICKNESS = 2
COLOR_EYE = (255, 0, 0)
COLOR_FACE = (0, 255, 0)
COLOR_SMILE = (0, 0, 255)

CAPTURE = cv2.VideoCapture("../res/video1.mp4")
FPS = int(CAPTURE.get(cv2.CAP_PROP_FPS))
```

```
while True:
    ret, frame = CAPTURE.read()

    if not ret:
        break

    gray_filter = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

     faces = CLASSIFIER_FACE.detectMultiScale(gray_filter, scaleFactor = 1.05, minNeighbors
= 5)

    for (x, y, width, height) in faces:
                    cv2.rectangle(frame, (x, y), (x + width, y + height), COLOR_FACE,
BOUNDING_THICKNESS)

        roi_color = frame[y:(y + height), x:(x + width)]
        roi_gray = gray_filter[y:(y + height), x:(x + width)]

         eyes = CLASSIFIER_EYE.detectMultiScale(roi_gray, scaleFactor = 1.1, minNeighbors =
5)

        for (x, y, width, height) in eyes:
                      cv2.rectangle(roi_color, (x, y), (x + width, y + height), COLOR_EYE,
BOUNDING_THICKNESS)

                  smiles = CLASSIFIER_SMILE.detectMultiScale(roi_gray, scaleFactor = 1.2,
minNeighbors = 125)

        for (x, y, width, height) in smiles:
                      cv2.rectangle(roi_color, (x, y), (x + width, y + height), COLOR_SMILE,
BOUNDING_THICKNESS)

    cv2.imshow("Detected face features", frame)

    if cv2.waitKey(FPS) & 0xFF == KEY_ESCAPE:
        break

CAPTURE.release()
cv2.destroyAllWindows()

# %% [markdown]
# ## Застосування на прикладі знаходження людей на відео

# %%
KEY_ESCAPE = 27

HOG_DESCRIPTOR = cv2.HOGDescriptor()
HOG_DESCRIPTOR.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

BOUNDING_THICKNESS = 2
COLOR_HUMAN = (0, 255, 255)
```

```
CAPTURE = cv2.VideoCapture("../res/video2.mp4")
FPS = int(CAPTURE.get(cv2.CAP_PROP_FPS))

while True:
   _, frame = CAPTURE.read()

   if not ret:
      break

   gray_filter = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

      people = HOG_DESCRIPTOR.detectMultiScale(gray_filter, winStride = (8, 8), padding = (16, 16), scale = 1.25)
      people = np.array([[human_x, human_y, human_x + human_w, human_y + human_h] for (human_x, human_y, human_w, human_h) in people[0]])

   for (x, y, width, height) in people:
                              cv2.rectangle(frame, (x, y), (width, height), COLOR_HUMAN, BOUNDING_THICKNESS)

   cv2.imshow("Detected individuals", frame)

   if cv2.waitKey(FPS) & 0xFF == KEY_ESCAPE:
      break

CAPTURE.release()
cv2.destroyAllWindows()
```

IV. Висновки.

У цій лабораторній роботі було досліджено технології порівняння цифрових зображень для стеження за об’єктами у відеопотоці. Було розроблено програмний скрипт, що реалізує стеження за об’єктом у цифровому відеопотоці, використовуючи методи HOG (Histogram of Oriented Gradients) та класифікатор каскадів Хаара, було здійснено стеження за об’єктами на відео. Ці інструменти були використані для виявлення рис обличчя та людей на відео. Результати були задовільними, що демонструє ефективність використаних методів для стеження за об’єктами у відеопотоці.

Виконав: ІП-14 Бабіч Д. В.