Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
Факультет комп’ютерних наук
Кафедра інженерії програмного забезпечення

Звіт
з лабораторної роботи №1
з дисципліни «Комп'ютерний зір»
на тему «Виділення пішоходів»

Виконав ст. гр. ІПЗм-22-7:
Миронюк С.А.

Перевірив викладач:
Работягов А.В.

Харків, 2023

Мета завдання:

- реалізувати систему комп'ютерного зору для виявлення пішоходів,
- ознайомитися із застосуванням методів машинного навчання.

OpenCV — це бібліотека з відкритим вихідним кодом, яка спрямована на комп’ютерне бачення в реальному часі. Ця бібліотека розроблена компанією Intel і є кросплатформною – вона може підтримувати Python, C++, Java тощо. Комп’ютерне бачення – це найсучасніша галузь комп’ютерних наук, яка має на меті дозволити комп’ютерам розуміти, що видно на зображенні. OpenCV є однією з найбільш широко використовуваних бібліотек для завдань комп’ютерного бачення, таких як розпізнавання обличчя, виявлення руху, виявлення об’єктів тощо.

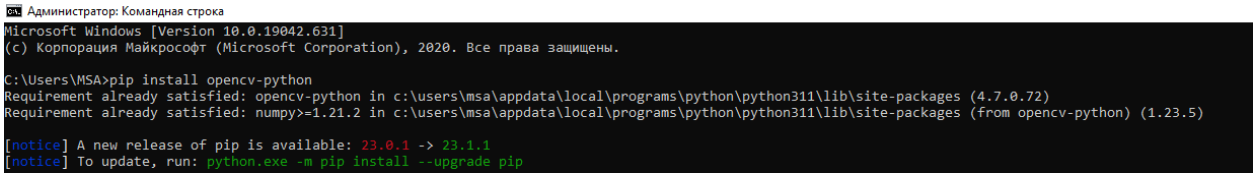
Виявлення пішоходів є дуже важливою областю досліджень, оскільки воно може, наприклад, покращити функціональність системи захисту пішоходів у безпілотних автомобілях.

Ми можемо витягти такі функції, як голова, дві руки, дві ноги тощо, із зображення людського тіла та передати їх для навчання моделі машинного навчання. Після навчання модель можна використовувати для виявлення та відстеження людей на зображеннях і відеопотоках. OpenCV має вбудований метод виявлення пішоходів. Він має попередньо підготовлену модель **HOG (гістограма орієнтованих градієнтів) + модель Linear SVM** для виявлення пішоходів на зображеннях і відеопотоках.

Гістограма орієнтованих градієнтів. Цей алгоритм перевіряє безпосередньо навколишні пікселі кожного окремого пікселя. Мета – перевірити, наскільки темнішим є поточний піксель порівняно з навколишніми пікселями. Алгоритм малює стрілки, які показують напрямок, у якому зображення стає темнішим. Він повторює процес для кожного пікселя зображення. Нарешті, кожен піксель буде замінено стрілкою, ці стрілки називаються градієнтами. Ці градієнти показують потік світла від світлого до темного. Використовуючи ці градієнти, алгоритми виконують подальший аналіз.

Хід роботи

Встановлення бібліотеки OpenCV-Python здійснюю за допомогою команди: `pip install opencv-python`

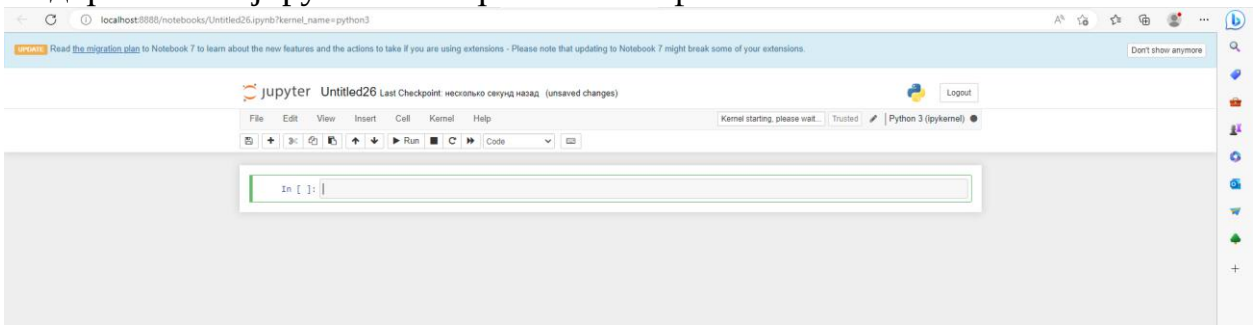


Встановимо pip install jupyterlab та запустимо його командою jupyter notebook.

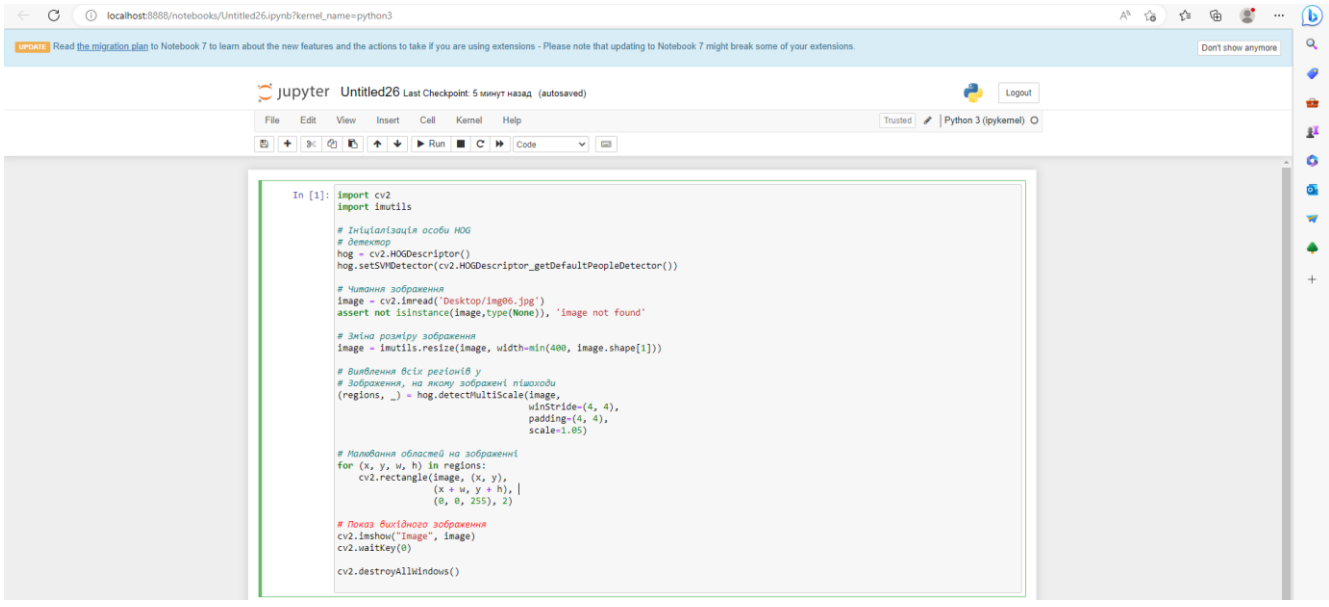
Запускаю jupyter notebook:

```
C:\Users\MSA>jupyter notebook

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions.

https://jupyter-notebook.readthedocs.io/en/latest/migrate_to_notebook7.html

Please note that updating to Notebook 7 might break some of your extensions.

[W 20:07:16.105 NotebookApp] Loading JupyterLab as a classic notebook (v6) extension.
[C 20:07:16.106 NotebookApp] You must use Jupyter Server v1 to load JupyterLab as notebook extension. You have v2.5.0 installed.
    You can fix this by executing:
        pip install -U "jupyter-server<2.0.0"
[I 20:07:16.137 NotebookApp] Serving notebooks from local directory: C:\Users\MSA
[I 20:07:16.137 NotebookApp] Jupyter Notebook 6.5.3 is running at:
[I 20:07:16.138 NotebookApp] http://localhost:8888/?token=08b0a5f67f36a5d33b6c6b2998a79beb203052f285850599
[I 20:07:16.138 NotebookApp]  or http://127.0.0.1:8888/?token=08b0a5f67f36a5d33b6c6b2998a79beb203052f285850599
[I 20:07:16.138 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 20:07:16.212 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/MSA/AppData/Roaming/jupyter/runtime/nbserver-6304-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=08b0a5f67f36a5d33b6c6b2998a79beb203052f285850599
     or http://127.0.0.1:8888/?token=08b0a5f67f36a5d33b6c6b2998a79beb203052f285850599
[I 20:07:22.353 NotebookApp] 302 GET /notebooks/Untitled25.ipynb?kernel_name=python3 (::1) 1.960000ms
0.03s - Debugger warning: It seems that frozen modules are being used, which may
0.00s - make the debugger miss breakpoints. Please pass -Xfrozen_modules=off
0.00s - to python to disable frozen modules.
0.00s - Note: Debugging will proceed. Set PYDEVD_DISABLE_FILE_VALIDATION=1 to disable this validation.
```

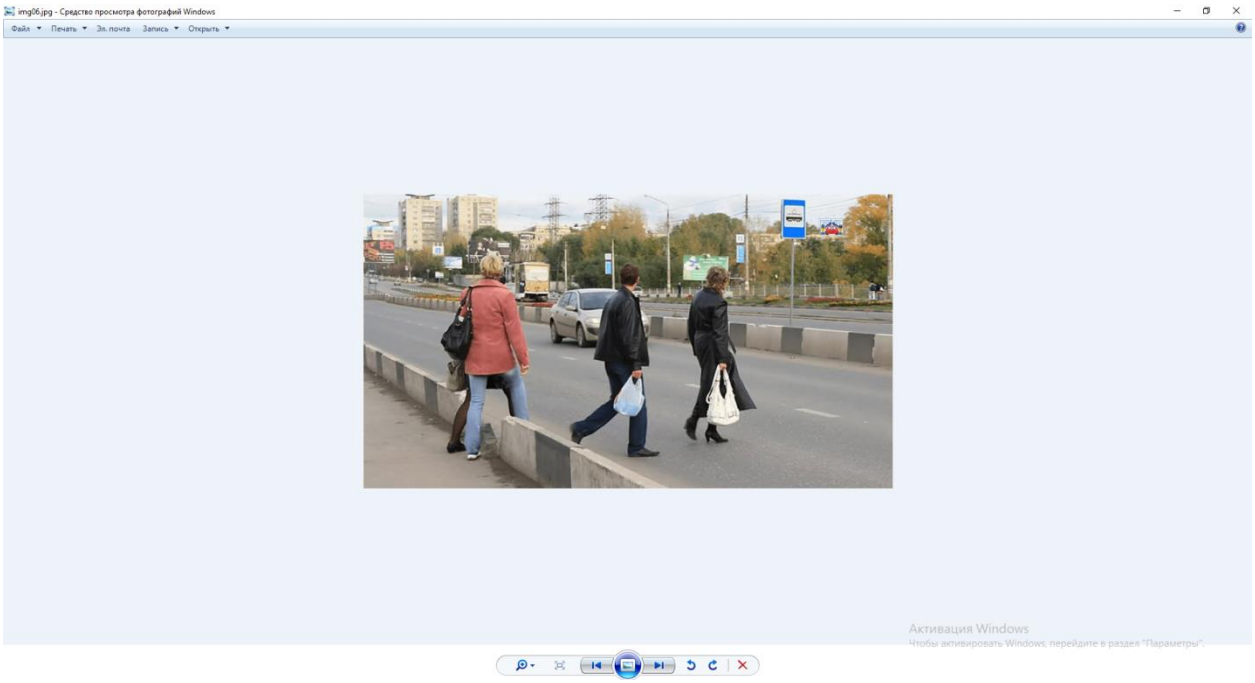
Відкривається jupyter та створюю новий файл:



Далі я пишу код для виявлення пішоходів:

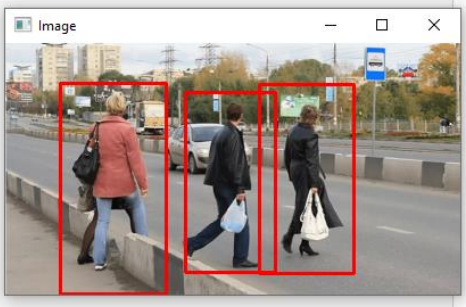


Вхідне зображення:

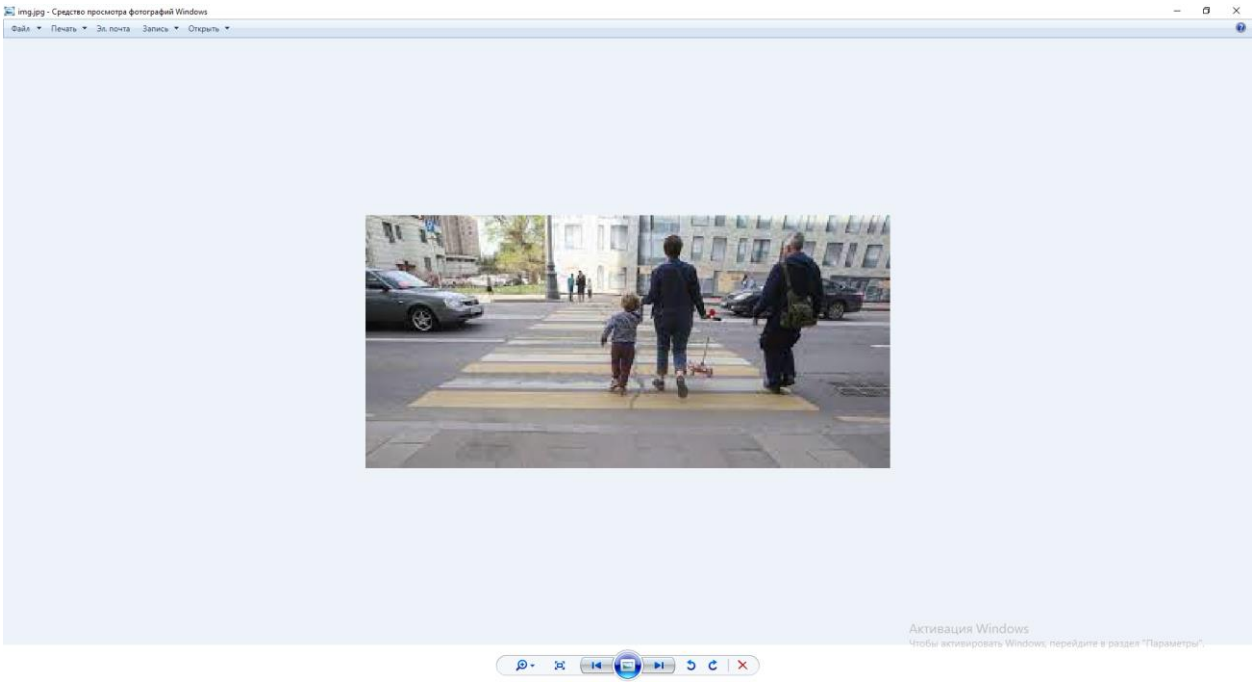


Робота коду:

In [*]:
```
import cv2
import imutils

# Ініціалізація особи HOG
# детектор
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# Читання зображення
image = cv2.imread('Desktop/img06.jpg')
assert not isinstance(image,type(None)), 'image not found'

# Зміна розміру зображення
image = imutils.resize(image, width=min(400, image.shape[1]))

# Виявлення всіх регіонів у
# Зображення, на якому зображені пішоходи
(regions, _) = hog.detectMultiScale(image,
                                    winStride=(4, 4),
                                    padding=(4, 4),
                                    scale=1.05)

# Малювання областей на зображенні
for (x, y, w, h) in regions:
    cv2.rectangle(image, (x, y),
                  (x + w, y + h),
                  (0, 0, 255), 2)

# Показ вихідного зображення
cv2.imshow("Image", image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

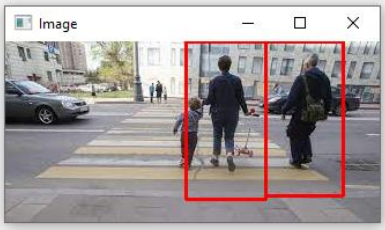


Вхідне зображення другого фото:



Робота коду:

In [*]:
```
import cv2
import imutils

# Ініціалізація особи HOG
# детектор
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# Читання зображення
image = cv2.imread('Desktop/img.jpg')
assert not isinstance(image,type(None)), 'image not found'

# Зміна розміру зображення
image = imutils.resize(image, width=min(400, image.shape[1]))

# Виявлення всіх регіонів у
# Зображення, на якому зображені пішоходи
(regions, _) = hog.detectMultiScale(image,
                                    winStride=(4, 4),
                                    padding=(4, 4),
                                    scale=1.05)

# Малювання областей на зображенні
for (x, y, w, h) in regions:
    cv2.rectangle(image, (x, y),
                  (x + w, y + h),
                  (0, 0, 255), 2)

# Показ вихідного зображення
cv2.imshow("Image", image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```



Висновок: в результаті виконаної роботи освоїв реалізацію системи комп'ютерного зору для виявлення пішоходів, ознайомився із застосуванням методів машинного навчання.