

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИЙ НАУК ТА КІБЕРНЕТИКИ

Звіт
до лабораторної роботи
"Трекінг об'єктів в реальному часі"
з курсу
"Розпізнавання образів та машинне навчання"

Виконав студент групи ТК-4
Вернигор Віталій Вадимович

Київ
2021

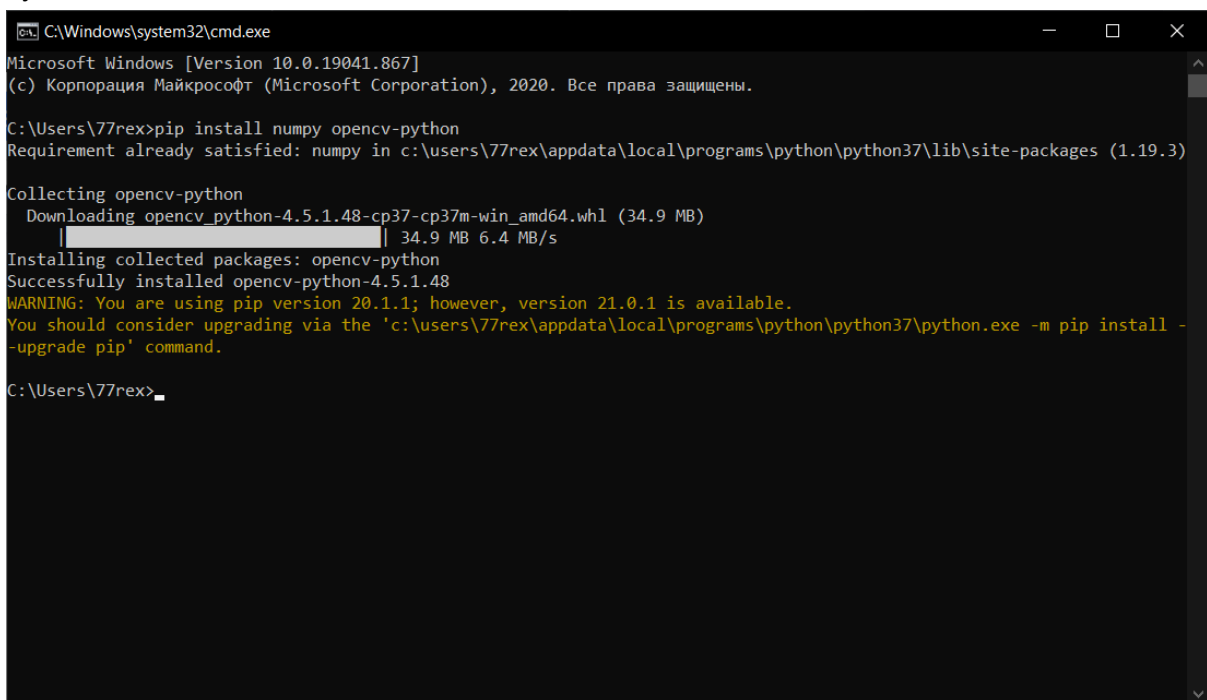
Постановка задачі

Лабораторний практикум передбачає створення системи для трекінгу будь яких об'єктів у реальному часі.

Реалізація

1. Встановлення.

Виконання лабораторної було почато з встановлення бібліотеки OpenCV Python. Для цього було застосовано `pip` - система управління пакетами, яка використовується для установки і управління програмними пакетами, написаними на Python. На скриншоті нижче наведено приклад встановлення пакетів "NumPy" та "OpenCV" для Python.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19041.867]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

C:\Users\77rex>pip install numpy opencv-python
Requirement already satisfied: numpy in c:\users\77rex\appdata\local\programs\python\python37\lib\site-packages (1.19.3)
Collecting opencv-python
  Downloading opencv_python-4.5.1.48-cp37-cp37m-win_amd64.whl (34.9 MB)
    | 34.9 MB 6.4 MB/s
Installing collected packages: opencv-python
Successfully installed opencv-python-4.5.1.48
WARNING: You are using pip version 20.1.1; however, version 21.0.1 is available.
You should consider upgrading via the 'c:\users\77rex\appdata\local\programs\python\python37\python.exe -m pip install -
-upgrade pip' command.

C:\Users\77rex>_
```

2. Застосування бібліотеки OpenCV.

Далі в основному файлі проекту "[objectTracking.py](#)" я ініціалізував бібліотеку OpenCV.

```
1 import cv2
```

Наступним кроком я ініціалізую трекер реалізований у бібліотеці OpenCV

```
11 tracker = cv2.legacy_TrackerMOSSE.create()
```

3. Зчитування зображення з вебкамери

Далі я підключаю програму до потоку зображень з вебкамери:

```
16 cap = cv2.VideoCapture(0)
17 # TRACKER INITIALIZATION
18 success, frame = cap.read()
19 bbox = cv2.selectROI("Tracking", frame, False)
20 tracker.init(frame, bbox)
```

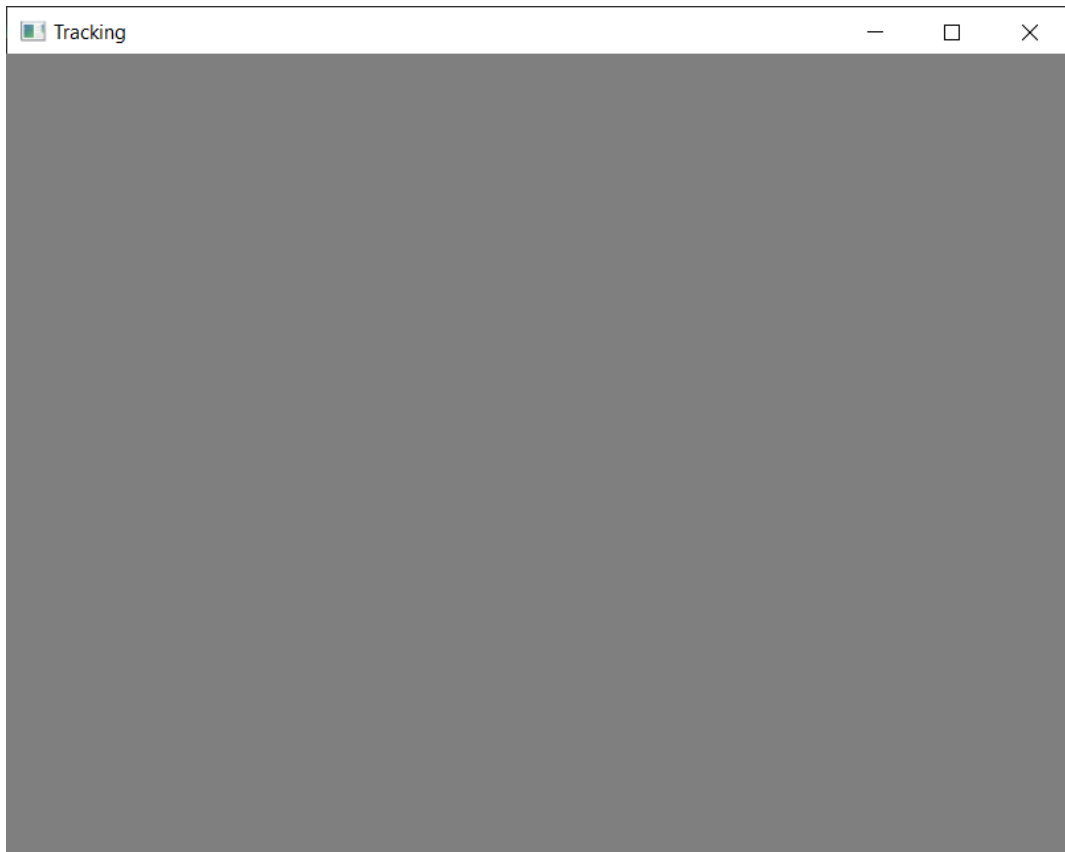
Першочергово сканується перший кадр, далі програма буде чекати на виділення об'єкту.

4. Створення вікна демонстрації

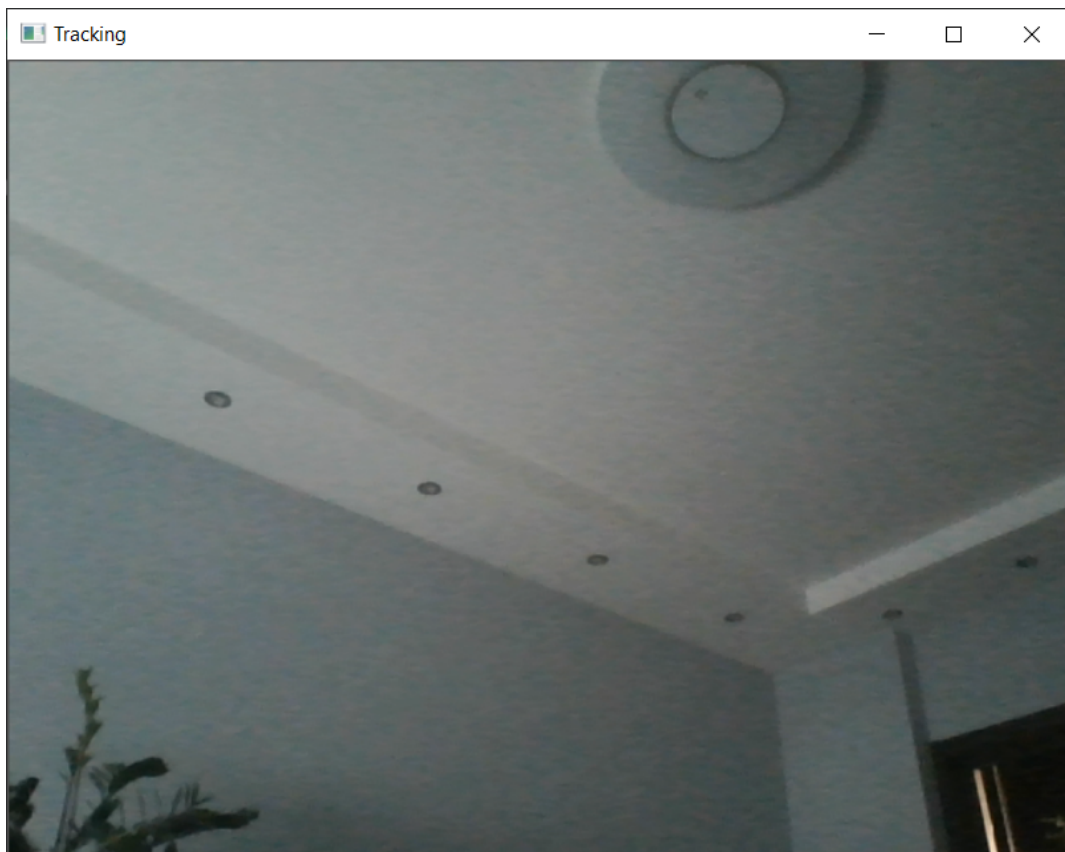
Далі я зробив найпростіше віконечко, куди виводити зображення та користну інформацію щодо фреймрейту та статусу трекера

```
23 def drawBox(img, bbox):
24     x, y, w, h = int(bbox[0]), int(bbox[1]), int(bbox[2]), int(bbox[3])
25     cv2.rectangle(img, (x, y), ((x + w), (y + h)), (255, 0, 255), 3, 3)
26     cv2.putText(img, "Tracking", (100, 75), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
```

ось так воно виглядає без картинки:



а ось так, коли я передав туди картинку з вебкамери



5. Оновлення картинки

Тепер, маючи віконце демонстрації я виводжу туди те, що зчитує моя програма з вебкамери та циклічно оновлюю.

```
29 while True:
30
31     timer = cv2.getTickCount()
32     success, img = cap.read()
33     success, bbox = tracker.update(img)
34
35     if success:
36         drawBox(img, bbox)
37     else:
38         cv2.putText(img, "Lost", (100, 75), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

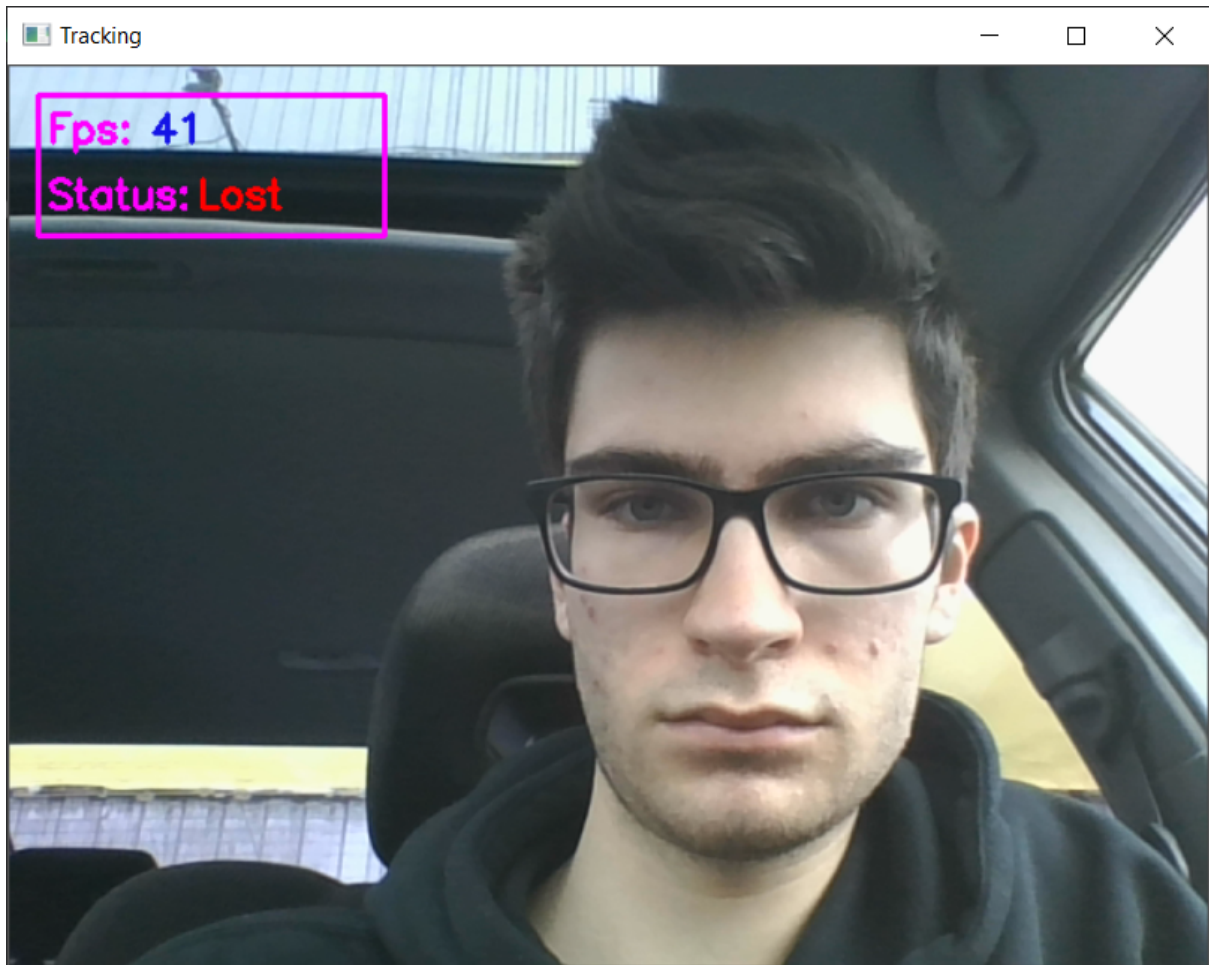
Також тут доробляється логіка програми. Наприклад додано статус втрати трекінгу а також лічильник тіков який знадобитися для рахування фреймрейту.

6. Лічильник кадрів

Також для відстеження навантажки на систему я додав лічильник кадрів:

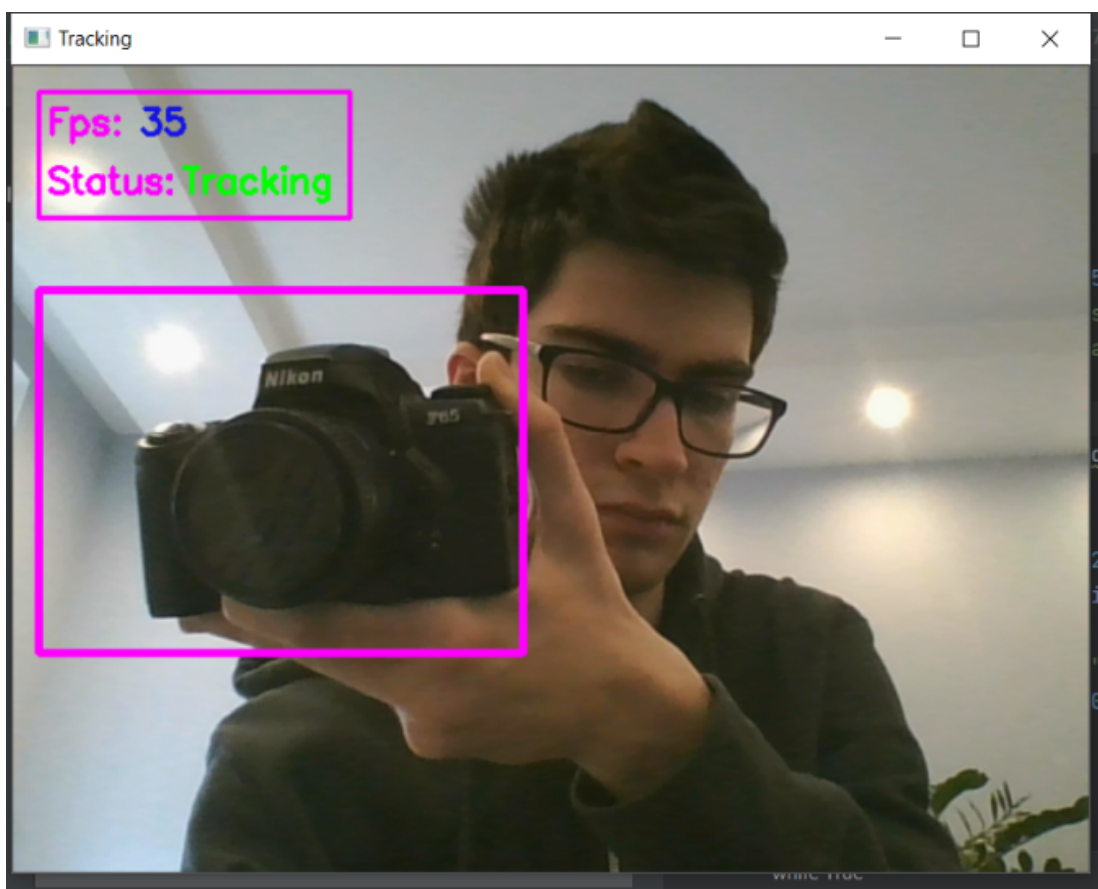
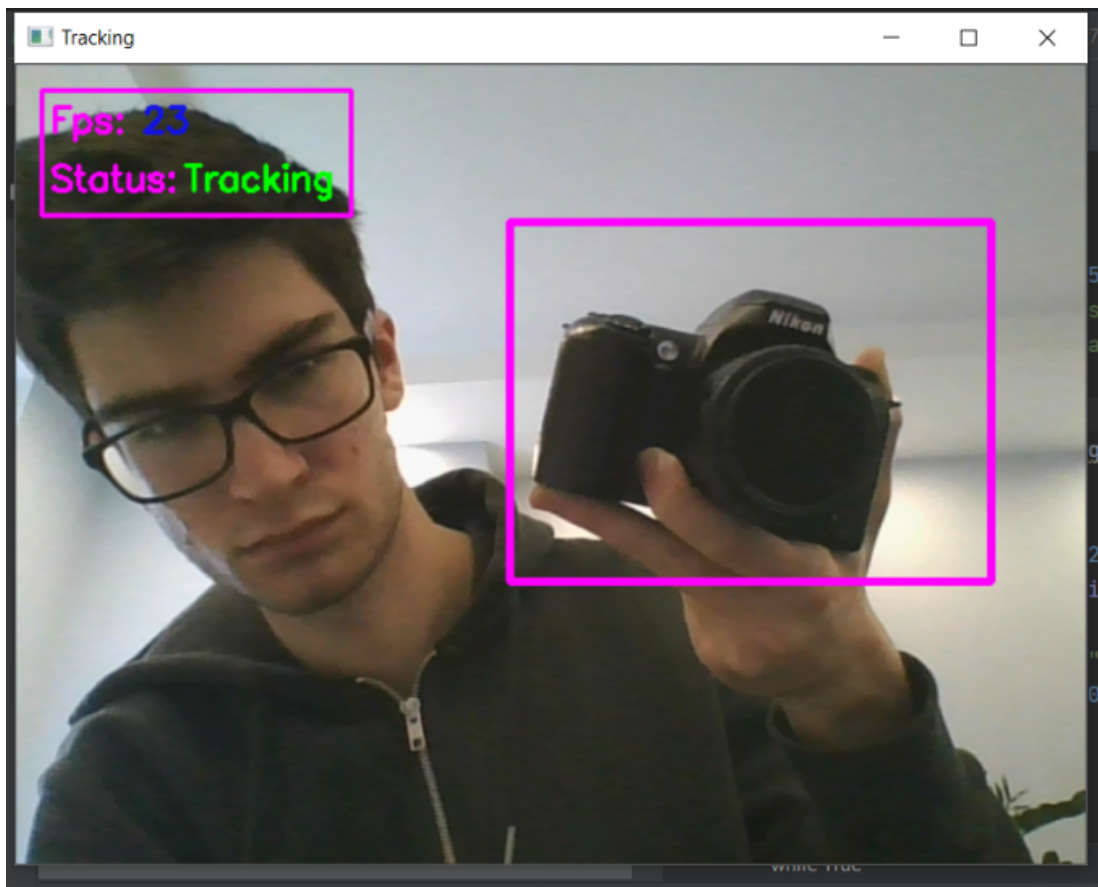
```
39
40 cv2.rectangle(img, (15, 15), (200, 90), (255, 0, 255), 2)
41 cv2.putText(img, "Fps:", (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 255), 2);
42 cv2.putText(img, "Status:", (20, 75), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 255), 2);
43
44
45 fps = cv2.getTickFrequency() / (cv2.getTickCount() - timer);
46 if fps>60: myColor = (20, 230, 20)
47 elif fps>20: myColor = (230, 20, 20)
48 else: myColor = (20, 20, 230)
49 cv2.putText(img, str(int(fps)), (75, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.7, myColor, 2);
```

Тепер вікно виводу має такий вигляд:



7.Тестовий запуск алгоритму

Спробувавши відстежити певний об'єкт отримуємо таке:



Висновок

Мною було розроблено систему пошуку обличчя на сцені. Для цього було використано бібліотеку алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом OpenCV, мову програмування Python.

Із плюсів використання бібліотеки OpenCV зазначено: проста та швидка установка, добре реалізована інтеграція з мовами програмування, швидкість роботи.

Література:

1. G. Bradski, A. Kaehler Learning OpenCV: Computer Vision with the OpenCV Library.— O'Reilly Media, Inc., 2008.— 580 p.
2. Кэлер А., Брэдски Г. Изучаем OpenCV. — М.: ДМК-Пресс, 2017. — 826 с.
3. Буэно, Суарес, Эспиноса. Обработка изображений с помощью OpenCV = Learning Image Processing with OpenCV. — М.: ДМК-Пресс, 2016. — 210 с.
4. Прохоренко Н. А. OpenCV и Java. Обработка изображений и компьютерное зрение. — СПб.: БХВ-Петербург, 2018. — 320 с.: ил.

Електронні ресурси:

<http://opencv.willowgarage.com/documentation/cpp/index.html>
<https://opencv.org/>
<https://docs.opencv.org/>
<http://robocraft.ru/tag/OpenCV/>

<https://github.com/Taska23/ROAS2>
<https://t.me/Taska2399>