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

ЗВІТ
з модульної контрольної роботи
з навчальної дисципліни «Технології Computer Vision»

Виконала:
Студентка 3 курсу
Навчальної групи ІС-12
Мельникова К.О.

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О.О.

Київ 2024

I. Білет № 3

II. Завдання:

Білет №3

1. Методи стеження за динамічними об’єктами.
2. Методи побудови векторних структур із растрових.
3. Розробити програмний скрипт з геометричними перетвореннями 2D графічного об’єкту.

III. Результати виконання модульної контрольної роботи.

3.1. Відповідь на теоретичне питання №1.

Відстеження динамічних об’єктів (або object-tracking) – це процес визначення та аналізу положення об’єкта в кожному кадрі відеопотоку.

До основних методів object-tracking-у відносять:

- Аналіз кольорових просторів. Процес, що полягає у визначені та аналізі особливостей виділеного сегменту гістограми зображення за параметрами та формою.
- Використання особливих точок - дескриптора зображень. Даний метод полягає у визначенні особливих точок сегменту зображення та аналізі їх дескрипторів на множині послідовних кадрів відеопотоку.
- Обчислення різниці між кадрами. Сутність даного методу полягає у аналізі структури послідовності кадрів та відзнак у сегменті кадру, включаючи аналіз кореляційних зав'язків.
- Віднімання фонових зображень. Даний метод полягає у аналізі структури послідовності кадрів та відзнак у сегменті кадру на підставі аналізу структури і властивостей фонових зображень

3.2. Відповідь на теоретичне питання №2.

Методи перетворення растрових зображень у векторні включають:

- трасування контурів (Визначення контурів на растровому зображенні та їх подальше перетворення у векторні лінії.)
- методики згладжування кривих (Застосування алгоритмів згладжування для отримання більш точних векторних кривих).

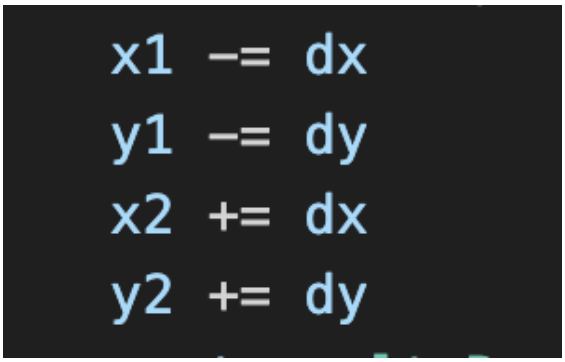
- розпізнавання форм (Використання алгоритмів машинного навчання для розпізнавання та перетворення окремих елементів зображення у векторні об'єкти).

3.3. Відповідь на практичне питання №3.

Розробимо програмний скрипт, що здійснює масштабування та переміщення з обертанням 2Д графічного об’єкту - прямокутника.

3.3.1. Математична модель.

Масштабування здійснюється за моделлю:

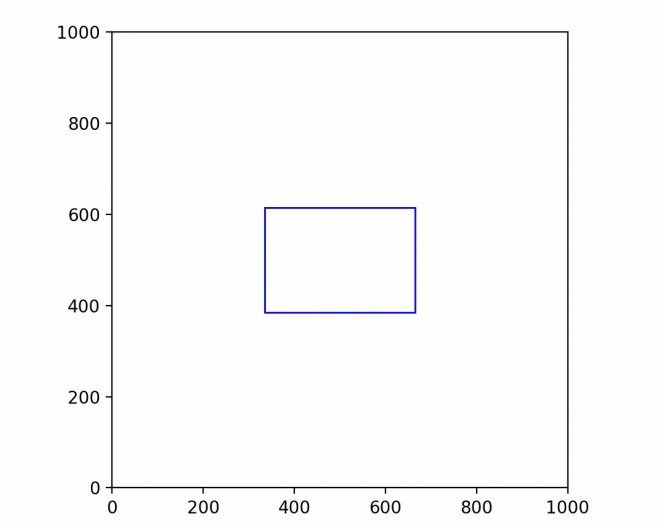


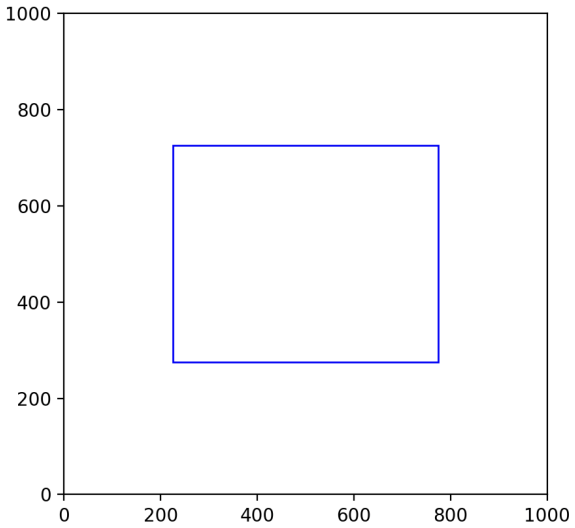
Переміщення та обертання:

```
center_x = (x1 + x3) / 2 + dx
center_y = (y1 + y3) / 2 + dy

vertices_matrix = np.array([[x1, y1, 1],
                            [x2, y2, 1],
                            [x3, y3, 1],
                            [x4, y4, 1]])

translation_matrix = np.array([[1, 0, -center_x],
                               [0, 1, -center_y],
                               [0, 0, 1]])

rotation_matrix = np.array([[np.cos(angle), -np.sin(angle), 0],
                            [np.sin(angle), np.cos(angle), 0],
                            [0, 0, 1]])

reverse_translation_matrix = np.array([[1, 0, center_x],
                                       [0, 1, center_y],
                                       [0, 0, 1]])

transformation_matrix = reverse_translation_matrix.dot(rotation_matrix).dot(translation_matrix)
updated_vertices = vertices_matrix.dot(transformation_matrix.T)
```

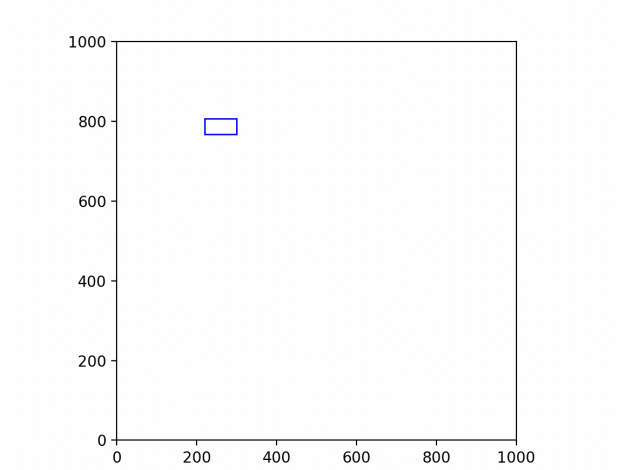
3.3.4. Результати роботи програми відповідно до завдання (допускається у формі скриншотів).

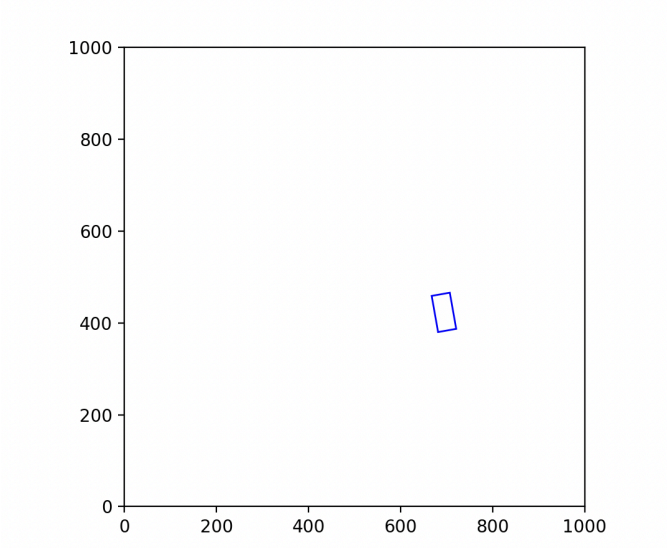
Масштабування:





Переміщення та обертання:





3.3.5.4. Програмний код, що забезпечує отримання результату (допускається у формі скриншотів).

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

xw, yw, st = 1000, 1000, 50
dx, dy = 10, 10
rect_width = 3 * st
rect_height = st

fig, ax = plt.subplots()
plt.xlim(0, xw)
plt.ylim(0, yw)
ax.set_aspect('equal')
```

```
x1 = (xw - rect_width) / 2
y1 = (yw - rect_height) / 2
x2 = x1 + rect_width
y2 = y1 + rect_height

rect = plt.Rectangle((x1, y1), rect_width, rect_height, edgecolor='blue',
facecolor='none')
ax.add_patch(rect)
plt.draw()
plt.pause(1)

for i in range(int(xw / st)):
   rect.set_edgecolor('white')
   x1 -= dx
   y1 -= dy
   x2 += dx
   y2 += dy
   rect = plt.Rectangle((x1, y1), x2-x1, y2-y1, edgecolor='blue', facecolor='none')
   ax.add_patch(rect)
   plt.draw()
   plt.pause(0.1)

plt.show()


fig, ax = plt.subplots()
plt.xlim(0, xw)
plt.ylim(0, yw)
ax.set_aspect('equal')
dx, dy = 50, 50
width, height = 80, 40
angle = np.radians(10)

x1, y1 = 50, yw - 50
x2, y2 = x1 + width, y1
x3, y3 = x2, y1 - height
x4, y4 = x1, y3

square = plt.Polygon([[x1, y1], [x2, y2], [x3, y3], [x4, y4]], edgecolor='blue',
facecolor='none')
ax.add_patch(square)
plt.draw()
plt.pause(1)
```

```
for i in range(100):
    square.set_edgecolor('white')
    center_x = (x1 + x3) / 2 + dx
    center_y = (y1 + y3) / 2 + dy

    vertices_matrix = np.array([[x1, y1, 1],
                                [x2, y2, 1],
                                [x3, y3, 1],
                                [x4, y4, 1]])

    translation_matrix = np.array([[1, 0, -center_x],
                                   [0, 1, -center_y],
                                   [0, 0, 1]])

    rotation_matrix = np.array([[np.cos(angle), -np.sin(angle), 0],
                                [np.sin(angle), np.cos(angle), 0],
                                [0, 0, 1]])

    reverse_translation_matrix = np.array([[1, 0, center_x],
                                           [0, 1, center_y],
                                           [0, 0, 1]])

    transformation_matrix =
reverse_translation_matrix.dot(rotation_matrix).dot(translation_matrix)
    updated_vertices = vertices_matrix.dot(transformation_matrix.T)

    x1, y1, _ = updated_vertices[0]
    x2, y2, _ = updated_vertices[1]
    x3, y3, _ = updated_vertices[2]
    x4, y4, _ = updated_vertices[3]

    square = plt.Polygon([[x1, y1], [x2, y2], [x3, y3], [x4, y4]], edgecolor='blue',
facecolor='none')
    ax.add_patch(square)
    plt.draw()
    plt.pause(0.1)

plt.show()
```