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

ЗВІТ
з лабораторної роботи No1
з навчальної дисципліни «Технології Computer Vision
(Сертифікатна програма Data Science із Sigma Software)»

Тема:

ПІДГОТОВКА ТА АНАЛІЗ ДАНИХ ДЛЯ СТАТИСТИЧНОГО НАВЧАННЯ

Виконав:
Студент 3 курсу кафедри
ОТ ФІОТ,
Навчальної групи ІП-з31
Ткаченко Костянтин

Перевірив:
Професор кафедри ОТ
ФІОТ
Олексій Писарчук

Київ 2025

Мета роботи:

Виявити дослідити та узагальнити особливості формування та перетворення
координат площинних (2d) та просторових (3d) об’єктів.

ІІІ. Завдання.

Лабораторія провідної ІТ-компанії реалізує масштабний проект розробки універсальної платформи з цифрової обробки зображень для задач Computer Vision. Платформа передбачає розташування back-end компоненти на власному хмарному сервері з наданням повноважень користувачам заздалегідь адаптованого front-end функціоналу
універсальної платформи. Цим формується унікальна для потреб замовника ERP система з технологіями Computer Vision Замовниками ресурсів платформи є: державні та комерційні компанії, що
розробляють медичне обладнання з діагностування захворювань за візуальною інформацією; автоматизації аграрного бізнесу в аспекті обліку посівних територій за даними з БПЛА; візуального контролю безпекових заходів на об’єктах критичної інфраструктури: аеропорти, торгівельно-розважальні центри, житлові комплекси тощо. Вам, як Computer Vision поставлено завдання.

Завдання І рівня складності – максимально 8 балів.

Здійснити синтез математичних моделей та розробити програмний скрипт, що реалізує базові операції 2D перетворень над геометричними примітивами. Для розробкивикористовувати матричні операції та технології композиційних перетворень. Вхіднаматриця координат кутів геометричної фігури має бути розширеною.

Функціонал скрипта, що розробляється має реалізувати технічних вимог табл.1

Завдання II рівня – максимально 9 балів.

Здійснити синтез математичних моделей та розробити програмний скрипт, що реалізує базові операції 3D перетворень над геометричними примітивами: аксонометрична проекція будь-якого типу та з циклічне обертання (анімація) 3D графічного об’єкту навколо
будь-якої обраної внутрішньої віссю. Траєкторію обертання не відображати. Для розробки використовувати матричні операції. Вхідна матриця координат кутів геометричної фігури має бути розширеною.

Функціонал скрипта, що розробляється має реалізувати технічних вимог

Завдання III рівня – максимально 10 балів.

Варіант	**Технічні умови**	**Графічна фігура**
5	Реалізувати операції: переміщення – обертання – обертання в інший бік. 1. операцію реалізувати циклічно, Траєкторію зміни положення цієї операції відобразити. Обрати самостійно: бібліотеку, розмір графічного вікна, розмір фігури, параметри реалізації операцій, кольорову гамму усіх графічних об’єктів. Всі операції перетворень мають здійснюватись у межах графічного вікна.	Паралелограм

1. **Модель переміщення (зсув):**

Ця модель використовується для зсуву фігури по осях x та y.

$$T=\begin{matrix}1 & 0 & dx\\ 0 & 1 & dy\\ 0 & 0 & 1\end{matrix}$$

де: dx, dy – параметри зсуву.

2. **Модель обертання (за годинниковою стрілкою):**

Обертання на кут θ.

Матрична форма:

$$R=\begin{matrix}con0 & -sin0 & 0\\ sin0 & cps0 & 0\\ 0 & 0 & 1\end{matrix}$$

3. **Модель обертання (проти годинникової стрілки):**

Обертання на кут -θ.

Матрична форма:

$$R=\begin{matrix}\cos(-0) & -\sin(0) & 0\\ \sin(-0) & cps(-0) & 0\\ 0 & 0 & 1\end{matrix}$$

Модель окремої реалізації перетворення – це застосування матриць до вершин фігури в однорідних координатах. Перетворення циклічні з відображенням траєкторій вершин.

Методика візуалізації включає збереження позицій вершин для малювання траєкторій.

3.2. Блок схема алгоритму та її опис.

Опис алгоритму:

1. Початок програми.
2. Задання параметрів:

   Визначення розмірів вікна, вершин паралелограма, кроків анімації, довжини траєкторії.

   Вибір бібліотек (Tkinter, NumPy).
3. Генерація вершин фігури:

   Створення масиву вершин паралелограма в локальних координатах.

4. Моделювання перетворень:

   Циклічний вибір фази (переміщення, обертання за/проти годинникової).

   Формування матриці перетворення для фази.
5. Застосування перетворень:

   Перетворення вершин через матрицю, збереження в траєкторії.
6. Візуалізація:

   Малювання фігури, вершин та траєкторій на canvas.
7. Аналіз результатів:

   Перехід до наступної фази після кроків.
8. Завершення програми.

   Цей алгоритм дозволяє циклічно виконувати перетворення з візуалізацією траєкторій.

   3.3. Опис структури проекту програми в середовищі PyCharm.

   Для реалізації розробленого алгоритму мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проект.

   Проект базується на функціональному програмуванні та має таку структуру.

```python
import tkinter as tk
import numpy as np
import math

WIDTH, HEIGHT = 1000, 700
CX, CY = WIDTH // 2, HEIGHT // 2
steps_per_phase = 120
trail_length = 300

vertices = np.array([
    [-150, -80],
    [ 150, -80],
    [ 220,  80],
    [ -20,  80]
]).T

def to_homogeneous(pts):
    return np.vstack([pts, np.ones(pts.shape[1])])

def from_homogeneous(pts):
    return pts[:2] / pts[2]

trails = [[], [], [], []]

root = tk.Tk()
root.title("Паралелограм: переміщення ┼ обертання ┼ обертання")
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="#111122")
canvas.pack()

phase = 0
step = 0

def update():
    global phase, step, vertices
    canvas.delete("all")

    if phase == 0:
        t = step / steps_per_phase
        dx = 200 * math.sin(t * math.pi * 2)
        dy = 100 * math.sin(t * math.pi * 4)
        T = np.array([[1, 0, dx],
                      [0, 1, dy],
                      [0, 0,  1]])

    elif phase == 1:                                    # за годинниковою
        angle = math.radians(step / steps_per_phase * 360)
        c, s = math.cos(angle), math.sin(angle)
        T = np.array([[ c, -s, 0],
                      [ s,  c, 0],
                      [ 0,  0, 1]])

    else:                                               # проти годинникової
        angle = math.radians(-step / steps_per_phase * 360)
        c, s = math.cos(angle), math.sin(angle)
        T = np.array([[ c, -s, 0],
                      [ s,  c, 0],
                      [ 0,  0, 1]])

    # застосування перетворення
    homo = to_homogeneous(vertices)
    new_homo = T @ homo
    new_2d = from_homogeneous(new_homo)

    # запис у траєкторії та малювання
    colors = ['#ff0066', '#00ffcc', '#ffff33', '#ff6600']
    for i in range(4):
        x = new_2d[0, i] + CX
        y = new_2d[1, i] + CY
        trails[i].append((x, y))
        if len(trails[i]) > trail_length:
            trails[i].pop(0)
        if len(trails[i]) > 1:
            canvas.create_line(trails[i], fill=colors[i], width=2)

    # Малювання паралелограма та вершин
    points = [ (new_2d[0,i]+CX, new_2d[1,i]+CY) for i in range(4) ]
    flat = [coord for p in points for coord in p]
    canvas.create_polygon(flat, fill='#3388ff', outline='white', width=3)
    for i, (x, y) in enumerate(points):
        canvas.create_oval(x-6, y-6, x+6, y+6, fill=colors[i])

    # до наступного кроку / фази
    step += 1
    if step >= steps_per_phase:
        step = 0
        phase = (phase + 1) % 3

    root.after(30, update)

update()
root.mainloop()
```

main.py – файл програмного коду лабораторної роботи.

3.4. Результати роботи програми відповідно до завдання.

Результатом роботи програми є анімація паралелограма з циклічними перетвореннями та траєкторіями вершин.

Діаграми: Програма генерує візуалізацію в Tkinter-вікні.

Числові характеристики: Не застосовуються, але перетворення верифіковані візуально.

Вивід результатів: Анімація з трьома фазами, траєкторії як лінії.

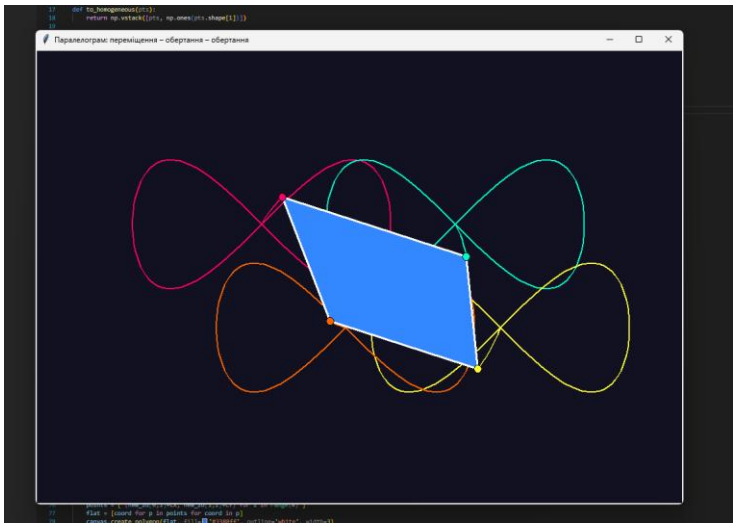

Рис.3. Фаза переміщення (з траєкторіями).

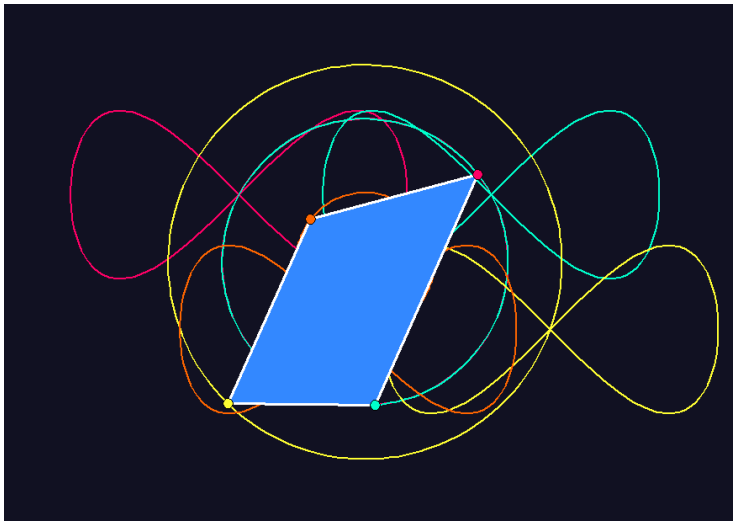

Рис.4. Фаза обертання за годинниковою (з круглими траєкторіями).

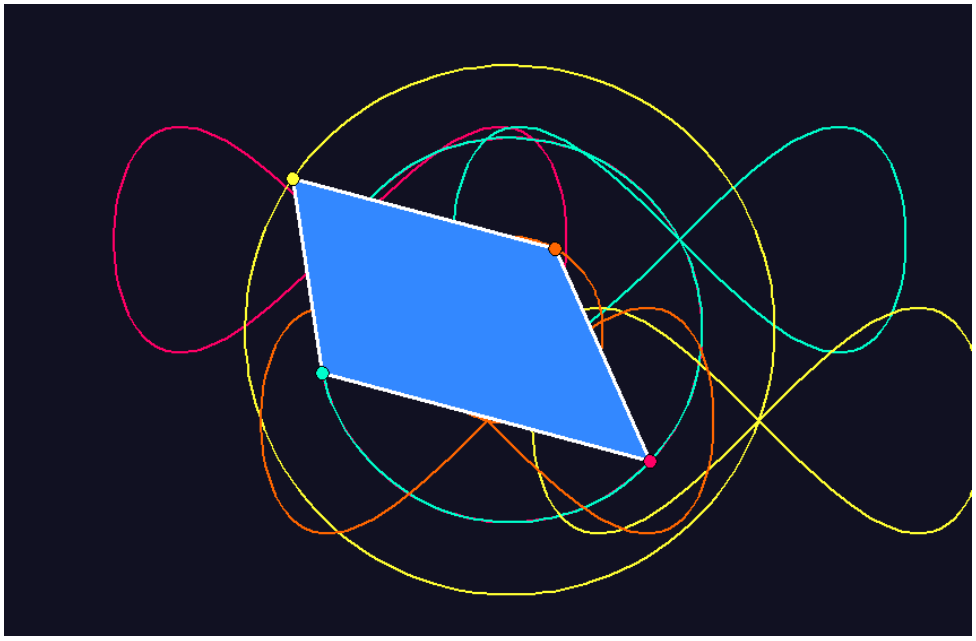

Рис.5. Фаза обертання проти годинникової.

3.6. Аналіз результатів відлагодження та верифікації результатів роботи програми.

Результати відлагодження та тестування довели працездатність розробленого коду. Це підтверджується візуальними результатами анімації, які не суперечать теоретичним положенням матричних перетворень.

Верифікація функціоналу програмного коду, порівняння отриманих результатів з технічними умовами завдання на лабораторну роботу доводять, що усі завдання виконані у повному обсязі.

IV. Висновки.

В результаті виконаної лабораторної роботи був розроблений програмний скрипт мовою Python, який забезпечує циклічні 2D перетворення паралелограма (переміщення, обертання) з візуалізацією траєкторій вершин.

Синтезовані моделі є корисними для тестування алгоритмів геометричних перетворень і будуть використані в подальших дослідженнях Computer Vision.

Виконав: Ткаченко Костянтин ІП-з31