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

ЗВІТ
з лабораторної роботи №3
з навчальної дисципліни «Computer Vision»

Тема:

ДОСЛІДЖЕННЯ АЛГОРИТМІВ ФОРМУВАННЯ ТА ОБРОБКИ РАСТРОВИХ ЦИФРОВИХ ЗОБРАЖЕНЬ

Виконав:
Студент 3 курсу кафедри ІПІ ФІОТ,
Навчальної групи ІП-11
Панченко С.В.

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О.О.

I. Мета роботи:

Виявити дослідити та узагальнити особливості реалізації алгоритмів формування та обробки векторних цифрових зображень на прикладі застосування алгоритмів інтерполяції, апроксимації та згладжування складних 3D растрових об’єктів та застосування технологій видалення невидимих граней та ребер.

II. 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 систем з технологіями Computer Vision. Змовниками 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 ІІІ рівня – максимально 9 балів.

Здійснити виконання завдання лабораторної роботи із застосуванням алгоритму інтерполяції для побудови векторного зображення 2D, 3D графічного об’єкту.

Здійснити виконання завдання лабораторної роботи із застосуванням алгоритму інтерполяції для побудови векторного зображення 2D, 3D графічного об’єкту та алгоритму видалення невидимих ліній та поверхонь.

Здійснити виконання завдання ІІ рівня складності – програмний скрипт No1.
Реалізувати розробку програмного скрипта No2, що реалізує виділення контору обраного об’єкту на цифровому растровому зображенні. За необхідності передбачити корекцію кольору цифрового растрового зображення для покращення якості виділення контору обраного об’єкту.

6	Відображення 3D фігури реалізується з використанням аксонометричної проекції будь- якого типу. Обрати самостійно: бібліотеку, розмір графічного вікна, розмір фігури, динаміку зміни положення фігури, кольорову гамму графічного об’єкту. Всі	Піраміда з чотирикутною основою. Метод інтерполяції: кривими Безьє. Метод видалення невидимих ліній та поверхонь: алгоритм плаваючого обрію.

	операції перетворень мають здійснюватися у межах графічного вікна.	

III. Результати виконання лабораторної роботи.

3.1. Синтезована математична модель перетворень графічних об’єктів відповідно до індивідуального завдання.

Синтезована математична модель перетворень графічних об'єктів відповідно до завдання

Відповідно до умов задачі синтезовано математичну модель операцій над структурою вхідного графічного об'єкту - піраміди з чотирикутною основою.

Модель реалізує операцію побудови векторного зображення 3-вимірної піраміди з чотирикутною основою як аксонометричну проекцію з використанням кривих Безьє як алгоритму інтерполяції.

Криві Безьє – це параметричні криві, які використовуються в комп'ютерній графіці для моделювання гладких кривих. Вони задаються набором контрольних точок, які визначають форму кривої.

$$B(t) = \sum_{i=0}^{n} P_i \binom{n}{i} (1-t)^{n-i} t^i \quad t \in [0,1]$$

Для побудови кривої Безьє n-го степеня використовується наступна формула:,

де P_i - це контрольні точки,

$\binom{n}{i}$ - біноміальний коефіцієнт, і t - параметр, що змінюється від 0 до 1.

У даному випадку, для побудови векторного зображення піраміди використовуються криві Безьє другого порядку (квадратичні криві Безьє) для інтерполяції між вершинами основи та вершиною піраміди. Кожна грань піраміди будується як окрема крива Безьє, що проходить через дві вершини основи та вершину піраміди.

Для видалення невидимих ліній та поверхонь використовується **алгоритм плаваючого обрію**. Цей алгоритм базується на ідеї, що видимість грані визначається її орієнтацією відносно спостерігача. Якщо нормаль до грані спрямована в бік спостерігача, то грань вважається видимою, інакше - невидимою.

Алгоритм плаваючого обрію можна описати наступними кроками:

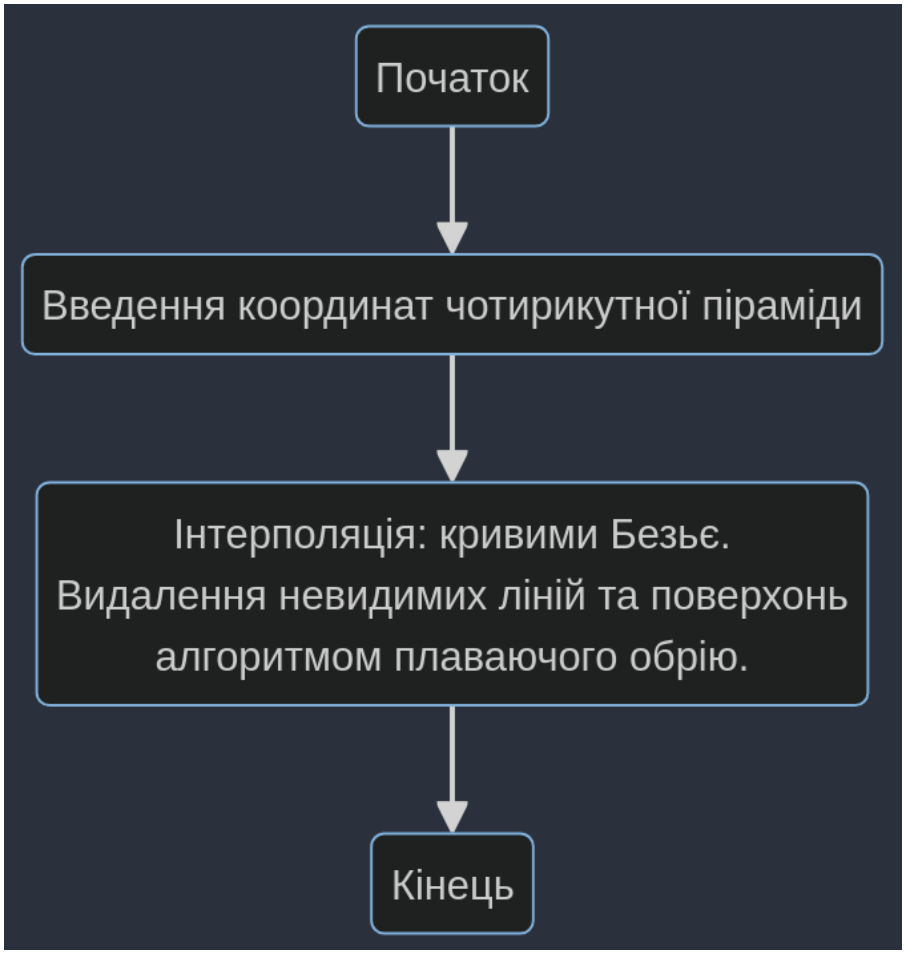
1. Для кожної грані піраміди обчислити нормаль до неї.
2. Визначити вектор напрямку від спостерігача до будь-якої точки на грані.

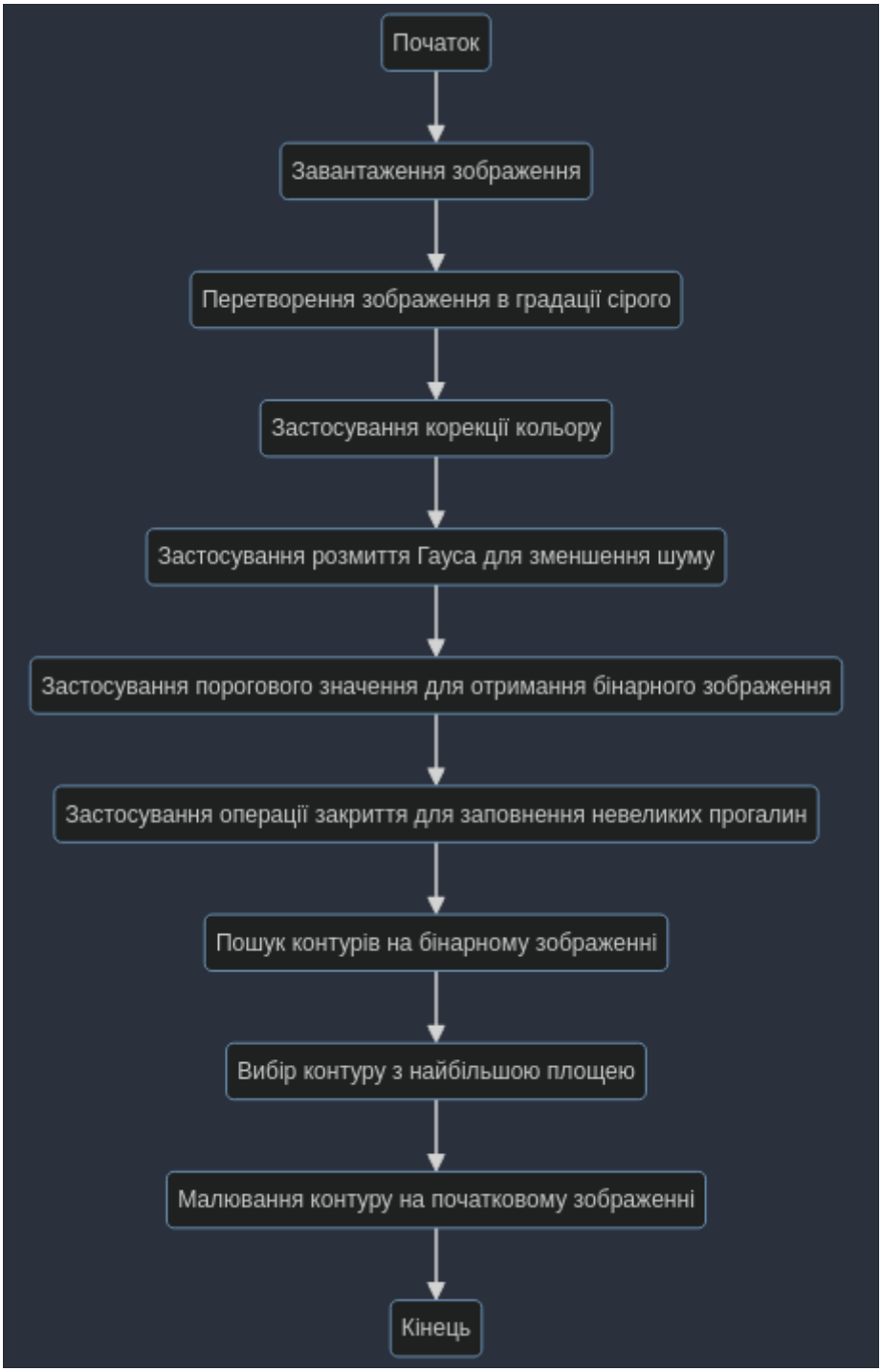
3. Обчислити скалярний добуток між нормаллю та вектором напрямку.
4. Якщо скалярний добуток менший за 0, то грань вважається видимою, інакше - невидимою.
5. Відобразити лише видимі грані піраміди.

Таким чином, синтезована математична модель включає використання кривих Безьє для інтерполяції та побудови векторного зображення піраміди, а також алгоритм плаваючого обрію для видалення невидимих ліній та поверхонь. Ця модель дозволяє реалістично відобразити 3-вимірну піраміду з чотирикутною основою як аксонометричну проекцію з урахуванням видимості граней.

3.2. Блок схема алгоритму та її опис.

Блок-схема розв’язку матиме вигляд на рис. 1 та 2.





Програма запускається при виконанні файла у середовищі PyCharm, і виконується автоматично і одноразово. Виконується векторного зображення ортогональної проекції чотирикутної піраміди у 3D. Також виконується пошук контурів на зображенні.

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 лінійній бізнес-логіці функційного програмування програмування та має таку структуру.

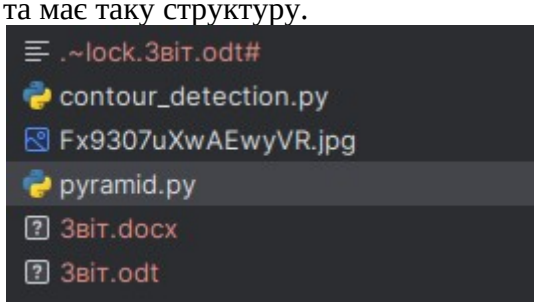


Рис.2. Структура проекту.

pyramid.py – файл з програмою для роботи з пірамідою
contour_detector.py – файл для виявлення контурів
Звіт.docx – файл зі звітом

3.4. Результати роботи програми відповідно до завдання.

Програма будує піраміду з чотирикутною основою, використовуючи інтерполяцію кривими Безьє. Потім застосовується алгоритм плаваючого обрію для видалення невидимих граней з точки спостереження. Результат відображається у 3D графіку за допомогою бібліотеки Matplotlib.

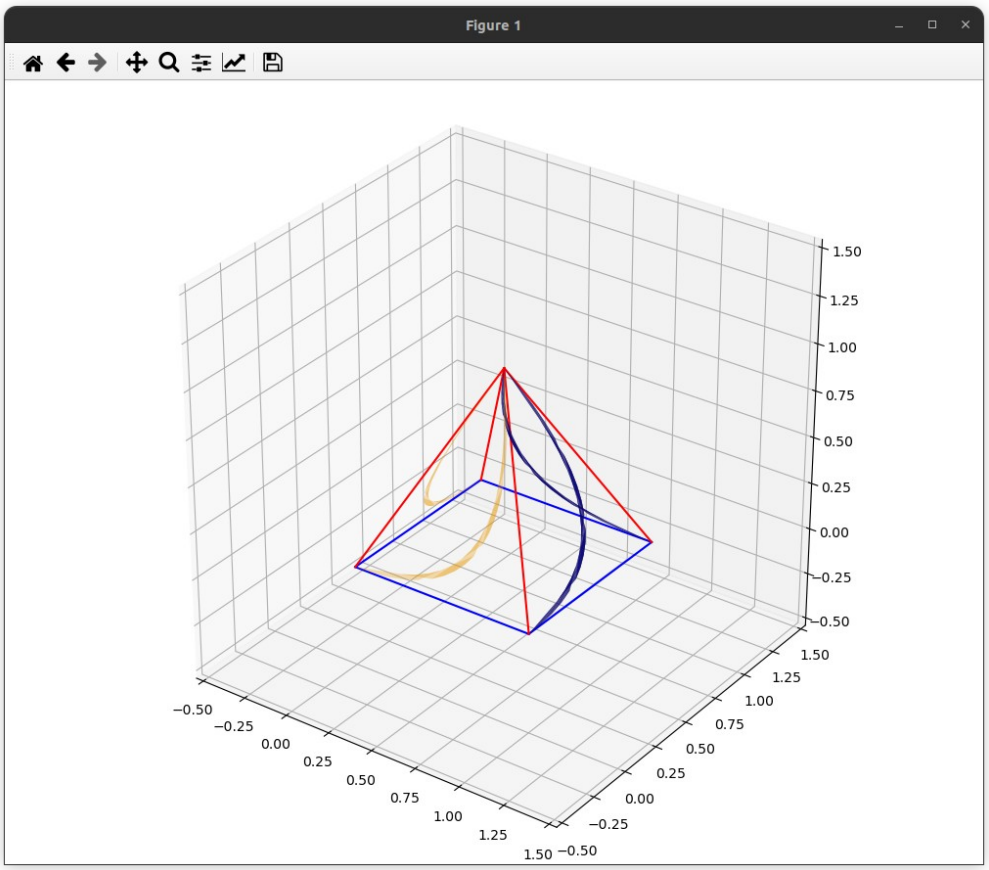


Рис 3 - Вигляд з боку

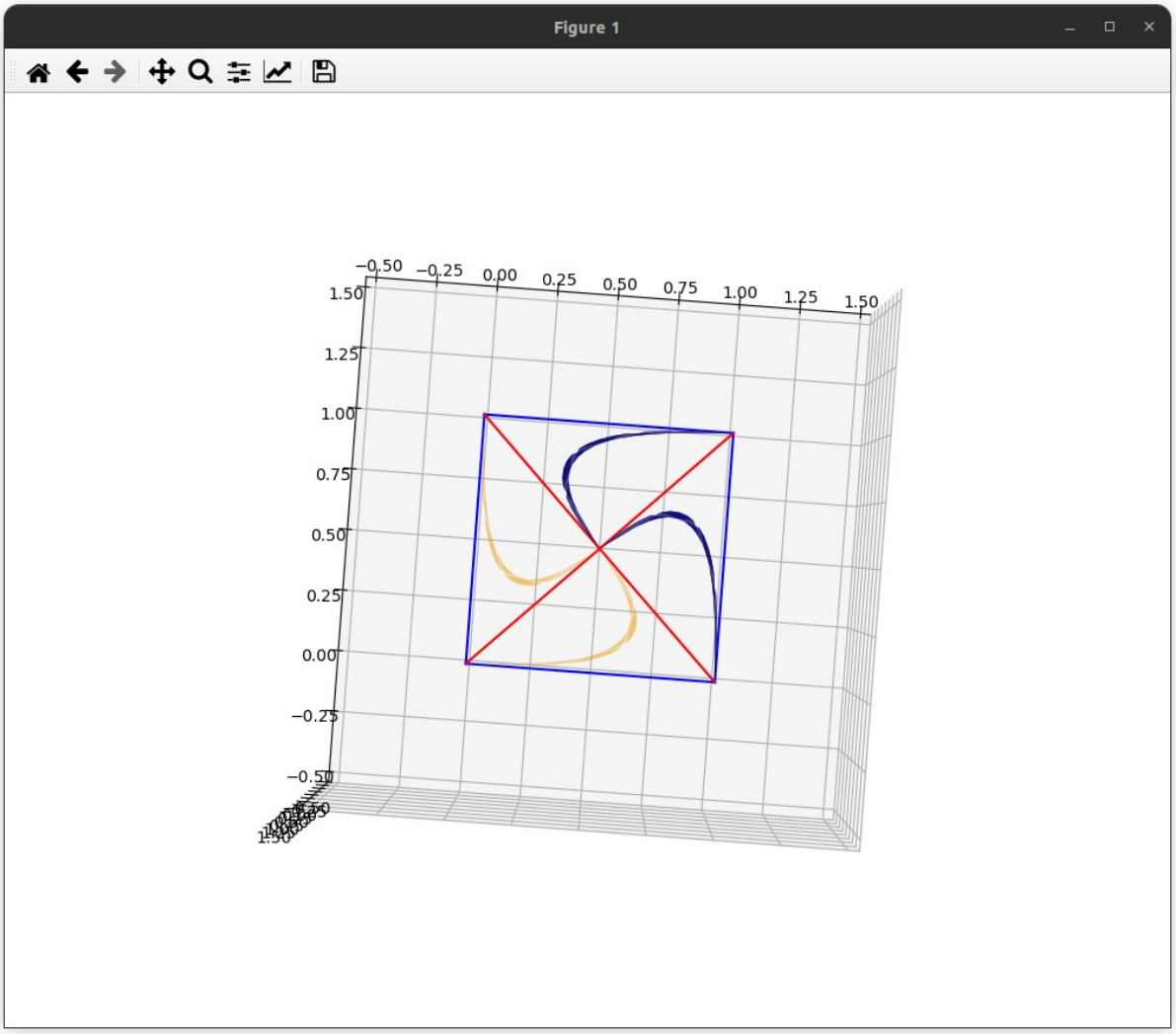


Рис. 4 - Вигляд згори

Як бачимо на рисунку нище, контури виявилися на зачісці.

Рис.5 - Виявлення контурів на зображенні.

3.5. Програмний код.

Програмний код послідовно реалізує алгоритм рис.1 та спрямовано на отримання результатів, поданих на рис.2-10.

Для спрощення програмного коду і раціоналізації обчислень застосовано функціональні механізми створення підпрограм.

При цьому використано можливості Python бібліотек: `numpy, matplotlib.`

```
# pyramid.py
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

```python
def bezier_curve(points, num_points):
    t = np.linspace(0, 1, num_points)
    curve = np.zeros((num_points, 3))
    n = len(points) - 1
    for i in range(num_points):
        for j in range(n+1):
            curve[i] += np.array(points[j]) * (np.math.comb(n, j) * (1-t[i])**(n-j) * t[i]**j)
    return curve

def create_pyramid(base_points, apex, num_points):
    pyramid = []
    for i in range(len(base_points)):
        points = [base_points[i], base_points[(i+1)%len(base_points)], apex]
        pyramid.append(bezier_curve(points, num_points))
    return pyramid

def floating_horizon(pyramid, view_point):
    visible_faces = []
    for face in pyramid:
        normal = np.cross(face[1]-face[0], face[-1]-face[0])
        if np.dot(normal, face[0]-view_point) < 0:
            visible_faces.append(face)
    return visible_faces

# Параметри піраміди
base_points = [(0, 0, 0), (1, 0, 0), (1, 1, 0), (0, 1, 0)]
apex = (0.5, 0.5, 1)
num_points = 20

# Створення піраміди
pyramid = create_pyramid(base_points, apex, num_points)

# Точка спостереження
view_point = (1.5, 1.5, 1.5)

# Видалення невидимих граней
visible_pyramid = floating_horizon(pyramid, view_point)

# Відображення піраміди з видимими та невидимими гранями
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

for face in pyramid:
    ax.plot_surface(face[:, 0].reshape(4, 5),
                    face[:, 1].reshape(4, 5),
                    face[:, 2].reshape(4, 5),
                    color='orange', alpha=0.3)

for face in visible_pyramid:
    ax.plot_surface(face[:, 0].reshape(4, 5),
                    face[:, 1].reshape(4, 5),
                    face[:, 2].reshape(4, 5),
                    color='blue', alpha=0.7)

# Plot the base
base_edges = [[base_points[i], base_points[(i + 1) % len(base_points)]] for i in
range(len(base_points))]
for edge in base_edges:
```

```
    ax.plot3D(*zip(*edge), color="b")

# Plot the sides
for point in base_points:
    ax.plot3D(*zip(point, apex), color="r")



ax.set_xlim(-0.5, 1.5)
ax.set_ylim(-0.5, 1.5)
ax.set_zlim(-0.5, 1.5)
ax.set_box_aspect((1, 1, 1))

plt.tight_layout()
plt.show()
```

```
# contour_detector.py
import cv2
import numpy as np

# Завантаження зображення
image = cv2.imread('Fx9307uXwAEwyVR.jpg')

# Перетворення зображення на відтінки сірого
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Застосування фільтра для зменшення шуму
blurred = cv2.GaussianBlur(gray, (5, 5), 0)

# Бінаризація зображення
_, thresh = cv2.threshold(blurred, 127, 255, cv2.THRESH_BINARY)

# Виділення контурів
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Вибір контуру з найбільшою площею (припускаємо, що це об'єкт)
max_contour = max(contours, key=cv2.contourArea)

# Намалювати контур на оригінальному зображенні
cv2.drawContours(image, [max_contour], 0, (0, 255, 0), 2)

# Відобразити результат
cv2.imshow('Contour Detection', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.6. Аналіз результатів відлагодження та верифікації результатів роботи програми.

Результати від лагодження та тестування довели працездатність розробленого коду.

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

IV. Висновки.

У ході лабораторної роботи було синтезовано математичну модель та розроблено програмний скрипт на Python для реалізації векторних алгоритмів над 2D, 3D графічними примітивами та з цифровими зображеннями. Використано метод інтерполяції кривими Безьє для побудови піраміди та алгоритм плаваючого обрію для видалення невидимих ліній. Також реалізовано виділення контуру об'єкту на цифровому зображенні з використанням бібліотеки OpenCV. Результати демонструють ефективність використаних підходів та набуті практичні навички у синтезі математичних моделей та розробці програмних скриптів для задач комп'ютерної графіки та обробки зображень.

Виконав: студент Панченко С.В.