



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №8
З дисципліни «Технології Computer Vision»
**ДОСЛДЖЕННЯ ТЕХНОЛОГІЙ ТРИВИМІРНОЇ РЕКОНСТРУКЦІЇ
ОБ'ЄКТІВ ЗА ЦИФРОВИМИ ЗОБРАЖЕННЯМИ**

Виконала
студент кафедри
ІСТ ФІОТ,
групи ІА-12:
Яковенко Д. О.

Перевірив:
пос. Баран Д. Р.

I. Мета:

Дослідити методологію і технології реконструкції 3D просторових об'єктів за їх 2D зображеннями методами багатовидової (стерео / сигнатурна) обробки.

II. Завдання:

Реалізація проекту триває та спрямовано на збільшення функціональності програмної компоненти. Лабораторія провідної ІТ-компанії реалізує масштабний проект розробки універсальної платформи з цифрової обробки зображенень 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 Visio.

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.

I рівень складності – максимально 8 балів.

Організувати та реалізувати роботу стереопарі та отримати цифрове статичне зображення самостійно обраного об'єкту із двох каналів з різними значеннями кутового ракурсу. Або обрати із відкритих джерел результати роботи стереопарі. Здійснити 3D реконструкцію обраного об'єкту та дослідити якість результату від параметрів стереопарі: база, ракурс на об'єкт (за умов наявності стереопарі, або відомих параметрів, що супроводжують відкриті джерела даних від стереопарі).

II рівень складності – максимально 9 балів.

Реалізувати умови завдання I рівня складності для кількості камер більше 2-х у багатовидовій (мультіканальній) системі відеоспостереження.

III. Результати виконання лабораторної роботи.

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

Відповідно до умов задачі синтезовано математичну модель, яка включає в себе декілька етапів обробки відео з трьох камер (MacBook та двох iPhone) та подальшу генерацію 3D-точкового хмари з стереопарі зображень. Для цього використовується бібліотека OpenCV, яка дозволяє отримувати кадри з різних джерел відео. Вихідний код налаштовує відеопотоки з вбудованої камери MacBook та двох камер iPhone, підключених через додаток **Cam Studio**. Отримані кадри з кожної камери відображаються у реальному часі. Користувач може переглядати відеопотоки з трьох різних джерел одночасно, що забезпечує зручний моніторинг.

Далі використовується пара зображень, що збережені у PNG форматі.

Використовується алгоритм стерео-співставлення StereoSGBM для обчислення карти глибини. Після отримання карти глибини, виконується реконструкція 3D-точок за допомогою функції `cv2.reprojectImageTo3D`, яка трансформує карту глибини в тривимірний простір, використовуючи матрицю Q , що враховує параметри камери. Згенеровані 3D-точки разом з кольоровою інформацією зберігаються у файлі формату PLY, і потім цей файл відкривається за допомогою встановленого додатку на Macbook – Preview.

Нехай (x_L, y_L) і (x_R, y_R) — координати відповідних точок на лівому та правому зображеннях відповідно. **Диспаритет d** — це різниця координат відповідних точок на лівому і правому зображеннях: $d = x_L - x_R$

Абсолютне відхилення інтенсивностей (Sum of Absolute Differences, SAD):

$$SAD(x, y, d) = \sum_{(i,j) \in W} |I_L(x+i, y+j) - I_R(x+i-d, y+j)|$$

Квадратичне відхилення (Sum of Squared Differences, SSD):

$$SSD(x, y, d) = \sum_{(i,j) \in W} (I_L(x+i, y+j) - I_R(x+i-d, y+j))^2$$

Нормалізована кореляція (Normalized Cross-Correlation, NCC):

$$NCC(x, y, d) = \frac{\sum_{(i,j) \in W} (I_L(x+i, y+j) - \bar{I}_L)(I_R(x+i-d, y+j) - \bar{I}_R)}{\sqrt{\sum_{(i,j) \in W} (I_L(x+i, y+j) - \bar{I}_L)^2} \sqrt{\sum_{(i,j) \in W} (I_R(x+i-d, y+j) - \bar{I}_R)^2}}$$

3.2. Блок схема алгоритму та її опис.

Алгоритм програми починається з імпорту бібліотеки OpenCV. Спочатку визначається функція `'get_video_capture'`, яка приймає джерело відеопотоку як аргумент і відкриває його за допомогою `'cv2.VideoCapture'`. Якщо відеопотік не вдається відкрити, функція виводить повідомлення про помилку і повертає `'None'`, інакше повертає об'єкт відеозахоплення. Далі програма відкриває відеопотоки з трьох камер: вбудованої камери MacBook, першого iPhone та другого iPhone. Для цього викликається функція `'get_video_capture'` з відповідними аргументами. Потім перевіряється, чи всі камери успішно відкриті. Якщо хоча б одна камера не відкрилася, виводиться повідомлення про помилку, і програма завершується. Якщо всі камери відкриті успішно, програма входить у безкінечний цикл. У цьому циклі читаються кадри з кожної камери за допомогою методів `'read'` об'єктів відеозахоплення. Якщо читування кадрів з будь-якої камери не вдалося, виводиться повідомлення про помилку, і цикл переривається. Коли кадри зчитані успішно, вони відображаються у трьох окремих вікнах за допомогою функції `'cv2.imshow'`. Програма продовжує відображати кадри, доки користувач не натисне клавішу `'q'`. Перевірка на натискання клавіші здійснюється за допомогою функції `'cv2.waitKey'`. Коли користувач натискає клавішу `'q'`, цикл переривається. Програма звільняє ресурси, закриваючи всі відеопотоки за допомогою методів `'release'` об'єктів відеозахоплення та закриваючи всі вікна, створені функцією `'cv2.imshow'`, за допомогою `'cv2.destroyAllWindows'`. На цьому робота програми завершується.

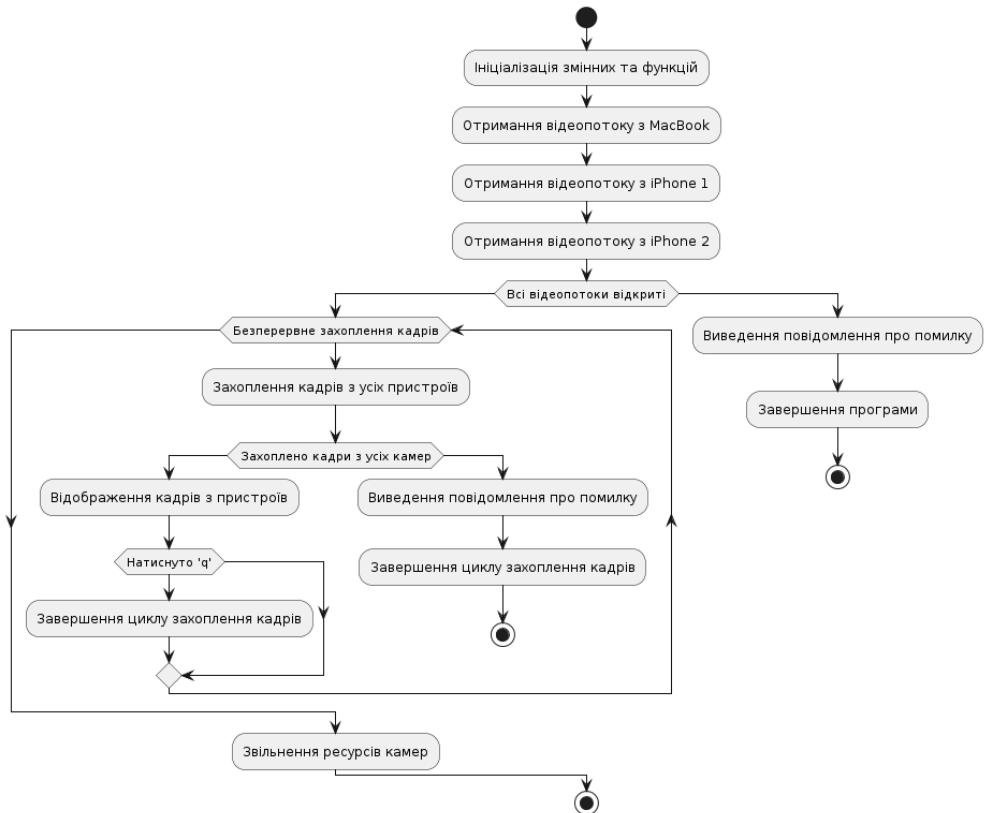


Рис.1. Блок-схема алгоритму програми.

Наступний алгоритм програми для відображення 3D об'єкту починається з імпорту необхідних бібліотек, таких як numpy, OpenCV та subprocess. Спочатку визначається заголовок файлу PLY у вигляді шаблону рядка, що містить метадані про формат файлу, кількість вершин і їх властивості, такі як координати та колір. Далі оголошується функція 'write_ply', яка записує дані тривимірної точки в файл PLY. Вона приймає як аргументи ім'я файлу, масив вершин і масив кольорів. Вершини і кольори спочатку перетворюються у відповідні форми, потім об'єднуються в один масив і записуються у файл у форматі PLY. У головному блоці програми відбувається завантаження зображень. Зображення завантажуються за допомогою функції 'cv2.imread' і зменшуються в розмірі за допомогою функції 'cv2.resize'. Далі налаштовуються параметри для обчислення диспаритету між двома зображеннями, що включають розмір вікна, мінімальний і максимальний диспаритет, а також різні параметри для налаштування методу блокового зіставлення. Програма обчислює карту диспаритету за допомогою методу 'StereoSGBM_create' з раніше налаштованими параметрами. Карта диспаритету нормалізується діленням на 16.0 для перетворення значень у плаваючу точку. Наступним кроком є створення тривимірної хмарки точок. Визначаються розміри зображення, а також фокусна відстань, яка розраховується як 80% від ширини зображення. Матриця Q визначається для перетворення координат зображення у тривимірні координати. Використовуючи функцію 'cv2.reprojectImageTo3D', карта диспаритету перетворюється на тривимірні точки. Потім обчислюється маска для виділення тільки тих точок, де диспаритет більше мінімального значення. Ці точки та відповідні кольори зберігаються у файл PLY за допомогою функції 'write_ply'.

Після збереження файлу програма запускає зовнішню команду для відкриття створеного файлу за допомогою системної програми. Одночасно у вікнах OpenCV відображаються зменшене ліве зображення і нормалізована карта диспаритету. Програма чекає на натискання будь-якої клавіші для закриття вікон і завершення роботи.

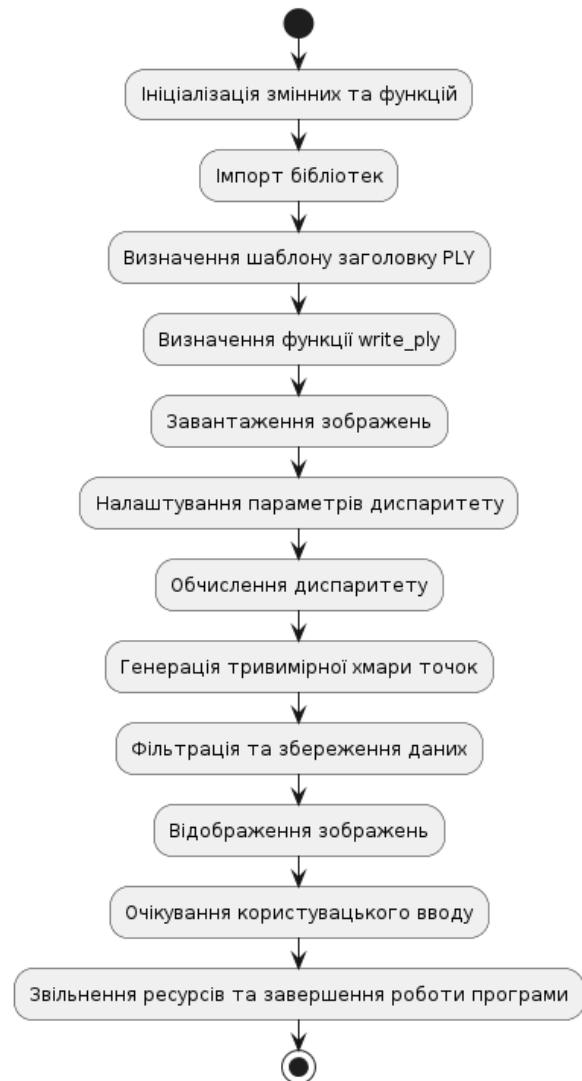


Рис.2. Блок-схема алгоритму для створення 3D об'єкту

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 функціонального програмування та має таку структуру.

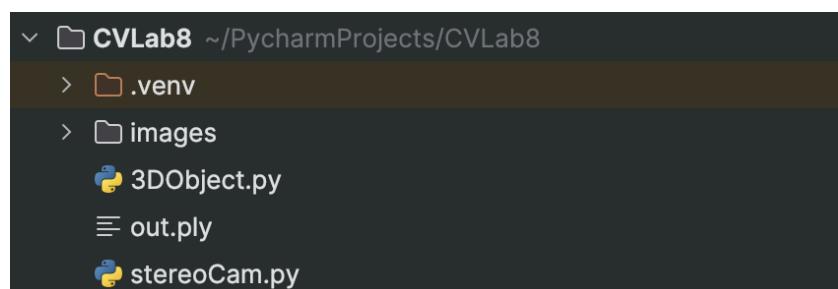


Рис.3. Структура проекту.

CVLab8 – головний каталог проекту

images – папка з зображеннями

3DObject.py – файл лабораторної роботи для побудови 3D об'єкту;

out.ply – файл у форматі PLY, який містить дані про хмару точок з обраною базовою лінією та кутом обертання

stereoCam.py – файл лабораторної роботи для захоплення кадрів з декількох відеокамер;

3.4. Результати роботи програми відповідно до завдання.

Результатом роботи програми є наступні зображення з трьох камер:

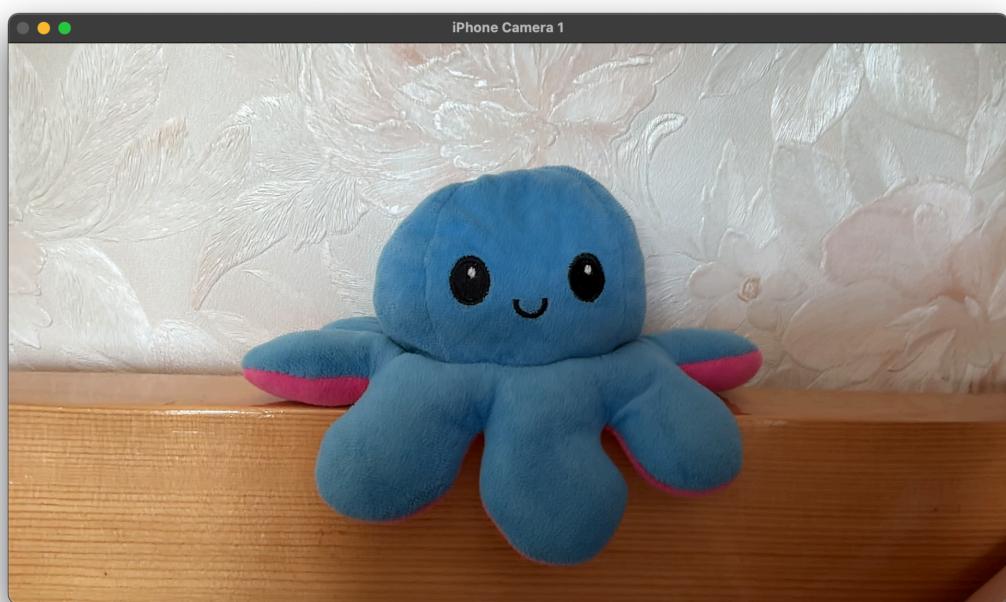




Рис.4. Результат виконання.

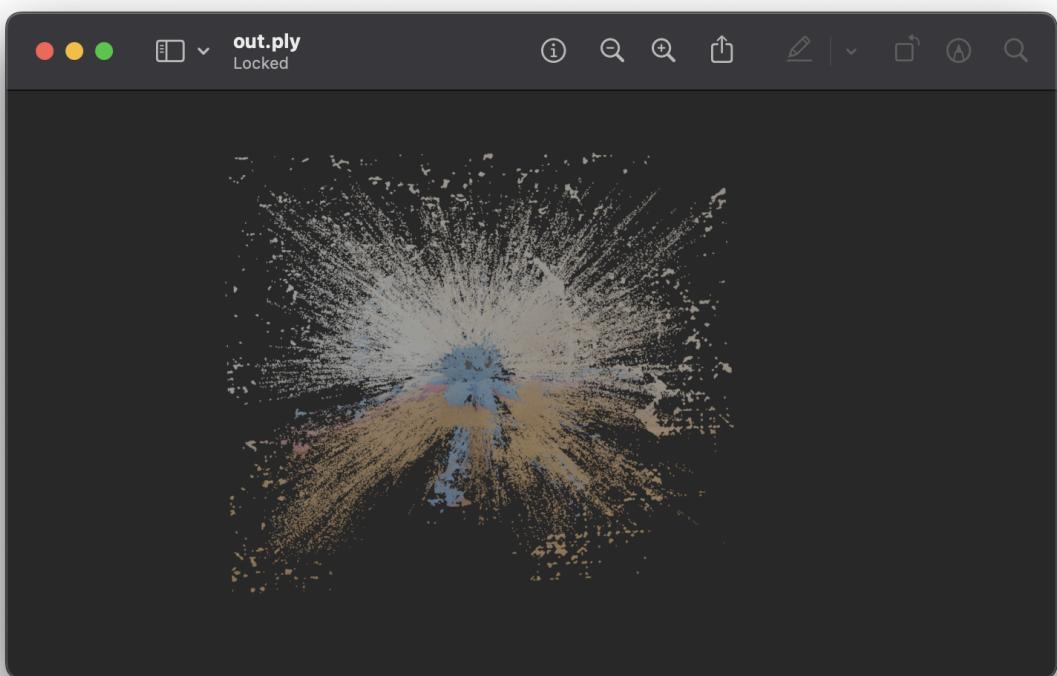
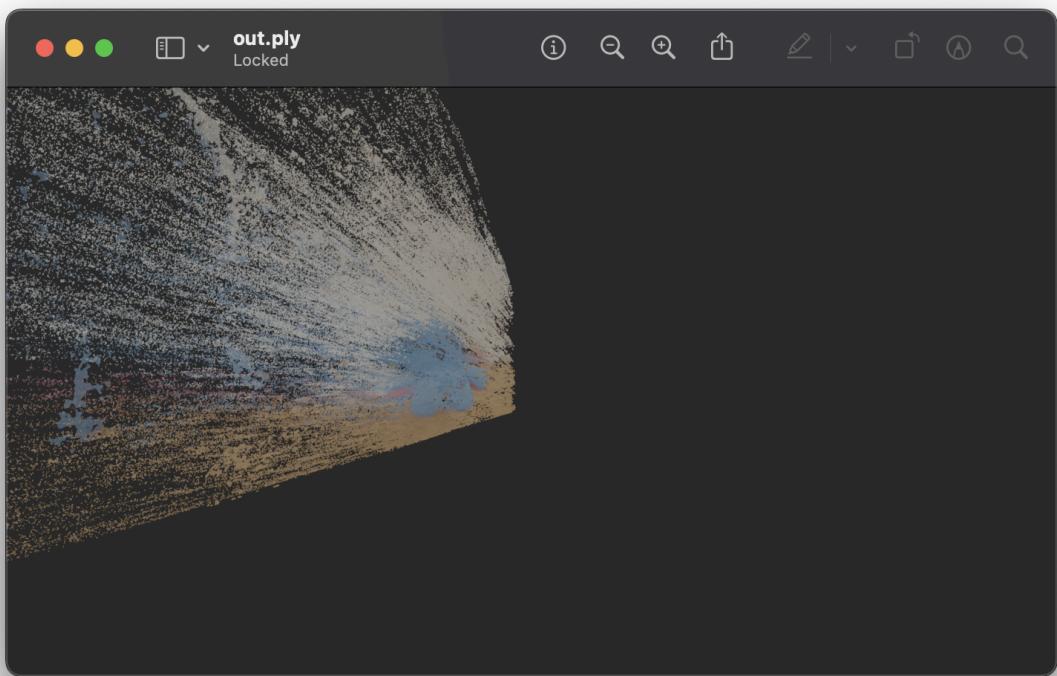


Рис.5. Результат виконання (ply-файл).

3.5. Програмний код.

Програмний код послідовно реалізує алгоритми для виконання завдання другого рівня складності.

При цьому використано можливості Python бібліотек: openCV, numpy, subprocess.

stereoCam.py:

```
import cv2

# Функція для отримання відеопотоку з камери
def get_video_capture(source):
    cap = cv2.VideoCapture(source)
    if not cap.isOpened():
        print(f"Cannot open video source {source}")
        return None
    return cap

# Відкрийте відеопотоки
macbook_camera = get_video_capture(0) # Вбудована камера ноутбука
iphone_camera_1 = get_video_capture(1) # Трансляція відео з першого
телефону
iphone_camera_2 = get_video_capture(0) # Трансляція відео з другого
телефону

# Перевірка, чи всі камери відкриті
if not macbook_camera or not iphone_camera_1 or not iphone_camera_2:
    print("One or more video sources could not be opened.")
    exit()

while True:
    # Читання кадрів з камер
    ret1, frame1 = macbook_camera.read()
    ret2, frame2 = iphone_camera_1.read()
    ret3, frame3 = iphone_camera_2.read()

    if not ret1 or not ret2 or not ret3:
        print("Failed to grab frames from one or more video sources.")
        break

    # Відображення кадрів
    cv2.imshow('MacBook Camera', frame1)
    cv2.imshow('iPhone Camera 1', frame2)
    cv2.imshow('iPhone Camera 2', frame3)

    # Вихід при натисканні клавіші 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Звільнення ресурсів
macbook_camera.release()
iphone_camera_1.release()
iphone_camera_2.release()
cv2.destroyAllWindows()
```

3DObject.py:

```
from __future__ import print_function
import numpy as np
import cv2
import subprocess
```

```

# PLY header template
ply_header = '''ply
format ascii 1.0
element vertex %(vert_num)d
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue
end_header
'''

# Function to write point cloud data to a PLY file
def write_ply(fn, verts, colors):
    verts = verts.reshape(-1, 3)
    colors = colors.reshape(-1, 3)
    verts = np.hstack([verts, colors])
    with open(fn, 'wb') as f:
        f.write((ply_header % dict(vert_num=len(verts))).encode('utf-8'))
        np.savetxt(f, verts, fmt='%f %f %f %d %d %d')

if __name__ == '__main__':
    print('loading images...')
    imgL = cv2.pyrDown(cv2.imread('images/1.png'))
    imgR = cv2.pyrDown(cv2.imread('images/2.png'))

    # disparity range
    window_size = 3
    min_disp = 2
    num_disp = 20 - min_disp
    stereo = cv2.StereoSGBM_create(minDisparity=min_disp,
                                    numDisparities=num_disp,
                                    blockSize=2,
                                    P1=8 * 3 * window_size ** 2,
                                    P2=32 * 3 * window_size ** 2,
                                    disp12MaxDiff=100,
                                    uniquenessRatio=0,
                                    speckleWindowSize=100,
                                    speckleRange=60
                                    )

    print('computing disparity...')
    disp = stereo.compute(imgL, imgR).astype(np.float32) / 16.0

    print('generating 3d point cloud...')
    h, w = imgL.shape[:2]
    f = 0.8 * w
    Q = np.float32([[1, 0, 0, -0.5 * w],
                   [0, -1, 0, 0.5 * h],
                   [0, 0, 0, -f],
                   [0, 0, 1, 0]])
    points = cv2.reprojectImageTo3D(disp, Q)
    colors = cv2.cvtColor(imgL, cv2.COLOR_BGR2RGB)
    mask = disp > disp.min()
    out_points = points[mask]
    out_colors = colors[mask]
    out_fn = 'out.ply'
    write_ply(out_fn, out_points, out_colors)
    print(f'{out_fn} saved')
    subprocess.run(["open", out_fn])

    cv2.imshow('left', imgL)
    cv2.imshow('disparity', (disp - min_disp) / num_disp)

```

```
cv2.waitKey()  
cv2.destroyAllWindows()
```

3.6. Аналіз результатів відлагодження та верифікації результатів роботи програми.

Результати відлагодження та тестування довели працездатність розробленого коду.

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

IV. Висновки.

У цій лабораторній роботі я розглянула процес створення тривимірної хмари точок з двох зображень, отриманих зі стереокамери. Було використано бібліотеки OpenCV та NumPy для обробки зображень та обчислення диспаритету між ними. Після обчислення диспаритету було відтворено тривимірну хмару точок. Одним із ключових етапів було обчислення диспаритету за допомогою методу StereoSGBM, який враховує особливості стереозору. Після цього ми використовували результати обчислення для перетворення зображень у тривимірну модель, яка може бути використана для різних аналітичних та візуалізаційних цілей. Також було здійснено зчитування кадрів з трьох джерел. Отримані зображення я використала в завданні, де необхідно було створити 3D об'єкт.

У результаті роботи було згенеровано тривимірну хмару точок та збережено її у форматі PLY, що дозволяє легко і зручно обробляти ці дані у різних програмах для подальшого аналізу чи візуалізації.

Ця лабораторна робота відображає важливий етап у процесі комп'ютерного зору та обробки зображень, і може бути використана як основа для подальших досліджень у галузі комп'ютерного зору, робототехніки та віртуальної реальності.

Виконала: студент Яковенко Д.О.