**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря Сікорського**

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи №8**

**з навчальної дисципліни «Computer Vision»**

**Тема:**

**ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ТРИВИМІРНОЇ РЕКОНСТРУКЦІЇ**

**ОБ’ЄКТІВ ЗА ЦИФРОВИМИ ЗОБРАЖЕННЯМИ**

**Виконав:**

Студент 3 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Головня О.Р.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

**Київ 2024**

**І. Мета роботи:**

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

**ІІ. Завдання**

Завдання І рівеня складності – максимально 8 балів.

Організувати та реалізувати роботу стереопари та отримати цифрове статичне

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

**ІІІ. Результати виконання лабораторної роботи.**

**3.1. Синтезована математична модель відповідно до індивідуального завдання.**

Цей скрипт реалізує реконструкцію 3D об'єкта за двома його 2D зображеннями. Відповідно до завдання, ось як відображено основні кроки математичної моделі реконструкції:

1. **Відкриття зображень:** Код відкриває два зображення **img1** і **img2** для подальшої обробки.
2. **Обчислення карт диспарності:** Використовується метод StereoSGBM для обчислення карти диспарності між двома зображеннями.
3. **Триангуляція точок:** Використовуючи карту диспарності та матрицю проекції Q, проводиться триангуляція точок для відновлення 3D координат об'єктів.
4. **Збереження результату:** Отримані точки та відповідні кольори зображень зберігаються у форматі PLY файлу.
5. **Відображення результату:** Після збереження файлу PLY, він зчитується за допомогою бібліотеки Open3D і відображається у вікні 3D візуалізації.

**3.2. Блок схема алгоритму та її опис.**

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



Рис. 1. Блок-схема логіки програми

**3.3. Опис структури проекту програми**

|  |
| --- |
|  |
| Рис.2. Структура проекту. |

photo\_1.jpg, photo\_2.jpg, photo\_3.jpg, photo\_4.jpg, photo\_5.jpg – зображення з різними ракурсами об’єкта

point\_cloud.ply – файл для опису геометрії(хмара точок)

Holovnia\_lr\_8.docx – звіт

Holovnia\_lab8.py - файл програми

**3.4. Результати роботи програми відповідно до завдання.**

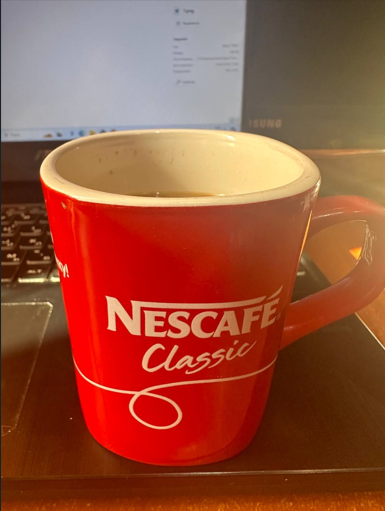


Рис. 3. Зображення ракурсів об’єкта



Рис. 4. Зображення мапи невідповідності

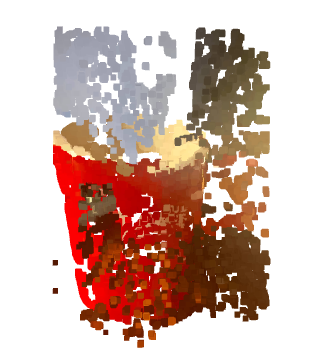
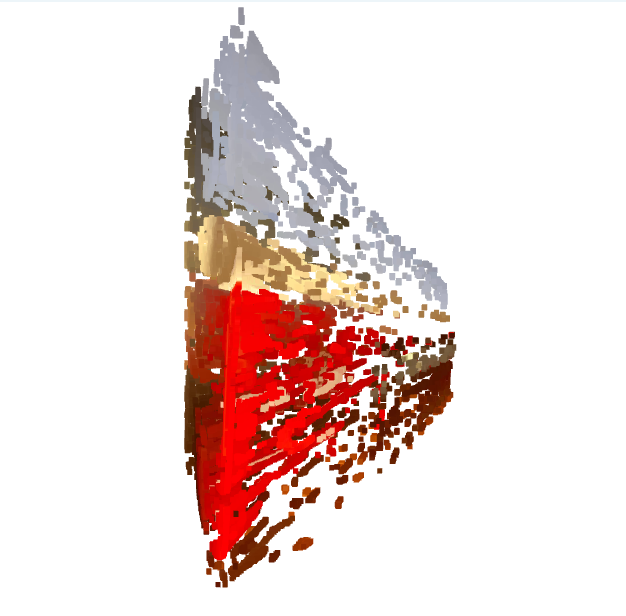
 



Рис. 5. Зображення хмари точок з різних ракурсів

**3.5. Програмний код.**

import numpy as np

import cv2

import open3d as o3d

def write\_point\_cloud(filename, vertices, colors):

vertices = vertices.reshape(-1, 3)

colors = colors.reshape(-1, 3)

vertices\_with\_colors = np.hstack([vertices, colors])

ply\_header = f"ply\nformat ascii 1.0\nelement vertex {len(vertices\_with\_colors)}\n" \

"property float x\nproperty float y\nproperty float z\n" \

"property uchar red\nproperty uchar green\nproperty uchar blue\nend\_header\n"

with open(filename, 'w') as file:

file.write(ply\_header)

np.savetxt(file, vertices\_with\_colors, fmt='%f %f %f %d %d %d')

def img\_reconstruction(image\_paths):

img1, img2 = [cv2.imread(path) for path in image\_paths[:2]]

if img1 is None or img2 is None:

raise ValueError("Images could not be loaded")

window\_size = 3

min\_disparity = 30

num\_disparities = 120 - min\_disparity

stereo = cv2.StereoSGBM\_create(

minDisparity=min\_disparity,

numDisparities=num\_disparities,

blockSize=16,

P1=8 \* 2 \* window\_size\*\*2,

P2=32 \* 2 \* window\_size\*\*2,

disp12MaxDiff=1,

uniquenessRatio=10,

speckleWindowSize=100,

speckleRange=32

)

disparity = stereo.compute(img1, img2).astype(np.float32) / 16.0

height, width = img1.shape[:2]

focal\_length = 0.9 \* width

Q = np.float32([[1, 0, 0, -0.5 \* width],

[0, -1, 0, 0.5 \* height],

[0, 0, 0, -focal\_length],

[0, 0, 1, 0]])

points = cv2.reprojectImageTo3D(disparity, Q.astype(np.float32))

colors = cv2.cvtColor(img1, cv2.COLOR\_BGR2RGB)

mask = disparity > disparity.min()

out\_points = points[mask]

out\_colors = colors[mask]

output\_file = 'point\_cloud.ply'

write\_point\_cloud(output\_file, out\_points, out\_colors)

cv2.namedWindow('map', cv2.WINDOW\_NORMAL)

cv2.imshow('map', (disparity - min\_disparity) / num\_disparities)

cv2.resizeWindow('map', 650, 650)

cv2.waitKey(0)

cv2.destroyAllWindows()

pcd = o3d.io.read\_point\_cloud(output\_file)

o3d.visualization.draw\_geometries([pcd], width=650, height=650, left=60, top=60)

if \_\_name\_\_ == '\_\_main\_\_':

image\_paths = ['photo\_3.jpg', 'photo\_2.jpg']

try:

img\_reconstruction(image\_paths)

except Exception as e:

print(f"An error occurred: {e}")

**3.6. Аналіз результатів відлагодження та верифікації результатів роботи програми.**

Результати відладки та тестування підтвердили працездатність розробленого коду. Порівняння отриманих результатів з технічними умовами завдання на лабораторну роботу та верифікація функціоналу програмного коду показали, що всі вимоги були виконані у повному обсязі.

**IV. Висновки.**

Під час виконання лабораторної роботи було розроблено програмний скрипт обробки зображень та візуалізація об’єктів у 3D. Досліджено методологію і технології реконструкції 3D просторових об’єктів за їх 2Dзображеннями методами багатовидової (стерео / сигнатурна) обробки.