Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Ковальова Філіпа Вячеславовича*
	(ПІБ)
академічної групи	*122М-22-1*
	(шифр)
спеціальності	*122 Комп'ютерні науки*
	(код і назва спеціальності)
освітньої програми	*«Комп'ютерні науки»*
	(назва освітньої програми)
на тему:	*Розробка та дослідження мобільного додатку для визначення карти відстаней до об'єктів зображення з використанням нейронних мереж*

Ф.В. Ковальов

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
розділів кваліфікаційної роботи				
спеціальний	*доц.Спірінцев В.В.*	75	добре	

Рецензент				

Нормоконтролер	*проф. Лактіонов І.С.*	75	добре	

Дніпро
2023

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » __ 20 23 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності _____ *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту *122м-22-1* *Ковальову Філіпу Вячеславовичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка та дослідження мобільного додатку для визначення карти відстаней до об'єктів зображення з використанням нейронних мереж*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – є процес обробки та зображення сцен, включаючи зображення отримані з камери мобільного пристрою.

Предмет досліджень – моделі та методи монокулярного аналізу глибини зображення, тобто аналізу відстані від єдиної камери до об'єктів сцени.

Мета НДР – підвищення ефективності визначення карти відстаней до об'єктів зображення з використанням технологій нейронних мереж для подальшої обробки вихідного зображення відповідно до отриманих даних за глибиною сцени, за рахунок розробки та дослідження мобільного додатку під ОС Android.

Вихідні дані для проведення роботи – теоретичні та експериментальні дослідження по визначенню карти глибини зображення довільної сцени.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень полягає у запропонованому підході, щодо реалізації комплексного рішення по визначенню карти відстаней до об'єктів зображення з використанням технологій нейронних мереж, що надає можливість виконувати подальшу обробку вихідного зображення відповідно до отриманих даних про глибину сцени на персональному девайсі.

Практична цінність результатів. Розроблене програмне забезпечення надає користувачу, на основі персональних даних, швидко здійснювати складні обчислення,

а також виконувати подальшу обробку вихідного зображення (а саме: проекціювати довільне зображення на вихідне, відповідно до отриманих даних про глибину сцени) отримувати та зберігати результати обробки, одразу на персональному девайсі - мобільному телефоні. Сфери застосування розробленого ПЗ: рекламні фірми (для швидкого і наочного представлення зовнішнього вигляду логотипу чи назви компанії на тому чи іншому будинку), дизайнери (для економії часу на створення рекламних макетів або портфоліо), розважальна сфера.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити безпосереднє використання запропонованого рішення щодо визначення карти відстаней до об'єктів зображення з використанням технологій нейронних мереж та подальшої обробки вихідного зображення відповідно до отриманих даних за глибиною сцени.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок –кінець)**
Аналіз та систематизація літератури з обраної тематики.	11.09.2023-25.09.2023
Обґрунтування мети створення системи та складу завдань. Постановка завдання.	26.09.2023-29.09.2023
Розробка алгоритму та структури додатку.	02.10.2023-09.10.2023
Створення графічного інтерфейсу додатку.	10.10.2023-16.10.2023
Розробка програмного модуля, що виконує прийом і обробку даних користувача.	17.10.2023-07.11.2023
Реалізація можливості підключення обробного модуля і мобільного додатку.	08.11.2023-19.11.2023
Тестування програми на конкретних прикладах.	20.11.2023-01.12.2023

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним завдяки скороченню загального часу створення проектів на замовлення бізнесу, що дозволить збільшити обсяги реалізації готових проектів.

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки удосконаленню підходу до визначення карти відстаней до об'єктів зображення з використанням технологій нейронних мереж та подальшої обробки вихідного зображення відповідно до отриманих даних за глибиною сцени.

7 ДОДАТКОВІ ВИМОГИ

Завдання видав _____ *Спірінцев В.В.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Ковальов Ф.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *09.10.2023 р.*

Термін подання кваліфікаційної роботи до ЕК *04.12.2023*

РЕФЕРАТ

Пояснювальна записка: 95 стор., 19 рис., 4 додатка, 35 джерел.

Об’єкт досліджень: процес обробки та зображення сцен, включаючи зображення отримані з камери мобільного пристрою.

Предмет досліджень: моделі та методи монокулярного аналізу глибини зображення, тобто аналізу відстані від єдиної камери до об'єктів сцени.

Мета роботи: підвищення ефективності визначення карти відстаней до об'єктів зображення з використанням технологій нейронних мереж для подальшої обробки вихідного зображення відповідно до отриманих даних за глибиною сцени за рахунок розробки та дослідження мобільного додатку під ОС Android.

Методи дослідження. Для вирішення поставлених задач використані методи аналізу даних: методи аналітичної та диференціальної геометрії, обчислювальні методи.

Новизна отриманих результатів полягає у запропонованому підході, щодо реалізації комплексного рішення по визначенню карти відстаней до об'єктів зображення з використанням технологій нейронних мереж, що надає можливість виконувати подальшу обробку вихідного зображення відповідно до отриманих даних про глибину сцени на персональному девайсі.

Практичне значення отриманих результатів. Розроблене програмне забезпечення надає користувачу, на основі персональних даних, швидко здійснювати складні обчислення, а також виконувати подальшу обробку вихідного зображення (а саме: проекціювати довільне зображення на вихідне, відповідно до отриманих даних про глибину сцени) отримувати та зберігати результати обробки, одразу на персональному девайсі - мобільному телефоні.

Область застосування. Розроблений мобільний додаток може бути використаний: рекламними фірмами (для швидкого і наочного представлення зовнішнього вигляду логотипу чи назви компанії на тому чи іншому об’єкті), дизайнерами (для економії часу на створення рекламних макетів або портфоліо), у розважальній сфері (для «примірки» різних елементів на певних об’єктах), у навчальному процесі (в рамках розділів з 3D-моделювання).

Значення роботи та висновки. Запропонований підхід щодо комплексного рішення по визначенню карти відстаней до об'єктів зображення з використанням технологій нейронних мереж з швидким та наочним представленням результатів обробки зображень на персональному девайсі, що дозволяє здійснювати реалізацію проектів зі значним скороченням матеріальних та часових ресурсів, що підтверджується розробленим програмним продуктом.

Прогнози щодо розвитку досліджень. Покращити інформаційну систему, додавши нової функціональності за рахунок впровадження нових підходів та методів по визначенню карти відстаней до об'єктів зображення.

Список ключових слів: карта відстаней, карта глибини, зображення, нейронна мережа, комп’ютерний зір, Python, QT, QML, сервер, клієнт.

ABSTRACT

Explanatory note: 95 pages, 19 figures, 4 appendices, 35 sources.

Object of research: the process of processing and imaging scenes, including images obtained from the camera of a mobile device.

Subject of research: models and methods of monocular analysis of image depth, that is, analysis of the distance from a single camera to objects in the scene.

Purpose of Master's thesis: increasing the efficiency of determining the map of distances to image objects using neural network technologies for further processing of the original image according to the received data on the depth of the scene due to the development and research of a mobile application for the Android OS.

Research methods. To solve the problems, data analysis methods were used: methods of analytical and differential geometry, computational methods.

Originality of research consists in the proposed approach regarding the implementation of a comprehensive solution for determining the map of distances to image objects using neural network technologies, which allows further processing of the original image in accordance with the obtained data on the depth of the scene on a personal device.

Practical value of the results. The developed software allows the user, based on personal data, to quickly perform complex calculations, as well as further process the original image (namely: project an arbitrary image onto the original one, according to the obtained data on the depth of the scene), receive and save the processing results immediately on a personal device – mobile phone.

Scope of application. The developed mobile application can be used by: advertising firms (for a quick and visual representation of the appearance of a logo or company name on a particular object), designers (to save time on creating advertising layouts or portfolios), in the entertainment sector (for “trying on” different elements on certain objects), in the educational process (within the sections on 3D modeling).

The value of the work and conclusions. An approach is proposed to a comprehensive solution for determining a map of distances to image objects using neural network technologies with a quick and visual presentation of the results of image processing on a personal device, which allows the implementation of projects with a significant reduction in material and time resources, which is confirmed by the developed software product.

Research forecast and development. Improve the information system by adding new functionality through the introduction of new approaches and methods for determining a map of distances to image objects.

List of keywords: distance map, depth map, images, neural network, computer vision, Python, QT, QML, server, client.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AD - Automatic Differentiation

CCD - Charged Coupled Device

CNN - Convolutional Neural Network

MVS - Multi-View Stereo

OpenCV - Open Computer Vision Library

QML - Qt Meta Language / Qt Modeling Language

SfM - Structure-from-Motion

VTK - The Visualization Toolkit

ЗМІСТ

ВСТУП

З розвитком інформаційних технологій, все більше даних отриманих з навколишнього середовища перетворюються і зберігаються в електронному вигляді. Це підвищує надійність зберігання інформації, швидкість доступу до неї, а також надає нові можливості її аналізу і обробки.

Актуальним завданням, на сьогоднішній день, є розробка зорових систем з відповідним ПЗ, що представляють інформацію про сцену в тривимірному вигляді. Дане завдання вирішується за допомогою 3D-моделювання - процесу створення математичного подання тривимірного об'єкту. Реконструкція тривимірних моделей застосовується для створення систем віртуальної реальності, удосконалення медичного обладнання, організації охоронної діяльності, створення топографічних систем, забезпечення роботи робототехнічних комплексів, а також в інших областях науки і техніки.

Для створення тривимірних моделей об'єктів існує безліч підходів в комп'ютерному зорі. При цьому, незалежно від підходу, зоровій системі заздалегідь невідомі тип, кількість і розмір об'єктів сцени. Тобто машинна система, подібно зоровій системі людини, повинна надавати інформацію про спостережуваний простір незалежно від його змісту.

Важливим етапом в побудові тривимірної моделі є отримання карти глибини. У тривимірній комп'ютерної графіці карта глибини - це канал зображення або зображення, яке містить інформацію, що стосується відстані поверхонь об'єктів сцени від точки огляду. Кожен піксель містить інформацію про віддаленість точки поверхні сцени від камери. Тобто оцінка глибини спрямована на отримання уявлення про просторову структуру видимої сцени.

Для отримання інформації про глибину сцени, можна використовувати спеціальні пристрої - сенсори глибини, наприклад безконтактний ігровий контролер для Xbox360 Microsoft Kinect або камери Intel RealSense. Принцип роботи полягає в подачі інфрачервоного випромінювання, яке відбиваючись від

об'єктів фіксується приймачами і перетворюється на значення відстані. Очевидно, що такі установки є витратними і вимагають складного спеціалізованого ПЗ.

Інший підхід - використання методів стереозору, де значення кожного пікселя обчислюється з пари зображень (стереопари), знятих двома камерами, розташованими на деякій відстані.

На даному етапі, ми спостерігаємо бурхливий розвиток технологій мобільних пристроїв. Поряд з поширеністю і доступністю сучасних смартфонів, варто зазначити можливість виконання на них досить складних операцій. Тому є актуальним створення мобільних додатків для вирішення найрізноманітніших завдань користувача, пов'язаних з ресурсовитратною обробкою даних.

Варто відзначити, що зручно було б проводити 3D-моделювання не на дорогому і потужному обладнанні, а на доступному кожному пристрої, наприклад - смартфоні. Очевидно, що установка спеціальних сенсорів і додаткових камер підвищує вартість гаджету. Тому, одним з ефективних шляхів вирішення даної проблеми є розробка саме програмного забезпечення для обчислення карти глибини. Таке ПЗ при отриманні на вхід одного двовимірного зображення, наприклад фото з камери або з галереї телефону, має виконувати обчислення карти відстаней від точки огляду до об'єктів сцени. Для вирішення цього завдання можна скористатися технологіями нейронних мереж. Результатом навчання мережі на великому наборі даних - є модель, на вхід якої можна подати одне зображення і результатом стане його очікувана карта глибини. Тобто немає необхідності у використанні додаткових апаратних засобів.

Актуальність теми. Карти глибини використовуються в комп'ютерному зорі для моделювання, реконструкції (процес отримання форми і вигляду реальних об'єктів) тривимірних фігур, при обробці тривимірних зображень за допомогою інструментів двовимірних зображень, для моделювання освітлення в тривимірній комп'ютерній графіці, а також для створення автостереограм (вид стереограми, яка дає стереоскопічний, об'ємний ефект без зовнішніх

сепаруючих застосувань - затворні окуляри, окуляри з поляризаційними фільтрами) і ілюзій тривимірного перегляду з допомогу стереоскопії. Обчислення відстаней від точок сцени до об'єктива камери є поширеним підходом при моделюванні напівпрозорих середовищ в сцені (туман, дим, вода). Сьогодні, при стрімкому розвитку соціальних мереж та, відповідно, популярності фотоіндустрії, актуально використання карт відстаней для імітації малої глибини різкості - коли деякі частини сцени виявляються не в фокусі, наприклад в портретній зйомці або при розмитті в різному ступені обраних користувачем областей зображення.

Карти глибин можуть застосовуватися в абсолютно різних сферах життя. Наприклад, актуальною є розробка систем візуальної навігації призначених для використання в транспортних засобах. Або розробка технологій для сліпих (тифлотехнологій) - людина може отримувати звукові сигнали від спеціалізованого пристрою і орієнтуватися за ними [3].

Крім того, знаючи, що карту глибини можна використовувати для реконструкції тривимірних сцен, можливе створення алгоритмів, що реалізують проекціювання зображень на об'єкти цієї сцени. А виходячи з того, що сьогодні наше життя неможливо уявити без засобів масової інформації, дане програмне забезпечення є актуальним для рекламодавців, рекламних компаній і дизайнерів. Використовуючи мобільний додаток, що реалізує такі алгоритми, фірми-рекламодавці зможуть швидко і наочно представляти вигляд логотипу чи назви фірми на тому чи іншому будинку, а дизайнери зможуть витрачати значно менше часу на створення рекламних макетів або портфоліо. До того ж, розроблений мобільний додаток є актуальним в розважальній сфері, наприклад, для простих користувачів, які могли б «приміряти» різні картини або постери на стіни своєї квартири. Або досить актуальною програма може бути для студентів, що вивчають 3D-моделювання, її можна використовувати як факультативний матеріал для наочного розглядання теми стереозору та підвищення зацікавленості у даній області. Отже, актуальність теми роботи пов'язана з великим поширенням і значними перспективами досліджуваного

об'єкта. Перевагою запропонованої роботи є швидке та наочне представлення результатів обробки на персональному девайсі.

Мета і задачі дослідження. Метою роботи є підвищення ефективності визначення карти відстаней до об'єктів зображення з використанням технологій нейронних мереж для подальшої обробки вихідного зображення відповідно до отриманих даних за глибиною сцени, за рахунок розробки та дослідження мобільного додатку під ОС Android.

Для досягнення поставленої мети в роботі необхідно вирішити наступні задачі:

- здійснити аналіз і систематизацію літератури за обраною тематикою;
- розробити алгоритм та структуру додатку;
- створити графічний інтерфейс додатку;
- розробка програмний модуль, що виконує прийом і обробку даних користувача;
- здійснити реалізацію підключення обробного модуля з мобільним додатком;
- здійснити тестування програми на конкретних прикладах.

Об’єкт досліджень: процес обробки та зображення сцен, включаючи зображення отримані з камери мобільного пристрою.

Предмет досліджень: моделі та методи монокулярного аналізу глибини зображення, тобто аналізу відстані від єдиної камери до об'єктів сцени.

Методи дослідження. Для вирішення поставлених задач використані методи аналізу даних: методи аналітичної та диференціальної геометрії, обчислювальні методи.

Новизна отриманих результатів полягає у запропонованому підході, щодо реалізації комплексного рішення по визначенню карти відстаней до об'єктів зображення з використанням технологій нейронних мереж, що надає можливість виконувати подальшу обробку вихідного зображення відповідно до отриманих даних про глибину сцени на персональному девайсі.

Практичне значення отриманих результатів. Розроблене програмне забезпечення надає користувачу, на основі персональних даних, швидко здійснювати складні обчислення, а також виконувати подальшу обробку вихідного зображення (а саме: проекціювати довільне зображення на вихідне, відповідно до отриманих даних про глибину сцени) отримувати та зберігати результати обробки, одразу на персональному девайсі - мобільному телефоні.

Особистий внесок автора. Запропонований підхід щодо комплексного рішення по визначенню карти відстаней до об'єктів зображення з використанням технологій нейронних мереж з швидким та наочним представленням результатів обробки зображень на персональному девайсі, що дозволяє здійснювати реалізацію проектів зі значним скороченням матеріальних та часових ресурсів, що підтверджується розробленим програмним продуктом.

Структура та обсяг кваліфікаційної роботи. Відповідно до мети, задач і предмета дослідження, кваліфікаційна робота складається з реферату, вступу, трьох основних розділів і висновків, списку використаних джерел та 4 додатків. Загальний обсяг роботи містить 95 сторінок друкованого тексту, із них основна частина - 24 сторінки з 19 рис., спеціальна – 39 сторінок, списку використаних джерел з 35 найменувань на 3 сторінках, 4 додатках на 10 сторінках.

РОЗДІЛ 1
АНАЛІЗ ТА ДОСЛІДЖЕННЯ ІСНУЮЧИХ ТЕХНОЛОГІЙ ВИЗНАЧЕННЯ КАРТИ ВІДСТАНЕЙ ДО ОБ'ЄКТІВ ЗОБРАЖЕННЯ

1.1. Основні принципи формування тривимірної моделі

На сьогоднішній день, реконструкція 3D-моделей застосовується для створення систем віртуальної реальності, удосконалення медичного обладнання, організації охоронної діяльності, а також в інших галузях науки і техніки.

Для створення тривимірних моделей об'єктів існує безліч підходів. Найчастіше використовують 3D-сканери, проте вони являють собою досить дороге і складне устаткування. Крім того, в одиницю часу може оброблятися лише мала ділянка поверхні, отже повне сканування вимагає великих витрат часу. Такі сканери варто використовувати, якщо необхідно змоделювати об'єкт повністю - з усіх боків. Наприклад, відсканувавши людину, можна використовувати отриману модель у відеогрі. Для деяких завдань, досить побудувати тільки видиму частину об'єкта. Наприклад, при розробці дизайну інтер'єру було б зручно «приміряти» на стіни будинку різні шпалери і картини або вибрати місце для рекламного банера на вулицях міста.

1.2. Карта глибини

Важливим етапом в побудові тривимірної моделі є отримання карти глибини. У тривимірної комп'ютерній графіці карта глибини - це канал зображення або зображення, яке містить інформацію, що стосується відстані поверхонь об'єктів сцени від точки огляду (рис. 1.1). Кожен піксель містить інформацію про віддаленість точки поверхні сцени від камери. Одноканальна 8-бітна карта глибини може представляти діапазон з 256 різних відстаней.

Рис.1.1. Зображення сцен та відповідні їм чорно-білі карти глибини

Карти глибини мають широкий спектр напрямків використання, зокрема:

- моделювання рівномірно щільних напівпрозорих середовищ в сцені (туман, дим, вода);
- імітація малої глибини різкості - коли деякі частини сцени виявляються не в фокусі. Наприклад, створення ефекту розмиття фону у зображенні при портретній зйомці;
- розмиття деяких обраних областей зображення в різному ступені;
- Shadow mapping - частина процесу, використовуваного для створення тіней, що відкидаються об’єктами при певному освітленні в тривимірній комп'ютерній графіці. В цьому випадку карти глибини розраховуються виходячи з розташування джерел світла, а не глядача;
- створення автостереограм (вид стереограми, яка дає стереоскопічний, об'ємний ефект без зовнішніх сепаруючих пристроїв - затворні окуляри, окуляри з поляризаційними фільтрами) і ілюзій тривимірного перегляду за

допомогою стереоскопії;

- підповерхневе розсіювання, як частина процесу додавання реалізму, шляхом моделювання напівпрозорих властивостей напівпрозорих матеріалів, таких як шкіра людини;
- у комп'ютерному зорі для моделювання, реконструкції (процес отримання форми і вигляду реальних об'єктів) тривимірних фігур;
- у машинному і комп'ютерному зорі - обробка тривимірних зображень за допомогою інструментів двовимірних зображень;
- розробка систем візуальної навігації, призначених для застосування у транспортних засобах, зокрема, в автомобілях. У такій системі на екран водія виводиться зображення дороги і об'єктів, що знаходяться на ній. Для кожного об'єкта вказується відстань до нього і його лінійні розміри.
- тифлотехнології (технології для сліпих). Наприклад, може бути створено мобільний пристрій, який кріпиться до одягу або на голову людини. Під час руху сліпа людина отримує звукові сигнали від пристрою і орієнтується за ними, визначаючи, чи далеко від нього знаходиться об'єкт або перешкода [3].

Обмеження:

- одноканальні карти глибини записують першу видиму поверхню (найближчу до камери), і тому не можуть відображати інформацію про поверхні, видимі або заломлені через прозорі об'єкти або відображені в дзеркалах. Це може знизити точність при моделюванні;
- одноканальні карти глибини не можуть передавати кілька відстаней в межах одного пікселя. Наприклад, з моделями волосся, хутра або трави [1].

1.3. Сенсори глибини

Для отримання інформації про глибину сцени, можна використовувати спеціальні пристрої - сенсори глибини, наприклад безконтактний ігровий контролер для Xbox360 Microsoft Kinect або камери Intel RealSense (рис. 1.2). Принцип роботи полягає в подачі інфрачервоного

випромінювання, яке відбиваючись від об'єктів фіксується приймачами і перетворюється на значення відстані [2].

Рис.1.2. Зображення сенсора глибини

Деякі датчики глибини мають камеру RGB (червоний, зелений, синій), а деякі - ні. Для визначення глибини завжди повинні бути присутніми два інших елемента: інфрачервоний (інфрачервоний) проектор та інфрачервона камера. ІК-проектор проекцією рисунок ІК-світла, який падає на об'єкти навколо нього, як безліч точок. Ми не бачимо точок, тому що світло проекціюється у інфрачервоному діапазоні кольорів (рис. 1.3). Але ІК-камера може бачити точки. ІК-камера, по суті, така ж, як звичайна RGB-камера, за винятком того, що зображення, які вона захоплює, знаходяться в інфрачервоному діапазоні.

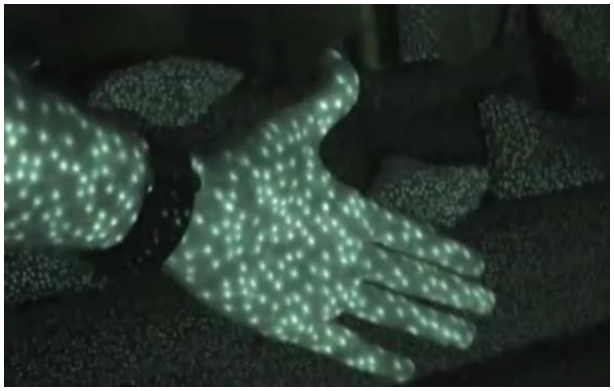

Рис.1.3. Візерунок точок, спроекційованих сенсором глибини у інфрачервоному діапазоні

Камера відправляє відеосигнал з цим спотвореним точковим растром в процесор датчика глибини, а він обчислює глибину, що залежить від зсуву точок (рис. 1.4). На близьких об'єктах візерунок розряджений, на далеких - щільний [2].

Рис.1.4. Карта відстаней побудована сенсором глибини

Очевидно, що такі установки є досить витратними і потребують спеціалізованого програмного забезпечення.

1.4. Комп'ютерний зір

Комп'ютерний зір (computer vision) - це наукова дисципліна, що вивчає теорію і базові алгоритми аналізу зображень і сцен. Основні методи комп’ютерного зору включають наступні: методи визначення афінної та проективної структур за рухом об’єкта, методи супроводження рухомого об’єкта, методи пошуку в цифрових бібліотеках, методи побудови 3D-моделей за послідовністю зображень, а також методи візуального аналізу та оцінювання кількості й параметрів об’єктів сцени [5]. Останнє коло задач найбільш часто розв’язується за допомогою методів комп’ютерного стереозору [3].

Інший підхід для побудови карти глибини - використання методів комп'ютерного зору, а саме методів створення тривимірних моделей, що

використовують послідовність зображень. В цьому підході, значення кожного пікселя карти глибини обчислюється з пари зображень, знятих двома камерами, розташованими на деякій відстані. Відповідно зображення вийдуть знятими з різних ракурсів. Використовуючи ці зображення, можливо обчислити відстань від об’єктиву камери до обраного об'єкта, яка розраховується через зсув цього об'єкта на другому зображенні відносно першого. Для даного методу можна використати стереоскопічну систему, що складається, наприклад, з двох веб-камер (оптичні вісі яких - паралельні) і комп'ютера (рис. 1.5).

Рис.1.5. Стереоскопічна система

Цей метод зазвичай складається з декількох етапів:

1. Обрання певного об'єкту на одному зображенні.
2. Знаходження цього об'єкту на іншому зображенні за допомогою техніки розпізнавання образів.
3. Обчислення різниці в пікселях у розташуванні об'єктів на зображеннях.
4. Знаходження відстані від камери до об'єкту, за допомогою геометричних розрахунків, користуючись обчисленою різницею в пікселях та значенням відстані між камерами.

Однак, система з камер повинна відповідати певним вимогам:

- знаходження камер в одній горизонтальній площині;
- паралельність оптичних вісей камер;
- одночасна зйомка з обох камер.

В стереоскопічних системах зазвичай розглядається дві площини: площина об’єкта, де знаходиться об’єкт, та площина, де формується зображення об’єкта і виконується його реєстрація на фотоплівку чи ПЗС-матрицю [3, 11, 13, 14].

1.5. Математичний апарат, що використовується в стереозорі

1.5.1. Проективна геометрія й однорідні координати

В геометрії стереозору значну роль відіграє проективна геометрія. До проективної геометрії є кілька підходів: геометричний (ввести поняття геометричних об'єктів, аксіом і з цього виводити всі властивості проективного простору), аналітичний (розглядати все в координатах, як в аналітичному підході до Евклідовій геометрії), алгебраїчний.

Евклідова геометрія (або елементарна геометрія) - геометрична теорія, заснована на системі аксіом, вперше викладеної в «Началах» Евкліда (III століття до н. е.). В «Началах» Евкліда була дана наступна система аксіом:

- від будь-якої точки до будь-якої точки можна провести пряму;
- обмежену пряму можна безперервно продовжувати по прямій;
- з будь-якого центру всяким радіусом може бути описане коло;
- всі прямі кути рівні між собою;
- якщо пряма, яка перетинає дві прямі, утворює внутрішні односторонні кути, менші двох прямих кутів, то, продовжені необмежено, ці дві прямі зустрінуться з того боку, де кути менше двох прямих кутів [15].

У 1899 році Гільберт запропонував першу досить сувору аксіоматику геометрії Евкліда. Евклідовий простір - в первісному значенні, простір, властивості якого описуються аксіомами геометрії Евкліда. Тут передбачається, що простір має розмірність рівну 3. В більш загальному сенсі, може позначати один з подібних і тісно пов'язаних об'єктів: кінцевовимірний дійсний векторний простір R^n з введеним на ньому позитивно визначеним скалярним

добутком. n-вимірний Евклідовий простір зазвичай позначається E^n. Евклідовий простір розмірності 2 (E^2) - Евклідова площина [16].

Розглянемо аналітичний підхід.

Точки проективної площини. Розглянемо двовимірний проективний простір (який ще називається проективною площиною). У той час як на звичайній Евклідовій площині точки описуються парою координат $(x,y)^T$, на проективній площині точки описуються трикомпонентним вектором $(x, y, w)^T$. При цьому для будь-якого ненульового числа a, вектори $(x, y, w)^T$ і $(ax, ay, aw)^T$ відповідають одній тій самій точці. А нульовий вектор $(0,0,0)^T$ не відповідає ніякій точці і викидається з розгляду. Такий опис точок площини називається однорідними координатами (homogeneous coordinates).

Точкам проективної площини можна зіставити точки звичайної Евклідової площини. Координатному вектору $(x, y, w)^T$ при $w \neq 0$ можна зіставити точку Евклідової площини з координатами $(x/w, y/w)^T$. Якщо ж $w = 0$, тобто координатний вектор має вигляд $(x, y, 0^T)$, то будемо говорити, що ця точка в нескінченності. Таким чином, проективну площину можна розглядати як Евклідову площину, доповнену точками з нескінченності.

Перейти від однорідних координат $(x, y, w)^T$ до звичайних Евклідових можна шляхом ділення координатного вектора на останню компоненту і подальшого її відкидання $(x, y, w)^T \rightarrow (x/w, y/w)^T$. А від Евклідових координат $(x, y)^T$ перейти до однорідних можна за рахунок доповнення координатного вектора одиницею: $(x, y)^T \rightarrow (x, y, 1)^T$.

Тривимірний проективний простір. За аналогією з проективною площиною, точки тривимірного проективного простору визначаються чотирьохкомпонентним вектором однорідних координат $(x, y, z, w)^T$. Для будь-якого ненульового числа a, координатні вектори $(x, y, z, w)^T$ і $(ax, ay, az, aw)^T$ відповідають одній і тій самій точці.

Як у випадку проективної площини, між точками тривимірного Евклідового простору і тривимірного проективного простору можна встановити відповідність. Вектору однорідних координат $(x, y, z, w)^T$ при $w \neq 0$ відповідає

точка Евклідового простору з координатами $(x/w, y/w, z/w)^T$. А про точку з вектором однорідних координат виду $(x, y, z, 0)^T$ кажуть, що вона лежить в нескінченності.

Прямі на проективній площині. Будь-яка пряма на проективній площині описується, подібно точці, трикомпонентним вектором $I = (a, b, c)^T$. Вектор, що описує пряму, визначений з точністю до ненульового множника. При цьому рівняння прямої буде мати вигляд: $l^T x = 0$.

У разі, коли $a^2 + b^2 \neq 0$ ми маємо аналог звичайної прямої $ax + by + c = 0$. А вектор (0,0, w) відповідає прямій, що лежить в нескінченності.

Проективне перетворення (Homography, projective transformation). З геометричної точки зору, проективне перетворення - це зворотне перетворення проективної площини (або простору), яке переводить прямі в прямі. У координатах, проективне перетворення виражається у вигляді невиродженої квадратної матриці H, при цьому координатний вектор x переходить в координатний вектор x 'за такою формулою: x' = H x.

1.5.2. Модель проективної камери

Сучасні CCD-камери (Charged Coupled Device, камера із надчутливою матрицею, перетворює заряди пікселів в аналоговий сигнал) добре описуються за допомогою наступної моделі, званої проективною камерою (projective camera, pinhole camera). Проективна камера визначається центром камери, головною віссю - променем, що починається в центрі камери і спрямованим туди, куди камера дивиться, площиною зображення - площиною на яку виконується проектування точок, і системою координат на цій площині. У такій моделі, довільна точка тривимірного простору X проектується на площину зображення в точку x, що лежить на відрізку CX, який з'єднує центр камери C з вихідною точкою X (рис. 1.6). Cp – головна вісь камери.

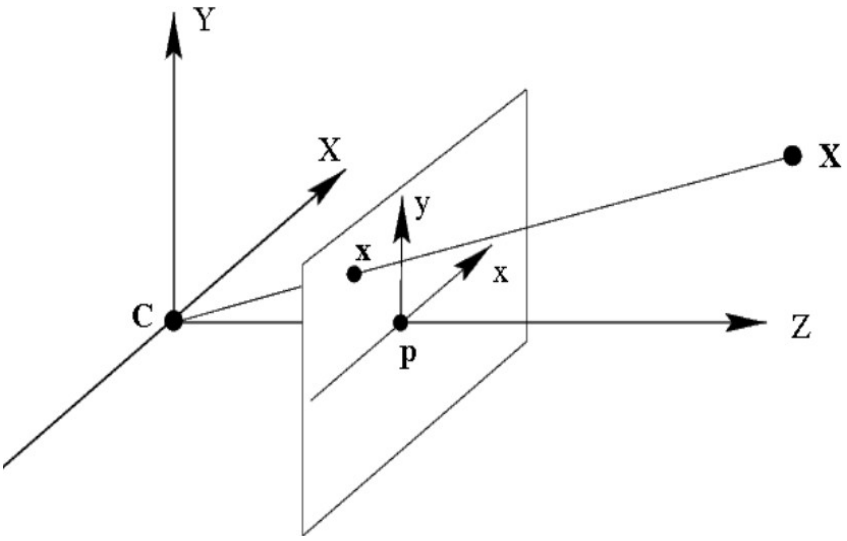

Рис.1.6. Модель проективної камери

Формула проектування має простий математичний запис в однорідних координатах:

$$x = P\,X\,, \tag{1.1}$$

де

X - однорідні координати точки простору;

x - однорідні координати точки площини;

P - матриця камери розміру 3 × 4.

Матриця P виражається наступним чином:

$$P = KR[I|-c] = K[R|t]\,, \tag{1.2}$$

де

K - верхня трикутна матриця внутрішніх параметрів камери розміру 3×3 (вираз 1.3);

R - ортогональна матриця розміру 3×3, яка визначає поворот камери щодо глобальної системи координат;

I - одинична матриця розміру 3×3, вектор c - координати центру камери;

t - дорівнює Rc.

Варто зазначити, що матриця камери визначена з точністю до постійного ненульового множника, який не змінить результатів

проектування точок за виразом (1.1). Однак цей постійний множник зазвичай вибирається так, щоб матриця камери мала наведений вище вид.

У самому простому випадку, коли центр камери лежить на початку координат, головна вісь камери співнаправлена з віссю Cz, осі координат на площині камери мають однаковий масштаб (що еквівалентно квадратним пікселям), а центр зображення має нульові координати, матриця камери дорівнюватиме P = K[I|0], де

$$K = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{1.3}$$

У реальних CCD камер пікселі зазвичай незначно відрізняються від квадратних, а центр зображення має ненульові координати. В такому випадку матриця внутрішніх параметрів набуде вигляду:

$$K = \begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{1.4}$$

Коефіцієнти *f, α_x, α_y* - називаються фокусною відстанню камери (відповідно до загальних і вздовж осей x і y).

Крім цього, в силу неідеальності оптики, на зображеннях, отриманих з камер, присутні спотворення-дисторсії (distortion). Дані спотворення мають нелінійний математичний запис:

$$\begin{aligned} x'' &= x'\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2 p_1 x' y' + p_2\left(r^2 + 2x'^2\right) \\ y'' &= y'\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_1\left(r^2 + 2y'^2\right) + 2 p_2 x' y', \end{aligned} \tag{1.5}$$

де

k1, k2, p1, p2, k3 - коефіцієнти дисторсії, що є параметрами оптичної системи;

$r^2 = x'^2 + y'^2$; (x', y') - координати проекції точки відносно центру зображення при квадратних пікселях і відсутності спотворень;

(x″, y″) - спотворені координати точки відносно центру зображення при квадратних пікселях.

Дисторсії не залежать від відстані до об'єкта, а залежать тільки від координат точок, в які проектуються пікселі об'єкта. Відповідно для компенсації дисторсій зазвичай виконується перетворення вихідного зображення отриманого з камери. Це перетворення буде одним і тим самим для всіх зображень, отриманих з камери, за умови сталості фокусної відстані (математично - однієї і тієї ж матриці внутрішніх параметрів).

У ситуації, коли відомі внутрішні параметри камери і коефіцієнти дисторсії, кажуть, що камера відкалібрована.

1.5.3. Пара камер

Визначити тривимірні координати спостережуваних точок можна за наявності як мінімум двох камер.

Матриці пари камер, калібрування.

Нехай є дві камери, задані своїми матрицями P і P' в деякій системі координат. У такому випадку говорять, що є пара відкаліброваних камер. Якщо центри камер не збігаються, то цю пару камер можна використовувати для визначення тривимірних координат спостережуваних точок.

Найчастіше, система координат вибирається так, що матриці камер мають вигляд P=K[I|0], P'=K'[R'|t']. Це завжди можна зробити, якщо вибрати початок координат, що збігається з центром першої камери, і направити вісь Z уздовж її оптичної осі.

Калібрування камер зазвичай виконується, за рахунок багаторазової зйомки деякого калібрувального шаблону, на зображенні можна легко виділити ключові точки, для яких відомі їх відносні положення в просторі. Далі складаються і вирішуються (наближено) системи рівнянь, що зв'язують координати проекцій, матриці камер і положення точок шаблону в просторі.

Існують загальнодоступні реалізації алгоритмів калібрування, наприклад, Matlab Calibration toolbox. Так само бібліотека OpenCV включає в себе алгоритми калібрування камер і пошуку каліброваного шаблону на

зображенні.

Епіполярная геометрія. Геометричні властивості, що зв'язують положення проекцій точки тривимірного простору на зображеннях з обох камер.

Нехай є дві камери (рис. 1.7). С - центр першої камери, С '- центр другої камери. Точка простору Х проекціюється в х на площину зображення лівої камери і в х' на площину зображення правої камери. Прообразом точки х на зображенні лівої камери є промінь хХ. Цей промінь проекціюється на площину другої камери в пряму l', звану епіполярною лінією. Образ точки Х на площині зображення другої камери обов'язково лежить на епіполярній лінії l'.

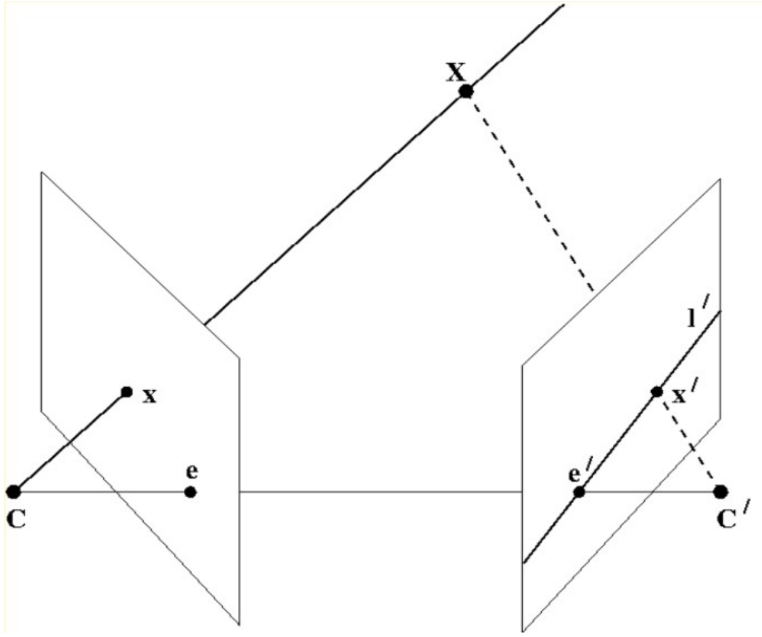

Рис.1.7. Епіполярна геометрія

Таким чином, кожній точці х на зображенні лівої камери відповідає епіполярна лінія l' на зображенні правої камери. При цьому пара для х на зображенні правої камери може лежати тільки на відповідній епіполярній лінії. Аналогічно, кожній точці х' на правому зображенні відповідає епіполярна лінія l на лівому.

Епіполярну геометрію використовують для пошуку стереопар, і для

перевірки того, що пара точок може бути стереопарою (тобто проекцією деякої точки простору). Епіполярна геометрія має дуже простий запис в координатах. Нехай є пара відкаліброваних камер, і нехай x - однорідні координати точки на зображенні однієї камери, а x' - на зображенні другої. Існує така матриця F розміру 3×3, що пара точок x, x' є стереопарою тоді і тільки тоді, коли:

$$x'^T F x = 0, \tag{1.6}$$

Матриця F називається фундаментальною матрицею (fundamental matrix). Її ранг дорівнює 2, вона визначена з точністю до ненульового множника і залежить тільки від матриць вихідних камер P і P'.

У разі, коли матриці камер мають вигляд P = K [I|0], P' = K' [R|t] фундаментальна матриця може бути обчислена за формулою:

$$F = K^{-1^T} R K^T \left[K R^T t \right]_x, \tag{1.7}$$

де для вектора e позначення $[e]_X$ обчислюється як:

$$\left[e\right]_x = \begin{pmatrix} 0 & -e_z & e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{pmatrix}, \tag{1.8}$$

За допомогою фундаментальної матриці обчислюються рівняння епіполярних ліній. Для точки x, вектор, що задає епіполярну лінію, буде мати вигляд l'= F x, а рівняння самої епіполярної лінії: $l'^T x' = 0$. Аналогічно для точки x', вектор, що задає епіполярну лінію, буде мати вигляд $l = F^T x'$.

Крім фундаментальної матриці, існує ще таке поняття, як істотна матриця (essential matrix): $E=K'^T F K$. У випадку, коли матриці внутрішніх параметрів будуть одиничними істотна матриця буде збігатися з фундаментальною. За істотною матрицею можна відновити становище і поворот другої камери щодо першої, тому вона використовується в задачах, в яких потрібно визначити рух камери.

Триангуляція точок (triangulation). Тепер власне перейдемо до того, як визначити тривимірні координати точки за координатами її проекцій. Цей процес називається тріангуляцією.

Нехай є дві відкалібровані камери з матрицями P_1 і P_2. x_1 і x_2 - однорідні координати проекцій деякої точки простору X. Тоді можна скласти наступну систему рівнянь:

$$\begin{cases} x_1 = P_1 X \\ x_2 = P_2 X \end{cases}, \tag{1.9}$$

На практиці для вирішення цієї системи застосовується наступний підхід. Векторно множать перше рівняння на x_1, друге на x_2, позбавляються від лінійно залежних рівнянь і призводять систему до виду AX = 0, де A має розмір 4 × 4. Далі можна виходити з того вектор X є однорідними координатами точки, покласти його останню компоненту рівною 1 і вирішувати отриману систему з 3х рівняння з трьома невідомими. Альтернативний спосіб - взяти будь-який ненульовий розв'язок системи A X = 0, наприклад обчислене, як сингулярний вектор, який відповідає найменшому сингулярному числу матриці A.

1.5.4. Побудова карти глибини

Ідея, що лежить в основі побудови карти глибини по стереопарі дуже проста. Для кожної точки на одному зображенні виконується пошук парної їй точки на іншому зображенні. А по парі відповідних точок можна виконати тріангуляцію і визначити координати їх прообразу в тривимірному просторі. Знаючи тривимірні координати прообразу, глибина обчислюється, як відстань до площини камери.

Парну точку потрібно шукати на епіполярной лінії. Відповідно, для спрощення пошуку, зображення вирівнюють так, що б всі епіполярні лінії були паралельні сторонам зображення (зазвичай горизонтальні). Більш того, зображення вирівнюють так, що б у точці з координатами (x_0, y_0) відповідна їй епіполярна лінія задавалася рівнянням $x = x_0$, тоді для кожної точки відповідну їй парну точку потрібно шукати в тій-же сходинці на зображенні з другої камери. Такий процес вирівнювання зображень називають ректифікацією (rectification). Зазвичай ректифікацію здійснюють шляхом ремеппінга

зображення і її поєднують з позбавленням від дисторсій (рис. 1.8) [17].

Рис.1.8. Ректифіковані зображення та відповідна їм карта зсуву

Після того, як зображення ректифіковано, виконують пошук відповідних пар точок. Найпростіший спосіб полягає в наступному (рис. 1.9): для кожного пікселя лівої картинки з координатами (x_0, y_0) виконується пошук пікселя на правій картинці. При цьому передбачається, що піксель на правому зображенні повинен мати координати (x_0 - d, y_0), де d - величина звана невідповідність/зміщення (disparity). Пошук відповідного пікселя виконується шляхом обчислення максимуму функції відгуку, в якості якої може виступати, наприклад, кореляція околів пікселів. В результаті виходить карта зсуву (disparity map).

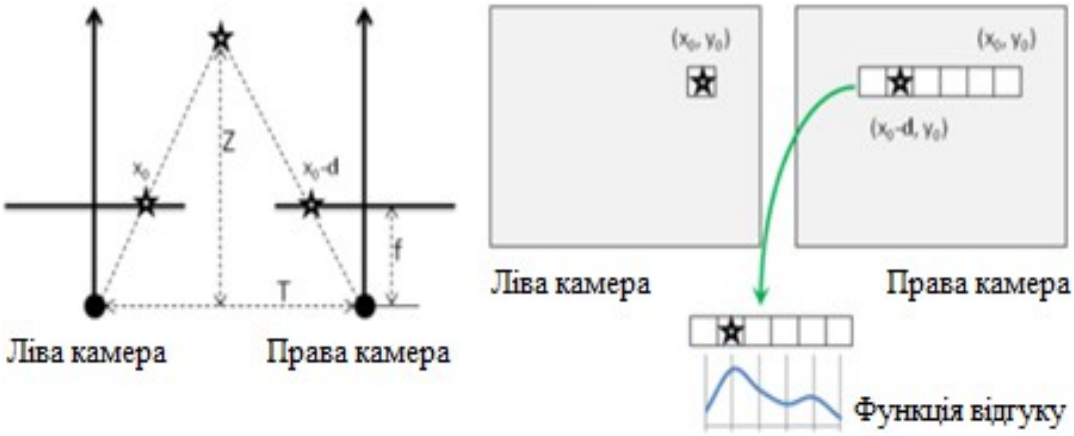

Рис. 1.9. Обчислення карти глибини

Значення глибини обернено пропорційно величині зсуву пікселів. Якщо використовувати позначення з лівої половини рис.1.9, то залежність між

disparity і глибиною можна виразити наступним чином:

$$\frac{T-d}{Z-f}=\frac{T}{Z}\rightarrow Z=\frac{fT}{d}, \tag{1.10}$$

Через зворотню залежність глибини і зміщення, роздільна здатність систем стерео зору, які працюють на основі даного методу, краще на близьких відстанях, і гірше на далеких [3, 4, 5, 11].

Для отримання карти глибини необхідно:

1) Два або більше зображень однієї і тієї ж сцени знятих з різних ракурсів. Зображення повинні відповідати наступним вимогам:

- кут зору повинен бути незалежний від поверхні освітлення (поширення світла не враховується в алгоритмі);
- камери повинні бути відкалібровані;
- деякі алгоритми вимагають дотримання обмежень впорядкованості (в лівому зображенні точки слідують в тому ж напрямку і порядку, що і в правому);
- стереопара повинна бути ректіфікована, тобто має бути виконано перетворення, при якому і ліве, і праве зображення проекціюються на площину, паралельну базової лінії (лінії, що з'єднує оптичні центри камер). Тоді якщо (x, y) - точка правої проекції, а (x_0, y_0) - відповідна точка лівої проекції, то y = y_0. Ректифікація зображень може здійснюватися автоматично, з використанням готових алгоритмів.
- відповідність кордонів інтенсивності і різких перепадів функції розбіжності - поширена вимога. Інтенсивність об'єктів змінюється досить різко на границях;
- для деяких алгоритмів вважається, що кожному пікселю лівого зображення відповідає піксель правого зображення, інакше можна вважати, що він закритий.

2) Зіставлення зображень.

Кожній точці (x, y) лівого зображення повинна бути поставлена у відповідність точка (x_0, y_0) правого зображення (та ж сама точка сцени). Цей

етап вважається найбільш витратним і може накладати обмеження на вхідні дані. Цей етап передбачає отримання функції розбіжності (невідповідності, disparity function), d(x, y): d(x, y) = x_0 - x, (вважаємо, що стереопара ректіфікована, тобто y і y_0 збігаються). Такою, d(x, y) є величина зворотня глибині точки сцени. Тоді карта глибини складається з пікселів, значення яких обернено пропорційні значенням функції зсуву в цій точці.

Варто відзначити, що чим ближче до камери розташований об'єкт сцени, тим більшим буде значення d. Це відбувається завдяки явищу паралакса. Паралакс - зміна видимого положення об'єкту відносно віддаленого фону в залежності від положення спостерігача [18]. Наприклад, коли людина їде на машині і дивиться у вікно, він помітить, що дерево на узбіччі пропаде з поля зору швидше ніж будинок на горизонті.

3) Відновлення тривимірного прообразу точок (рис. 1.10).

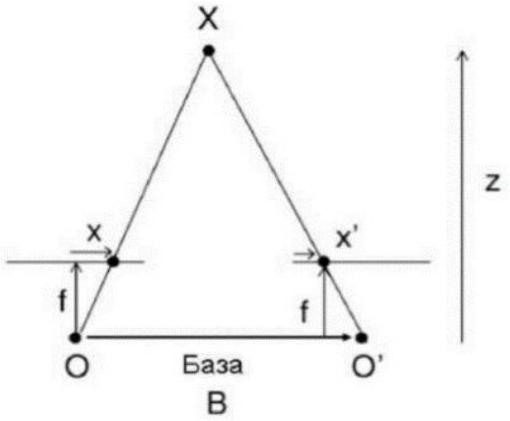

Рис.1.10. Розрахунок координат тривимірної точки

Відстань до об’єкту тут розраховується методом подібних трикутників:

$$Z = \frac{Bf}{d(x,y)}, \tag{1.11}$$

А координати, відповідно, наступним чином:

$$X = \frac{Z\left(x - \frac{w}{2}\right)}{f}, \quad Y = \frac{Z\left(y - \frac{h}{2}\right)}{f}, \tag{1.12}$$

де

w і *h* - ширина та висота зображення;

f - фокусна відстань камери (відстань від центру камери до поверхні її екрану),

B - відстань між точками зйомки двох зображень. Будемо вважати, що кут повороту камери не змінювався при зйомці з двох точок [3].

Встановимо початок системи координат в центр лівої камери (рис. 1.11) та спрямуємо осі системи координат таким чином, щоб вісь OX була спрямована до центра правої камери, вісь OY була спрямована за оптичною віссю лівої камери, до якої повинна бути паралельною оптична вісь правої камери, а напрям осі OZ був спрямований вертикально вгору. Нехай лінія L_1 задає перетин вертикально розташованої площини об'єкта з площиною, в якій розташовані оптичні осі камер.

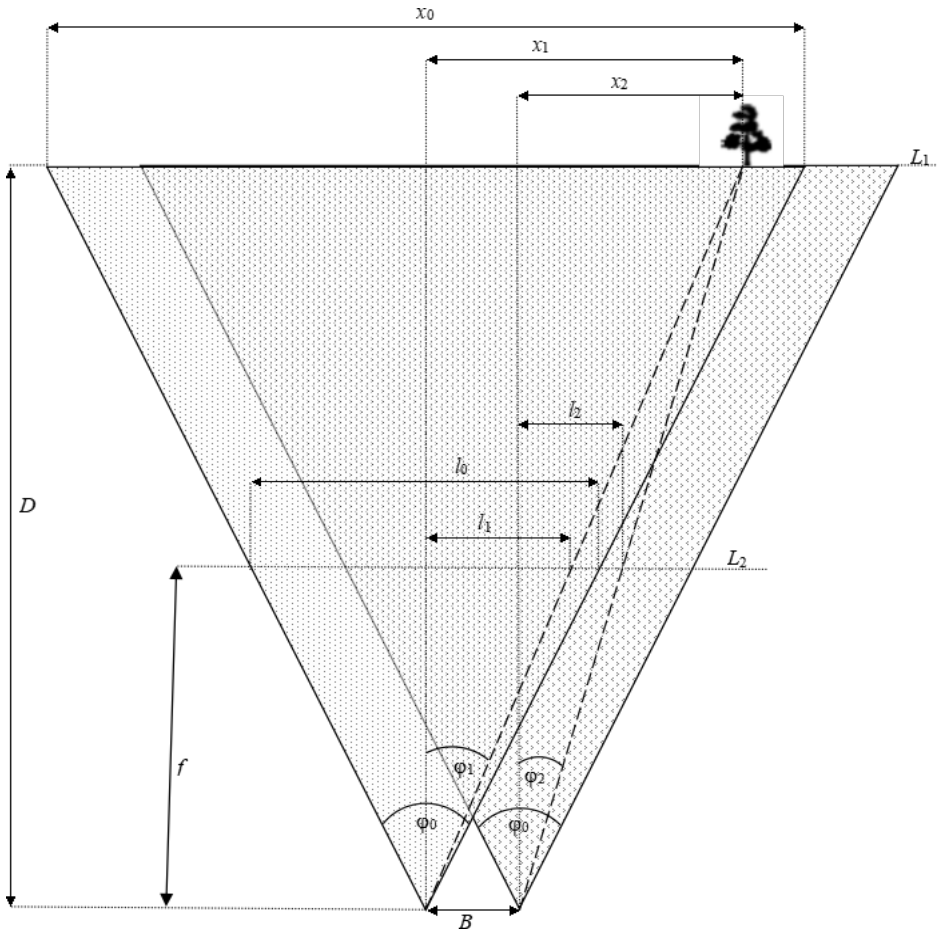

Рис.1.11. Геометрична модель стереоскопічної системи

Тоді з геометричних відношень щодо об’єктної площини (лінія L_1) випливають відношення

$$\begin{aligned} x_1 &= D\tan(\varphi_1) \\ x_2 &= D\tan(\varphi_2) \\ x_1 - x_2 &= B = D[\tan(\varphi_1) - \tan(\varphi_2)] \end{aligned}, \qquad (1.13)$$

де

D – відстань до об’єктної площини.

З (1.13) отримуємо:

$$D = \frac{B}{\tan(\varphi_1) - \tan(\varphi_2)}, \qquad (1.14)$$

У площині зображення об’єкта (лінія L_2) маємо такі співвідношення

$$\tan(\varphi_1) = \frac{l_1}{f}, \quad \tan(\varphi_2) = \frac{l_2}{f}, \quad f = \frac{l_0}{2\tan\left(\frac{\varphi_0}{2}\right)}, \qquad (1.15)$$

З (1.15) отримуємо проміжні формули

$$\tan(\varphi_1) = \frac{l_1}{l_0} 2\tan\left(\frac{\varphi_0}{2}\right), \quad \tan(\varphi_2) = \frac{l_2}{l_0} 2\tan\left(\frac{\varphi_0}{2}\right), \qquad (1.16)$$

Та, нарешті, кінцеву формулу

$$D = \frac{Bl_0}{(l_1 - l_2)2\tan\left(\frac{\varphi_0}{2}\right)}, \qquad (1.17)$$

У загальному випадку, розрахунок відстані до об’єктної площини можна виконати за такою формулою:

$$D = \frac{Bl_0}{2\tan\left(\frac{\varphi_0}{2}\right)|l_1 - l_2|}, \qquad (1.18)$$

де

B – відстань між камерами;

l_0 – ширина огляду вздовж осі Ox;

φ_0 – горизонтальний кут огляду камери;

$|l_1 - l_2|$ – зсув між об'єктом на лівому і правому зображенні.

З урахуванням того, що φ_0=53° та перерахунку l_0 і $|l_1 - l_2|$ в пікселі (тобто $l_0 = p_0 \Delta x$; $l_1 = p_1 \Delta x$, $l_2 = p_2 \Delta x$, де Δx – крок дискретизації вздовж осі OX), отримуємо формулу:

$$D = \frac{B \cdot p_0}{|p_1 - p_2|}, \tag{1.19}$$

де

$|p_1 - p_2|$ – зсув між зображеннями об'єкта в пікселях;

p_0 – загальна кількість пікселів у рядку зображення [3, 4, 5, 6, 7].

Варто відзначити, що зручно було б проводити 3D-моделювання не на дорогому і потужному обладнанні, а на доступному кожному пристрої, наприклад - смартфоні.

Очевидно, що установка спеціальних сенсорів і додаткових камер підвищує вартість гаджета. Тому є сенс в розробці саме програмного забезпечення для обчислення карти глибини і подальшої 3D-реконструкції.

Тобто необхідне ПЗ, яке на вхід отримує одне двовимірне зображення, наприклад фото з камери телефону, виконує обробку, обчислює карту глибини і виконує 3D-моделювання.

1.6. Нейронна мережа

Для вирішення завдання отримання карти відстаней, маючи єдине зображення, можна скористатися технологіями нейронних мереж. Згорткова нейронна мережа (CNN, Convolutional Neural Network) - різновид штучних нейронних мереж, яка входить до складу технологій глибокого навчання. Нейронна мережа складається з рівнів - шарів. За допомогою даної мережі можливо виконувати розпізнавання образів та оцінка карти глибини (рис. 1.12). згортковою мережа називається тому, що основою є операція згортки. Тобто кожна ділянка зображення поелементно множиться на матрицю згортки. Результати підсумовуються і записуються в результуюче зображення.

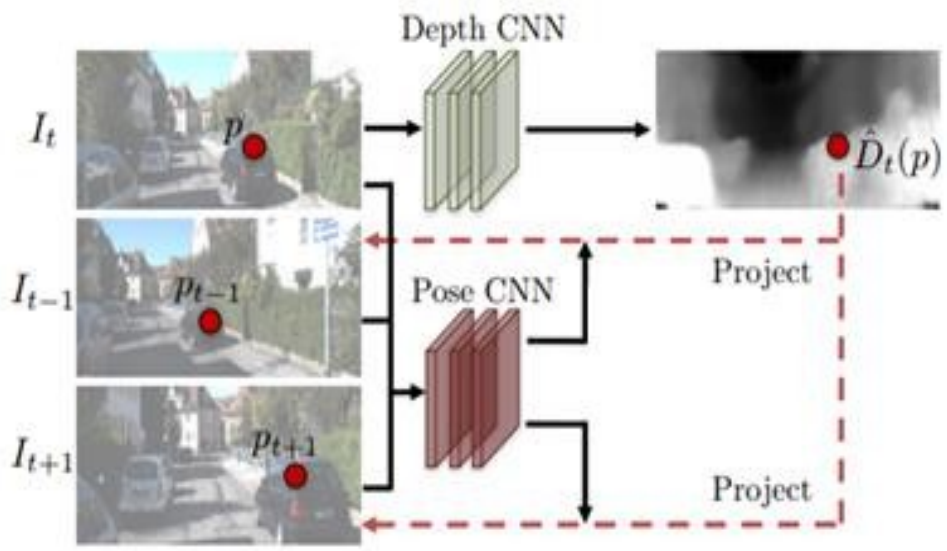

Рис.1.12. Отримання карти глибини із набору зображень (I_t - поточне зображення, I_{t-1} - попередне, I_{t+1} - наступне). Зелені та червоні прямокутники – шари відповідних мереж. p – точка зображення, *Dt(p)* – розрахована відстань до об’єкта зображення у цій точці

Нейронна мережа повинна навчатися на великому наборі даних (dataset). Такий набір розділений на безліч сцен, кожна з яких складається з послідовності зображень знятих з певною частотою при русі камери. Зі схожою частотою сенсор обчислює карту глибини сцени. Результатом навчання мережі є модель, на вхід якої можна подати одне зображення і результатом стане його передбачувана карта глибини.

Важливим завданням є вибір нейронної мережі і визначення набору даних на яких буде навчатися модель.

Нейронна мережа MegaDepth. Монокулярне прогнозування глибини зображення фундаментальна проблема в комп'ютерному зорі. Останнім часом методи глибокого навчання привели до значного прогресу, але вони обмежені доступними даними для навчання. Такі методи процвітають, коли тренуються з великими обсягами даних. На жаль, збір даних загального характеру у вигляді пар (RGB-зображення, карта глибини) виконати дуже складно. Товарні датчики RGB-D, такі як Kinect, широко використовувалися для цієї мети, але обмежуються використанням всередині приміщень. Лазерні сканери

дозволяють використовувати важливі набори даних, такі як Make3D і KITTI, але такі пристрої незручні в експлуатації (у разі промислових сканерів) або створюють карти розріджених глибин (в разі ЛИДАР). Більш того, і Make3D, і KITTI зібрані в певних сценах (університетське містечко і на автомобілі, відповідно).

Поточні набори даних, засновані на 3D-датчиках, мають ключові обмеження, включаючи зображення тільки для приміщень (NYU), невелика кількість навчальних прикладів (Make3D) і розріджену вибірку (KITTI).

MegaDepth пропонує використовувати зображення з точок огляду, що перекриваються, з Інтернету, як практично необмежене джерело для генерування навчальних даних за допомогою сучасних методів, таких як структурований рух (SfM - Structure-from-Motion) і мульти-представлення стерео (MVS - Multi-View Stereo) .

OpenMVS (Multi-View Stereo) – бібліотека комп'ютерного зору, орієнтована на відновлення 3D-моделі по Multi-View Stereo. Хоча існують і інші проекти з відкритим вихідним кодом, призначені для конвеєрів «SfM» (наприклад, OpenMVG), які відновлюють пози камери і розріджену тривимірну хмару точок з набору вхідних данних потока. OpenMVS прагне заповнити цю прогалину, надавши повний набір алгоритмів для відновлення повної поверхні сцени, що підлягає відновленню; вхід являє собою набір пози для камери плюс розріджену хмару точок, а вихід - текстуровану сітку).

Вихідні дані цих систем використовуються в якості вхідних даних методів машинного навчання для прогнозування глибини одного зображення. Використовуючи велику кількість різноманітних навчальних даних з фотографій, зроблених по всьому світу, глибина прогнозується з високою точністю і узагальненістю. Грунтуючись на цій ідеї, сформований великомасштабний набір даних карт глибин MegaDepth (MD).

Існують і інші бібліотеки, що дозволяють будувати карту глибини для зображення, проте вони мають недоліки, які були виправлені в MegaDepth.

Наприклад, деякі проекти можуть передбачати карту глибини для окремих об'єктів, однак цей метод обмежений зображеннями об'єктів, а не сцен.

Інший спосіб збору даних про глибину для навчання - попросити людей вручну додавати маркування глибини на зображеннях, так як люди вміють визначати відносні (порядкові) відносини глибини (наприклад, ближче / далі), але це більш затратно у часі.

Набір даних MegaDepth був побудований в ході виконання наступних етапів. Спочатку були завантажені інтернет-фотографії пам'яток з сайту Flickr.com. Потім кожна пам'ятка була реконструйована у 3D, використовуючи методи SfM і MVS. Так була отримана модель SfM і щільна карта глибини для кожного реконструйованого зображення. Однак, ці карти глибини містять значний шум і викиди, і навчання на них глибокої мережі не дасть корисних результатів. Тому виконується серія етапів обробки, які готують ці карти глибини для використання в навчанні, і додатково використовують семантичну сегментацію для автоматичної генерації порядкових даних глибини (поділ зображення на окремі групи пікселів, області, відповідні одному об'єкту з одночасним визначенням типу об'єкта в кожній області).

Описаний вище підхід використовується для реконструкції 200 3D-моделей з пам'яток по всьому світу, які представляють близько 150K реконструйованих зображень. Після запропонованої фільтрації залишається 130K дійсних зображень. З цих 130K фотографій близько 100K зображень використовуються для Евклідової глибини, а інші 30K зображень використовуються для отримання порядкових даних глибини. Разом ці дані становлять набір даних MegaDepth (MD), доступний за адресою.

З 200 реконструйованих моделей в наборі даних MD випадковим чином вибрано 46 для формування тестового набору (місця розташування, які не використано під час навчання). Для інших 154 моделей випадковим чином виконано розбиття зображення з кожної моделі на навчальні та перевірочні набори з співвідношенням 96% і 4% відповідно. Нейронна мережа реалізована в

PyTorch і навчається з використанням Adam протягом 20 епох з розміром послідовності 32 [8].

PyTorch - це фреймворк глибокого навчання, ключовою особливістю якого є побудова графа обчислень «на льоту», на відміну від тих же TensorFlow і Theano (і надбудови - Keras), в яких спочатку потрібно побудувати граф, а потім «скомпілювати» його і запустити [9].

Адам - алгоритм для градієнтної оптимізації стохастичних цільових функцій першого порядку, заснований на адаптивних оцінках моментів нижчого порядку [10].

Полігональна сітка. За допомогою отриманої карти можливо побудувати полигональную сітку (polygon mesh) - сукупність вершин, ребер і граней, які визначають форму багатогранного об'єкта в тривимірній комп'ютерній графіці і об'ємному моделюванні. Потім полигональную сітку можна заповнити текстурою - вихідним 2D зображенням, або ж виконати деформацію зображення, відповідно до значень глибини (image warping). Так ми отримаємо тривимірну модель видимої частини двовимірного зображення.

1.7. Аналіз наявних існуючих рішень

Пошук існуючих рішень (додатків, функціонал яких відповідає темі проекту) виконувався у PlayMarket за ключовими словосполученнями depth map, depth cam, depth blur, OpenCV. Тестування проводилось на гаджетах моделей Samsung Galaxy J5 та Huawei P10, операційна система - Android.

Найбільш популярні - додатки, що пропонують виконати розмиття деяких областей зображення, але зазвичай для цього не обчислюється карта відстаней. Така програма передбачає, що користувач сам буде відмічати області для розмиття або намалювавши їх, або використавши вже готові фігури-шаблони (кола, прямокутники та ін.). Серед програм найбільш схожих за призначенням до розробленого доатку наступні:

1) BoofCV Computer Vision by BoofCV (4.9 Мб). Демонструє багато можливостей використання методів комп'ютерного зору. Дозволяє отримати карту глибини, але не в звичному чорно-білому вигляді, а кольорову. Це дешо погіршує розуміння відстаней. До того ж для визначення карти необзідно спочатку виконати калібрування. Інструкції для калібрування не є повними і зрозумілими, необхідний шаблон у вигляді шахової дошки, що є не зручним. При тестуванні на розуміння і виконання цього етапу знадобилося близько 20 хвилин. До недоліків ще можна віднести те, що карту глибини неможливо зберігти засобами програми.

2) Мобільний додаток Depth Map by WilliamRoyle (58 Мб). Розробники пропонують свій додаток для отримання карти глибини із знятого смартфоном зображення. Суттєві недоліки: на карті глибини можуть бути познаечні червоні області - неідентифікована відстань, а також цей додаток гарантовано працює лише на одній моделі телефону (LG G5). Девайс на якому проводилося тестування не підтримує цей додаток.

3) Photo 3D by Digital Footmark (1.6Мб). Додаток для створення стерео-шпалер для екрану смартфону. Програма дозволяє користувачу відмітити області глибини зображення та лінію горизонту на свій розсуд, тобто автоматичного розрахування карти відстаней у додатку не передбачено. Це і є головним недоліком додатку.

4) DPTH: Al refocus by RoadAR, Inc (26 Мб). Переваги - автоматичне визначення глибини зображень з девайсів навіть з однією камерою. Програма обчислює карту відстаней, а спеціальні повзунки надають користувачу можливість налаштувати розмиття області зображення з певною глибиною. Обробка зображення - обчислення карти відстаней займає близько 30 секунд (тестування проводилося на різних смартфонах та іноді час обробки становив більше хвилини). Недоліки: зберігти оброблене зображення можна тільки з водяним знаком, для збереження без нього необхідна особлива версія за додаткову плату.

5) 2D to 3D by no-magic.info (0.88 Мб). Додаток дозволяє створити

стерео-шпалери для екрану телефону. У налаштуваннях можна змінити величину паралаксу, обмеження зсуву, точку огляду, обрати маску глибини та ін. Але головним недоліком є те, що обрати власний головний фон (тобто власне зображення, що буде оброблятися) - неможливо. При натисканні на відповідний пункт меню програма видає повідомлення про помилку (на обох девайсях для тестування).

6) God Defocus camera Plus (Bokeh, Portrait) by Chocolate Software 82 (Мб). Розробники обіцяють у реальному часі виконувати розмиття фону. Є можливість обрати у налаштуваннях силу розмиття. Але недоліком є те, що додаток «зависає» і нормальна робота неможлива.

1.8. Висновки до першого розділу

В даному розділі кваліфікаційної роботи були розглянуті можливі засоби отримання карти глибини. Найбільш оптимальним за швидкістю та вартістю рішенням виявилось використання нейронної мережі.

Також, в даному розділі було здійснено порівняльний аналіз існуючих технологій, за тематикою обчислення карти глибини зображення довільної сцени. Аналіз показав, що існуючі технічні рішення незважаючи на їх переваги (можливості розмиття областей зображення, зручний інтерфейс) мають ряд недоліків: низька швидкість роботи, неможливість виконання миттєвої зйомки і подальшого редагування отриманих кадрів, тільки ручне створення карти відстаней та помилки/збої в роботі.

Додатків, які дають змогу, на основі карти глибини, виконувати проекцію зображень на об’єкти фото - не виявлено.

Враховуючи вищезазначені недоліки робимо висновок, що розробка відповідного мобільного додатку з теми визначення карти відстаней до об’єктів зображення та проекціювання зображень на об’єкти сцени є актуальною та потребує подальшого вивчення.

РОЗДІЛ 2
ІНФОРМАЦІЙНА СИСТЕМА ОТРИМАННЯ КАРТИ ВІДСТАНЕЙ ЗОБРАЖЕННЯ ТА ЙОГО ПОДАЛЬШОЇ ОБРОБКИ

2.1. Опис структури системи

Для отримання карти відстаней до об’єктів зображення використана модель нейронної мережі. Якщо подати єдине зображення на вхід такої моделі та задати бажані параметри (розмір зображення, колірну схему, тощо), на виході буде отримана карта глибини. Обчислення, необхідні для отримання результату, є витратними за ресурсами часу та пам’яті. Завданням проекту є створення програмного забезпечення саме для мобільних пристроїв, які зазвичай мають досить обмежену обчислювальну потужність.

Рис.2.1. Схема послідовності обробки даних користувача

Отже, оптимальним вирішенням даної проблеми є розділення системи на дві частини – обробний модуль (серверна частина) та модуль для взаємодії користувача і системи (клієнтська частина). Серверна частина буде реалізована як програма встановлена на комп'ютері, клієнтська - додаток на мобільному пристрої.

Користувач надає доступ до власних даних мобільному додатку, який в свою чергу відправляє дані до серверу. Сервер оцінює глибину за допомогою нейронної мережі та виконує проекціювання зображення. Після цього, змінені дані повертаються до користувача (див. рис. 2.1).

2.2. Серверна частина системи

Для створення обробного модуля - сервера, була використана мова програмування Python, бібліотеки OpenCV, PyTorch, VTK, фреймворк socketserver.

2.2.1. Мова програмування Python

Python - це мова програмування загального призначення, створена Гвідо ван Россум, яка за короткий час стала дуже популярною, в основному завдяки своїй простоті і легкості читання коду. Це дозволяє програмісту висловити свої ідеї в меншій кількості рядків коду, не зменшуючи його читабельність. В порівнянні з іншими мовами, такими як C/C++, Python повільніша. Але ще однією важливою особливістю мови Python є те, що вона може бути легко розширена за допомогою C/C++. Ця можливість допомагає писати обчислювальні коди на C/C++ і створювати для них оболонку Python, для подальшого їх використання в якості модулів Python. Це дає нам дві переваги: по-перше, наш код працює так само швидко, як і вихідний код C/C++ (так як це реальний код C++, що працює у фоновому режимі), і, по-друге, код на Python дуже простий [21].

2.2.2. Бібліотека OpenCV

OpenCV (Open Computer Vision Library, бібліотека комп'ютерного зору з відкритим вихідним кодом [22]) - ліцензована бібліотека з відкритим вихідним кодом, що включає кілька сотень алгоритмів комп'ютерного зору. Крім класичних, в бібліотеку входять і сучасні алгоритми комп'ютерного зору і машинного навчання. У даний час OpenCV підтримує широкий спектр мов програмування, таких як C ++, Python, Java, Matlab та інші, і доступна на різних платформах, включаючи Windows, Linux, OS X, iOS, Android. OpenCV- Python - це Python API (інтерфейс прикладного програмування) для OpenCV. Він співставляє в собі найкращі якості OpenCV C ++ API і мови Python.

OpenCV має модульну структуру, це означає, що пакет включає в себе кілька загальних або статичних бібліотек. Доступні наступні модулі:

- Core functionality (основна функціональність) - компактний модуль, що визначає основні структури даних, включаючи щільний багатовимірний масив Mat і основні функції, використовувані всіма іншими модулями.
- Image processing (обробка зображень) - модуль обробки зображень, що включає лінійну та нелінійну фільтрацію зображень, геометричні перетворення зображення (зміна розмірів, афінне та перспективне викривлення, загальне табличне перепризначення), перетворення колірного простору, гістограми тощо.
- video (відео) - модуль аналізу відео, який включає в себе оцінку руху, віднімання фону і алгоритми відстеження об'єктів.
- calib3d - основні алгоритми багатовидової геометрії, калібрування одиночної та стереокамери, оцінка позиції об'єкта, алгоритми стереовідповідності та елементи 3D-реконструкції.
- features2d - найважливіші детектори особливостей, дескриптори та дескриптори.
- objdetect - виявлення об'єктів і екземплярів попередньо визначених класів (наприклад, обличчя, очі, гуртки, люди, автомобілі тощо).
- highgui - простий у використанні інтерфейс для простих можливостей

інтерфейсу.

- video I/O - простий у використанні інтерфейс для захоплення відео та відеокодеків.
- gpu - GPU-прискорені алгоритми з різних модулів OpenCV.
- деякі інші допоміжні модулі, такі як FLANN і Google тестові обгортки, прив'язки Python та інші [24].

OpenCV Python - це клас-обгортка для вихідної бібліотеки (оригінальної реалізації) C++, яка буде використовуватися з Python. А підтримка Numpy полегшує вирішення розрахункових задач. Numpy - це високо оптимізована бібліотека для числових операцій, що дає можливість використання синтаксису в стилі MATLAB. Всі структури масиву OpenCV перетворюються в масиви Numpy. Тому будь-які операції, які можна виконувати в Numpy, можна комбінувати з OpenCV, що збільшує коло вирішуваних задач Крім того, деякі інші бібліотеки, такі як SciPy, Matplotlib, які підтримують Numpy, можуть бути використані з OpenCV. Таким чином, OpenCV-Python є придатним інструментом для швидкого прототипування проблем комп'ютерного зору [25].

2.2.3. Бібліотека PyTorch

PyTorch - бібліотека машинного навчання для мови Python з відкритим вихідним кодом, створена на базі Torch.

Torch - MATLAB-подібна бібліотека для мови програмування Lua з відкритим виходом коду, надає велику кількість алгоритмів для глибинного навчання та наукових розрахунків. Написана на C, прикладна частина виконана на LuaJIT, підтримуються розрахунки з використанням CUDA і OpenMP [27].

PyTorch забезпечує плавний перехід від прототипування (швидка «чорнова» реалізація базової функціональності для аналізу роботи системи в цілому) досліджень до розгортання виробництва. Використовується для обробки мови. Розробляється переважно групою штучного інтелекту Facebook. Бібліотека «Pyro» компанії Uber для ймовірнісного програмування

заснована на PyTorch.

PyTorch надає дві основні високорівневі моделі:

- Тензорні розрахунки (за аналогією з NumPy) з розвиненою підтримкою прискорення на GPU.
- Глибокі нейронні мережі на базі системи autodiff (AD, Автоматична диференціація - технологія для автоматичного вдосконалення комп'ютерних програм. Набір методів для чисельної оцінки похідної функції, заданої програмою. Принцип AD: кожна програма, виконує послідовність елементарних арифметичних операцій (+, -, *, / ...) і функцій (exp, log, sin, cos...). Тому, застосовуючи правило ланцюга кілька разів для цих операцій, похідні будь-якого порядку можуть бути обчислені автоматично, використовуючи не набагато більше арифметичних операцій, ніж вихідна програма [26]).

Тензори не є чимось особливим, це просто багатомірні масиви. Тензори PyTorch схожі на масиви пакета Numpy, але додатково можуть оброблятися на відеоприскорювачах. PyTorch підтримує різні типи тензорів.

- Модуль Autograd. PyTorch використовує метод автоматичної диференціації. Виконується запис розрахунків, отриманих в прямій спрямованості, після чого виконується відтворення в зворотньому напрямку для розрахунку градієнтів. Цей метод особливо корисний при побудові нейронних мереж, оскільки дозволяє розраховувати диференціальні поправки параметрів одночасно з прямим проходом.
- Модуль Optim. torch.optim - модуль, що реалізує кілька алгоритмів оптимізації, які використовуються при побудові нейронних мереж. Реалізована більшість найпоширеніших та часто використовуваних методів.
- Модуль nn. Модуль PyTorch autograd дозволяє легко визначити розрахункові графи і працювати з градієнтами, однак може бути занадто низьким рівнем для визначення складних нейронних мереж. Більш високорівневою абстракцією для таких прикладів є модуль nn [28].

2.2.4. Фреймворк socketserver

Для реалізації можливості підключення обробного модуля і мобільного додатку була використана технологія сокетів, а саме фреймворк socketserver - фреймворк для мережевих серверів.

Модуль socketserver спрощує написання мережевих серверів. Існує чотири основні класи конкретних серверів (class socketserver):

- TCPServer - тут використовується Internet протокол TCP, який забезпечує безперервний потік даних між клієнтом і сервером; транспортний протокол передачі даних, попередньо встановлює з'єднання з мережею.
- UDPServer - використання дейтаграм, які є дискретними пакетами інформації, що можуть надходити не по порядку або бути втрачені під час транзиту; транспортний протокол, передача повідомлень-дейтаграм без необхідності установки з'єднання в мережі.
- UnixStreamServer та UnixDatagramServer - це менш поширені класи подібні до класів TCPServer і UDPServer, але тут використовуються сокети домену Unix; вони не доступні на не-Unix-платформах.

Ці чотири класи синхронно обробляють запити; кожен запит має бути завершений до того, як може бути запущений наступний запит. Такий підхід не є актуальним, якщо кожен запит триває довго, оскільки вимагає великих обчислень або тому, що він повертає багато даних, які клієнт повільно обробляє. Рішення полягає у створенні окремого процесу або потоку для обробки кожного запиту; У класах ForkingMixIn і ThreadingMixIn є підтримка асинхронної поведінки.

Створення сервера вимагає декількох кроків. По-перше, необхідно створити клас обробника запитів, створивши півклас класу BaseRequestHandler і визначити його метод handle(); цей метод оброблятиме вхідні запити. По-друге, необхідно створити екземпляр одного з класів сервера, передавши йому адресу сервера і клас обробника запитів. Рекомендується використовувати сервер із конструкцією with. Потім необхідно використати метод

handle_request() або serve_forever() об'єкта сервера для обробки одного або декількох запитів. Щоб закрити сокет (якщо не було використано оператор with) необхідно викликати server_close().

При успадкуванні від ThreadingMixIn для реалізації потокового підключення, необхідно явно заявити, як потоки будуть поводитися в умовах різкого вимкнення. Клас ThreadingMixIn визначає атрибут daemon_threads, який вказує, чи повинен сервер чекати завершення потоку. Необхідно встановити прапорець явно, щоб потоки діяли автономно; за замовчуванням - False, тобто Python не завершить роботу, доки всі потоки, створені ThreadingMixIn, не завершаться. Серверні класи мають однакові зовнішні методи і атрибути, незалежно від того, який мережевий протокол вони використовують [29].

2.2.5. Бібліотека VTK

The Visualization Toolkit (VTK, інструментарій візуалізації) - це програмне забезпечення з відкритим кодом для маніпулювання та відображення наукових даних. VTK - відкрита кросплатформна програма для тривимірного моделювання, обробки зображень і прикладної візуалізації. Бібліотека VTK написана на C++, проте також доступні модулі на Tcl/Tk, Java і Python. Вона включає найсучасніші інструменти для 3D-рендеринга, набір віджетів для 3D-взаємодії і широкими можливостями обробки 2D-графіки.

VTK є частиною набору підтримуваних платформ Kitware для розробки програмного забезпечення. Платформа використовується у всьому світі в комерційних додатках, а також у дослідженнях і розробках.

Спочатку програма була створена в 1993 році як додаток до книги «The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics» («Інструментарій візуалізації: Об'єктно орієнтований підхід до тривимірної графіці») видавництва Prentice-Hall. Авторами книги і програми є три розробника: Will Schroeder, Ken Martin і Bill Lorensen, які написали їх у вільний від роботи час [30, 31].

2.3. Клієнтська частина

Розробка додатку виконувалася у середовищі програмування QtCreator, що була обрана через простоту інтерфейсу та завдяки можливості налагоджування програм з використанням Android емулятора. Створення графічного інтерфейсу додатку було реалізовано за допомогою QML, а логічна частина програми - зв’язок елементів інтерфейсу, реакція програми на дії користувача, обмін даними з обробним модулем, була задана за допомогою Qt.

2.3.1. Поєднання QT та QML

Qt - кросплатформний фреймворк для розробки програмного забезпечення на мові програмування C++ [32].

QML, шо входить до складу Qt, надає повну свободу дій при створенні інтерфейсу користувача.

QML (Qt Meta Language або Qt Modeling Language) - декларативна мова програмування, заснована на JavaScript, призначена для створення високодинамічних додатків. QML є частиною Qt Quick, середовища розробки призначеного для створення інтерфейсу користувача, поширюваного разом з Qt. В основному використовується для створення додатків, орієнтованих на мобільні пристрої з сенсорним управлінням.

QML-документ являє собою дерево елементів. QML елемент, так само, як і елемент Qt, являє собою сукупність блоків: графічних (таких, як rectangle, image) і поведінкових (таких, як state, transition, animation). Ці елементи можуть бути об'єднані, для побудови комплексних компонент, починаючи від простих кнопок і повзунків, закінчуючи повноцінними додатками, що працюють з інтернетом [33].

Також для елементів встановлюються різні властивості, що визначають поведінку програми. Поведінка програми може бути додатково написана через JavaScript. Крім того, QML інтенсивно використовує Qt, що дозволяє типам та

функціям Qt бути доступними безпосередньо з додатків QML [34].

Модуль Qt QML надає платформу для розробки додатків і бібліотек на мові QML. Він визначає і реалізує інфраструктуру мови і движка і надає API, що дозволяє розробникам додатків розширювати мову QML за допомогою призначених для користувача типів і інтегрувати код QML з JavaScript і C ++. Модуль Qt QML надає як API QML, так і API C++. Хоча модуль Qt QML надає мову для додатків QML, модуль Qt Quick надає безліч візуальних компонентів, підтримку перегляду моделей, анімації та багато іншого для створення користувацьких інтерфейсів [34].

Модуль Qt Quick - це стандартна бібліотека для написання додатків QML. У той час, як модуль Qt QML забезпечує QML-движок і мовну інфраструктуру, модуль Qt Quick надає всі основні типи, необхідні для створення інтерфейсів користувача з QML. Він надає візуальне полотно і включає типи для створення та анімації візуальних компонентів, отримання вхідних даних користувача, створення моделей даних і видів. Модуль Qt Quick забезпечує як QML API, що надає типи QML для створення інтерфейсів користувача з мовою QML, так і C++ API для розширення програм QML з кодом C++.

Qt Quick надає все необхідне для створення багатофункціонального додатку з плавним і динамічним інтерфейсом користувача. Вона дозволяє користувальницьким інтерфейсам будуватися навколо поведінки компонентів інтерфейсу користувача і як вони з'єднуються один з одним, і забезпечує візуальне полотно з власною системою координат і двигуном рендерингу [34].

2.3.2. Механізм сигналів та слотів

Мета-об'єктна система (Qt Meta Object System) забезпечує механізми сигналів і слотів для міжоб'єктного зв'язку. Ознаки такої системи:

1. Клас QObject є базовим класом для об'єктів, що підтримують механізм системи мета-об'єктів.

2. Макрос Q_OBJECT всередині приватного розділу опису класу використовується для включення функцій мета-об'єктів, таких як динамічні властивості, сигнали і слоти.

3. Компілятор мета-об'єкта (moc, Meta-Object Compiler) забезпечує кожний підклас QObject необхідним кодом для реалізації особливостей мета- об'єкта.

Мос читає вихідний файл C++. Якщо він знаходить опис класів, що містять макрос Q_OBJECT, він створює інший вихідний C++ файл, який містить мета - об'єктний код для кожного з цих класів. Цей згенерований вихідний файл або #include'd у вихідний файл класу, або, як правило, компілюється і пов'язується з реалізацією класу. Крім забезпечення сигналів і слотів механізмом для зв'язку між об'єктами (основна причина введення системи), мета-об'єктний код надає такі додаткові можливості:

- QObject::metaObject() повертає асоційований мета-об'єкт для класу.
- QMetaObject::className() повертає ім'я класу як рядок під час виконання, не вимагаючи підтримки нативної інформації про тип виконання (RTTI) через компілятор C++.
- Функція QObject::inherits() повертає, чи є об'єкт екземпляром класу, який успадковує вказаний клас у дереві успадкування QObject.
- QObject::tr() і QObject::trUtf8() переводять рядки для інтернаціоналізації.
- QObject::setProperty() і QObject::property() динамічно встановлє і отримує властивості за назвою.
- QMetaObject::newInstance() створює новий екземпляр класу.
- Також можливе виконання динамічного приведення типів за допомогою qobject_cast(). Функція qobject_cast() поводиться аналогічно до стандартного C++ dynamic_cast(), з перевагами, що не вимагає підтримки RTTI і працює через динамічні бібліотеки. Якщо приведення типу успішне (об'єкт має сумісний тип, що визначається під час виконання), функція повертає ненульовий покажчик, або 0 (якщо тип несумісний). Наприклад, припустимо, що MyWidget успадковується від QWidget і оголошується макросом Q_OBJECT: QObject *obj = new MyWidget;.

Змінна obj типу QObject *, насправді відноситься до об'єкта MyWidget, тому ми можемо належним чином передати його - QWidget *widget = qobject_cast<QWidget *>(obj);.

Приведення від QObject до QWidget є успішним, оскільки об'єкт фактично є MyWidget, який є підкласом QWidget [34].

Взаємодія з об'єктами QML з C++. Всі типи об'єктів QML походять від типу QObject, незалежно від того, чи вони внутрішньо реалізовані двигуном або визначені сторонніми джерелами. Це означає, що движок QML може використовувати Qt Meta Object System для динамічного створення будь-якого типу об'єкта QML і перевірки створених об'єктів. Тоді є можливим інтегрувати дані об'єктів QML у програму C ++. Як тільки об'єкт QML створений, його можна отримати з C++ для того, щоб встановлювати властивості, викликати методи і отримувати повідомлення про сигнали.

Завантаження QML-об'єктів з C++. Документ QML можна завантажити з використанням QqmlComponent, що розглядає документ QML як об'єкт C++, який потім може бути змінений з коду C++. Наприклад, програмний код файлу MyItem.qml наведений на Лістингу 1.

Лістинг 1 – Приклад найпростішого QML документу.

```
import QtQuick 2.0
Item { width: 100; height: 100 }
```

Цей документ QML можна завантажувати за допомогою QQmlComponent. Це вимагає виклику QQmlComponent::create() для створення нового екземпляра компонента. object є екземпляром створеного компонента MyItem.qml. Змінити властивості елемента можна за допомогою QObject::setProperty() (Лістинг 2).

Лістинг 2 – Створення екземпляру компоненту QML документу за допомогою QQmlComponent.

```
QQmlEngine engine;
QQmlComponent component(&engine,
QUrl::fromLocalFile("MyItem.qml"));
QObject *object = component.create();
```

```
...
object->setProperty(“width”, 500);
delete object;
```

Альтернативно, можна передати об'єкт до його фактичного типу і викликати певні методи. У цьому випадку базовим об'єктом MyItem.qml є елемент, який визначається класом QquickItem (Лістинг 3).

Лістинг 3 - Створення екземпляру об'єкту за допомогою QquickItem.

```
QQuickItem *item = qobject_cast<QQuickItem*>(object);
item->setWidth(500);
```

Клас QQmlEngine забезпечує середовище для створення компонентів QML. Кожен компонент QML створюється в QQmlContext. QQmlContext є необхідним для передачі даних до компонентів QML. У QML контексти розташовані ієрархічно, і ця ієрархія управляється QQmlEngine. До створення будь-яких компонентів QML програма повинна створити QQmlEngine, щоб отримати доступ до контексту QML. Наступний приклад показує, як створити простий текст. Контексти надають дані, які будуть піддаватися впливу компонентів QML, створеним за допомогою механізму QML. Кожен QQmlContext містить набір властивостей, відмінних від його властивостей QObject, які дозволяють явно прив'язувати дані до контексту за назвою. Властивості контексту визначаються та оновлюються викликом QQmlContext::setContextProperty(). Об’єкт (наприклад модель) створювана у Qt повинна бути прив'язана до контексту, так він зможе бути доступний з QML-файлу.

Доступ до завантажених об'єктів QML за назвою об'єкта. Компоненти QML мають нащадків, що можуть бути отримані за допомогою властивості QObject::objectName з QObject::findChild(). Наприклад, якщо кореневий елемент в MyItem.qml мав дочірній елемент Rectangle, нащадок може бути отриманий, як наведено у Лістингу 4.

Лістинг 4 – Отримання та змінення властивостей нащадку заданого

об’єкту.

```
QObject *rect = object->findChild<QObject*>("rec");//rec-
ObjectName
if (rect)
    rect->setProperty("color", "red");// встановлення кольору
```

Якщо об'єкт має декілька дітей з однаковим об'єктним ім'ям, необхідно використовувати QObject::findChildren() може використовуватися для пошуку всіх дітей з відповідним ім'ям objectName.

Підключення сигналів QML до слотів. Всі сигнали QML автоматично доступні для C++ і можуть бути підключені до слотів за допомогою QObject::connect(), як і будь-який звичайний сигнал Qt C++.

Компонент QML з сигналом qmlSignal, який випускається з параметром типу string. Цей сигнал підключається до слота об'єкта C++ за допомогою QObject::connect() так, що метод cppSlot() викликається кожного разу, коли випускається qmlSignal [34]

2.4. Висновки до другого розділу

У другому розділі кваліфікаційної роботи запропонована структура системи отримання карти відстаней до об'єктів зображення з використанням технологій нейронних мереж, що дає можливість виконувати подальшу обробку вихідного зображення, а саме проекціювання інших зображень на об’єкти, відповідно до отриманих даних про глибину сцени.

Така система повинна складатися з двох частин: серверної і клієнтської. На підставі того, що обчислення карти відстаней - досить ресурсовитратна операція і потребує значних витрати часу, вирішено було виконувати обробку зображення на більш потужному сервері – стаціонарному комп’ютері. Це зменшить час очікування результатів. Мобільний додаток, в свою чергу, забезпечить зручний інтерфейс для введення даних користувача до системи.

Враховуючи те, що Python є поширеною мовою програмування для

роботи із нейронними мережами, сервер планується реалізовувати з її використанням. Для обробки зображень буде використано бібліотеки OpenCV, PyTorch та VTK, а для обміну даними із клієнтом - фреймворк socketserver.

Для створення мобільного додатку обрано фреймворк Qt, що дозволяє описати логічну частину програми (зв’язок елементів інтерфейсу, реакція програми на дії користувача, обмін даними з обробним модулем) на мові програмування C++, а інтерфейс додатку – на мові QML (яка заснована на JavaScript і надає можливість задавати поведінку елементів керування окремо від основної логіки програми).

РОЗДІЛ 3
ДОСЛІДЖЕННЯ РОЗРОБЛЕНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Опис структури проекту мобільного додатку

Для надання можливості взаємодії користувача із програмою, а також для забезпечення взаємодії елементів керування додатку, до структури проекту було додано: 2 файли заголовків з розширенням .h, 3 файли вихідного коду - .cpp, 4 файли з елементами інтерфейсу додатку - .qml, 1 файл ресурсів проекту, який містить назви вихідних файлів (крім main.cpp) і файлу проекту - .qrc, 1 файл проекту - .pro та 1 файл .java, що забезпечує спеціальні операції (рис. 3.1).

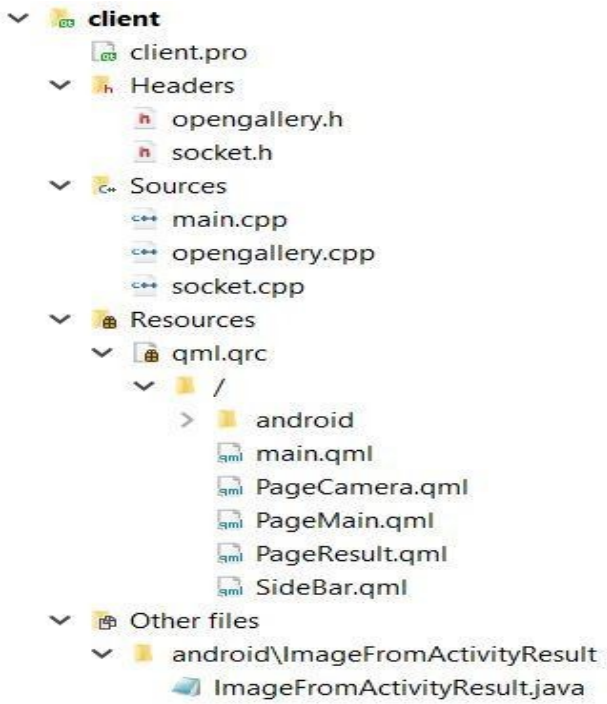

Рис.3.1. Структура проекту мобільного додатку

Структура C++ файлів

Структура C++ файлів проекту Qt аналогічна структурі файлів вихідного коду проекту C++ (Лістинг 5). Спочатку необхідний опис використання попередньо скомпільованих заголовків (PCH). Точкою входу в програму є функція main.

Лістинг 5 - Мінімальний вихідний код програми Qt.

```
#include <QApplication>
int main(int argc, char **argv) // точка входу
{
QApplication app (argc, argv);// створення екземпляру QApplication
return app.exec(); // запуск додатку
}
```

QApplication - клас, що бере до уваги крім вхідних аргументів та ін., цикл подій або event loop. Event loop - це цикл, який очікує вводу даних від користувача до інтерфейсу програми (GUI).

При виклику app.exec() запускається цикл подій і чекає на події задані для обробки. Такою подією може бути, наприклад, натискання кнопки на графічному інтерфейсі.

Структура файлу проекту

Процес збірки додатку відбувається за допомогою qmake. Це утиліта зі складу Qt, яка допомагає здійснити процес збірки для різних платформ. qmake автоматично генерує make-файли, засновуючись на інформації з файлу проекту файл з розширенням .pro.

Прості файли проекту використовують декларативний стиль, визначаючи стандартні змінні для вказівки вихідних файлів та файлів заголовків, які використовуються в проекті.

У файлі проекту змінні використовуються для зберігання списків рядків. У найпростіших проектах ці змінні повідомляють qmake про параметри конфігурації, що використовуються, або надають назви файлів і шляхи для використання в процесі збирання.

qmake шукає певні змінні в кожному файлі проекту, і використовує їх, щоб визначити, що написати в Makefile. Наприклад, списки значень у змінних HEADERS і SOURCES використовуються, щоб повідомити qmake про заголовок і вихідні файли в тому ж каталозі, що й файл проекту [34]. В ДОДАТКУ А наведено використання змінних на прикладі файлу client.pro.

Структура та елементи QML файлу

Кожен елемент в QML файлі спирається на конкретний C++ клас. Ці класи лише експортуються в QML, є приватними і недоступні користувачу безпосередньо з C++. Елементи малюються на екрані через OpenGL за допомогою QML Scenegraph (Дерево сцени QML).

У перших строках файлу необхідно зазначити які бібліотеки будуть використані при створені елментів. Імпортувати бібліотеки до програми можна за допомогою оператора імпорту import у файлі .qml - import QtQuick 2.11.

Qt Quick надає основні будівельні блоки для всіх користувальницьких інтерфейсів, таких як об'єкти для виводу зображень і тексту, а також для обробки вхідних даних користувача. Після імпорту типи та функціональні можливості, які надає Qt Quick можна використовувати в документі QML.

Бібліотека QtQuick QML забезпечується модулем Qt Quick. Qt Quick Controls 2 надає набір елементів керування, які можна використовувати для побудови повного інтерфейсу.

Після імпорту необхідних бібліотек необхідно задати кореневий елемент додатку. Таким є ApplicationWindow - тип QML, що позначає вікно верхнього рівня.

У середині ApplicationWindow можуть міститися будь-які інші елементи, наприклад, використані у даному додатку SwipeView, TabBar, Item, Popup, Page, Rectangle, Button, Label, Text та інші. Кожен об’єкт має властивості, що встановлюють як буде відображений об’єкт на візуальній сцені (Лістинг 6).

Лістинг 6 - Опис об'єкта у документі QML

```
Тип_ об’єкту {
властивість1: значення1 властивість2: значення2
    ...
}
```

Розглянемо елемент SwipeView у вікні верхнього рівня, файл main.qml. SwipeView, як і будь-який елемент, може мати: id – назву, за якою інші

елементи ідентифікують або звертаються до нього. objectName – назву, за якою елемент доступний із C++ файлів вихідного коду. Аналогічно із більшістю елементів, SwipeView може задавати своє розташування за допомогою властивості anchors.

Визначення персоналізованих типів QML повторного використання.

Одним з найважливіших понять у QML є тип повторного використання. Додаток, може мати декілька подібних елементів (наприклад, кнопок). QML дає змогу позначити їх як повторно використовувані, спеціальні типи, щоб мінімізувати дублювання коду. Для цього розробник може описати новий тип в окремому QML файлі, і цей тип стає можливим повторно використовувати кілька разів у програмі.

SwipeView містить 3 вкладки, які задані назвами користувальницьких типів, попередньо описаних у QML файлах PageCamera.qml, PageMain.qml, PageResult.qml (Лістинг 7).

У властивості елементу currentIndex необхідно задати номер вкладки (з 0), що буде відображений при відкритті додатку. Рядок Component.onCompleted: contentItem.interactive = false відповідає за неможливість переходу між вкладками горизонтальним «свайпом» по екрану.

Лістинг 7 – Опис елементу SwipeView

```
SwipeView {
id: swipeView // идентифікатор
objectName: “swipeView” //ім’я об’єкту
anchors.fill: parent // розміри - як у предка
currentIndex: 1 // перша сторінка для відображення
// заборона переміщення свайпом
Component.onCompleted: contentItem.interactive = false
}
```

Такий запис визначає ієрархію об'єктів з кореневим об'єктом SwipeView, який має дочірні об'єкти PageCamera, PageMain, PageResult. Предок цих об'єктів автоматично встановлюється в SwipeView, і аналогічно, ці об'єкти додаються до властивості children об'єкта SwipeView [34].

Обробка вхідних даних користувача.

Серед властивостей об’єктів можна зазначити реакцію на певні події викликані діями користувача. Реакція елементу задається у фігурних дужках програмним кодом мови javascript. Наприклад, при натисканні на кнопку, можна вивести повідомлення або змінити властивості чи викликати певні події іншого елементу (Лістинг 8).

Лістинг 8 – Опис кнопки та її реакції на подію натискання

```
Button {
  id: firstButton
  onClicked: { // події у відповідь на клік
 console.log(“Some information”) // вивід інформації у консоль
 secondButton.text = “Active” //зміна тексту певного елементу
 sideBar.open() // виклик події певного елементу
     }
}
```

Взаємодія C++ та QML

Взаємодія C++ та QML реалізована за допомогою механізму сигналів та слотів. Елементи QML мають змогу випускати сигнали. Для опису сигналу необхідно скористатися ключовим словом signal, у якості прикладу розглянемо сигнал setPitch - встановлення сили деформації, файл main.qml (Лістинг 9).

Лістинг 9 – Опис сигналу у QML документі

```
...
ApplicationWindow { // головне вікно
    id: window // ідентифікатор
    signal setPitch(int pitch) // опис сигналу, аргумент pitch
...
}
```

Сигнал буде подано, при натиску «Send» файлу PageMain.qml (Лістинг 10).

Лістинг 10 – Виклик сигналу у відповідь на подію натискання кнопки

```
ToolButton { // елемент - кнопка меню
    id: buttonSend // ідентифікатор
    text: “Send” // текст
    onClicked: {... setP (p);...} // виклик сигналу за кліком
... }
```

У C++ частині необхідно описати слот, що буде приймати сигнал та у відповідь на нього виконувати певні дії, задані розробником. Слоти повинні описуватись у частині public slots, файл socket.h (ДОДАТОК А).

У файлі вихідного коду Socket.cpp позначено дії, що виконаються коли від QML елемента надійде сигнал - до значення змінної p буде встановлено значення аргументу методу-слоту (ДОДАТОК А).

Для того, щоб пов’язати сигнал і слот, та щоб у разі надходження сигналу з QML виконався метод-слот необхідного екземпляру класа, необхідно:

1. Створити екземпляр класу QQmlApplicationEngine, що надає можливість завантаження програми з одного файлу QML. Цей клас поєднує в собі QQmlEngine і QQmlComponent.

2. Завантажити програму з файлу main.qml за допомогою load(). Отримати кореневий об’єкт (що має нульовий індекс в ієрархії).

3. Створити екземпляр класу, слот якого виконається при надходженні сигналу.

4. Використати функцію connect, що створює з'єднання даного типу від сигналу в об'єкті відправника до методу в об'єкті приймача (Лістинг 11). Повертає дескриптор до з'єднання, яке можна використовувати для відключення його пізніше.

Лістинг 11 – Синтаксис функції connect

```
QMetaObject::Connection QObject::connect(const QObject *sender, PointerToMemberFunction signal, const QObject *receiver, PointerToMemberFunction method, Qt::ConnectionType type = Qt::AutoConnection)
```

Сигнал повинен бути функцією, описаною як сигнал в файлі заголовку. Слот може бути приєднаний до заданого сигналу, якщо сигнал має стільки аргументів, що і слот, і існує неявне перетворення між типами відповідних аргументів в сигналі і слоті. Приклад використання у файлі main.cpp наведений на Лістингу 12.

Лістинг 12 – Використання функції connect у вихідному файлі main.cpp

```
QQmlApplicationEngine engine;
engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
if (engine.rootObjects().isEmpty()) // якщо не містить елементів
return -1; // завершення роботи
QObject* root = engine.rootObjects()[0];//отримання кореневого ел.
Socket *sendI = new Socket(root); //створ. екз. класу
QObject::connect(root, SIGNAL(setP(int)), sendI, SLOT(setP(int)));
//з'єднання сигналу та слоту
```

Наведений програмний код дозволить встановлювати значення змінної описаної в файлі заголовку при кожному поданні сигналу у QML файлі.

Визначення типів QML з C ++

Клас C++, нащадок QObject, може бути зареєстрований у системі типів QML, щоб дозволити використовувати його як тип даних у межах QML-коду. Після створення екземпляра класу можна керувати ним з QML, мати доступ до атрибутів, методів та сигналів класу.

Необхідно викликати qmlRegisterType(), щоб зареєструвати клас як тип QML у просторі імен певного типу (Лістинг 13). Потім надається можливість імпортувати цей простір імен для використання типу.

Лістинг 13 – Синтаксис функції qmlRegisterType

```
int qmlRegisterType(const char *uri, int versionMajor, int
versionMinor, const char *qmlName)
```

Ця функція реєструє тип C++ у системі QML, щоб зробити його доступним у просторі імен з uri, з ім'ям qmlName, з номером версії versionMajor.versionMinor. Повертає ідентифікатор типу QML.

У проекті створено клас OpenGallery, що описується в фалі заголовку opengallery.h та має файл вихідного коду opengallery.cpp. Приклад реєстрації типу MyType, з файлу main.cpp наведено в ДОДАТКУ А. Після приведеного опису, необхідно імпортувати тип у QML файл. Наприклад, у початку файлу PageMain.qml необхідно додати import MyType 1.0 [34].

3.2. Відкриття зображення з галереї телефону

Для вибору зображення користувачу зручно було б використовувати стандартну галерею пристрою, в якої добре знайомий та інтуїтивно зрозумілий інтерфейс. Для реалізації такої можливості недостатньо наявних у QML типів. Отже, було створено новий тип на основі QQuickPaintedItem.

Клас QPainter виконує прорисовку низького рівня на віджетах та інших пристроях для прорисовки. QPainter надає високо оптимізовані функції для виконання більшості графічних операцій. Він може виконувати прорисовку ліній, складних форм (сектори, хорди), вирівняного тексту і растрових зображень. QPainter може працювати на будь-якому об'єкті, який успадковує клас QPaintDevice з аргументами у конструкторі QImage, QOpenGLPaintDevice, QPagedPaintDevice, QPaintDeviceWindow, QPicture, and QPixmap [34].

Клас QQuickPaintedItem надає можливість використання API QPainter у графі сценарію QML (QML Scene Graph). Він встановлює текстурований прямокутник у графі сцени і використовує QPainter, щоб намалювати текстуру. Цільовим об’єктом рендерінгу може бути QImage. Тоді QPainter спочатку перетворюється на зображення, після чого його вміст завантажується до текстури.

Щоб написати власний елемент з прорисовкою, необхідно успадкувати QQuickPaintedItem, а потім реалізувати чисту віртуальну загальнодоступну функцію: paint(), що й виконує прорисовку зображення. Зображення буде знаходитися всередині прямокутника, що охоплює координати від 0,0 до width(), height(). update() викликається, щоб виконати перерисовку paint().

Отже, цей механізм використано для прорисовки обраного з галереї зображення в створеному елементі QML [34].

Для відкриття галереї пристрою, створений клас успадковується також від QAndroidActivityResultReceiver. Для його використання необхідно у файл проекту додати QT += androidextras.

QAndroidActivityResultReceiver - інтерфейс, що використовується для

зворотних викликів з onActivityResult() в основній Android Activity. Створено підклас цього класу для оповіщення про результати, використовуючи API QtAndroid::startActivity(). Функція handleActivityResult повинна бути перевизначена, щоб отримати результати операції після її запуску за допомогою QtAndroid::startActivity() (Лістинг 14).

Лістинг 14 – Синтаксис функції для отримання результатів операції після її запуску за допомогою QtAndroid::startActivity()

```
Void QAndroidActivityResultReceiver::handleActivityResult(int receiverRequestCode, int resultCode, const QAndroidJniObject &data)
```

receiverRequestCode є кодом запиту, унікальним для цього приймача, який спочатку був переданий до функцій startActivity(). ResultCode є результатом, що повертається операції, і data - це або null, або об'єкт Java класу android.content.Intent. Обидва останні аргументи ідентичні аргументам, переданим на onActivityResult() [35].

Activity (операція) - це компонент додатку, який виводиться екран, і з яким користувач може взаємодіяти для виконання будь-яких операцій, наприклад, набрати номер телефону, зробити фото. Кожній операції присвоюється вікно для прорисовки відповідного інтерфейсу користувача. Зазвичай, вікно відображається на весь екран.

Як правило, додаток складається з декількох activity, які слабо пов'язані з іншим. Одна з операцій в додатку визначається як «основна», виводиться при першому запуску програми. У свою чергу, кожна операція може запустити іншу операцію для виконання різних дій. Кожного разу, коли запускається нова операція, попередня операція зупиняється, однак система зберігає її в стеці («стек зворотних переходів»). При запуску нової операції вона поміщається в стек переходів назад і нової операція відображається для користувача. Стек переходів назад працює за принципом «останній вхід - перший вихід», тому після того, як користувач завершить поточну операцію і натисне кнопку Назад (поточна операція буде видалена зі стеку і знищена), і повернеться до

попередньої операції.

Коли операція зупиняється через запуск нової, для повідомлення про зміну її стану використовуються методи зворотнього виклику життєвого циклу операції. Існує декілька таких методів, які може приймати операція після зміни свого стану – створення, зупинка, відновлення, знищення операції; Кожен зворотній виклик має можливість виконати певну дію, яка підходить для відповідної зміни стану. Наприклад, у випадку зупинки операції необхідно звільнити будь-які великі об'єкти (підключення до мережі, бази даних). При відновленні операції - повторно отримати ресурси та відновити виконання перерваних дій. Такі зміни стану є частиною життєвого циклу операцій.

Запуск операції. Для запуску нової операції достатньо викликати метод startActivity(), передавши до нього об’єкт Intent (з англ. Намір), який описує її. Тут вказується операція для запуску або її тип (тоді система обирає операцію, що найбільш підходить). Метод може містити та використовувати невелику порцію даних. Визначати необхідну операцію можна за допомогою імені класу. Приклад, запуск з однієї операції іншої з ім’ям SignInActivity (Лістинг 15).

Лістинг 15 – Запуск однієї операції із тіла іншої

```
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent); // почати операцію
```

У додатку іноді необхідно виконати певну дію (наприклад, відправити лист), користуючись даними з операції. Така дія може бути відсутня в додатку, тоді можна використати операцію з іншого додатку на пристрої. У такому випадку можна створити Intent, що описує необхідну дію, після чтого система виконує його запуск із іншого додатку. Наприклад, якщо користувачу необхідно відправити лист (Лістинг 16).

Лістинг 16 – Відправлення електронного листу на перелік адрес

```
Intent intent = new Intent(Intent.ACTION_SEND); //створення наміру
intent.putExtra(Intent.EXTRA_EMAIL, recipientArray);
startActivity(intent); //EXTRA_EMAIL - масив адрес для відправки
```

Коли користувач натискає кнопку «Open», викликається метод openGallery() класу OpenGallery. Де використовується клас QAndroidJniObject, що надає API для виклику Java-коду з C++.

Створюємо екземпляр класу QAndroidJniObject та вказуємо властивість операції, яку необхідно виконати. Створюємо ще один екземпляр, що буде наміром – Intent. За допомогою функції callObjectMethod виконується виклик методу об’єкту Java methodName, і повертає новий QAndroidJniObject для повернутого об'єкта Java (Лістинг 17).

Лістинг 17 – Синтаксис функції callObjectMethod

```
QAndroidJniObject QAndroidJniObject::callObjectMethod(const char
*methodName) const
```

У разі вдалого створення, робимо виклик методів setAction та setType, що встановлять для наміру створену властивість операції та тип об’єкта для якого потрібно виконати операцію, відповідно. Простір імен QtAndroid надає різні функції для розробки Android. Функція startActivity запускає операцію, задану наміром, і надає результат асинхронно через resultReceiver, якщо це не є null (Лістинг 18). Якщо resultReceiver буде null, буде викликатися метод startActivity() у androidActivity(). Інакше буде викликатися startActivityForResult().

Лістинг 18 - Синтаксис функції startActivity

```
void QtAndroid::startActivity(const QAndroidJniObject &intent, int
receiverRequestCode, QAndroidActivityResultReceiver *resultReceiver =
nullptr)
```

receiverRequestCode є кодом запиту, унікальним для resultReceiver, і повертається разом з результатом, що дозволяє використовувати один і той же приймач для більш ніж одного наміру. У користувальницькому класі OpenGallery описаний метод, що викликає операцію Java (ДОДАТОК А).

Після виклику QtAndroid::startActivity, результат операції буде отриманий у OpenGallery::handleActivityResult. Якщо операція відкриття

галереї пройшла успішно RESULT_OK, виконується виклик методу getImage із пакету Java. Позначення «[B» говорить про те, що результат повертається у байтах. Далі виконується приведення типу QAndroidJniObject до jbyteArray, а далі до QByteArray. QByteArray можна перетворювати на QImage і виводити за допомогою методів QQuickPaintedItem, або зберігати у пам’яті пристрою (ДОДАТОК А).

У Java файлі, в свою чергу, описана функція отримання зображення. Після того, як користувач натиснув на одне з зображень галереї, його вибір був встановлений в Intent. За допомогою ByteArrayOutputStream виконується зчитування байтів із даних отриманих в Intent. Масив байт повертається, в якості результату роботи функції [49].

3.3. Обмін даними клієнта із сервером

Обмін даними із сервером реалізовано на базі стандартних методів класу QTcpSocket, що надає можливість використання сокета TCP.

TCP (Transmission Control Protocol) - це надійний транспортний протокол, орієнтований на потік. Він особливо добре підходить для безперервної передачі даних. QTcpSocket є зручним підкласом QAbstractSocket, який дозволяє встановлювати з'єднання TCP і передавати потоки даних.

У конструкторі користувальницького класу створено сокет - екземпляр класу QTCPSocket (Лістинг 19).

Лістинг 19 – Створення сокету

```
QTCPSocket* socket = new QTcpSocket(this);
```

Сигнали, що випускаються класом QTcpSocket поєднано із методами користувальницького класу, як наведено у Лістингу 20:

Лістинг 20 – Поєднання сигналу сокета із користувальницьким методом

```
connect(socket, &QTcpSocket::connected, this, &SocketTest::connected);
```

Сигнали: connected - сигнал випускається після виклику connectToHost() і успішного встановлення з'єднання. disconnected - сигнал випускається, коли сокет роз’єднався. bytesWritten - сигнал випускається кожного разу, коли дані записуються в поточний канал запису пристрою. readyRead - сигнал випускається кожного разу, коли доступні нові дані для читання з поточного каналу зчитування пристрою (ДОДАТОК А). У тілі методу, пов’язаного із цим сигналом, спочатку із сокету зчитується розмір отримуваної інформації, а потім і сама інформація частинами. Коли розмір отриманої інформації співпаде із очікуваним - дані будуть збережені та виведені на екран телефону.

Для передачі даних до серверу використовується функція write(), що записує дані з рядка з 8-бітових символів (що закінчуються нульовою послідовністю) на пристрій і повертає кількість фактично записаних байтів або -1, якщо сталася помилка (Лістинг 21).

Лістинг 21 – Запис даних до сокету.

```
socket->write(image);
```

Або ж дані можна передавати до сокету за допомогою класу QDataStream, що забезпечує серіалізацію двійкових даних до пристрою QIODevice. Потік даних - це двійковий потік кодованої інформації, який на 100% незалежний від операційної системи хоста комп'ютера, процесора або порядку байтів. Пристрій QIODeviceявляє собою носій, з якого можна зчитувати дані і записувати дані.

У якості QIODevice виступає створений сокет. При введенні або виведенні складних типів дуже важливо переконатися, що одна і та ж версія потоку використовується для читання і запису (Лістинг 22).

Лістинг 22 – Створення потоку введення даних до сокету

```
QDataStream out(m_socket);
out.setVersion(QDataStream::Qt_5_11); // встановлення версії
out << static_cast<qreal>(m_p); // відправка значення до сокету
```

3.4. Опис серверу

3.4.1. Обмін даними сервера з клієнтом

Обмін даними сервера з клієнтом здійснюється за допомогою фреймворку socketserver, що підключено до файлу серверам - `import socketserver`. Для обміну передбачено асинхронний сервер на основі класу ThreadedTCPRequestHandler. Створено клас обробника запитів, використавши клас BaseRequestHandler і визначити його метод handle(), що оброблятиме вхідні запити (Лістинг 23).

Лістинг 23 – Опис методу, що викликається у разі підключення клієнта

```
class  ThreadedTCPRequestHandler(socketserver.BaseRequestHandler):
  def handle(self):
     try:
       print("\nNew connection...")
# задання розміру даних, що приймаються за один раз
... # отримання даних
... # отримання даних, обробка даних
# обчислення глибини
...
# відправлення розміру
...
# відправлення даних
...
print("  Sent: ", len(data))
```

Необхідно встановити дозвіл на повторне використання адрес. Та встановити daemon_threads у значення True, щоб не очікувати завершення роботи потоків (Лістинг 24).

Лістинг 24 – Встановлення параметрів серверу.

```
daemon_threads = True allow_reuse_address = True
```

Метод __init__грає роль конструктору класу (Лістинг 25).

Лістинг 25 – Опис методу __init__серверу

```
def __init__(self, server_address, RequestHandlerClass)
```

Для роботи сервера визначено Хост та Порт на якому він буде розташований. Екземпляр класу server запущено із використанням конструкції with, що дозволяє автоматично закривати сервер після завершення його роботи, тобто можливе підключення багатьох користувачів навіть після перезапуску програми. За допомогою бібліотеки threading для кожного нового користувача створюється окремий потік. Використано метод serve_forever() об'єкта сервера для обробки одного або декількох запитів. Щоб закрити сокет (якщо не було використано оператор with) необхідно викликати server_close(). Сервер завершує роботу після натиску клавіши Enter (Лістинг 26).

Лістинг 26 – Опис роботи серверу.

```
if __name___== "__main___":
HOST, PORT = "", 7071
print('Server start work!')
server = ThreadedTCPServer((адреса, користувальницький клас)
with server: # забезпечує коректне припинення роботи
...
```

3.4.2. Обробка даних сервером

Дані на сервер надходять у вигляді масивів байтів, отже для використання стандартних методів необхідно перетворити їх на зображення. Для цього використані PIL - Python бібліотека для обробки зображень, та бібліотека io, що містить основні інструменти для роботи з потоками (Лістинг 27).

Лістинг 27 – Інтерпретація масиву отриманих байтів у якості зображення

```
a = pil.open(io.BytesIO(data))
```

Спершу із фонового зображення розраховується карта глибини - метод calc_depth(stable, data). Stable - версія моделі нейронної мережі, data - фонове

зображення. Створено клас-обгортку ModelInterface для використанні моделі нейронної мережі, для того щоб модель або тип мережі могли бути змінені без суттєвих змін у коді серверу. Метод predict отримує на вхід фонове зображення та його розміри і повертає карту глибини, що змінює свій розмір на очікуваний, перетворюється на масив байт та повертається клієнту - мобільному додатку (Лістинг 28).

Лістинг 28 – Функція обчислення карти глибини.

```
def calc_depth(stable, data):
...
# використання моделі нейронної мережі для отримання карти глибини
...
# зміна розміру карти глибини
...
# інтерпретація результатів у якості байт та повернення масиву байт і масиву чисел карти
```

Для того, щоб края спроекційованого зображення не були «рваними», застосуємо прийом згладження. Спочатку виконується розмиття зображення, а потім значення пікселей вище певної межі стають білими, нижче - чорними. Таким чином, отримуються плавні контури фону зображення (Лістинг 29).

Лістинг 29 – Отримання плавних контурів об’єктів зображення.

```
gf = gaussian_filter(disp, sigma=4) # обробка фільтром
# перетворення на двуколірне зображення із пороговим значенням
```

Потім контрастність зображення збільшується (Лістинг 30).

Лістинг 30 – Підвищення контрасту зображення.

```
def normilize_image(I):
# опис алгоритму підвищення контрастності
...
# мінімальне значення кольору (найближче до чорного)
..
# максимальне значення кольору (найближче до білого)
...
return arr # повернення результату - масиву
```

Метод calc_projected_image приймає на вхід карту глибини, зображення для проеціювання, розміри фонового зображення, розміри області проеціювання та дані акселерометру. Масиви даних перетворюються у дані типів VTK, за допомогою встановлення відповідних параметрів.

У тілі даного методу, за допомогою методів VTK, будується модель, відповідно до аргументів методу та виконується сам рендерінг. Після рендерінгу, вікно збережується у якості масиву NumPy. Отже, отримано деформоване зображення відповідно до форми об’єктів фонового.

Тепер необхідно накласти один рисунок на інший з урахуванням альфа-каналу. Це реалізовано у функції transparentOverlay, що у якості параметрів приймає карту глибини, фонове та деформоване зображення, координати виділеної області. У алгоритмі передбачено скриття пікселів (див. Додаток Д). Результат перетворюється у масив байтів та відправляєьться клієнту-користувачу (Лістинг 31).

Лістинг 31 – Накладання фонового і зпроекційованого зображення, приведення результату до масиву байт.

```
result = transparentOverlay(thr, firstIm, secondIm, (coord1,coord2,
0)
success, encoded = cv2.imencode('.jpg', result) # кодування
byte_arr = encoded.tobytes()m# приведення до масиву байт
```

3.5. Опис мобільного додатку

3.5.1. Інтерфейс програми

Інтерфейс програми складається із трьох вкладок, між якими користувач може вільно переміщуватись, натискаючи на їх назви: «Camera», «Home», «Processed» (рис. 3.2).

Кожна вкладка містить елементи управління, що дозволяють користувачу взаємодіяти із додатком.

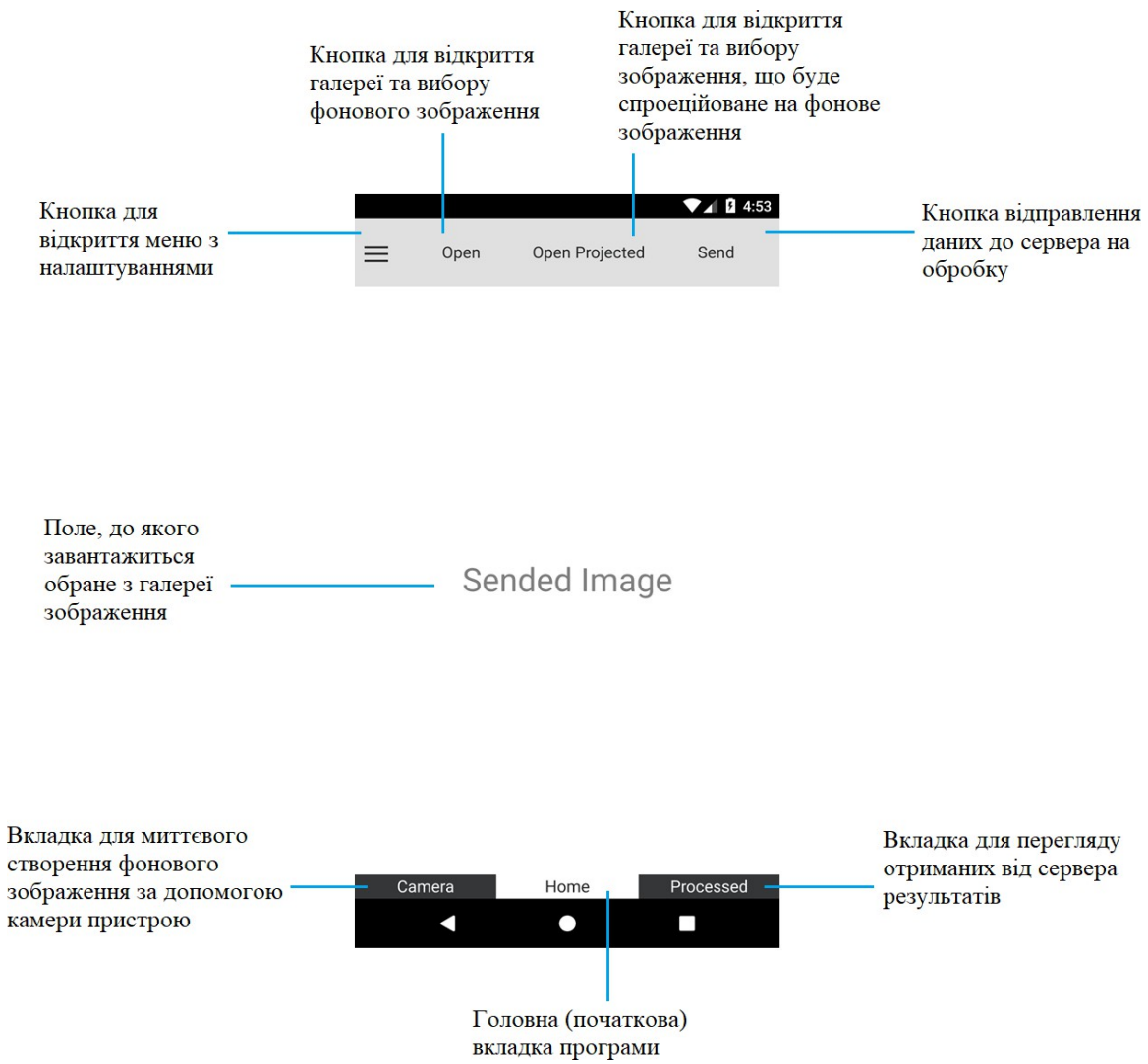

Рис. 3.2. Інтерфейс головного вікна додатку

3.5.2. Вибір даних користувача для обробки

У додатку користувач працює з двома зображеннями:

Перше - фонове, для якого й обчислюється карта відстаней. З урахуванням отриманих даних, виконується реконструкція видимої частини об’єктів фону.

Друге зображення проекціюється на реконструйовані об’єкти. Тобто воно приймає форму, ніби обгортає, рельєф сцени. В якості такого зображення може виступати будь-який плакат, картина, візарунок, постер, логотип. Або це може бути зображення тексту, назви чи слогану компанії.

3.5.3. Вибір фонового зображення

Для вибору фонового зображення можна відвідати вкладку «Камера», на якій, за допомогою камери мобільного телефону, зробити фото інтер’єру, будівлі, тощо (рис. 3.3). При натисканні кнопки «Discard» створена фотографія не зберігається, і користувачу надається можливість зробити новий знімок. Якщо вибрати кнопку «Save», програма автоматично перейде до вкладки «Home», а у відповідне поле по центру буде завантажений щойно створений кадр, отже він буде використаний у якості фону.

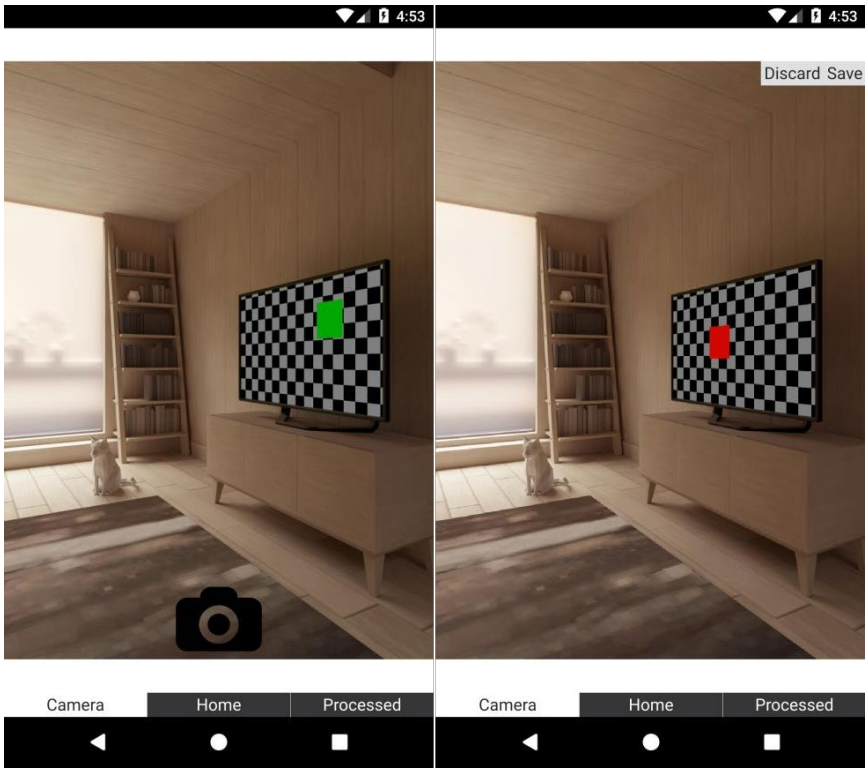

Рис.3.3. Інтерфейс вкладки «Camera»

Інший спосіб обрати фото сцени - на вкладці «Home» натиснути кнопку «Open». Після цього на екрані буде відображена галерея телефону, де із зображень, що зберігаються в пам’яті пристрою користувач зможе обрати необхідне.

3.5.4. Вибір зображення, що буде спроекційовано на фонове

Для того, щоб обрати зображення, яке буде проеціюватися на об’єкти фону, необхідно натиснути кнопку «Open Projected», та у галереї пристрою обрати бажане фото чи рисунок.

Для подальшої роботи потрібно обрати прямокутну область фону, на яку буде виконуватись проекція. Зазвичай, обирається частина деякого об’єкту сцени (поверхня будівлі, стінка квартири, двірцята машини, тощо). Виділення відбувається при натисненні у будь-якій точці фонового зображення та переміщенні пальцю/стилусу по поверхні екрану. Треба зазначити, що єдиний натиск екрану (тап) знімає виділення та відправлення даних на обробку стає неможливим (рис. 3.4).

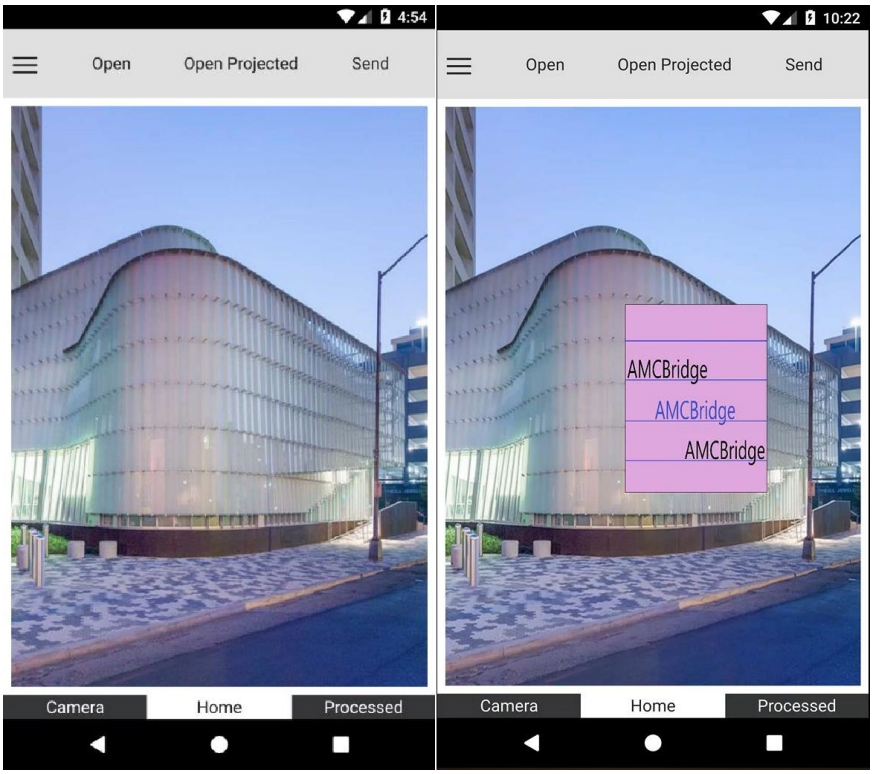

Рис. 3.4. Завантаження зображень до головного вікна додатку

3.5.5. Зміна налаштувань у меню

У розробленому додатку передбачена можливість зміни стандартних налаштувань.

Потрібно зазначити, що зйомка може бути виконана з різних положень, тобто камера може розташовуватись вище чи нижче об’єкту, на який буде виконано проекцію. В залежності від цього напрям проекціювання буде різним (при зйомці на рівні деформація відбуватись не буде). Якщо камера нижче предмету, то горизонтальні лінії на зображенні деформуються догори, відповідно якщо камера вище - донизу.

У момент, коли користувач робить фото камерою телефону, додаток фіксує дані акселерометру пристрою (акселерометр - прилад, який дозволяє смартфону визначати своє положення в просторі), залежно від яких розраховується проекція і напрям деформації.

Якщо користувач обирає зображення з галереї, дані про положення камери - відсутні. У такому випадку передбачена можливість ручного встановлення у меню («≡») налаштувань деформації при проекціюванні (рис. 3.5).

У розділі меню «Direction» можна обрати одну з двох кнопок. Вибір кнопки з позначенням «˄» означає, що камера знаходиться нижче об’єкта зйомки, кнопки «˅» - вище.

У розділі меню «Deformation» за допомогою повзунка можна обрати силу деформації від 0 (деформація відсутня) до 90 (максимально можлива деформація).

Налаштування можна залишити за замовчуванням - камера знаходиться нижче об’єкта (кнопка «˄») зйомки та середній рівень деформації (45).

Натиснувши на кнопку «Information» можна побачити опис розділів налаштування.

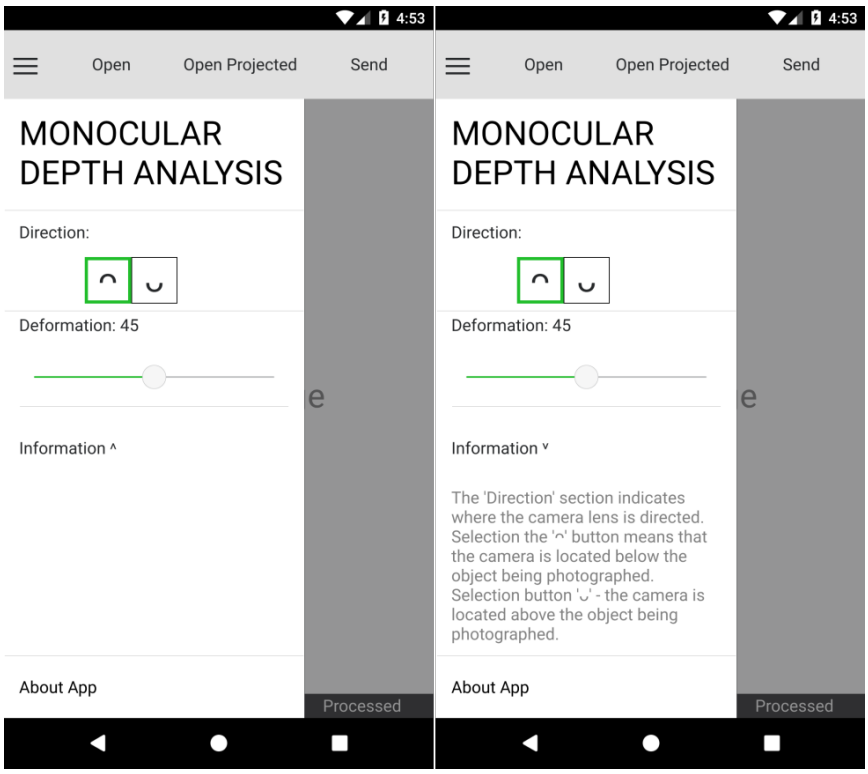

Рис. 3.5. Зовнішній вигляд меню

3.5.6. Отримання інформації про додаток

Щоб отримати інформацію про додаток (версію, розробників) необхідно на вкладці «Home» у лівому верхньому куті натиснути на іконку «≡». В результаті чого буде відкрито меню, у нижній області якого розташована кнопка «About App». При натисканні на дану кнопку, на екрані з’явиться вікно з необхідною інформацією. Вікно можна закрити натиснувши на затінену область поза цим вікном.

3.5.7. Введення IP-адреси серверу

Дане програмне забезпечення, на даний момент, не опубліковано у мережі Інтернет. Тому окремої машини з обчислювальним сервером не виділено - у різний час сервер може мати різні адреси. Через це у додатку передбачена можливість зміни адреси серверу. Діалогове вікно із полем для

вводу IP-адреси з’являється на екрані після першого натиснення кнопки «Send» або якщо сервер з такою адресою не існує (рис. 3.6).

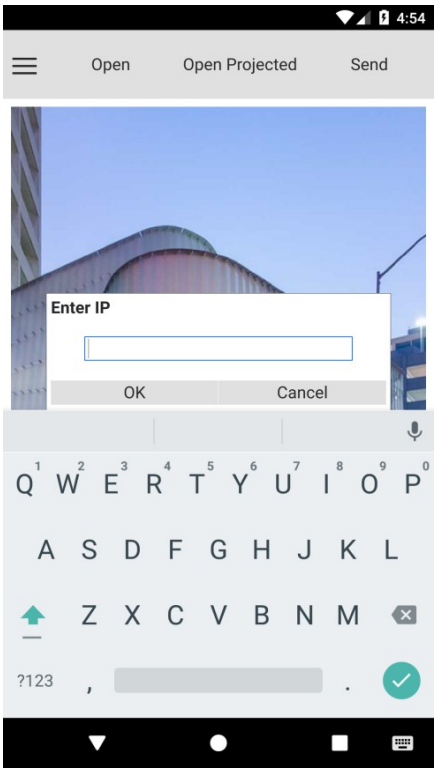

Рис. 3.6. Зовнішній вигляд вікна для введення адреси сервера

3.5.8. Відправлення запиту та отримання результатів

Після вибору двох зображень та області проекціювання необхідно натиснути кнопку «Send» на вкладці «Home». Якщо одне із зображень не обрано або не відмічена область проекції додаток відобразить повідомлення про помилку. У випадку, коли наявні усі дані, програма автоматично перейде на вкладку «Processed», де спеціальний елемент управління вказує на виконання обробки. Користувач також має можливість відмінити відправку даних (рис. 3.7).

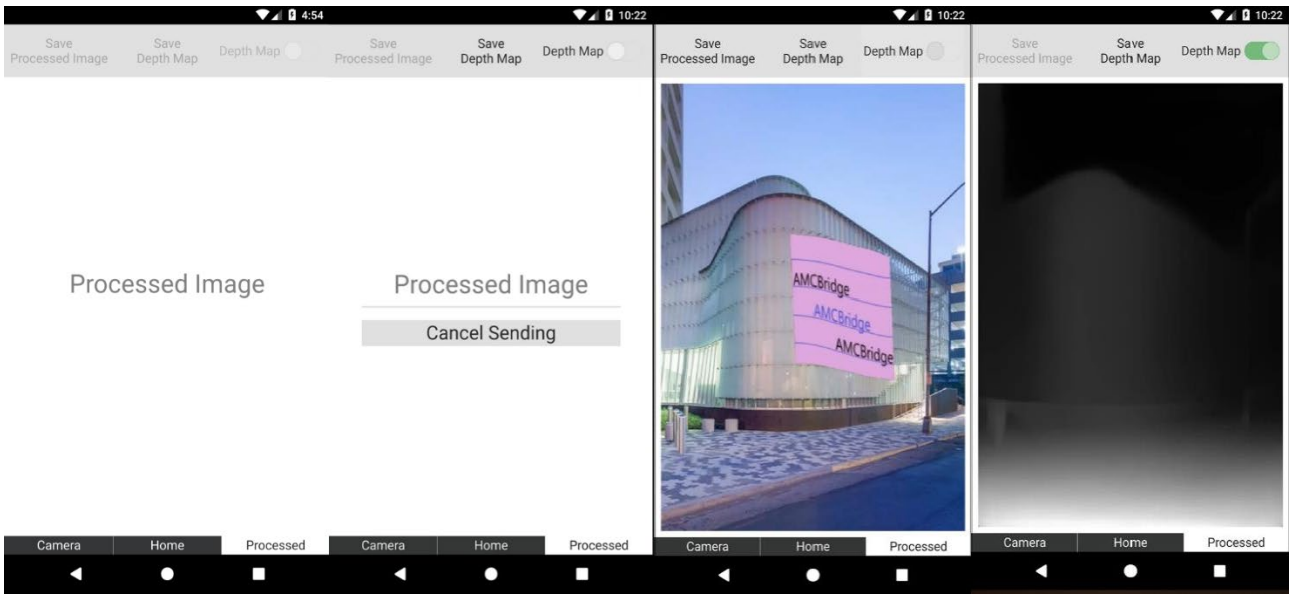

Рис. 3.7. Вікно очікування і результатів обробки зображення

Спочатку додаток отримує карту глибини фонового зображення та виводить її замість поля очікування. При цьому у правому верхньому куті екрану перемикач «Depth Map» встановлюється у ввімкнений стан. Користувач має змогу перемикати його на свій розсуд. При вимкненні, карта відстаней пропаде і, на її місці, знов з’явиться екран очікування результатів проеціювання. Коли результати від сервера будуть отримані, вони також будуть відображені у центрі екрану. У лівому верхньому куті є дві кнопки «Save Processed Image» та «Save Depth Map», які зберігають у пам’яті пристрою відповідні зображення у форматі JPEG. При натисканні даних елементів керування з’явиться вікно з повідомленням про успішність (або неуспішність) операції (рис. 3.8).

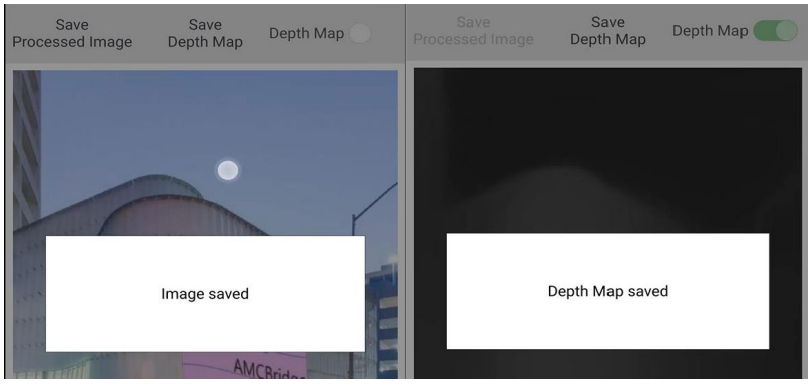

Рис. 3.7. Збереження даних

3.6. Програмно-апаратні вимоги

Вимоги для роботи мобільного додатку: смартфон із основною камерою, ОС Android 8.0 та пізніші версії, API Level > 26.

Запуск серверу повинен відбуватися на пристрої із наступними параметрами: стаціонарний комп'ютер чи ноутбук, ОС Microsoft Windows, процесор Intel(R) Core(TM) i5-8250U (1.6 ГГц, 1.8 ГГц), кількість ядер 4, обсяг оперативної пам'яті 16.0 Гб, відеокарти NVIDIA GeForce 930MX, Intel(R) UHD Graphics 620.

3.7. Висновки до третього розділу

У третьому розділі кваліфікаційної роботи наведені результати дослідження мобільного додатку під ОС Android, для визначення карти відстаней до об'єктів зображення. При розробці проекту були використані технології нейронних мереж.

Розроблене програмне забезпечення складається з двох частин - серверної і клієнтської. У даному розділі розкритий основний функціонал мобільного додатку; наведені приклади опису елементів керування; зазначені етапи обробки зображення сервером та принцип його з'єднання із клієнтом; наведені приклади результатів роботи програми.

Результатом розробки є програмне забезпечення, що дає змогу користувачу, на основі персональних даних (фотографій) отримувати карту глибини, проекціювати довільне зображення на вихідне і зберігати результати обробки на мобільному телефоні.

Розроблений інтерфейс зрозумілий, простий та лаконічний, створений таким чином, щоб не відволікати увагу користувача від роботи.

Для розуміння принципу роботи додатку створено керівництво користувача, крім цього програма оснащена підказками.

ВИСНОВКИ

Сьогодні, спостерігається бурхливий розвиток технологій мобільних пристроїв, і поряд з їх доступністю відзначається можливість виконання на них досить складних операцій. Наприклад, отримання даних про глибину сцени, що можуть бути корисними при вирішенні задач пов’язаних із тривимірним моделюванням. Зокрема, виявлення форми та відстані до об’єктів на зображенні. Тому дослідження з даної тематики є актуальними.

Під час виконання кваліфікаційної роботи, була вивчена і систематизована теорія з обраної тематики. Проведений порівняльний аналіз існуючих технологій для обчислення карти глибини зображення довільної сцени. В результаті чого були виявлені їх основні переваги та недоліки.

Мобільних додатків, які дають змогу, на основі даних про глибину сцени, виконувати проекціювання зображень на об’єкти фото, не було виявлено, що підкреслило актуальність розробки відповідного програмного забезпечення.

Результатом виконання кваліфікаційної роботи є розробка та дослідження мобільного додатку під ОС Android, для визначення карти відстаней до об'єктів зображення. При розробці проекту були використані технології нейронних мереж.

Розроблене програмне забезпечення складається з двох частин - серверної і клієнтської, та надає можливість виконувати подальшу обробку вихідного зображення, а саме проекціювати довільне зображення на вихідне, відповідно до отриманих даних про глибину сцени.

Серверна частина реалізована на мові програмування Python з використанням бібліотек socketserver, OpenCV, PyTorch, VTK.

Клієнтська частина – це мобільний додаток, код якого написаний з використанням фреймворку Qt та декларативної мови програмування QML.

Програмна система надає змогу користувачу, на основі персональних даних, швидко здійснювати складні обчислення, а також отримувати та

зберігати результати обробки, одразу на персональному девайсі - мобільному телефоні.

Дизайн програми, зрозумілий, простий та лаконічний, створений таким чином, щоб не відволікати увагу користувача від роботи. Також, для більшої зручності, програма оснащена підказками.

Мета і завдання, поставлені у роботі, виконані у повному обсязі. Додаток може бути використаний:

- фірмами, що займаються рекламою, для швидкого і наочного представлення зовнішнього вигляду логотипу чи назви компанії на тому чи іншому будинку;
- дизайнерами для економії часу на створення рекламних макетів або портфоліо;
- у розважальній сфері, простими користувачами для «примірки» різних картин або постерів на стінах своєї квартири;
- студентами для вивчення розділів 3D-моделювання у якості факультативного матеріалу для наочного розглядання теми стереозору та підвищення зацікавленості у даній області.

В ході виконання завдань проекту, були систематизовані і закріплені знання з програмування та обробки зображень, а також здобуті, вдосконалені та розширені практичні навички і уміння в роботі за спеціальністю 122 Комп’ютерні науки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Depth map [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Depth_map

2. How Depth Sensors Works In 5 Minutes [Електронний ресурс] – Режим доступу до ресурсу:https://jahya.net/blog/how-depth-sensor-works-in-5- minutes/

3. Методичний посібник до вивчення розділу «Методи обробки зображень та комп'ютерний зір», РВВ ДНУ ім. Олеся Гончара, Дніпропетровськ, ЛІРА, 2016. с. 127-135

4. Визильтер Ю.В. Обработка и анализ изображений в задачах машинного зрения: курс лекций и практических занятий / Ю.В.Визильтер, С.Ю. Желтов, А.В. Бондаренко, М.В. Ососков, А.В. Моржин. – М.: Физматкнига, 2010. – 672 с.

5. Форсайт Д. А. Компьютерное зрение. Современный подход / Д.А. Форсайт, Ж. Понс. – М. : Изд. «Вильямс», 2004. – 928 с.

6. Хорн Б.К.П. Зрение роботов / Б.К.П. Хорн, пер. с англ. – М. : Мир, 1989. – 487 с.

7. Прэтт У. Цифровая обработка изображений / У. Прэтт; пер. с англ.; под ред. Д.С. Лебедева. – М. : Мир, 1982 р.

8. MegaDepth [Електронний ресурс] – Режим доступу до ресурсу: http://www.cs.cornell.edu/projects/megadepth/paper.pdf

9. Інструкція по установці PyTorch [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/ru-ru/windows/ai/windows-ml/tutorials/pytorch-installation

10. Adam [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1412.6980

11. Основы стереозрения [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/130300/

12. Грицик В. В. Методи проектування систем нейрокомп'ютерного обладнання для мобільних робототехнічних систем / Грицик В. В., Цмоць І. Г.,

Теслюк В. М. // Доповіді НАН України. – Київ, 2013. – Число 1. – С. 30- 36. Hartley, R. I. and Zisserman, A. “Multiple View Geometry in Computer Vision” – NY: Cambridge University Press, 2000.

13. Gary Bradski, Adrian Kaehler “Learning OpenCV”, O’Reilly Media, 2008.

14. Евклидова геометрия [Електронний ресурс] / Вікіпедія. Вільна енциклопедія - Режим доступу до ресурсу: Режим доступу до ресурсу: https://uk.wikipedia.org/Евклідова_геометрія

15. Евклідів простір [Електронний ресурс]/ Вікіпедія. Вільна енциклопедія - Режим доступу до ресурсу: Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Евклідів_простір

16. Stereo Vision [Електронний ресурс] – Режим доступу до ресурсу: http://vision.middlebury.edu/stereo

17. Паралакс [Електронний ресурс] / Вікіпедія. Вільна енциклопедія - Режим доступу до ресурсу: Режим доступу до ресурсу:https://uk.wikipedia.org/wiki/Паралакс

18. Семантична сегментація зображень з використанням згорткових нейронних мереж [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2019/paper/viewFile/6464/5365

19. MegaDepth [Електронний ресурс] – Режим доступу до ресурсу: http://www.cs.cornell.edu/projects/megadepth/

20. Introduction to OpenCV-Python Tutorials [Електронний ресурс] – Режим доступу до ресурсу: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_intro/py_intro.html#intro

21. OpenCV [Електронний ресурс] – Режим доступу до ресурсу: http://opencv.org

23. OpenCV Python Tutorial: Computer Vision With OpenCV In Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.edureka.co/blog/python-opencv-tutorial/

24. OpenCV Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://docs.opencv.org/3.2.0/d1/dfb/intro.html

25. Introduction to OpenCV-Python Tutorials [Електронний ресурс] – Режим доступу до ресурсу: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_intro/py_intro.html#intro

26. Automatic differentiation [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Automatic_differentiation)

27. Torch [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Torch

28. PyTorch [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/PyTorch

29. socketserver — A framework for network servers [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/socketserver.html

30. VTK [Електронний ресурс] – Режим доступу до ресурсу: https://vtk.org/

31. Visualization Toolkit [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Visualization_Toolkit

32. Qt [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Qt

33. QML [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/QML

34. Qt Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://doc.qt.io/qt-5/

35. Activities [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/components/activities.html

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

Підключення сигналу з QML документу до слоту з C++ документу

```
// MyItem.qml
import
QtQuick 2.0
Item { // опис елементу керування
    id: item
    width: 100; height: 100 // задання ширини та висоти
    signal qmlSignal(string msg) // опис сигналу із аргументом
    MouseArea { // областя натискання
        anchors.fill: parent
        onClicked: item.qmlSignal("Hello from QML")
    }
}
//myclass.cpp
class MyClass : public QObject // створення класу
{
    Q_OBJECT // макрос, щовказує на належність метаоб'єктній
    системі
public slots:
    void cppSlot(const QString &msg) { // опис слоту
        qDebug() << "Called the C++ slot with message:" << msg;
    }
};
int main(int argc, char *argv[]) {
    QGuiApplication app(argc, argv); //створення екземпляру
    додатку
    //завантаження головного файлу QML
    QQuickView view(QUrl::fromLocalFile("MyItem.qml"));
    QObject *item = view.rootObject(); //отримання кореневого
    елем.
    MyClass myClass;
    QObject::connect(item,    SIGNAL(qmlSignal(QString)),
                              &myClass, SLOT(cppSlot(QString)));
//з'єднати сигнал з MyItem.qml та слот myClass-
    view.show(); // вивести на екран
    return app.exec(); //запустити додаток
}
```

Фрагмент програмного коду файлу з розширенням .pro.

```
    // Список модулів Qt, що використовуються в проекті. QT включає
  модулі ядра і gui за замовчуванням, тому опис додає вказані модулі
  до цього списку.
    QT += androidextras quick network multimedia sensors
    // qmake дозволяє підтримку C++11 за допомогою опції CONFIG
    CONFIG += c++11
    //  Дає вказівку компілятору виводити попередження, якщо
  використовується функція Qt, позначена застарілою.
    DEFINES += QT_DEPRECATED_WARNINGS
    // Список файлів вихідного коду, які будуть використовуватися
  при створенні проекту.
    SOURCES += main.cpp \
        socket.cpp \
        opengallery.cpp
    // Список файлів ресурсів (.qrc), які будуть включені до проекту.
    RESOURCES += qml.qrc
    // Значення за замовчуванням для встановлення на пристрій.
    qnx: target.path = /tmp/$${TARGET}/bin
    else: unix:!android: target.path = /opt/$${TARGET}/bin
    !isEmpty(target.path): INSTALLS += target
    // Список імен файлів заголовків (.h), які використовуються при
  створенні проекту.
    HEADERS += socket.h \
        opengallery.h
    // Визначає список файлів, які потрібно включити до остаточного
  проекту. Ця функція підтримується специфікаціями UnixMake
    DISTFILES += \
      android/ImageFromActivityResult/ImageFromActivityResult.java
    // Шлях до каталогу, що містить ресурси проекту (зображення,
  іконки, тощо). PWD - повний шлях до каталогу, що містить поточний
  файл.
ANDROID_PACKAGE_SOURCE_DIR = $$PWD/android
```

Опис слоту в C++ документі, що викликається у відповідь до надходження сигналу з QML документа

```
    class Socket : public QObject
    {
        Q_OBJECT // макрос
    public:
        explicit SocketTest(QObject *parent =
    nullptr); public slots:
```

```
    void setP(int p); // опис слоту
private:
    int m_p; // змінна, що буде встановлена у тілі слоту
};
```

Визначення слоту у вихідному .cpp документі

```
void Socket::setP(int p) // опис слота
{
    m_p = p; // встановлення змінної у значення аргументу
}
```

Реєстрація користувальницького класу у якості типу

```
...
#include “MyType.h”
int main(int argc, char *argv[])
{ ... // ф-ція реєстрації типу MyType
qmlRegisterType<MyType>(“MyType”, 1, 0, “
MyType”);
...
}
```

Імпорт і використання користувальницького типу у QML документі

```
import QtQuick 2.11
import QtQuick.Controls 2.4
import MyType 1.0 // імпорт користувальницького типу
Page { // елемент - сторінка
...
MyType { // елемент користувальницького типу
       id: img // ідентифікатор
       width: parent.widthProj // встановлення ширини
       height: parent.heightProj // встановлення висоти
       ...
    }
    ...
}
```

Метод класу OpenGallery для відкриття галереї

```
void OpenGallery::openGallery() // метод відкриття галереї
{
```

```
            QAndroidJniObject ACTION_PICK =
  QAndroidJniObject::fromString("android.intent.action.GET_CONTENT");
        QAndroidJniObject
            intent("android/content/Intent"); if
            (ACTION_PICK.isValid() && intent.isValid())
            { // якщо намір коректний встановлюємо дію та тип значення, що
  повертається - зображення
intent.callObjectMethod("setAction",
"(Ljava/lang/String;)Landroid/content/Intent;",
ACTION_PICK.object<jstring>());
                intent.callObjectMethod("setType",
  "(Ljava/lang/String;)Landroid/content/Intent;",
  QAndroidJniObject::fromString("image/*").object<jstring>());
                QtAndroid::startActivity(intent.object<jobject>(),
  101, this); // розпочати операцію
            }
        }
```

Функція handleActivityResult користувальницького класу MyType

```
  void MyType::handleActivityResult(int receiverRequestCode, int
  resultCode, const QAndroidJniObject & data) {
  // метод, що виконується у відповідь на отримання результатів
        jint RESULT_OK =
  QAndroidJniObject::getStaticField<jint>("android/app/Activity",
  "RESULT_OK"); // якщо операція успішна
        ...
  // отримаємо обране зображення у якості байт, виклик Java-функції
  getImage пакету та класу ImageFromActivityResult
            ...
        // перетворення типу даних QandroidJniObject до jbyteArray
            ...
        //якщо отримана довжина масиву байт > 0
        ...
        // перетворення типу даних до QByteArray
                        ...
}
                }
            }
        }
```

Опис Java-функції, що отримує зображення із галереї

```
        public class ImageFromActivityResult {
```

```
            // ф-ція отриманна даних зображення
      public static byte[] getImage(...) throws IOException {
       // отримання даних із наміру
       InputStream is =
  activity.getContentResolver().openInputStream(intent.getData());
            // створення буферу, у який будуть записані дані
      ...
            // запис даних до буферу
      ...
      // оновити буфер
      ...
       // повернути масив байт із даними зображення
          }

      }
```

Опис методу отримання даних із сокету

```
      void SocketTest::readyRead()
      {
      // метод, що викликається при надходженні даних до слоту
      ...
      // створення потоку вводу із із сокету
...

      // зазначення версії потоку
          // якщо це перші отримані дані
         // якщо дані не заданого розміру, завершуємо роботу
      // встановлюємо очікуваний розмір
      // зчитування частинами
      // дані отримані
      // обробка отриманих даних
      }
      }
```

Перетворення масива NumPy до типу даних VTK

```
      imageImport = vtk.vtkImageImport() # встановлення типу даних VTK
      ...
      imageImport.SetNumberOfScalarComponents(4) # встановлення
  кількості каналів зображення
      imageImport.SetImportVoidPointer(data) # встановлення вхідних
      даних
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ «ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Наукового керівника Спірінцева В'ячеслава Васильовича, к.т.н., доц. каф. ПЗКС
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

на магістерську роботу

студента Ковальова Філіпа Вячеславовича
(прізвище, ім'я, по батькові)

курсу II групи 122м-22-1
спеціальності 122 Комп'ютерні науки
освітньої програми Комп'ютерні науки
на тему Розробка та дослідження мобільного додатку для визначення карти відстаней до об'єктів зображення з використанням нейронних мереж

Актуальність теми. Актуальним завданням, на сьогоднішній день, є розробка зорових систем з відповідним ПЗ, що представляють інформацію про сцену в тривимірному вигляді. Дане завдання вирішується за допомогою 3D-моделювання - процесу створення математичного подання тривимірного об'єкту. Для створення тривимірних моделей об'єктів існує безліч підходів в комп'ютерному зорі. При цьому, незалежно від підходу, зоровій системі заздалегідь невідомі тип, кількість і розмір об'єктів сцени. Тобто машинна система, подібно зоровій системі людини, повинна надавати інформацію про спостережуваний простір незалежно від його змісту. Важливим етапом в побудові тривимірної моделі є отримання карти глибини. У тривимірній комп'ютерної графіці карта глибини - це канал зображення або зображення, яке

містить інформацію, що стосується відстані поверхонь об'єктів сцени від точки огляду. Кожен піксель містить інформацію про віддаленість точки поверхні сцени від камери. Тобто оцінка глибини спрямована на отримання уявлення про просторову структуру видимої сцени. На даному етапі, ми спостерігаємо бурхливий розвиток технологій мобільних пристроїв. Поряд з поширеністю і доступністю сучасних смартфонів, варто зазначити можливість виконання на них досить складних операцій. Мобільних додатків, які надають змогу, на основі даних про глибину сцени, виконувати проекціювання зображень на об’єкти фото, не було виявлено, що підкреслило актуальність розробки відповідного програмного забезпечення.

Мета досліджень Полягає в розробці та дослідженні мобільного додатку під ОС Android, для визначення карти відстаней до об'єктів зображення з використанням технологій нейронних мереж, що надає можливість виконувати подальшу обробку вихідного зображення відповідно до отриманих даних про глибину сцени.

Коротка характеристика розділів роботи Перший розділ роботи присвячений огляду проблеми та існуючих засобів отримання карти глибини, також здійснено порівняльний аналіз існуючих технологій за тематикою обчислення карти глибини зображення довільної сцени. У другому розділі кваліфікаційної роботи запропонована структура системи отримання карти відстаней до об'єктів зображення з використанням технологій нейронних мереж, що надає можливість виконувати подальшу обробку вихідного зображення, а саме проекціювання інших зображень на об’єкти, відповідно до отриманих даних про глибину сцени. Така система складається з двох частин: серверної і клієнтської. На підставі того, що обчислення карти відстаней - досить ресурсовитратна операція і потребує значних витрати часу, вирішено було виконувати обробку зображення на більш потужному сервері – стаціонарному комп’ютері. Це зменшить час очікування результатів. Мобільний додаток, в свою чергу, забезпечить зручний інтерфейс для введення даних користувача до системи. У третьому розділі наведені результати дослідження мобільного

додатку під ОС Android, для визначення карти відстаней до об'єктів зображення. Розкрито основний функціонал мобільного додатку, наведені приклади опису елементів керування, зазначені етапи обробки зображення сервером та принцип його з'єднання із клієнтом, наведені приклади результатів роботи програми.

Практичне значення роботи Розроблене програмне забезпечення надає змогу користувачу, на основі персональних даних, швидко здійснювати складні обчислення, а також виконувати подальшу обробку вихідного зображення (а саме: проекціювати довільне зображення на вихідне, відповідно до отриманих даних про глибину сцени) отримувати та зберігати результати обробки, одразу на персональному девайсі - мобільному телефоні. Сфери застосування розробленого ПЗ: рекламні фірми (для швидкого і наочного представлення зовнішнього вигляду логотипу чи назви компанії на тому чи іншому будинку), дизайнери (для економії часу на створення рекламних макетів або портфоліо), розважальна сфера (для «примірки» різних картин або постерів на стінах своєї квартири) і т.ін.

Зауваження та недоліки В роботі не в повній мірі розкрите питання використання технології нейронних мереж при розпізнавання образів та оцінці карти глибини. Хоча це ні в якій мірі не впливає на загальну схвальну оцінку роботи.

Висновки та оцінка Магістерська кваліфікаційна робота виконана самостійно у відповідності з завданням із дотриманням усіх вимог. Під час виконання кваліфікаційної роботи студент Ковальов Ф.В. проявив себе грамотним, кваліфікованим спеціалістом, здатним приймати складні технічні рішення. Вважаю, що магістерська кваліфікаційна робота заслуговує оцінку «добре», а Ковальов Ф.В. – присвоєння кваліфікації «магістра» з комп'ютерних наук.

Науковий керівник Спірінцев В.В., доцент, доцент кафедри ПЗКС
(прізвище, ім'я, по батькові, посада, місце роботи)

«___»_____20__ р. _____

ДОДАТОК В

РЕЦЕНЗІЯ

на кваліфікаційну роботу

студента Ковальова Філіпа Вячеславовича
(прізвище, ім’я, по батькові)

курсу ІІ групи 122м-22-1
кафедри програмного забезпечення комп’ютерних систем
спеціальності 122 Комп’ютерні науки
Тема роботи Розробка та дослідження мобільного додатку для визначення карти відстаней до об’єктів зображення з використанням нейронних мереж

Стисла характеристика розділів роботи Перший розділ роботи присвячений огляду проблеми та існуючих засобів отримання карти глибини, також здійснено порівняльний аналіз існуючих технологій за тематикою обчислення карти глибини зображення довільної сцени. У другому розділі кваліфікаційної роботи запропонована структура системи отримання карти відстаней до об'єктів зображення з використанням технологій нейронних мереж, що надає можливість виконувати подальшу обробку вихідного зображення, а саме проекціювання інших зображень на об’єкти, відповідно до отриманих даних про глибину сцени. У третьому розділі наведені результати дослідження мобільного додатку під ОС Android, для визначення карти відстаней до об'єктів зображення. Розкрито основний функціонал мобільного додатку, наведені приклади опису елементів керування, зазначені етапи обробки зображення сервером та принцип його з’єднання із клієнтом, наведені приклади результатів роботи програми.

Пропозиції, внесені студентом, рівень їх наукового обґрунтування В даній кваліфікаційній роботі студентом надано декілька пропозицій щодо вирішення поставлених задач. Кожна з пропозицій була обґрунтована та підкріплена науковими даними.

Практичне значення роботи Розроблене програмне забезпечення надає змогу

користувачу, на основі персональних даних, швидко здійснювати складні обчислення, а також виконувати подальшу обробку вихідного зображення (а саме: проекціювати довільне зображення на вихідне, відповідно до отриманих даних про глибину сцени) отримувати та зберігати результати обробки, одразу на персональному девайсі - мобільному телефоні. Сфери застосування розробленого ПЗ: рекламні фірми (для швидкого і наочного представлення зовнішнього вигляду логотипу чи назви компанії на тому чи іншому будинку), дизайнери (для економії часу на створення рекламних макетів або портфоліо), розважальна сфера (для «примірки» різних картин або постерів на стінах своєї квартири) і т.ін.

Якість оформлення роботи Магістерська кваліфікаційна робота, яку подано на рецензію, виконана у повному обсязі у встановлений термін. Робота є добре структурованою та достатньо проілюстрованою. Викладена основна суть проблеми, що вирішується в ході виконання роботи, і шляхів її вирішення.

Недоліки в роботі Відсутність локалізації запропонованого додатку. Хоча це ні в якій мірі не впливає на загальну схвальну оцінку роботи.

Загальний висновок Магістерська кваліфікаційна робота виконана у відповідності з завданням із дотриманням всіх вимог.

Оцінка магістерської роботи Робота заслуговує оцінки «добре», а студент Ковальов Ф.В. – присвоєння кваліфікації «магістра» з комп'ютернихнаук.

Рецензент _____

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

«___»_____20__ р. _____

(підпис)

ДОДАТОК Г

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	**Опис**
Пояснювальні документи	
Диплом_Ковальов.doc	Пояснювальна записка роботи. Документ Word.
Диплом_Ковальов.pdf	Пояснювальна записка роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і відкомпільовану програму
Презентація	
Презентація_Ковальов.pptx	Презентація роботи