МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра інформаційних систем
(повна назва кафедри)

Магістерська робота

РОЗРОБКА ІНТЕРАКТИВНИХ НАВЧАЛЬНИХ МАТЕРІАЛІВ З ВИКОРИСТАННЯМ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

Виконала: студентка групи ПМІМ-22с
спеціальності
122 – Комп'ютерні науки
(шифр і назва спеціальності)

_____ Мичка О. О.
(підпис) (прізвище та ініціали)

Керівник _____ Бернакевич І.Є.
(підпис) (прізвище та ініціали)

Рецензент _____ Щербатий М.В
(підпис) (прізвище та ініціали)

Львів – 2022

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладна математика та інформатика

Кафедра Інформаційних систем

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Г. Шинкаренко
"05" 09 **2022 року**

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Мичка Ольга Олегівна
прізвище, ім'я, по батькові)

1. Тема роботи Розробка інтерактивних навчальних матеріалів з використанням доповненої реальності

керівник роботи Бернакевич Ірина Євстахіївна, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від "____" _____ 20__ року №_____

2. Дата подання студентом роботи 12 грудня 2022 року

3. Вихідні дані до роботи Технологія доповненої реальності, технологія реєстрації зображень, розпізнавання цілей, платформа Vuforia, мови програмування C# та Python, алгоритми SIFT, ORB, BRISK, бібліотека OpenCV, мобільна платформа Android

4. Зміст магістерської роботи (перелік питань, які потрібно розробити)
1. Аналіз технології доповненої реальності та алгоритму SIFT розпізнавання цілей;
2. Дослідження ефективності алгоритмів доповненої реальності;
3. Розробка інтерактивних навчальних матеріалів у вигляді тестового мобільного додатку з підтримкою AR і з можливістю візуалізації 3D моделей після розпізнавання зображення.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Код програмної реалізації

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 01 вересня 2022

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі для магістерської роботи	10.09.2022	
2	Ознайомлення із відповідною літературою та технічною документацією	20.09.2022	
3	Аналіз технології доповненої реальності та середовищ для розробки додатку	03.10.2022	
4	Розробка мобільного додатку із використанням технології доповненої реальності	10.10.2022	
5	Аналіз алгоритмів доповненої реальності	04.11.2022	
6	Створення програми на основі OpenCV для зіставлення зображень	20.11.2022	
7	Аналіз ефективності алгоритмів	25.11.2022	
8	Підготовка та оформлення презентації для доповіді	05.12.2022	
9	Підготовка матеріалів до друку	11.12.2022	

Студент _____ Мичка О. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Бернакевич І. Є.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Актуальність теми

Актуальність впровадження технології доповненої реальності в будь-яку сферу допомагає полегшити роботу, де важливе заохочення користувачів проявляти інтерес до викладеної інформації, тим самим покращуючи візуалізацію навчальних тез.

Доповнена реальність поєднує реальне середовище з комп'ютерним, щоб «доповнити» те, як користувач відчуває навколишній світ. AR зазвичай досягає цього, використовуючи фізичну площину, як «фон» гри чи програми, а потім додаючи елементи, створені віртуально для взаємодії з реальним світом. Крім того, на відміну від віртуальної реальності, яка занурює користувачів у повністю вигаданий всесвіт, доповнена реальність створює суміш фізичного оточення та цифрових структур.

Існуючі інструменти полегшують процес створення AR для розробника, оскільки створені для взаємодії з доповненою реальністю і містять різноманітні інтеграції пов'язані з конкретною задачею програміста.

Мета

Метою даної магістерської роботи є дослідження існуючих алгоритмів доповненої реальності, їх порівняльна характеристика за допомогою створеної програми та розробка інтерактивних навчальних матеріалів, що полегшить процес вивчення анатомії для студентів у розширеному вигляді за допомогою доповнення 3D моделлю з використанням мобільного пристрою.

Об'єкт дослідження

Об'єктом дослідження обрано програмне забезпечення Vuforia SDK та інструмент розробки AR-додатків для мобільних пристроїв Unity3D, мови програмування C#, Python та бібліотека OpenCV.

Предмет дослідження

Предметом дослідження цієї роботи є методи і алгоритми, що застосовуються у доповненої реальності, функціональність Unity3D для створення додатку на Android OS, мови програмування C# та Python з додатковою бібліотекою OpenCV.

Ключові слова

Доповнена реальність, Vuforia SDK, AR, Unity3D, OpenCV, 3D модель, розпізнавання зображення, ключові точки, дескриптор, SIFT, ORB, BRISK, навчальні матеріали, порівняння.

ABSTRACT

Topicality

The relevance of implementing augmented reality technology in any field helps to facilitate work, where it is important to encourage users to show interest in the information presented, thereby improving the visualization of educational theses.

Augmented reality combines a real-world environment with a computer environment to "augment" the user's experience of the world around them. AR typically achieves this by using a physical plane as the "background" of a game or application, then adding elements created virtually to interact with the real world. Also, unlike virtual reality, which immerses users in a completely fictional universe, augmented reality creates a mixture of physical surroundings and digital structures.

Existing tools make the process of creating AR easier for the developer, as they are designed to interact with augmented reality and contain various integrations related to the specific task of the programmer.

Purpose of the research

The purpose of this master's work is to study the existing algorithms of augmented reality, their comparative characteristics using the created program and the development of interactive educational materials that will facilitate the process of studying anatomy for students in an expanded form through the addition of a 3D model using a mobile device.

Object of research

Vuforia SDK software and Unity3D mobile AR application development tool, C#, Python programming languages and OpenCV library are chosen as the research object.

Subject of research

The subject of research of this work is methods and algorithms used in augmented reality, Unity3D functionality for creating an application on Android OS, C# and Python programming languages with an additional OpenCV library.

Keywords

Augmented reality, Vuforia SDK, AR, Unity3D, OpenCV, 3D model, image recognition, key points, descriptor, SIFT, ORB, BRISK, training materials, comparison.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AR – (*Augmented Reality*) доповнена реальність.

VR – (*Virtual Reality*) віртуальна реальність.

SDK – (*Software Development Kit*) це набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми за визначеною технологією або для певної платформи.

Vuforia SDK – платформа доповненої реальності та інструментарій розробника програмного забезпечення доповненої реальності.

Unity – це інструмент для розробки дво- і тривимірних додатків та ігор [1].

SIFT – (*Scale-invariant feature transform*) алгоритм із області комп'ютерного зору [10].

ORB – (*Oriented FAST and Rotate BRIEF*) детектор локальних ознак [19].

BRISK – (Binary Robust Invariant Scalable Keypoints) алгоритм виявлення та опису точки ознаки з інваріантністю масштабу та інваріантністю обертання [24].

ПЗ – сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації цих програм.

ВСТУП

З кожним роком все більша увага приділяється доповненій реальності як інструменту для розробки цілого ряду навчальних засобів. По мірі того, як технології продовжують розвиватись, викладачі постійно шукають нові та креативні способи навчання своїх студентів.

Однією з найсучасніших альтернатив є використання додатків доповненої реальності у навчанні, що може включати створення цікавих тривимірних демонстрацій предметів, яке збільшує залученість учнів у процес вивчення нового матеріалу. Засвоїти та опрацювати інформацію про будь-який курс за допомогою AR стає легше та швидше, ніж вчитися з Інтернету чи книг.

Впровадження освітніх додатків із ресурсами доповненої реальності на мобільних пристроях викликає необхідність розуміння ефективності алгоритмів на яких базується середовище розробки, здатних задовольнити вимоги та ресурси, необхідні для проекту. Завдяки великій різноманітності інструментів, доступних для створення програмного забезпечення доповненої реальності, буде здійснено порівняльний аналіз між алгоритмами на яких вони найчастіше базуються, щоб надати користувачам інформацію про те, які функції доповненої реальності найефективніші в розробці освітніх рішень.

Актуальність впровадження технології доповненої реальності в будь-яку сферу допомагає полегшити роботу, де важливе заохочення користувачів проявляти інтерес до викладеної інформації, тим самим покращуючи візуалізацію навчальних тез.

Мета роботи – представлення порівняльного аналізу трьох алгоритмів SIFT, ORB та BRISK для вибору найкращого у роботі з розпізнаванням зображень, розробка інтерактивних навчальних матеріалів, створення додатку, що полегшить процес вивчення анатомії для студентів у розширеному вигляді за допомогою доповнення 3D моделлю з використанням мобільного пристрою.

Для досягнення мети роботи були сформовані наступні завдання:

- дослідити алгоритм розпізнавання цілей доповненої реальності SIFT;
- провести порівняльну характеристику алгоритмів, які використовуються у доповненій реальності та їх ефективність за допомогою створення програми, що базується на OpenCV бібліотеці;
- розробити інтерактивні навчальні матеріали у вигляді тестового мобільного додатку з підтримкою доповненої реальності і з можливістю візуалізації 3D моделей після розпізнавання зображення.

1 ДОПОВНЕНА РЕАЛЬНІСТЬ

1.1 Визначення доповненої реальності

Технологія доповненої реальності (AR) – це технологія, яка пов’язує віртуальну інформацію з реальним оточенням. Технічні засоби, які вона використовує, включають мультимедіа, 3D-моделювання, відстеження та реєстрацію в реальному часі, інтелектуальну взаємодію тощо. Її принцип полягає у застосуванні віртуальної інформації, що генерується комп’ютером, такої як текст, зображення, тривимірні моделі, музика, відео до реального світу після моделювання [2].

Десять років тому досвідчені дослідники були б серед небагатьох, хто зміг створити AR додатки. Зазвичай вони обмежувалися демонстраційними прототипами або виробництвом спеціального проекту, який працював протягом обмеженого періоду часу. Тепер розробка досвіду AR стала реальністю для широкого кола розробників мобільного програмного забезпечення. Протягом останніх кількох років ми спостерігали великий прогрес у обчислювальній потужності, мініатюризації датчиків, а також все більш доступних мультимедійних бібліотек. Ці досягнення дозволяють розробникам створювати додатки AR легше, ніж будь-коли раніше [3].

Щоб мати можливість створювати складні програми AR, потрібно зрозуміти, що насправді таке доповнена реальність.

Технологічні вимоги доповненої реальності набагато більші ніж для віртуальної реальності, і саме тому її розвиток зайняв більше часу. Однак ключові компоненти, необхідні для створення системи доповненої реальності залишилися такими ж з тих пір, як Айвен Сазерленд почав працювати і сформував SketchPad проект у 1960-х роках [4].

1.2 Ключові підходи технології доповненої реальності

Для поточного покоління смартфонів існує два основних підходи, які можна реалізувати для систем доповненої реальності. Ці системи характеризуються специфічними методами виявлення та діапазоном

взаємодій. Обидва підтримують різні області застосування. Системи доповненої реальності на основі сенсорів і комп'ютерного зору використовують відео, щоб бачити крізь дисплей, покладаючись на камеру та екран мобільного телефону.

1.2.1 AR на основі датчиків

Перший тип системи називається AR на основі датчиків, яку часто називають GPS плюс інерційна AR (або іноді зовнішні системи AR). Доповнена реальність такого типу використовує датчики місцезнаходження та орієнтації телефону, комбінація яких забезпечує глобальне місцезнаходження користувача у фізичному світі.

Датчик розташування в основному підтримується приймачем GNSS (супутникова система навігації). Одним з найпопулярніших таких приймачів є GPS, який присутній на більшості смартфонів.

Виміряне положення та орієнтація портативного пристрою надає інформацію про відстеження для реєстрації віртуальних об'єктів на фізичній сцені. Місцезнаходження, яке повідомляє модуль GPS, може бути неточним і оновлюватися повільніше під час руху, що може призвести до затримки. Одним з найпопулярніших типів додатків AR, що використовують системи на основі сенсорів є браузер AR, який візуалізує точки інтересу, тобто просту графічну інформацію про речі навколо нас. Якщо спробувати деякі з найпопулярніших продуктів, як-от Junaio, Layar або Wikitude, можна помітити цей ефект відставання [5].

Перевага цієї технології полягає в тому, що сенсорна доповнена реальність може працювати практично в будь-якому фізичному положенні на відкритому повітрі по всьому світу. Однією з обмежень таких систем є те, що вони не працюють у приміщенні (або не дуже добре) або в прихованих місцях, де не видно неба, наприклад, у лісі чи на вулиці з високими будинками навколо.

1.2.2 AR на основі комп'ютерного зору

Іншим популярним типом є доповнена реальність на основі комп’ютерного зору. Ідея полягає в тому, щоб використовувати потужність вбудованої камери замість того, щоб просто знімати та відображати фізичний світ. Ця техніка часто використовується в поєднанні з обробкою зображень і алгоритмами комп’ютерного зору, які аналізують зображення для виявлення будь-якого об’єкта, видимого камері.

Сучасна технологія дозволяє розпізнавати різні типи площинного графічного вмісту, наприклад, спеціально розроблений маркер (відстеження на основі маркерів) або більш природний вміст (відстеження без маркерів). Одним з недоліків є те, що AR на основі комп’ютерного бачення важка в обробці і може дуже швидко розрядити батарею. Хоча смартфони останніх поколінь більш пристосовані для вирішення цього типу проблем, оскільки вони оптимізовані для більшого об’єму споживання енергії.

1.3 Принципи роботи доповненої реальності

Додатки доповненої реальності можна охарактеризувати як двоетапний процес згідно визначення з книги [6]. Загалом, розглянемо два основних етапи, які повинні відбуватися на кожному кроці застосунку:

1. У додатку потрібно визначити поточний стан фізичного світу та визначити поточний стан віртуального світу.
2. Програма повинна відображати віртуальний світ під час реєстрації в реальному світі таким чином, щоб учасник відчував зразки віртуального світу як частину свого фізичного світу, а потім повернутися до кроку 1, щоб перейти на наступний крок.

Якщо розглядати ці два кроки, то можна легко побачити, що для їх досягнення можна використовувати багато різних методів, і для їх реалізації можна маніпулювати багатьма різними технологіями.

Існує три основні компоненти системи доповненої реальності для підтримки щойно перерахованих кроків:

1. Сенсор(и) для визначення стану фізичного світу, де розгорнута програма;
2. Процесор для оцінки даних сенсорів, для реалізації «правил» віртуального світу та для генерування сигналів, необхідних для керування дисплеєм;
3. Дисплей придатний для створення враження, що віртуальний і реальний світ співіснують, що впливає на відчуття учасником поєднання фізичного та віртуального світу.

Розглянемо всі категорії по порядку і детальніше побачимо роль, яку кожен із цих компонентів відіграє у загальній системі.

Щоб правильно реагувати на фізичний світ, додаток повинен мати інформацію про реальний світ у реальному часі.

У системах AR використовуються три основні категорії датчиків:

1. Датчики, що використовуються для відстеження
2. Датчики для збору інформації про навколишнє середовище
3. Датчики для збору вводу користувачів

Для того, щоб використати комп’ютерний зір, необхідний датчик – це камера, яка "бачить" реальний світ та програмне забезпечення для аналізу зображень. Виходячи з того, що зібрано камерою, можна визначити, де вона знаходиться і як вона орієнтована по відношенню до сцени. Після отримання всієї необхідної інформації, програмне забезпечення розраховує, де повинна розміститись камера, щоб побачити певний вид. Таким чином, в середовищі мають бути підказки, які камера зможе використати як орієнтири.

Для того, щоб спростити проблему комп'ютерного зору, більшість програм AR використовують орієнтири, які штучно розміщуються в оточенні, оскільки їх легше розпізнати.

1.4 Розпізнавання на основі комп’ютерного зору та Vuforia

В той час, коли зазвичай камеру телефону використовують лише для відтворення реального світу як фону моделі, доповнена реальність на основі комп’ютерного зору йде ще далі, обробляючи кожен кадр зображення, щоб знайти знайомі візерунки (або особливості зображення) у знімках камери.

У типовому додатку AR на основі комп’ютерного бачення плоскі об’єкти (наприклад, маркери кадрів або об’єкти відстеження природних особливостей) використовуються для розташування камери в локальній системі координат. Це на відміну від глобальної (земної) системи координат, яка застосовується в AR на основі датчиків, дозволяє більш точно та стабільно накладати віртуальний вміст у цій системі координат. В результаті отримання інформації про відстеження дозволяє оновлювати інформацію про віртуальну камеру в нашому механізмі візуалізації 3D-графіки та автоматично надає нам реєстрацію.

Щоб успішно реалізувати AR на основі комп’ютерного зору, нам потрібно зрозуміти, які фізичні об’єкти ми можемо використовувати для відстеження камери. Наразі існують два основних підходи для цього: маркери кадрів і цілі відстеження природних ознак, як показано на наступному рисунку 1.1.

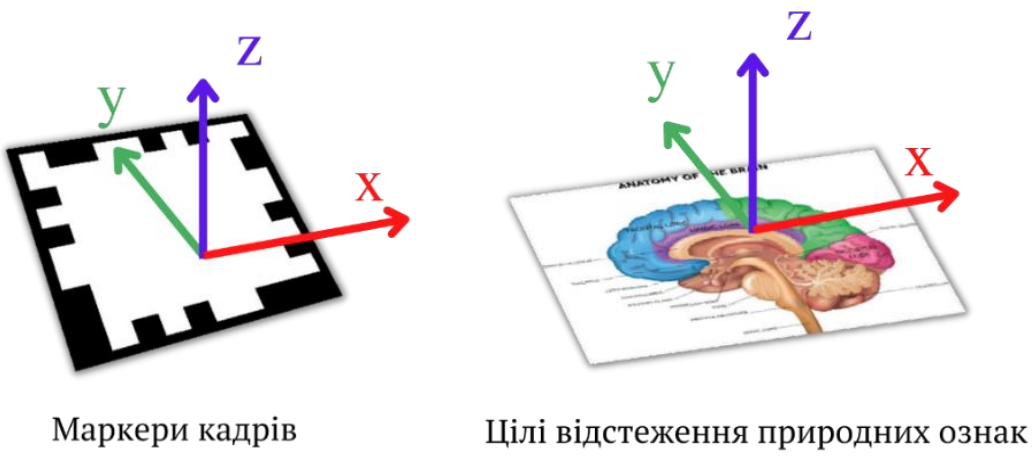


Рисунок 1.1 – Два основні підходи для відстеження камери

Маркери кадрів

На початку розвитку мобільної доповненої реальності надзвичайно важливо було використовувати ефективні алгоритми обчислень. Алгоритми комп'ютерного зору традиційно вимогливі, оскільки вони, як правило, покладаються на аналіз зображень, складні геометричні алгоритми та математичні перетворення, підсумовуючи велику кількість операцій, які мають виконуватися на кожному проміжку часу. Тому одним із перших підходів до AR на основі комп'ютерного зору було використання порівняно простих типів об'єктів, які можна було б виявити за допомогою алгоритмів із низькими вимогами до обчислень, таких як маркери зображення. Ці маркери зазвичай визначаються лише на рівні сірого, що спрощує їх аналіз і розпізнавання в традиційному фізичному світі.

Після того, як отримане зображення камери перетворюється на зображення у відтінках сірого, застосовується поріг, тобто рівень сірого перетворюється на чорно-біле зображення. Наступний крок, алгоритм виявлення прямокутника шукає ребра в цьому спрощеному зображенні, за яким потім слідує процес виявлення замкнутих контурів і, можливо, фігур паралелограма. Подальші дії робляться, щоб переконатися, що виявлений контур дійсно є паралелограмом (тобто він має рівно чотири точки і пару паралельних прямих). Після підтвердження форми аналізується вміст маркера. На етапі перевірки шаблону для ідентифікації маркера витягується двійковий шаблон у межах маркера. Це важливо, щоб мати можливість накладати різний віртуальний вміст на різні маркери.

На останньому кроці оцінки позиції обчислюється переміщення та поворот камери в локальній системі координат маркера або навпаки.

Цілі відстеження природних особливостей

Загальна ідея відстеження природних ознак полягає у використанні кількості (теоретично лише трьох, а на практиці кількох десятків чи сотень) локальних точок на цілі для обчислення пози камери. Проблема полягає в

тому, що ці точки мають бути надійно виявлені та відстеженні. Це досягається за допомогою передових алгоритмів комп'ютерного зору для виявлення та опису цікавої точки. Точки інтересу мають гострі, чіткі деталі – це можуть бути кути, наприклад, із використанням градієнтних орієнтацій, які підходять для точок інтересу, позначених жовтими хрестиками на наступному рисунку 1.2.

Як бачимо, коло або пряма не мають чітких рис і не підходять для точок інтересу:

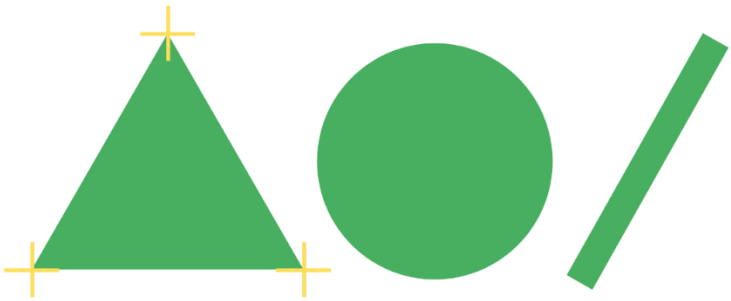


Рисунок 1.2 – Визначення цікавих точок на різних фігурах

Багато особливостей можна знайти на добре текстурованих зображеннях, наприклад, зображення анатомії мозку людини, яке використовується в цій магістерській роботі:

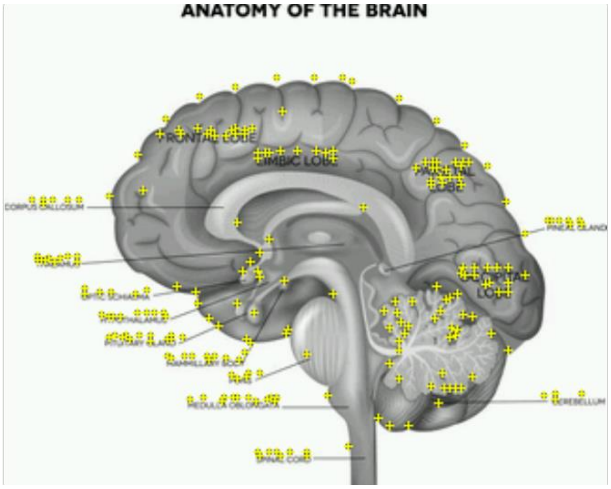


Рисунок 1.3 – Зображення з виявленими цікавими точками

Варто пам'ятати, що на зображеннях з однорідними кольоровими ділянками або м'якими краями, наприклад, блакитне небо або деякі

зображення, відтворені комп’ютерною графікою, неможливо чітко визначити характерні точки.

1.5 Випадки застосування технології доповненої реальності в освіті та навчанні

Сьогодні досліджують AR-рішення для кращого та ефективного навчання. Розглянемо випадки використання доповненої реальності в освіті в наступних галузях.

Медична сфера

Студенти-медики можуть удосконалити як свої знання, так і навички, скориставшись перевагами доповненої реальності. Приклади використання технології AR включають:

- створення моделей людського тіла, які дозволяють студентам-медикам глибоко вивчити анатомію;
- забезпечення більше можливостей для навчання за допомогою моделювання;
- практичні операції на віртуальних пацієнтах.

Історія

За допомогою технології AR та мобільних пристроїв користувачі можуть приносити 3D-об’єкти до класу, а потім ходити та досліджувати їх. Це можуть бути торнадо, вулкани, історичні документи або навіть ДНК. Це було реалізовано у додатку Google Expeditions, що охоплює різні теми та пропонує більше ста AR-експедицій через систему кровообігу, історію технологій та посадку на Місяць [7].

Космічна галузь

Космічна галузь однією з перших, хто застосовує передові технології. Тому, такі технології, як AR, можуть допомогти космонавтам у виконанні таких завдань, як підтримка космічної станції. Використовуючи захисні окуляри, працівники можуть отримувати наочні інструкції з роботи, не

звертаючись до посібників. NASA вже протестувало проект Sidekick, який використовує HoloLens для надання віртуальних ілюстрацій та інструкцій, що допомагають членам екіпажу виконувати складні завдання. На думку NASA, ця можливість може «зменшити вимоги до підготовки екіпажу» [8].

Військова підготовка

Хоча навчання відіграє життєво важливу роль у військовому секторі, не завжди можливо розмістити солдатів у певному місці для проведення тренувань. Технологія доповненої реальності використовується для створення середовища, необхідного для навчання солдатів, дозволяючи їм частіше тренуватися. Хоча AR не може повністю замінити традиційну військову підготовку, вже існують системи, які позбавляють необхідності їздити у віддалені місця та допомагають солдатам тренуватися, не піддаючи їх небезпеці.

2 АНАЛІЗ АЛГОРИТМУ РОЗПІЗНАВАННЯ ЗОБРАЖЕННЯ В ДОПОВНЕНІЙ РЕАЛЬНОСТІ

2.1 SIFT – масштабонезалежне перетворення ознак

Цей алгоритм із області комп'ютерного зору, який виявляє і описує локальні ознаки зображення [9]. Саме його використовує Vuforia SDK для розпізнавання зображень Image Targets. Також цей алгоритм застосовується для сканування образів, побудови карт для навігації роботів, 3D-реконструкції, розпізнавання жестів, відстеження об'єктів та інше.

Для будь-якого об'єкта на зображенні можна отримати цільові точки, щоб створити «опис ознак» об'єкта. Вони здобуті з зображення для ідентифікації об'єкта при знаходженні його на іншому тестовому зображенні, що містить безліч інших об'єктів. Для надійного розпізнавання важливо, щоб отримані ознаки на початковому зображенні можна було розрізнити навіть при зміні масштабу зображення, освітлення і наявності шуму. Такі точки, зазвичай, знаходяться на високо контрастних ділянках зображення, наприклад на межах об'єкта.

Іншою важливою характеристикою цих ознак є те, що відносні позиції між ними не повинні змінюватися від одного зображення до іншого. Тобто, характерні точки, що знаходяться на рухомих чи гнучких об'єктах, зазвичай не працюватимуть, якщо відбуватимуться будь-які зміни у їх внутрішній геометрії при порівнянні двох зображень. Однак, на практиці алгоритм SIFT визначає значно більшу кількість проявів на зображенні, що зменшує внесок похибок, спричинених локальними змінами, до середньої похибки загальних порівнянь ознак.

SIFT може достовірно ідентифікувати об'єкти навіть серед шуму або часткового перекриття, тому що дескриптор ознак, що використовується в SIFT є інваріантним до лінійного масштабування, змін просторової орієнтації, освітлення, і частково до афінного перетворення.

Ключові стадії:

- **Масштабно-інваріантне визначення ознак**

Перший етап обчислення здійснює пошук у всіх масштабах та місцях зображення. Він ефективно реалізовується за допомогою функції різниці гаусіанів для визначення потенційних зацікавлених точок, інваріантних за масштабом та орієнтацією.

- **Локалізація ключових точок**

Для кожного місця розташування кандидата підходить детальна модель для визначення місця та масштабу. Ключові точки підбираються виходячи з мір їх стійкості.

- **Призначення орієнтації**

Кожній точці ключових точок присвоюється одна або кілька орієнтацій на основі локальних напрямків градієнта зображення. Усі майбутні операції виконуються на даних з зображення, які були перетворені відносно призначеної орієнтації, масштабу і місця розташування для кожної ознаки, забезпечуючи тим самим інваріантність цих перетворень.

- **Дескриптор ключової точки**

Локальні градієнти зображення вимірюються у вибраній шкалі в області навколо кожної ключової точки. Вони перетворюються на представлення, що дозволяє досягти значних рівнів локального викривлення форми та зміни освітленості.

2.2 Детальний алгоритм SIFT

2.2.1 Виявлення екстремумів масштабного простору

Перший етап виявлення ключових точок – це визначення локацій та масштабів, які можна повторно присвоїти під різними видами одного і того ж об'єкта.

Виявлення місць, які незмінні до масштабних змін зображення, можна здійснити шляхом пошуку стабільних ознак у всіх можливих масштабах,

використовуючи безперервну функцію масштабування, відому як масштабний простір [10].

Масштабний простір зображення визначається як функція $L(x, y, \sigma)$, яка отримана зі згортки вхідного зображення $I(x, y)$ з оператором розмиття Гауса $G(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{2.1}$$

де * – це згортка x та y, і $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$.

Спочатку береться вихідне зображення і декілька разів застосовується розмиття по Гаусу з різними значеннями сигма. Це призводить до певної к-сті розмитих зображень однакового розміру. У контексті SIFT це відомо, як октава.

Після встановлення всіх зображень застосовується оператор лапласіана гаусіана (LoG – Laplacian of Gaussian), який використовується для точного визначення країв зображення. Обчислюється похідна другого порядку зображення, яка визначає всі краї та кутові точки, які будуть використовуватися як потенційні ключові точки зображення. Оскільки похідна другого порядку надзвичайно чутлива до шуму, розмиття по Гаусу допомагає стабілізувати похідну. Проте, є ще одна проблема з похідними другого порядку — їх обчислення дорого об’ємне, що не ідеально, якщо ми хочемо використовувати SIFT для додатків реального часу. Щоб зменшити витрати на обчислення похідної другого порядку, виконується апроксимація. Ми апроксимуємо похідну другого порядку як різницю гаусіанів (DoG – Difference of Gaussians). Наступна формула показує наближення [11]:

$$\begin{aligned} D(x, y, \sigma) &= \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned} \tag{2.2}$$

Тут D являє собою різницю гаусіанів, G представляє оператор розмиття Гауса, L є оператором Лапласа, k — мультиплікативна константа, яка визначає кількість розмиття в кожному зображенні в масштабному

просторі. Масштабний простір визначається як набір зображень, які були або збільшені, або зменшені з метою обчислення ключових точок. Наприклад, на рисунку 2.1 показано два набори зображень; один – це вихідний набір із п’яти зображень, які були розмиті з різним радіусом розмиття, а інший – набір зменшених зображень:

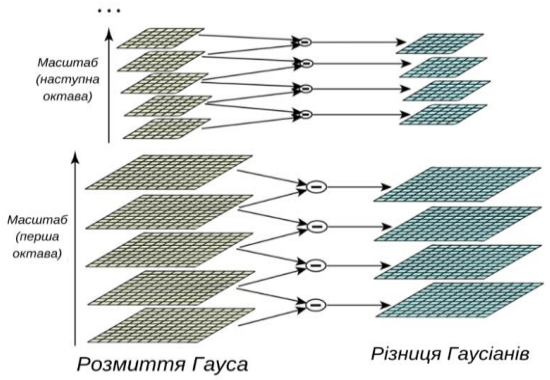


Рисунок 2.1 – Два набори зображень, розмитих по Гаусу

Різниця гаусіанів забезпечує близьке наближення до масштабно-нормованого лапласіана гаусіана $\sigma^2\nabla^2 G$, яке вивчав Ліндеберг (1994). Він показав, що для справжньої незмінності масштабу необхідна нормалізація оператора Лапласа коефіцієнтом σ^2.

Зв'язок між DoG і $\sigma^2 LoG$ можна зрозуміти з рівняння дифузії (параметризувавши σ більш звичним $t = \sigma^2$):

$$\frac{\partial G}{\partial \sigma} = \sigma\nabla^2 G. \tag{2.3}$$

Звідси ми бачимо, що $\nabla^2 G$ можна обчислити від наближення скінченної різниці до $\frac{\partial G}{\partial \sigma}$, використовуючи різницю сусідніх масштабів в $k\sigma$ та σ:

$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x,y,k\sigma)-G(x,y,\sigma)}{k\sigma-\sigma} \tag{2.4}$$

і крім цього,

$$G(x,y,k\sigma) - G(x,y,\sigma) \approx (k-1)\sigma^2\nabla^2 G. \tag{2.5}$$

Це показує, що коли функція різниці гаусіанів відрізняється постійним коефіцієнтом, вона вже включає в себе нормалізацію масштабу σ^2, необхідну

для масштабонезалежного оператора Лапласа. Коефіцієнт $(k-1)$ в рівнянні є постійним для всіх масштабів і тому не впливає на розташування екстремумів. Похибка наближення йде до нуля, оскільки k прямує до 1, але на практиці виявлено, що наближення майже не впливає на стабільність знаходження або локалізацію екстремумів навіть для значних відмінностей у масштабі, таких як $k = \sqrt{2}$.

2.2.2 Інтерполяція сусідніх даних для точності розташування

На цьому етапі знаходяться точки, які є локальними екстремумами. Це означає, що потрібно визначити точки, які найкраще відображають область зображення (іншими словами, околиці точки) у різних масштабах. Для того, щоб знайти ці ключові точки, ми перебираємо кожен піксель і порівнюємо його з усіма сусідами. До цього часу вважалося, що сусідами є вісім пікселів, які примикають до пікселя, але для SIFT розглядається не тільки ці вісім пікселів, а й дев'ять пікселів на попередніх зображеннях і під цим зображенням у простір шкали або октава, як на рисунку 2.2. Тут порівнюються значення пікселя з його 26 сусідніми пікселями. Ми вибираємо цю точку як локальний екстремум, якщо вона є мінімальним або максимальним пікселем серед сусідніх. У середньому рідко порівнюються значення пікселя з усіма 26 значеннями; достатньо лише кілька порівнянь, щоб перевірити.

Як бачимо з рисунку 2.2 максимум та мінімум зображень різниці Гауса визначаються шляхом порівняння пікселя (позначеного символом X) з 26 його сусідами в 3x3 областях за поточним та сусіднім масштабом (позначені кружечками).

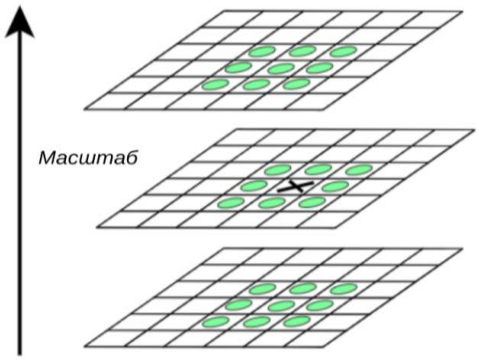


Рисунок 2.2 – Знаходження локальних максимумів та мінімумів

Після виявлення ключової точки шляхом порівняння сусідніх пікселів, наступним кроком є виконання детальної відповідності даних поблизу місця розташування, масштабу та співвідношення основних кривин. Ця інформація дозволяє відхиляти точки, які мають низький контраст (і тому чутливі до шуму) або погано локалізовані уздовж краю.

Початкова реалізація цього підходу просто розташовувала ключові точки у місці розташування та масштабі центральної вибіркової точки. Однак було розроблено кращий метод пристосування 3D квадратичної функції до локальних точок вибірки для визначення інтерпольованого місця максимуму, який забезпечує значне поліпшення відповідності та стабільності. Цей підхід використовує розклад Тейлора (до квадратичних доданків) функції масштабного простору $D(x, y, \sigma)$, зміщену до поточної точки як початок координат:

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \tag{2.6}$$

де D та його похідні обчислюються в точці, яку ми зараз перевіряємо на екстремуми, $x = (x, y, \sigma)^T$ – це зміщення від цієї точки. Місце екстремуму $\hat{x}$ визначається, приймаючи похідну цієї функції відносно x і прирівнюючи її до нуля, дає [12]:

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x}. \tag{2.7}$$

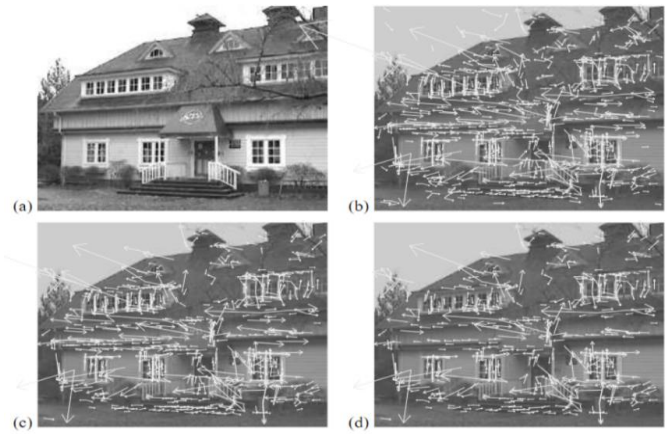


Рисунок 2.3 – Цей рисунок показує етапи вибору ключових точок

Як стверджується у методі, параметри можуть бути оцінені за допомогою стандартних наближень різниці з сусідніх точок вибірки в DoG, що призводить до лінійної системи 3×3, яку можна ефективно розв'язати. Якщо зсув $\hat{x}$ більший за 0,5 в будь-якому вимірі, то це означає, що екстремум лежить ближче до іншої точки вибірки. У цьому випадку виникає необхідність переоцінити $\hat{x}$, оскільки відповідне сусідство для наближення зміниться. Точки, які не збираються швидко, відкидаються як нестабільні. Остаточне зміщення $\hat{x}$ додається до місця його вибіркової точки, щоб отримати інтерпольовану оцінку для розташування екстремуму.

Значення функції на локалізованому екстремумі $D(\hat{x})$ можна отримати, задавши

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x} \tag{2.8}$$

а будь-які точки зі значенням нижче певного порогу відхиляються, як точки низької контрастності.

На рисунку 2.3 показані ефекти вибору ключових точок на природне зображення. Щоб уникнути занадто великого захаращення, використовується зображення низької роздільної здатності 233 на 189 пікселів, а ключові точки відображаються як вектори, що дають розташування, масштаб та орієнтацію кожної ключової точки (орієнтація призначення описана нижче). На рисунку 2.3 (a) показано вихідне

зображення, яке зображено зменшеним контрастом. На рисунку 2.3 (b) показано 832 ключових точок за всіма виявленими максимумами та мінімумами функції різниці гаусіанів, тоді як 2.3 (c) показує 729 ключових точок, які залишаються після видалення тих, у кого значення $|D(\hat{x})|$ менше 0,03. Частина (d) буде пояснена в наступному пункті.

2.2.3 Усунення контурних країв

Для стабільності недостатньо відхилити ключові точки з низькою контрастністю. Функція різниці гаусіанів матиме сильну реакцію по краях, навіть якщо розташування вздовж краю погано визначене і, отже, нестійке до невеликої кількості шуму.

Погано визначена найвища точка функції різниці гаусіанів матиме велику головну кривину через край, але малу в перпендикулярному напрямку. Основні викривлення можна обчислити з 2x2 матриці Гессе H, обчисленої в місці розташування та масштабі ключової точки:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}. \tag{2.9}$$

Похідні оцінюються, беручи різниці сусідніх точок вибірки.

Власні значення H пропорційні основним викривленням D. З підходу Гарріса і Стефенса [13] ми можемо уникати явного обчислення власних значень, оскільки нам потрібно лише їх співвідношення. Нехай α буде власне значення з найбільшою величиною, а β буде з найменшою. Тоді ми можемо обчислити суму власних значень із залишку H та їх добутку з визначника:

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta, \tag{2.10}$$

$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

У тому випадку, коли детермінант є негативним, викривлення має різні знаки, і точка відхиляється як не екстремум. Нехай r – відношення між найбільшим та найменшим власним значенням, так що $\alpha = r\beta$. Тоді, отримаємо вираз

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}, \tag{2.11}$$

що залежить лише від співвідношення власних значень, а не від їх індивідуальних значень. Величина $(r+1)^2/r$ зростає як мінімум, коли дві власні величини рівні і вона збільшується з r. Крім того, щоб перевірити, що відношення основних кривин нижче деякого порогу r, нам потрібно лише перевірити умову:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}. \quad (2.12)$$

Це дуже ефективно для обчислення, з менше ніж 20 операцій з плаваючою точкою необхідного для тестування кожної ключової точки. В цій роботі використовується значення $r = 10$, що виключає ключові точки, які мають співвідношення між основними кривизнами більшими за 10. Перехід від рисунку 2.3 (c) до (d) показує наслідки цієї операції.

2.2.4 Призначення орієнтації

Поки що ми маємо стабільні ключові точки, і ми знаємо, в яких масштабах вони були виявлені. Отже, ми маємо масштабну інваріантність. Тепер ми намагаємося призначити орієнтацію кожній ключовій точці. Ця орієнтація допомагає нам досягти інваріантності обертання. Ми намагаємося обчислити величину та напрямок гаусових розмитих зображень для кожної ключової точки.

Масштаб ключової точки використовується для вибору згладженого зображення Гауса L з найближчим масштабом, щоб усі обчислення виконувались в масштабонезалежний спосіб. Для кожного зразка зображення $L(x,y)$ в цьому масштабі величина градієнту $m(x,y)$ та орієнтація $\theta(x,y)$ попередньо обчислюється з використанням піксельних різниць:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}, \quad (2.13)$$

$$\theta(x,y) = tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))).$$

Величина та орієнтація розраховуються для всіх пікселів навколо ключової точки. Ми створюємо гістограму на 36 областей, яка охоплює

діапазон орієнтацій на 360 градусів. Кожен зразок, доданий до гістограми, зважується за його величиною градієнта та за гаусово-зваженим круговим вікном з масштабом σ, яке в 1,5 рази більше масштабу ключової точки.

Найвищі точки орієнтовної гістограми відповідають домінуючим напрямках локальних градієнтів. Спочатку знаходиться найвища точка гістограми, а потім будь-який інший локальний пік, який знаходиться в межах 80% від найвищої точки і також використовується для створення ключової точки з цією орієнтацією. Лише приблизно 15% точкам присвоєно декілька орієнтацій, але вони значною мірою сприяють стабільності. На кінець, парабола підходить до трьох значень гістограми, найближчих до кожного піку, щоб інтерполювати положення піку для кращої точності.

2.2.5 Дескриптор ключової точки

Попередні операції визначили розташування зображення, масштаб та орієнтацію зображення кожній ключовій точці. Наступним кроком є обчислення дескриптора для локальної області зображення, яка є дуже відмінною, але є максимально незмінною для інших варіантів, таких як зміна освітленості.

Просте співвідношення виправлень зображень є дуже чутливим до змін, які викликають неправильну реєстрацію зразків, наприклад, зміну афінної або тривимірної точки зору чи нежорсткі деформації. Кращий підхід продемонстровано Едельманом, Інтратором та Поджіо (1997 рік). Були проведені детальні експерименти, використовуючи 3D-комп'ютерні моделі об'єктів і форм тварин, які показують, що відповідні градієнти, дозволяючи змінити їх положення, призводять до набагато кращої класифікації за 3D-обертання. Ця ідея надихнула на представлення, що описане нижче, але воно дозволяє змінити позицію за допомогою іншого обчислювального механізму.

2.2.6 Визначення дескриптора

До цього часу ми досягли інваріантності масштабу та обертання. Тепер нам потрібно створити дескриптор для різних ключових точок, щоб мати можливість відрізнити їх від інших ключових точок. Щоб створити дескриптор, ми беремо вікно розміром 16x16 навколо ключової точки та розбиваємо його на 16 вікон розміром 4x4. Це можна побачити на наступному рисунку:

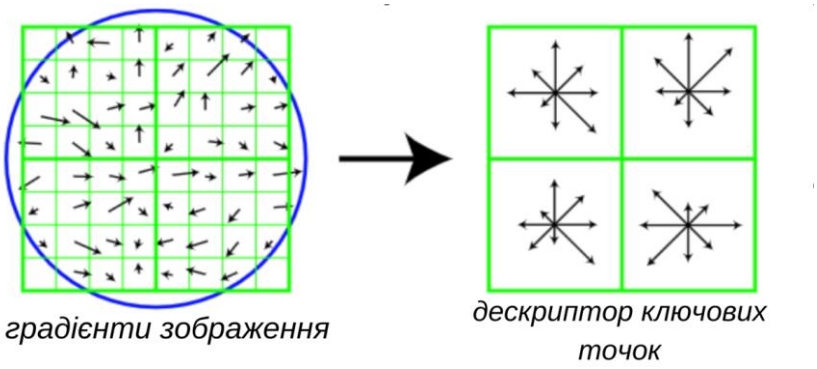


Рисунок 2.4 – Дескриптор ключових точок

Дескриптор ключових точок створюється спочатку обчисленням величини градієнта та орієнтації в кожній вибірковій точці зображення в області навколо місця розташування ключової точки, як показано зліва на рисунку 2.4. Він формується з вектору, що містить значення всіх записів гістограми орієнтації, які відповідають довжинам стрілок праворуч на рисунку 2.4, де показані гістограми масиву 2x2 орієнтації, тоді як підходи, наведені нижче, показують, що найкращі результати досягаються за допомогою масиву гістограми 4x4 з 8 орієнтирами в кожній. Тому тут використовують вектор елементів $4x4x8 = 128$ для кожної ключової точки.

На кінець, функціональний вектор модифікується для зменшення наслідків зміни освітленості. По-перше, вектор нормалізується на одиницю довжини. Зміна контрасту зображення в якій кожне значення пікселя помножене на константу, помножить градієнти на одну і ту ж константу, тому ця зміна контрасту буде скасована нормалізацією вектора. Зміна яскравості, при якій до кожного пікселя зображення додається константа, не

вплине на значення градієнта, оскільки вони обчислюються з піксельних різниць. Внаслідок цього дескриптор інваріантний для афінних змін освітленості.

2.3 Розпізнавання цілей Vuforia

Vuforia — це платформа доповненої реальності та набір інструментів розробника програмного забезпечення AR для мобільних пристроїв, розроблений Qualcomm. Бібліотека безкоштовна для використання та підтримує відстеження маркерів кадрів і природних ознак. Тим самим, Vuforia використовує технології комп'ютерного зору, а також відстеження плоских зображень чи простих об'ємних реальних об'єктів (наприклад, кубічних) в реальному часі.

Цільові зображення визначаються на основі природних особливостей, які витягуються з цільового зображення, а потім порівнюються під час виконання з функціями на живому зображенні з камери [14]. Щоб створити розпізнавання, яке точно виявиться, слід використовувати зображення, які володіють властивостями наведеними у таблиці 2.1.

Таблиця 2.1

Атрибут	Приклад
Багатий на деталі	Вулиця, група людей, колаж чи набір предметів
З добре вираженим контрастом	Має і світлі і темні області, які добре освітлені і не тьмяні у кольорі
Без повторюваних узорів	Трава, вигляд на будинок з однаковими вікнами, і інші повторювані елементи

Додатковий рейтинг визначає, наскільки добре зображення можна виявити та відстежити за допомогою SDK Vuforia. Цей рейтинг

відображається в менеджері цілей і повертається для кожної завантаженої цілі через веб-API.

Додатковий рейтинг може коливатися від 0 до 5 для будь-якого зображення. Чим вищий показник збільшення розміру цілі зображення, тим сильніша здатність його виявлення та відстеження. Нульова оцінка вказує на те, що ціль взагалі не відслідковується системою AR, тоді як зірковий рейтинг 5 вказує на те, що зображення легко відстежується системою AR.

Цільові зображення представляють зображення, які Vuforia Engine може виявляти та відстежувати. На відміну від традиційних маркерів, матричних кодів даних та QR-кодів, цільові зображення не потребують розпізнавання спеціальних чорно-білих областей або кодів. Vuforia виявляє та відслідковує функції, які природно знаходяться в самому зображенні, порівнюючи ці природні особливості з відомою базою даних цільових ресурсів. Після виявлення цільового зображення буде відслідковуватися зображення до тих пір, поки воно знаходиться хоча б частково в полі зору камери.

Існує дві фази розвитку з цільовими зображеннями. Спочатку потрібно розробити цільові зображення, а потім завантажити їх у Vuforia Target Manager для обробки та оцінки [15].

Цільові зображення можуть бути включені у програму за допомогою вбудованої бази даних пристроїв або в Інтернеті. Менеджер Vuforia Target створює цільові зображення та обробляє зображення для створення як даних, так і візуальних уявлень цільових особливостей за допомогою методу SIFT. Він також забезпечує очікуване виявлення та оцінку ефективності відстеження цілі.

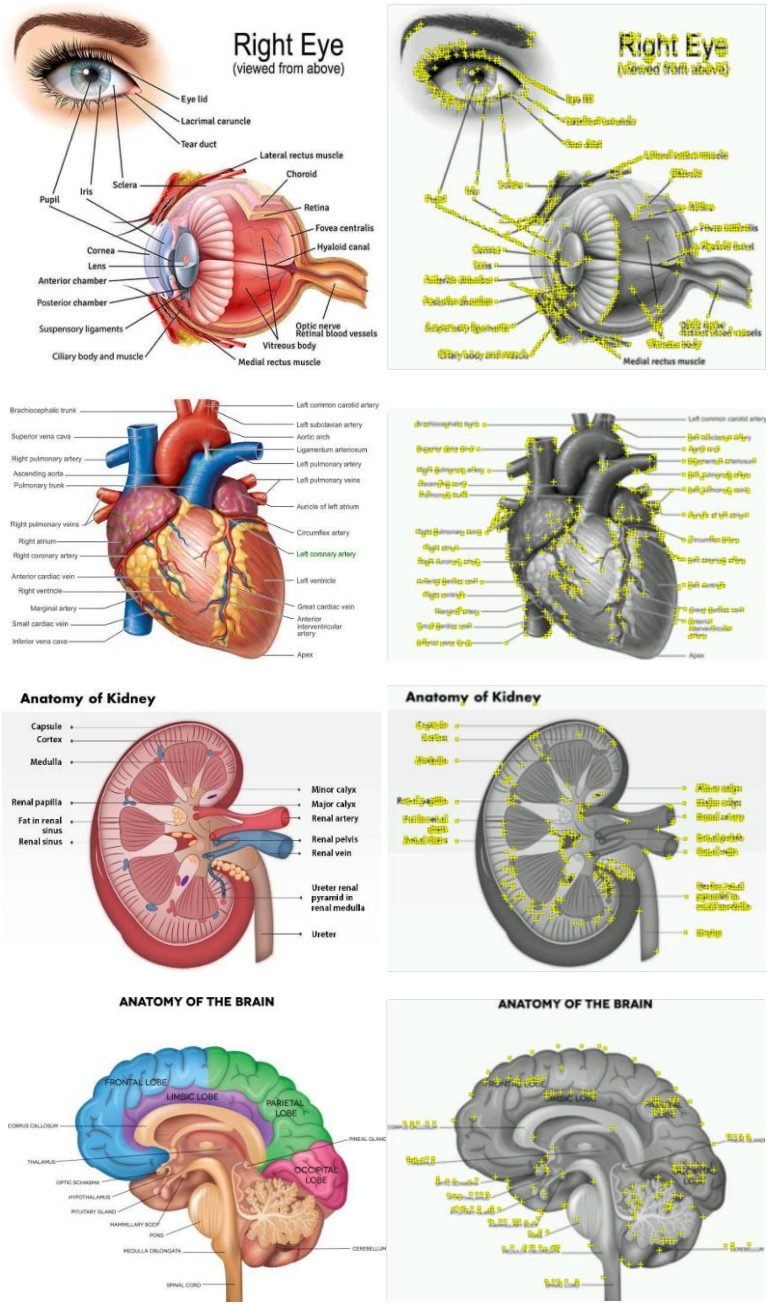


Рисунок 2.5 – Приклад початкових зображень та їх ключові точки

При використанні системи природних особливостей (natural features tracking) ми виділяємо два сімейства підходів в залежності від характеру використовуваних елементів зображення. Перша формується методами на основі країв, які співвіднесені з проекціями тривимірних кутів цільового об'єкта з областю високого градієнта зображення. Друге сімейство включає всі методи, засновані на інформації, наданій пікселями всередині проекції об'єкта.

Історично всі ранні підходи до відстеження були засновані головним чином на граничних методах, оскільки ці методи є одночасно ефективними з точки зору обчислень і відносно простими в реалізації. Вони також природно стійкі до змін освітлення, навіть для дзеркальних матеріалів, що не обов'язково вірно для методів, які розглядають внутрішні пікселі. Детальніший алгоритм було розглянуто у Розділі 2.

3 ПОРІВНЯННЯ ЕФЕКТИВНОСТІ РОЗПІЗНАВАННЯ ЗОБРАЖЕННЯ ЗА ДОПОМОГОЮ РІЗНИХ АЛГОРИТМІВ

3.1 Вступ

У цьому розділі досліджується порівняльна характеристика алгоритмів, які використовуються для ідентифікації зображення за допомогою виділення ключових точок і ознак. У контексті цього дослідження основні критерії, такі як кількість ключових точок і ознак, витягнутих алгоритмами, час виконання алгоритму та якість витягнутих цільових точок, розглядаються як показники ефективності. Для отримання результатів порівняння використовувалися однакові стеки даних.

Класифікація зображень є одним із найперспективніших застосувань машинного навчання, спрямованого на створення алгоритмів із здатністю розпізнавати та класифікувати вміст зображень із точністю, близькою до людської.

Реєстрація зображень — це процес відповідної схожості, вирівнювання та накладання двох або більше зображень сцени, зроблених в різний час, різними датчиками та/або з різних точок огляду. Іноді цей процес називають зіставленням (англ. matching) [16].

Час і точність реєстрації зображень, що базуються на ознаках в основному залежать від обчислювальної ефективності та надійності вибраного дескриптора ознак. Таким чином, вибір алгоритму виявлення ознак є вагомим чинником у програмах відображення та розпізнавання.

У цьому розділі буде представлена порівняльна характеристика алгоритмів, які використовуються для ідентифікації зображення за допомогою виділення ключових точок і ознак, а саме SIFT, ORB і BRISK. Це також вирішує ключову дилему: який алгоритм найкращий?

У літературі такі показники, як кількість ключових точок, виділених на одному зображенні, кількість правильних зіставлень між двома зображеннями та час виконання алгоритму використовуються для порівняння алгоритмів [17-18]. Ця робота базуватиметься на зазначених критеріях.

3.2 Огляд алгоритмів

Алгоритми природних ознак складаються з двох основних кроків, якими є знаходження ключових точок та обчислення їх дескриптора. Ключові точки — це окремі точки текстури об’єкта, такі як краї, кути, плями тощо, які можна легко знайти та виділити з решти зображення. Тим часом дескриптор ключової точки характеризує її, тому ключову точку можна знайти на зображенні та відрізнити від інших ключових точок, і, що є більш важливим, зіставити її з собою на іншому зображенні, навіть якщо присутні такі перетворення, як зміщення, зміни масштабу та обертання.

При розрахунку ключової точки можуть застосовуватися різні алгоритми залежно від типу програми, розглянемо декілька з них.

3.2.1 ORB

Алгоритм ORB (Oriented FAST and Rotated BRIEF) [19] – є ефективною альтернативою алгоритму SIFT [10]. ORB в основному є комбінацією детектора кутів для виявлення ключових точок FAST [20] і дескриптора BRIEF [21], але він також містить багато модифікацій для підвищення продуктивності. Методи орієнтованого FAST і оберненого BRIEF мають досить хорошу продуктивність.

ORB спочатку знаходить ключові точки за допомогою FAST, а потім застосовує вимірювання кутів Гарріса для кожного розташування ключової точки, таким чином забезпечуючи не максимальне придушення в межах зображення, оскільки FAST не забезпечує належного вимірювання кутів і не має надійності для багатомасштабних функцій.

Для визначення ключових точок різного розміру використовується пірамідне подання багатомасштабних функцій [22]. Для обчислення дескриптора ключової точки використовується модифікований BRIEF, де локальна орієнтація обчислюється за допомогою використання центроїда інтенсивності [23], який є зваженим усередненням інтенсивності пікселів у локальній ділянці, яка не відповідає центру ключової точки.

3.2.2 BRISK

BRISK (Binary Robust Invariant Scalable Keypoints) [24] – це алгоритм, який забезпечує інваріантність як масштабу, так і обертання. Щоб обчислити розташування об’єктів, він використовує кутовий детектор AGAST [25], який покращує FAST, збільшуючи швидкість, зберігаючи ту саму продуктивність виявлення. Для незмінності масштабу BRISK виявляє ключові точки в масштабно-просторовій піраміді, виконуючи не максимальне придушення та інтерполяцію для всіх масштабів. Щоб описати особливості зображення беруться точки вибірки, які розташовані в концентричних колах навколо об’єкта, причому кожна точка вибірки представляє розмиття за Гаусом навколишніх пікселів. Стандартне відхилення цього розмиття збільшується з відстанню від центру об’єкта.

Для визначення орієнтації використовується кілька порівнянь точок вибірки на великій відстані. Для кожного порівняння на великій відстані векторний зсув між точками вибірки зберігається та зважується відносною різницею інтенсивності. Потім ці зважені вектори усереднюються для визначення домінуючого напрямку градієнта ділянки. Потім шаблон вибірки масштабується та обертається, а дескриптор складається з 512 порівнянь точок вибірки на короткій відстані (наприклад, точки вибірки та її найближчих сусідів), що представляють локальні градієнти та форму в межах ділянки. Загалом BRISK вимагає значно більше обчислень і трохи більше місця для зберігання, ніж BRIEF або ORB.

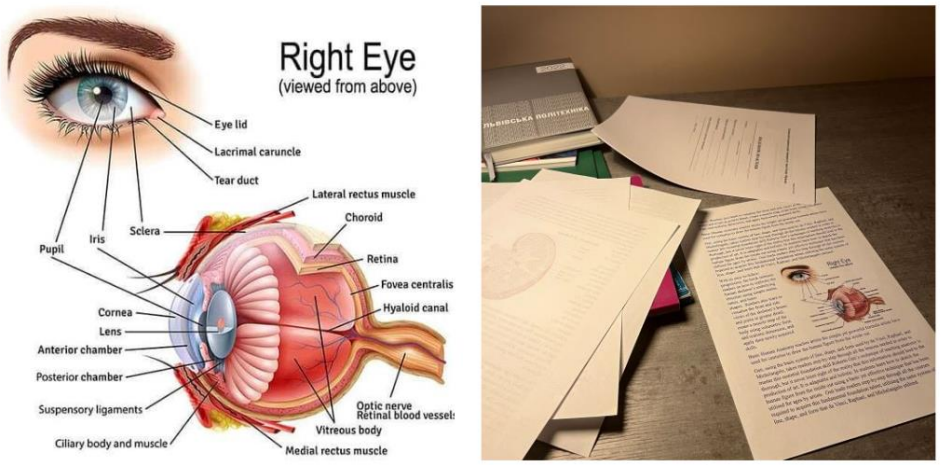
3.2.3 SIFT

Одним із найпотужніших природних алгоритмів, що включає інваріантність масштабу, обертання та стійкість до змін освітлення є SIFT (Scale Invariant Feature Transform), який детальніше досліджений у розділі 2. Цей алгоритм містить чотири основні кроки, на яких спочатку визначаються ключові точки на зображенні шляхом пошуку пікселів, які представляють екстремуми простору масштабів різниці Гауса (DoG). На наступному кроці ненадійні ключові точки видаляються, наприклад, ключові точки з погано визначеним розташуванням (по краях) або з низьким контрастом. На третьому кроці орієнтація призначається кожній ключовий точці, тому дескриптори можуть бути представлені відносно орієнтації ключової точки, і, таким чином, цільова точка є інваріантною щодо повороту зображення. Як показано на рис. 2.4, дескриптор обчислюється шляхом знаходження величини градієнта та орієнтації в кожній точці зображення в області навколо кожної ключової точки. Потім ці зразки накопичуються в орієнтаційні гістограми, що підсумовують вміст у підобластях, при цьому довжина кожної стрілки відповідає сумі величин градієнта поблизу цього напрямку в межах області [10].

3.3 Програмна реалізація

Для аналізу ефективності алгоритмів у PyCharm 2021 було розроблено фреймворк з використанням мови Python і бібліотеки OpenCV. Ця програма може запускати алгоритми ORB [19], SIFT [10] і BRISK [24]. Спочатку зчитується фіксоване зображення (рис. 3.1 a), виділяються ключові моменти та особливості з нього. Потім вони використовуватимуться для зіставлення з зображенням, взятого з реального світу за допомогою камери. На наступному кроці програма читає це фото для порівняння алгоритмів. Підготовлене фото – це знімок аркуша А4 на якому розміщена наша цільова картинка, який розташований на столі поміж інших предметів під певним кутом (рис. 3.1 b).

Ми спробуємо знайти фіксоване зображення у зображенні реального світу за допомогою зіставлення. Підготовлені фото використовуються окремо для кожного алгоритму. Таким чином, вважалося отримати правильні значення.

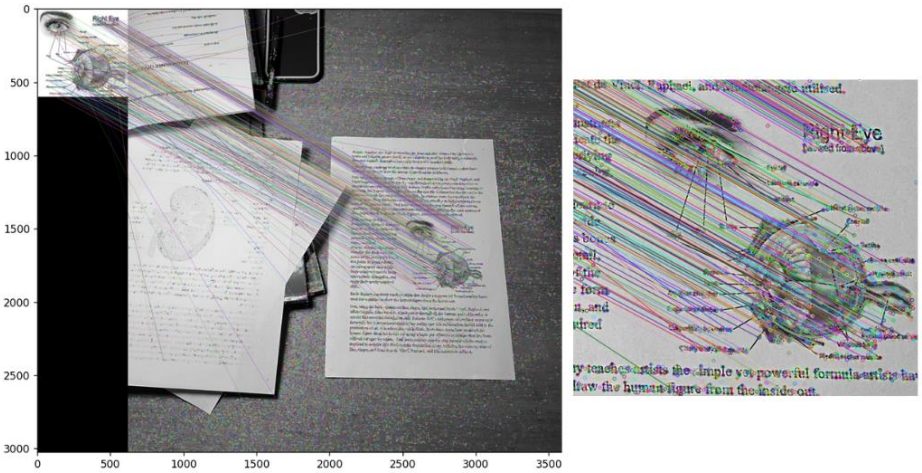


(a) (b)

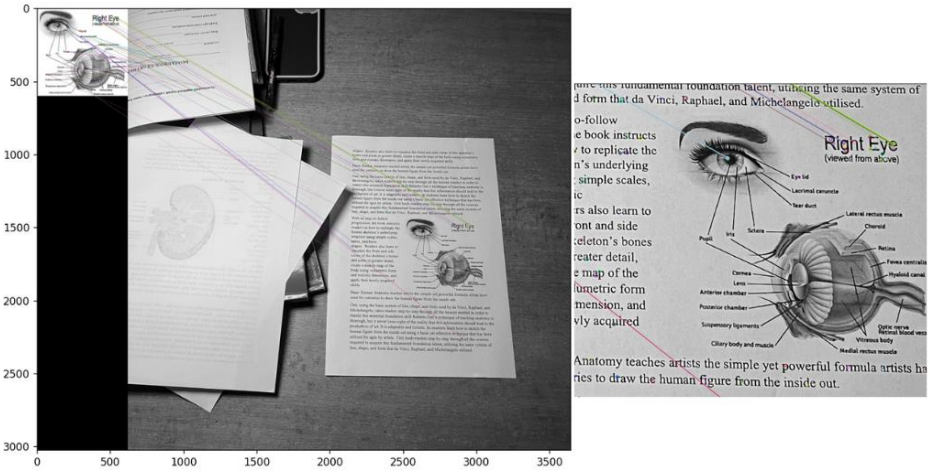
Рисунок 3.1 – Зображення (a) фіксованої картинки з ключовими точками, (b) знімок аркушу A4 з цільовим зображенням для розпізнавання.

Код було запущено на комп’ютері з процесором Intel Core i7-8750H, 2.20 ГГц, 16 ГБ оперативної пам’яті, NVIDIA GeForce GTX 1050 Ti та Microsoft Windows 10.

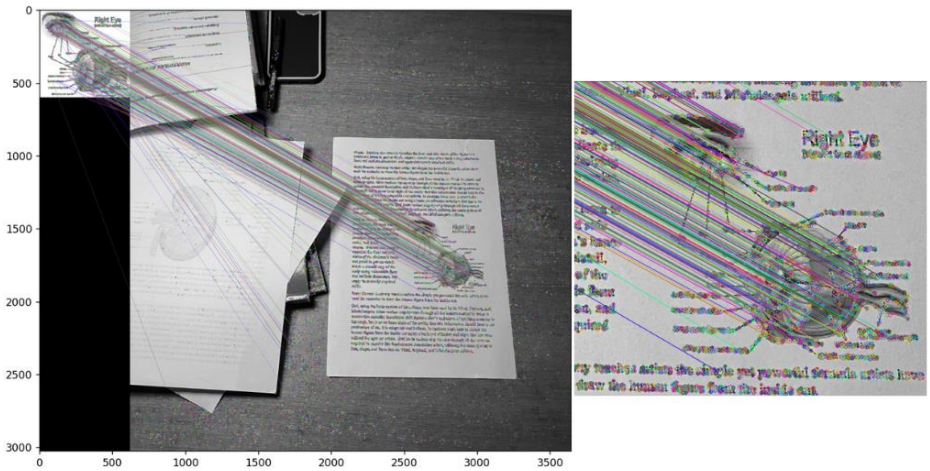
3.4 Демонстрація результатів



(a) Зіставлення зображень за допомогою алгоритму SIFT



(b) Зіставлення зображень за допомогою алгоритму ORB



(c) Зіставлення зображень за допомогою алгоритму BRISK

Рисунок 3.2 – Набори пар зображень та їх реєстрація за допомогою відповідних алгоритмів (a) SIFT, (b) ORB, (c) BRISK.

У цьому аналізі було виділено певні критерії, щоб отримати краще та повніше розуміння ефективності дескрипторів. У таблиці 3.1 представлені результати протестованого набору даних.

Таблиця 3.1 Порівняльна характеристика алгоритмів

Алгоритм	Ключові точки виявлені у наборі зображень		Зіставлені ознаки	Час виявлення та обчислення ознак (сек.)	Час зіставлення ознак (сек.)	Загальний час роботи (сек.)
	1-е фото	2-е фото				
SIFT	1546	53090	1051	1,7652	1,0934	2,9992
ORB	500	500	113	0,5036	0,0155	0,6442
BRISK	5593	46094	2083	0,8388	1,4260	2,4054

Як видно з результатів, алгоритм SIFT є найтривалішим за загальним часом виконання, а ORB став найшвидшим працюючим алгоритмом з результатом 0.6442 секунди. Цей час безпосередньо впливає на продуктивність програми. Проте у зіставленні ознак алгоритми SIFT та BRISK показали більший відсоток правильних збігів на противагу ORB, як показано на рисунку 3.3.

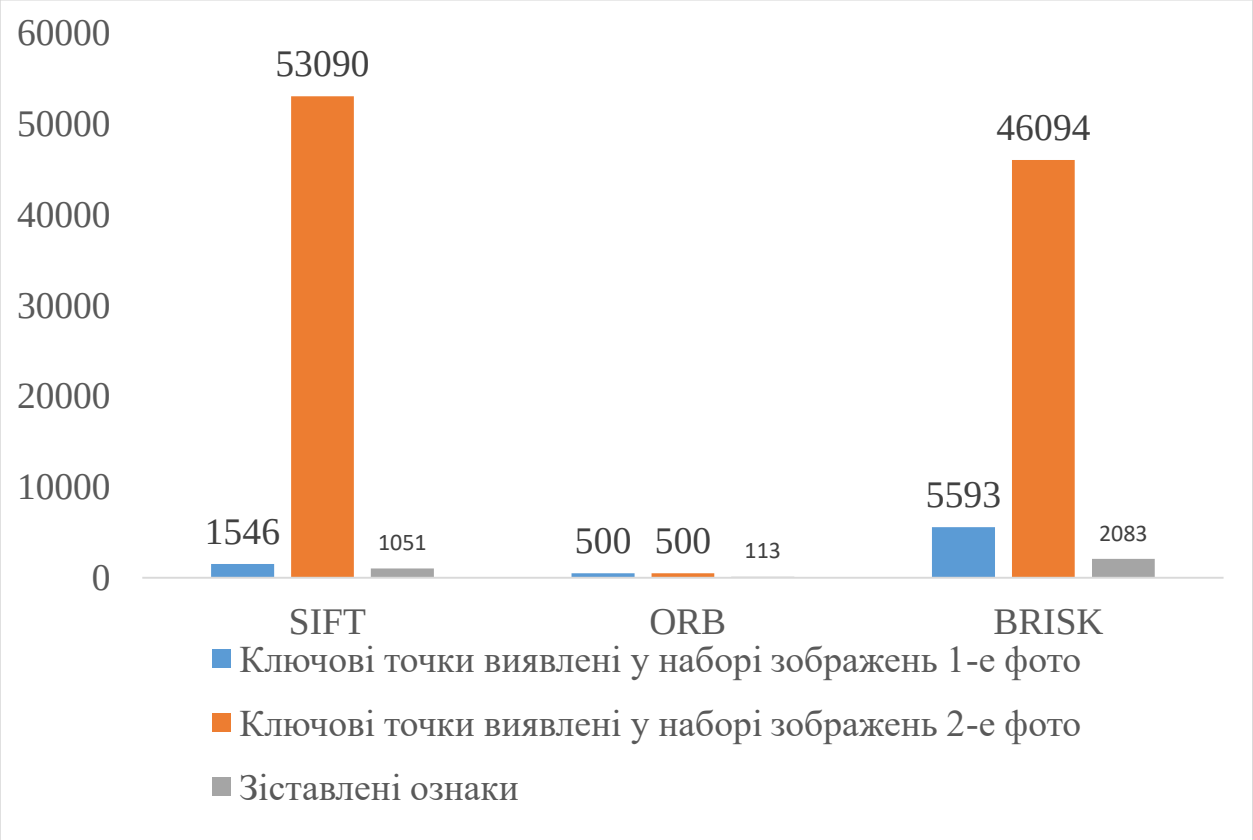


Рисунок 3.3 – Виявлені ключові точки та величина їх зіставлення між зображеннями.

Однією з головних причин, чому алгоритми SIFT та BRISK працюють повільніше, ніж алгоритм ORB, є час, необхідний для виявлення ключових моментів і ознак та їх зіставлення. Водночас BRISK займає 0.8388 секунди, алгоритм ORB виконує виявлення ключових точок у середньому за 0.5036 секунди, а для алгоритму SIFT тривалість складає 1.7652 секунди. Хоча, можна було б стверджувати, що останній є найгіршим алгоритмом по отриманих результатах, проте SIFT та BRISK мають конкурентоспроможні оцінки відповідності, оскільки BRISK, як показано на рисунку 3.4, має показники витраченого часу зіставлених ознак більші, ніж у SIFT. Тому саме за цим критерієм почесне друге місце дістається обом дескрипторам.

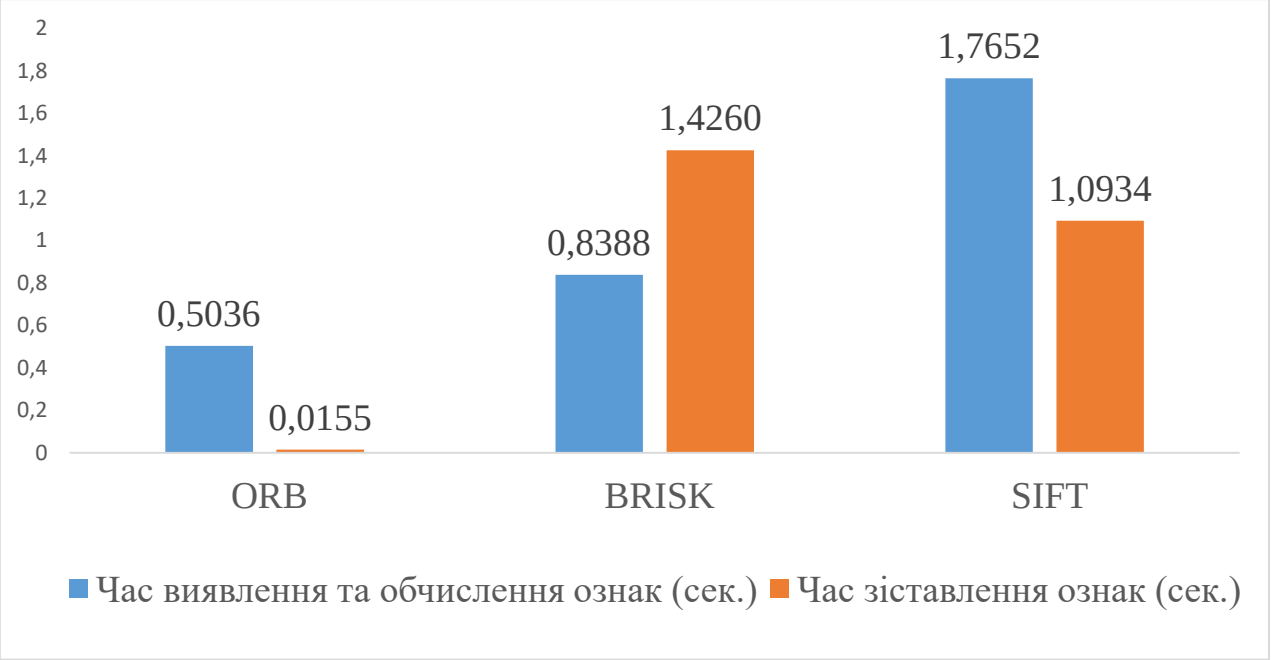


Рисунок 3.4 – Час, витрачений на знаходження ключових точок і їх зіставлення.

3.5 Висновки

З аналізу отриманих результатів можна виділити кілька відповідних спостережень, кожне з яких було обумовлено показниками, використаними в цьому дослідженні. Крім того, інтегруючи набір даних, можна отримати уявлення про різні фактори, які впливають на поведінку алгоритмів.

У результаті цієї роботи було помічено, що SIFT і BRISK виконали більш точні відповідності ознак, у протилежність ORB має кращі результати,

ніж інші два алгоритми з точки зору часу виконання процесів виявлення ознак та їх зіставлення. На додаток до того, що SIFT є повільним алгоритмом, таким як BRISK, він не має успіху у зіставленні ознак, але має найвище значення в критерії виділення ключових точок для другого зображення, що було знято в реальному світі.

Отже, якщо маємо програму, у якій обробляються зображення зроблені камерою чи взяті з реального часу – виявлення ознак для кожного зображення може призвести до уповільнення. Якщо бажана кількість ключових точок збігається на зображенні, ознака може не бути зіставлена знову на наступному зображенні, яке буде оброблено. У результаті ORB працює швидше, ніж інші два порівнювані алгоритми, але дає менший результат ознак, що збігаються в противагу SIFT та BRISK. Тому буде корисно обирати алгоритм з огляду на область застосування програми доповненої реальності, яка буде реалізована. Наприклад, якщо потрібно реалізувати мобільний додаток – час буде важливим критерієм, і ORB забезпечує хорошу продуктивність у цьому відношенні. Алгоритми SIFT або BRISK будуть більш доцільними у високопродуктивних програмах, де час не має критичного значення, а правильна відповідність функцій важливіша.

4 РОЗРОБКА ДОДАТКУ З ПІДТРИМКОЮ ДОПОВНЕНОЇ РЕАЛЬНОСТІ ТА МОЖЛИВІСТЮ РОЗПІЗНАВАННЯ ЗОБРАЖЕННЯ

4.1 Реалізація додатку в Unity3D

Ідея реалізації застосунку полягає в тому, щоб розробити додаток для мобільних пристроїв на базі OS Android з підтримкою доповненої реальності, який би дозволив розпізнати зображення за допомогою стандартної камери пристрою і запустив візуалізацію 3D моделі на смартфоні з метою застосування у навчальному процесі.

Для створення програми проведено аналіз усіх фреймворків, які призначені для розробки додатку з підтримкою доповненої реальності і найкращим ПЗ було обрано Vuforia SDK. Цей вибір спричинений тим, що програмне забезпечення Vuforia широко відоме як провідне в галузі AR за рахунок використаної в ньому кращої в своєму роді технології машинного зору, потужних функцій відстеження і широкої підтримки платформ.

У процес створення застосунку входила реєстрація на сайті Unity та сайті розробника Vuforia, завантаження і інсталяція Unity та розширення Vuforia для Unity разом з її імпортом. Згодом, було додано зображення-цілі (Image Targets), які через ліцензійний ключ (Рис. 3.1) були інтегровані до проекту.

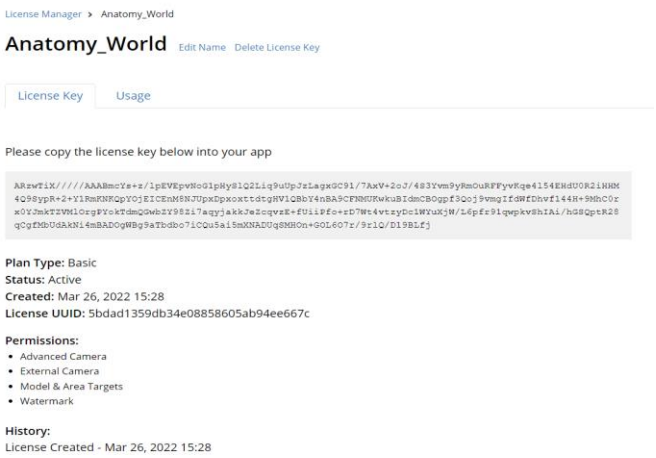


Рисунок 4.1 – Ліцензійний ключ зареєстрованого додатку

Наступний процес включає створення головного екрану з відкриттям камери.

Також крім системних методів представлених Vuforia було додано скрипти C# реалізовані для взаємодії і виконання всіх передбачених функцій:

- MenuScript – клас, що містить метод для зміни між сценами;
- EyeAnimation – клас, що містить методи для зміни обертання та масштабу 3D моделі будови ока;
- HeartAnimation – клас, що містить методи для зміни обертання та масштабу 3D моделі будови серця;
- KidneyAnimation – клас, що містить методи для зміни обертання та масштабу 3D моделі будови нирки;
- BrainAnimation – клас, що містить методи для зміни обертання та масштабу 3D моделі будови мозку;
- PanelOpener – клас, що містить методи для відкриття інформаційних панелей для кожної моделі відповідно;
- ButtonQ – кнопка для відкриття панелей з інформацією для кожної моделі, що містить дії на клік;
- ClickyButton – клас, що містить методи OnPointerDown() та OnPointerUp() для кліку кнопки зі зміною дизайну при натисканні;
- Rotate – кнопка для керування обертанням, розташована для зручності у правій панелі, що містить дії на клік для кожної моделі: EyeAnimation.AnimRotate(), HeartAnimation.AnimRotate(), BrainAnimation.AnimRotate(), KidneyAnimation.AnimRotate() та метод ClickyButton() з посиланням на зображеннями кнопки при звичайному стані та при кліку;
- Scale – кнопка для керування масштабом, розташована праворуч, що містить дії на клік для кожної моделі: EyeAnimation.AnimScale(), HeartAnimation.AnimScale(), BrainAnimation.AnimScale(),

KidneyAnimation.AnimScale() та метод ClickyButton() з посиланням на зображеннями кнопки при звичайному стані та при кліку;

- Back – кнопка для повернення масштабу після його збільшення, розташована у нижньому куті правої панелі, що містить дії на клік для кожної моделі: EyeAnimation.AnimBack(), HeartAnimation.AnimBack(), BrainAnimation.AnimBack(), KidneyAnimation.AnimBack() та метод ClickyButton() з посиланням на зображеннями кнопки при звичайному стані та при кліку;
- TrackableSettings(), MenuOptions(), OptionsConfig() – функціонал додаткового меню-кнопки на екрані з камерою;
- MenuOptions.ToggleAutofocus(), MenuOptions.ToggleTorch() – методи для кнопок додаткового меню автофокус і спалах камери відповідно.

4.2 Огляд результатів

Відстеження зображення здійснюється за допомогою камери для пошуку ознак потрібного зображення та оцінки їх відносної позиції перед камерою.

Алгоритм кроків для використання додатку:

1. Потрібно, щоб зображення, яке необхідно розпізнати було у полі зору камери, отже можемо роздрукувати його на будь-якій плоскій поверхні або відкрити на мобільному пристрої чи ноутбуці.

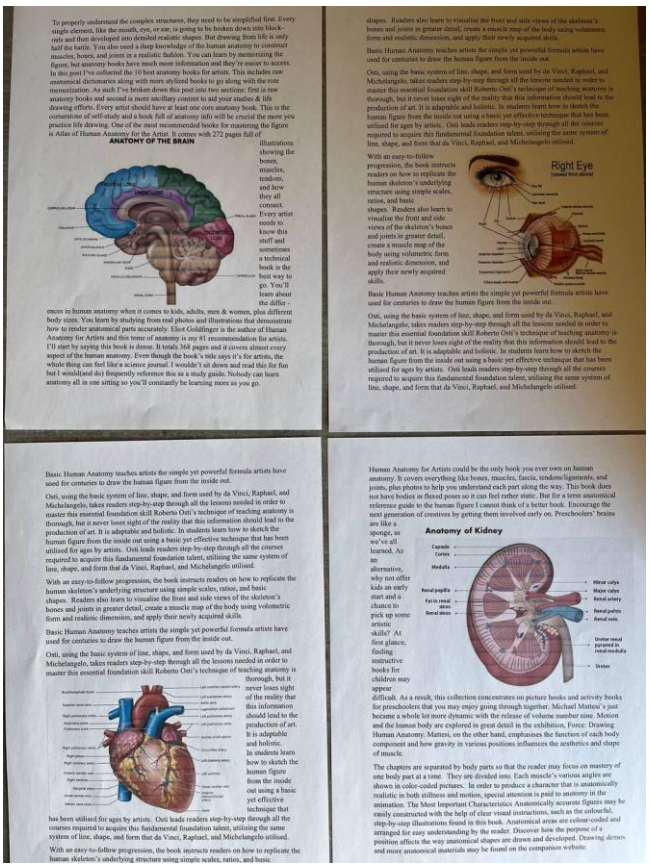


Рисунок 4.2 – Аркуші з чотирма Image Targets зображеннями

2. Відкриваємо додаток AR Anatomy, натиснувши на іконку додатку. Надається доступ до камери і відображається верхня панель з додатковими кнопками з п’ятьма функціями: відкриття інформаційної панелі над 3D моделлю, обертання, збільшення масштабу та повернення до вихідного розміру моделі, автофокус та спалах.

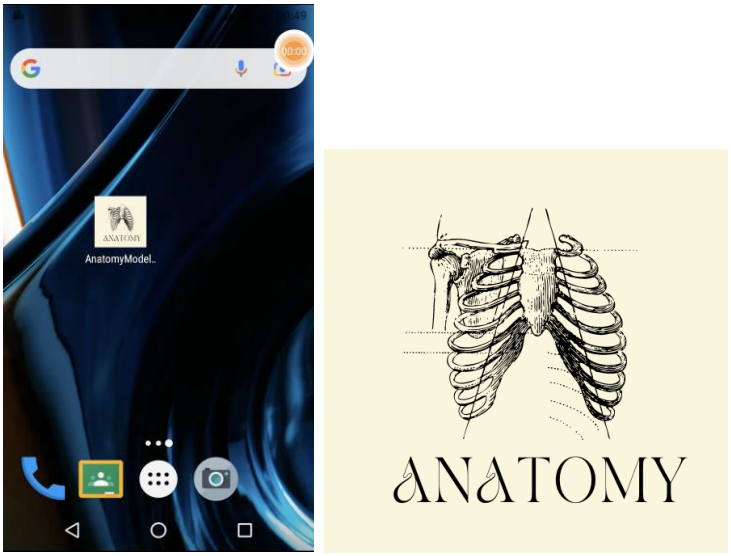


Рисунок 3.3 – Іконка додатку

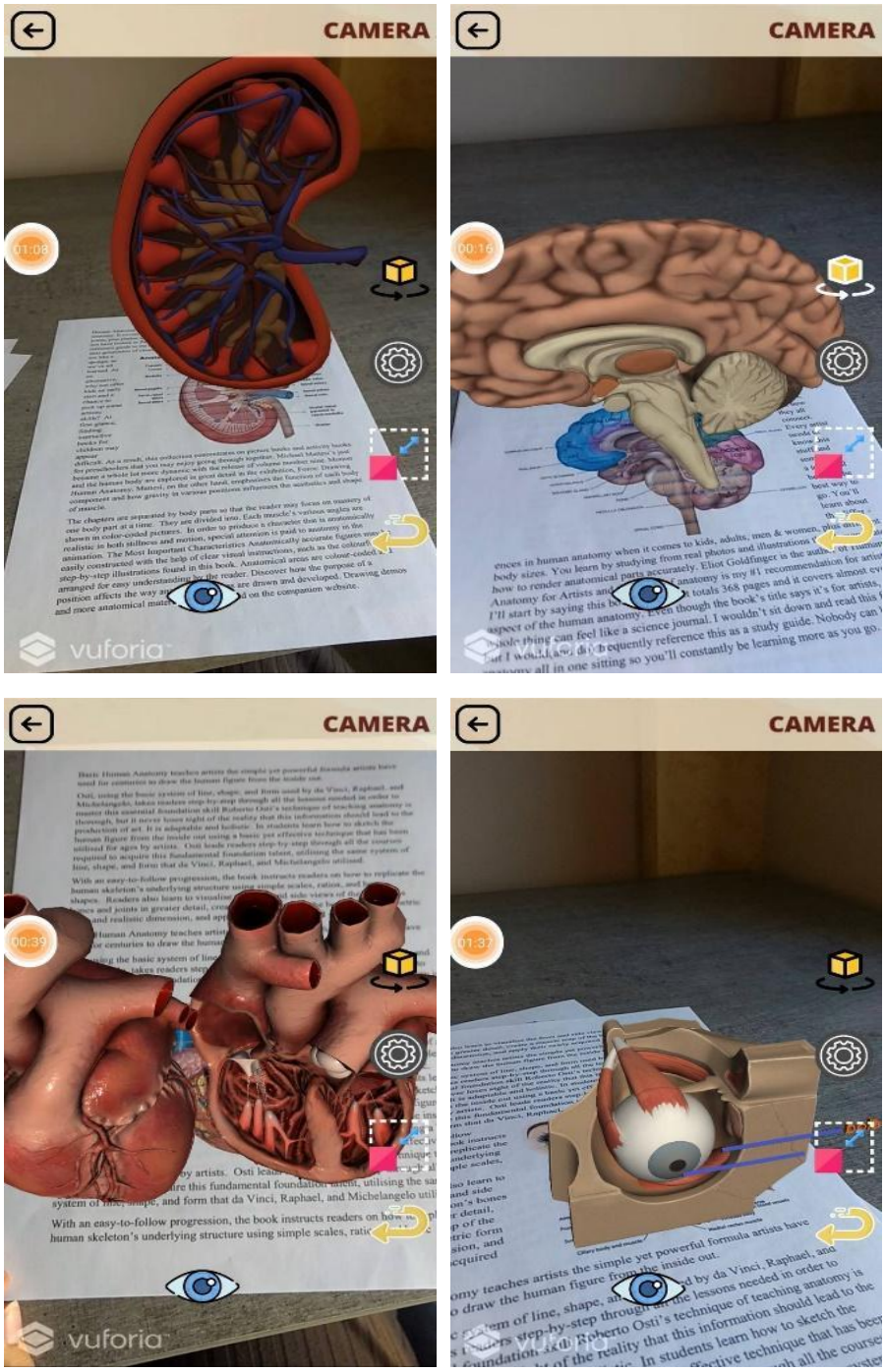


Рисунок 4.4 – Приклад розпізнавання зображень

3. Після натискання на центральну кнопку ButtonQ над моделлю, яка в цей момент розпізнається з’являється панель з короткою інформацією про об’єкт.

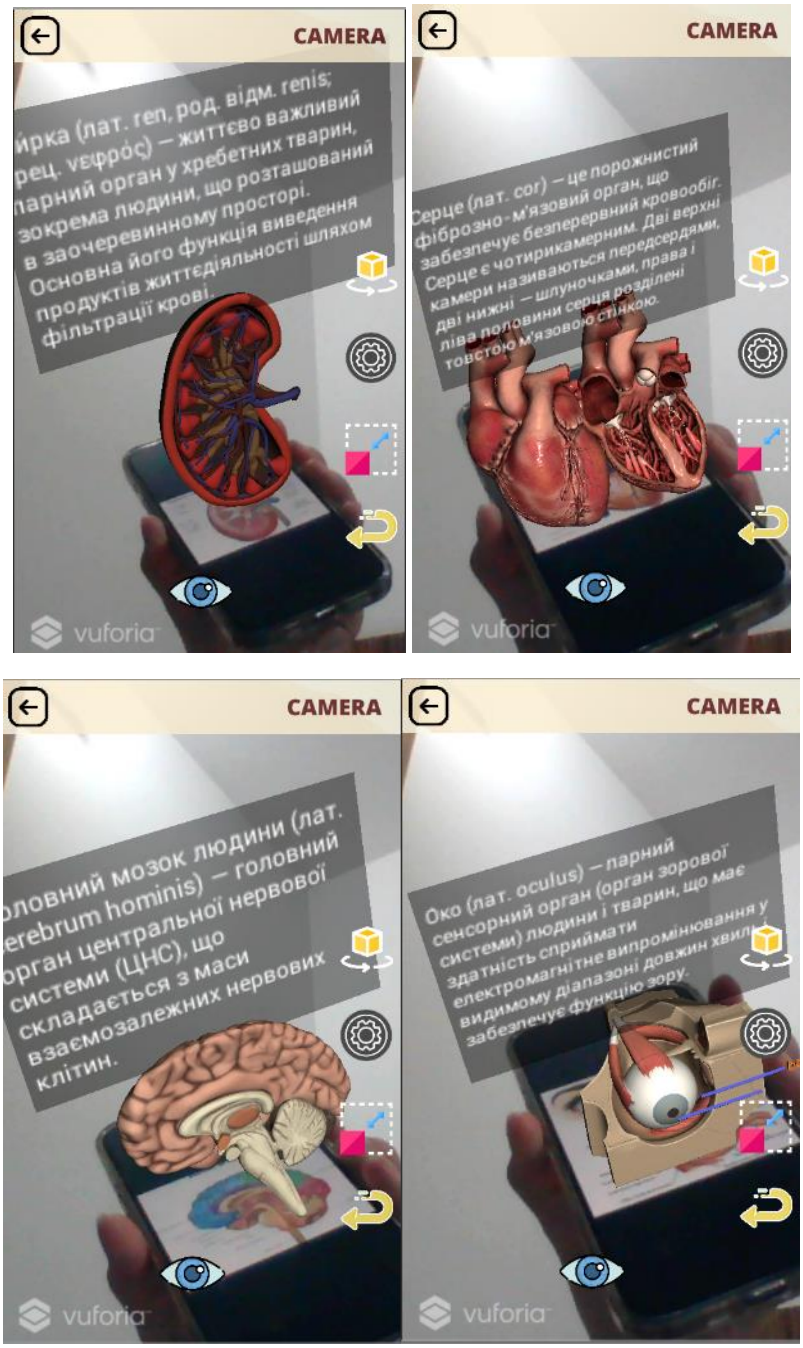


Рисунок 4.5 – Результат додавання інформаційної панелі

4. Після натискання на кнопку Rotate модель, яка в цей момент розпізнається робить оберт на 360 градусів.

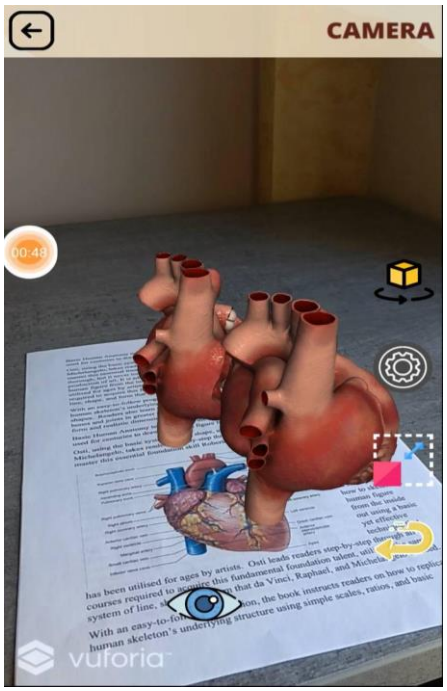


Рисунок 4.6 – Результат обертання 3D моделі за допомогою кнопки Rotate

5. Після кліку на кнопку Scale об’єкт збільшується в масштабі.

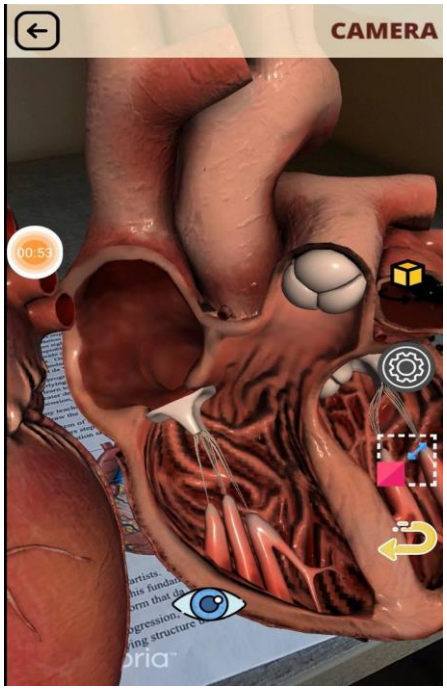


Рисунок 4.7 – Результат збільшення масштабу моделі за допомогою кнопки Scale

6. Після кліку на кнопку Back об’єкт повертається до вихідного розміру масштабу.

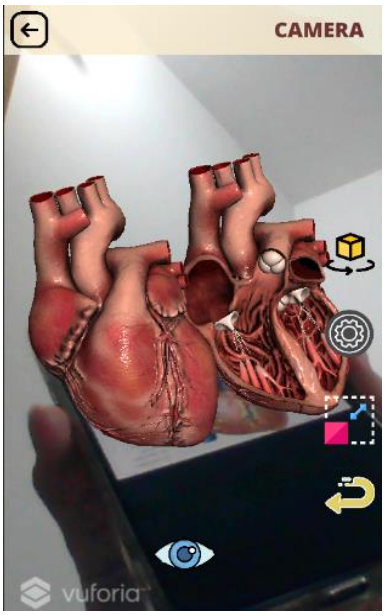


Рисунок 4.8 – Результат повернення вихідного розміру 3D моделі

7. Відкриття додаткового меню з автофокусом та спалахом.

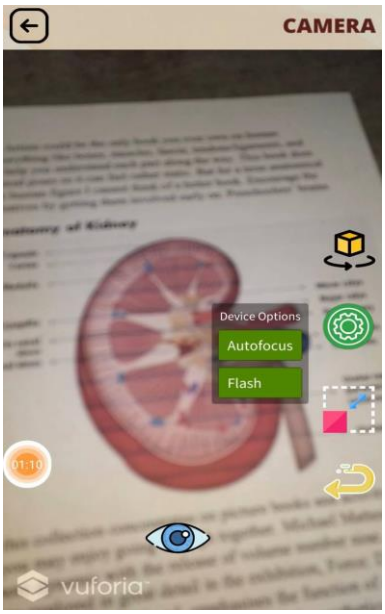


Рисунок 4.9 – Результат відкриття додаткового меню

ВИСНОВКИ

Завдяки доповненій реальності звичайний урок у класі чи аудиторії може перетворитись на захоплюючий досвід. Технологія AR пропонує віртуальні приклади та додає ігрові елементи до навчальних матеріалів, що в результаті дає більш інтерактивне залучення учнів на заняттях. Крім цього це допомагає студентам краще запам’ятовувати інформацію, яку вони вивчили чи побачити зсередини складні предмети, що зображенні лише на сторінках підручника.

У даній роботі було проведено деталізований огляд технології доповненої реальності та розроблено додаток, який демонструє 3D-моделі будови органів тіла і дозволяє студентам побачити коротку інформацію, про кожну з них, обертати та взаємодіяти з ними з метою залучення у навчальному процесі. Сам модуль доповненої реальності представляє собою додаткове середовище, відтворене у вигляді візуалізації 3D-моделей після розпізнавання зображення, яке розміщене в реальному часі на аркуші паперу перед користувачем. Інтерфейс також передбачає можливість обертання об’єктів, збільшення і зменшення їх розмірів, запуск аудіо запису, що містить коротку інформацію про модель та, безпосередньо, переміщення у будь-яке зручне місце використовуючи аркуш. Додаток був створений за допомогою фреймворку Vuforia в ігровому рушії Unity3D з додатковими скриптами мовою програмування C#.

Також було розглянуто метод визначення локальних ознак зображення SIFT, на якому базується модуль Vuforia Image Targets, і на основі якого відбувається процес розпізнавання зображень. Основною перевагою SIFT є достовірна ідентифікація об'єктів навіть серед шуму або часткового перекриття. Крім цього, завдяки достатній кількості кадрів на секунду додаток працює у режимі реального часу без затримок у відтворенні.

Додатково було проведено порівняльний аналіз алгоритмів SIFT, ORB та BRISK, які часто застосовуються у доповненій реальності.

На базі створеної програми у середовищі мови програмування Python та інтеграції бібліотеки OpenCV було отримано певні результати, які, щоб продемонструвати більш значущим чином, представлено у вигляді таблиці отриманих спостережень дослідження (Таб. 3.1). Зроблено графічне представлення критеріїв оцінки ефективності (Рис. 3.3 – Рис. 3.4), показуючи кількісну оцінку для кожного алгоритму.

З аналізу дослідження алгоритмів було помічено, що SIFT і BRISK виконали більш точне зіставлення ознак, тоді як ORB перевершив два інших алгоритми з точки зору часу зіставлення. SIFT, окрім того, що такий ж повільний алгоритм, як BRISK, має найвище значення в критеріях виділення ключових точок для другого зображення, зробленого в реальному світі, проте не був успішним у зіставленні цих ознак.

Тому можна прийти до висновку, що якщо користувачу потрібна програма, яка працює швидко, тоді ORB найкраще забезпечить цю властивість. Проте, якщо важливим фактором є точність знаходження відповідних ключових точок та їх зіставлення – хорошим вибором буде SIFT або BRISK, які показали близькі результати продуктивності.

Отже, технологія AR може бути використана, як новий інструмент в освіті. Тож замість того, щоб звично читати книги чи слухати лекції, студенти можуть почати займатися за допомогою мобільних пристроїв, яке мають широке застосування у наш час. Це збільшує зацікавленість та покращує досвід навчання.

На мою думку, як тільки AR знайде переконливу, доступну та повно функціональну платформу і чим більшою стане кількість споживачів, які стануть фахівцями AR, її потенціал почне реалізовуватися повністю і зможе створити новий світ майбутнього.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Unity (рушій гри) [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Unity_(%D1%80%D1%83%D1%88%D1%96%D0%B9_%D0%B3%D1%80%D0%B8)

2. Hu Tianyu et al. Overview of augmented reality technology. Computer Knowledge and Technology / Hu Tianyu et al, 2017. – 194 -196 с.

3. Jens Grubert. Augmented Reality for Android Application Development / Jens Grubert, Dr. Raphael Grasset, 2013.

4. Dieber Schmalstoeg. Augmented Reality: Principles and Practice / Dieber Schmalstoeg, Tobias Hollerer, 2016.

5. Jens Grubert. Augmented Reality for Android Application Development / Jens Grubert, Dr. Raphael Grasset, 2013. – 10 с.

6. Alan B. Craig, Understanding Augmented Reality: Concepts and Applications / Alan B. Craig. – Kindle Edition, 2013.

7. Google Expeditions AR [Електронний ресурс] – Режим доступу: https://edu.google.com/products/vr-ar/expeditions/

8. NASA, Microsoft Collaborate to Bring Science Fiction to Science Fact [Електронний ресурс] – Режим доступу: https://www.nasa.gov/press-release/nasa-microsoft-collaborate-to-bring-science-fiction-to-science-fact

9. Michele Gattullo, Marker Based vs. Natural Features Tracking Augmented Reality Visualization of the 3D Foot Phantom. / Michele Gattullo, Ernesto Carrabba, Goran Devedzic, Sasa Cukovic. – Conference Paper, 2015. – 6-8 с.

10. David G. Lowe. Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision / David G. Lowe, 2004. – 91-110 с.

11. Saurabh Kapur. Computer Vision with Python 3 / Saurabh Kapur, 2017. – 137 с.

12. David Lowe, Invariant Features from Interest Point Groups / Matthew Brown, David Lowe. – Department of Computer Science, 2002.

13. Harris C. A combined corner and edge detector, Fourth Alvey Vision Conference / Harris C. and Stephens M., 1988. – 147-151 с.

14. Best Practices for Designing and Developing Image-Based Targets [Електронний ресурс] – Режим доступу: https://library.vuforia.com/features/images/image-targets/best-practices-for-designing-and-developing-image-based-targets.html

15. Target Manager [Електронний ресурс] – Режим доступу: https://developer.vuforia.com/target-manager

16. Zitová. Image registration methods [Електронний ресурс]: a survey. Image and Vision Computing / Zitová & J. Flusser, 2003. – 1 с. – Режим доступу: http://library.utia.cas.cz/prace/20030125.pdf

17. Pose Tracking from Natural Features on Mobile Phones, Proceedings of 7th IEEE and ACM International Symposium on Mixed and Augmented Reality [авт. кол. : Wagner D., Reitmayr G., Mulloni A., Drummond T., and Schmalstieg D.]. – 2008. – 125-134 с.

18. Wagner D., Schmalstieg D., and Bischof H., Multiple Target Detection and Tracking with Guaranteed Framerates on Mobile Phones, International Symposium on Mixed and Augmented Reality, 2009. – 57-64 с.

19. ORB An Efficient Alternative to SIFT or SURF, Proceedings of 7th IEEE International Conference on Computer Vision [авт. кол. : Rublee E., Rabaud V., Konolige K., and Bradski G.]. – 2011. – 2564-2571 с.

20. Rosten E. Faster and better: A machine learning approach to corner detection, IEEE T. Pattern Anal. / Rosten E., Porter R., and Drummond T., 2010. – 105–119 с.

21. Calonder M. Brief: Binary Robust Independent Elementary Features, European Conference on Computer Vision, Heraklion / Calonder M, Lepetit V, Strecha C, and Fua P., 2010. – 778-792 с.

22. Suleiman. An Energy-Efficient Hardware Implementation of HOG-Based Object Detection at 1080HD 60 fps with Multi-Scale Support. Journal of Signal Processing Systems / Suleiman, & V. Sze, 2016. – 325–337 с. [Електронний ресурс] – Режим доступу: https://doi.org/10.1007/s11265-015-1080-7

23. P. L. Rosin. Measuring Corner Properties. Comp. Vis. and Image Understanding / P. L. Rosin., 1999. – 291-307 с.

24. Leutenegger. BRISK: Binary Robust Invariant Scalable Keypoints. ICCV/ Leutenegger, S., Chli, M., Sieg-wart, R., 2011. – 2548-2555 с.

25. Mair. E. Adaptive and Generic Corner Detection Based on the Accelerated Segment Test / Mair. E., 2010. – 183-196 с.

ДОДАТКИ

Додаток А

MenuScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MenuScript : MonoBehaviour
{
   public void ChangeScene(string sceneName)
   {
               Application.LoadLevel(sceneName);
   }
}
```

EyeAnimation.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EyeAnimation : MonoBehaviour
{
    private Animator Anim;

    void Start()
    {
        Anim = GetComponent<Animator>();
        Anim.speed = 0f;
    }

    void Update()
    {

    }

    public void AnimScale(){
        Anim.Play("ScaleEye", -1, 0f);
        Anim.speed = 1f;
    }

    public void AnimRotate(){
        Anim.Play("RotateEye", -1, 0f);
        Anim.speed = 0.2f;
    }

    public void AnimBack(){
        Anim.Play("EyeBack", -1, 0f);
        Anim.speed = 1f;
    }
}
```

HeartAnimation.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HeartAnimation : MonoBehaviour
{
    private Animator Anim;
```

```
    // Start is called before the first frame update
    void Start()
    {
        Anim = GetComponent<Animator>();
        Anim.speed = 0f;
    }

    // Update is called once per frame
    void Update()
    {

    }

    public void AnimScale(){
        Anim.Play("ScaleHeart", -1, 0f);
        Anim.speed = 1f;
    }

    public void AnimRotate(){
        Anim.Play("RotateHeart", -1, 0f);
        Anim.speed = 0.2f;
    }

    public void AnimBack(){
        Anim.Play("HeartBack", -1, 0f);
        Anim.speed = 1f;
    }
}
```

KidneyAnimation.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class KidneyAnimation : MonoBehaviour
{
   private Animator Anim;

    void Start()
    {
        Anim = GetComponent<Animator>();
        Anim.speed = 0f;
    }
    void Update()
    {

    }

    public void AnimRotate(){
        Anim.Play("RotateKidney", -1, 0f);
        Anim.speed = 0.2f;
    }
}
```

BrainAnimation.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BrainAnimation : MonoBehaviour
{
    private Animator Anim;
```

```
    void Start()
    {
        Anim = GetComponent<Animator>();
        Anim.speed = 0f;
    }

    void Update()
    {

    }

    public void AnimScale(){
        Anim.Play("ScaleBrain", -1, 0f);
        Anim.speed = 1f;
    }

    public void AnimRotate(){
        Anim.Play("RotateBrain", -1, 0f);
        Anim.speed = 0.2f;
    }

    public void AnimBack(){
        Anim.Play("BrainBack", -1, 0f);
        Anim.speed = 1f;
    }
}
```

PanelOpener.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PanelOpener : MonoBehaviour
{
    public GameObject PanelInfo;
    public GameObject PanelInfo2;
    public GameObject PanelInfo3;
    public GameObject PanelInfo4;

    public void OpenPanel()
    {
        if(PanelInfo != null){
            bool isActive = PanelInfo.activeSelf;
            PanelInfo.SetActive(!isActive);
        }
    }

    public void OpenPanel2()
    {
        if(PanelInfo2 != null){
            bool isActive = PanelInfo2.activeSelf;
            PanelInfo2.SetActive(!isActive);
        }
    }

    public void OpenPanel3()
    {
        if(PanelInfo3 != null){
            bool isActive = PanelInfo3.activeSelf;
            PanelInfo3.SetActive(!isActive);
        }
    }

    public void OpenPanel4()
```

```
    {
        if(PanelInfo4 != null){
            bool isActive = PanelInfo4.activeSelf;
            PanelInfo4.SetActive(!isActive);
        }
    }

}
```

ClickyButton.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;

public class ClickyButton : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    [SerializeField] private Image _img;
    [SerializeField] private Sprite _default, _pressed;

    public void OnPointerDown(PointerEventData eventData) {
        _img.sprite = _pressed;
    }


    public void OnPointerUp(PointerEventData eventData) {
        _img.sprite = _default;
    }
}
```

Додаток Б

BRISK.py

```
import cv2
from matplotlib import pyplot as plt
import time

start = time.time()

image1 = cv2.imread('eye image scaled.jpg')
image2 = cv2.imread('eye_realsize.jpg')

# Запустіть детектор BRISK
brisk = cv2.BRISK create()

# Перетворення на сірий
image1 = cv2.cvtColor(image1,cv2.COLOR_BGR2GRAY)
image2 = cv2.cvtColor(image2,cv2.COLOR_BGR2GRAY)

startDetect = time.time()

# знайдіть ключові точки та дескриптори за допомогою brisk
keypoints1, descriptor1 = brisk.detectAndCompute(image1,None)
keypoints2, descriptor2 = brisk.detectAndCompute(image2,None)

endDetect = time.time()
total_timeDetect = endDetect - startDetect
print("Виявлення та обчислення час (секунди): ", str(total timeDetect))

startMatching = time.time()

# BFMatcher з параметрами
bf = cv2.BFMatcher(normType = cv2.NORM_HAMMING,
                           crossCheck = True)

matches = bf.match(queryDescriptors = descriptor1,
                            trainDescriptors = descriptor2)

# Розсортуйте їх у порядку їх відстані
matches = sorted(matches, key = lambda x: x.distance)

endMatching = time.time()
total_timeMatching = endMatching - startMatching
print("Зіставлення час (секунди): ", str(total_timeMatching))

end = time.time()
total_time = end - start
print("Загальний час (секунди): ", str(total_time))

print("Кількість ключових точок: (оригінал) ", len(descriptor1))
print("Кількість ключових точок: (реальний світ) ", len(descriptor2))

print("matches ", len(matches))

result =
cv2.drawMatches(image1,keypoints1,image2,keypoints2,matches[:300],None,flags
= 2)
plt.imshow(result),plt.show()
```

SIFT.py

```
import cv2
from matplotlib import pyplot as plt
import time

start = time.time()

image1 = cv2.imread('eye_image_scaled.jpg')
image2 = cv2.imread('eye realsize.jpg')

# Запускаємо детектор SIFT
sift = cv2.xfeatures2d.SIFT_create()

# Конвертуємо зображення у CIPE
image1 = cv2.cvtColor(image1,cv2.COLOR_BGR2GRAY)
image2 = cv2.cvtColor(image2,cv2.COLOR_BGR2GRAY)

startDetect = time.time()

# знаходимо ключові точки та дескриптори за допомогою SIFT
keypoints1, des1 = sift.detectAndCompute(image1,None)
keypoints2, des2 = sift.detectAndCompute(image2,None)

endDetect = time.time()
total_timeDetect = endDetect - startDetect
print("Виявлення та обчислення час (секунди): ", str(total_timeDetect))

startMatching = time.time()

# BFMatcher (Brute-force descriptor matcher) із параметрами за замовчуванням
bf = cv2.BFMatcher(normType = cv2.NORM_L2,
                   crossCheck = True)

matches = bf.match(queryDescriptors = des1,
                           trainDescriptors = des2)

# Сортуємо їх у порядку їх відстані
matches = sorted(matches, key = lambda x: x.distance)

endMatching = time.time()
total timeMatching = endMatching - startMatching
print("Зіставлення час (секунди): ", str(total_timeMatching))

end = time.time()
total time = end - start
print("Загальний час (секунди): ", str(total_time))

print("Кількість ключових точок: (оригінал) ", len(des1))
print("Кількість ключових точок: (реальний світ) ", len(des2))

print("matches ", len(matches))

result =
cv2.drawMatches(image1,keypoints1,image2,keypoints2,matches[:300],None,flags
= 2)
plt.imshow(result),plt.show()
```

ORB.py

```
import cv2
from matplotlib import pyplot as plt
import time

start = time.time()

image1 = cv2.imread('eye image scaled.jpg')
image2 = cv2.imread('eye_realsize.jpg')

# Запускаємо детектор ORB
orb = cv2.ORB create()

# Конвертуємо зображення у CIPE
image1= cv2.cvtColor(image1,cv2.COLOR_BGR2GRAY)
image2= cv2.cvtColor(image2,cv2.COLOR BGR2GRAY)

startDetect = time.time()

# знаходимо ключові точки та дескриптори за допомогою ORB
keypoints1, des1 = orb.detectAndCompute(image1,None)
keypoints2, des2 = orb.detectAndCompute(image2,None)

endDetect = time.time()
total_timeDetect = endDetect - startDetect
print("Виявлення та обчислення час (секунди): ", str(total timeDetect))

startMatching = time.time()

# Створення Brute Force Matcher об'єкту
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck = True)

# Виконуємо зіставлення між дескрипторами ORB зображень
matches = bf.match(queryDescriptors = des1,
                           trainDescriptors = des2)

# The matches with shorter distance are the ones we want.
matches = sorted(matches, key = lambda x : x.distance)

endMatching = time.time()
total_timeMatching = endMatching - startMatching
print("Зіставлення час (секунди): ", str(total_timeMatching))

end = time.time()
total_time = end - start
print("Загальний час (секунди): ", str(total_time))

print("Кількість ключових точок: (оригінал) ", len(des1))
print("Кількість ключових точок: (реальний світ) ", len(des2))

# Вивід загальної кількості точок збігу між зображеннями навчання та запиту
print("matches ", len(matches))


result =
cv2.drawMatches(image1,keypoints1,image2,keypoints2,matches[:50],None,flags
= 2)
plt.imshow(result),plt.show()
```