Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук.
(повна назва)

Кафедра Програмної інженерії.
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів розпізнавання маркерів у технології Augmented Reality

(тема)

Виконав:

Випускник 2 курсу, групи ІПЗм-19-2

Косик Д.О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми Освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Керівник доц. Валенда Н.А.
(посада, прізвище)

Допускається до захисту

Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
« 26 » березня 2021 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Косика Дмитра Олександровича
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів розпізнавання маркерів у технології Augmented Reality»

затверджена наказом університету від 26.03.2021 № 385 Ст

2. Термін подання студентом роботи до екзаменаційної комісії «11» травня 2021 р.

3. Вихідні дані до роботи *алгоритми розпізнавання маркерів, доповнена реальність*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз предметної галузі і постановка задачі, алгоритми розпізнавання маркерів, бібліотеки та інструменти розробників для створення додатків з AR технологією*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, ілюстрацій (слайдів) *мета завдання, обґрунтування доцільності розроблення, постановка задачі, методи і алгоритми, опис отриманих результатів, інтерфейс програмної системи, демонстраційні матеріали*

6 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Валенда Н.А.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	25.01.2021 – 08.02.2021	*Виконано*
2	Огляд існуючих методів та алгоритмів	08.02.2021 – 20.02.2021	*Виконано*
3	Дослідження алгоритмів розпізнавання маркерів	20.02.2021 – 20.03.2021	*Виконано*
4	Спецчастина	20.03.2021 – 10.04.2021	*Виконано*
5	Підготовка пояснювальної записки	10.04.2021 – 01.05.2021	*Виконано*
6	Підготовка презентації та доповіді	01.05.2021 – 04.05.2021	*Виконано*
7	Нормоконтроль, рецензування	04.05.2021 – 11.05.2021	*Виконано*
8	Попередній захист	12.05.2021	*Виконано*
9	Занесення диплома в електронний архів	12.05.2021	*Виконано*
10	Допуск до захисту у зав. кафедри	13.05.2021	*Виконано*

Дата видачі завдання 25 січня 2021р.

Студент Косик Дмитро Олександрович
(підпис)

Керівник роботи _____ доц. Валенда Н.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до передатестаційної роботи магістра містить: 75 с., 40 рис., 17 дж.

ВИЛУЧЕННЯ ІНФОРМАЦІЇ, ДОПОВНЕНА РЕАЛЬНІСТЬ, ДОСЛІДЖЕННЯ МЕТОДІВ МАРКУВАННЯ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, РОЗПІЗНАВАННЯ МАРКЕРІВ, JSARTOOLKIT, VUFORIA, OPENCV.

Об’єктом дослідження є задачі розпізнавання маркерів у технології Augmented Reality.

Метою роботи є аналіз існуючих методів розпізнавання маркерів у технології Augmented Reality.

Методи розробки базуються на таких технологіях, як JSARToolKit, Vuforia, Metaio SDK, OpenCV.

Результатом роботи є дослідження методів розпізнавання маркерів в технології доповненої реальності, проведення аналіз та моделювання предметної області, проведення дослідження існуючих засобів та інструментів.

EXTRACTION OF INFORMATION, AUGMENTED REALITY, RESEARCH OF MARKING METHODS, INTELLECTUAL ANALYSIS OF DATA, RECOGNITION, MARKING, JSARTOOLKIT, VUFORIA, OPENCV.

The object of research is the task of recognizing marketing managers in augmented reality technologies.

The aim of the work is to analyze the existing methods of recognition of marketing managers in augmented reality technologies.

Development methods are based on such technologies as JSARToolKit, Vuforia, Metaio SDK, OpenCV.

The result of the work is a study of methods for recognizing markers in augmented reality technology, analysis and modeling of the subject area, a study of existing tools and instruments.

Я, *Косик Дмитро Олександрович,* студент групи ІПЗм-19-2, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів розпізнавання маркерів у технології Augmented Reality», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

Вступ 7
1 Аналіз предметної області і постановка завдання 9
1.1 Аналіз предметної галузі.....9
1.2 Що може бути маркером та як ідентифікувати об’єкт 12
1.3 Важливість маркерів в доповненій реальності 13
1.4 Техніка дослідження алгоритму розпізнавання маркерів.....14
1.5 Постановка задачі.....20
2 Дослідження методів розпізнавання маркерів.....21
2.1 Методи розпізнавання маркерів 21
2.2 Узагальнений алгоритм розпізнавання маркера.....21
2.3 Виявлення на основі ознак.....30
2.4 Виявлення країв 33
2.5 Впровадження та результати 34
2.6 Розпізнавання маркера на основі краю.....34
2.7 Оклюзії 36
2.8 Розпізнавання маркера SURF-Golay 37
2.8 Адаптивне перемикання.....40
3 Огляд бібліотек для розробки додатків з використанням AR технології 42
3.1 Бібліотеки компютерного зору та доповненої реальності 42
3.2 OpenCV бібліотека для комп’ютерного зору, з відкритим кодом 45
3.3 JSARToolKit 48
4. Розробка прототипу 50
4.1 Програмна реалізація додатку з використання JSARToolKit.....50
4.2 Архітектура додатку 50
4.3 Розробка додатку 51
Висновки 57

Перелік джерел посилання.....59
Додаток А. Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії.....61
Додаток Б. Звіт результатів перевірки на унікальність тексту 62
Додаток В. Наукові публікації 63
Додаток Г. Слайди презентації 74
Додаток Д. Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015 ...75

ВСТУП

Доповнена реальність (AR) – це сучасна технологія, за допомогою якої можна розширити реальний світ за допомогою мобільного пристрою, наповнюючи його цифровим вмістом, ця технологія базується на реальному оточенні і додає до нього додаткові комп'ютерні дані, такі як різні зображення, анімація або тривимірні об'єкти, на відміну від віртуальної реальності [1], [2].

Зі зростанням числа розумних мобільних пристроїв і постійним поліпшенням їх технічних можливостей, доповнена реальність проникла у деякі аспекти людського життя.

Щоб накласти цифрову інформацію, AR-додатки використовують такі методи:

- SLAM (simultaneous localization and mapping) – це набір алгоритмів, що використовують дані різних сенсорів для побудови мапи у невідомому просторі та визначення положення пристрою;
- markers – метод, що базується на розпізнаванні маркерів;
- position – метод, що базується на положенні мобільного пристрою у просторі. Використовує GPS, компас та інші сенсори.

Кожен з наведених методів має свої переваги та недоліки. Метод працюючий з маркерами базується на розпізнаванні зображення та предметів на ньому, тому його доцільно використовувати у рекламі або торгівлі. Методи SLAM та Position частково схожі між собою, адже обидва використовують сенсори для визначення положення пристрою. Position є більш простим та часто використовується у іграх чи картах. Метод SLAM є більш складним тому що окрім даних з сенсорів він також аналізує зображення з камери [3]. Цей метод поширений у мобільних додатках, яким необхідно вміти визначати поверхні.

Метою роботи є аналіз та пошук найбільш ефективного методу розпізнавання маркерів в технології доповненої реальності серед існуючих методів.

Задачами дослідження є проведення експериментів спрямованих на виявлення алгоритму з найбільшими показниками точності результату, а також розробка веб-застосунку та порівняння його з іншим мобільним застосунком, для виявлення кращого методу розробки та більш ефективних технологій.

Об’єктом дослідження є аналіз методів та алгоритмів розпізнавання маркерів.

Предметом дослідження є методи розпізнавання маркерів у технології доповненої реальності, такі як:

- узагальнений алгоритм розпізнавання маркера;
- виявлення маркера на основі ознак;
- виявлення маркера на основі краю.

А також бібліотеки та інструменти розробників, за допомогою яких створюється додатки доповненої реальності, такі як:

- AForge.Net;
- Metaio SDK;
- Vuforia SDK;
- OpenCV;
- JSARToolkit.

Одна з ідей роботи, поширення AR технології в сфері культури опублікована у збірнику тез з VIII Міжнародної науково-практичної конференції у м. Київ, Україна [4].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз предметної галузі

Створення додатків, які добре працюють з доповненою реальністю, вимагає певних знань про алгоритми обробки зображень, які працюють за кадром. Одне з найбільш фундаментальних понять стосується маркерів. Вони покладаються на ключові точки та їх дескриптори, виявлені в записах реального світу.

Додатки, які розроблені з використанням AR технології приховують значну частину складності. Розробник просто прив’язує віртуальні об’єкти до світу. Це гарантує, що голограма залишається приклеєною до фізичного місця, куди її поклали.

Об’єкти залишаються на місці, навіть якщо система з часом дізнається більше про навколишнє середовище. Наприклад, розробник розміщує предмет на стіні за 2 метри від нього. Далі він підходить ближче до стіни. Датчики отримали більш точні вимірювання відстаней і з’ясували, що стіна насправді була б на відстані 2,1 метра від початкового положення, а не лише 2,0 метра. Двигун повинен відображати ці нові знання в положенні 3D-об'єкта.

У традиційній єдиній світовій системі координат кожен об’єкт має фіксовані координати x, y, z. Маркери замінюють положення та обертання компонента перетворення, прикріпленого до 3D-об'єкта. Сприйманий реальний світ має пріоритет над статичною системою координат.

Але як же віртуальні предмети, прив’язуються до реальних позицій та дотримуються їх? Це відбувається за допомогою маркерів [5].

Документація Google ARCore показує, що маркерні основи базуються на типі, який називається "відстежуваними" – виявленими точками об'єктів та площинами. Оскільки площини згруповані точковими елементами, які мають однакову площинну поверхню, все зводиться до точок об’єктів.

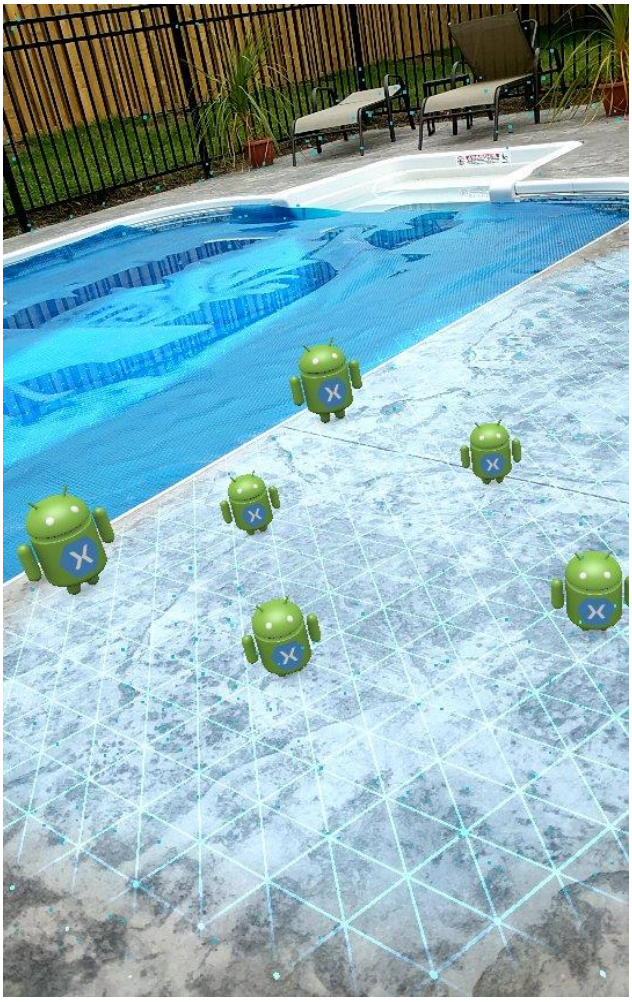

Рисунок 1.1 - Відображення маркерів на реальних об’єктах

На зображенні показано особливості, виявлені Google ARCore. Вони позначені як бірюзові крапки та лінії. Як можна побачити, ARCore виявляє велику кількість крапок на бетонній підлозі, так як вона має дуже помітну текстуру, також він виявляє деяку кількість крапок на об’єктах які знаходяться на задньому плані і являються не так сильно помітними. Однак на воді не виявлено жодної особливості, так як вона є дуже гладкою і складно виявити на ній текстуру.

В комп’ютерному баченні є так звані основні точки, це відмінне розташування на зображеннях - наприклад, кути, краплі або Т-образні переходи зазвичай є хорошими кандидатами. Однієї цієї інформації недостатньо для того, щоб відрізнити одне від іншого або для надійного пошуку точок об’єкта знову. Тому також сусідство аналізується та зберігається як дескриптор.

Найважливішою властивістю основних точок є надійність. Алгоритм повинен мати можливість знаходити одну і ту ж фізичну точку інтересу за різних умов перегляду. Багато чого може змінитися під час використання програми AR:

- кут / перспектива камери;
- обертання;
- масштаб;
- блискавка;
- розмиття від руху або фокусування;
- загальний шум зображення.

Для того щоб знайти надійні функціональні точки, існують різні фреймворки AR, які знаходять ці особливості. Наприклад Microsoft HoloLens працює з великою кількістю даних датчиків. Особливо цінна інформація про глибину на основі відбитого структурованого інфрачервоного світла. Мобільний AR, як Google ARCore та Apple ARKit, може працювати лише з 2D кольоровою камерою [3].

Пошук відмінних рис на зображеннях є активним напрямом досліджень протягом тривалого часу. Один із найвпливовіших алгоритмів називається "SIFT" (Scale Invariant Feature Transform) - масштабувати інваріантне перетворення об’єкта. Він був розроблений Девідом Г. Лоу та опублікований у 2004 році. Інший "традиційний" метод називається "SURF" (Speeded up robust features) – прискорення надійних функцій? Х. Бей та інші. Обидва вони використовуються і сьогодні. Однак обидва алгоритми запатентовані і зазвичай занадто повільні для використання в реальному часі на мобільних пристроях [6] [7].

Невідомо, якими алгоритмами користуються Windows Mixed Reality, ARCore та ARKit. Однак існує кілька надійних і безпатентних алгоритмів.

Хорошим кандидатом є алгоритм "BRISK" (Binary Robust Invariant Scalable Keypoints) - двійкові міцні інваріантні масштабовані ключові точки. Це досить швидко та ефективно, щоб послужити основою загального підходу "SLAM" одночасного визначення положення камери та відображення реального світу.

1.2 Що може бути маркером та як ідентифікувати об’єкт

Припустимо, є зображення і необхідно ідентифікувати об’єкт або маркер, неважливо, головне щоб було, відносно чого доповнювати реальність. Це можна зробити як з використанням маркеру так і без нього. Наприклад будь який штрих-код, напис або QR-код, це вже є маркер.

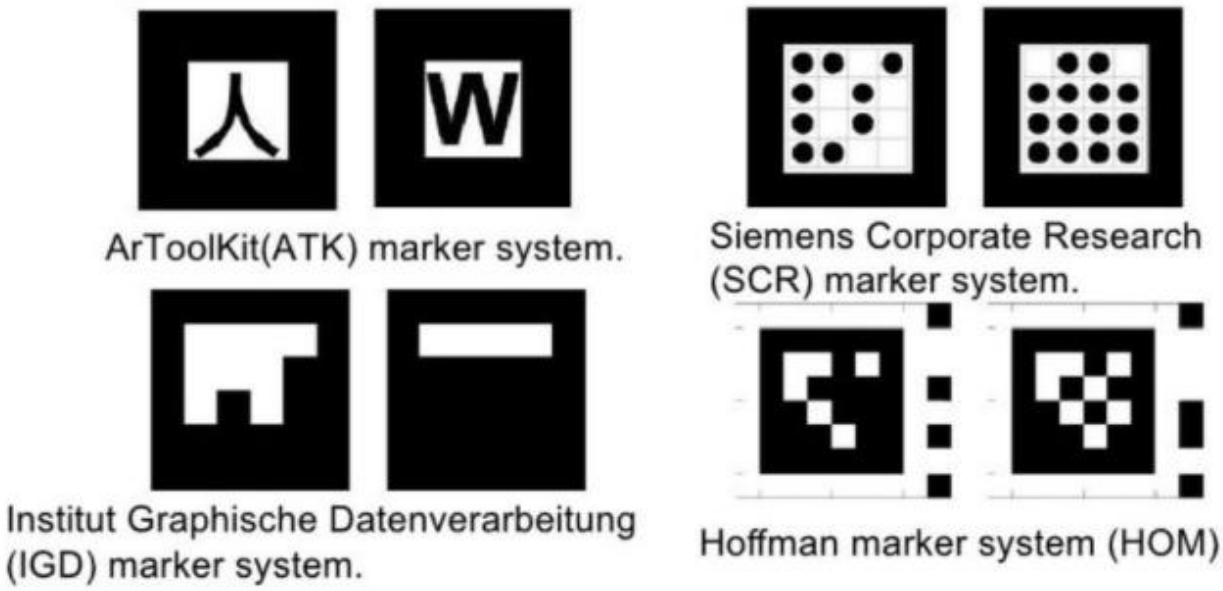

Рисунок 1.2 – Приклади маркерів

Прилад, наприклад телефон, може розпізнати такі типи маркерів та доповнити реальність. Геомертична фігура, наприклад квадрат, це також класичний варіант маркеру, тому що досить легко побудувати площину і знайти гомографію.

Але просто квадрат не є гарним варіантом маркеру, так як він виділяється з реальності і вносить дисонанс, краще використовувати більш реальні речі, наприлад фото або картину, це також є один з варіантів маркеру.

Що стосується без маркерної технології. Можна отримати координати положення приладу в просторі і тим саме це також буде графічний маркер. Тобто є

можливість розпізнати частину нашої реальності і таким чином сказати що це є наш маркер.

1.3 Важливість маркерів в доповненій реальності

На даний момент, можна спостерігати величезну появу відстеження природних об’єктів: наприклад відстеження рухів гравця за допомогою Microsoft Kinect, різні всюдисущі інтерфейси на основі жестів, професійні безмаркерові оптичні 3D-сканери тощо. З цієї точки зору область відстеження штучних об’єктів може легко виглядати застарілою. Тим не менше, насправді це не так.

Багато оптичних сканерів все ще використовують маркери для більш точного вирівнювання, навіть незважаючи на те, що вони можуть реконструювати форму без них (наприклад, GOM Atos). Також зйомка руху фільму все ще часто базується на використанні діодних маркерів. Таку саму ситуацію можна спостерігати в області взаємодії людини і комп'ютера. Управляти додатком можна за допомогою простих жестів (стискання, масштабування, панорамування тощо), однак у багатьох випадках використання маркерів дозволяє більш точне введення або навіть нові функції, які практично неможливо реалізувати без маркерів.

Як приклад можна взяти додаток для розширеного прототипу, описаний у виданні «Прискорене використання доповненої реальності для створення прототипів». Маркери, розміщені на поверхні прототипу автомобіля, представляють різні частини автомобіля, які вставляються на місце події [7]. Складність вибору відповідної 3D-моделі та складність регулювання положення в кадрі значно зростають із збільшенням кількості необхідних деталей. Навіть суто точність точного регулювання положення однієї 3D-моделі не є незначною. Навпаки, додавати квадратний маркер у сцену – це природно і просто. Теоретично маркер у цьому випадку міг бути замінений природним об’єктом. Однак використання природного об'єкта може суттєво знизити точність оцінки пози і тим

більше поставить проблему розмежування декількох подібних природних об'єктів. Отже, така заміна буде контрпродуктивною у такому застосуванні.

1.4 Техніка дослідження алгоритму розпізнавання маркерів

Для того, щоб описати техніку дослідження первинного алгоритму розпізнавання маркерів, необхідно більш детально розглянути зайняті маркери та їх ідентичності, потім окреслити обраний алгоритм розпізнавання маркерів, потім представити створені віртуальні середовища і, нарешті, детально представити створену систему AR-системи.

1.4.1 Маркери та їх ідентичність

Різні типи маркерів можна використовувати для збільшення реальності, де згідно з теорією застосування доповненої реальності [3] на основі маркерів та властивостями конструкцій системи зору на основі маркерів [1], найпоширенішими маркерами є: смуги (QR, DataMatrix, PDF-417), кола (CircleInner, CircleSplit, CircleOuter, SpotCode), квадрати (маркери ідентифікаторів, двійкові) та зображення (StudierStube). У теорії застосувань доповненої реальності на основі маркерів показано, що квадратний маркер з товстою межею є найбільш точним для систем на основі маркерів. Це пов’язано з:

- форма маркера – її можна легко знайти, навіть використовуючи різні методи або алгоритми;
- кольорова контрастність – навіть на різних камерах кольорові відмінності легко помітити;
- тип кордону – є більш надійним у більшості ситуацій, які можна

застосувати.

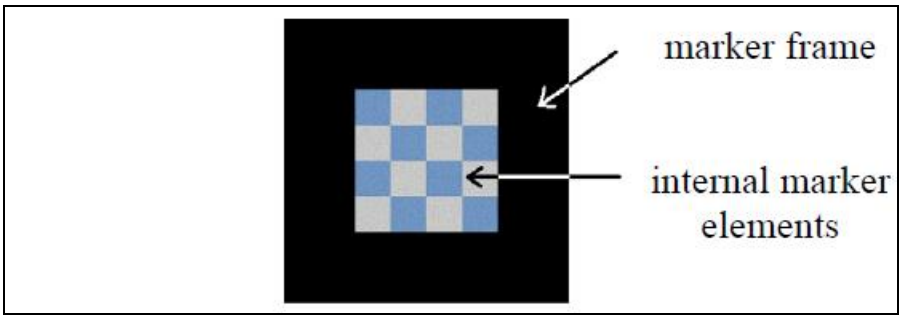

Рисунок 1.3 – Маркерна рамка з окресленими внутрішніми елементами всередині

Квадратний маркер може бути створений з різною кількістю внутрішніх елементів (4x4, 5x5 і до 177x177), що безпосередньо впливає на час обробки. Складова рамка маркеру показана на рис. 2.2.

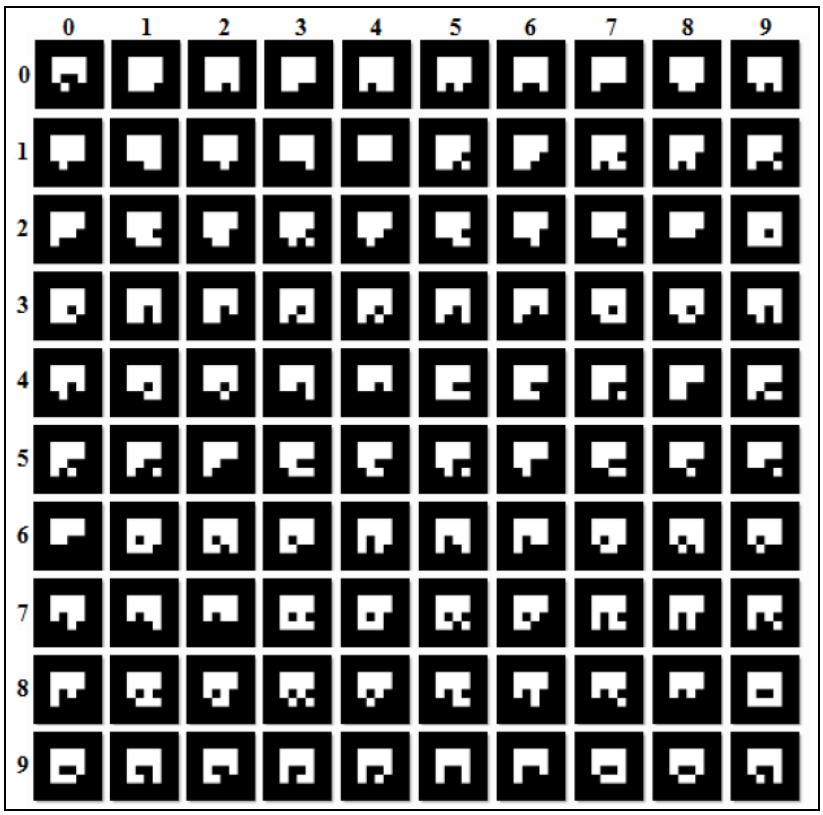

Рисунок 1.4 – Досліджені маркери та їх відповідний ідентифікатор (номер рядка; номер стовпцю)

Усі маркери мають однакову товщину межі. Кожен зайнятий маркер має

ідентифікатор, присвоєний відповідно до рис. 2.3 (де перша ідентифікаційна цифра - це номер рядка, друга - номер стовпця), в результаті чого ідентифікатор від 00 до 99. Усі маркери стійкі до поворотів уздовж осі z , тобто поворот маркера вгору (вгору) не вплине на точність розпізнавання маркера.

1.4.2 Алгоритм розпізнавання маркера

Розглянемо алгоритм розпізнавання маркерів, який базується на бібліотеці NyARToolKit, яка походить від бібліотеки ARToolKit, яка була прийнята для мови програмування Java, а також для інших мов програмування, таких як C # та операційних систем (наприклад, Android), які підтримують віртуальні машини [8]. Головна перевага використання бібліотеки NyARToolKit полягає в тому, що в ній вирішена проблема відстеження точки зору користувача [9]. Функції з бібліотеки дозволяють відстежувати кут огляду без втрати віртуального об'єкта. Це забезпечує вигляд чіткого доповненого подання. Основні етапи обробки алгоритму починаються з введення зображення, яке згодом, використовуючи поріг освітлення, бінаризується та маркується. Після цього, коли список компонентів отримано, починається виявлення контуру. На основі результатів виявлення контурів виконується виявлення кутів субпікселів і нормалізовані знайдені шаблони. На основі внутрішніх параметрів камери та знайденого списку компонентів складається список областей маркерів. Потім виконується узгодження шаблону. Шаблони маркерів беруться з заздалегідь підготовлених окремих файлів *.patt*. Розмір маркера та список параметрів рядків, взятий із виявлення кута субпікселя, обробляється для обчислення гомографії. Після застосування маркерів перетворення камери трансформація отримується і використовується для остаточного кроку розпізнавання. У разі успішного розпізнавання виводиться ідентифікаційний номер розпізнаного маркера, інакше вказується помилка розпізнавання.

1.4.3 Налаштування та впровадження віртуального середовища.

Для того, щоб провести різні експерименти, для початку необхідно налаштувати та підготувати віртуальні сцени, вони показані на рис. 1.5 та рис. 1.6. На зображеннях можна побачити такі об'єкти (з їх можливими системами координат): сцена, маркер з ID 00, камера та джерела світла. Розглянемо чотири віртуальні сцени, різні за освітленістю [10]:

а) світло – з природним оточуючим світлом (рис. 1.5);

б) сутінки – з рівномірно розподіленою тінню та розмитим навколишнім освітленням (рис. 1.5).

в) тінь 1 – з яскравим джерелом світла з лівого боку (рис. 1.6);

г) тінь 2 – із розмитим джерелом світла з лівого боку (рис. 1.6).

Обробка у віртуальному середовищі реалізується наступним чином. Кожні 5–25 мс маркер (починаючи від маркера з ID 00) повертається на 1 ° вздовж осі x від -90 ° до 90 ° у першій віртуальній обстановці Тіні 1, після цього обертання виконується вздовж осі y. Наступний маркер вибирається та обробляється з подібною послідовністю дій. Після того, як усі 100 маркерів, що обробляють результати розпізнавання, записуються в матрицю розміром 181x200 елементів (кожен елемент містить розпізнаний індекс маркера або знак відмови). Вибране спрямоване джерело світла знаходиться з лівого боку камери. Постійне місце створюється для визначення змін характеристик інтенсивності освітлення при зміні ступеня. Потім сцена віртуального середовища змінюється, і вся обробка повторюється [11].

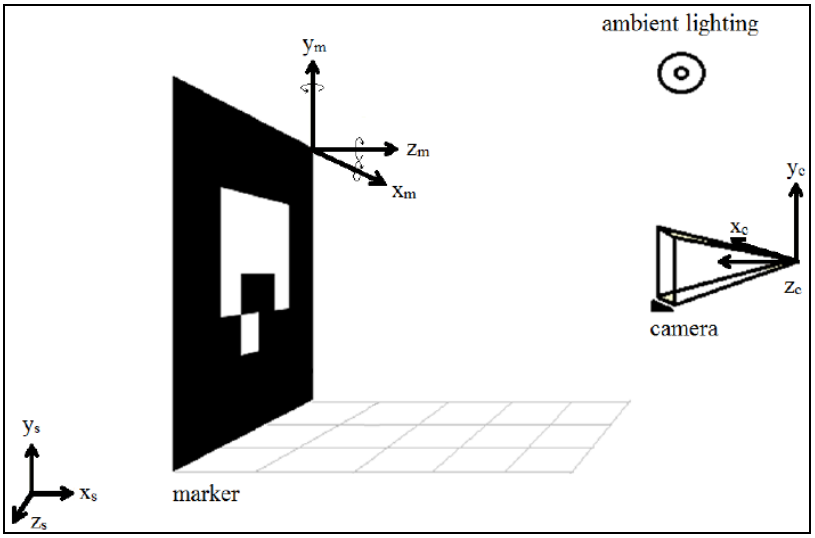

Рисунок 1.5 – Сцена розслідування у віртуальному середовищі (Світло/Сутінки)

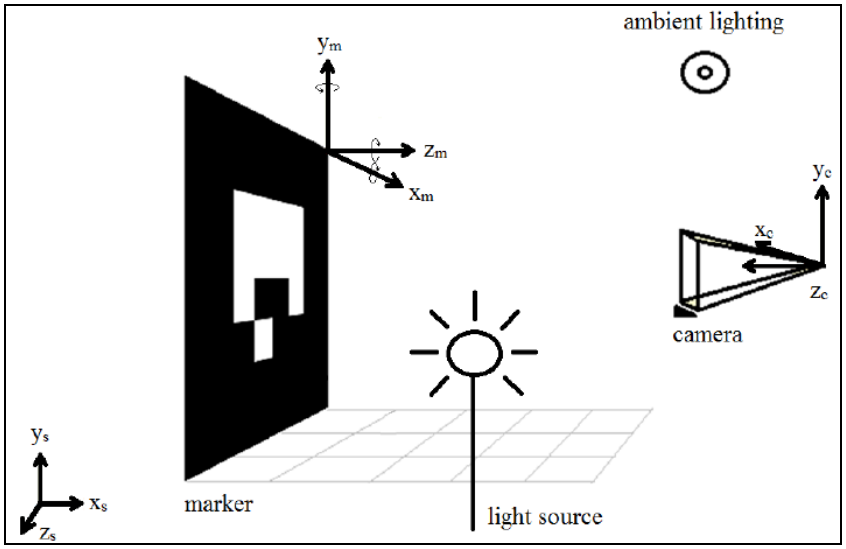

Рисунок 1.6 – Сцена розслідування у віртуальному середовищі (Тінь1/Тінь2)

1.4.4 Впровадження системи доповненої реальності

Систему доповненої реальності можна створити за допомогою комп’ютера та різних гаджетів з камерою. Найпопулярнішими гаджетами для систем AR є: встановлені на голові, окуляри, контактні лінзи, віртуальний дисплей сітківки, портативний пристрій. Насправді, підготовка до випробування є складною, оскільки вона повинна мати гарантовану систему координат, повністю

контрольовані умови освітлення, підтримувати кути обертання вздовж обох осей і точність (на кожен градус), а всі використані похибки інструментів не повинні перевищувати допустимі. Усі вищезазначені проблеми мінімізуються або зменшуються, якщо тестування системи AR проводиться у віртуальному середовищі [12]. Для цілей розробки створена система доповненої реальності включає такі функціональні частини:

- завантаження зображення;
- попередня обробка зображень;
- обробка зображень із виділенням функцій;
- розширене завантаження відеоданих;
- AR система потокового відео.

Кожна частина системи відповідно до її власних експлуатаційних особливостей може бути змінена або скасована, залежно від бажаних результатів.

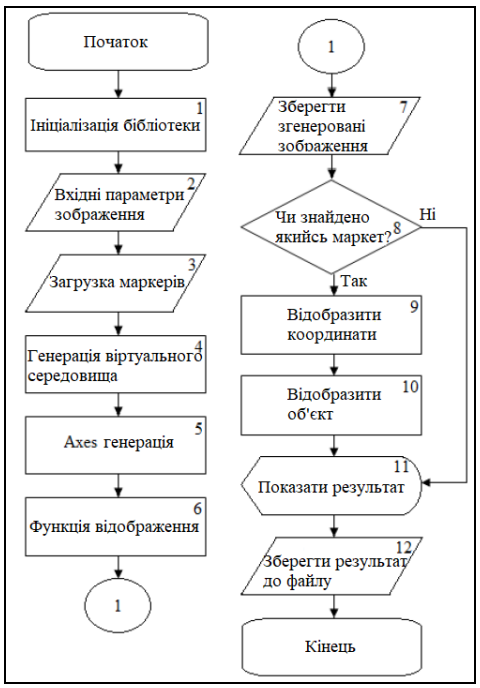

Рисунок 1.7 – Блок-схема впровадження системи доповненої реальності

Один із способів розробки такого роду системи доповненої реальності показаний на блок-схемі на рис. 2.6. На кроках 1 і 2 бібліотеки ініціалізуються та вводяться параметри зображення. На етапі 3 завантажуються 100 маркерів

(показано на рис. 2.3). На кроках 4–7 створюється віртуальне середовище, маркери обертаються та обробляються. На кроці 8 проводиться розпізнавання маркера. На кроках 9–11 вимальовуються координати та об’єкт (у даному випадку простий кольоровий куб з ідентифікатором маркера зверху), показуючи, чи виявлено маркер і чи правильно виявлено. На завершальному кроці 12 усі дані про розпізнавання маркерів записуються у файл.

1.5 Постановка задачі

Метою атестаційної роботи є досліждення методів розпізнавання маркерів у технології доповненої реальності. Проаналізувавши предметну галузь, можна сказати що AR технологія набуває великої популярності, але на даний момент не існує максимально ефективних алгоритмів маркерування. Кожен з алгоритмів має свої переваги, але в цей же час має свої недоліки.

Таким чином, для досягнення мети магістерської роботи, необхідно:

- дослідити існуючі методи та алгоритми маркерування доповненої реальності;
- дослідити існуючі бібліотеки та інструменти розробників, для створення додатків які використовують технологію доповненої реальності;
- створити прототип додатку з використанням певних технологій.

У результаті досліження повинно бути проаналізовано існуючі методи та алгоритми меркерування, рекомендації щодо їх використання, а також розроблений прототип, який буде повірняний з іншим додатком та будуть проаналізовані його переваги та недоліки.

2 ДОСЛІДЖЕННЯ МЕТОДІВ РОЗПІЗНАВАННЯ МАРКЕРІВ

2.1 Методи розпізнавання маркерів

Опишемо кілька методів розпізнавання та класифікації, які можуть бути використані для опису ознак у процесі виявлення маркерів. Ці методи включають системи навчання, багатовимірна класифікація або нейронні мережі, але вони скоріше застосовні для процесу прийняття рішень. Існуючі програми доповненої реальності в режимі реального часу використовують інші підходи [1] [2]. Як правило, для розпізнавання маркера зазвичай використовують два підходи: виявлення на основі ознак та виявлення на основі країв, є також інші підходи, але вони використовуються не так часто, їх також буде коротко розглянуто.

2.2 Узагальнений алгоритм розпізнавання маркера

Це самий основний алгоритм компютерного зору, який пов’язаний з розпізнаванням маркерів [13]. Спочатку необхідно оригінальне зображення привести в градацію сірого, далі необхідно виконати бінаризацію зображення, потім визначити замкнуті області, виділити контури та виділити кути маркеру і в кінці перетворити координати.

Кожен з цих кроків є важливою частиною алгоритму, вони повинні виконуватись послідновно для досягнення максимально позитивного результату. У результаті буде отримано якісний маркер, який можна ефективно використовувати у додатках з доповненою реальністю.

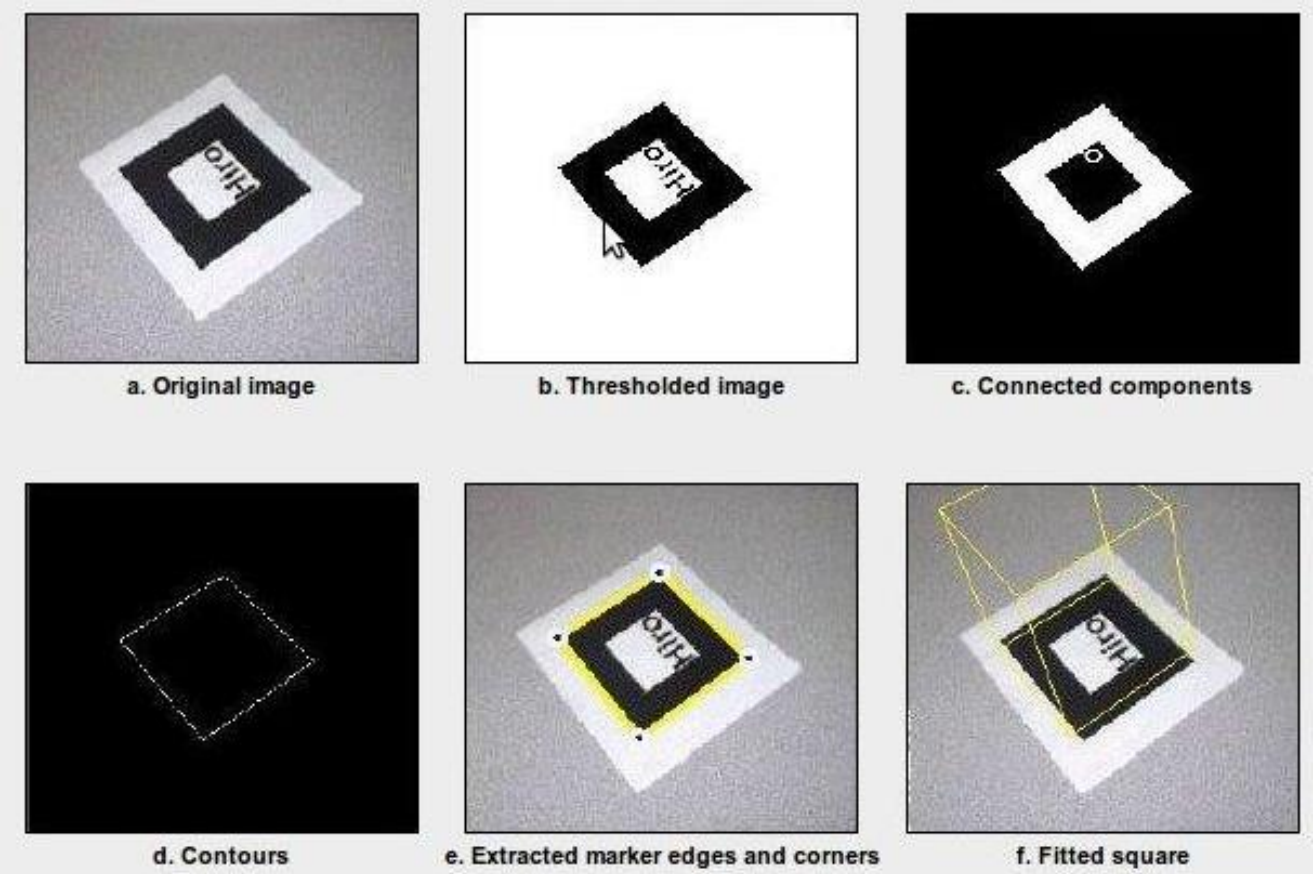

Рисунок 2.1 – Узагальнений алгоритм розпізнавання маркеру

2.2.1 Переклад оригінального(кольорового) зображення в градацію сірого

Існує три алгоритма, за допогою яких відбувається переклад кольорового зображення в градацію сірого. Розглянемо кожен з них, а також наведемо приклади зображень, як на практиці виглядають ці методи, рисунок 2.2:

- lightness – GS = (max(R, G, B) + min(R, G, B))/2;
- luminosity – GS = 0.21 * R + 0.72 * G + 0.07 * B;
- average – GS = (R + G + B) / 3.

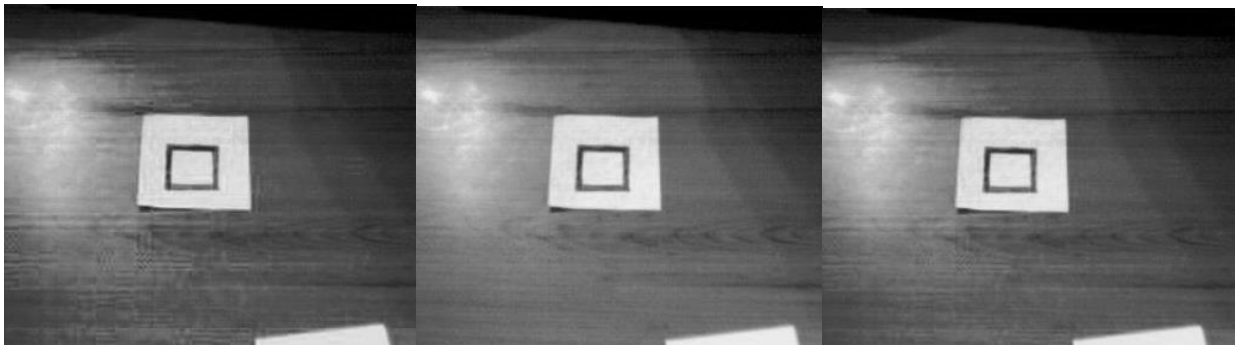

Рисунок 2.2 – Lightness, Luminosity, Average

Як можна помітити, візуально вони майже нічим не відрізняються, але на практиці є значна різниця. Як правило, використовують саме метод Luminosuty.

2.2.2 Бінаризація зображення

Наступний крок в узагальненому алгоритмі розпізнавання маркеру, є бінаризація зображення, це більш складніший та цікавіший етап.

Для того, щоб була змога перекласти зображення в так би мовити двокольоровий стан(чорно-білий), використовується певний поріг. Але навіщо потрібен цей поріг і до чого його примінювати не зовсім зрозуміло.

Один з найпростіших прикладів задати поріг, це наприклад, якщо ми маємо 256 кольорів, то поріг може бути 128, таким чином, получиться перекласти зображення в чорно-білий стан.

Також існує гірсограма кольору, користуючись якою можна вибрати поріг, але це складніший шлях.

Існує шість груп методів, за допомогою яких можна перетворити зображення в чорно-білий стан:

- методи на основі вивчення ентропії;
- методи, які базуються на пошуку подібності між сірим і чорно-білим зображенням;

- методи на основі класеризації;
- методи засновані на "формі" гістограми;
- методи що використовують кореляційні залежності та особливості статистичного розподілу між пікселями в областях зображення;
- методи засновані на локальній адаптації порога для кожного пікселя зображення.

Рисунок 2.3 – Бінаризація з заданим порогом

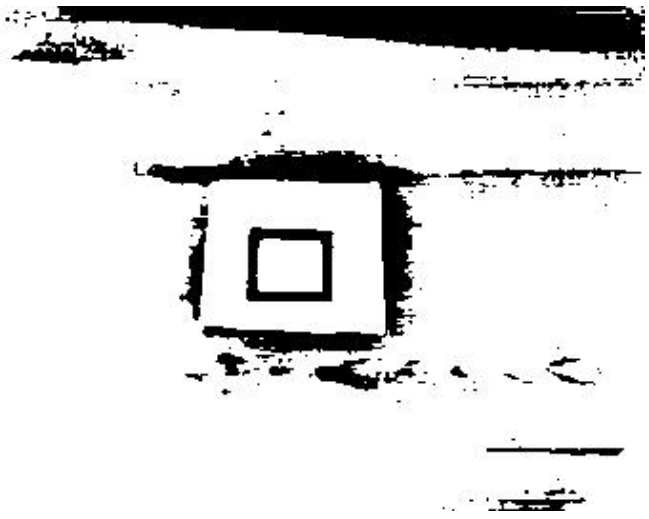

Рисунок 2.4 – Адаптивна бінаризація

Існує понад сорок методів бінаризації. Наведемо декілька прикладів бінаризації зображення із заданим порогом, адаптивного та за допомогою методу Оцу.

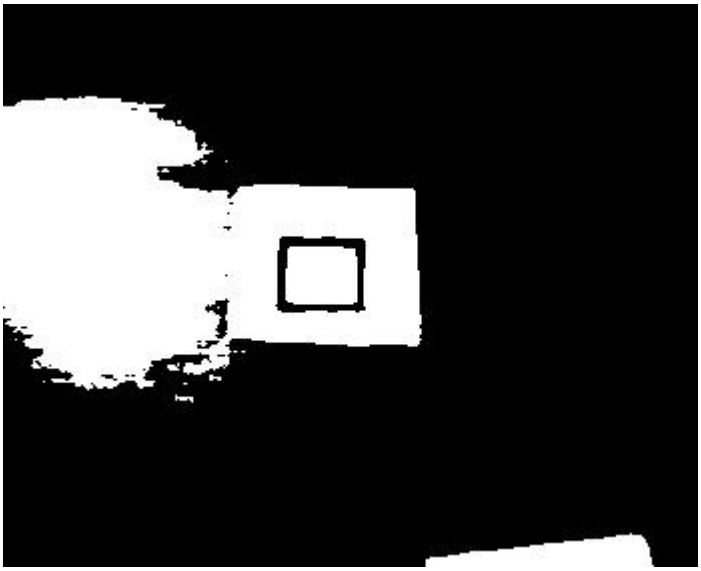

Рисунок 2.5 – Бінаризація методом Оцу

2.2.3 Визначення замкнутих областей

На даному кроці, потрібно визначити замкнуті області. В основному, на цьому кроці йде комбінація алгоритмів, але найчастіше, просто застосовують алгоритми «заливки» областей і таким чином виділяють замкнуті області.

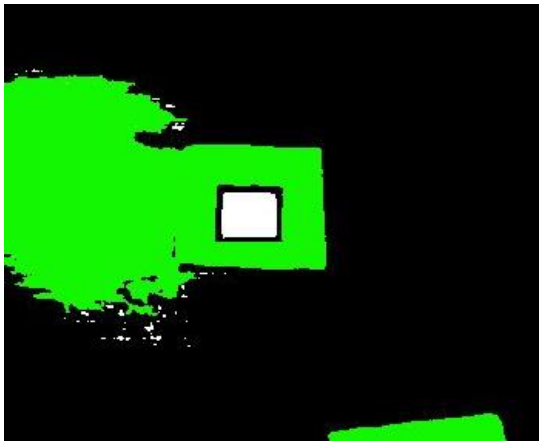

Рисунок 2.6 – Перевірка на замкнутість

Саме на цьому кроці можуть бути різні варіації, наприклад, на початку можна виділити контури, а потім їх перевірити на замкнутість.

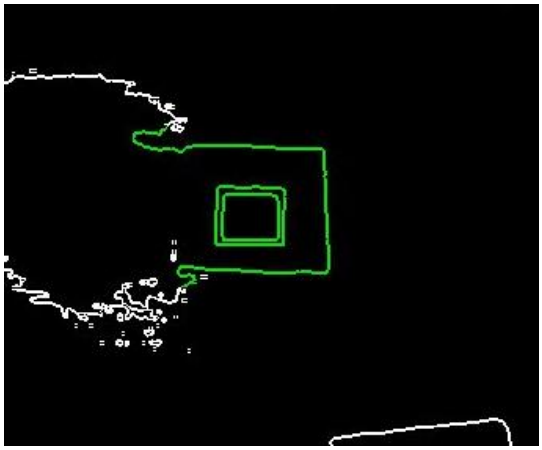

Рисунок 2.7 – Спочатку виділення контурів, а потім перевірка на замкнутість

Яким же чином можна виділити контури? Існує декілка вирішень цього питання. Розглянемо основні з них:

- детектор контуру Марра-Хілдрета;
- детектор контуру Canny;
- детектор контуру на основі булевої функції;
- евклідова відстань та детектор контуру на основі кута вектора;
- виявлення глибинного краю за допомогою кількох спалахів;
- детектор контуру Sobel.

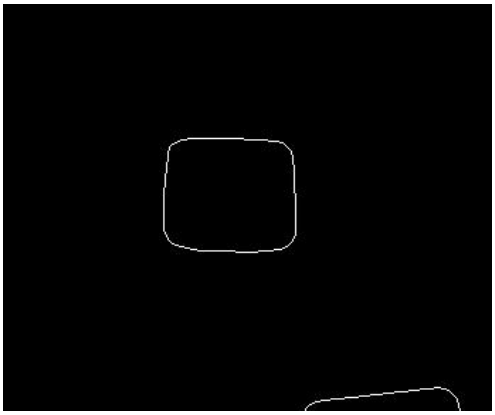

Рисунок 2.8 – Canny детектор

Алгоритми Canny та Sobel використовуються найчастіше. На зображенні видно білу, замкнуту область, саме в них і відбувається пошук контурів.

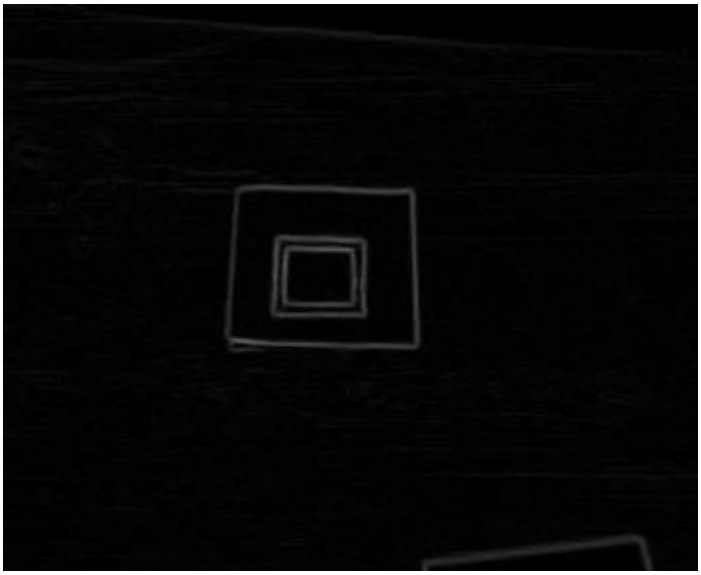

Рисунок 2.9 – Детектор контуру Sobel

2.2.4 Виділення кутів маркера

Для того, щоб виділити контур, спочатку необхідно зіставити його з маркером. На зображенні може бути велика кількість різних контурів, необхідно знайти фігуру, схожу на чотирикутник.

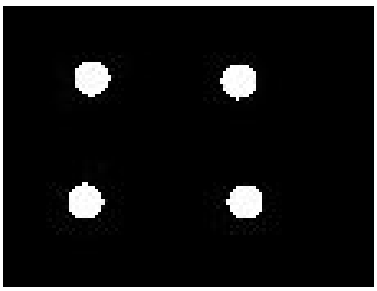

Рисунок 2.10 – Координати кутів маркера

Алгоритм Дугласа-Пекера чудово підходить для вирішення цього завдання, він дозволяє зменшити кількість точок кривої. Є уже готові функції за допомогою яких можна отримати потрібний результат, наприклад функція approxPolyDP в бібліотеці OpenCV вже виконує ці дії.

2.2.5 Перетворення координат

Останній крок узагальненого алгоритму, це перетворення координат. Оскільки, виконавши попередні кроки, маємо координати кутів маркера, які в ідеалі, мають бути перпендикулярними, але насправді вони розташовані під іншим кутом. Окрім цього, сторони квадрату це осі координат. Тож, таким чином, відносно об’єкту, можна отримати точку відліку початку координат, а також положення камери.

Графічно, цей метод виглядає таким чином, як показано на рисунку 2.11.

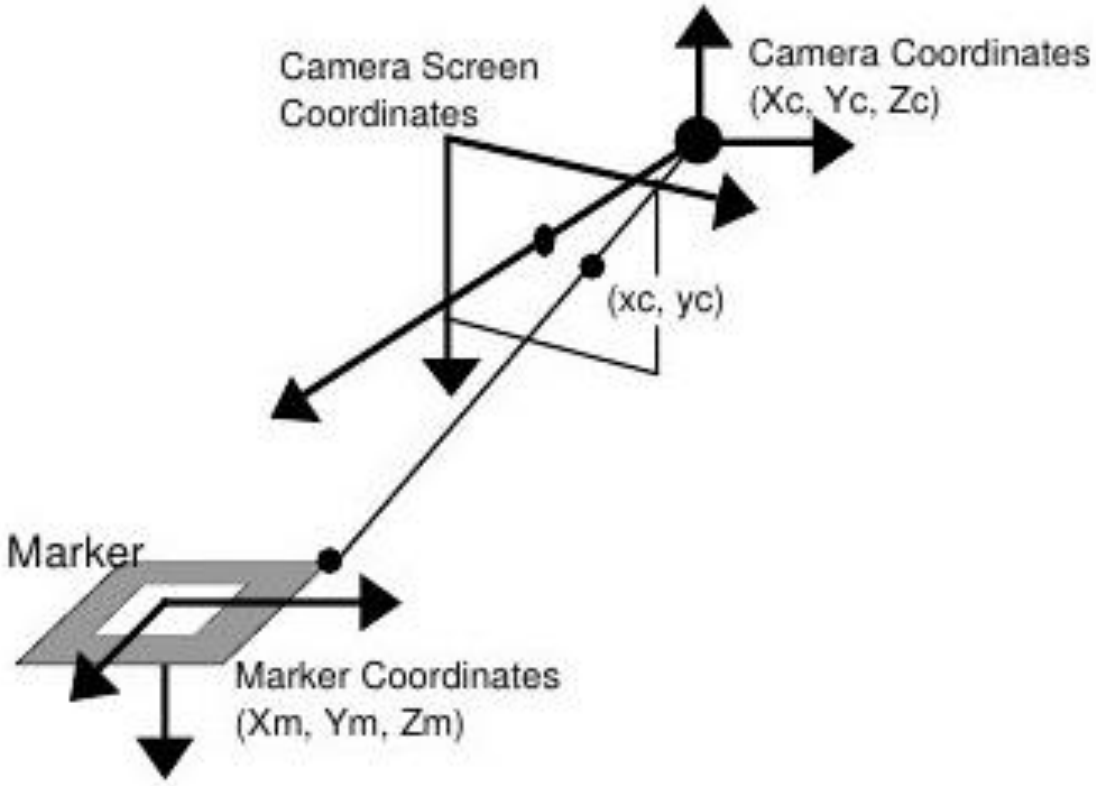

Рисунок 2.11 – Положення камери відносно об’єкту

Уся суть заключається в тому, що розмір проекції змінюється, при зміні кута камери.

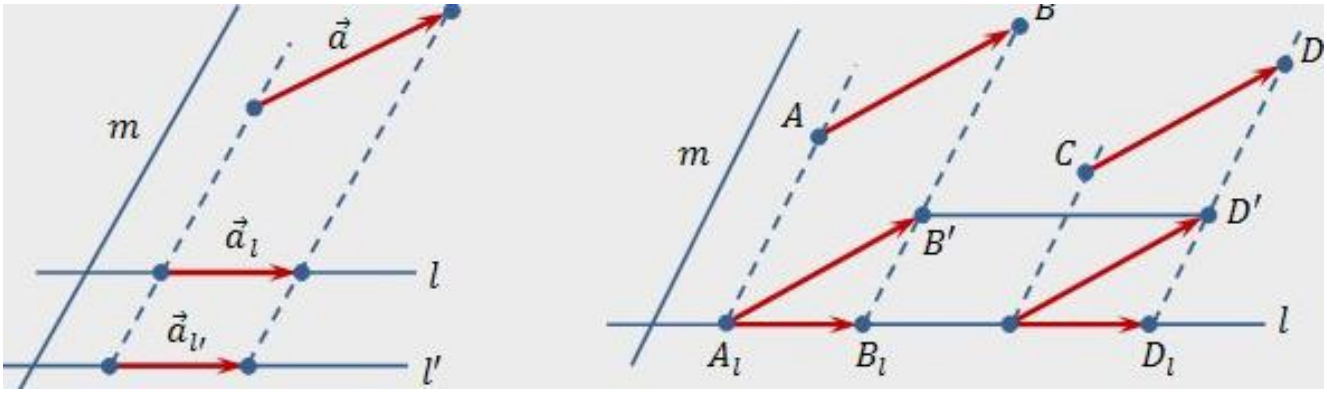

Рисунок 2.12 – Проекція вектору на площину

Таким чином, знаючи точку відліку та місце розташування камери, можна намалювати проекцію 3D моделі. На рисунку 2.12 можна помітити, що це є зображення куба.

Але не завжди є можливість чітко зрозуміти розташування та кут нахилу камери, тому що при використанні звичайного квадрату з чистими полями, він є симетричний, і тому розпізнавання може бути лише часткове. Тому можна вирішити це питання, додаваннями додаткового маркеру в квадрат.

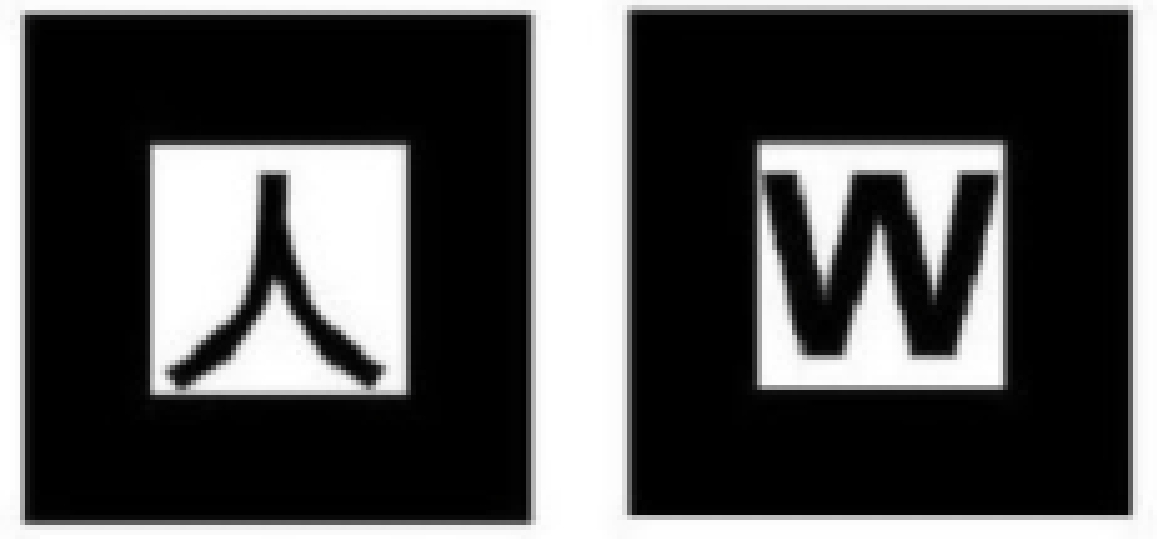

Рисунок 2.13 – Додатковий маркер, розташований у маркері

Таким чином за допомогою перетворення Хафа, є можливість додатково отримати кут нахилу та повороту.

2.3 Виявлення на основі ознак

Виявлення ознак – це багатоетапний процес. Його компоненти варіюються залежно від алгоритмів. Далі розглянемо короткий опис типового алгоритму виявлення:

Виявлення ключових точок.

Наприклад, це може бути алгоритм виявлення кутів, який враховує контраст між сусідніми пікселями на зображенні.

Щоб зробити кандидатів на точки функцій незмінними за шкалою та менше залежати від шуму, зазвичай розмивають зображення. Різномасштабні варіанти зображення (октави) покращують незалежність від масштабу. Хорошим поясненням є те, що ви зазвичай хочете виявити дерево в цілому для досягнення надійного відстеження; не кожна окрема гілочка. Алгоритми SIFT та SURF використовують цей підхід [6].

На жаль, розмиття є обчислювально дорогим. Для сценаріїв реального часу інші алгоритми, такі як BRISK, можуть забезпечити кращий загальний досвід. Ці алгоритми (включаючи BRISK) часто базуються на похідній алгоритму FAST Ростена та Драммонда [7].

FAST аналізує кругове оточення кожного пікселя *p*. Чи є яскравість навколишнього пікселя нижчою або ж вищою, ніж *p* (включаючи поріг)? Якщо певна кількість підключених пікселів відповідає цьому критерію, алгоритм знайшов кут. На це вказує пунктирна лінія на зображенні нижче. Пікселі від 7 до 10 темніше, ніж *p*; інші яскравіші.

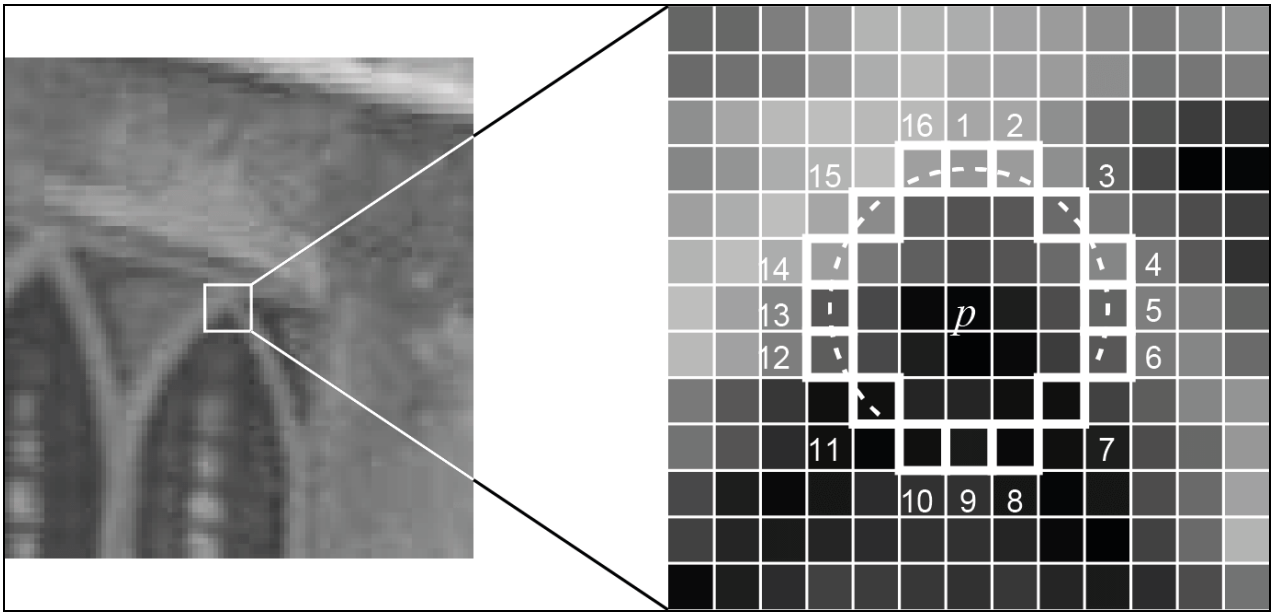

Рисунок 2.14 – Алгоритм FAST Ростена та Драммонда

Для BRISK принаймні 9 послідовних пікселів у 16-піксельному колі повинні бути достатньо яскравішими або темніше центрального пікселя. Крім того, BRISK також використовує зображення зменшеного розміру (піраміда масштабного простору) для досягнення кращої незмінності масштабу - навіть досягнення точності субпікселів.

Опис ключових точок.

Кожна з усіх виявлених точок повинна мати унікальний відбиток пальця. Алгоритм повинен знову знайти функцію в іншому зображенні. Це може мати іншу перспективу, блискавичну ситуацію тощо. Навіть за цих обставин матч повинен бути можливим.

Дескриптор BRISK - це двійковий рядок із 512 бітами. По суті, це об’єднання результатів порівняння яскравості між різними зразками, що оточують центральну ключову точку.

Як показано на малюнку, сині точки утворюють концентричні кола для позицій відбору проб. Червоні кола позначають окремо відібрані області на основі гауссового згладжування, щоб уникнути ефектів згладжування.

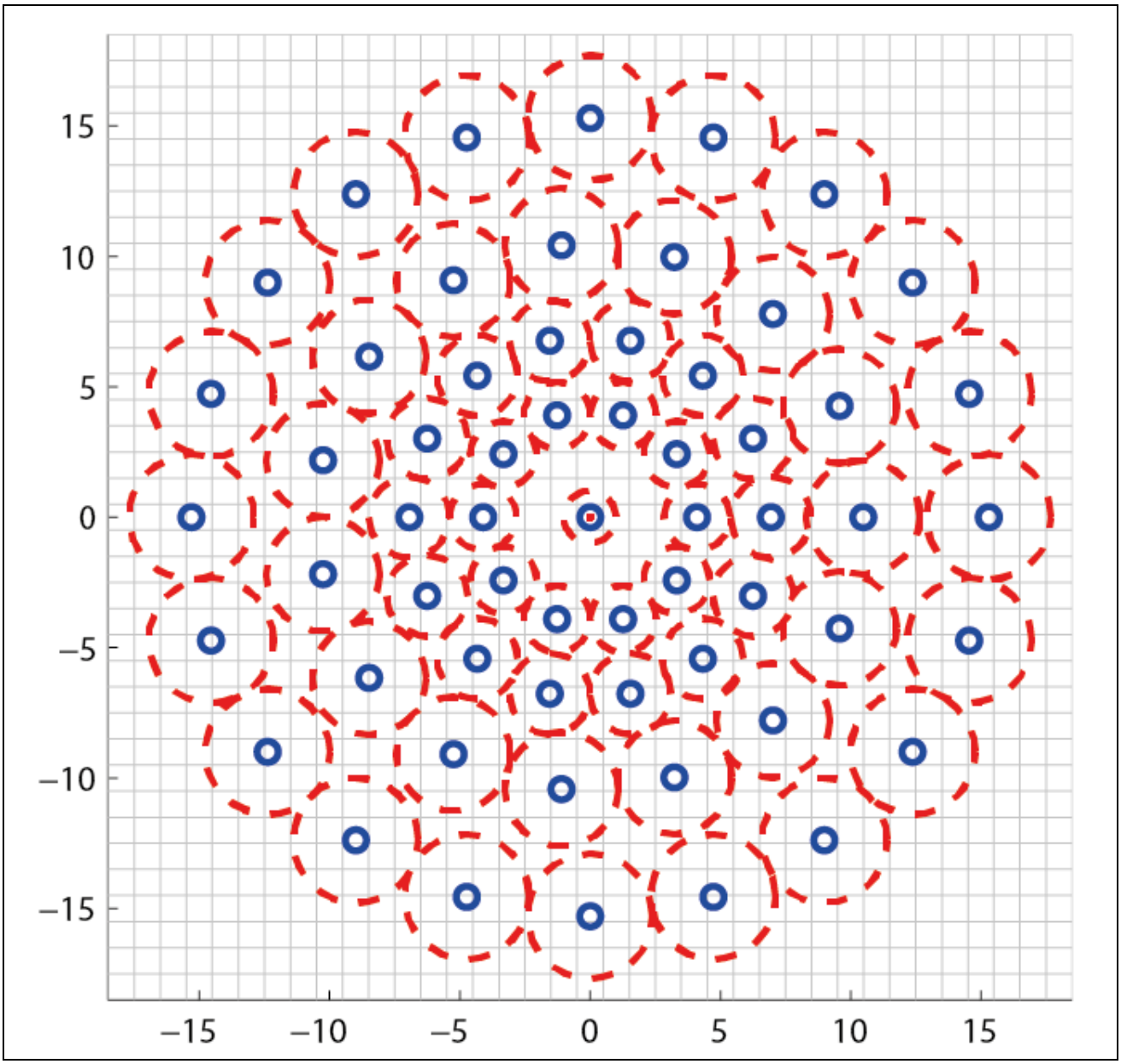

Рисунок 2.15 – Шаблон вибірки BRISK

На основі цих точок відбору проб обчислюються результати порівняння яскравості - залежно від того, який зразок яскравіший, це дає 0 або 1.

Крім того, BRISK забезпечує інваріантність обертання шляхом обчислення характерного напрямку. Це визначається найбільшими градієнтами між двома зразками на великій відстані один від одного. По суті, це означає: під яким кутом прорізання області ключової точки призведе до найбільшої різниці яскравості між обома сторонами вирізу?

2.4 Виявлення країв

Методи цієї групи базуються на частковій реконструкції морфології зображення. Добре відомий ARToolKit використовує розпізнавання на основі виявлення лінії [8]. Маркер ARToolKit розроблений як чорно-біле зображення всередині чорного прямокутника. Як описано в методах реєстрації на основі відстеження текстур за допомогою ARToolKit, попередня обробка зображення супроводжується кластеризацією підключених пікселів. Згодом знайдено контур кожного зв'язаного кластера пікселів, а точки, що представляють кути маркерів, використовуються для обчислення гомографії. Ця матриця гомографії використовується для корекції перспективи знайденого маркера. Крім того, для ідентифікації маркера використовується відповідність шаблону. Зображення всередині чорного прямокутника потрібно порівнювати із збереженими візерунками (шаблонами). Маркер ідентифікується на основі кореляції з шаблоном.

ARTag базується на іншому підході. У ньому використовуються маркери, подібні до ARToolKit. Основна відмінність полягає в тому, що чорно-білий шаблон, який представляє двійковий код, знаходиться всередині межі. Визначення меж на основі ідентифікації крайових пікселів та їх кластеризації в лінії використовується для розпізнавання маркера. Отже, чотирикутники створюються із пересічених ліній. Потім для обчислення гомографії використовують кути чотирикутників. Нарешті, код виправлення помилки всередині межі маркера використовується для ідентифікації маркера. Цей метод кластеризації крайових пікселів дозволяє ARTag мати справу з частковими оклюзіями бічних маркерів. Однак він не може впоратися з кутовою оклюзією.

У системі під назвою RUNE-tag використовуються точки, розташовані на диску. Диск розділений на сектори, і кожен сектор можна розділити на концентричні кільця. У кожному концентричному кільці можна поставити крапку. На думку авторів, розміщуючи крапки на різних колах, можна створити до 20 000 різних маркерів і розпізнати маркер з оклюзією до 2/3 крапок.

Також існує гібридний метод, який поєднує в собі метод виявлення маркерів, що базується на краях, і метод виявлення ознак. Серед кутів маркера виявляються особливості всередині маркера. Коли відстеження кутів маркера не вдається, виконується відстеження об’єктів за раніше виявленими об’єктами. Цей підхід використовується для подолання проблеми оклюзії та тремтіння.

2.5 Впровадження та результати

Розглянемо яким чином можна реалізувати два підходи до виявлення маркерів – виявленних на основі морфології та виявлення S-G. Перший швидкий, тим не менше, він не здатний боротися з прикусами. Останній заснований на вилученні та узгодженні функцій. Незважаючи на те, що він здатний мати справу навіть із суто закупореними маркерами, його ефективність значно нижча.

2.6 Розпізнавання маркера на основі краю

Представлений підхід складається з наступних добре відомих етапів: попередня обробка зображення камери, морфологічний аналіз попередньо обробленого зображення та пошук маркера в морфологічному описі зображення. Його загальна ідея базується на підході, представленому в методах реєстрації на основі відстеження текстур за допомогою ARToolKit [8]. Етап попередньої обробки складається з перетворення відтінків сірого, видалення непотрібних деталей за допомогою розмиття за Гаусом і, нарешті, адаптивного порогового значення. Суть цієї фази полягає у видаленні із зображення якомога більшої кількості деталей, оскільки кількість деталей безпосередньо впливає на продуктивність.

На другому етапі контури виявляються в межах порогового зображення і за

допомогою морфологічного аналізу будується список виявлених багатокутників. Кожен багатокутник у списку описується набором вершин та сусідів. Сусід може бути як горизонтальним, так і вертикальним. Горизонтальний сусід – це багатокутник, який знаходиться поруч із поточно обробленою. Вертикальний сусід – це багатокутник у межах обробленого в даний час.

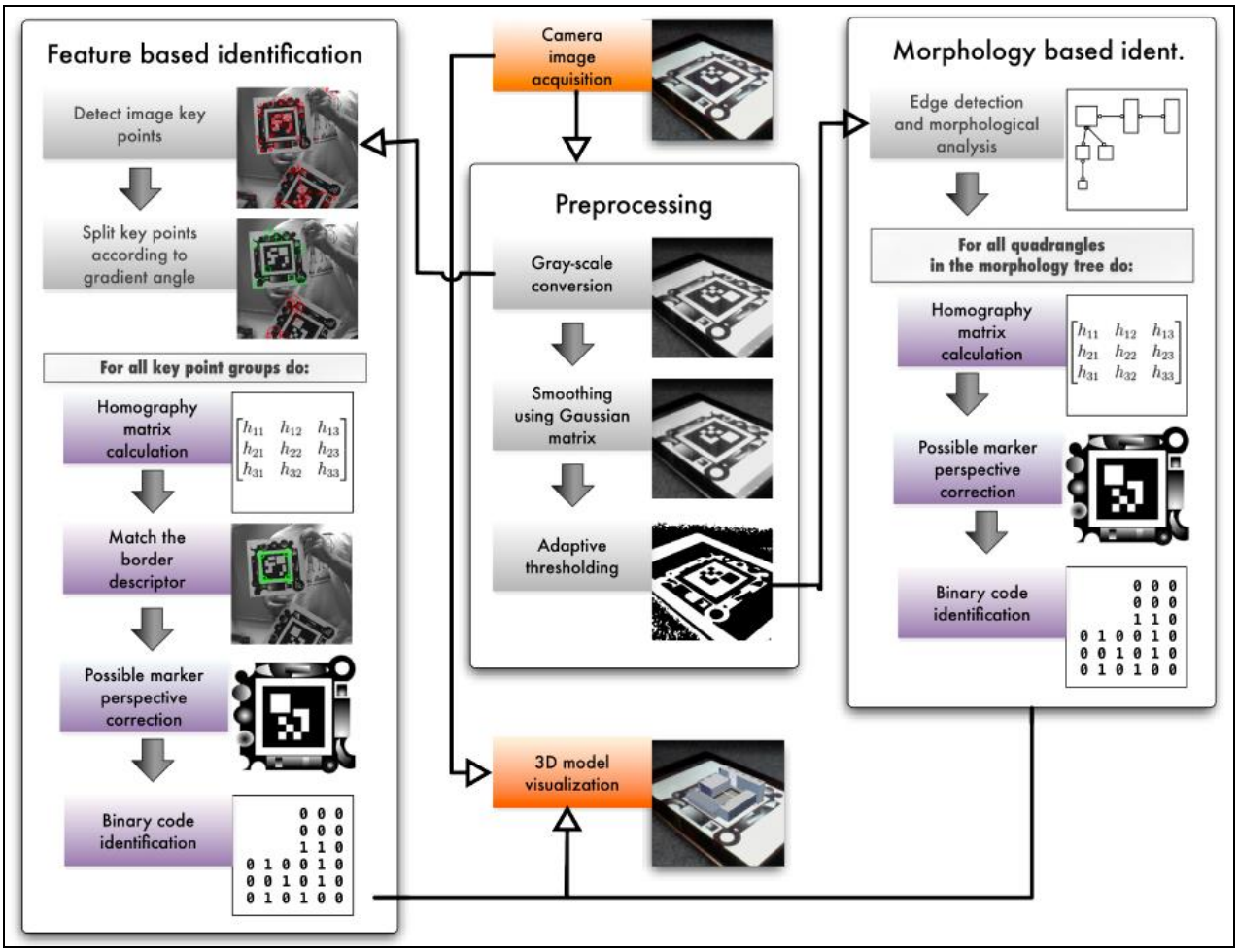

Рисунок 2.16 – Схема розпізнавання маркерів, яка окреслює основний принцип обох реалізованих методів

Фінальним кроком є цикл через усі виявлені полігони. Мета – описати чорний маркер з білим квадратиком у верхньому лівому куті. Кожен багатокутник тестується за таких умов: він має чотири вершини, у нього знову є вертикальний сусід з чотирма вершинами. Іншими словами: це чотирикутник всередині чотирикутника. Цей простий тест усуває більшість об’єктів у дереві. Для кожного

потенційного маркера обчислюється матриця гомографії, яка дозволяє усунути перспективні деформації. Нарешті, можна виявити двійковий код на поверхні маркера. Реконструйований код перевіряється за допомогою алгоритму виправлення помилок Голея. У випадку, якщо код знайдено і він підключений до 3D-моделі, модель розміщується всередині зображення камери за допомогою матриці зворотної гомографії. Очевидно, що навіть часткова оклюзія маркера спричиняє його зникнення, оскільки багатокутники вже не виконують необхідних умов.

2.7 Оклюзії

Після виявлення та ідентифікації всіх маркерів на зображенні камери за допомогою методу, описаного вище, наш двигун намагається ідентифікувати закупорені нерухомі маркери. Двигун виявляє всі особливості з високим власним значенням крім того, цей метод використовується для отримання оновлених місць розташування об’єктів з точністю до пікселів.

У методі, функції порівнюються з кутами маркерів, які були виявлені в попередній рамці камери. Якщо раніше збережений кут і виявлена функція лежать в одному вікні 3 пікселі, існує велика ймовірність, що виявлена особливість є кутом заблокованого маркера. Ширина вікна становить лише 3 пікселі, отже, ймовірність невідповідності надзвичайно мала. Головною перевагою такого підходу до виявлення особливостей є те, що немає необхідності виявляти оклюзії ліній. Весь процес можна описати за таким алгоритмом:

IF раніше знайдений маркер не виявляється:
    знайти особливості з високим власним значенням
    FOR кожного раніше виявленого кута маркера:
        порівняти всі особливості з кутом
        IF раніше виявлений кут маркера та об’єкт лежать у вікні 3 пікселі:

позначте об’єкт як кут оклюзованого маркера

FOR кожного знайденого оклюзованого маркера:

обчислювальна матриця перетворення

Виявлено, що цей підхід також може бути використаний для усунення проблем з різкими змінами умов освітлення. Ці зміни освітлення можуть порушити процес виявлення, але як тільки стаціонарний маркер буде знайдений, процес виявлення не залежить від умов освітлення.

2.8 Розпізнавання маркера SURF-Golay

Перед тим як розглянути метод SURG-Golay, спочатку коротко розглянемо недоліки звичайного методу SURF і чому його краще не використовувати в сучасній розробці [6].

- Метод має чутливість, яку потрібно постійно налаштовувати під різні умови;
- Метод може давати помилкові спрацювання;
- При побудуванні методу, використовуються емпіричні коефіцієнти, які також потребують налаштування під різні умови;
- Метод має різні передумови для розпаралелювання та оптимізації.

Рисунок 2.17 – Приклад відображення контрольних точок на макері, методом SURF

Як можна побачити на даному зображенні є купа контрольних точок, які відображаються у вигляді кіл різного розміру, а також перекривають одна одну, через що не видно взагалі, які саме ключові точки має маркер. Для прикладу наведемо маркер, на якому є ключові точки, який використовується в іншій технології.

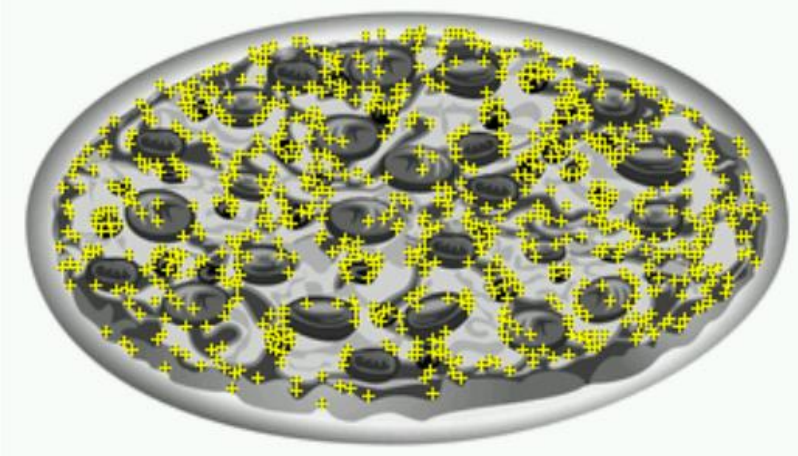

Рисунок 2.18 – Приклад відображення контрольних точок в технології Vuforia

У даній технології чітко видно всі контрольні точки, вони помічені жовтими хрестиками і не перекривають одна одну.

Чому це так важливо? Саме головне, при розробці та використанні додатків з доповненою реальність, це не розпізнати маркер, а вміння слідкувати за ним, гасити помилкові спрацювання та аномальні викиди точок, які може давати метод SURF, а також розроблювати «трекінг». По зображенням вище видно, як відбувається трекінг за допомогою цього методу, спілкувати за таким маркером дуже тяжко, тож даний метож не є гарним варіантом для використання.

Також розглянемо такий гібридний метод виявлення маркерів – SURF-Golay (S-G) [6] – в якому шаблон маркера розділений на дві частини: межу маркера та вміст маркера рис 2.19. Ці дві частини шаблону можна поєднувати незалежно. Вміст маркера складається виключно із зображення коду Голея. Маркерна межа складається з довільних геометричних фігур. Для ідентифікації маркера

використовується лише вміст маркера.

Першим кроком зіставлення маркерів є виявлення точок на обробленому зображенні за допомогою стандартного детектора SURF. Потім вектори дескрипторів для кожної ключової точки знаходять за допомогою екстрактора функцій SURF. Ці вектори узгоджуються шляхом обчислення евклідової відстані між кожною парою точок. Далі використовується фільтр симетричного узгодження для покращення відповідних пар ключових точок. Далі, фільтрується набір ключових точок, застосовуючи кутовий фільтр. Для кожного набору точок, виявлених кутовим фільтром, ми обчислюємо матрицю гомографії для корекції перспективних деформацій.

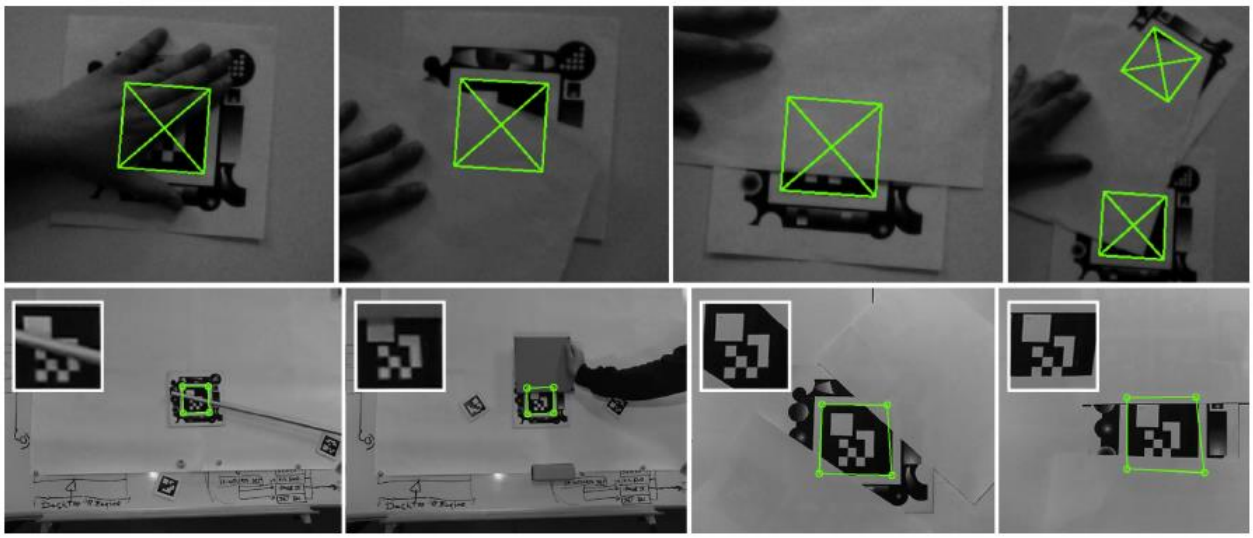

Рисунок 2.19 – Оклюзії. Приклади виявлення оклюзованих маркерів на основі кутів показані в першому рядку. Другий рядок представляє можливості методу S-G для виявлення оклюзованих маркерів. Останні два зображення показують дуже добре оклюзовані маркери (до 60%).

Застосувавши перетворення гомографії до зображення, можна визначити код Голея. Перспективно виправлене зображення обрізається та обробляється детектором коду Голея. Якщо знайдено код Голея, це означає, що ідентифікований весь маркер. Ця ідентифікація дає дуже важливий зворотний зв'язок для виявлення маркера SURF [6]. Враховуючи надійність детектора Голея, помилкові спрацьовування практично неможливі. Тому, якщо код ідентифікований, ми

можемо бути впевнені, що це один із пошукових маркерів. Це також означає, що гомографія була обчислена правильно. Це важливе поліпшення надійності виявлення, яке не використовується в інших методах. Взагалі кажучи, ненадійна ідентифікація маркерів є одним із ключових недоліків методів, заснованих на ознаках.

Метод виявлення S-G орієнтований на стійкість проти оклюзій маркерів. Його надійність визначається використанням детектора SURF, який шукає на зображенні ключові точки. Ці ключові точки розподіляються рівномірно на межі маркера. Незважаючи на високу асиметричність, маркер оброблений так, що всі частини однаково важливі. Тому не має значення, яку частину (будь-який кут чи бік) маркера закрито. Боротьба з оклюзіями на межі маркера є природною для цього методу виявлення, тому не потрібна спеціальна обробка. Зміст маркера (код виправлення помилок Голея) дещо більш чутливий до оклюзій, але все ще здатний мати справу з реалістичними оклюзіями (пальці, показчики). Здатність методу виявлення S-G справлятися з оклюзіями не залежить від руху маркера; він працює однаково для стаціонарних та нестаціонарних маркерів.

2.9 Адаптивне перемикання

Очевидно, що два методи, описані вище, доповнюють один одного. Тому був розроблений адаптивний метод перемикання. Оскільки реалізований додаток працює в режимі реального часу, використовувані методи виявлення орієнтовані на продуктивність. Основний потік програми завантажує зображення камери і відправляє його в механізм візуалізації. Тому частота кадрів потоку камери не залежить від продуктивності відстеження. Незважаючи на те, що може спостерігатися невелике відставання між рухом маркера та регулюванням 3D-об'єкта, загальна зручність використання була збільшена. Частота кадрів фонового візуалізації становить приблизно 30 кадрів в секунду на машині з використанням

OpenGL (Intel Core i5 3,8 ГГц, 16 ГБ оперативної пам'яті, SSD, Nvidia GeForce GTX 1060 6 Gb).

Одночасно потік виявлення на основі морфології завантажує останнє зображення камери і запускає процес ідентифікації маркера (приблизно 14 кадрів в секунду з одним виявленим маркером). Як тільки маркери знайдені, список виявлених маркерів надається в механізм візуалізації та оновлюються позиції 3D-моделі. Потім цей процес повторюється для наступних відеокадрів. Список виявлених маркерів також порівнюється зі списком маркерів, знайдених у попередньому кадрі. Якщо раніше виявлений маркер відсутній у поточному кадрі, запускається новий потік виявлення. Цей потік застосовує метод виявлення S-G, який дозволяє знайти навіть помітно закритий маркер.

Наступний кадр камери знову аналізується методом, заснованим на морфології, по-перше, хоча існує велика ймовірність того, що маркер все-таки закритий. Це може здатися марною тратою ресурсів, але існує проблема у формулюванні умови, яка дозволяє додатку надійно ввімкнути метод швидкого відстеження. Тим не менше, час потенційно непотрібного морфологічного аналізу становить менше 10% від загального споживання часу (продуктивність S-G становить приблизно 2 кадрів в секунду).

Якщо говорити в цілому, то для створення доповненої реальності існує два підходи. Перший підход використовує уже підготовані маркери, які повинні заходитись на якомусь предметі, тобто повинні бути роздруковані, таким чином при наведенні камери, користувач буде бачити якийсь візуальний ефект, для другого підходу немає необхідності роздруковувати маркери. Для реалізації цих підходів використовуються алгоритми комп'ютерного зору, вони необхідні для того, щоб розпізнати об'єкт в кадрі, та доповнити його візуальними ефектами.

Із вище описаних методів, самим надійним є Surf-Golay, а узагальнений алгоритм розпізнавання маркера є самим простим в розумінні та використанні, але кожен з них має своє місце в сфері доповненої реальності і користуєтья популярністю.

3. ОГЛЯД БІБЛІОТЕК ДЛЯ РОЗРОБКИ ДОДАТКІВ З ВИКОРИСТАННЯМ AR ТЕХНОЛОГІЇ

3.1 Бібліотеки компютерного зору та доповненої реальності

У цьому розділі буде коротко розглянуто декілька платфор та бібліотек, які дозволяють розроблювати додатки з використанням технології доповненої реальності.

3.1.1 Vuforia SDK

Vuforia Engine – це найбільш широко використовувана платформа, для розробки додатків з використанням доповненої реальності, а також інструментарій розробника. Вона підтримує провідні телефони, планшети та окуляри. На рисунку 3.1 зображено результат розробки на цій платформі.

Рисунок 3.1 – Результат роботи додатку, розробленому за допомогою платформи Vuforia

За допомогою цієї платформи, можна легко додати розширену функціональність комп’ютерного зору. Вона підтримує Unity Engine, а також три основні мобільні платформи: Android, iOS та UWP. Також Vuforia має посилання на документацію Vuforia Engine API, яка включає в себе API C# для Unity, API C++ для iOS, Android та UWP.

3.1.2 Metaio SDK

Metaio SDK це також один з наборів інструментів, за допомогою яких розроблюються додатки з використанням доповненої реальності. Даний набір оптимізований при підтримці і допомозі Intel під її платформи. Він є безкоштовним, а також доступний на таких платформах як Android, iOS, ПК платформи та підтримує 2D, 3D та SLAM трекінг.

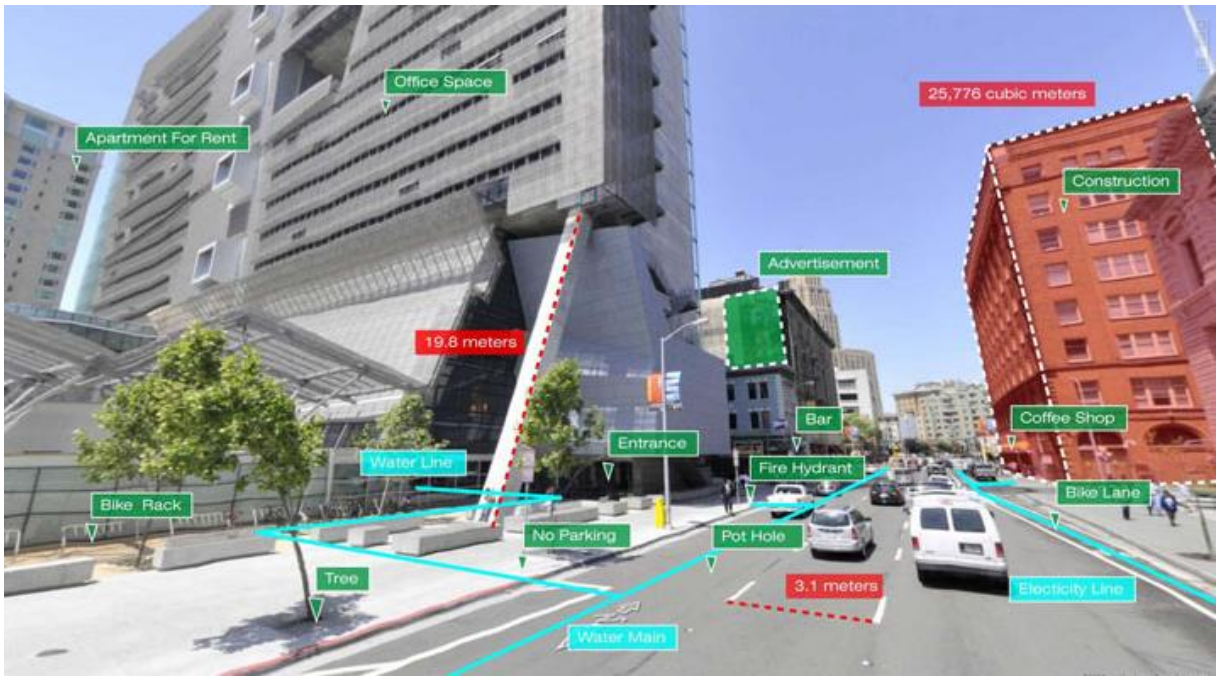

Рисунок 3.2 – Приклад роботи Metaio SDK

Набір Metaio SDK лежить в основі таких додатків, як каталог продуктів IKEA, McDonalds McMission та інші. Він є досить популярним в сфері дополненої реальності, але головна його перевага це безпека, тому що це єдиний SDK, який здійснює надійний безмаркерний трекінг трьохмірних об’єктів реального світу.

На момент версії Metaio SDK 5.0, даний набір мав такі характеристика:

- трекінг по обличчям;
- за допомогою шейдерів, покращена якість візуалізації;
- підтримка 3D моделей та безмаркерний трекінг.

3.1.3 AForge.Net

AForge.Net це фреймвор написаний на мові C#, який створений для розробки додатків у галузі комп’ютерного зору, штучного інтелекту, найронних мереж, машинного навчання, робототехніки та багато іншого [13].

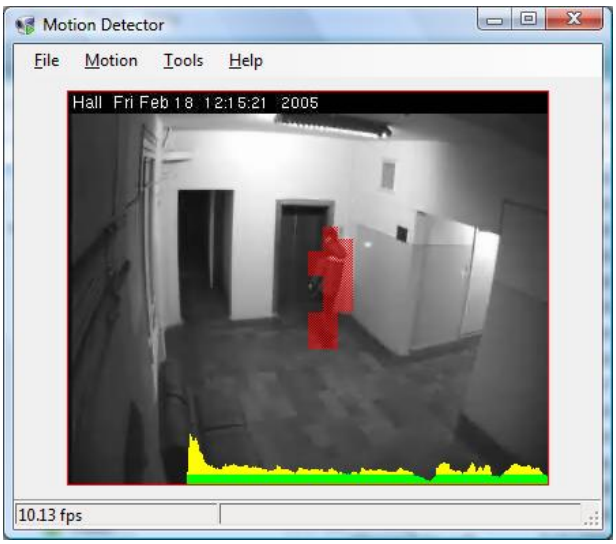

Рисунок 3.3 – Детектор руху написаний за допомогою AForge.Net

Структура даного фреймворку складається з набору бібліотек, кожна з яких має свої особливості і предназначені для різних функцій:

- AForge.Vision це бібліотека комп’ютерного зору;
- AForge.Video це набір бібліотек для обробки відео;
- AForge.Imaging це бібліотека для обробки зображень;
- AForge.Robotics це бібліотека, що забезпечує підтримку деяких наборів робототехніки [14].

Даний фреймворк надається не лише з різними бібліотеками, а і з багатьма зразками програм, які демонструють використання фреймворку, а також файлами документації.

3.2 OpenCV бібліотека для комп’ютерного зору, з відкритим кодом

Теорія комп’ютерного зору є основною для розвитку технологій доповненої реальності, і перш за все в галузі використання маркерів. Основний напрямок даної дисципліни – це аналіз та обробка зображень та відео. Алгоритми комп'ютерного зору дозволяють виділяти ключові особливості на зображенні, наприклад кути, контури та краї, а також проводити пошук фігур і об'єктів в реальному часі, виконувати 3D реконструкцію з кількох фотографій і багато іншого.

В області доповненої реальності алгоритми комп'ютерного зору використовуються для пошуку в відеопотоці спеціальних маркерів. Залежно від завдання, в якості маркера можуть виступати як спеціально сформовані зображення, так і особи людей. Після знаходження маркера в відеопотоці і обчисленні його місця розташування, з'являється можливість побудови матриці проекції і позиціонування віртуальних моделей. За допомогою них можна накласти віртуальний об'єкт на відеопотік таким чином, що буде досягнутий ефект присутності. Основна складність якраз і полягає в тому, щоб знайти маркер, визначити його місце розташування в кадрі і спроектувати відповідним чином віртуальну модель.

За останнє десятиліття була створена велика теоретична база в сфері обробки зображень та пошуку на ньому різних об'єктів. Перш за все, це стосується методів контурного аналізу, template matching, feature detection і генетичних алгоритмів. З точки зору побудови доповненої реальності найчастіше використовуються останні два підходу.

Оскільки теорія комп’ютерного зору є настільки важливою в доповненій реальності, було б помилково не розгянути одну з найпопулярніших бібліотек OpenCV.

На даний момент, OpenCV є дійсно найпопулярнішою та найбільш розвиненою бібліотекою комп’ютерного зору.

В склад цієї бібліотеки входять такі модулі, як:

- обробка зображень;
- побудова простих користувацьких інтерфейсів, а також загрузка та збереження відео і аудіо файлів;
- калібрування камери, пошук стерео-відповідностей та елементи обробки трьохвимірних даних;
- методи та моделі машинного навчання.

Дуже багато проектів, які стосуються комп’ютерного зору, використовують саме цю бібліотеку. Також варто відмітити що велика кількість бібліотек доповненої реальності, також використовують алгоритми, створені в OpenCV.

У 2014 році в OpenCV з’явились маркери під назвою ArUco. Ці маркери застосовуються для оцінки позиціонування та калібковки камери. Якщо говорити з більш научної точки зору, то цей маркер називається фідуціалом, тобто це об’єкт, який розміщується в полі зору системи візуалізації. Це може бути будь що, розміщене на зображенні, або одна мітка або набір міток в сітці оптичного приладу. Так виглядають приклади маркерів ArUco.

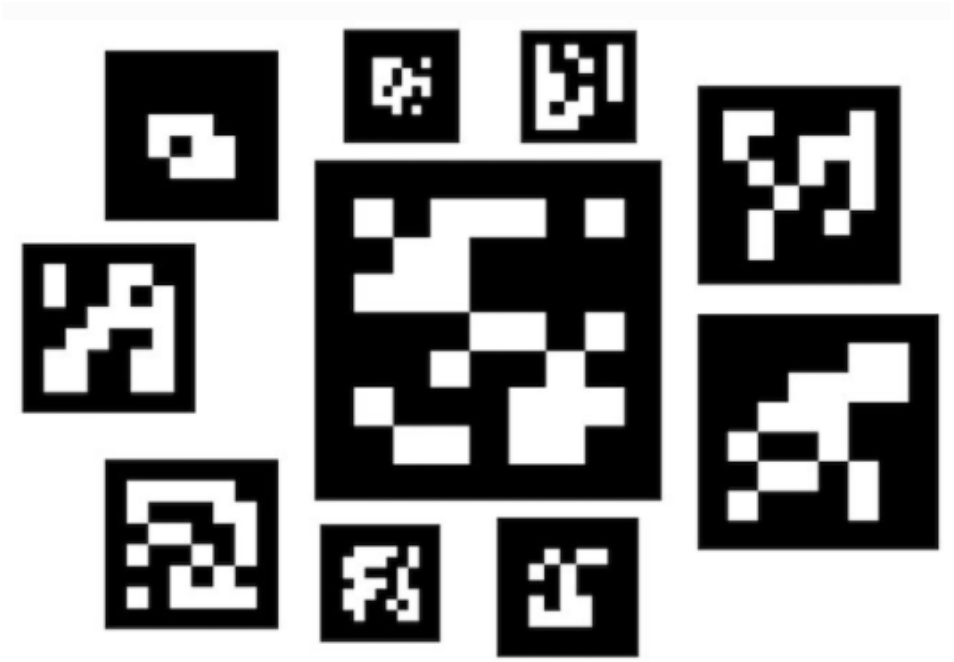

Рисунок 3.4 – Маркери ArUco

Цей маркер представляє з себе чорний квадрат з яскраво вираженими контурами, в якому розміщені білі деталі, за допомогою яких він ідентифікується. А чорна границя полегшує його пошук.

Ідея цих маркерів полягає в тому, що розпечатавши їх на реальні об'єкти та зробивши фото реального світу, можна знайти ці маркери і накласти поверх них доповнену реальність.

Рисукно 3.5 – Використання маркерів в реальому світі

На рисунку 3.5 видно, як маркери розміщені по кутам картини, і коли відбувається ідентифікація маркерів, то зображення замінюється на інше зображення, або можна замінити його на відео.

Такий спосіб використовується також в робототехніці. Якщо помістити маркери на шляху складського робота та якому встановлена камера, то коли камера знайде один з маркерів, то є можливість точно зрозуміти місце знаходження робота. А оскільки кожен маркер має унікальний ідентифікатор, то ми точно знаємо де знаходиться кожен з маркерів.

3.3 JSARToolKit

JSARToolKit це JavaScript бібліотека, яка призначена для відсліжування AR маркерів на відео. По суті це нова версія ARToolKit бібліотеки, за основу якої був взятий ARToolKit v5.3.1, скомпільований з мови програмування C++ на JavaScript, головна особливість JSARToolKit, це можливість запускати його у браузерах. Це означає, що додаток доповненої реальності буде працювати без встановлення и на приладі будь якого формату.

Рисунок 3.6 – Приклад роботи JSARToolKit

За допомогою ARToolKit та three.js, а також однією з самих популярних бібліотек WebGL, є можливість відображати доповнену реальність з 3D-графікою, через стек з повністю відкритим вихідним кодом, саме за допогою цього стеку технологій буде розроблено протопит [15].

WebGl – це бібліотека, яка розроблена для мови програмування JavaScript, за допомогою якої можливо створювати інтерактивну 3D-графіку, яка виконується майже в усіх веб-браузерах. А за рахунок використання засобів OpenCV, частина коду може виконуватись на відеокартах, що значно підвищить роботу додатку [16].

На даний час існує велика кількість бібліотеки, які виконують різні фукнції в сфері доповненої реальності та комп'ютерного зору. Основою майже усіх бібліотек є OpenCV, алгоритми цієї бібліотеки є дуже популярними, тож майже усі бібліотеки використовують їх.

Кожна з бібліотека виконує різні функції та назначена для різних цілей, для розробки мобільних додатків, веб-додатків, ігр з використанням доповненої реальності, додатків для навчання та інші, тож дуже тяжкл порівнювати їх між собою, але всі вони в даний момент користуються великою популярністю в розробкі додатків з доповненою реальністю.

4. РОЗРОБКА ПРОТОТИПУ

4.1 Програмна реалізація додатку з використання JSARToolKit

Для програмної реалізації, була вибрана бібліотека JSARToolKit5 з WebRTC API getUserMedia за допогомою якої створюються веб-додатки доповненої реальності. Також було використано технологію WebGL для обробки зображення, через її високу ефективність.

Якщо говорити про JSARToolKit5 – це порт JavaScript останньої версії бібліотеки доповненої реальності ARToolKit. JSARToolKit5 можна використовувати для додавання об’єктів у зображення та відео, щоб вони мали вигляд, ніби вони є частиною реальної сцени. Також є можливість зробити віртуальні об’єкти інтерактивними та анімованими [8].

Щоб продемонструвати типи програм, які ви можливо створювати за допомогою JSARToolKit5, розробимо невеликий додаток з використанням AR технології, який доповнює зображення з камери вашого пристрою [16].

4.2 Архітектура додатку

Для того, щоб функціонування даного додатку було вдалим, необхідо мати такі модулі:

- камера, цей модуль гарантує, що перегляд кадрів фіксується і передається на трекер. Необхідно лише ініціалізувати камеру, що транслювання почалось та зупинилось.
- конвертер зображення, потрібен для того, щоб перетворити кадри з камери у необхідний формат для рендерингу WebGL.
- трекер, цей модуль необхідний для відстеження об’єктів реального світу. Він містить в собі алгоритми комп’ютерного зору.

- відрисовка текстури, цей модуль потрібний для того, щоб створити зображення, яке буде зберігатись в об’єкті.

- база маркерів, необхідно створити об'єкт маркерів, де будуть знаходитись маркери, які буде розпізнавати додаток.

Всі ці елементи повинні бути проіціалізовані в логіці програми.

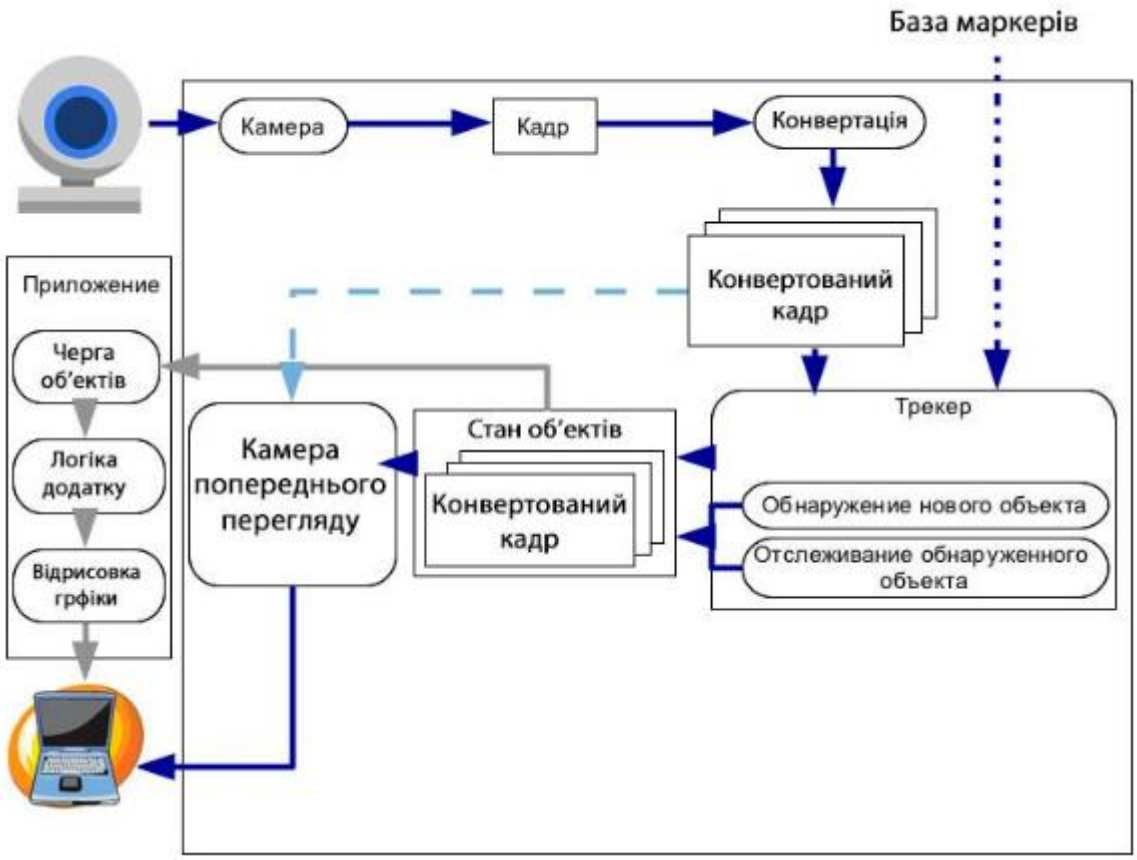

Рисунок 4.1 – Архітектура додатку

4.3 Розробка додатку

JSARToolKit працює, відстежуючи спеціальні зображення у вашому відео. Ці спеціальні зображення називаються маркерами AR, і JSARToolKit може зрозуміти, де вони знаходяться на відео та в якому напрямку вони спрямовані. Отримавши позиції та орієнтацію маркерів AR від JSARToolKit, можна намалювати 3D-об’єкти

поверх зображення в потрібних місцях, щоб здавалося, що об’єкти є частиною зображення.

Щоб завантажити JSARToolKit, необхідно виконати наступний код.

```
<script src="build/artoolkit.min.js"></script>
```

Рисунок 4.2 – Завантаження ARToolKit

Для того, щоб створити додаток AR, потрібен маркер AR, зображення та спосіб намалювати 3D-графіку поверх зображення. Для маркера буде використано спеціальний вбудований маркер, який називається BarcodeMarker. Зображення буде отримано з камери пристрою за допомогою API getUserMedia. А для створення 3D-графіки буде використано Three.js. Перше, що необхідно зробити – це створити джерело зображення. Для цього буде використано API getUserMedia, щоб отримати URL-адресу для камери пристрою, а потім використати цю URL-адресу як джерело для зображення. Завдяки цьому ми маємо зображення, яке показує камера пристрою.

Самий простий спосіб зробити все це – використати допоміжну функцію в JSARToolKit5: ARController.getUserMedia. Зворотний виклик onSuccess в об’єкті параметрів викликається за допомогою готового до використання елемента.

```
var video = ARController.getUserMedia({
        maxARVideoSize: 320, // do AR processing on scaled down video of this size
        facing: "environment",
        onSuccess: function(video) {
                console.log('got video', video);
        }
});
```

Рисунок 4.3 – Створення джерела зображення

Далі потрібно створити ARController. ARController відстежує маркери, які зареєстровані, і зчитує пікселі з джерела зображення.

```
var arController = new ARController(video, 'Data/camera_para.dat');
arController.onload = function() {
        console.log('ARController ready for use', arController);
};
```

Рисунок 4.4 – Створення ARController

Для повторного використання напишемо скрипт, завантаження камери ARCameraParam.

```
var camera = new ARCameraParam('Data/camera_para.dat');
camera.onload = function() {
        var arController = new ARController(video.videoWidth, video.videoHeight, camera);
        console.log('ARController ready for use', arController);
};
```

Рисунок 4.5 – Завантаження камери

Після того, коли було підключено відео до менеджера, потрібно розпочати відстеження маркерів AR у зображенні. Побачивши маркер AR, необхідно отримати його деталі та з’ясувати, де він знаходиться і в який бік він вказує.

```
// Set the ARController pattern detection mode to detect barcode markers.
arController.setPatternDetectionMode( artoolkit.AR_MATRIX_CODE_DETECTION );

// Add an event listener to listen to getMarker events on the ARController.
// Whenever ARController#process detects a marker, it fires a getMarker event
// with the marker details.
//
var detectedBarcodeMarkers = {};
arController.addEventListener('getMarker', function(ev) {
        var barcodeId = ev.data.marker.idMatrix;
        if (barcodeId !== -1) {
                console.log("saw a barcode marker with id", barcodeId);

                // Note that you need to copy the values of the transformation matrix,
                // as the event transformation matrix is reused for each marker event
                // sent by an ARController.
                //
                var transform = ev.data.matrix;
                if (!detectedBarcodeMarkers[barcodeId]) {
                        detectedBarcodeMarkers[barcodeId] = {
                                visible: true,
                                matrix: new Float32Array(16)
                        }
                }
                detectedBarcodeMarkers[barcodeId].visible = true;
                detectedBarcodeMarkers[barcodeId].matrix.set(transform);

        }
});

var cameraMatrix = arController.getCameraMatrix();

for (var i in detectedBarcodeMarkers) {
        detectedBarcodeMarkers.visible = false;
}

// Process a video frame to find markers in it.
// Each detected marker fires a getMarker event.
//
arController.process(video);
```

Рисунок 4.6 – Відстеження маркеру

ARToolKit дбає про всю математику і дає зручну матрицю перетворень. Щоб отримати маркери на зображенні.

Оскільки вже отримано позиції маркерів з JSARToolKit, далі потрібно скопіювати їх до об’єктів Three.js. За допомогою Three.js можна зробити анімовані об’єкти, зробити так щоб вони зникали коли маркер виходить з поля зору камери та з’являлись коли маркер знаходиться в полі зорі камери. Також можна надати логіку, щоб при натисканні клавіші миші на об’єкт відбувались якісь дії.

```
<script src="build/artoolkit.min.js"></script>
<script src="js/artoolkit.three.js"></script>
<script>
ARController.getUserMediaThreeScene(
        facing: 'environment',
        onSuccess: function(arScene, arController, arCameraParam) {

                arController.setPatternDetectionMode(artoolkit.AR_MATRIX_CODE_DETECTION);

                // Track the barcode marker with id 20.
                // markerRoot is a THREE.Object3D that tracks the marker position.
                //
                var markerRoot = arController.createThreeBarcodeMarker(20);

                // Add the openable box to the marker root.
                //
                var box = createOpenableBox();
                markerRoot.add(box);

                // Add the marker root to the AR scene.
                //
                arScene.scene.add(markerRoot);


                // Add event handlers to make the box open/close on tap.
                //
                window.addEventListener('touchend', function(ev) {
                        if (box.hit( ev.touches[0], arScene.camera )) {
                                box.toggleOpen();
                        }
                }, false);

                window.addEventListener('mouseup', function(ev) {
                        if (box.hit( ev, arScene.camera )) {
                                box.toggleOpen();
                        }
                }, false);
```

Рисунок 4.7 – Підключення бібліотека Three.js

В цілому виконавши ці дій вже можна використовувати додаток. Як варіант можна не використовувати бібліотеку Three.js, а виконувати всі інтеграції вручну, і це навіть не принесе більш роботи, просто буде потрібно купа шаблонів, без яких можна обійтися, використовуючи Three.js.

У результаті виконання, буде отримано таке зображення. Зліва сторони зображення предмета, про який потрібно отримати додаткову інформацію, з правої сторони маркер, на який буде накладено доповнену реальність.

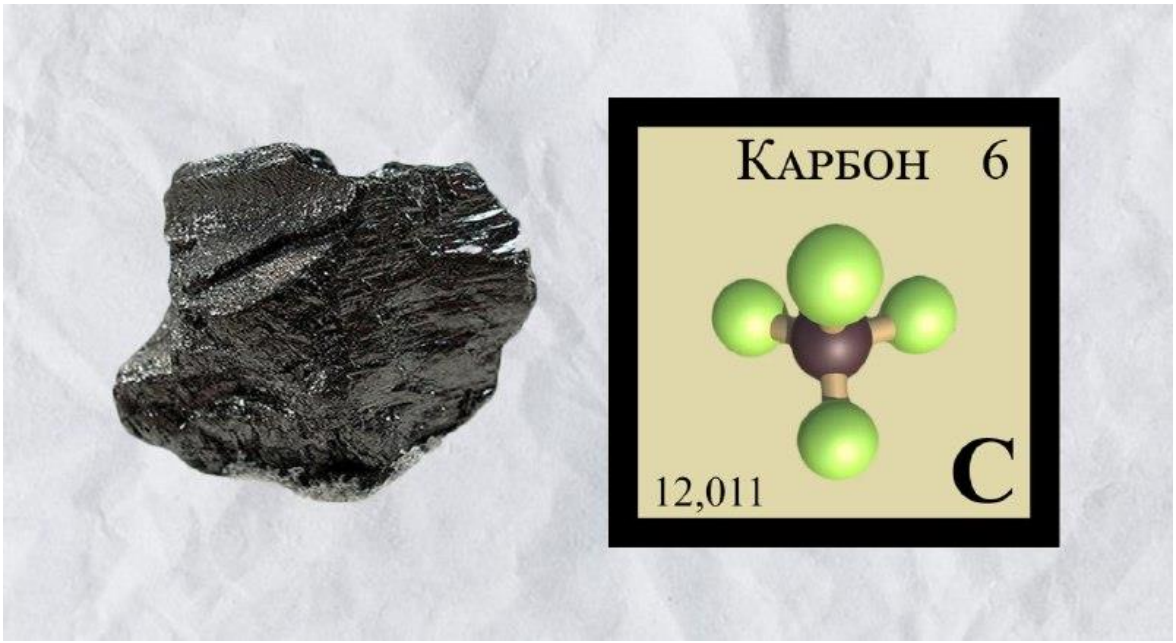

Рисунок 4.8 – Результат виконання додатку

У результаті роботи, було створено веб-додаток, який розпізнає маркери, порівняємо його з іншим схожим додатком, який було створено з використанням інших технологій.

На рисунку 3.1 зображено додаток, який створений на двигуні Unity 3D з використанням інструменту розробника Vuforia SDK. І він має суттєві недоліки в порівнянні з додатком, зображеним на рисунку 4.8. У першу чергу, для його розробки потрібно багато часу працювати в графічному редакторі, накладувати 3D моделі поверх зображення і в ручну робити трекінг, на відміну він розробленого додатку, де достатньо в коді вказати яке зображення відноситься до якого маркеру. Також потрібно для кожного зображення знайти відповідну картинку, за яку буде чіплятися програма, в даному ж випадку, є база з маркерами, кожен з яких має свій id і необхідно просто підв’язати зображення до маркера по його id. Також суттєвий плюс цього додатку який розроблено на JSARToolKit це те, що його можливо тестувати сидячи за ПК з браузеру, на відміну від іншого додатку, нову версію

якого постійно потрібно завантажувати на мобільний пристрій. Окрім того, що є можливість накладувати на маркери зображення, таким самим чином можна накласти відео файл, і транслювати його при наведені камерою на маркер. Також веб-додаток є більш надійним.

Такий додаток можна використовувати в будь якій сфері, наприклад медицина, наука, військова сфера та інші, для опису будь який об’єктів.

Проаналізувавши всю роботу, можна сказати що технологія доповненої реальності почала займати велику нішу в ІТ сфері і зараз застосовується у різних областях. У даній роботі були показані приклади використання цієї технології в різних додатках та порівняно два додатки які розроблювались за допомогою різнизх технологій.

ВИСНОВКИ

У процесі даної дослідницької роботи були розглянуті теоретичні питання, пов'язані з різними алгоритмами розпізнавання маркерів, які використовуються в технології доповненої реальності, а саме:

- узагальнений алгоритм розпізнавання маркеру;
- алгоритм розпізнавання маркеру на основі ознак;
- алгоритм розпізнавання маркеру на основі країв;
- алгоритми SURF та SURF-Golay.

Кожен із цих алгоритмів має свої переваги і недоліки (особливо продуктивність та стійкість до оклюзій). Тому було запропоноване також адаптивне рішення, яке перемикає метод виявлення відповідно до різних умов.

Однією з ключових проблем є суттєва оклюзія маркера. Оклюзії маркера можуть виникати з ряду причин – фізична оклюзія маркера, відбиття світла, перешкоди світла, раптові зміни освітлення, шум камери тощо. Впроваджений метод, заснований на морфології, є швидким і точним, лише з незначними тривожними ефектами, такими як тремтіння 3D об'єкт. Якщо його доповнить метод, заснований на функціях відстеження у вікні 3 пікселі, він може мати справу зі значними оклюзіями. Однак мають бути виконані дві умови: маркер повинен бути нерухомим (отже, ні камера, ні маркер не можуть рухатися) і повинен бути видно принаймні один кут. Особливо першою передумовою є ключове обмеження цього рішення.

Навпаки, метод на основі SURF є природним стійким до оклюзій. Більше того, якщо розширити цей метод за допомогою коду виправлення помилок Golay дозволяє виявити велику кількість маркерів у зображенні камери. Тим не менше, його продуктивність значно нижча.

Представлена адаптивна комутація забезпечує значне покращення юзабіліті. Система розпізнавання маркерів пристосована до платформ з обмеженими ресурсами, наприклад різні мобільні пристрої. Більше того, цей адаптивний підхід

до перемикання методів виявлення дозволяє створювати більш складні стратегії відстеження. Наприклад, можна використовувати ефективне відстеження ефективності. Швидке виявлення аналізує ціле зображення камери, доповнюючи точний пошук методом лише в межах цікавих областей навколо раніше знайдених маркерів. Метою є мінімізація витрати часу, а також максимальна точність.

У результаті роботи розроблено додаток доповненої реальності, за допомогою якого можна отримати дані про об’єкт та вивести їх на дисплей пристрою та порівняно його з інших додатком який має схожий функціонал, але розроблений за допомогою інших технологій. Також було розглянуто сфери застосування такого додатку. За допомогою додатку користувач може отримувати дані про об’єкт.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Папагианнис Хелен. Дополненная реальность. Все, что вы хотели узнать о технологии будущего / Хелен Папагианнис., 2021. – 288 с (дата звернення: 15.03.2021).

2. Maevv. AR — Дополненная Реальность [Електронний ресурс] (Дата звернення 23.05.2019) / Maevv – URL: https://habr.com/ru/post/419437/ (дата звернення: 15.03.2021).

3. S. Siltanen. Theory and applications of marker-based augmented reality, 2012. – URL: http://www.vtt.fi/inf/pdf/science/2012/S3.pdf (дата звернення: 16.03.2021).

4. Косик Д.О. Маркери доповненої реальності – як можливість вдосконалення реального світу / Косик Д.О, Валенда Н.А. // Priority direction of Science and Technology Development // Тези – м. Київ – 2021 – с. 272-277.

5. A. C. Rice. Analyzing fundamental properties of marker-based vision system designs / A. C. Rice, R. K. Harle, A. R. Beresford., 2006. – 460 с. – (2) (дата звернення: 25.03.2021).

6. Surf: Speeded up robust features./ H. Bay, T. Tuytelaars, L. V. Gool. – 2006. – URL: researchgate.net/publication/225761164_SURF_Speeded_up_robust_features (дата звернення: 30.03.2021).

7. Augmented reality usage for prototyping speed up / J. Stastny, D. Procházka, T. Koubek, J. Landa. – 2011. – №2. – Режим доступу до ресурсу: https://www.researchgate.net/publication/50353588_Augmented_reality_usage_for_prototyping_speed_up (дата звернення: 30.03.2021).

8. H. Kato. A registration method based on texture tracking using ARToolKit / H. Kato, K. Tachibana, M. Billinghurst, M. Grafe. – 2003. – URL: https://www.researchgate.net/publication/4085551_A_registration_method_based_on_texture_tracking_using_ARToolKit (дата звернення: 05.04.2021).

9. Heikkinen I. Создание приложений дополненной реальности с помощью JSARToolKit [Електронний ресурс] / Ilmari Heikkinen. – 2012. – URL:

https://www.html5rocks.com/ru/tutorials/webgl/jsartoolkit_webrtc/ (дата звернення: 05.04.2021).

10. Bogdan Sus, Nataliia Tmienova, Oleksandr Bauzha, Taras Chaikivskyi, Ilona Revenchuk. Software System for Virtual Laboratory Works. XV International Scientific and Technical Conference Computer Science and Information Technologies - CSIT-2020, Zbarazh Castle, Ukraine.- 23-26 September, 2020.-P.396-400 (дата звернення: 11.04.2021).

11. Sus, B., Tmienova, N., Revenchuk, I., Bauzha, O., Stirenko, S. Gamification approach to the creation of virtual laboratory works and educational courses Sus, B., Tmienova, N., Revenchuk, I., Bauzha, O., Stirenko, S.CEUR Workshop Proceedings, 2020, 2711, с. 68-78 (дата звернення: 13.04.2021).

12. Richard S. Computer Vision: Algorithms and Applications / Szeliski Richard. – Лондон, 2010. – 979 с (дата звернення: 15.04.2021).

13. Smelyakov K., Chupryna A., Bohomolov O., Ruban I. The Neural Network Technologies Effectiveness for Face Detection // IEEE Third International Conference on Data Stream Mining & Processing (DSMP), August 21-25. – 2020. – P. 201-205 (дата звернення: 15.04.2021).

14. Bilous N.V., Rakova A. O. REFERENCE POINTS METHOD FOR HUMAN HEAD MOVEMENTS TRACKING // Радиоэлектроника, информатика, управление. – Запоріжжя: ЗНТУ, -2020 №3, pp.121-129 (Web of Science, фаховий) (дата звернення: 16.04.2021).

15. Mustafa, S.K. , Kopot, M., Ayaz Ahmad, M, Lyubchenko, V., Lyashenko, V. Interesting applications of mobile robotic motion by using control algorithms Mustafa, S.K. , Kopot, M., Ayaz Ahmad, M, Lyubchenko, V., Lyashenko, V. International Journal of Advanced Trends in Computer Science and Engineering, 2020, 9(3), с. 3847-3852 (дата звернення: 19.04.2021).

16. Taras Chaykivskyy. Експеримент із технологією доповненої реальності у вебі (front-end only) [Електронний ресурс] / Taras Chaykivskyy, Мартою Дідич. – 2016. – URL: https://dou.ua/lenta/articles/web-ar/ (дата звернення: 27.04.2021).