МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Т.в.о. завідувача кафедри
_____ Сафонова С.О.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Дослідження і реалізація методів побудови доповненої реальності для мобільних пристроїв

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 123 "Комп'ютерна інженерія"

Науковий керівник роботи:	_____ (підпис)	С.О. Сафонова (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Я.О.Критська (ініціали, прізвище)
Студент:	_____ (підпис)	Р.Ю. Усик (ініціали, прізвище)
Група:		КІ-18дм

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень "магістр"
Спеціальність 123 – "Комп'ютерна інженерія"
(шифр і назва)
Спеціалізація
(шифр і назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри комп'ютерної інженерії
д.т.н., доц. *С.О. Сафонова*
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Усику Роману Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження і реалізація методів побудови доповненої реальності для мобільних пристроїв

керівник проекту (роботи) Сафонова С.О., к.т.н., доцент
(прізвище, ім.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «11» 10 2019 р. № ____
2. Строк подання студентом роботи 15.01.2020
3. Вихідні дані до роботи Матеріали науково-дослідної практики.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд методів побудови доповненої реальності
2. Огляд алгоритмів і бібліотек комп'ютерного зору
3. Дослідження алгоритмів візуальної локалізації і трекінгу
4. Дослідження алгоритму PTAM – parallel tracking and mapping

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	*Критська Я.О.*		

7. Дата видачі завдання *06.09.2020*

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання до магістерської роботи	*10.09.19-15.09.19*	
2	Аналіз завдання, огляд літератури	*16.09.19-08.10.19*	
3	Аналіз технічних засобів	*09.10.19-20.10.19*	
4	Розробка методу	*21.10.19-06.11.19*	
5	Програмна реалізація	*17.11.19-06.12.19*	
6	Охорона праці	*07.12.19-15.12.19*	
7	Оформлення пояснювальної записки	*16.12.19-28.12.19*	
8	Підготовка презентації та доповіді	*03.01.20-13.01.20*	

Студент _____ *Усик Р.Ю.*
(підпис) (прізвище та ініціали)

Науковий керівник _____ *Сафонова С.О.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Усик Р.Ю. Дослідження і реалізація методів побудови доповненої реальності для мобільних пристроїв.

Метою даної магістерської роботи є дослідження способів побудови доповненої реальності в природному оточенні на мобільних пристроях на основі візуального трекінгу і реалізація на основі вибраних алгоритмів бібліотеки для мобільних пристроїв на платформі Android з урахуванням особливостей платформи. Результатом роботи є програмна реалізація алгоритму PTAM та його адаптація для платформи Android.

Ключові слова: доповнена реальність, розпізнавання маркерів, генетичні алгоритми, feature detection, алгоритми комп'ютерного зору, алгоритм PTAM, побудова та оптимізація карти.

АННОТАЦИЯ

Усик Р.Ю. Исследование и реализация методов построения дополненной реальности для мобильных устройств.

Целью данной магистерской работы является исследование способов построения дополненной реальности в природном окружении на мобильных устройствах на основе визуального отслеживания и реализация на основе выбранных алгоритмов библиотеки для мобильных устройств на платформе Android с учетом особенностей платформы. Результатом работы является программная реализация алгоритма PTAM и его адаптация для платформы Android.

Ключевые слова: дополненная реальность, распознавание маркеров, генетические алгоритмы, feature detection, алгоритмы компьютерного зрения, алгоритм PTAM, построение и оптимизация карты.

ABSTRACT

Usyk R.Yu. Research and implementation of methods for building augmented reality for mobile devices.

The aim of certification diploma is to study the methods of construction of the augmented reality in natural surrounding on mobile devices on the basis of the visual tracking, and to realize library, based on the chosen algorithms of library for mobile devices on the platform Android taking into account the features of platform. The result of the work is the program implementation of the algorithm of PTAM and its adaptation for the platform of Android.

Key words: augmented reality, recognition of markers, genetic algorithms, feature detection, computer vision algorithms, algorithm PTAM, construction and optimization of the map.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AR (augmented reality) – доповнена реальність

OpenCV – бібліотека комп’ютерного зору з відкритим кодом

SLAM – Simultaneous Localization and Mapping

ВСТУП

Останній час активно розвиваються нові інтерфейси між людиною та машиною. Один з них – доповнена реальність, коли зображення з реального світу доповнюється віртуальними об’єктами. Найбільший вплив на її популярність зробив розвиток мобільних пристроїв – наближення їх обчислювальної потужності до настільних комп’ютерів та збільшення розподільної здатності дисплеїв та камер.

В мобільній доповненій реальності користувач дивиться на зображення, отримане з відеокамери на їх мобільному пристрої, та сцени, які вони спостерігають (реальний світ), доповнюються інтегрованими тривимірними віртуальними об’єктами. Ця технологія має величезний потенціал в багатьох областях.

Якщо віртуальний об'єкт просто накладається на реальне зображення, а не інтегрується в нього, то для створення середовища доповненої реальності можуть бути використані додаткові сенсори, присутні в сучасних мобільних пристроях, такі як акселерометр, компас, GPS. Використовуючи інформацію про місцезнаходження, користувач може переміщатися по світу доповненої реальності. Якщо віртуальні об'єкти мають безпосередній зв'язок з реальним світом, більший ніж просто глобальне положення, наприклад віртуальна будівля, побудована на реальному пустирі, то для такої доповненої реальності необхідна додаткова інформація, така як кордони пустиря і його розміри. Отримання цієї додаткової інформації зазвичай досягається за допомогою спеціальних маркерів або за допомогою спеціальних функцій розпізнавання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА АКТУАЛЬНІСТЬ

Доповнена реальність (augmented reality) – це технології, що дозволяють доповнювати зображення реальних об’єктів різноманітними об’єктами комп’ютерної графіки, а також поєднувати зображення, отримані з різних джерел: відеокамер, тепловізорів, спектрометрів тощо. На відміну від «віртуальної реальності», яка передбачає повністю штучний, синтезований світ (відеоряд), доповнена реальність впроваджує синтезовані об’єкти до природної відео сцени [1].

Хоча активно розвиток доповненої реальності почався з 90х років, першою технологією, яка дозволила створювати доповнену реальність у реальному часі на персональних комп’ютерах і інших платформах, та поширила її, була бібліотека ARToolKit.

1.1 Сучасні сфери застосування доповненої реальності

В рамках рекламних кампаній розробляються додатки, які відображають тривимірний контент при наведені камери смартфона на обкладинку журналу, рекламний буклет, коробку від піци або злітну смугу крізь вікно аеропорту (рис. 1.1).

Рисунок 1.1 – Застосування доповненої реальності в маркетингу

Додатки доповненої реальності дозволяють побачити замість звичайної листівки анімовану тривимірну сценку крізь дисплей смартфону, або оживити картину у галереї.

Багато дослідників вважають, що насправді технологія вже почала демонструвати свою істинну цінність завдяки спрощенню багатьох звичних для споживачів речей.

Наприклад, вже існують додатки по підбору зачісок і одягу, які широко використовуються в індустрії краси (рис. 1.2); у автомобільному секторі, де користувачі тепер можуть використати доповнену реальність, щоб зануритися в досвід водіння автомобілем, який вони хочуть придбати.

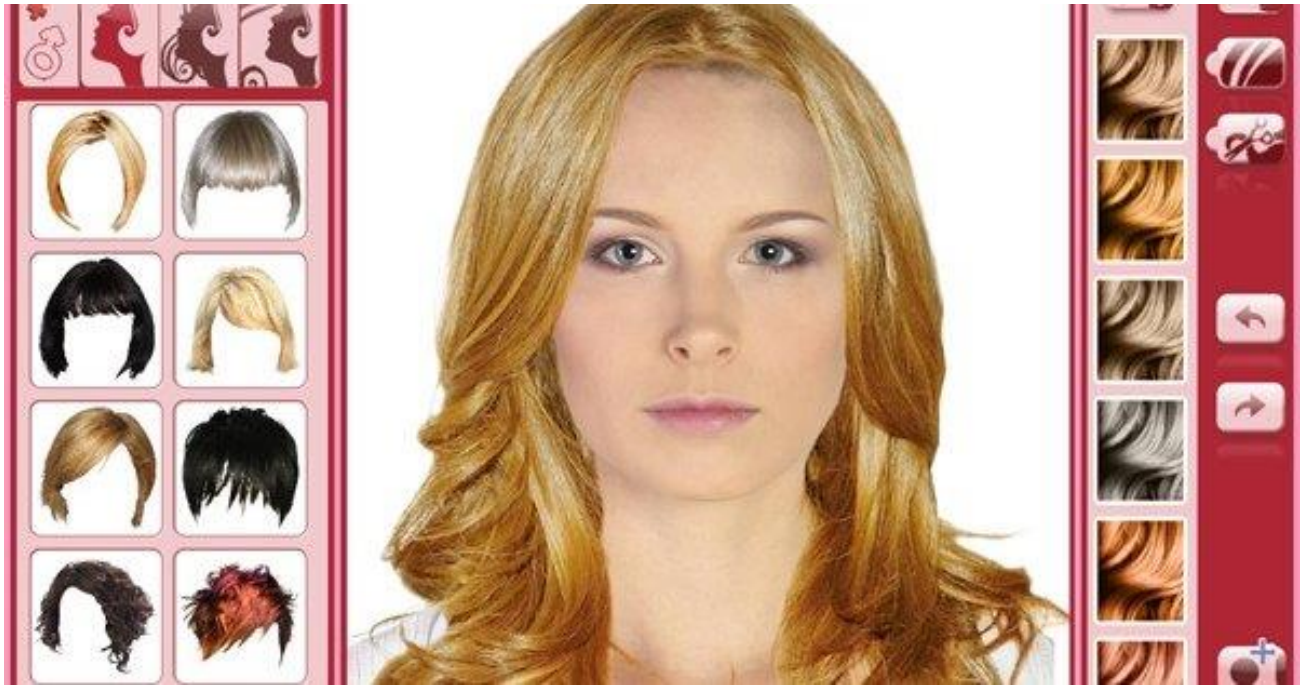


Рисунок 1.2 – Застосування доповненої реальності в індустрії краси

Деякі бренди, такі як Lego і Jurassic World, вже експериментують з технологією і відмічають, що її потенційна дія на аудиторію величезна. AR як і раніше має дивовижний вау-фактор. Прикладом тому є гра Pokemon Go, в якій досі мільйони гравців знаходять, захоплюють і тренують віртуальних істот, що з'являються на екрані, ніби вони знаходяться в тому ж реальному місці, що і гравець (рис. 1.3).

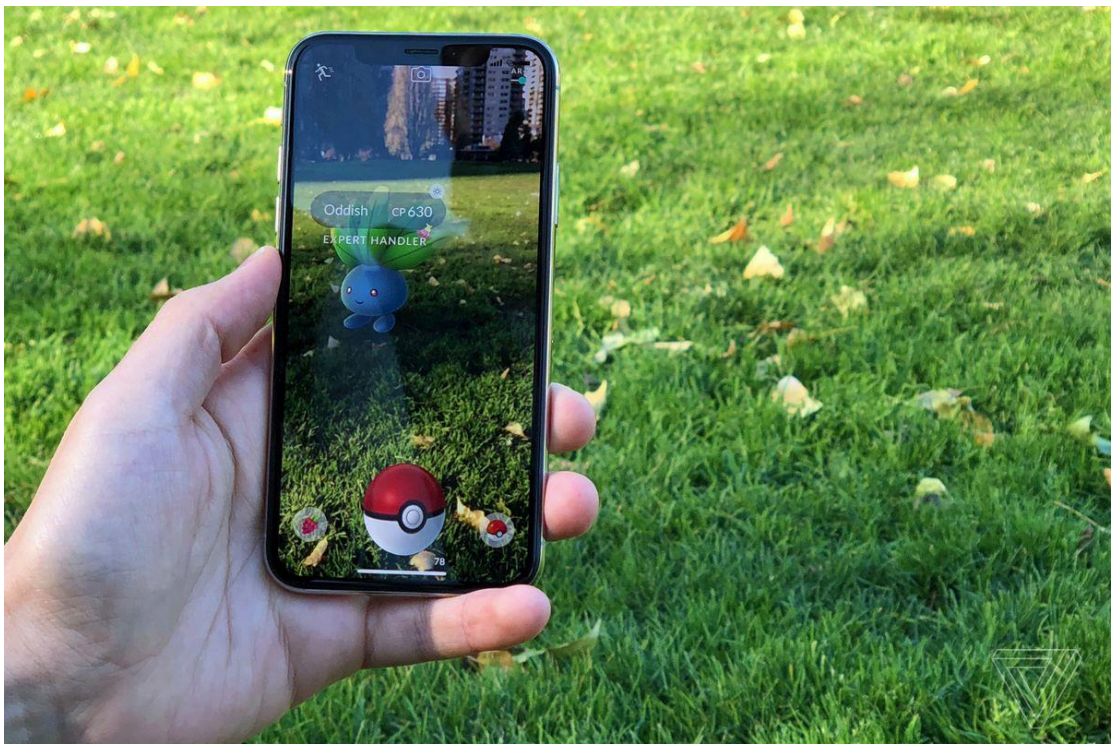


Рисунок 1.3 – Інтерфейс гри Pokemon Go

Провідні бренди і роздрібні торговці, такі як Sephora, Nestlé і Jaguar Land Rover, продемонстрували особливе лідерство в цій області. Вони експериментували з використанням AR для надання персональних консультацій, інформації про походження товарів або додаткових послуг для їх продуктів, що привело до успішних, надихаючих кампаній, які виходять далеко за рамки звичайних ігор.

А, наприклад, Ikea повністю інтегрувала AR у свій додаток, за допомогою якого дозволяє користувачам перевірити, як меблі можуть виглядати в їх будинках (рис.1.4).

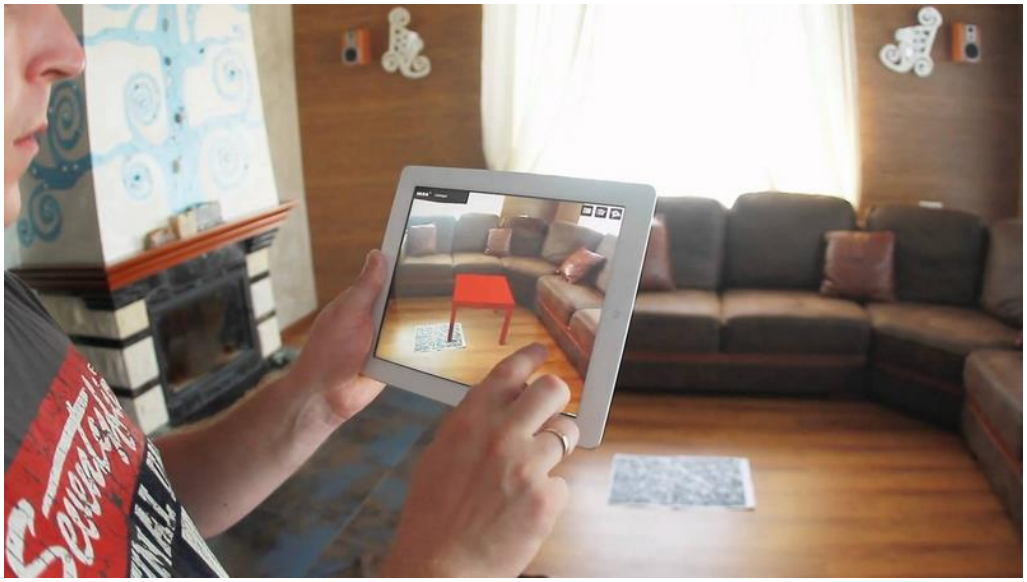


Рисунок 1.4 – Приклад роботи додатку від Ikea

У медицині додатки доповненої реальності застосовуються під час навчання та операцій з метою візуалізації внутрішніх органів або вказівок (рис. 1.5).

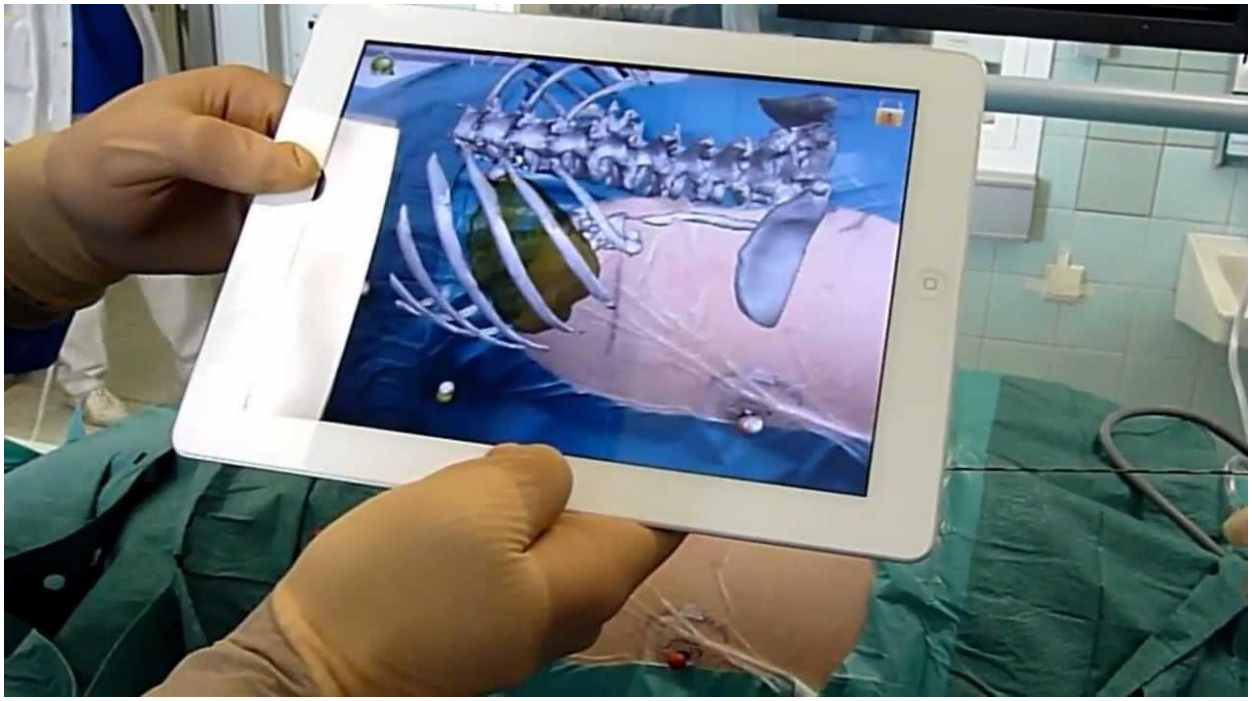


Рисунок 1.5 – Доповнена реальність в медицині

Цікавість до AR продовжує рости в геометричній прогресії. Тепер він підживлюється і штучним інтелектом, який дозволяє камерам "розуміти" світ і накладати на нього цифровий контент. У поєднанні з обладнанням, що стає потужнішим і легшим, найближчі роки будуть ключовими для розвитку доповненої реальності.

1.2 Історія розвитку додатків доповненої реальності

У 2008 році перші AR – додатки були створені для смартфонів, і люди по всьому світу змогли уперше скористатися новітньою технологією. Перший додаток призначався для користувачів Android, і це дозволило їм використати свої камери, щоб побачити на екрані різні об'єкти віртуальної реальності в 3d. Рішення незабаром з'явилося на iPhone, і запущено в якості навігаційного додатку, названого Wikitude Drive, який представляє собою щось схоже на Вікіпедію.

Wikitude використовує інтерфейс в стилі Google Maps і дає можливість зберігати географічні координати об'єктів, що становлять інтерес (по суті, геотеггінг), а потім робить цю інформацію доступною всім користувачам, які опиняться в тому ж місці.

"Wikitude працює подібно просунутому навігатору, який об'єднує в собі карту, GPS і путівник", – пояснює голова відділу широкосмугового доступу стільникової компанії T-Mobile Річард Уормслі.

Щоб додаток був корисним кому-небудь крім ентузіастів необхідний якісний контент, а не просто аматорські фотографії, надіслані блогерами, озброївшись смартфонами. Втім, додатки, що допомагають вирішувати специфічні завдання і надають різноманітний контент, існують.

Нині все більше компаній використовують доповнену реальність у своїх додатках. Найпростіше використати можливості AR в мобільних додатках – в цьому випадку апаратна основа вже є, залишається лише написати програмну частину.

Гарним прикладом (і одним з перших успішних комерційних проектів) можна назвати гру HoloGrid: Monster Battle (рис. 1.6). Це гра з набором карт, основна дія відбувається в доповненій реальності, де фігурами служать "живі" монстри. Ідея цієї гри з'явилася в 2016 році і тоді ж розробники змогли зібрати $100 000. Грати тут можуть дві людини, використовуючи два мобільні пристрої.

Рисунок 1.6 – Доповнена реальність у грі HoloGrid: Monster Battle

Є і освітні проекти, подані в ігровій формі. Один такий зараз збирає кошти на Kickstarter. Називається він Imagina Books.

Проект передбачає створення цілої серії освітніх книг, які адаптовані під доповнену реальність. Треба навести планшет або телефон з предвстановленим додатком на сторінку книги з ілюстрацією, наприклад, мозку, і цей орган можна розглядати з усіх боків на екрані (рис. 1.7).

Ще один освітній проект – Orboot. Це глобус, тільки не звичайний, а глобус доповненої реальності. Потрібно навести планшет або телефон з предвстановленим додатком на яку-небудь частину глобуса, і можна побачити тварин, які мешкають в цій частині світу, пам'ятки або страви, що популярні серед жителів (рис. 1.8).

Крім того, користувач додатка познайомиться з винаходами, створеними в різний час представниками тієї або іншої країни, чудесами природи або культурою і мистецтвом різних країн. Власне, усе це може бути цікаво не лише для дітей, але і для дорослих.

Рисунок 1.7 – Доповнена реальність у додатку Imagina Books

Рисунок 1.8 – Доповнена реальність у додатку Orboot

Ще один цікавий проект – це аквадрон Ziphius. Цілком реальний, а не віртуальний пристрій, який відмінно тримається на воді (рис. 1.9). Керується він за допомогою мобільного телефону або планшета. Але програмне забезпечення оснащене функцією доповненої реальності, яка дозволяє перетворити звичайний заплив дрона на пару сотень метрів в справжню пригоду. Він може бути піратом, ніндзя і навіть акробатом.

З ним можна влаштовувати запливи на перегонки, а можна грати в доповненій реальності. До речі, дрон має можливість спостерігати за реальністю як під водою, так і над її поверхнею.

Рисунок 1.9 – Аквадрон Ziphius

Врешті решт, всім відомі маски зі Snapchat або Instagram також є технологією доповненої реальності.

В цілому, є багато інших проектів з AR-сфери, але вони практично нічим не відрізняються від тих, що описані вище. Найчастіше це ігри, рідше – щось матеріальне, як правило – окуляри або шолом (Google Glass – яскравий приклад такого роду проектів). Із загального ряду вибивається водяний дрон, але це - швидше виключення. Як би то не було, екосистема додатків і обладнання для AR поступово росте. З урахуванням того, що

Apple починає бурхливу роботу в цьому напрямі, вже найближчим часом можна чекати проникнення AR до багатьох сфер нашого життя.

1.3 Можливості технологій доповненої реальності в області розробки і проектування

Потенціал технологій доповненої реальності у сфері розробки і проектування колосальний. За допомогою віртуальної реальності користувачі можуть занурюватися в повністю віртуальний світ, тоді як доповнена реальність дозволяє працювати в контексті реального світу.

Хоча іноді в проектуванні задіяний лише один об'єкт, існує безліч ситуацій, коли продукт повинен взаємодіяти з навколишнім світом і потенційними користувачами. Саме в цьому випадку доповнена реальність може принести чималу користь.

Поки що переважно використовуються фізичні прототипи, і так буде ще довгий час. Але у багатьох ситуаціях цифровий продукт, показаний в кінцевому варіанті його використання, може виявитися набагато ефективніше.

Прикладом можуть послужити рішення, що взаємодіють з людиною або навколишніми продуктами. Можливість оцінювати повномасштабний інтерактивний прототип в контексті і взаємодіяти з ним, будь то промислове устаткування, продукти архітектурного масштабу або модернізовані системи і продукти, відкриває необмежені можливості.

Цікаво, що деякі компанії-постачальники та виробники товарів і послуг під своєю торговою маркою (вендори) вже пробують використати дисплеї з підтримкою доповненої реальності і технології ручної взаємодії (тобто за допомогою рук, а не контролера).

Поєднуючи подібні рішення з відповідним устаткуванням, можно відкрити нові способи взаємодії з геометричними моделями і виконувати симуляції за межами плоских дисплеїв з високим розділенням, щоб досягти набагато більших результатів.

Ще одна область, в якій очікується активне застосування технологій доповненої реальності, – це обробка даних в середовищі виробництва продуктів і надання послуг.

Наприклад, в сучасних заводських цехах намагаються впроваджувати електронні засоби обробки і відображення даних, але ця тенденція обмежена існуючими плоскими дисплеями. З розвитком технологій доповненої реальності виникає ряд можливостей, що здатні докорінно змінити цей процес.

Зараз існують технології, які дозволяють накладати складні тривимірні інструкції, безпосередньо пов'язані з фізичним об'єктом. Уявіть робітника на лінії складання, що слідує вказівкам по складанню продукту, причому необхідні дії не просто показуються на гарнітурі, а відображаються безпосередньо на головному дисплеї разом з додатковою інформацією.

Хоча для розгортання технологій доповненої реальності потрібно нескладні компоненти, для їх підготовки все одно потрібні потужні обчислювальні ресурси.

Доповнена реальність багато в чому відрізняється від віртуальної. Її суть в об'єднанні реального і віртуального світів. В результаті можливості цих технологій в області розробки, проектування і виробництва відрізняються. Тоді як за допомогою віртуальної реальності користувачі можуть занурюватися в повністю цифровий світ, доповнена реальність дозволяє доповнювати і розширювати реальний світ за допомогою цифрових даних. Саме завдяки можливості використання в контексті реального світу ця технологія виглядає багатообіцяюче.

З поширенням підключених пристроїв незабаром збільшиться об'єм інформації, яку зможемо використати як при проектуванні, так і при експлуатації. З'являються рішення, за допомогою яких користувачі можуть збирати ці дані, аналізувати і фільтрувати їх для отримання цінних відомостей, необхідних в тій або іншій ситуації.

Багато хто вважає, що технології віртуальної реальності отримають щонайширше поширення з появою рішень, що дозволяють користувачам передавати цю інформацію на місце проведення робіт. Такі рішення вже розробляються. Але потенціал цих технологій украй великий навіть на більш ранніх етапах життєвого циклу продукту, при його розробці.

Можливість досліджувати продукт так, неначе він вже був зроблений, одразу на робочому місці, може змінити галузь кардинальним чином. При цьому базова вимога до використання точних і реалістичних тривимірних даних не виключається, а, навпаки, стає основоположною для усього рішення.

Досвід і навички, отримані за останні два десятиліття в області симуляції, рендерінга і візуалізації продуктів, застосовуватимуться і розширюватимуться. І хоча гарнітури доповненої реальності стають самостійними і не вимагають для використання пов'язаних пристроїв, залишиться потреба у високопродуктивних робочих станціях, що дозволяють створювати матеріали для відображення і взаємодії.

Потенціал ринку програм і облаштувань введення доповненої реальності величезний. Будь-яке явище має три стадії розвитку: поява, розквіт і захід. Зараз технологія Augmented Reality знаходиться на першому етапі.

За кордоном доповнена реальність використовується вже десять років, але доки не отримала масового поширення. Європа, Америка, Японія і інші країни стоять на порозі другого етапу розвитку, а ринки країн СНД доки ще на початку шляху. Проте інтерес до нових розробок в цій області є, і є фахівці, які мають креативні ідеї і готові створювати програми.

1.4 Постановка задачі дослідження

Метою цієї роботи є дослідження методів побудови доповненої реальності в природному оточенні на мобільних пристроях на основі візуального трекінгу і реалізація на основі вибраних алгоритмів бібліотеки для мобільних пристроїв на платформі Android з урахуванням особливостей платформи.

В якості вирішуваних завдань можна виділити наступні:

– аналіз сучасних сфер використання доповненої реальності;

– аналіз сучасних технологій доповненої реальності;

– дослідження існуючих методів побудови доповненої реальності;

– вивчення готових рішень і бібліотек для побудови доповненої реальності і аналіз можливостей, які вони надають;

– вивчення алгоритмів, що дозволяють здійснювати в реальному часі трекінг і побудову карти оточення з достатньою продуктивністю і вибір відповідних;

– реалізація бібліотеки з урахуванням особливостей вибраної платформи;

– реалізація збереження мінімальної кількості інформації про місце, достатньої для подальшої точної локалізації в ньому;

– оцінка якості трекінгу і продуктивності отриманого рішення;

– дослідження способів прив'язки отриманої карти оточення до реальних місць і можливості автоматичної побудови карти оточення з використанням заздалегідь доступної інформації;

– дослідження способів поліпшити стійкість локалізації до змін в оточенні.

2 МЕТОДИ ПОБУДОВИ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

Можна виділити два головних принципи побудови доповненої реальності :

– на основі маркера;

– на основі координат місця розташування користувача.

Безмаркерні технології часто застосовуються в мобільних пристроях і будуються за допомогою спеціальних датчиків.

Можливі декілька підходів до реалізації доповненої реальності без маркерів. Один з них використання гіроскопа телефону. Ця реалізація дозволяє відстежувати координати об'єктів і здійснювати коректну взаємодію з ними, проте великий мінус полягає в тому, що відсутня будь-яка інформація про реальний світ.

Інший варіант - це використання методу одночасної локалізації і побудови карти SLAM (англ. Simultaneous Localization and Mapping). Цей метод надає повну інформацію про навколишній світ і можливість відстежувати як свої координати, так і координати об'єктів.

Розглянемо побудову доповненої реальності за допомогою маркерів і алгоритмів комп'ютерного зору. Під маркером розуміється об'єкт, розташований в навколишньому просторі, який знаходиться і аналізується спеціальним програмним забезпеченням для подальшої відрисовки віртуальних об'єктів. На основі інформації про положення маркера в просторі програма може досить точно спроектувати на нього віртуальний об'єкт, від чого буде досягнутий ефект його фізичної присутності в навколишньому просторі (рис. 2.1).

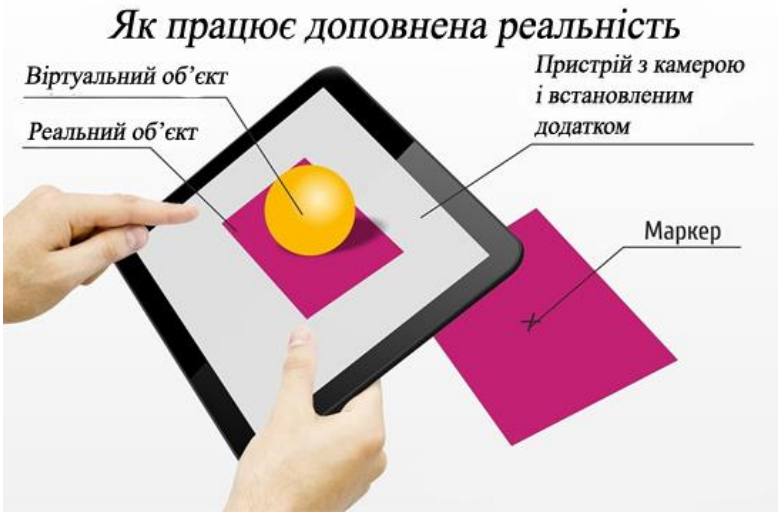


Рисунок 2.1 – Принцип роботи доповненої реальності з маркером

Використовуючи додаткові графічні фільтри і високоякісні моделі, віртуальний об'єкт може стати практично реальним і важко відмітним від інших елементів інтер'єру або екстер'єру.

Досить часто в ролі маркера виступає аркуш паперу з деяким спеціальним зображенням. Тип малюнка може варіюватися досить сильно і залежить від алгоритмів розпізнавання зображень. Кількість маркерів досить широка: ними можуть бути і геометричні фігури простої форми (наприклад, круг, квадрат), і об'єкти у формі прямокутного паралелепіпеда, і навіть очі і обличчя людей (рис. 2.2).

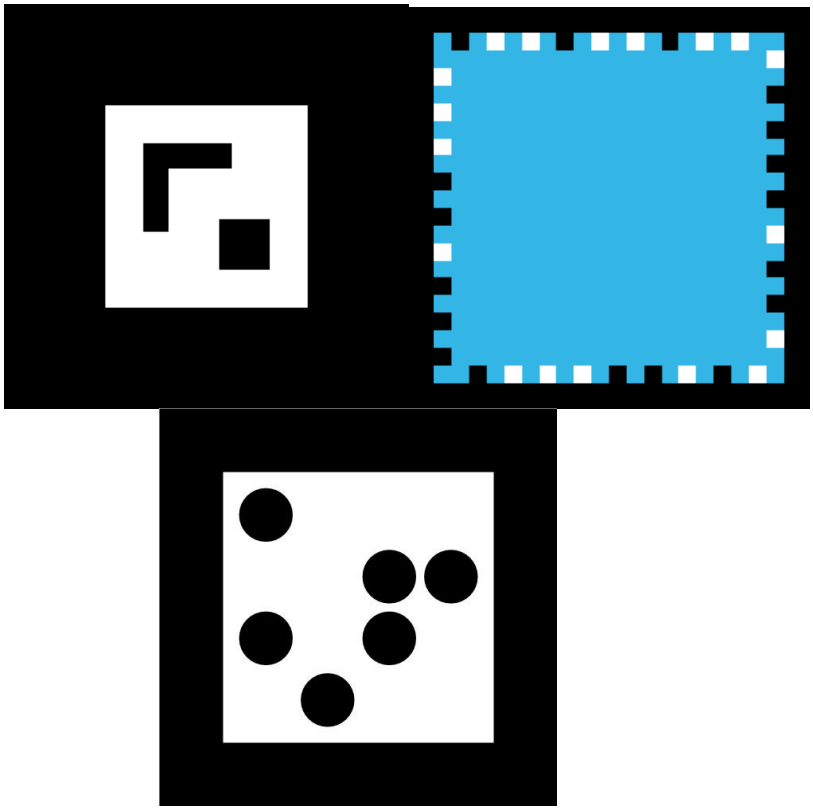


Рисунок 2.2 – Приклади маркерів доповненої реальності

2.1 Методика розпізнавання маркерів

Маркером може бути будь-яка фігура (об'єкт). Але на даному етапі розвитку техніки досить часто бувають обмеження: роздільна здатність камери (наприклад, веб-камери), особливості передачі кольору, освітлення. Окрім того, увесь процес роботи додатку відбувається у реальному часі, тож необхідні доволі потужні обчислювальні засоби.

Для того, щоб знизити навантаження вибирається чорно-білий маркер простої (наприклад, квадратної або прямокутної) форми з вписаним у нього ідентифікатором-образом (рис. 2.3, 2.4)

Рисунок 2.3 – Institut Graphische Datenverarbeitung (IGD) маркери

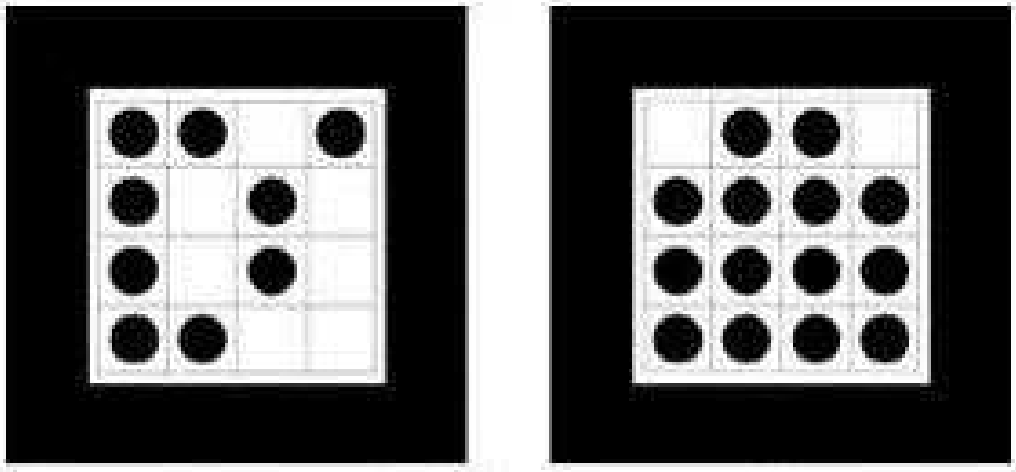


Рисунок 2.4 – Siemens Corporate Research (SCR) маркери

У будь-якому випадку алгоритм розпізнавання маркеру виглядає наступним чином [1]:

1) переведення зображення у градацію сірого;
2) бінарізація зображення (знаходження порогу);
3) визначення замкнутих областей;
4) виділення контурів;
5) виділення кутів маркеру;
6) перетворення координат.

Кожен з пунктів методики є окремим науковим завданням. Для передачі кольорового зображення у градації сірого можливо використовувати наступні формули. Вони не потребують великих обчислювальних ресурсів.

Переведення у градацію сірого, використовуючи властивість Світлоти (Lightness):

$$GS = \frac{max(R,G,B) + min(R,G,B)}{2}. \quad (2.1)$$

Переведення у градацію сірого, використовуючи властивість Світимості (Luminosity):

$$GS = 0{,}21 * R + 0.72 * G + 0.07 * B. \quad (2.2)$$

Переведення у градацію сірого, використовуючи середнє значення (Average):

$$GS = \frac{R+G+B}{3}. \quad (2.3)$$

Для переведення зображення у двокольоровий стан використовується певний поріг. Для вибору порогу варто скористатися гістограмою кольору. Авжеж, можна задати поріг вручну, але тоді знижається точність і подальша робота буде або дуже складною, або взагалі неможливою.

Усі методи перетворення зображення у чорно-білий стан можна розділити на шість великих груп [2]:

1) методи, засновані на формі гістограми;
2) методи на основі кластерізації;
3) методи на основі вивчення ентропії;
4) методи, засновані на пошуку схожості між сірим і чорно-білим зображенням;
5) методи, що використовують кореляційні залежності та особливості статичного розподілу між пікселями в областях зображення;
6) методи, засновані на локальній адаптації порогу для кожного пікселю зображення.

В ході аналізу було виявлено, що на практиці краще використовувати метод локальної адаптації (рис. 2.5).

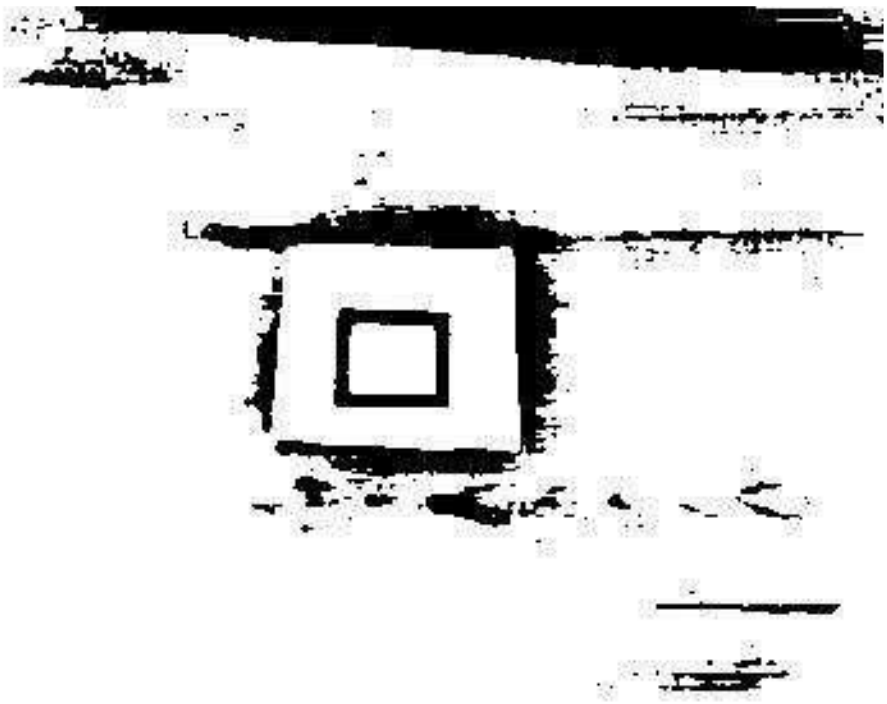


Рисунок 2.5 – Результати роботи методу локальної адаптації

Для визначення замкнутих ділянок на білому фоні використовується комбінація з алгоритмів заливки «білих» областей та виділення контурів (рис. 2.6) [3].

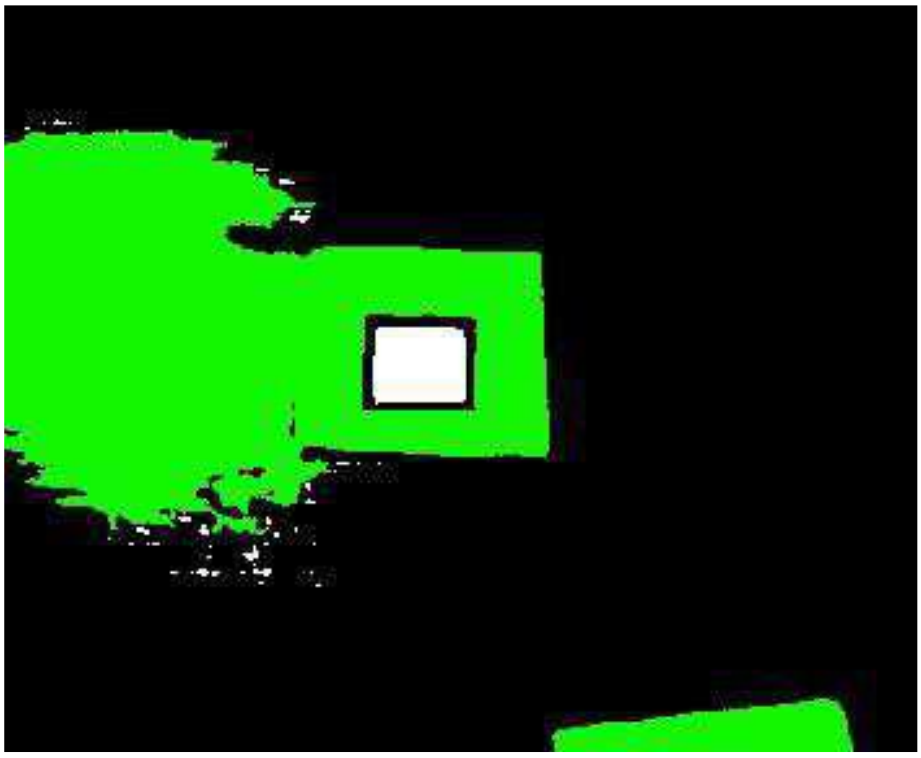


Рисунок 2.6 – Результати роботи заливки «білих» областей

Зараз для виділення контурів використовують близько шести основних методів:

1) Marr-Hildreth Edge Detector;

2) Canny Edge Detector;

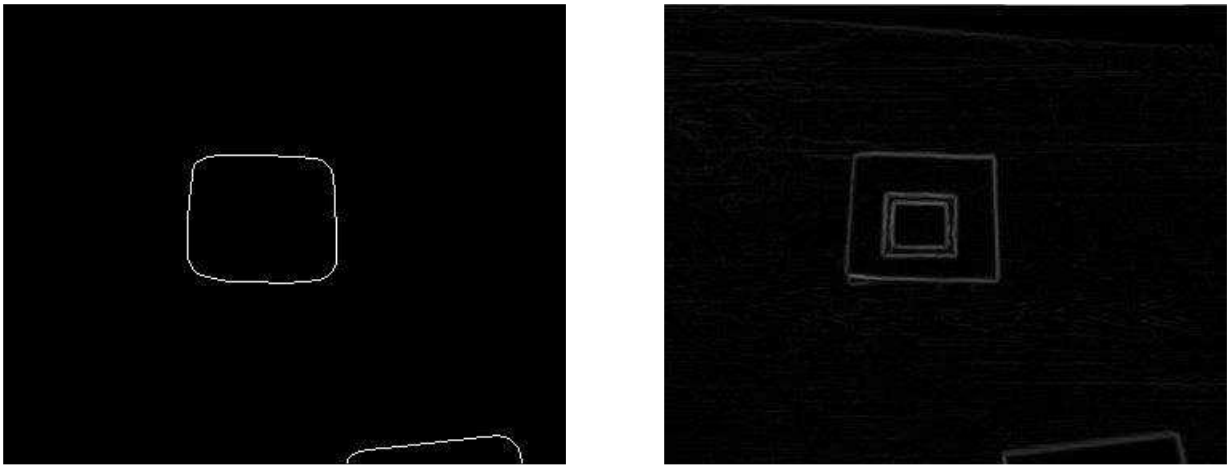
3) Boolean function based Edge Detector;

4) Euclidian distance and Vector Angle based Edge Detector;

5) Depth Edge Detection using Multi-Flash Imaging;

6) Sobel Edge Detector.

У ході роботи використовувалися алгоритми Собеля та Канні (рис. 2.7 (а,б)).



а б

Рисунок 2.7 – Результати роботи алгоритмів: а – Канні; б – Собеля

Для того, щоб зіставити отримані контури з наявним маркером можна використати алгоритм Рамера-Дугласа-Пекера (алгоритм ітеративної найближчої точки, алгоритм розбиття та злиття), який дозволяє зменшити кількість точок кривої, апроксимованої більшою серією точок. В OpenCV є функція approxPolyDP, яка реалізує заданий алгоритм.

Таким чином, визначили координати кутів маркеру, які у ідеалі є перпендикулярними, а у реальності розташовані під іншим кутом. Окрім того, у ідеалі і в реальності, сторони квадрату є осями координат. Таким чином, можна визначити положення камери стосовно заданого об'єкта та точку відліку початку координат (рис. 2.8) [4].

Ідея даного методу у тому, що при зміні кута, з якого дивиться камера, змінюється розмір проекції.

Комбінації розглянутих методів і алгоритмів дозволяють провести розпізнавання маркеру та його перетворення у додатках доповненої реальності. При розробці цих додатків постійно доводиться шукати компроміси: від технічних обмежень (отримання

сигналу з веб-камери) до швидкості виконання алгоритмів та обробки зображень (так як перетворення виконується у реальному часі).

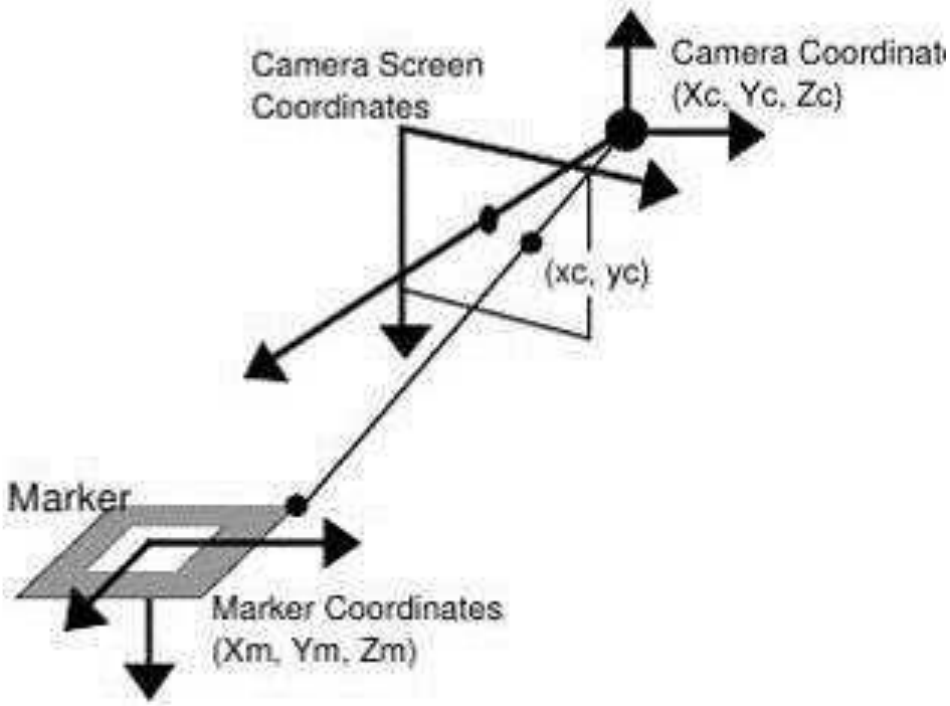


Рисунок 2.8 – Метод визначення координат

2.2 Алгоритми комп’ютерного зору для побудови доповненої реальності

Теорія комп'ютерного зору (англ. computer vision) є основоположною для розвитку технологій доповненої реальності, і передусім в області використання маркерів. Основний напрям цієї дисципліни – це аналіз і обробка зображень (у тому числі і відеопотоку).

Алгоритми комп'ютерного зору дозволяють виділяти ключові особливості на зображенні (кути, межі області), здійснювати пошук фігур і об'єктів в реальному часі, виконувати 3d реконструкцію за декількома фотографіями і багато що інше.

У області доповненої реальності алгоритми комп'ютерного зору використовуються для пошуку у відеопотоці спеціальних маркерів. Залежно від завдання маркером можуть виступати як спеціально сформовані зображення, так і обличчя людей [6].

Після знаходження маркера у відеопотоці і обчисленні його місця розташування з'являється можливість побудови матриці проекції і позиціонування віртуальних моделей. За їх допомогою можна накласти віртуальний об'єкт на відеопотік таким чином, що буде досягнутий ефект присутності. Основна складність 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 у сфері обробки зображень і пошуку на ньому різних об'єктів. Передусім, це стосується методів контурного аналізу, template matching, feature detection і генетичних алгоритмів. З точки зору побудови доповненої реальності часто використовуються останні два підходи. Розглянемо кожний з них.

2.2.1 Генетичні алгоритми

Генетичні алгоритми – це евристичні алгоритми пошуку, які використовуються для вирішення завдань оптимізації і моделювання шляхом випадкового підбору, комбінування і варіації параметрів, що шукають, з використанням механізмів, які нагадують біологічну еволюцію.

У комп'ютерному зорі вони використовуються для пошуку об'єкту деякого заданого класу на статичному зображенні або відеопотоці. Спочатку необхідно провести навчання алгоритму за допомогою двох різних наборів зображень:

– зображення, що містять потрібний об'єкт;

– неправдиві зображення без об'єкту, що шукають.

При цьому для навчання використовується велика кількість зображень, і чим їх більше – тим краще працюватиме алгоритм. Для кожної картинки робиться виділення різних ключових особливостей: межі, лінії, центральні елементи (рис.2.9).

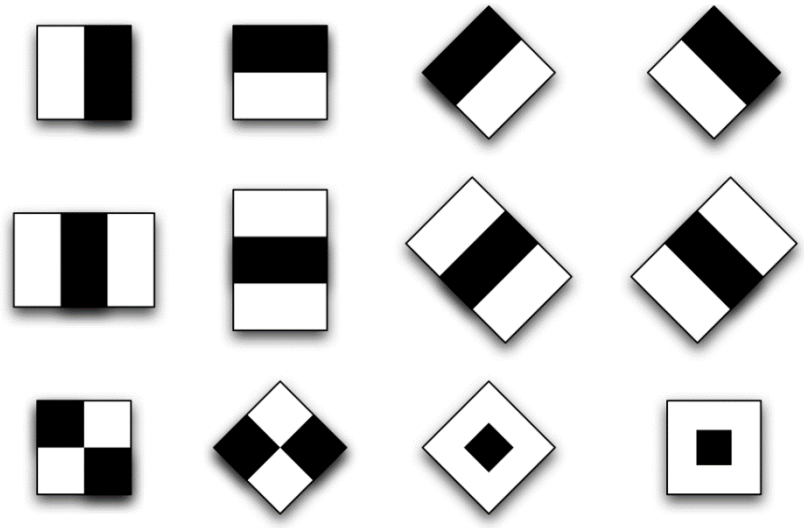


Рисунок 2.9 – Примітиви Хаару, що використовуються в алгоритмі

За їх допомогою робиться побудова статистичної моделі, яка потім використовується для пошуку об'єкту на зображенні [7].

Прикладом використання цього підходу може служити алгоритм розпізнавання осіб і очей на відеопотоці. Поступово навчаючи алгоритм, можна добитися високих результатів знаходження заданого класу об'єктів. Проте необхідність навчання якраз і робить використання генетичних алгоритмів досить проблематичним. Для коректної роботи потрібна істотна кількість різних зображень (що містять потрібний об'єкт та що не містять), тому час побудови класифікатора для кожного об'єкту може займати тривалий час.

2.2.2 Алгоритм feature detection

Концепція feature detection в комп'ютерному зорі відноситься до методів, які націлені на обчислення абстракцій зображення і виділення на ньому ключових особливостей. Ці особливості можуть бути як у вигляді ізольованих точок, так і кривих або пов'язаних областей. Не існує строгого визначення того, що таке ключова особливість зображення. Кожен алгоритм розуміє під цим своє (кути, грані, області і тому подібне).

Часто для пошуку маркерів використовуються алгоритми, які виконують пошук і порівняння зображень за ключовими точками. Ключова точка – це деяка ділянка картинки, яка є відмітною для заданого зображення (рис. 2.10). Що саме береться за цю точку – безпосередньо залежить від використовуваного алгоритму.

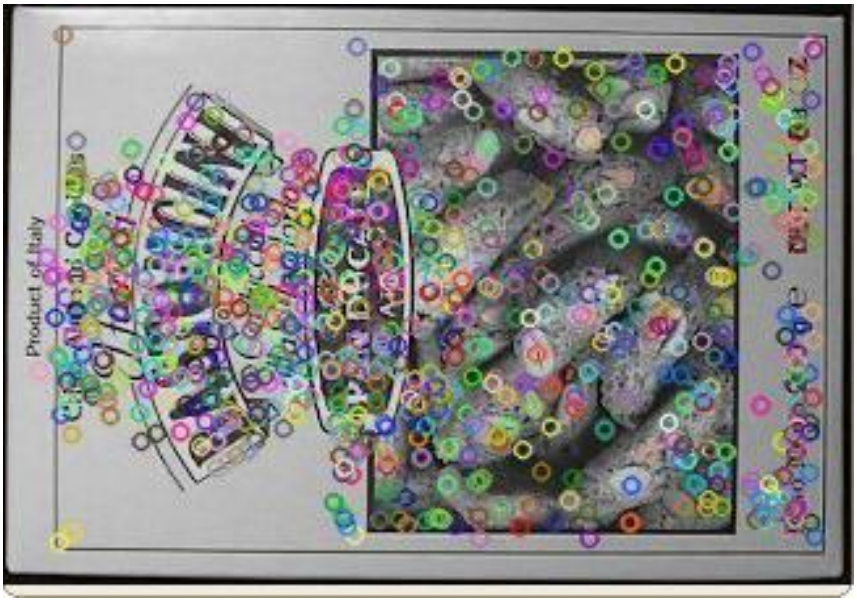


Рисунок 2.10 – Приклад ключових точок на зображенні.

Для їх знаходження і подальшого порівняння використовуються три складові:

– детектор (англ. feature detector) – здійснює пошук ключових точок на зображенні;

– дескриптор (англ. descriptor extractor) – робить опис знайдених ключових точок, оцінюючи їх позиції через опис навколишніх областей;

– матчер (англ. matcher) – здійснює побудову відповідностей між двома наборами точок.

Спочатку за допомогою детектора здійснюється пошук ключових точок шаблонного (шуканого) зображення. Отримані точки потім описуються за допомогою дескриптора. Ця інформація зберігається в окремий файл (чи базу даних), щоб не виконувати цей процес знову. При обробці відеопотоку з метою пошуку заданого шаблону описаний процес виконується для кожного кадру (за винятком збереження даних). Для встановлення відповідності між ключовими точками і дескрипторами застосовується матчер (рис. 2.11).

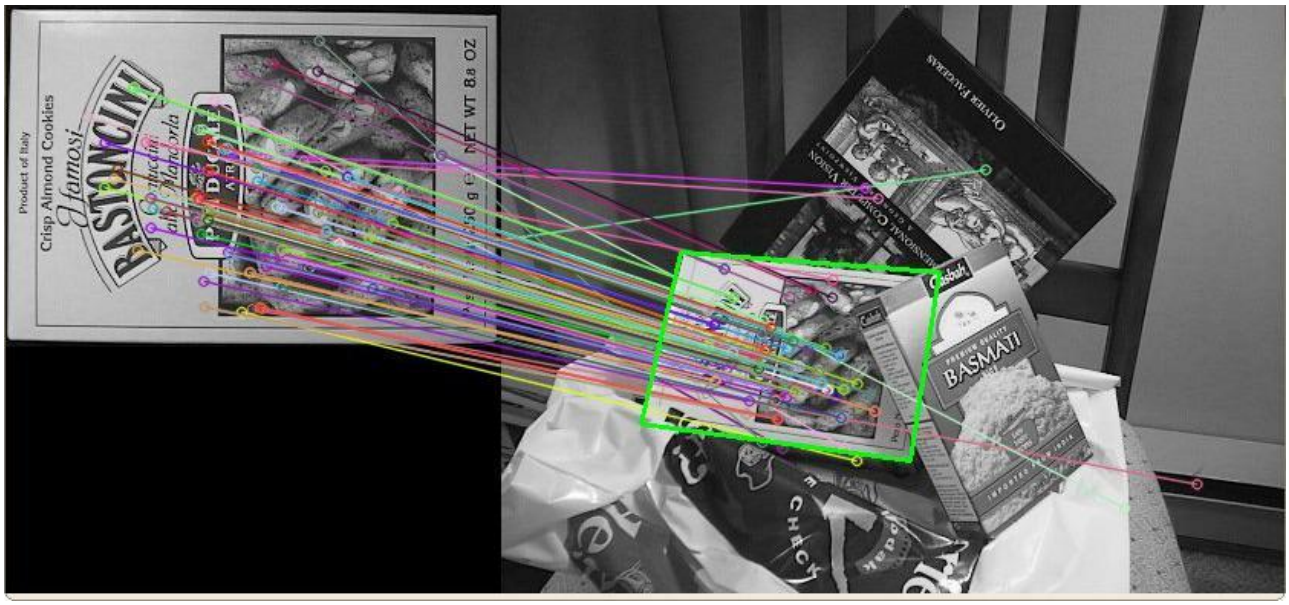


Рисунок 2.11 – Відповідності між точками шаблону і зображенням, що тестується

Різні алгоритми працюють з різною швидкістю і ефективністю. В умовах застосування їх для побудови доповненої реальності необхідно використати тільки ті, які показують високу швидкість роботи при досить хорошій якості відстежування позицій ключових точок. Інакше можно отримати помітні відставання у відеоданих, що знімаються.

Для підвищення швидкості роботи алгоритмів feature points detection застосовуються різні способи фільтрації точок, щоб мінімізувати їх число і відсіяти зовсім

неефективні поєднання. Таким чином, можна добитися не лише підвищення швидкості роботи алгоритмів, але і якості трекінгу маркерів [8].

2.2.3 Бібліотеки комп'ютерного зору

За минуле десятиліття різними організаціями і науковими співтовариствами було створено декілька бібліотек комп'ютерного зору. Оскільки питання продуктивності є досить гострим в цій дисципліні та більшість з них написані на мові C++. Окрім високої швидкості виконання це також дає можливість їх застосування на різних платформах.

На сьогоднішній день існує кілька відомих бібліотек для розробки програм розпізнавання образів. Найпоширеніші: OpenCV, VXL, LTI.

OpenCV надає широкий ряд інструментальних засобів для обробки і розпізнавання графічних зображень. Бібліотека сумісна з Intel Image Processing Library (IPL), яка здійснює операції низького рівня в цифрових образах. OpenCV здебільшого високорівнева бібліотека. Вона була розроблена для затвердження загального стандартного інтерфейсу комп’ютерного зору для додатків у цій області. Іншою метою було зробити платформи Intel привабливими для розробників таких програм за рахунок додаткового прискорення OpenCV за допомогою Intel Performance Libraries (зараз використовуються IPP – низькорівневі бібліотеки для обробки сигналів, зображень, а також медіа-кодеки; MKL – спеціальна версія LAPACK і FFTPack).

VXL – це C++ бібліотека, яка реалізує кілька загальних алгоритмів комп’ютерного зору і пов’язаних з ним функцій.

LTI-Lib – об’єктно-орієнтована бібліотека для комп’ютерного зору. Вона також реалізована на мові C++. Включає в себе класи, які інкапсулюють багатопоточність, синхронізацію, доступ до портів, і т.д. Це гарантує, що розробники не повинні мати проблем зі зміною операційних систем чи апаратного забезпечення.

Крім цих, існують ще декілька комерційних бібліотек. VisionLab – одна з них. Це набір компонентів для основних задач комп’ютерного зору. Бібліотека дозволяє швидко створити повнофункціональну програму-детектор руху для індустрії безпеки, практично не пишучи свого програмного коду. Вона також включає технології Hough Lines, визначення країв, контурів, виявлення облич і об’єктів за Хааром, виявлення/спостереження та багато іншого.

Для некомерційних цілей VisionLab є повністю функціональною і безкоштовною. Технологія VisionLab включає в себе повний набір компонентів відео-захоплення. Вона

підтримує старий Win32 API (Video For Windows (VFW), WaveAPI, аудіо ACM), новий DirectShow, і навіть дозволяє при необхідності компонувати їх.

Пакет Luxand FaceSDK застосовується як високо сумісна з Win32 DLL програма і підтримує популярні середовища розробки. Користувачі Microsoft Visual C++ і Borland Delphi отримають потрібні заголовки файлів, спрощуючи інтегрування з новими або існуючими проектами.

Luxand FaceSDK використовує складні алгоритми для визначення осіб на цифрових фото швидко і надійно. SDK обробляє зображення, визначає людські обличчя, повертає координати більше 40 точок обличчя, очі, кути очей, брови, кути рота, ніс і т.д. Функція визначення рис обличчя дозволяє автоматично проводити завдання, такі як видалення ефекту червоних очей і налаштування відтінку шкіри.

Був проведений порівняльний аналіз швидкодії перших трьох бібліотек на основі чотирьох найрозповсюдженіших задач комп'ютерного зору (рис. 2.12):

– 2D DFT: пряме перетворення Фур'є зображень 512 × 512;

– Resize: зміна розміру з 512 × 512 до 384 × 384, білінійна інтерполяція, 8-біт, було використано 3 спрямовані зображення;

– Optical Flow (оптичний потік): спостереження за 520 точками за допомогою вікна 41×41 та 4-рівневої піраміди;

– Neural Nets (нейронні мережі): було використано FANN (Fast Artificial Neural Network) бенчмарк – відкрите програмне забезпечення для побудови нейронних мереж.

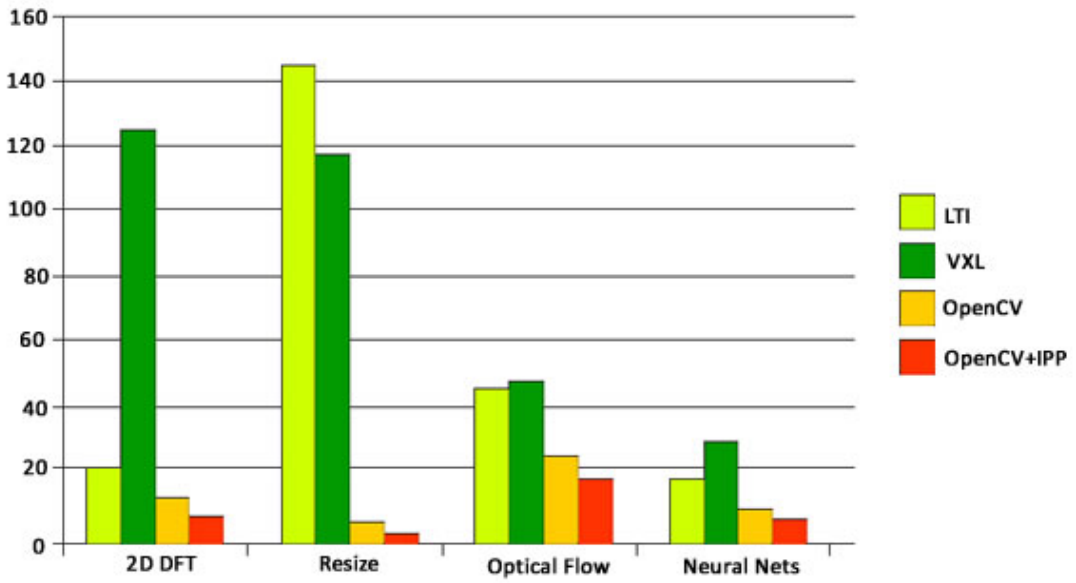


Рисунок 2.12 – Порівняльний аналіз шкидкодії бібліотек LTI, VXL, OpenCV та OpenCV+IPP

За результатами проведеного тестування OpenCV буде найшвидшим рішенням для приведених вище задач.

Слід зазначити, що нині OpenCV (англ. Open Source Computer Vision Library) є, напевно, найвідомішою і розвиненішою бібліотекою комп'ютерного зору.

До її складу входять модулі:

– обробки зображень;

– побудови простих призначених для користувача інтерфейсів;

– завантаження/збереження відео і аудіо даних;

– аналізу руху і відстежування об'єктів (оптичний потік, шаблони руху, усунення фону);

– калібрування камери, пошуку стерео-відповідностей і елементів обробки тривимірних даних;

– пошуку, аналізу і порівняння ключових точок зображень (Feature Detection and Description);

– детектування об'єктів на зображенні (вейвлеты Хаару, HOG і т. д.);

– методи і моделі машинного навчання (SVM, дерева ухвалення рішень і т. д.).

OpenCV випускається під ліцензією BSD з відкритим початковим кодом, внаслідок чого її дозволено використати як для академічних цілей, так і для комерційних. Розробка бібліотеки почалася в 1999 році і триває до цього дня. Багато проектів, що зачіпають комп'ютерний зір, користуються саме цією бібліотекою. Варто відмітити, що чимале число бібліотек доповненої реальності так чи інакше використовують реалізації алгоритмів, створених в OpenCV [9].

AForge.NET. AForge.NET являється C# фреймворком з відкритим початковим кодом, створеним для розробників і дослідників в області комп'ютерного зору і штучного інтелекту. Основним ідеологом і розробником цього проекту є одна людина (Андрій Кирилов), проте упродовж усього періоду свій вклад вносили розробники з різних країн світу.

Фреймворк включає наступні компоненти:

– AForge.Imaging – обробка зображень і набір різних фільтрів;

– AForge.Vision – набір методів і алгоритмів комп'ютерного зору;

– AForge.Video – обробка відеопотоку;

– AForge.Neuro – побудова і робота з нейронними мережами;

– AForge.Genetic – набір генетичних алгоритмів;

– AForge.Robotics – спеціальний набір методів для застосування в області робототехніки та інші.

На основі AForge.NET була розроблена бібліотека Gratf, використовувана для побудови доповненої реальності. Вона написана на мові C# і має хорошу переносимість на різні платформи.

Окрім вказаних вище існує ряд інших кросплатформених бібліотек комп'ютерного зору. Серед них можна виділити ROS (Robot Operating System) – бібліотека з відкритим початковим кодом, вживана для створення програмного забезпечення роботів. VXL, Integrating Vision Toolkit, ViSP – C++ фреймворки з набором модулів по обробці і аналізу зображень, відеопотоку, пошуку шаблонів і об'єктів, класифікаторів і багатьох інших. Список подібних бібліотек досить широкий, і чимале число з них побудовані на основі OpenCV. З цієї причини перераховувати їх не має сенсу

3 АЛГОРИТМИ ВІЗУАЛЬНОЇ ЛОКАЛІЗАЦІЇ І ТРЕКІНГУ

У цьому розділі розглянемо різні варіанти алгоритмів, які можна використовувати для вирішення завдання трекінгу по відео з камери для побудови доповненої реальності.

Усі методи, що дозволяють робити трекінг природного оточення на основі візуальної інформації, розділяються на дві основні категорії:

– методи, що працюють з готовою моделлю оточення;

– методи, що працюють в невідомому оточенні без апріорної інформації про нього.

3.1 Трекінг з використанням моделі оточення

Розглянемо методи, що дозволяють робити трекінг природного оточення за наявності тривимірної моделі усього оточення або об'єкту в ньому. Більшість методів є рекурсивними, тобто вони використовують положення на попередньому кадрі як апріорну інформацію про поточну позицію.

3.1.1 Метод семплювання точок з ребер

Рекурсивний метод, що грунтується на семплюванні точок з ребер початкового об’єкту. Для апріорного положення поточного кадру визначається безліч видимих ребер об'єкту і з них рівномірно семплюється набір точок. Потім аналогічно виділяються ребра з поточного кадру і будується набір точок для них. За допомогою мінімізації помилки репроекції виділених точок визначається положення поточного кадру.

Цей метод добре підходить для трекінгу полігональних об'єктів або об'єктів, що мають чіткі контури, оскільки він грунтований на виділенні меж. Інформація про текстуру об'єкту не використовується в процесі роботи, тому метод семплювання точок може бути успішно використаний для однотонних об'єктів і є стійким до змін освітлення.

3.1.2 Метод точок інтересу

Окрім самої моделі об'єкту для застосування цього методу використовуються декілька кадрів зйомки об'єкту з різних ракурсів з відомими положеннями камери. На кожному такому кадрі виділяються ключові точки на зображенні і з них за допомогою зворотного проектного перетворення (репроекції) виходить безліч точок інтересу - точок на поверхні об'єкту. Для трекінгу поточного кадру знаходиться найближчий кадр зі збережених, грунтуючись або на гістограмах самих зображень, або на положенні камери на попередньому кадрі. Потім, використовуючи апріорне знання про поточне положення камери, вибраний найближчий кадр з допомогою репроекції перетворюється в проміжне зображення об'єкту, близьке до поточного ракурсу. І вже на цьому проміжному зображенні знаходяться співвідношення для точок інтересу, видимих на поточному кадрі.

Розглянемо фрагмент *U* зображення *I(x, y)* з центром в точці *(u, v)*, і його копії, зрушені на величину *(x, y)*. Для кожної точки фрагмента можна обчислити зважений квадрат різниці між зрушеним і початковим фрагментом і розглянути функцію:

$$S(x,y) = \sum_{(u,v)\epsilon U} \omega(u,v)\big(I(u+x, v+y) - I(u,v)\big)^2. \quad (3.1)$$

Функція $I(u+x, v+y)$ може бути розкладена в ряд Тейлора в околиці центру *(u,v)*, що дозволяє перейти від (3.1) до виразу:

$$S(x,y) \approx \sum_{(u,v)\epsilon U} \omega(u,v)\big(I_x(u,v)x + I_y(u,v)y\big)^2, \quad (3.2)$$

де I_x, I_y – часткові похідні яскравості в горизонтальному і вертикальному напрямах.

Вираз (3.2) можна записати в матричному виді:

$$S(x,y) \approx (xy)M\begin{pmatrix}x\\y\end{pmatrix}, \quad (3.3)$$

де $M = \sum_{(u,v)\epsilon U} \omega(u,v)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$ – матриця локальної структури.

Як вагова функція $\omega(u,v)$ зазвичай використовується функція Гауса. Кут характеризується великими змінами функції *S(x, y)* по усіх можливих напрямах *(x, y)*, що еквівалентно великим по модулю власним значенням матриці *M*.

Звідси можна зробити висновки:

– якщо власні значення λ_1 і λ_2 близькі до нулю, то піксель з центром в *(x, y)* не є точкою інтересу, оскільки він лежить в однорідній області;

– якщо $\lambda_1 \approx 0$, а λ_2 набуває великого по модулю значення, то піксель *(x, y)* належить краю;

– якщо обидва власні значення великі і набувають позитивних значень, то піксель *(x, y)* є кутом.

Більшість операторів детектування кутів грунтовані на властивостях матриці *M*. У 1988 році Харрісом і Стефенсом було запропоновано використати міру відгуку кута:

$$z(x,y) = det(M) - k \times tr(M)^2, \quad (3.4)$$

де k – емпірично знайдений параметр порядку 0,04-0,06, а *det(M)* і *tr(M)* – визначник і слід матриці.

При негативному відгуку точка класифікується як точка, що потрапила на край; при відгуку, близькому до нуля, точка вважається такою, що потрапила в "пласку" область. При великих позитивних значеннях *z(x, y)* вважається, що точка є кутом, оскільки в ній яскравість сильно змінюється на всіх напрямках. Детектор Харріса (3.4) інваріантний до обертання і зрушення зображення, а також до зрушення і рівномірної лінійної зміни яскравості.

Описані детектори, хоча і називаються детекторами кутів, знаходять не власне кути, а будь-які ділянки зображення, в яких є велика зміна градієнта на всіх напрямках при заданому масштабу. Детектори досить швидкі, оскільки зводяться до диференціювання яскравості зображення, підсумовування похідних яскравості в локальній околиці кожної точки і знаходження міри відгуку кута. Окрім детектора Харріса існують інші методи виявлення кутів, що дозволяють знаходити кути залежно від масштабу зображення.

Найбільшу популярність отримали детектори SIFT (scale-invariant feature transform – масштабно-незалежне перетворення особливостей) і його прискорений варіант SURF (speeded-up robust features – прискорені стійкі особливості). Детектор SIFT грунтований на ідеї пошуку локальних максимумів в так званому просторі змінного масштабу (scale space). Для заданого зображення *I(x, y)* простір змінного масштабу є безліччю значень функціонала:

$$L\ (x, y; \sigma)\ = G\ (x, y; \sigma)\ * I\ (x, y), \tag{3.5}$$

де $\sigma > 0$ – параметр згладжування,

символ "$*$" означає згортку,

$G(x, y; \sigma)$ – двовимірна функція Гауса.

На рисунку 3.1 представлена схема побудови детектора SIFT та розглянуті зрізи простору змінного масштабу, що відрізняються постійним позитивним множником *k*.

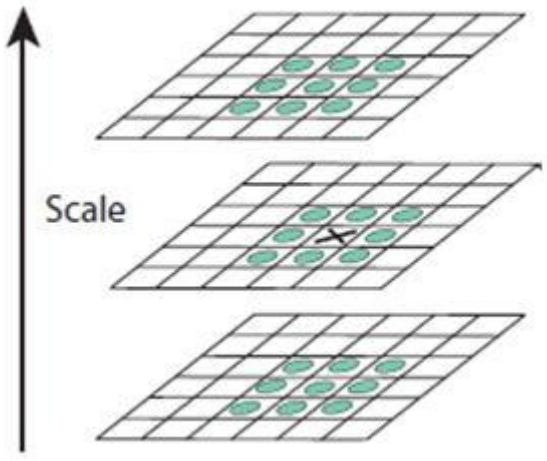


Рисунок 3.1 – Схема побудови детектора SIFT

В якості точок інтересу пропонуються точки, що відповідають локальним екстремумам функції:

$$D(x, y; \sigma) = L(x, y; k\sigma) - L(x, y; \sigma). \quad (3.6)$$

Для отримання S зрізів простору змінного масштабу обчислюється згортка початкового зображення з ядром Гауса з параметром згладжування, що послідовно змінюється σ_0, $k\sigma_0$,…, $k^s\sigma_0$. Далі знаходяться локальні екстремуми функції *D(x, y, σ).* Для цього значення функції *D(x, y,* $k^i\sigma_0$*)* в кожній точці *(x, y)* порівнюється зі значеннями у восьми сусідніх пікселях при тому ж значенні параметра масштабу, а також в 18-ти сусідніх пікселях, що належать попередньому і подальшому зрізам простору змінного масштабу. Локальні екстремуми, в яких значення |*D (x;σ)* | не перевершує деякий заданий поріг, відкидаються. Потім будується матриця H других часткових похідних (Гессіан) функції *D(x, y, σ).* Якщо величина $tr(H)^2/det(H)$ менше деякого порогу, то точка вважається характерною.

Детектор SURF використовує ту ж ідею простору змінного масштабу, що і детектор SIFT, але функція Гауса у вираженні (3.5) наближається прямокутним фільтром 9×9.

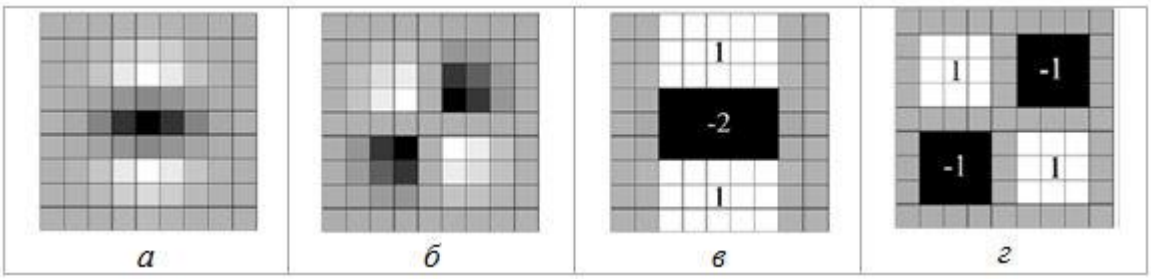


Рисунок 3.2 – Фільтри для знаходження другої похідної яскравості зображення по напрямах *y* і *xy*; а, б – фільтри Гауса (SIFT); в, г – прямокутні фільтри з цілочисельними вагами (SURF)

На рис. 3.2 показані фільтри для отримання приватних похідних початкового зображення *I(x, y)* по *yy* і по *xy*. При цьому ліворуч наводяться (обрізані) фільтри других похідних Гаусіана, а справа – прямокутні фільтри, обчислюючі ці похідні приблизно. Згортка початкового зображення з такими фільтрами, ваги яких є цілими числами, обчислюється дуже швидко. Далі в якості характерних точок виявляються локальні максимуми у вікні 3×3×3 функції:

$$det(Happrox) = I_{xx}I_{yy} - \left(0{,}9I_{xy}\right)^2. \quad (3.7)$$

В середині 2000-х років у зв'язку зі збільшеним попитом на рішення завдань комп'ютерного зору в реальному часі з'явилися евристичні алгоритми швидкого пошуку точок інтересу.

Найбільш яскравим представником цього класу алгоритмів є алгоритм FAST (features from accelerated segment test – особливості, отримані з прискореної перевірки сегментів). Цей алгоритм не вимагає обчислення похідних яскравості.

На рис. 3.3 показана аналізована точка і пікселі, що оточують її. Яскравість пікселів, що лежать на колі, порівнюється з яскравістю центральної точки, і на підставі ряду перевірок приймається рішення, чи є центральна точка характерною. Послідовність перевірок і їх загальне число підбираються і оптимізовані заздалегідь на основі великої навчальної вибірки зображень. В результаті, перевірки виконуються дуже швидко.

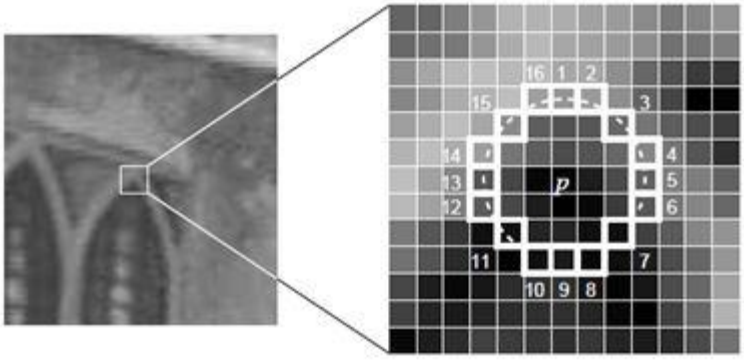


Рисунок 3.3 – Пікселі, що використовуються у швидких перевірках алгоритму FAST

Для ухвалення рішення є точка кутом або ні потрібно лише декілька десятків операцій порівняння. Алгоритм FAST добре зарекомендував себе в додатках, що здійснюють стеження за об'єктами в реальному часі.

3.2 Трекінг в невідомому оточенні

В основі більшості систем, що застосовуються для трекінгу в невідомому оточенні по відео з камери, лежить підхід Simultaneous Localization and Mapping (SLAM). Ця загальна назва для систем, що дозволяють по відео з однієї камери одночасно будувати карту заздалегідь невідомого оточення і робити локалізацію в ньому. В основному вони розроблялися у сфері робототехніки для навігації роботів в невідомому середовищі.

Нині методи рішення задачі SLAM швидко розвиваються. Це обумовлено збільшенням продуктивності обчислювальних машин, поліпшенням якості датчиків і появою нових, розвитком робототехники. Завдання SLAM є дуже важливим, оскільки без її рішення навряд чи можливе створення по-справжньому автономного робота.

Завдання SLAM складається з безлічі підзадач. Як правило, виділяються наступні:

– рекурсивний фільтр;

– знаходження орієнтирів в просторі;

– пошук відповідностей між орієнтирами;

– перерахунок положення робота;

– уточнення положення орієнтирів на карті.

При усьому різноманітті вживаних в завданні синхронного визначення місця розташування і побудови методів, можна виділити деяку класичну, дуже загальну схему роботи алгоритму (рис. 3.4).

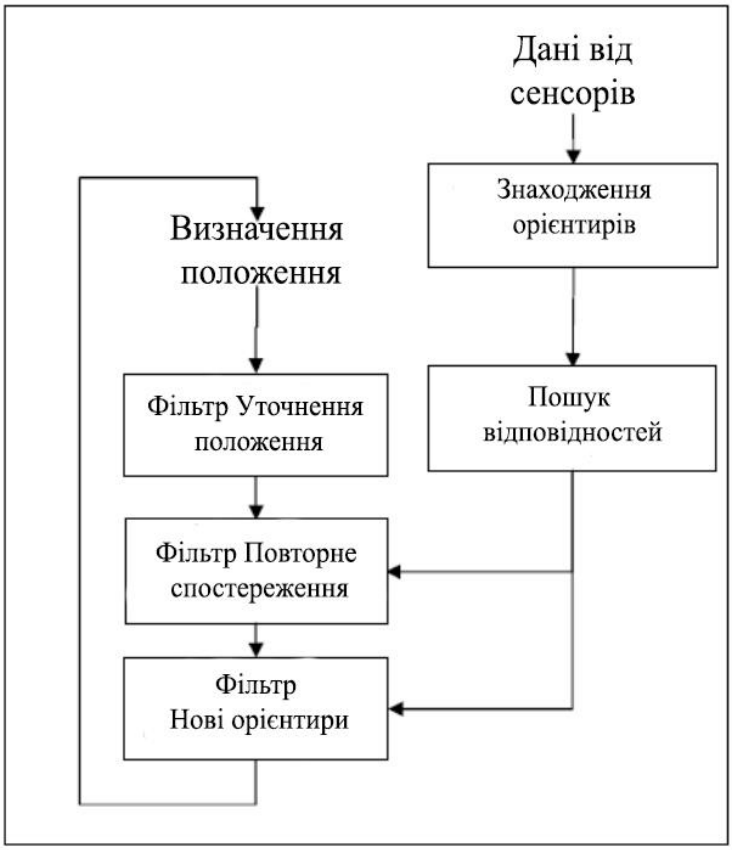


Рисунок 3.4 – Класична схема алгоритму SLAM

На кожному кроці алгоритму на вхід подаються дані від сенсорів. За цими даними в просторі знаходяться орієнтири і визначаються їх описи, необхідні для пошуку відповідностей. В процесі роботи будується структура, яка зберігає орієнтири і їх описи. При пошуку відповідностей для кожного виявленого орієнтиру шукається відповідність в

цій структурі. Якщо відповідність не знайдена, то орієнтир просто додається в структуру. Якщо ж було виявлено відповідність, то орієнтир використовується для обчислення положення.

Після обчислення положення застосовується рекурсивний фільтр для зниження рівня шуму у вичисленому положенні (уточнення положення). Потім для виявлених орієнтирів, у яких знайшлися відповідності, виконується уточнення їх положення за допомогою рекурсивного фільтру.

На відміну від руху робота при розробці системи трекінгу для доповненої реальності треба враховувати можливість різких і непередбачуваних рухів камери. Розглянемо різні варіанти таких систем, їх достоїнства і недоліки.

Можна виділити два типи SLAM–систем:

– грунтовані на фільтрах (фільтр Калмана чи фільтр часток);

– системи, що не використовують фільтри, а вирішують задачу оптимізації для визначення поточної позиції.

Основний недолік методів, що грунтуються на фільтрах – поточна позиція і повний стан карти сильно пов'язані і повинні оновлюватися на кожному кадрі. При цьому в алгоритмах, що не застосовують фільтри, можливе оновлення поточного положення камери з використанням тільки частини карти без оновлення стану самої карти. Тому вони здатні показувати кращу продуктивність трекінгу.

Будь-яка система SLAM включає два основні модулі: трекінгу і побудови карти. Модуль трекінгу відповідає за визначення поточного положення камери для кожного нового кадру, використовуючи апріорне положення (це може бути або просто положення попереднього кадру, або приблизно змодельоване нове положення на основі моделі руху камери). Потім, зіставляючи дані нового кадру з наявною картою, обчислюється необхідне перетворення і визначається положення поточного кадру.

Модуль побудови карти створює карту оточення у вигляді тривимірної хмари точок з дескрипторами - розрідженим або щільним. В процесі ініціалізації будується невелика початкова карта, яка потім оновлюється і доповнюється в процесі роботи системи. Положення точок карти у світовій системі координат обчислюється за допомогою тріангуляції відповідностей однакових ключових точок на різних кадрах. Оскільки оновлення карти не зобов'язане здійснюватися на кожному кадрі, для нього можуть бути використані методи, працюючі достатньо довго, наприклад bundle adjustment.

За способом обробки кадрів, що входять, методи візуальної локалізації поділяються на прямі і непрямі.

Прямі методи враховують яскравість кожного пікселя зображення для визначення поточної позиції, а непрямі спочатку виділяють на зображеннях особливі точки і працюють вже з набором ключових точок і їх дескрипторами. До плюсів прямих методів можна віднести кращу роботу у слабо текстурованих сценах, у яких неможливо виділити достатню кількість ключових точок. При цьому прямі методи гірше пристосовані до змін в освітленні і набагато вимогливіші до продуктивності. Їх робота в реальному часі стала можлива останнім часом тільки за рахунок паралелізування і обчислень на GPU.

У непрямих методах один з найважливіших кроків будь-якого алгоритму – виділення особливих точок на зображенні. Найбільш ефективним способом, використовуваним у більшості систем, є детектор кутів FAST. Дескриптором ключових точок називається набір значень, що описують точку і застосовуються для їх порівняння. В якості дескрипторів можливі різні варіанти: SIFT, SURF, локальні околиці пікселів.

З алгоритмів SLAM з відкритим початковим кодом можна виділити:

– Parallel Tracking and Mapping (PTAM). Перший алгоритм, в якому трекінг і побудова карти були розділені для виконання в різних потоках. Краще всього підходить для застосування до побудови доповненої реальності і допускає адаптацію для роботи на мобільних пристроях з невеликою обчислювальною потужністю.

– Semi – Direct Visual Odometry (SVO). Гібридний метод, що поєднує прямий і непрямий підхід. Показує дуже добрі результати трекінгу, але вимагає камери з високою частотою кадрів (не менше 70 кадрів в секунду), що робить його непридатним для роботи на більшості мобільних пристроїв.

– Large – Scale Direct monocular SLAM (LSD – SLAM). Прямий метод, що використовує імовірнісний підхід для побудови карти у вигляді щільної хмари точок. Дозволяє будувати якісну модель оточення в реальному часі, але також вимогливий до продуктивності пристрою.

Після огляду різних алгоритмів візуальної локалізації для реалізації бібліотеки був вибраний алгоритм PTAM, виходячи з його успішного застосування для побудови доповненої реальності на персональному комп'ютері і можливості адаптації для роботи на мобільних пристроях.

4 АЛГОРИТМ PTAM – PARALLEL TRACKING AND MAPPING

Початковий алгоритм PTAM розроблявся в лабораторії комп'ютерного зору в Оксфорді і його код був викладений у відкритий доступ під ліцензією GPL. У статті Джорджа Клейна і Девіда Мюррея «Паралельне відстеження і картографування для малих робітничих областей AR» (eng. Parallel Tracking and Mapping for Small AR Workspaces) 2007 року описані ключові ідеї PTAM і його застосування до побудови доповненої реальності на комп'ютері в оточенні невеликого розміру – кімната або офіс, в яких можливо побудувати досить повну карту місця, обходячи його з камерою в процесі роботи системи. Спираючись на дані із статті і початкового коду оригінального алгоритму, розглянемо принцип роботи цього методу і підходи, вживані при його реалізації.

Ключовою особливістю цього алгоритму є розділення трекінгу і побудова карти для виконання в різних потоках. Це дозволяє використати для побудови карти досконаліші методи, раніше недоступні в алгоритмах, що працюють в реальному часі. Основний потік, який відповідає за трекінг, обробляє поточний кадр з камери і на основі наявної карти визначає поточний стан камери. Для побудови карти автори вводять поняття ключових кадрів – кадрів з відеопотоку, що найкраще представляють інформацію про оточення. Потік побудови карти відповідає за додавання в карту нових ключових кадрів, оновлення положень точок на карті під час вступу нових вимірів і фільтрацію карти – видалення зайвих крапок і ключових кадрів.

4.1 Визначення та ініціалізація PTAM

У цьому розділі будемо вважати світовою системою координат систему координат, введену в процесі ініціалізації системи і не прив'язану до реальної геопозиції. Назвемо поточним становищем камери перетворення E_{CW}, представлене матрицею розміру 4×4, що переводить точку зі світової системи координат у простір камери:

$$P_{jC} = E_{CW} P_{jW}, \tag{4.1}$$

де P_{jC} – точка в локальній системі координат камери, представлена в однорідних координатах;

P_{jW} – координати цієї точки в світі.

Матриця E_{CW} – основний результат, що отримується системою при трекінгу кожного кадру. Для зручності обчислень це ж перетворення може бути представлене як елемент SE(3) – групи L_{i}, що визначає множину усіх можливих рухів і поворотів в тривимірному просторі.

Модель проекції камери визначає проекцію точки з системи координат камери на екран:

$$\binom{u_i}{v_i} = CamProj(E_{CWP_{jW}}). \tag{4.2}$$

Важливою умовою, необхідною для трекінгу з використанням цієї моделі, є можливість продиференціювати це рівняння для обчислення зміни положення камери між кадрами :

$$E'_{CW} = ME_{CW} = \exp(\mu)\, E_{CW}. \tag{4.3}$$

Як модель проекції камери тут використовується варіант моделі камери з точковою діафрагмою, що підтримує радіальне спотворення від лінз. Параметри моделі включають фокусні відстані f_u і f_v , головну точку знімка (u_0, v_0), а також параметр спотворення ω. Усі параметри визначаються на кроці калібрування камери і під час роботи системи вважаються відомими. Функція проекції цієї моделі, що перетворює точку в локальній системі координат камери, яка представлена в однорідних координатах, в координати екрану, визначається формулою:

$$CamProj\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \binom{u_0}{v_0} + \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} \frac{r\prime}{r} \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}, \tag{4.4}$$

$$r = \sqrt{\frac{x^2+y^2}{z^2}}, \tag{4.5}$$

$$r' = \frac{1}{\omega}\arctan\,(2r\tan\frac{\omega}{2}). \tag{4.6}$$

Карта, яка будується в процесі роботи системи, є набором ключових кадрів і точок з дескрипторами. Ключовий кадр S_i містить в собі положення камери у момент його зйомки E_{CW} і набір вимірів для точок карти, що містяться в ньому. Дескрипторами точок карти є околиці пікселів на ключових кадрах, що містять ці точки.

Крок ініціалізації є обов'язковим для будь-якої системи візуального трекінгу, працюючого з однією камерою. У цей момент будується первинна карта оточення і визначається початкове положення камери. Для успішної ініціалізації системи потрібно два кадри, на яких була знята одна і та ж сцена з різних точок огляду, і в процесі ініціалізації по цих кадрах обчислюється або фундаментальна матриця, або матриця гомографії.

Ініціалізація PTAM вимагає від користувача вручну задати два ключові кадри, провівши достатнє паралельне зрушення камери між ними. При надходженні першого кадру за допомогою детектора кутів на ньому виділяється набір ключових точок. Потім відповідності для вихідних ключових точок на нових кадрах підтримуються впродовж усієї процедури ініціалізації. Тому цей етап вимагає відсутності різких і швидких рухів камери.

Після додавання користувачем другого ключового кадру з наявних відповідностей точок за допомогою алгоритму MLESAC визначається 8 можливих розкладань перетворення гомографії. З них вибирається краща гомографія, що не породжує точок з негативною глибиною, і визначається відносне перетворення камери між першим і другим кадром.

Початкова карта у вигляді хмари точок будується на кроці ініціалізації за допомогою алгоритму bundle adjustment. Масштаб сцени визначається в припущенні про те, що зрушення камери в процесі ініціалізації дорівнювало 10 см. За центр системи координат отриманої карти приймається медіана усіх точок початкової карти.

4.2 Алгоритми обробки кадру та трекінгу

Кожен кадр, що отримується з камери, заздалегідь конвертується в чорно–біле зображення, і з нього будується піраміда з чотирьох рівнів зображень меншого розміру. Нульовий рівень L_0 – це початкове зображення (наприклад, розміром 640×480), на кожному наступному рівні ширина і висота зменшуються в два рази.

Потім для кожного зображення з піраміди за допомогою детектора кутів FAST витягаються ключові точки. Оскільки детектор кутів використовує коло фіксованого розміру, наявність чотирьох рівнів зображення дозволяє визначати ключові точки різного масштабу і зробити систему стійкішою до зміни масштабу. На наступному кроці для кожної знайденої ключової точки обчислюється значення Shi – Tomasi і усі точки, значення для яких не перевершують зазначеного порогу, що відрізняється для різних

рівнів зображення, виключаються з розгляду. Це дозволяє краще контролювати кількість особливих точок на різних рівнях і залишати тільки найкращі з них.

Першим кроком для визначення поточного положення камери є його апроксимація з використанням моделі руху камери. Це наближення називається апріорною позицією камери. В якості моделі руху в PTAM використовується модель згасаючої швидкості. Ця модель оновлює швидкість на кожному кадрі по наступній формулі:

$$V_{new} = 0.9 * (V_{frame} + V_{old}) , \quad (4.7)$$

де V_{frame} – швидкість руху, обчислена з відповідностей між положеннями точок карти на поточному кадрі і попередньому;

V_{old} –попереднє значення швидкості.

При рівномірних рухах камери така модель дозволяє мінімізувати кількість обчислень, необхідних для визначення точного положення, оскільки апріорне наближення виявляється близько до нього.

Для уточнення апріорного положення камери використовується карта оточення, доступна на даний момент. Точки карти проектуються на екран на основі апріорного положення, і з них будується множина "можливо видимих точок", що потрапляють в межі екрану після цієї проекції. Знаходження відповідностей між ключовими точками виконується в два етапи.

На першому етапі вибирається невелике число (близько 50) точок з верхніх рівнів піраміди зображень (великі ознаки), і їх відповідності шукаються в досить великому радіусі відносно апріорного положення. Це допомагає зберегти можливість трекінгу навіть при різких рухах камери, оскільки великі ознаки на зображеннях зберігаються в таких ситуаціях краще.

На другому кроці використовуються видимі точки, що залишилися, але тут вже пошук відповідностей здійснюється в маленьких околицях проекцій точок.

Обчислення швидкості руху камери між двома кадрами також допомагає оцінити можливе розмиття зображення і відповідно модифікувати процес трекінгу. Якщо швидкість руху велика, використання точних ознак з низьких рівнів піраміди неможливе через велике розмиття, тому другий крок алгоритму не застосовується і трекінг використовує тільки великі ознаки, які менше схильні до розмиття. Якщо ж камера майже стаціонарна, використання великих неточних ознак може привести до тремтіння, відповідно використовується відразу другий крок і пошук в невеликих околицях

апріорних положень. Ці зміни дозволяють збільшити стабільність алгоритму в ситуаціях різного характеру рухів камери.

Щоб знайти відповідність для точки карти ρ на поточному кадрі, здійснюється пошук її дескриптора – околиці пікселів на кадрі, з якого ця точка побудована, у фіксованій околиці її передбачуваного положення на поточному кадрі. Для компенсації спотворень, що виникають через різні кути огляду цієї точки, перед пошуком до дескриптора застосовується афінне перетворення, що задається матрицею:

$$A = \begin{bmatrix} \frac{\partial u_c}{\partial u_s} & \frac{\partial_c}{\partial v_s} \\ \frac{\partial v_c}{\partial u_s} & \frac{\partial v_c}{\partial v_s} \end{bmatrix}, \tag{4.8}$$

де $(u_s,\ v_s)$ – це зміщення, що відповідають горизонтальному і вертикальному зміщенню на один піксель в кадрі, з якого побудований дескриптор;

(u_c, v_c) - аналогічні значення для поточного кадру.

Пошук дескриптора після такого перетворення реалізується за допомогою обчислення метрики zero-mean SSD. Значення пікселів спочатку центруються для зменшення впливу освітлення шляхом віднімання середньої інтенсивності зображення, а потім вважається сума квадратів відстаней між пікселями. Обчислюючи значення метрики для дескриптора і околиць кутів у фіксованій області поточного кадру, відповідність знаходиться, якщо відстань не перевершує заданого порогу.

Після знаходження множини відповідностей для точок карти S на поточному кадрі кожній точці з цієї множини відповідає точка на зображенні $(\hat{u}, \hat{v})^T$. Для визначення зміни положення камери на основі обчислених відповідностей використовується метод найменших квадратів в застосуванні до завдання мінімізації помилки репроекції:

$$\mu' = \underset{\mu}{\operatorname{argmin}} \sum_{j \epsilon S} Obj\left(\frac{|e_j|}{\sigma_j}, \sigma_T\right), \tag{4.9}$$

де $|e_j|$ – вектор помилки репроекції;

функція $Obj(\cdot, \sigma_T)$ – зважене середнє Тьюкі, цільова функція для оцінки середнього значення виборки, що стійка до наявності викидів;

параметр σ_T визначається як робастна оцінка стандартного відхилення розподілу, обчислена зі значень помилок.

Якість трекінгу визначається долею успішно знайдених видимих ключових точок на поточному кадрі. Якщо вона не перевищує деякого порогу (наприклад 0.3), якість трекінгу перестає вважатися хорошею і нові точки не можуть бути додані в карту. Якщо ж ця доля менше нижнього порогового значення (0.13), система вважає, що поточне положення камери є некоректним і намагається провести відновлення – локалізацію на поточній карті без урахування апріорного положення. Для цього використовується найменше зображення з піраміди: порівнюючи середньоквадратичні відхилення зображень, серед усіх ключових кадрів знаходиться найближчий до поточного. Потім вирішується завдання оптимізації для знаходження перетворення поточного кадру відносно знайденого.

4.3 Алгоритми побудови карти

При побудові карти система вирішує декілька завдань:

– додавання в карту нових ключових кадрів і точок;

– уточнення положень точок карти з використанням нових вимірів;

– знаходження нових відповідностей між точками карти і ключовими кадрами;

– видалення поганих точок з карти.

4.3.1 Додавання ключових кадрів

Після кроку ініціалізації карта оточення містить тільки два ключові кадри і невелику кількість точок. Згодом при русі камери і віддаленні від початкового положення вона розширюється шляхом додавання нових ключових кадрів. Для того, щоб поточний кадр був доданий як ключовий, необхідно щоб виконувалися наступні умови:

– якість трекінгу має бути гарною;

– з моменту додавання попереднього ключового кадру в карту повинно пройти достатньо часу (наприклад, 20 кадрів);

– положення камери повинне знаходитися на деякому віддаленні від положення найближчого ключового кадру, що вже міститься в карті. При цьому конкретне значення цієї відстані залежатиме від глибини і масштабу карти: чим ближче камера знаходиться до точок карти, тим воно менше.

При додаванні нового ключового кадру в карту за його положення береться поточне положення камери, доступне з системи трекінгу. Також у цей момент вже є ключові точки, отримані детектором кутів FAST на різних рівнях зображення. Після застосування методу пригнічення немаксимумів (non-maximal suppression) і відсікання точок з маленькими значеннями метрики Shi-Tomasi, формується набір точок-кандидатів на додавання в карту.

Для обчислення глибини точки і її положення у світі недостатньо знати її координати на одному кадрі. Для тріангуляції вибирається сусідній ключовий кадр – з найближчим положенням камери до нового. Відповідності між ключовими точками знаходяться за допомогою епіполярного пошуку. Спочатку для точки-кандидата визначається епіполярна пряма на другому кадрі. Потім серед усіх кутів, що лежать на цій прямій, відбувається пошук відповідності: околиці пікселів порівнюються за допомогою обчислення суми квадратів відстаней. Якщо відповідність знайдена, точка тріангулюється і додається в карту.

4.3.2 Алгоритми оптимізації карти

Окрім додавання нових кадрів і точок при побудові карти проводиться постійне оновлення положень точок карти з урахуванням нових вимірів. Нехай поточний стан карти містить N ключових кадрів з положеннями камер $E_{K_1W,\dots,}E_{K_NW}$, і M точок $p_1, \dots, p_M$. Завданням глобальної оптимізації карти називається оптимізація усіх параметрів карти – позицій ключових кадрів $(\mu_2, \dots, \mu_N)$ (оскільки позиція першого ключового кадру завжди фіксована) і положень точок карти $(p_1, \dots, p_M)$. За допомогою алгоритму Левенберга-Марквардта вирішується завдання мінімізації помилок репроекції:

$$(\mu_2, \dots, \mu_N), (p_1, \dots, p_M) = \underset{\{\mu\}\{p\}}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j \epsilon S_i} Obj\left(\frac{|e_j|}{\sigma_j}, \sigma_T\right) \ . \tag{4.10}$$

Складність рішення системи рівнянь, що отримуються при рішенні цієї задачі, дорівнює $O(N^3)$, і проведення повної оптимізації при великому розмірі карти (>100 ключових кадрів) може займати досить тривалий час, десятки секунд. Це допустимо у разі, коли система оглядає частину оточення із вже побудованою картою, але може зупиняти процес побудови карти.

Тому окрім повної оптимізації додається можливість локальної оптимізації з використанням тільки частини ключових кадрів $X \cup Y$ і частини точок карти Z. А саме, в якості підмножини ключових кадрів X вибирається останній кадр, доданий в карту і найближчі чотири до нього. Усі точки, що містяться в кадрах з X, утворюють множину точок Z. Також до множини даних кадрів додається безліч Y: це кадри, що не містяться в X, але включають в себе точки з безлічі Z. Тобто локальна оптимізація використовує тільки околицю останнього кадру, доданого в карту. Це дозволяє скоротити асимптотику роботи оптимізації до $O(N\,M)$ у гіршому разі і успішно використати її в процесі побудови карти.

Важливим кроком для поліпшення якості карти є додавання нових відповідностей між точками карти і ключовими кадрами. У момент додавання в карту точка містить виміри тільки в двох кадрах, з яких вона отримана тріангуляцією. Згодом для цієї точки знаходяться відповідності і в інших кадрах, якщо вона в них міститься.

5 РЕАЛІЗАЦІЯ PTAM ДЛЯ ПЛАТФОРМИ ANDROID

Початковий алгоритм PTAM був адаптований Клейном і Мюрреєм для можливості роботи на смартфоні iPhone 3g, але ця версія не була викладена у відкритий доступ. Оскільки продуктивність процесора iPhone у той час була в 15-30 разів гірше, ніж у середнього комп'ютера, авторам довелося сильно спрощувати початковий алгоритм і жертвувати точністю трекінгу, щоб досягти потрібної продуктивності. У статті було показано, що навіть на такому пристрої виявляється можливим успішний трекінг маленького оточення з використанням цього алгоритму.

Попри те, що нині різниця в продуктивності смартфонів і комп'ютерів помітно скорочується, просте портування коду комп'ютерної версії PTAM виявляється недостатнім для його роботи в реальному часі на більшості пристроїв. У статті [5] автори пропонують варіант відповідної адаптації і оптимізації PTAM, у свою чергу додаючи використання орієнтації пристрою, отриманої за допомогою сенсорів IMU, для уточнення позиціонування. Далі розглянемо деталі реалізації алгоритму для платформи Android, а також зміни і поліпшення, грунтовані на роботах Клейна і Мюррея, які були внесені в процесі реалізації в початковий алгоритм для кращої роботи на мобільних пристроях при збереженні досить точного і стійкого трекінгу.

5.1 Вибір засобів розробки

Платформа Android була вибрана в якості цільової платформи як найпоширеніша платформа серед сучасних мобільних пристроїв. Додатки, що розробляються для цієї платформи, виконуються на багатьох пристроях з процесорами різної архітектури: x86, ARM та інших. Для досягнення мультиплатформеності основна частина додатків і бібліотек реалізується на мові Java, і отриманий байткод виконується на віртуальній машині Dalvik VM. При цьому існує можливість для написання частини додатка з використанням нативного коду на C/C++. За допомогою Android Native Development Kit (NDK) він компілюється в динамічні бібліотеки під потрібні платформи. Це дозволяє використовувати ще раз основну частину коду і бібліотек, викликаючи його через JNI-обгортку з основного додатку.

Основна частина коду цієї системи була написана на C++. Вибір мови був обумовлений можливістю перевикористання великої частини початкового коду десктопної версії PTAM і наявністю хороших бібліотек комп'ютерного зору. Обгортки на Java були використані тільки для передачі даних з камери пристрою в систему і передачі результату трекінгу назад в додаток. Незважаючи на накладні витрати, пов'язані з викликами через JNI, використання нативного коду дозволяє оптимізувати частини додатку, що містять велику кількість обчислень. У цій реалізації були використані наступні бібліотеки:

– CVD – високопродуктивна крос-платформена бібліотека для комп'ютерного зору, обробки зображень і відео. Використовувалася для зручної роботи із зображеннями;

– TooN – бібліотека лінійної алгебри з великою кількістю ефективних алгоритмів. Використовувалася для роботи з векторами і матрицями: обчислення сингулярного розкладання матриці, розкладання Холецкого;

– Agast – ефективна реалізація різних детекторів кутів, наприклад FAST і AGAST;

– Tinyxml – бібліотека для роботи з XML, використовувалася для серіалізації карти оточення.

5.2 Архітектура додатку

В якості інтерфейсу бібліотеки, доступного додатку, виступають класи-обгортки, які завантажують PTAM у вигляді динамічної бібліотеки libPTAM.so і потім звертаються

до неї з використанням JNI (дод. А). На рисунку 5.1 представлена діаграма класів зовнішнього інтерфейсу.

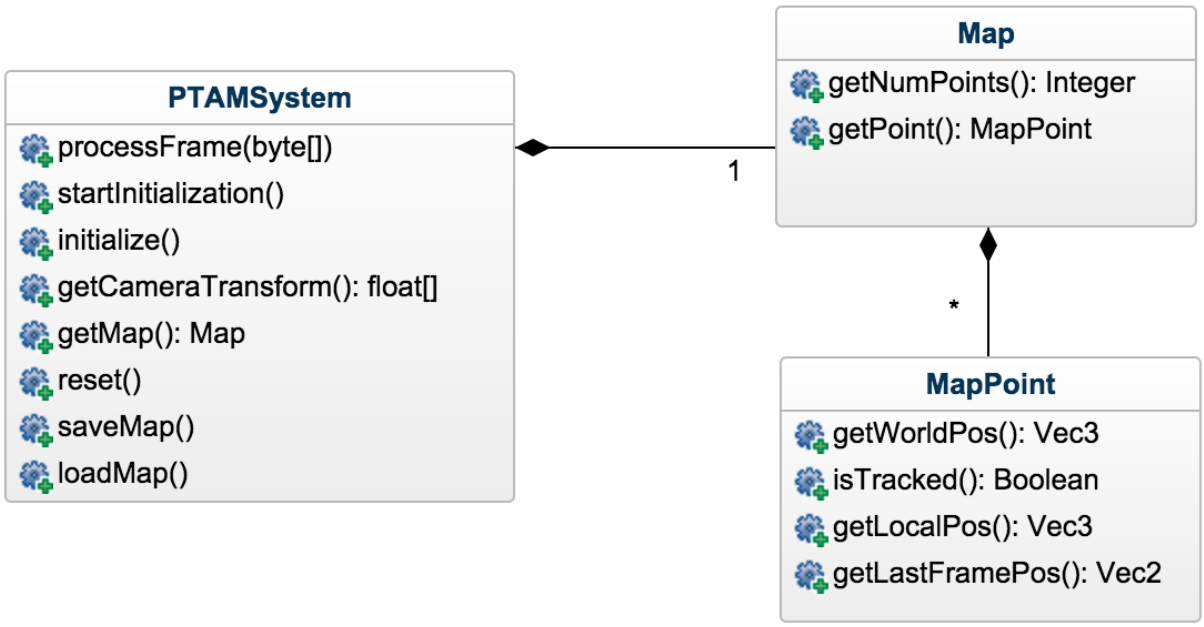


Рисунок 5.1 – Діаграма класів зовнішнього інтерфейсу бібліотеки

Основним класом системи є клас PTAMSystem. Він дозволяє провести ініціалізацію системи і дізнаватися поточне положення камери. При початку роботи вказується розмір зображення, що відповідає отримуваним кадрам з камери, які потім передаються в систему за допомогою методу processFrame. Збереження і завантаження карти по назві здійснюється з використанням методів saveMap і loadMap. Також є можливість отримувати поточну карту оточення і інформацію про точки карти: положення у світі і в системі координат камери, а також додаткову інформацію про стан поточного трекінгу: чи була ця точка знайдена на останньому кадрі і її положення на ньому.

На відміну від оригінального PTAM при такій архітектурі система ніяк не зав'язана на відрисовку контенту в додатку і може бути зручно використана тільки для здійснення трекінгу. При цьому можливе отримання низькорівневої інформації про трекінг і карти оточення, недоступної в готових бібліотеках для побудови доповненої реальності для мобільних пристроїв.

5.3 Адаптація алгоритму

5.3.1 Реалізація функції трекінгу

Продуктивність процесу трекінгу грає вирішальну роль в роботі усієї системи, оскільки він здійснюється для кожного кадру, отриманого з камери. Відповідно для досягнення частоти в 30 кадрів в секунду він повинен займати не більше 30 мс. Розглянемо детальніше оригінальний алгоритм трекінгу і можливості його спрощення без сильних втрат в якості.

Алгоритм функції трекінгу включає наступні кроки:

а) при отриманні нового кадру обчислюється апріорна модель положення камери з відомого попереднього положення і оціненої швидкості руху;

б) з поточного кадру будується піраміда зображень різного масштабу і на ній запускається пошук кутів за допомогою алгоритму FAST;

в) з точок карти вибирається невеликий набір кращих ключових точок і проектується на поточний кадр;

г) у невеликих околицях точок карти після проекції на зображення шукаються відповідності серед ключових точок, виділених детектором кутів на кроці б;

д) з отриманих відповідностей уточнюється апріорне положення камери;

е) кроки в-д повторюються з використанням вже більшої кількості точок карти для точнішого визначення поточного положення.

Детектор кутів FAST є єдиною частиною системи, не залежної від розміру карти і кількості ключових кадрів, що відповідно не допускає прискорення за допомогою зменшення карти. Адаптуючи початковий алгоритм для роботи на iPhone, автори з'ясували, що через невелику частоту камери і слабкий процесор розмитість отримуваних кадрів не дозволяє спиратися при трекінгу тільки на точки, отримані детектором кутів FAST. Там підхід уточнення положення в два етапи з використанням кутів був замінений на пошук точок карти в грубіших рівнях піраміди зображень (L1 або L2, залежно від міри розмитості зображення). Ця зміна не сильно погіршувала точність трекінгу, але поліпшила продуктивність і стійкість до швидких поворотів камери.

В процесі реалізації бібліотеки для Android було проведено порівняння початкового варіанту трекінгу з вище описаним спрощеним. Але оскільки воно не виявило істотної різниці в продуктивності при погіршенні якості трекінгу, було вирішено повернутися до алгоритму, що включає детектор кутів FAST зі зменшеною кількістю точок карти, використовуваних на обох кроках алгоритму трекінгу. Кількість точок на

кроці грубого наближення була обмежена до 30, загальна кількість точок для обчислення нової позиції не повинна перевищувати 500.

5.3.2 Оптимізація ключових кадрів

Розглянемо обчислювальну складність початкового алгоритму побудови карти і його масштабованість при збільшенні кількості ключових кадрів і кількості точок в карті. Нехай карта оточення складається з N точок і M ключових кадрів, причому кожна точка міститься в T ключових кадрах. Різні частини алгоритму bundle adjustment мають різну асимптотику: проектування, диференціювання і зворотна підстановка здійснюються за $O(NT)$, генерація скороченої системи – $O(NT)^2$ і рішення системи займає $O(M)^3$. Тому найкращим варіантом оптимізації алгоритму є видалення з карти зайвих точок і ключових кадрів. Видалення поганих точок з карти вже було реалізоване в оригінальному алгоритмі PTAM. А саме, точки вважалися поганими, якщо алгоритм bundle adjustment визначав їх як викиди в процесі оптимізації. При первинній побудові карти через досить невеликий кут огляду камери усі ключові кадри, що додаються в систему, важливі для забезпечення достатньої щільності точок в карті. Але згодом можлива ситуація, коли деякі кадри будуть зайвими, тобто вони оглядають ту частину карти, яка вже добре визначена іншими ключовими кадрами.

Додамо в процес побудови карти додатковий крок фільтрації готової карти. Коли кількість ключових кадрів перевершує порогове значення (наприклад M = 50), потік, що будує карту здійснює ранжирування поточних кадрів по їх корисності. Кожна точка голосує за той ключовий кадр, який визначає її краще за інших, потім ці голоси підсумовуються для усіх ключових кадрів. Якщо після такого ранжирування в карті є ключові кадри, корисність яких не перевершує фіксованого порогового значення, вони видаляються з карти.

5.4 Реалізація збереження карти

Реалізація збереження, завантаження і перемикання між картами була грунтована на статті [10], що описує розширення початкового алгоритму PTAM для роботи з декількома картами. Карта оточення складається з двох основних частин – набір точок карти з дескрипторами і набір ключових кадрів. В якості формату серіалізації карти був

вибраний формат XML, що дозволяє зручно зберігати необхідні атрибути ключових точок і кадрів. Додаткова синхронізація була додана для можливості заблокувати карту, щоб інші потоки не могли її змінювати вчасно збереження або завантаження.

Для кожної точки карти зберігаються наступні атрибути:

– положення точки у світі, основна інформація про точку карти;

– дескриптор точки – початковий ключовий кадр, рівень в піраміді зображень, з якого прийшла ця точка і її координати на зображенні. Потрібний для пошуку відповідностей під час трекінгу;

– додаткова інформація від трекера про те, як ця точка проявляла себе при трекінгу – кількість разів, коли їй знаходилася відповідність на кадрі і викидом. Ці дані використовуються при виборі маленької підмножини точок для першої фази трекінгу.

Інформація про ключовий кадр включає:

– чорно-білу версію зображення з камери. Потрібна для обчислення дескрипторів точок і знаходження нових ключових точок на кадрі;

– положення камери в цьому кадрі. Основна інформація про ключовий кадр, що використовується при трекінгу;

– особливі точки на зображенні, виділені у момент додавання ключового кадру в карту, і виміри, що відповідають їм: рівень в піраміді зображень, положення на зображенні і в просторі камери.

5.5 Результати роботи додатку

Розробка і тестування здійснювалися на смартфоні Asus Zenfone 2 з операційною системою Android 5.0 і наступними технічними характеристиками: процесор Quad - core 2.3 GHz, оперативна пам'ять 4 GB RAM. Відео з камери пристрою роздільною здатністю 640x480, з частотою 30 кадрів в секунду. Тестовий додаток, що використовує реалізовану бібліотеку для трекінгу, надає можливість додавання тривимірного об'єкту на сцену в заданих локальних координатах.

В якості оточення для тестування системи була вибрана кімната з великою кількістю предметів, що надають достатню кількість ознак для трекінгу. В процесі ініціалізації (рис. 5.2) відбувається повільний рух камери паралельно столу і будується початкова карта з близько 150 точок і два ключових кадри. Вважатимемо рух камери швидким, якщо вона менш ніж за дві секунди переходить до кадру, що не перетинається з початковим.

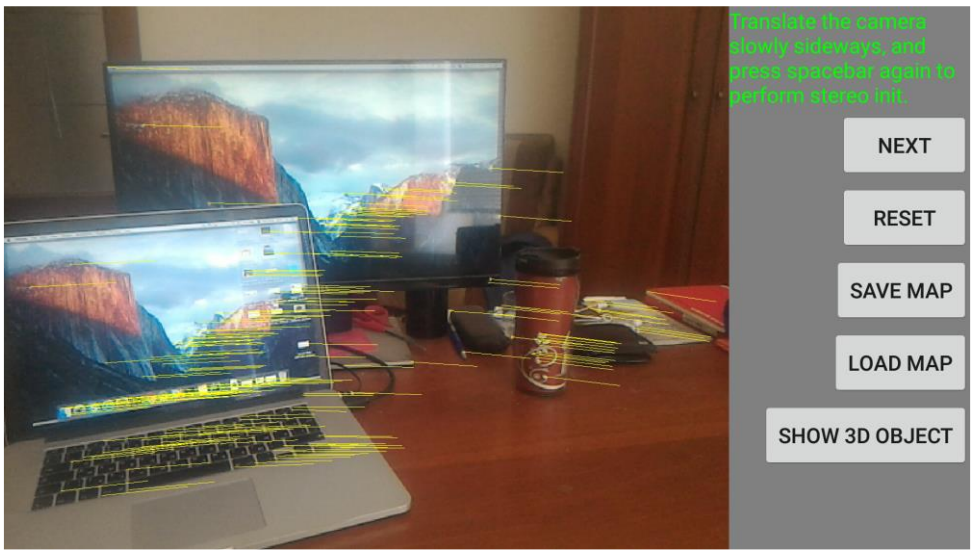


Рисунок 5.2 – Процес ініціалізації у додатку

Якщо відразу після ініціалізації камера швидко пересувається на місце, що не містить точок початкової карти, трекінг втрачається. Тому для успішної побудови карти спочатку необхідно повільно і рівномірно пересувати камеру по оточенню, тоді якість трекінгу виходить достатньою (введена метрика якості трекінгу не нижче 0.6), і нові кадри і точки успішно додаються в карту.

Подальший процес трекінгу (рис. 5.3) вже стійкіший до швидких переміщень камери: навіть якщо за наявності сильно розмитих кадрів виявляється мало точок карти, при стабілізації камери система швидко відновлюється і локалізується в побудованій карті.

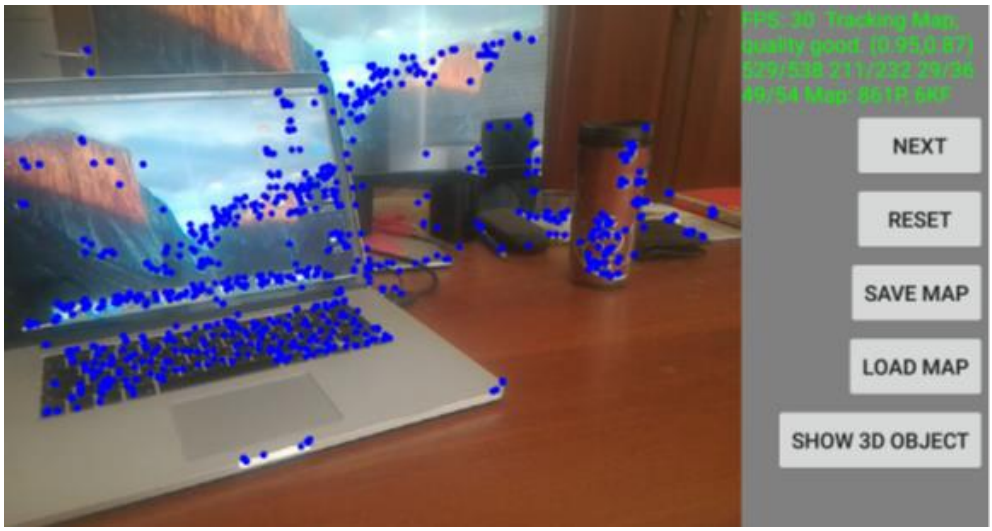


Рисунок 5.3 – Процес трекінгу карти

На рисунку 5.4 показана реалізація додавання об’єкту доповненної реальності у додатку.

Рисунок 5.4 – Реалізація додавання об’єкту

Розглянемо графіки, що ілюструють процес побудови карти оточення з використанням цієї системи. Тут камера після проведення ініціалізації рухалася рівномірно, постійно оглядаючи нові ділянки оточення. Підсумковий розмір карти складає близько 150 ключових кадрів і 3000 точок. На рисунку 5.5 видно, що ключові кадри рівномірно додавалися в карту при русі камери, відповідно потік побудови карти успішно справлявся зі своїми завданнями.

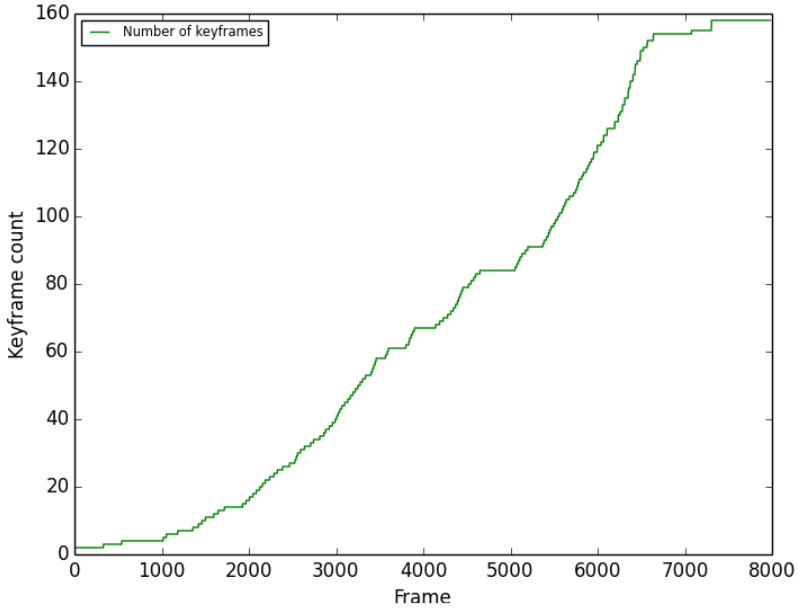


Рисунок 5.5 – Кількість ключових кадрів

При цьому, оскільки камера практично не поверталася до місць із вже побудованою картою, усі ключові кадри виявлялися значимими і тому їх кількість не зменшувалася. Залежність кількості точок від часу роботи системи на рисунку 5.6 показує,

як при збільшенні розміру карти окрім додавання нових точок в карту відбувається процес фільтрації карти і видалення зайвих або поганих точок.

Графік на рисунку 5.7 підтверджує високу продуктивність системи і її придатність для роботи в реальному часі: середній час обробки одного кадру для карти розміром до 3000 точок не перевищує 15 мс.

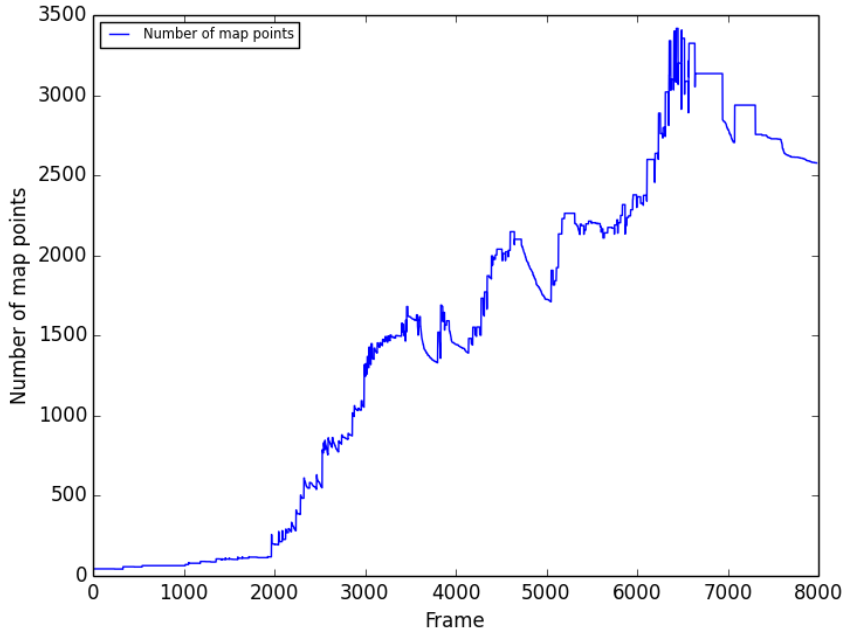


Рисунок 5.6 – Залежність кількості точок від часу роботи системи

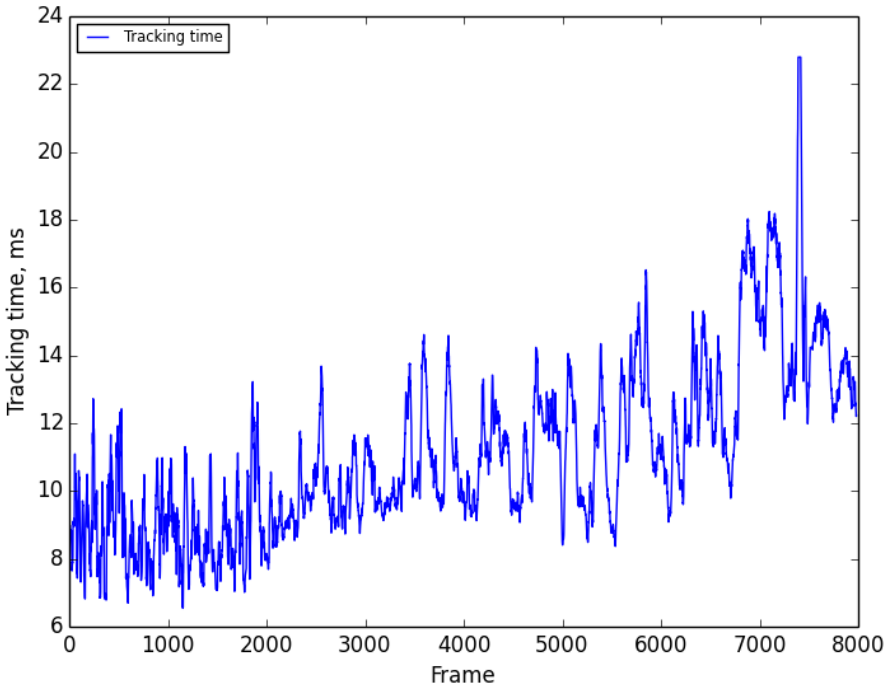


Рисунок 5.7 – Час трекінгу одного кадру

5.6 Недоліки і можливі поліпшення

Одним з недоліків використання алгоритму PTAM є те, що він в основному спирається на візуальну інформацію про оточення і у меншій мірі на просторову. Тому просте збереження карти для подальшої локалізації виявляється нестійким навіть до невеликих змін в оточенні. Для поліпшення стабільності локалізації варто розглянути додавання в карту нових видів ознак, стійкіших до зміни в оточенні і зміни освітлення. Простим варіантом таких ознак можуть бути ребра і грані, виділені із зображень.

Виділення достатньої кількості ключових точок за допомогою детектора кутів є необхідним для успішної роботи системи. Це накладає певні обмеження на оточення, в якому вона може працювати: в однотонних і слабкотекстурованих сценах можливе виникнення проблем. Також система є нестійкою до сильних змін в освітленні. Процес двоетапної ініціалізації не дозволяє працювати в оточенні занадто великого масштабу, оскільки стає складно провести ініціалізацію з достатнім лінійним зрушенням камери.

Найцікавішим варіантом розвитку такої системи є додавання можливості побудови тривимірної моделі оточення і використання її для подальшого трекінгу.

6 ПОБУДОВА КАРТИ З ПРИВ'ЯЗКОЮ ДО МІСЦЕВОСТІ

Алгоритми, розглянуті в роботі, дозволяють робити локалізацію в заздалегідь невідомому оточенні. При цьому при розміщенні віртуального об'єкту в реальному місці буває недостатньо помістити його в задану точку в системі координат камери, а хотілось би отримувати прив'язку до реальних об'єктів. Наприклад, це може бути конкретна пам'ятка або будівля. Таким чином, отриманої за допомогою PTAM карти місцевості виявляється недостатньо і з'являється необхідність побудови схожої карти для трекінгу, але з прив'язкою до конкретного місця.

У цьому розділі розглянемо способи комбінації трекінгу невідомого оточення і апріорної інформації про місце для отримання точної прив'язки до глобальних координат. В якості грубого наближення поточного положення спостерігача найзручнішим і природнішим способом є координати GPS.

6.1 Вибір джерела візуальної інформації

В якості джерела візуальної інформації для локалізації на місцевості можуть виступати будь-які зображення або відео, доступні в потрібній кількості для цієї місцевості. Підходящим варіантом джерела таких зображень є панорами з Google Maps Street View, оскільки вони з високою щільністю рівномірно покривають місцевість.

У статті [11] описано використання таких панорам для визначення точного положення зйомки фотографій. На рисунку 6.1 показана схема роботи цієї системи локалізації.

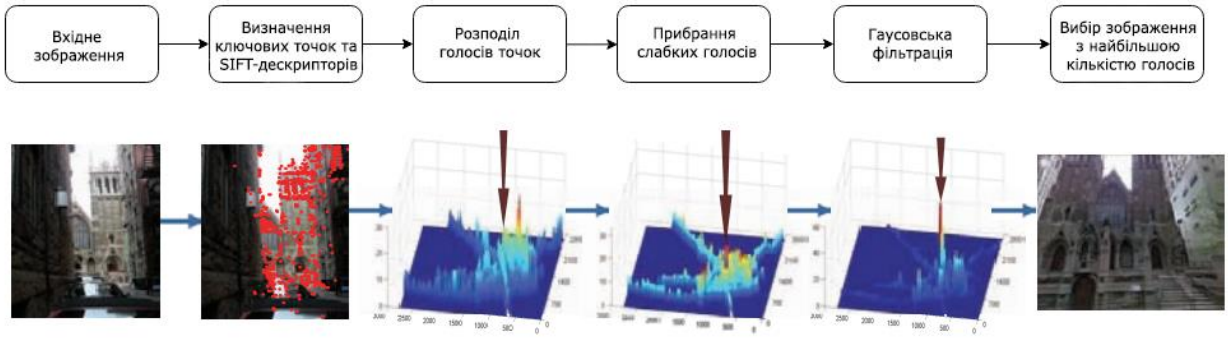


Рисунок 6.1 – Принцип роботи алгоритму локалізації на основі зображень

Узявши за основу датасет з 100000 панорам, автори запропонували наступний алгоритм для локалізації зображення за запитом:

– в процесі попередньої обробки початкового датасета будується структура даних дерево найближчих сусідів, що містить SIFT-дескриптори усіх зображень з набору;

– для ключових точок тестового зображення обчислюються SIFT- дескриптори;

– після пошуку найкращих відповідностей між дескрипторами кожна ключова точка зображення "голосує" за своє місце розташування;

– отриманий розподіл фільтрується і вибирається положення з найбільшим числом голосів.

Цей метод дозволяє з високою точністю визначати реальне положення знятої фотографії. Додатковим плюсом використання панорам з Google Street View є можливість отримання просторової інформації про місцевість. А саме, разом із зображенням сферичної панорами, знятої в заданих координатах GPS, для кожної точки зображення можна отримати інформацію про площину, на якій ця точка знаходиться. Відповідно, обчислюючи відстані до таких площин, можна побудувати карту глибин і побудувати тривимірну хмару точок.

Альтернативним підходом є використання нетекстурованої карти, також відомої як 2.5D карта: двовимірні координати будівель і їх висоти. Таку карту для міст можна отримувати з відкритих даних OpenStreetMap. У роботі [12] був запропонований метод для локалізації на вулиці, заснований на такій карті.

Як ознаки там використовуються ребра, що виділяються на зображеннях. Ключова ідея методу полягає в тому, що велика частина видимих ребер знаходяться на будівлях і є або горизонтальними, або вертикальними. Це дозволяє здійснювати точну локалізацію спостерігача, зіставляючи видимі ребра з їх положеннями на карті OpenStreetMap.

6.2 Прив'язка карти PTAM до місцевості

В якості першого кроку до автоматичної побудови карти місцевості розглянемо наступне завдання: маючи побудовану карту з PTAM, намагатимемося прив'язати її до реальної місцевості. Для простоти розглянемо випадок, коли відома точна GPS-координата поточного місця і можна однозначно визначити панораму, що відповідає поточному місцю.

Google Street View API дозволяє викачувати панорами в хорошій якості за допомогою http-запросів. А саме, спочатку виходить файл з метаінформацією у форматі xml, в якому міститься опис панорами – ідентифікатор поточної панорами і сусідніх з нею, назва вулиці і закодована карта глибини. По ідентифікатору панорами можна викачувати сферичне зображення у бажаній якості по частинах, задаючи рівень і індекси частин. Карта глибин представляє з себе зображення розміру 512x256, в якому записані індекси площин, а також опис самих площин. Зіставляючи точки на панорамі з цим зменшеним зображенням, можна отримати площину, на якій знаходиться точка і відповідно вичислити відстань до неї.

Для пошуку відповідностей між зображеннями скористаємося детектором ключових точок SIFT. Після знаходження відповідностей між ключовими точками поточного кадру і точками на отриманій панорамі з відомими реальними положеннями у світі, треба знайти перетворення, що їх зв'язує. Для цього скористаємося алгоритмом знаходження кращої гомографії на основі методу RANSAC.

На рисунку 6.2 можна бачити приклад результату такої прив'язки: ліворуч – сферична панорама з Google Street View, справа – кадр того ж місця з камери. Лініями показані відповідності між ключовими точками карти і точками на панорамі.

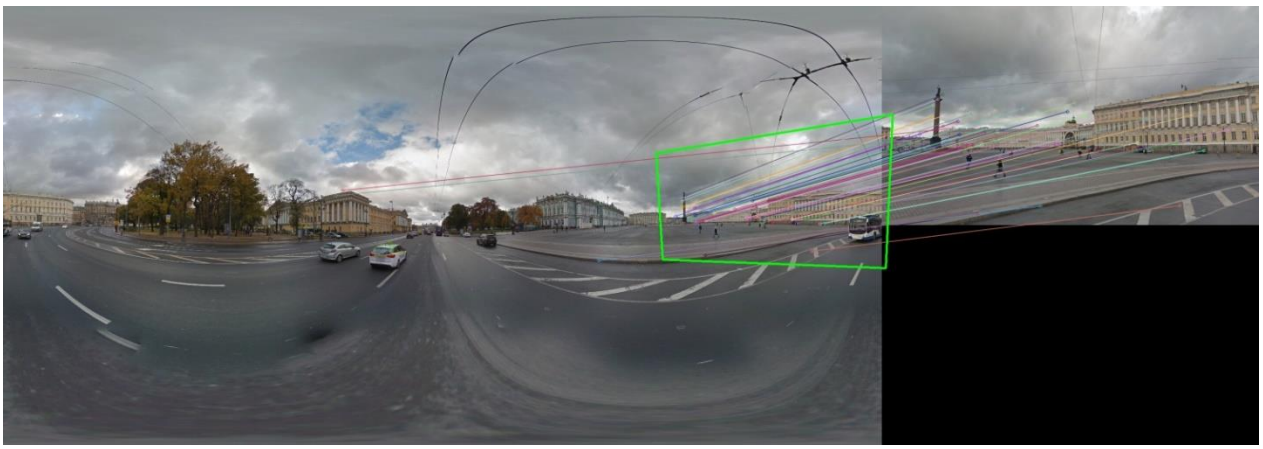


Рисунок 6.2 – Приклад прив'язки поточного кадру до панорами

Застосовуючи описаний метод в процесі роботи системи, можна отримувати точну прив'язку локальної системи координат, побудованої при ініціалізації PTAM, до світових координат. Основною умовою для проведення успішної локалізації є наявність даних, що точно описують поточне оточення і що дозволяють знайти достатню кількість відповідностей. Також можна застосовувати для локалізації стійкіші до змін в оточенні ознаки, що використовують просторову інформацію, наприклад ребра або грані.

6.3. Ініціалізація з використанням панорам

Необхідність ручної ініціалізації робить незручним використання системи, заснованої на PTAM, в оточенні великого масштабу. У роботі [13] був запропонований метод підготовки карти оточення у вигляді хмари точок для подальшого трекінгу з використанням 3d реконструкції з відео. В якості джерела було використано відео, зняте зі всенаправленої камери. Виділяючи з відео ключові кадри за допомогою семплювання з постійним інтервалом, отримані зображення, що потім використовувалися для побудови тривимірної хмари точок з використанням техніки structure from motion.

Для побудови хмари точок з панорам Google Street View розглянемо два підходи:

а) виділяючи ключові точки на одній панорамі, обчислювати їх положення у світі, використовуючи інформацію про відстані з карти глибин, і додавати в хмару точок, з якої виходить карта оточення;

б) знаходячи відповідності між точками сусідніх панорам, робити тріангуляцію і додавати їх в карту.

Перший підхід є оригінальним методом, не вивченим до цього. Він поєднує в собі використання візуальної і просторової інформації про місце, яку можна автоматично отримувати знаючи наближене значення геопозиції з датчика GPS.

Із застосуванням цього підходу був розроблений наступний алгоритм:

– за допомогою Google Street View API викачується потрібна панорама і карта глибин для неї;

– сферична панорама розділяється на пласкі ділянки і вони перетворюються для виправлення спотворень. Ділянки вибираються так, щоб вони приблизно відповідали кадрам, знятим звичайною камерою з центру;

– отримані зображення стають ключовими кадрами карти;

– для кожного з кадрів будується піраміда зображень, аналогічна PTAM, і детектором кутів виділяються ключові точки;

– використовуючи карту глибин, визначаються світові координати для точок;

– отримана структура зберігається і використовується в PTAM в якості початкової карти.

Прототип побудови такої карти був реалізований у вигляді окремої утиліти з використанням бібліотеки комп'ютерного зору OpenCV. Отримуючи на вхід GPS координати місця, вона знаходить панораму, що відповідає їм, і будує описану карту в тому ж форматі, в якому зберігається карта PTAM.

Для тестування запропонованих методів прив'язки карти до панорам і автоматичної побудови карти був вибраний набір з десяти панорам вулиць. За допомогою утиліти побудови карти були побудовані карти відповідного оточення і збережені на пристрій для подальшого трекінгу за допомогою PTAM. Проводилися два типи експериментів:

– спочатку робилася ініціалізація PTAM, потім запускався пошук відповідностей у наявній панорамі;

– трекінг за допомогою PTAM тільки на основі наявної карти.

Основною проблемою виявилися значні невідповідності зображень з панорам і поточних кадрів з камери. Прив'язка до панорами працювала досить добре, коли на зображеннях були характерні об'єкти, що не зустрічаються в інших частинах зображення: таким чином були знайдені відповідності для 8 з 10 панорам.

Другий підхід трекінгу тільки на основі готової карти не дав досить добрих результатів: іноді системі вдавалося локалізуватися на якійсь невеликій частині карти, що містить хороші ознаки (наприклад пам'ятник характерної форми), але після відведення камери від цієї ділянки трекінг втрачався.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯЇХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даної роботи був огляд сучасних сфер використання та технологій доповненої реальності, розробка бібліотеки для мобільних пристроїв на платформі Android з урахуванням особливостей платформи. Так як в процесі проектування використовувалося комп’ютерне обладнання, то аналіз потенційно небезпечних і шкідливих чинників виконується для персонального комп'ютера, на якому буде виконуватися розробка.

7.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці» [1] визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

Основним організаційним напрямом у здійсненні управління в сфері охорони праці є усвідомлення пріоритету безпеки праці і підвищення соціальної відповідальності держави і особистої відповідальності.

7.2 Аналіз стану умов праці

Робота проходитиме в приміщенні багатоквартирного будинку. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп’ютером.

7.2.1 Вимоги до приміщень

Геометричні розміри приміщення зазначені в табл. 4.1.

Таблиця 4.1 – Розміри приміщення

Найменування	**Значення**
Довжина, м	5
Ширина, м	5
Висота, м	2.8
Площа, $м^2$	25
Об’єм, $м^3$	70

Згідно з [2] розмір площі для одного робочого місця оператора персонального комп’ютера має бути не менше 6 кв. м, а об’єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

7.2.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за [3] (табл. 4.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Приміщення знаходиться на другому поверсі трьох поверхової будівлі і має об’єм 70 $м^3$, площу – 25 $м^2$. Обладнано одне місце праці укомплектоване ПК.

Таблиця 4.2 - Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	700	680 ÷800
Висота простору для ніг, мм	750	не менше 600
Ширина простору для ніг, мм	550	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	500	400 ÷500
Ширина сидіння, мм	450	не менше 400
Глибина сидіння, мм	470	не менше 400
Висота поверхні спинки, мм	400	не менше 300
Ширина опорної поверхні спинки, мм	400	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	750	700 ÷800

Температура в приміщенні протягом року коливається у межах 18–24°С, відносна вологість — близько 50%. Швидкість руху повітря не перевищує 0,2 м/с. Шум знаходиться на рівні 50 дБА. Система вентилювання приміщення — природна неорганізована, а опалення — централізоване.

7.2.3 Навантаження та напруженість процесу праці

За фізичним навантаженням робота відноситься до категорії легкі роботи (Іа), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав’язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту.

Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви програм тривалістю 15 хв. через кожну годину роботи.

7.3 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

7.3.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Аналіз небезпечних та шкідливих факторів виконується у табличній формі (табл. 4.3). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання НПАОП 0.00-7.15-18 [6] «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними

пристроями», які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП.

Основними робочими характеристиками персонального комп'ютера є:

- робоча напруга U=+220В +-5%;
- робочий струм I=2А;
- споживана потужність Р=350 Вт.

Робоче місце має відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [3].

Таблиця 4.3 – Аналіз небезпечних і шкідливих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
фізичні			
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	[4]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[5]
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	[5]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	[6] [3]
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці- сидіння користувача,) та організації робочого часу - безпервна робота)	2	[6] [3]

7.3.2 Пожежна безпека

Небезпека розвитку пожежі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ, вентиляції і кондиціювання.

Запобігти утворенню горючого середовища (замінити горючі речовини і матеріали на негорючі і важкогорючі) не надається технічно можливим. Тому проектом передбачаються засоби запобігання утворення (або внесення) в горюче середовище джерел запалювання.

Згідно ДБН В.2.5-28:2018 [5] таке приміщення, площею 25 м^2, відноситься до категорії "В" (пожежонебезпечної) та для протипожежного захисту в ньому можливо встановлення автоматичної пожежної сигналізації із застосуванням датчиків-сповіщувачів РІД-1 (сповіщувач димовий ізоляційний) в кількості 1 шт., і застосуванням первинних засобів пожежогасіння.

Продуктами згорання, що виділяються на пожежі, є: окис вуглецю; сірчистий газ; окис азоту; синильна кислота; акромін; фосген; хлор і ін. При горінні пластмас, окрім звичних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідридні кислоти, формальдегіди, хлористий водень, фосген, синильна кислота, аміак, фенол, ацетон, стирол.

7.3.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві

рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

7.4 Гігієнічні вимоги до параметрів виробничого середовища

7.4.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. Оптимальні значення мікроклімату для робочого місця відповідають ДСН 3.3.6.042-99 [2] (табл. 4.4):

Таблиця 4.4 – Норми мікроклімату робочої зони об’єкту

Період року	**Категорія робіт**	**Температура C^0**	**Відносна вологість %**	**Швидкість руху повітря, м/с**
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату. Дане приміщення обладнане системою опалення, кондиціонування повітря. Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

Рівні позитивних і негативних іонів у повітрі мають відповідати ДСН 3.3.6.042-99 [2].

7.4.2 Освітлення

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності.

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає ДБН В.2.5-28:2018 [5]. Джерелом природного освітлення є

сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n, \tag{4.1}$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \cdot b = 5 \cdot 5 = 25$ м^2,

$S = 1/8 \cdot 25 = 3{,}125$ м^2.

Приймаємо 2 вікна площею S=1,6 м^2 кожне.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні.

Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \tag{4.2}$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 25 м^2;

Z – поправочний коефіцієнт світильника (Z=1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.2), отримуємо:

$$n = \frac{300 \cdot 25 \cdot 1{,}1 \cdot 1{,}5}{5400 \cdot 0{,}575 \cdot 2} \approx 2{,}0$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

7.4.3 Шум та вібрація, електромагнітне випромінювання

Рівень шуму, зумовлений як роботою системного блоку, клавіатури, так і друкуванням на принтері, а також зовнішніми чинниками, коливається у межах 50–65 дБА ДСН 3.3.6.042-99 [2].

Віброізоляцію можливо здійснювати за допомогою спеціальної прокладки під системний блок, яка послаблює передачу вібрацій робочого столу. Вібрація на робочому місці в приміщенні, що розглядається, відповідає нормам ДСН 3.3.6.042-99 [2].

7.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання наводимо приклади деяких заходів безпеки.

1) Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;
- експлуатацію сертифікованого обладнання;
- дотримання заходів електробезпеки;
- забезпечення оптимальних параметрів мікроклімату;
- забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);
- облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря.

2) Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

- постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, ламповых патронів, а також мережевих кабелів живлення, за допомогою яких електроприлади під'єднують до електромережі;

- постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;

- не тягнути за мережевий кабель, щоб витягти вилку з розетки;

- не закривати меблями, різноманітним інвентарем вимикачі, штепсельні розетки;

- не підключати одночасно декілька потужних електропристроїв до однієї розетки, що може викликати надмірне нагрівання провідників, руйнування їхньої ізоляції, розплавлення і загоряння полімерних матеріалів;

- не залишати включені електроприлади без нагляду;

- не допускати потрапляння всередину електроприладів крізь вентиляційні отвори рідин або металевих предметів, а також не закривати їх та підтримувати в належній чистоті, щоб уникнути перегрівання та займання приладу;

- не ставити на електроприлади матеріали, які можуть під дією теплоти, що виділяється, загорітися (канцелярські товари, сувенірну продукцію тощо).

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом, приміщення в якому проводиться робота відноситься до першого класу (без підвищеної небезпеки). Коефіцієнт використання вертикальних заземлювачів $\eta_в$ в залежності від розміщення заземлювачів та їх кількості знаходиться в межах 0,4…0,99. Взаємну екрануючу дію горизонтального заземлювача (з'єднувальної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача η_c.

Послідовність розрахунку.

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \qquad (4.3)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.}=R_д$.

Підставивши числові значення у формулу (4.3), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40 - 4} \approx 4 \text{ Ом}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо ρ=40 Ом•м (табличне значення).

3) Розрахунковий питомий опір ґрунту, $\rho_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом·м за формулою:

$$\rho_{розр..} = \psi \cdot \rho, \tag{4.4}$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів I кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в}$=1,7 і горизонтальних $\rho_{розр.г}$=5,5 Ом·м.

$$\rho_{розр.в} = 1,7 \cdot 40 = 68 \text{ Ом•м}$$

$$\rho_{розр.г} = 5,5 \cdot 40 = 220 \text{ Ом•м}$$

4) Розраховується опір розтікання струму вертикального заземлювача $R_в$, Ом, за формулою (4.5).

$$R_в = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot l_в} \cdot \left(\ln \frac{2 \cdot l_в}{d_{ст}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_в}{4 \cdot t - l_в} \right), \tag{4.5}$$

де $l_в$ – довжина вертикального заземлювача (для труб - 2−3 м; $l_в$=3 м);

$d_{ст}$ – діаметр стержня (для труб - 0,03−0,05 м; $d_{ст}$=0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за формулою (4.6):

$$t = h_в + \frac{l_в}{2}, \tag{4.6}$$

де $h_в$ – глибина закладання вертикальних заземлювачів (0,8 м); тоді $t = 0,8 + \frac{3}{2} = 2,3$ м

$$R_в = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5 \text{ Ом}$$

5) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_в$:

$$n = \frac{2 \cdot R_в}{R_д} = \frac{2 \cdot 18,5}{4} = 9,25 \tag{4.7}$$

I визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з'єднувальної стрічки $\eta_в$ =0,57 (табличне значення).

6) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_в$, шт:

$$n_в = \frac{2 \cdot R_в}{R_д \cdot \eta_в} = \frac{2 \cdot 18,5}{4 \cdot 0,57} = 16,2 \approx 16 \quad (4.8)$$

7) Визначається довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м:

$$l_с = 1,05 \cdot L_в \cdot (n_в - 1), \quad (4.9)$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);

$n_в$ – необхідна кількість вертикальних заземлювачів.

$$l_с = 1,05 \cdot 3 \cdot (16 - 1) \approx 48 \text{ м}$$

8) Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_с} \cdot \ln \frac{2 \cdot l_с^{\,2}}{d_{см} \cdot h_г}, \quad (4.10)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м;

$h_г$ – глибина закладання горизантальних заземлювачів (0,5 м);

$l_с$ - довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0,95 \cdot 0,15 \cdot 0,5} = 8,1 \text{ Ом}$$

9) Визначається коефіцієнт використання горизонтального заземлювача $\eta_с$. відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання з'єднувальної смуги $\eta_с$=0,3 (табличне значення).

10) Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$R_{заг} = \frac{R_в \cdot R_г}{R_в \cdot \eta_с + R_г \cdot n_в \cdot \eta_в} \le R_д. \quad (4.18)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг}$ <4 Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \le R_д$$

7.6 Екологія

Діяльність за темою магістерської роботи, а саме: огляд сучасних сфер використання та технологій доповненої реальності, розробка бібліотеки для мобільних пристроїв на платформі Android в процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища» [7], Законом України «Про забезпечення санітарного та епідемічного благополуччя населення» [8], Законом України «Про відходи» [9].

В процесі діяльності з виконання дипломного проектування виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

- Відпрацьовані люмінесцентні лампи - І клас небезпеки.
- Змінні носії інформації - IV клас небезпеки.
- Відпрацьовані вогнегасники - IV клас небезпеки.
- Макулатура - IV клас небезпеки.

Висновки до розділу 7

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики. Розглянуті заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій. У розділі «Екологія» розглянуті питання впливу на навколишнє природнє середовище діяльності за темою магістерської роботи.

ВИСНОВКИ

У результаті виконання роботи були вивчені готові рішення і бібліотеки для побудови доповненої реальності, і зроблено аналіз можливостей, які вони надають. Проведений аналіз сучасних сфер використання технологій доповненої реальності.

Були досліджені існуючі методи побудови доповненої реальності, а саме методики розпізнавання маркерів, алгоритми комп'ютерного зору, генетичні алгоритми, алгоритм future detection. Були вивчені алгоритми візуальної локалізації і трекінгу, методи семплювання точок з ребер і точок інтересу, методи трекінгу з використанням моделі оточення та в невідомому оточенні, що вирішують задачу побудови доповненої реальності в природному оточенні на мобільних пристроях на основі візуального трекінгу. Проведено дослідження алгоритмів обробки кадру, побудови та оптимізації карти.

Використовуючи вибраний алгоритм PTAM, була розроблена і програмно реалізована бібліотека для мобільних пристроїв на платформі Android з урахуванням особливостей платформи. Також проведено тестування її продуктивності для успішної роботи в реальному часі. Реалізована можливість збереження отриманої карти оточення для подальшого використання.

Була розглянута можливість використання панорам Google Street View для вирішення завдання точної локалізації у світовій системі координат. В якості варіанту рішення запропонований алгоритм прив'язки карти оточення, побудованої з використанням PTAM, до панорами з відомою геопозицією. Також реалізований прототип автоматичної побудови карти оточення на основі панорам для подальшого трекінгу

Перелік корисних посилань до розділу 7

1. Закон України «Про охорону праці». Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12 - 10.14.1992 р.

2. Державні санітарні норми. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» - Режим доступу: https://zakon.rada.gov.ua/rada/show/va042282-99 - 01.02.1999 р.

3. Державні санітарні правила і норми. ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» - Режим доступу: https://zakon.rada.gov.ua/rada/show/v0007282-98 - 10.12.1998 р.

4. Державний стандарт України. ДСТУ Б В.2.5-82:2016 «Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом» - Режим доступу: http://epicentre.co.ua/dstu/doc28522.html - 01.07.2016 р.

5. Державні будівельні норми. ДБН В.2.5-28:2018 «Природне і штучне освітлення» - Режим доступу: http://www.minregion.gov.ua/wp-content/uploads/2018/12/V2528-1.pdf - 03.10.2018

6. Нормативно-правовий акт з охорони праці. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» - Режим доступу: https://zakon.rada.gov.ua/laws/show/z0508-18 - 14.02.2018 р.

7. Закон України «Про охорону навколишнього природного середовища» - Режим доступу: https://zakon.rada.gov.ua/laws/show/1264-12 - 26.06.1991 р.

8. Закони України «Про охорону навколишнього природного середовища» - Режим доступу - https://zakon.rada.gov.ua/laws/show/4004-12 - 24.02.1994 р.

9. Закон України «Про відходи» - Режим доступу: https://zakon.rada.gov.ua/laws/show/187/98-%D0%B2%D1%80 – 05.03.1998 р.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Zhang X. Visual Marker Detection and Decoding in AR Systems: A Comparative Study [Text] / X. Zhang, S. Fronz, N. Navab. – ISMAR: Proceedings of the 1st International Symposium on Mixed and Augmented Reality, 2002. – 69 p.

2. Sezgin M. Survey over image thresholdingtechniques and quantitative performance evaluation [Електроний ресурс]/ M. Sezgin. – Режим доступу: http://www.busim.ee.boun.edu.tr/~sankur/SankurFolder/Threshold_survey.pdf

3. Nadernejad E. Edge Detection Techniques: Evaluations and Comparisons. [Електроний ресурс] / E. Nadernejad. – Режим доступу: http://www.m-hikari.com/ams/ams-password-2008/amspassword29-32-2008/nadernejadAMS29-32-2008.pdf

4. Kato H. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. [Електроний ресурс] / H. Kato, M. Billinghurst. – Режим доступу: http://www.hitl.washington.edu/artoolkit/ Papers/IWAR99.kato.pdf

5. Кляйн Д. Паралельне відстеження і картографування для малих робітничих областей AR [Текст]/Д. Кляйн, Д. Мюррей. – Нара, Японія, 2007. – 21 с.

6. Шапіро, Л. Комп'ютерний зір [Текст] / Л. Шапіро, Дж. Стокман; пер. з англ. – М.: Біном. Лабораторія знань, 2006. – 752 с.

7. Путятін, Є.П. Методи та алгоритми комп'ютерного зору [Текст] : Навч. посібник / Є.П. Путятін, В.О. Гороховатський, О.О. Матат. – Харків : ТОВ «Компанія СМІТ», 2006. – 236 с.

8. Гонсалес Р. Цифрова обробка зображень [Текст] / Р. Гонсалес, Р. Вудс - М : Техносфера, 2005. – 1072 с

9. Форсайт Д.А. Комп'ютерний зір. Сучасний підхід. [Текст] / Д.А. Форсайт, Д. Понс; пер. з англ. – К.: Вільямс, 2004. – 928 с.

10. Castle R. Video-rate Localization in Multiple Maps for Wearable Augmented Reality[Text] / R. Castle, G. Klein, D. Murray. // Proc 12th IEEE Int Symp on Wearable Computers, – Pittsburgh, USA, 2008. – p. 15–22.

11. Roshan Z. A. Accurate Image Localization Based on Google Maps Street View[Text] / Z. A. Roshan, S. Mubarak. // Proceedings of the 11th European Conference on Computer Vision: Part IV, 2010. – p. 255–268.

12. Arth C. Instant Outdoor Localization and SLAM Initialization from 2.5D Maps [Text] /C. Arth, C. Pirchheim, J. Ventura. // IEEE Transactions on Visualization and Computer Graphics, 2015. – № 11. – p. 1309–1318.

13. Jonathan V. Wide-area scene mapping for mobile visual tracking[Text] / V. Jonathan, H. Tobias. // 11th IEEE International Symposium on Mixed and Augmented Reality, ISMAR, Atlanta, GA, USA, 2012. – p. 3–12.

ДОДАТОК А

```
#include <jni.h>
#include <PTAM/TrackerData.h>
#include "PTAM/Tracker.h"
#include "PTAM/PtamSystem.h"
#include "Logger.h"
#include "TrackerRenderer.h"
using namespace PTAM;
using namespace CVD;
extern "C" {
PtamSystem* ptamSystem(jlong handle) {
    return reinterpret_cast<PtamSystem*>(handle);
}
JNIEXPORT jlong JNICALL
Java_mit_spbau_ptam_PtamSystem_nCreate(JNIEnv* env, jclass type, jint
width, jint height) {
    ImageRef imageSize(width, height);
    PtamSystem* system = new PtamSystem(imageSize);
    start_logger("ARSystem");
    return reinterpret_cast<jlong>(system);
}
JNIEXPORT void JNICALL
Java_mit_spbau_ptam_PtamSystem_nProcessFrame(JNIEnv* env, jclass type,
jlong handle, jint width,
    jint height, jbyteArray jData) {
    PtamSystem* arSystem = reinterpret_cast<PtamSystem*>(handle);
    jsize size = width * height;
    Image<byte>* pImage = arSystem->image();
    env->GetByteArrayRegion(jData, 0, size, (jbyte*) pImage->data());
    arSystem->processFrame(*pImage);
}
JNIEXPORT void JNICALL
Java_mit_spbau_ptam_PtamSystem_nNextTrackingState(JNIEnv* env, jclass
type, jlong handle) {
    Tracker* pTracker = ptamSystem(handle)->tracker();
    pTracker->PokeTracker();
```

```
}
JNIEXPORT void JNICALL
Java_mit_spbau_ptam_PtamSystem_nReset(JNIEnv* env, jclass type, jlong
handle) {
    Tracker* pTracker = ptamSystem(handle)->tracker();
    pTracker->Reset();
}
JNIEXPORT jstring JNICALL
Java_mit_spbau_ptam_PtamSystem_nGetMessage(JNIEnv* env, jclass type,
jlong handle) {
    Tracker* pTracker = ptamSystem(handle)->tracker();
    string message = pTracker->GetMessageForUser();
    return env->NewStringUTF(message.c_str());
}
JNIEXPORT jlong JNICALL
Java_mit_spbau_ptam_PtamSystem_nGetMap(JNIEnv* env, jclass type, jlong
handle) {
    return reinterpret_cast<jlong>(ptamSystem(handle)->map());
}
JNIEXPORT jlong JNICALL
Java_mit_spbau_ptam_PtamSystem_nInitRenderer(JNIEnv* env, jclass type)
{
    TrackerRenderer* renderer = new TrackerRenderer;
    return reinterpret_cast<jlong>(renderer);
}
JNIEXPORT void JNICALL
Java_mit_spbau_ptam_PtamSystem_nRenderTrackingInfo(JNIEnv* env, jclass
type, jlong hSystem,
    jlong hRenderer) {
    PtamSystem* ptam = ptamSystem(hSystem);
    TrackerRenderer*                    renderer                    =
reinterpret_cast<TrackerRenderer*>(hRenderer);
    const ImageRef& imageSize = ptam->image()->size();
    Vector<2> scale;
    scale[0] = 2.0f / static_cast<float>(imageSize.x);
    scale[1] = 2.0f / static_cast<float>(imageSize.y);
    Map* pMap = ptam->map();
    if (pMap->IsGood()) {
        //tracking
```

```
75          pMap->LockMap();
76          renderer->setProgram();
77          renderer->setColor(0, 1, 0);
78          for (auto& mapPoint: pMap->vpPoints) {
79              TrackerData* tData = mapPoint->pTData;
80              if (tData && mapPoint->bFoundRecent) {
81                  renderer->renderPoint(
82                      tData->v2Image[0] * scale[0] - 1.0,
83                      1.0f - tData->v2Image[1] * scale[1]);
84              }
85          }
86          pMap->UnlockMap();
87      } else {
88          //initializing
89          Tracker* pTracker = ptam->tracker();
90          std::list<Trail> trails = pTracker->getTrails();
91          renderer->setProgram();
92          renderer->setColor(1, 1, 0);
93          for (auto& trail: trails) {
94              renderer->renderLine(
95                  trail.irInitialPos.x * scale[0] - 1.0,
96                  1.0 - trail.irInitialPos.y * scale[1],
97                  trail.irCurrentPos.x * scale[0] - 1.0,
98                  1.0 - trail.irCurrentPos.y * scale[1]);
99          }
100     }
101 }
102 JNIEXPORT void JNICALL
103 Java_mit_spbau_ptam_PtamSystem_nGetRotation(JNIEnv* env, jclass type,
104 jlong handle, jfloatArray jRotation) {
105     jfloat* rotation = env->GetFloatArrayElements(jRotation, NULL);
106     const SE3<>& se3 = ptamSystem(handle)->tracker()->cameraPose();
107     const Matrix<3, 3>& m = se3.get_rotation().get_matrix();
108     for (int i = 0; i < 3; ++i) {
109         for (int j = 0; j < 3; ++j) {
110             rotation[3 * i + j] = (float) m[i][j];
111         }
112     }
113     env->ReleaseFloatArrayElements(jRotation, rotation, 0);
```

```
114 }
115 JNIEXPORT void JNICALL
116 Java_mit_spbau_ptam_PtamSystem_nGetPosition(JNIEnv* env, jclass type,
117 jlong handle, jfloatArray jPosition) {
118     jfloat* position = env->GetFloatArrayElements(jPosition, NULL);
119     const SE3<>& se3 = ptamSystem(handle)->tracker()->cameraPose();
120     const Vector<3>& pos = se3.get_translation();
121     for (int i = 0; i < 3; ++i) {
122         position[i] = (float) pos[i];
123     }
124     env->ReleaseFloatArrayElements(jPosition, position, 0);
125 }
126 JNIEXPORT jdouble JNICALL
127 Java_mit_spbau_ptam_PtamSystem_nGetLastTrackingTime(JNIEnv*        env,
128 jclass type, jlong handle) {
129     return ptamSystem(handle)->tracker()->lastTrackingTime();
130 }
131 JNIEXPORT jfloat JNICALL
132 Java_mit_spbau_ptam_PtamSystem_nGetTrackingQuality(JNIEnv* env, jclass
133 type, jlong handle) {
134     return ptamSystem(handle)->tracker()->trackingQuality();
135 }
136 JNIEXPORT void JNICALL
137 Java_mit_spbau_ptam_PtamSystem_nSaveMap(JNIEnv*   env,   jclass   type,
138 jlong handle, jstring fileName) {
139     const char* name = env->GetStringUTFChars(fileName, false);
140     ptamSystem(handle)->mapMaker()->SaveMap(name);
141     env->ReleaseStringUTFChars(fileName, name);
142 }
143 JNIEXPORT void JNICALL
144 Java_mit_spbau_ptam_PtamSystem_nLoadMap(JNIEnv*   env,   jclass   type,
145 jlong handle, jstring fileName) {
146     const char* name = env->GetStringUTFChars(fileName, false);
147     ptamSystem(handle)->mapMaker()->LoadMap(name);
148     env->ReleaseStringUTFChars(fileName, name);
149 }
150 JNIEXPORT jint JNICALL
151 Java_mit_spbau_ptam_Map_nGetNumPoints(JNIEnv *env, jclass type, jlong
152 handle) {
```

```
153     return (jint) reinterpret_cast<Map*>(handle)->vpPoints.size();
154 }
155 JNIEXPORT jint JNICALL
156 Java_mit_spbau_ptam_Map_nGetNumKeyFrames(JNIEnv *env, jclass type,
157 jlong handle) {
158     return (jint) reinterpret_cast<Map*>(handle)->vpKeyFrames.size();
159 }
160 JNIEXPORT jlong JNICALL
161 Java_mit_spbau_ptam_Map_nGetMapPoint(JNIEnv *env, jclass type, jlong
162 handle, jint i) {
163     MapPoint *point = reinterpret_cast<Map*>(handle)->vpPoints[i];
164     return reinterpret_cast<jlong>(point);
165 }
166 JNIEXPORT void JNICALL
167 Java_mit_spbau_ptam_MapPoint_nGetImagePos(JNIEnv *env, jclass type,
168 jlong handle,
169                                           jfloatArray imagePos_) {
170     jfloat *imagePos = env->GetFloatArrayElements(imagePos_, NULL);
171     MapPoint* mapPoint = reinterpret_cast<MapPoint*>(handle);
172     TrackerData* tData = mapPoint->pTData;
173     if (tData && mapPoint->bFoundRecent) {
174         imagePos[0] = (jfloat) tData->v2Image[0];
175         imagePos[1] = (jfloat) tData->v2Image[1];
176     }
177     env->ReleaseFloatArrayElements(imagePos_, imagePos, 0);
178 }
179 JNIEXPORT void JNICALL
180 Java_mit_spbau_ptam_MapPoint_nGetWorldPos(JNIEnv *env, jclass type,
181 jlong handle,
182                                           jfloatArray worldPos_) {
183     jfloat *worldPos = env->GetFloatArrayElements(worldPos_, NULL);
184     MapPoint* mapPoint = reinterpret_cast<MapPoint*>(handle);
185     worldPos[0] = (jfloat) mapPoint->v3WorldPos[0];
186     worldPos[1] = (jfloat) mapPoint->v3WorldPos[1];
187     worldPos[2] = (jfloat) mapPoint->v3WorldPos[2];
188     env->ReleaseFloatArrayElements(worldPos_, worldPos, 0);
189 }
190 JNIEXPORT void JNICALL
```

```
191 Java_mit_spbau_ptam_MapPoint_nGetLocalPos(JNIEnv *env, jclass type,
192 jlong handle,
193                                               jfloatArray localPos_) {
194     jfloat *localPos = env->GetFloatArrayElements(localPos_, NULL);
195     MapPoint* mapPoint = reinterpret_cast<MapPoint*>(handle);
196     TrackerData* tData = mapPoint->pTData;
197     if (tData && mapPoint->bFoundRecent) {
198         localPos[0] = (jfloat) tData->v3Cam[0];
199         localPos[1] = (jfloat) tData->v3Cam[1];
200         localPos[2] = (jfloat) tData->v3Cam[2];
201     }
202     env->ReleaseFloatArrayElements(localPos_, localPos, 0);
203 }
204 JNIEXPORT jboolean JNICALL
205 Java_mit_spbau_ptam_MapPoint_nIsTracked(JNIEnv *env, jclass type,
206 jlong handle) {
207     MapPoint* mapPoint = reinterpret_cast<MapPoint*>(handle);
208     return (jboolean) mapPoint->bFoundRecent;
209 }
210 }
```

ДОДАТОК Б

Комп’ютерна презентація

Рисунок В.1- Слайд №1

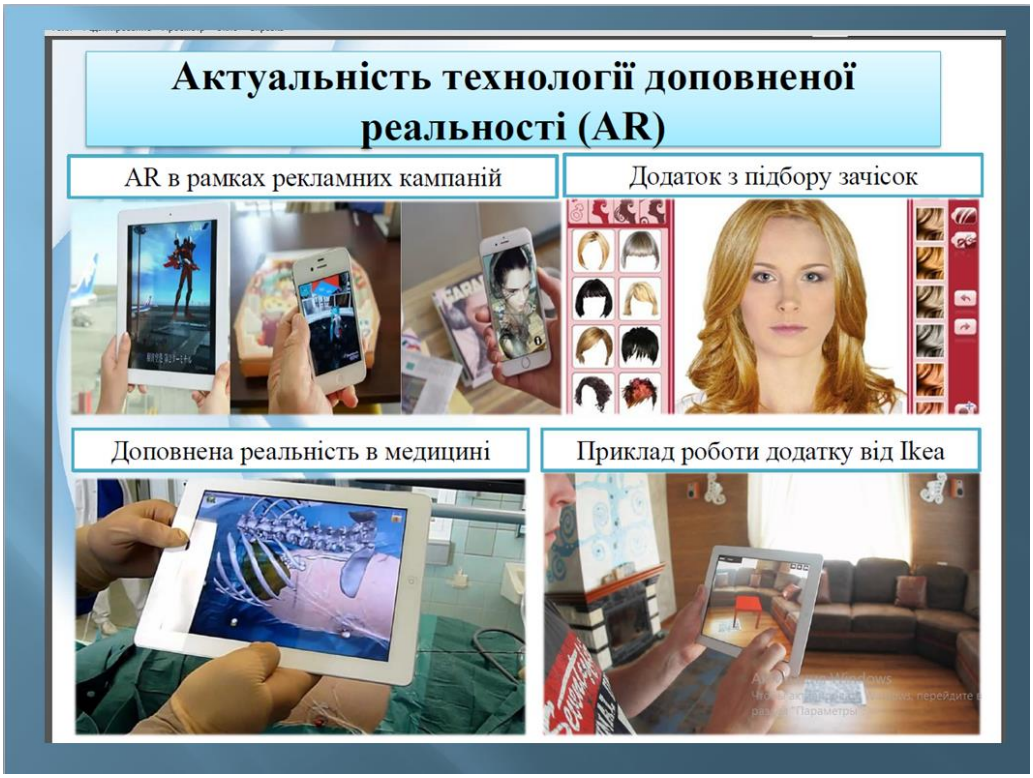


Рисунок В.2- Слайд №2

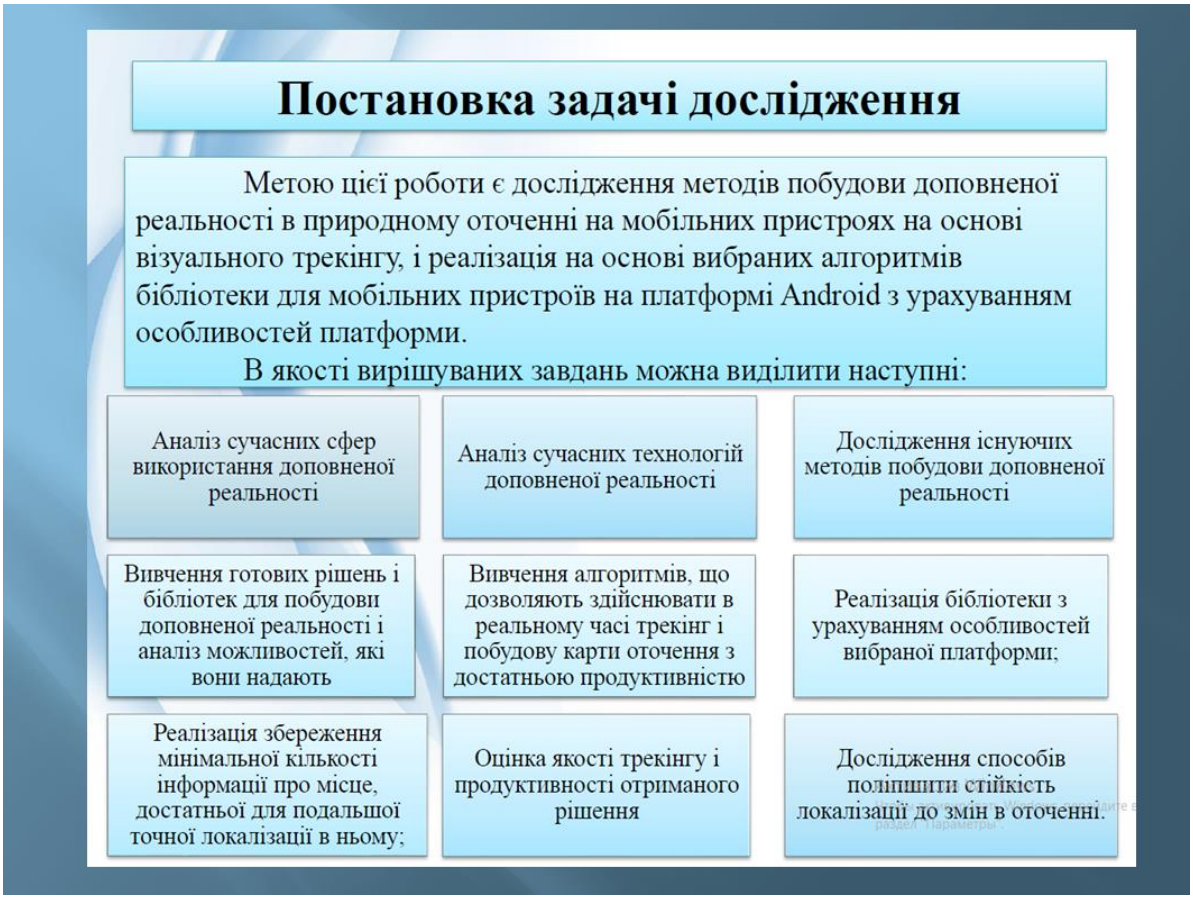


Рисунок В.3- Слайд №3

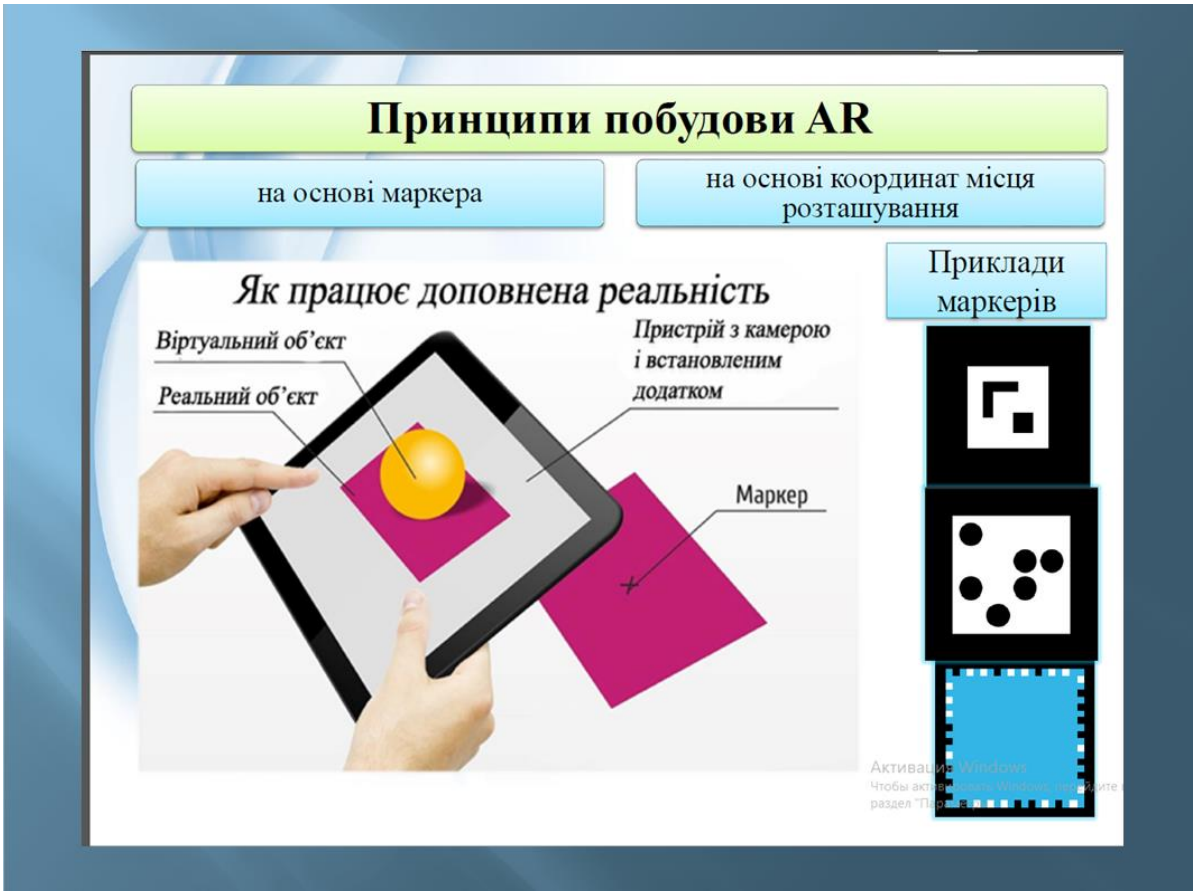


Рисунок В.4- Слайд №4

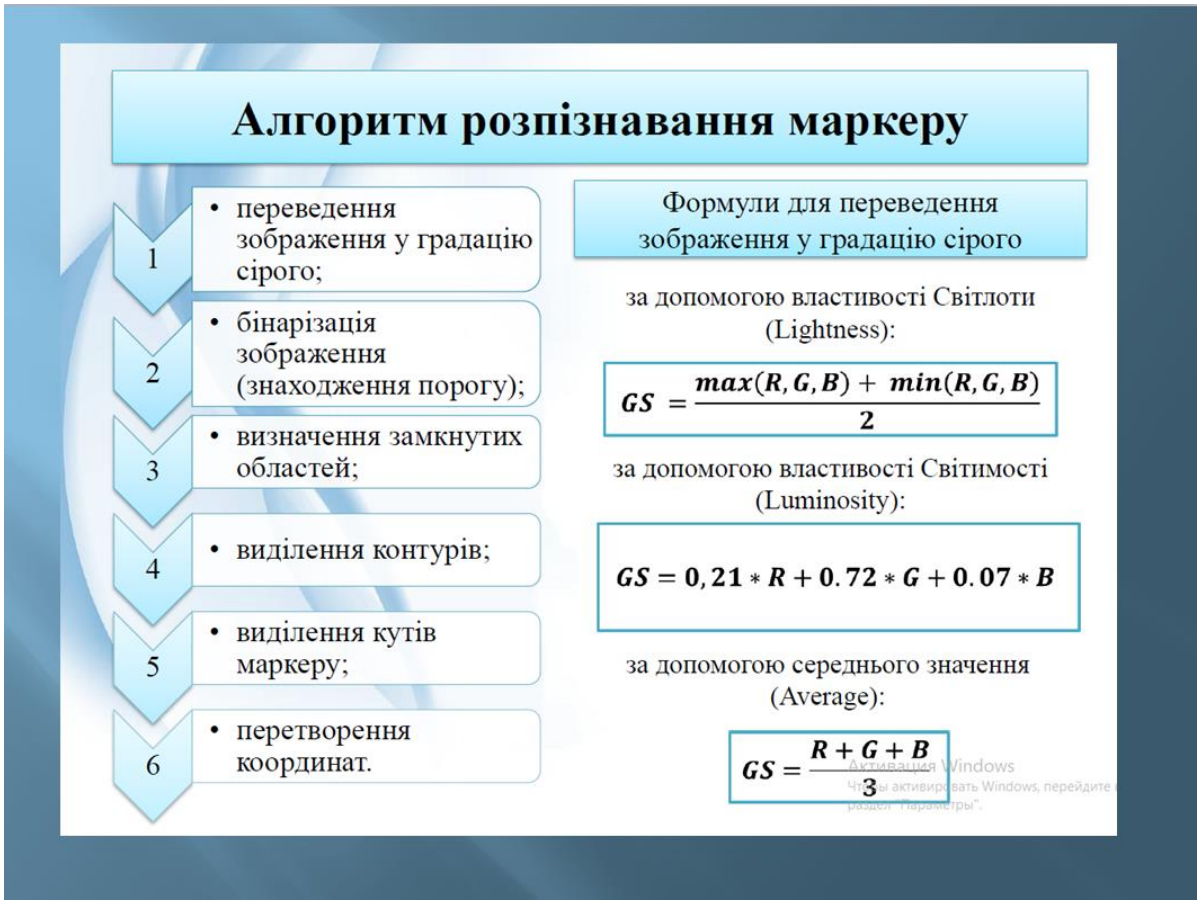


Рисунок В.5- Слайд №5

Рисунок В.6- Слайд №6

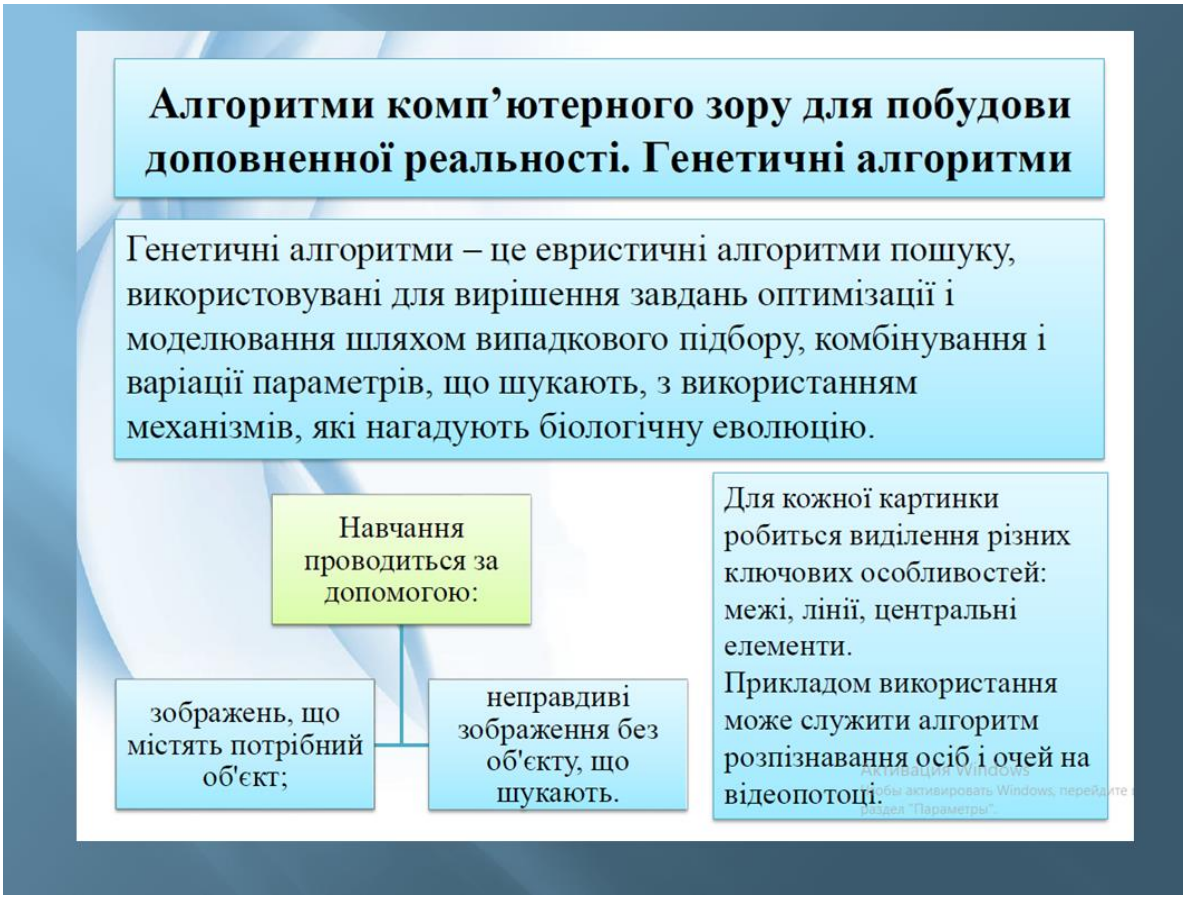


Рисунок В.7- Слайд №7

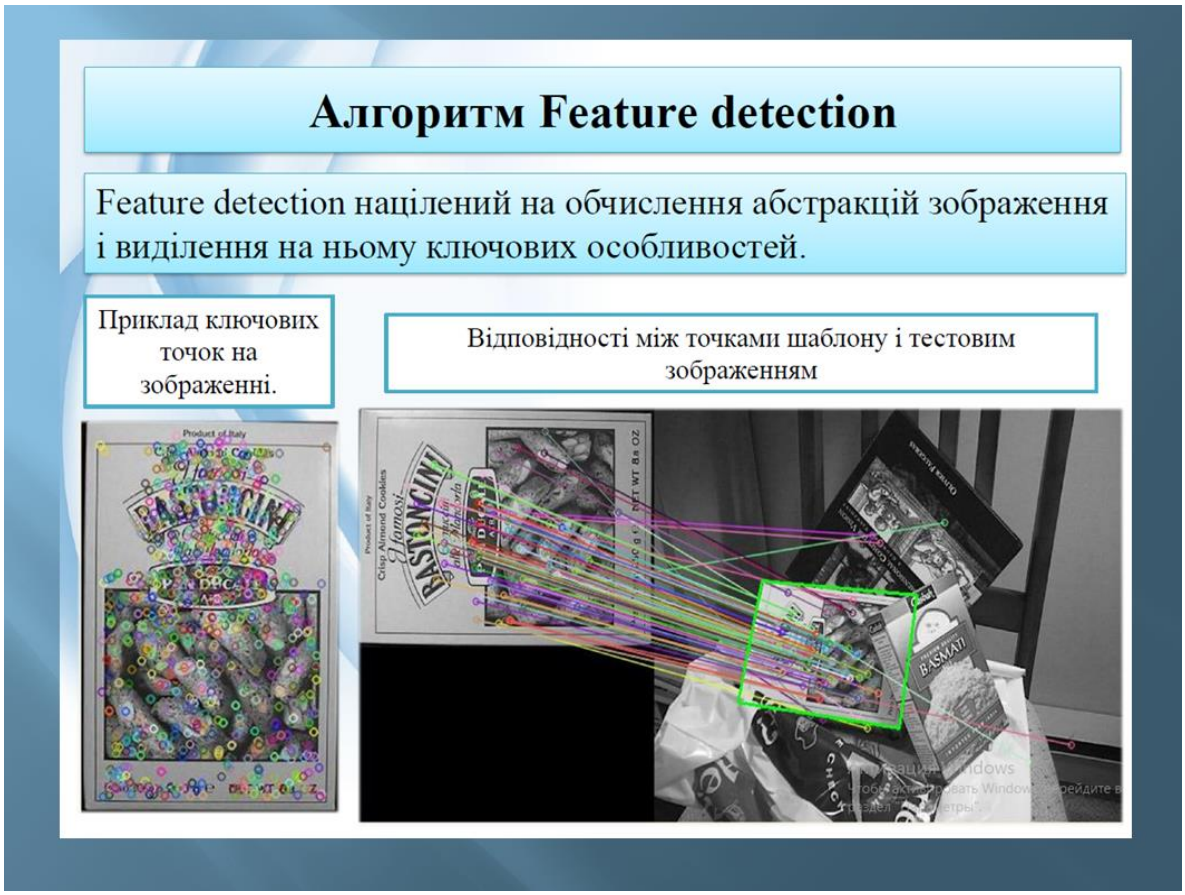


Рисунок В.8- Слайд №8

Рисунок В.9- Слайд №9

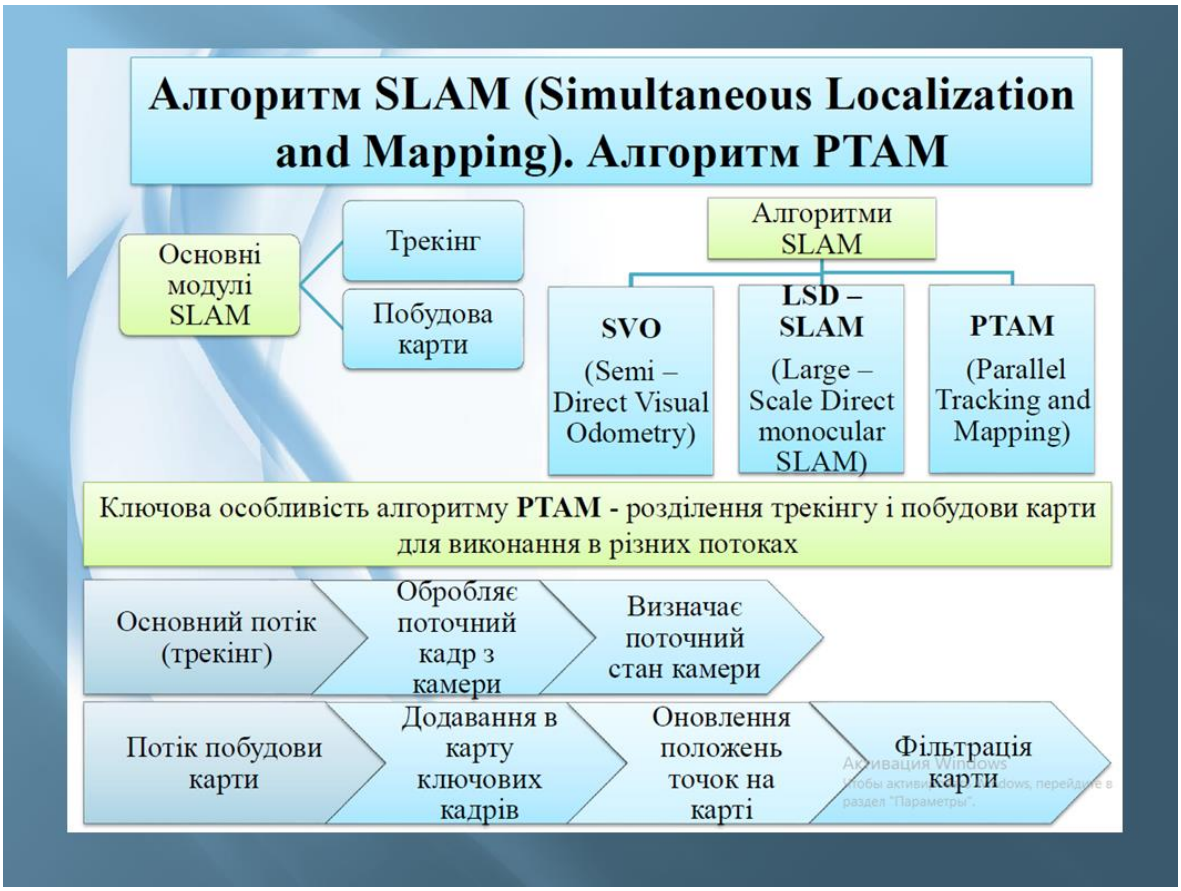


Рисунок В.10- Слайд №10

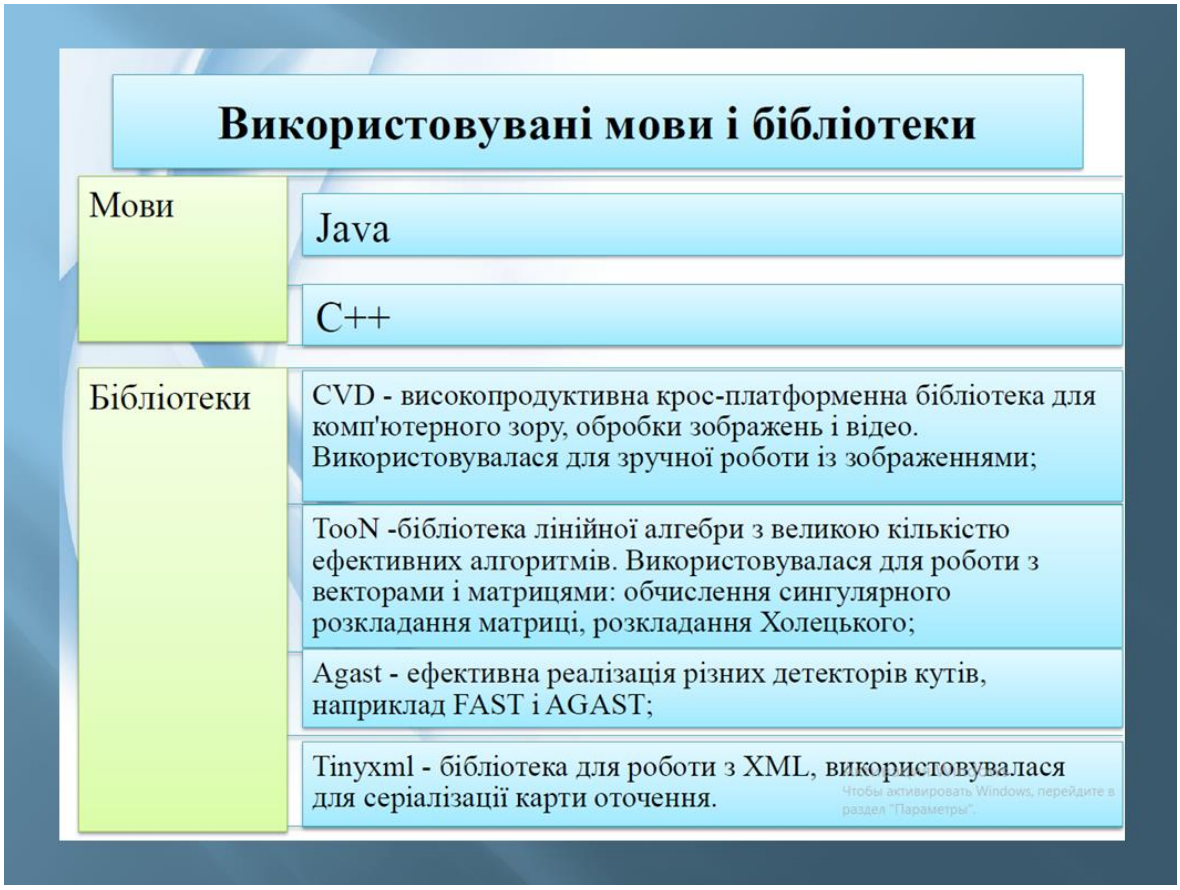


Рисунок В.11- Слайд №11

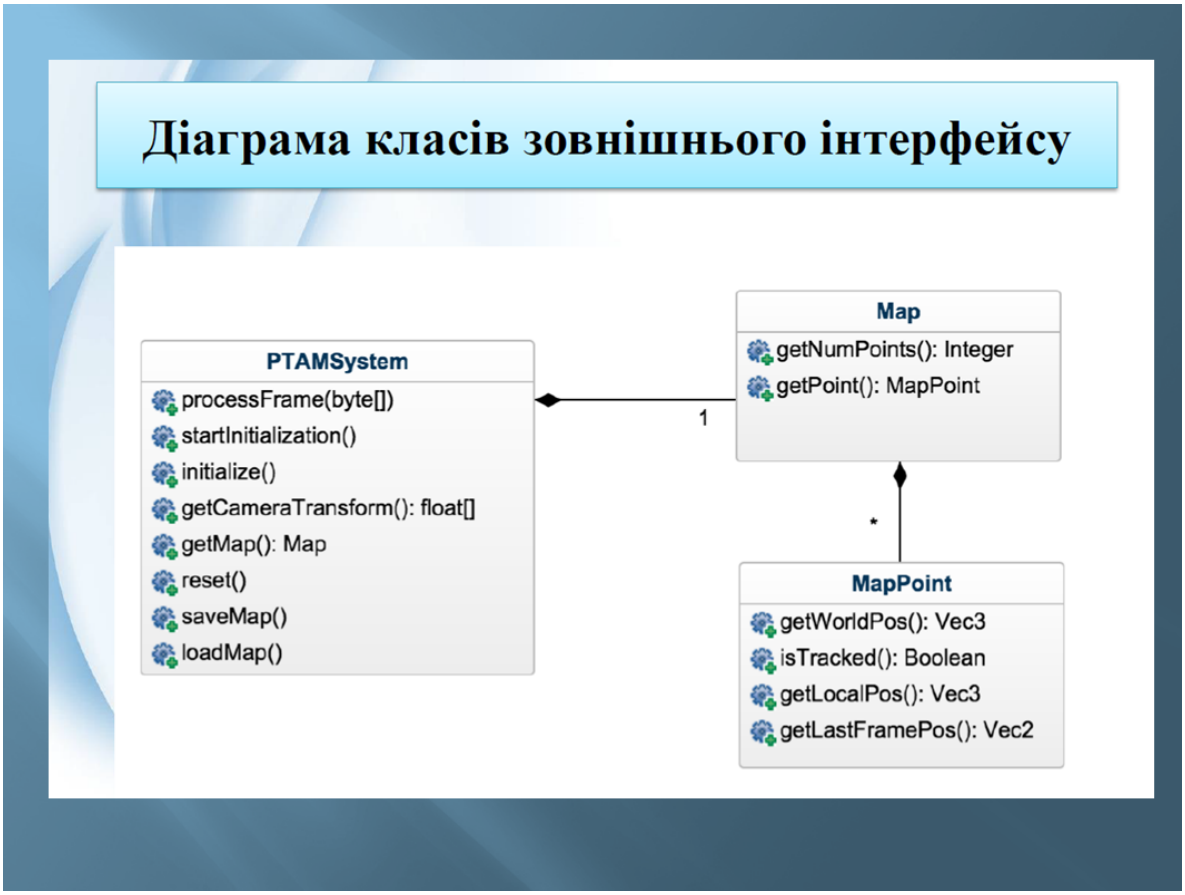


Рисунок В.12- Слайд №12

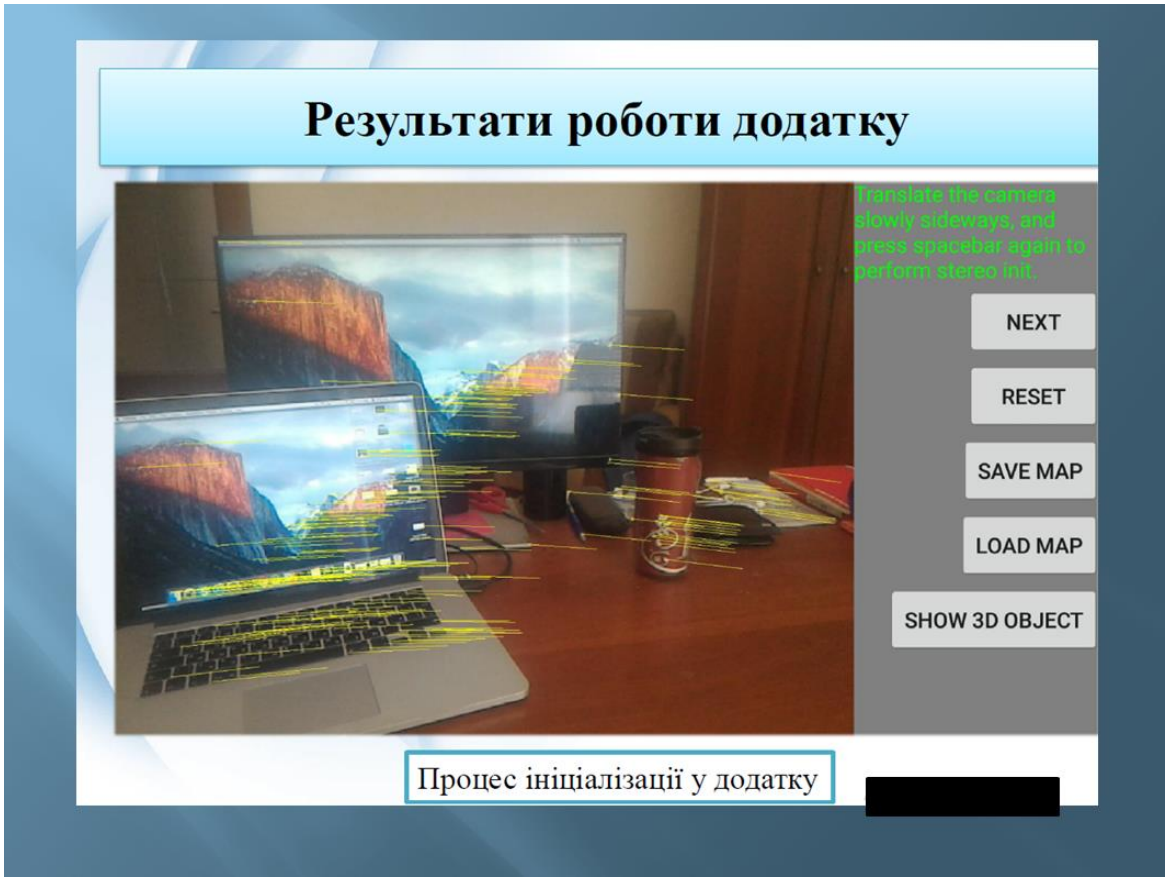


Рисунок В.13- Слайд №13

Рисунок В.14- Слайд №14

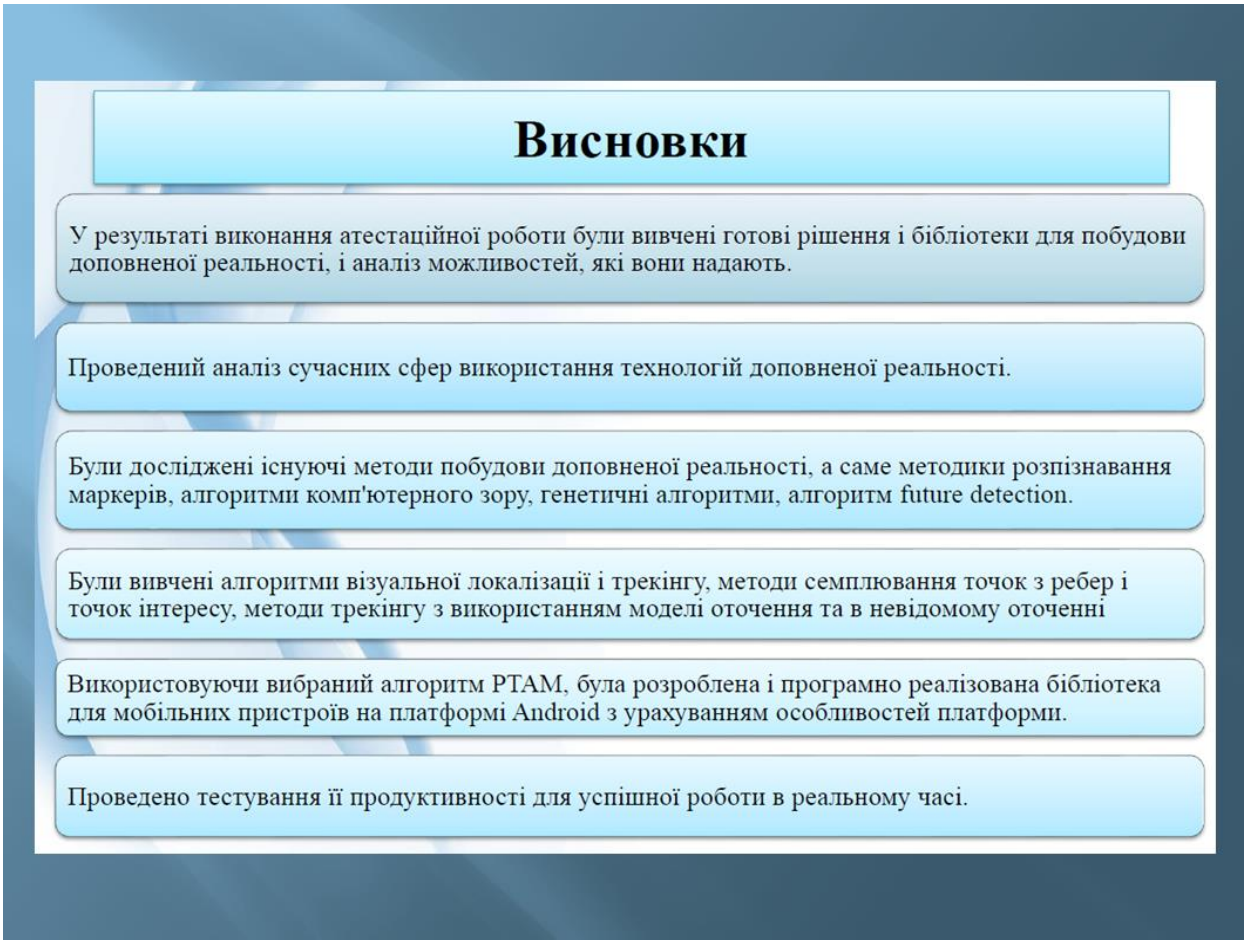


Рисунок В.15- Слайд №15