Міністерство освіти і науки України

черкаський державний технолоГічний університет

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

Кафедра інформаційних технологій проектування

**Пояснювальна записка**

до кваліфікаційної роботи

|  |
| --- |
| магістра |
| (освітньо-кваліфікаційний рівень) |

|  |
| --- |
| на тему: «Дослідження технологій SLAM в доповненій реальності» |

|  |  |
| --- | --- |
|  | Виконав: студентка 2 курсу, групи МІТП-1703 |
|  | Спеціальності |
|  | 122 «Комп’ютерні науки» »  (шифр і назва спеціальності) |
|  | Освітньої програми |
|  | Інформаційні технології проектування , |

|  |  |  |
| --- | --- | --- |
|  | Нечипоренко О.В. | |
|  |  | (прізвище та ініціали) |
|  | Керівник | Карапетян А.Р. |
|  |  | (прізвище та ініціали) |
|  | Рецензент |  |
|  |  | (прізвище та ініціали) |

Черкаси 2018 року

Зміст

[ВСТУП 1](#_Toc532327393)

[РОЗДІЛ 1 ВСТУП ДО SLAM 3](#_Toc532327394)

[1.1 Загальні відомості про SLAM 3](#_Toc532327395)

[1.2 Історія проблеми SLAM 6](#_Toc532327396)

[1.3 Математична складова SLAM 10](#_Toc532327397)

[1.3.1 ЕМ-алгоритм 11](#_Toc532327398)

[1.3.2 Фільтр Калмана 14](#_Toc532327399)

[1.3.3 Фільтр часток 20](#_Toc532327400)

[1.3.4 Поширення сталого інтервалу 21](#_Toc532327401)

[1.3.5 Регулювання зв’язків 22](#_Toc532327402)

[Висновки до розділу 1 24](#_Toc532327403)

[РОЗДІЛ 2 ОГЛЯД РЕАЛІЗАЦІЙ АЛГОРИТМУ SLAM В ДОПОВНЕНІЙ РЕАЛЬНОСТІ 26](#_Toc532327404)

[2.1 ARKit 26](#_Toc532327405)

[2.2 ARCore 34](#_Toc532327406)

[2.3 Wikitude 37](#_Toc532327407)

[2.4 Kudan 38](#_Toc532327408)

[2.5 Facebook AR 39](#_Toc532327409)

[2.6 Vuforia 7 41](#_Toc532327410)

[2.7 AR Foundation 43](#_Toc532327411)

[2.8 Tango 47](#_Toc532327412)

[Висновки до розділу 2 47](#_Toc532327413)

[РОЗДІЛ 3 ПРОБЛЕМИ ДОПОВНЕНОЇ РЕАЛЬНОСТІ ТА ПРОПОЗИЦІЇ ЇХ ВИРІШЕННЯ 49](#_Toc532327414)

[3.1 Проблеми в доповненій реальності 49](#_Toc532327415)

# ВСТУП

Доповнена реальність на перший погляд може здатися не такою захопливою, як віртуальна реальність, проте може внести неабияку користь в повсякденне життя. Вона має в собі величезний потенціал, оскільки переносить елементи з віртуального світу в реальний, доповнюючи речі, які ми здатні бачити та чути, віртуальними елементами що є нулями та одиницями.

Доповнена реальність (в перекладі з англійської augmented reality або AR) – це доповнення фізичного світу за допомогою цифрових даних, яке забезпечується комп'ютерними пристроями (смартфонами, планшетами та окулярами AR) в режимі реального часу. Простіше кажучи, на відміну від віртуальної реальності (Virtual Reality – VR), яка вимагає повного занурення у віртуальне середовище, AR використовує середовище навколо нас та просто накладає поверх нього певну частинку віртуальної інформації, наприклад 3D моделі.

Можливості використання AR технологій практично безмежні. З кожним днем вони змінюють способи нашого спілкування, споживання інформації та ведення бізнесу. Доповнена реальність застосовується в освіті, медицині, авіації, маркетингу, туризмі, дизайні, шопінгу та іграх.[19]

Існують додатки доповненої реальності що використовують маркери – фізичні зрображення, проскановуючи які додаток використовує їх позицію для відображення контенту. Проте справді цікавими є додатки, що використовують безмаркерний підхід – вони аналізують простір навколо смартфону та створюють його карту, розпізнають площини, з якими можна взаємодіяти. Для реалізації такого функціоналу використовується SLAM – одночасну локалізацію та картографування.

Метою магістерської роботи є ознайомитися з проблемою SLAM, з її рішеннями для доповненої реальності. Необхідно проаналізувати вже існуючі реалізації SLAM, їх особливості, переваги та недоліки. На основі проведених досліджень необхідно розробити план вдосконалення існуючої реалізації, запропонувати покращення, проаналізувати можливі ризики їх впровадження та ймовірність отримання позитивного результату після їх розробки.

Задачами магістерської роботи є ознайомлення з проблемою SLAM, розгляд алгоритмів для її рішення, аналіз реалізацій SLAM для доповненої реальності та створення програм з їх допомогою, систематизування наявних проблем, формування пропозиції вирішення описаних проблем.

# РОЗДІЛ 1 ВСТУП ДО SLAM

## 1.1 Загальні відомості про SLAM

SLAM (simultaneous localization and mapping – одночасна локалізація і картографування) – це алгоритмічні обчислювальні задачі побудови і оновлення карти невідомого оточення з одночасним відстежуванням місцеположення об’єкту, що виконує сканування, рухаючись в просторі. SLAM використовується при розв'язуванні задач навігації і картографії у робототехніці. Існує декілька алгоритмів, що реалізують SLAM, щонайменше приблизно, в кінцевий час для певних умов. До популярних методів розв'язування належать фільтр часток, розширений фільтр Калмана і GraphSLAM.[1]

Алгоритми SLAM обмежуються наявними ресурсами, таким чином не можуть бути абсолютно досконалими, бо досягають оперативної доступності. Опубліковані методи і підходи реалізовані в безпілотних автомобілях, безпілотних літаючих засобах, автономних підводних апаратах, планетоходах, згодом імплементувалися в побутових роботах і навіть всередині людського тіла.[1] Також SLAM використовується у віртуальній реальності для того, щоб користувач міг взаємодіяти з віртуальними об’єктами без використання додаткових пристроїв, і в доповненій реальності для розміщення об’єктів у відповідності з реальним простором, особливо у випадках коли користувач рухається (приклад на рис. 1.1). [2]



Рисунок 1.1 Приклад простого AR додатку [2]

Існує велика кількість різних підходів до рішення SLAM, адже для кожної задачі і імплементації створюється щось нове:

* EKF SLAM;
* FastSLAM 1.0;
* FastSLAM 2.0;
* L-SLAM;
* Graph SLAM;
* DP-SLAM;
* Parallel Tracking and Maping (PTAM);
* Occupancy grid SLAM;
* LSD-SLAM;
* S-PTAM;
* ORB-SLAM;
* ORB-SLAM2;
* MonoSLAM;
* Visual Slam;
* CoSLAM;
* SeqSLAM;
* Incremental Smoothing and Maping (iSAM);
* Topological SLAM та багато інших.

З боку віртуальної, доповненої та змішаної реальності найцікавішим є Visual SLAM. На сьогоднішній день Visual SLAM є найбільш придатним для сканування невідомих середовищ, кімнат, просторів, і 3D моделі або реальні об’єкти є основними джерелами інформації що отримується за допомогою камери, яка є найбільш важливою для доповненої реальності, проте наведені далі принципи можуть бути використані і з іншими джерелами даних.

Рендеринг

Внутрішні

сенсори

Позиції

Карта

Точки

Камера

Видобуток

точок

Розповсюдження

Оновлення

Картографування

Рисунок 1.2 Спрощена схема роботи SLAM [2]

На рис. 1.2 показана спрощена схема роботи SLAM, яку можна розшифрувати наступним чином:

1. Внутрішні сенсори складаються з гіроскопу чи інших сучасних сенсорів для виміру кутової швидкості та акселерометри для вимірювання прискорення на трьох осях та рухи користувача. Ці пристрої називаються Inertial Measurement Unit ( IMU).
2. Основною задачею розповсюджуючого підрозділу є інтегрування IMU даних і вираховування позиції. Проте, через можливу неточність IMU даних, не можна повністю опиратися лише на розповсюджуючі дані.
3. Для того, щоб вирішити описану в попередньому пункті проблему, використовується камера. Знімки навколишнього середовища виконуються з постійною швидкістю, зазвичай це 60 FPS. Сучасні камери також мають датчики глибини.
4. Далі ці кадри обробляються і з них вибирають точки, що мають важливі ознаки для розуміння їх позиції.
5. Ці точки передаються картографічному агенту, що доповнює ними карту середовища.
6. Також, отримані точки передаються в оновлюючий підрозділ, який аналізує наявність даної точки на вже складеній карті і завдяки цим даним може скласти відомості про розташування користувача.
7. Використовуючи отриману нову позицію користувача, оновлюючий підрозділ корегує дані, отримані від розповсюджуючого підрозділу. [2]

Необхідність отримати одночасно як позицію камери в реальному просторі так і карту цього простору, коли вони обидва є невідомими, відрізняє проблему SLAM від інших задач. на приклад, розпізнавання на основі маркера (Vuforia до 7 версії) не є SLAM, адже маркер у формі зображення відомий заздалегідь. 3D реконструкція з закріпленими камерами також не є SLAM, з причини того, що позиції камер відомо, хоча сама карта місцевості створюється в процесі роботи програми.

SLAM це проблема типу “Курка чи яйце”: для створення карти необхідно отримувати точну позицію, а для точної позиції необхідно мати гарну карту. SLAM чимось нагадує людину, що шукає свій шлях в незнайомому середовищі. Спочатку, людина дивиться навколо і шукає знайомі їй маркери чи знаки. Якщо людина не розпізнає жодного орієнтиру, можна вважати що вона загубилась. Проте, чим довше людина буде вдивлятися в середовище що її оточує, тим більше орієнтирів вона упізнає, і тоді може почати будувати уявну карту, того місця. [2]

## 1.2 Історія проблеми SLAM

Походженням імовірнісної проблеми SLAM вважається конференція IEEE Robotics and Automation 1986 р., яка проходила в Сан-Франциско, штат Каліфорнія. Це був період, коли імовірнісні методи лише починали вводитись як в робототехніку, так і в штучний інтелект. Ряд дослідників розглядали застосування теоретико-оціночних методів для картографування та проблем локалізації. До них належали Пітер Черезан, Джим Кроулі та Х'ю Дюррант-Вайт. Протягом усієї конференції відбувалось безліч дискусій про послідовне відображення, однією з яких була розмова між Раджа Шатілою, Олівером Фаугера та Рендалом Смітом. Результатом цих бесід стало визнання того, що послідовне імовірнісне відображення є фундаментальною проблемою робототехніки з вагомими питаннями концептуалізації та обчислення, які потребують вирішення.

Протягом наступних кількох років було підготовлено низку ключових документів. Робота Сміта та Чізмана «Про представлення просторової невизначеності» та Дюррант-Вайта «Незвична геометрія в робототехніці» встановили статистичну основу для опису відносин між орієнтирами (лендмарками, маяками) та маніпулюванням геометричною невизначеністю. Основною метою цієї роботи було показати, що повинна бути висока ступінь кореляції між оцінками розташування різних орієнтирів на карті, і що ці співвідношення будуть зростати з подальшими спостереженнями.

В цей же час Аяче та Фауґерс почали працювати над візуальною навігацією в науковій праці «Будівництво, розпізнавання та об’єднання візуальних карт з шумами», Кроулі в роботі «Моделювання світу та оцінка позиції мобільного робота використовуючи ультразвуковий діапазон» і Шатіла та Лаумонд в «Посилання на позиції та постійне моделювання світу для мобільних роботів» описали гідроакустичний спосіб навігації мобільних роботів, використовуючи алгоритми фільтра типу Калмана.

Ці два напрямки досліджень мали багато спільного і об’єднались у визначній статті «Оцінка невизначеності просторових відносини в робототехніці» написаній Смітом. Цей документ показав, що коли мобільний робот рухається через невідоме середовище, беручи відносні спостереження за орієнтирами, оцінки цих орієнтирів обов'язково співвідносяться один з одним через загальну помилку у оцінюваному розташуванні машини. Причина цього крилась в тому, що для повного вирішення проблеми одночасної локалізації та картографії потрібний об’єднаний стан, що складається з позиції машини та позицій всіх орієнтирів, які повинні оновлюватися після спостереження кожного орієнтиру. У свою чергу, для таких обрахунків необхідно, щоб оцінювач використовував величезний вектор станів (за кількістю орієнтирів, що зберігаються на карті) з обчисленням масштабування як квадрат числа орієнтирів. Важливо зазначити, що ця робота не розглядала особливості конвергенції карти або її стаціонарну поведінку. Справді, в той час було загальноприйнято, що розрахункові помилки карти не збігаються, і, навпаки, приводять до поведінки випадкового блукання машини з необмеженим зростанням помилок.

Таким чином, враховуючи обчислювальну складність задачі відображення та без знання поведінки конвергенції карти, дослідники зосереджували увагу на серії наближень до послідовної задачі відображення, які дозволили мінімізувати або усунути кореляції між орієнтирами, таким чином зменшуючи повний фільтр до серії відокремлених орієнтирів відносно фільтрів транспортних засобів. Також з цих причин теоретична робота над проблемою одночасної локалізації та картографування була тимчасово припинена, причому робота, що продовжувалася в цій галузі, часто зосереджувалася або на відображенні, або на локалізації, бачачи їх як окремі проблеми.

Концептуальний прорив стався одночасно з усвідомленням того, що проблема одночасної локалізації і картографування, колись сформульована як проблема єдиної оцінки, була насправді конвергентною. Найголовніше, було визнано, що кореляції між орієнтирами, які більшість дослідників намагалися звести до мінімуму, насправді є критичною частиною проблеми, і, навпаки, чим більше ці кореляції зростали, тим краще рішення.

Структура проблеми SLAM, результат конвергенції та монетування абревіатури SLAM вперше були представлені в оглядовому документі з мобільної робототехніки «Визначення місцеположення автоматично керованої машини» на Міжнародному симпозіумі 1995 року з дослідження робототехніки. Основна теорія збіжності та багато початкових результатів були розроблені Ксорбою в роботах «Одночасна локалізація та побудова карт» та «Новий підхід до одночасної локалізації та побудови карт».

Кілька груп, які вже працювали над картографуванням та локалізацією, зокрема, в Массачусетському технологічному інституті, Сарагосі, ACFR в Сіднеї, та інших, почали серйозно працювати над SLAM – в той час цю проблему ще називали CML (concurrent mapping and localization – паралельне відображення та локалізація) – у приміщенні, на відкритому повітрі та підводних середовищах. В цей час робота була зосереджена на підвищенні ефективності обчислень та вирішенні проблем при об'єднанні даних або закритті циклів.

Міжнародний симпозіум з дослідження робототехніки (ISRR'99) 1999 року був зустріччю, де проводилась перша сесія по SLAM, і де відбулася збіжність між методами SLAM на базі фільтра Калмана та методами імовірнісної локалізації та картографування Трюн. У 2000 р. Міжнародна конференція IEEE з робототехніки та автоматизації (ICRA), присвячена SLAM, залучила 15 дослідників для зосередження на таких питаннях, як алгоритмічна складність, об'єднання даних та проблеми реалізації. Наступний семінар SLAM на ICRA 2002 привернув увагу 150 дослідників з широким спектром інтересів та програм. [3]

На початку 2000-х застосування візуальних датчиків стало важливим аспектом дослідження SLAM, частково тому, що зображення забезпечує цінне джерело інформації про структуру навколишнього середовища (що містить більше інформації, ніж, наприклад, гнучкий звуковий сигнал). Велика кількість досліджень візуального SLAM стала використовувати стерео камери або камери в сукупності з іншими датчиками (такими як акселерометри або GPS), але приблизно з 2001 р. в ряді робіт було показано, як SLAM може бути успішно реалізований, використовуючи лише одну камеру (цей підхід відомий як монокулярний SLAM), як приклад можна привести роботу «SLAM з однією камерою» Ендрю Дейвісона в Оксфордському університеті.

Це дослідження мало вирішальне значення в росповсюдженні SLAM для більш широкої сфери використання, адже такі пристрої, що оснащені тільки однією камерою – вебкамери і мобільні телефони – є значно поширенішими і доступнішими ніж спеціальне надчутливе обладнання. Більш недавні роботи показали, як монокулярний SLAM може бути використаний для створення масштабних карт, як можна автоматично підсилити карти з використанням 3D-структур, а також відновити надзвичайно деталізовані форми в режимі реального часу. [4]

## 1.3 Математична складова SLAM

Постановка задачі SLAM: Дана послідовність даних спостереження сенсору  за дискретні проміжки часу *t*, задачею SLAM є розрахувати і визначити розташування агента  і мапу оточення . Всі величини зазвичай ймовірнісні, тому необхідно обчислити рівняння 1.1:

|  |  |
| --- | --- |
|  | (1.1) |

Застосування правила Баєса дає основу для послідовного оновлення апостеріорного розташування 1.2, при даній мапі і функції переходу :

|  |  |
| --- | --- |
|  | (1.2) |

Аналогічно мапа може оновлюватися послідовно шляхом представленим на 1.3:

|  |  |
| --- | --- |
|  | (1.3) |

Як і для більшості задач наближення, рішення можна знайти при наближенні двох змінних до локального оптимального рішення шляхом почергового оновлення двох рівнянь у формі ЕМ-алгоритму.

До статистичних методів, що використовуються для задач апроксимації наведених вище, належать: фільтр Калмана, фільтр часток (який є методом Монте-Карло) і узгоджене сканування діапазонних даних. Вони дозволяють визначити оцінку функції апостеріорної ймовірності для позиції камери і параметрів мапи. Техніки оцінювання приналежності до множини в основному засновуються на поширенні сталого інтервалу. [12] Вони забезпечують множину, яка містить позицію камери і множину апроксимації мапи. Регулювання зв’язків (bundle adjustment) є наступною популярною технікою у SLAM, яка використовує дані зображень поєднані з оцінками розташування камери і орієнтирів на місцевості, тим самим підвищуючи точність мапи. Вона використовується в комерційних SLAM системах, таких як Проект Tango компанії Google.

### 1.3.1 ЕМ-алгоритм

EM-алгоритм (англ. Expectation-maximization algorithm) – алгоритм, що використовується в математичній статистиці для знаходження оцінок максимальної схожості параметрів ймовірних моделей, у випадку, коли модель залежить від деяких прихованих змінних. Кожна ітерація алгоритму складається з двох кроків. На E-кроці (expectation) вираховується очікуване значення функції правдоподібності, при цьому приховані змінні розглядаються як спостережувані. На M-кроці (maximization) вираховується оцінка максимальної схожості, таким чином збільшується очікувана схожість, вирахувана на E-кроці. Потім це значення використовується для E-кроку на наступній ітерації. Алгоритм виконується до збіжності. Часто EM-алгоритм використовують для розділення суміші функції Гаусса.

Опишемо алгоритм. Нехай  – деяке з значень спостережуваних змінних, а  – прихованні змінні. Разом  і  утворюють повний набір даних. Взагалі,  може бути деякою підказкою, яка полегшує рішення проблеми у випадку, якщо вона відома. Наприклад, якщо є суміш розподілів, функція правдоподібності легко виражається через параметри відокремлених розподілів суміші.

Покладемо  – густину імовірності (в безперервному випадку) або функція ймовірностей (в дискретному випадку) повного набору даних з параметрами . Цю функцію можна розуміти як правдоподібність всієї моделі, якщо розглядати її як функцію параметрів . Зауважимо, що умовний розподіл прихованої компоненти при деякому спостереженні та фіксованому наборі параметрів може бути вираженим формулою 1.4:

|  |  |
| --- | --- |
| , | (1.4) |

використовуючи розширену формулу Байеса і формулу повної ймовірності. Таким чином, нам необхідно знати тільки розподіл спостережуваної компоненти при фіксованій прихованій  і ймовірності прихованих даних .

EM-алгоритм ітеративно покращує початкову оцінку , обчислюючи нові значення оцінок , , і так далі. На кожному кроці перехід до  від  виконується як показано на 1.5:

|  |  |
| --- | --- |
| , | (1.5) |

де – математичне сподівання логарифма правдоподібності.

Іншими словами, ми не можемо відразу обчислити точну правдоподібність, але за відомими даними  ми можемо знайти апостеріорну оцінку ймовірностей для різних значень прихованих змінних. Для кожного набору значень  і параметрів  ми можемо обчислити математичне сподівання функції правдоподібності з даного набору . Воно залежить від попереднього значення , бо це значення впливає на ймовірності прихованих змінних .

 обчислюється за формулою 1.6:

|  |  |
| --- | --- |
| , | (1.6) |

тобто умовне математичне сподівання  при умові .

Іншими словами,  – це значення, максимізуючи (M) умовне математичне сподівання (E) логарифма правдоподібності при даних значеннях спостережуваних змінних і попередньому значенні параметрів. У безперервному випадку значення  вираховується за рівнянням 1.7 [13]:

|  |  |
| --- | --- |
|  | (1.7) |

Існує альтернативний опис EM-алгоритму. За певних обставин зручно розглядати EM-алгоритм як два кроки максимізації, що чергуються. Розглянемо функцію 1.8:

|  |  |
| --- | --- |
| , | (1.8) |

де  – розподіл ймовірностей неспостережуваних змінних ;  – умовний розподіл неспостережуваних змінних при фіксованих спостережуваних  і параметрах розподілення ймовірностей неспостережуваних змінних ;  – ентропія і  – відстань Кульбака-Лейблера. [13]

Тоді кроки EM-алгоритму можна показати як:

* E(Expectation) крок: Вибираємо , щоб максимізувати  (формула 1.9):

|  |  |
| --- | --- |
|  | (1.9) |

* M(Maximization) крок: Вибираємо , щоб максимізувати  (формула 1.10):

|  |  |
| --- | --- |
|  | (1.10) |

### 1.3.2 Фільтр Калмана

Фільтр Калмана (англ. Kalman filter), відомий також як лінійно-квадратичне оцінювання (англ. linear quadratic estimation, LQE), – це алгоритм, що використовує послідовності вимірювань протягом часу, які містять шум (випадкові відхилення) та інші неточності, й видає оцінки невідомих змінних, що є потенційно точнішими за базовані на самих лише вимірюваннях. Формальніше, фільтр Калмана працює рекурсивно на потоках зашумлених вхідних даних, і видає статистично оптимальну оцінку базового стану системи. Фільтр названо на честь Рудольфа Калмана, одного з головних розробників його теорії.

Фільтр Калмана має численні застосування у технології. Поширеним є застосування для наведення, навігації та керування транспортними засобами, особливо літаками та космічними апаратами. Крім того, фільтр Калмана є широко застосовуваною концепцією в аналізі часових рядів, що використовується у таких галузях як обробка сигналів та економетрія. Фільтри Калмана також є однією з головних тем у плануванні та керуванні роботизованим рухом, й іноді включаються до оптимізації траєкторії.

Цей алгоритм працює як двокроковий процес. На кроці передбачення фільтр Калмана видає оцінки змінних поточного стану, разом із їхніми невизначеностями. Щойно отримано спостереження виходу наступного вимірювання (неодмінно якоюсь мірою спотворене відхиленням, включно з випадковим шумом), ці оцінки уточнюються з використанням середнього зваженого, в якому більше ваги надається оцінкам з вищою визначеністю. Через рекурсивну природу алгоритму він може працювати в реальному часі, використовуючи лише наявні вхідні вимірювання, попередньо обчислений стан та його матрицю невизначеності; ніякої додаткової інформації не потрібно.

Поширеним є хибне уявлення, що фільтр Калмана передбачає, що всі вектори помилок та вимірювання мають нормальний розподіл. Оригінальна робота Калмана виводила цей фільтр з використанням теорії прямокутної проекції, щоби показати, що коваріація мінімізується, і цей результат не вимагає жодного припущення, наприклад, що похибки мають нормальний розподіл. Він потім показав, що цей фільтр видає точну оцінку умовної ймовірності в особливому випадку, коли похибки мають нормальний розподіл.

Також було розроблено розширення та узагальнення цього методу, такі як розширений фільтр Калмана (англ. Extended Kalman filter, EKF) та беззапаховий фільтр Калмана (англ. Unscented Kalman filter, UKF), що працюють на нелінійних системах. Базовою моделлю є баєсова модель, подібна до прихованої марковської моделі, але в якій простір станів латентних змінних є безперервним, і де всі латентні та спостережувані змінні мають нормальні розподіли.

При використанні фільтра Калмана для отримання оцінок вектора стану процесу по серії зашумленних вимірювань необхідно представити модель даного процесу відповідно до структури фільтра – у вигляді матричного рівняння певного типу. Для кожного такту  роботи фільтра необхідно відповідно до наведеного нижче опису визначити матриці: еволюції процесу ; матрицю спостережень ; коваріаційну матрицю процесу ; коваріаційну матрицю шуму вимірювань ; при наявності керуючих впливів матрицю їх коефіцієнтів .

Модель системи/процесу має на увазі, що справжній стан в момент  виходить з істинного стану в момент  відповідно до рівняння 1.11:

|  |  |
| --- | --- |
| , | (1.11) |

де  – матриця еволюції процесу/системи, яка впливає на вектор стану в момент ;  – матриця управління, яка прикладається до вектору керуючих впливів ;  – нормальний випадковий процес з нульовим математичним очікуванням і коваріаційною матрицею , який описує випадковий характер еволюції системи/процесу(рівняня 1.12).

|  |  |
| --- | --- |
| , | (1.12) |

У момент  проводиться спостереження (вимір)  істинного вектора стану , які пов'язані між собою рівнянням 1.13:

|  |  |
| --- | --- |
| , | (1.13) |

де  – матриця вимірювань, що зв'язує істинний вектор стану і вектор проведених вимірювань,  – білий гаусівський шум вимірювань з нульовим математичним очікуванням і коваріаційною матрицею  (1.14):

|  |  |
| --- | --- |
| , | (1.14) |

Початковий стан і вектори випадкових процесів на кожному такті  вважаються незалежними.

Багато реальних динамічних систем не можна точно описати даною моделлю. На практиці не врахована в моделі динаміка може серйозно зіпсувати робочі характеристики фільтра, особливо при роботі з невідомим стохастическим сигналом на вході. Більш того, неврахована в моделі динаміка може зробити фільтр нестійким. З іншого боку, незалежний білий шум в якості сигналу не буде приводити до розбіжності алгоритму. Завдання відділення шумів вимірювань від неврахованої в моделі динаміки складна, вирішується вона за допомогою теорії робастних систем управління.

Для обчислення оцінки стану системи на поточний такт роботи фільтру Калмана необхідні оцінка стану (у вигляді оцінки стану системи і оцінки похибки визначення цього стану) на попередньому такті роботи і вимірювання на поточному такті.

Далі під записом  будемо розуміти оцінку істинного вектора  в момент  з урахуванням вимірів з моменту початку роботи і по момент  включно.

Стан фільтра задається двома змінними:

*  – апостеріорна оцінка стану об'єкта в момент , отримана за результатами спостережень аж до моменту  включно;
*  – апостеріорна коваріаційнв матриця помилок, що задає оцінку точності отриманої оцінки вектора стану і включає в себе оцінку дисперсій похибки обчисленого стану і коваріації, що показують виявлені взаємозв'язку між параметрами стану системи.

Ітерації фільтра Калмана діляться на дві фази: екстраполяція і корекція. Під час екстраполяції фільтр отримує попередню оцінку стану системи  на поточний крок за підсумковою оцінкою стану з попереднього кроку (або попередню оцінку на наступний такт за підсумковою оцінкою поточного кроку, в залежності від інтерпретації). Цю попередню оцінку також називають апріорною оцінкою стану, так як для її отримання не використовуються спостереження відповідного кроку. У фазі корекції завжди апріорна екстраполяція доповнюється відповідними поточними вимірами для корекції оцінки. Скоригована оцінка також називається апостеріорною оцінкою стану, або просто оцінкою вектора стану. Зазвичай ці дві фази чергуються: екстраполяція проводиться за результатами корекції до наступного спостереження, а корекція проводиться спільно з доступними на наступному кроці спостереженнями, і т. д. Однак можливий й інший розвиток подій, якщо з деякої причини спостереження виявилося недоступним, то етап корекції може бути пропущений і виконана екстраполяція по нескорректованій оцінці (апріорній екстраполяції). Аналогічно, якщо незалежні вимірювання доступні лише в окремі такти роботи, то все одно можливі корекції (зазвичай з використанням іншої матриці спостережень ).

Далі розглянемо роботу класичного оптимального фільтра Калмана. На етапі екстраполяції (прогнозування) вектора стан системи за оцінкою вектора стану і застосованого вектору управління з кроку  на крок  – вормула 1.15.

|  |  |
| --- | --- |
|  | (1.15) |

Коваріаційна матриця для екстрапольованого вектора стану має вигляд 1.16:

|  |  |
| --- | --- |
|  | (1.16) |

На етапі корекції відхилення отриманого на кроці  спостереження від спостереження, очікуваного при виробленій екстраполяції виражається рівнянням 1.17:

|  |  |
| --- | --- |
|  | (1.17) |

Коваріаційна матриця для вектора відхилення (вектора помилки) має вигляд 1.18:

|  |  |
| --- | --- |
|  | (1.18) |

Оптимальна по Калману матриця коефіцієнтів посилення 1.19 формується на підставі коваріаційних матриць, наявної екстраполяції вектора стану та отриманих вимірювань (за допомогою коваріаційної матриці вектора відхилення):

|  |  |
| --- | --- |
|  | (1.19) |

Коррекція раніше отриманої екстраполяції вектора стану – отримання оцінки вектора стану системи у вигляді 1.20:

|  |  |
| --- | --- |
|  | (1.20) |

Розрахунок коваріаційної матриці оцінки вектора стану системи відбувається за рівнянням 1.21:

|  |  |
| --- | --- |
|  | (1.21) |

Вираз для коваріаційної матриці оцінки вектора стану системи справедливо тільки при використанні наведеного оптимального вектора коефіцієнтів. У загальному випадку цей вираз має більш складний вигляд.

Незважаючи на те, що фільтр Калмана застосовується тільки до лінійних систем, і не підходить для корекції положення автономного робота, він цілком підходить і ефективний для оцінок положення нерухомих просторових точок.

Розширений фільтр Калмана (Extended Kalman Filter – EKF) є нелінійною версією фільтра Калмана. Спочатку EKF міг вважатися стандартом в теорії нелінійного обчислення положення, навігаційних систем і GPS. Однак з появою беззапахового фільтра Калмана (unscented Kalman filter – UKF), становище змінилося. У розширеному фільтрі Калмана стан системи і спостереження не зобов'язані бути лінійними функціями стану, а повинні бути диференційованими (1.22 та 1.23).

|  |  |
| --- | --- |
|  | (1.22) |
|  | (1.23) |

Основна ідея, що застосовується в розширеному фільтрі Калмана, полягає у наближенні функцій стану і спостереження з використанням їх перших похідних.

Розширений фільтр Калмана в загальному випадку не є оптимальним, на відміну від лінійного аналога. Також, якщо початкове обчислення стану системи помилкове, або процес змодельований некоректно, результати можуть швидко розходитися через лінеаризації. Ще однією проблемою є те, що обчислювальна матриця коваріації схильна до недооцінки реальної коваріації і, тому, ризики стають мало пов'язаними з реальною ситуацією без додавання «стабілізуючого шуму».[14]

Якщо функція перерахунку стану системи і функція спостережень сильно нелінійні, розширений фільтр Калмана дає погані результати. Це пов'язано з обчисленням коваріації за допомогою лінеаризації нелінійної моделі. Беззапаховий фільтр Калмана використовує детерміновану вибірку, відому як беззапахове перетворення (the unscented transform) для вибору мінімального набору точок (так званих сигма-точок) навколо середнього значення. Сигма-точки пропускаються через нелінійні функції, за якими потім відновлюється середнє значення і коваріація.

В результаті отримують фільтр, який більш точно фіксує середнє значення і коваріацію. Крім цього, подібний підхід знімає вимогу на обчислення Якобіанів, що для складних функцій може становити серйозну проблему.

### 1.3.3 Фільтр часток

Даний фільтр також відомий як фільтр Монте-Карло. Кожна частка характеризується як можливий стан робота в даний момент часу. Являючись математичними моделями, ці фільтри описують розподіл ймовірності у вигляді дискретного набору частинок в просторі станів.

Першим кроком ітерації локалізації фільтра часток є генерація нового часткового розподілу для даної моделі руху і застосованого управління. Наступним кроком розраховується вага кожного значення. Часткам, для яких прогнозовані значення відповідають виміряним, дається більша вага. І останній етап це повторна вибірка часток. Вона ґрунтується на вагах часток з попереднього розподілу, які беруться випадковим чином, створюючи новий розподіл.

Для побудови припущення про поточний стан робота використовується часткова фільтрація Рао-Блеквелла, що використовує орієнтири. Кожна оцінка орієнтира представляється 2x2 розширеним фільтром Калмана.

Головною проблемою цього методу є логарифмічна складність фільтра, яка сильно залежить від кількості часток. При їх малій кількості, протягом тривалого часу роботи виникають помилки, тим самим спотворюючи отриманий образ простору. Однак чим більше число часток фільтр буде мати у своєму розпорядженні, тим більше часу буде потрібно для обчислень.

Фільтр часток реалізується в DP-SLAM (Distributed Particle – Simultaneous Localization And Mapping). Цей метод ґрунтується на використанні технології відображення розподілених часток – distributed particle mapping (DP-Mapping). Метод був розроблений студентами Університета Дьюка Остіном Елізаром та Рональдом Парром. Цей метод зручно використовувати коли в досліджуваному просторі немає можливості знайти яскраво виражених орієнтирів.

Карта середовища представляється у вигляді стіки з заповненими клітками, в яких є перешкоди. Зберігати таку карту зручно у вигляді масива, де елементи, що відображають позицію перешкоди мають значення 1, а всі інші – 0. Візуально карта приміщення створена DP-SLAM показана на рис. 1.3. [15]



Рисунок 1.3 Карта DP-SLAM з 9000 часток [15]

### 1.3.4 Поширення сталого інтервалу

В обчислювальній математиці, поширення сталого інтервалу є проблемою договірних інтервальних доменів зі змінними R без видалення значень, які узгоджуються з набором обмежень (тобто рівнянь або нерівностей). Це може бути використано для поширення невизначеностей у ситуаціях, коли помилки представлені інтервалами. Поширення інтервалів розглядає задачу оцінювання як проблему задоволення обмежень.

Підрядник, пов'язаний з рівнянням, що включає змінні  є оператором, який укладає інтервали  (які повинні містити ) не видаляючи при цьому значень для змінних, які узгоджуються з рівнянням.

Підрядник вважається атомним, якщо він не будується як частина інших підрядників. Основна теорія, яка використовується для побудови атомних підрядників, базується на інтервальних аналізах.

Для більш складних обмежень слід використовувати розподілення на атомні обмеження (тобто обмеження, для яких існує атомний підрядник). Розглянемо, наприклад, обмеження 1.24.

|  |  |
| --- | --- |
|  | (1.24) |

Обмеження 1.24 можна розбити на систему рівнянь 1.25 з відповідними інтервалами.

|  |  |
| --- | --- |
|  | (1.25) |

Принцип розподілу інтервалів полягає в тому, щоб викликати всіх доступних атомних підрядників до тих пір, поки не буде більше скорочень. Внаслідок теореми Кнастер-Тарського процедура завжди сходиться до інтервалів, які містять всі допустимі значення для змінних. Розповсюдження інтервалів швидко збігається до результату і може вирішити проблеми, пов'язані з кількома сотнями змінних.[17]

### 1.3.5 Регулювання зв’язків

В ситуації з набором фотографій, що зображують 3D точки з різних ракурсів, регулювання зв’язків можна визначити як проблему одночасного вдосконалення 3D-координат, що описують геометрію сцени, параметри відносного руху та оптичні характеристики камери, які використовуються для отримання зображень, відповідно до критерію оптимальності, що включає відповідні проекційні зображення всіх точок.

Регулювання зв’язків майже завжди використовується як останній крок кожного алгоритму 3D-реконструкції що ґрунтується на опорних точках простору. Отримання оптимальної реконструкції становить оптимізаційну проблему в 3D структурі та параметрах перегляду (тобто позиція камери і, можливо, внутрішнє калібрування та радіальні спотворення). Якщо помилка зображення є нульове середнє Гауса, то регулювання пучка повинно бути оцінкою максимальної правдоподібності.

Назва алгоритму походить від зв’язків (пучків) світлових променів, що походять з кожної тривимірної опорної точки, і зближуються на оптичному центрі кодної камери, які оптимально налаштовуються щодо структури та параметрів перегляду. Регулювання зв’язків спочатку була задумана в області фотограмметрії протягом 1950-х років, проте все частіше використовувалася в комп'ютерному зорі.

Регулювання зв’язків зводиться до мінімізації похибки відтворення між розташуванням зображень спостережуваних та прогнозованих та точок зображення, що виражається як сума квадратів великої кількості нелінійних функцій з реальними значеннями. Таким чином, мінімізація досягається за допомогою алгоритму нелінійних найменших квадратів. З них алгоритм Левенберга-Марквардта виявився одним з найуспішніших завдяки простоті його впровадження та використанню ефективної стратегії загасання, що дає можливість швидкого сходження з широкого кола первинних припущень. Ітеративно лінеаризуючи функцію, яка буде мінімізована в рамках поточної оцінки, алгоритм Левенберга-Марквардта передбачає розв'язок лінійних систем, представлених нормальними рівняннями. При вирішенні задач мінімізації, що виникають в рамках регулювання зв’язків, нормальним рівнянням є розріджена блокова структура, що зумовлена відсутністю взаємодії між параметрами для різних опорних точок та камер. Це може бути використано для отримання величезних обчислювальних переваг, використовуючи розріджений варіант алгоритму Левенберга-Марквардт, який явно використовує нормальну схему нулів рівнянь, запобігаючи зберіганню та керуванню нульовими елементами.

Регулювання зв’язків –це спільне вдосконалення набору початкових оцінок параметрів камери та структурних ймовірнісних змінних, які найбільш точно передбачають розташування спостережуваних опорних точок у наборі доступних зображень. Більш формально, припустимо, що  3D-точкок видно з точок спостережень і нехай буде проекцією тієї -тої опорної точки на зображенні . Нехай позначає бінарні змінні, які дорівнюють 1, якщо точку  видно у зображенні , інакше – 0. Припустимо також, що кожна камера  параметризується за допомогою вектора  та кожна 3D опорна точка  за допомогою вектора . Регулювання зв’язків мінімізує загальну помилку відтворення по відношенню до всіх 3D опорних точок та параметрів камери, як показано на 1.26

|  |  |
| --- | --- |
|  | (1.26) |

Де  це передбачена проекція точки  на зображенні  та  позначає евклідову відстань між точками зображення, представленими векторами  та . Зрозуміло, що регулюваня зв'язків за визначенням толерантне до відсутності проеціювання зображень і мінімізує фізично значущий критерій. [18]

## Висновки до розділу 1

Одночасна локалізація і картографування (англ. Simultaneous localization and mapping (SLAM)) – це задача побудови та оновлення карти невідомого середовища і одночасного визначення положення робота на цій карті. Задача SLAM – одна з найактуальніших задач сучасної робототехніки, і сьогодні над її вирішенням працює величезна кількість людей, про що свідчить поява безлічі публікацій з найрізноманітнішими підходами до вирішення. Загалом, для побудови карти робиться опис об'єктів поруч з роботом, визначаються координати цих об'єктів, об'єкти наносяться на карту.[16]

SLAM використовується в безпілотних автомобілях, безпілотних літаючих засобах, автоматизованих роботах, в доповненій реальності та віртуальній реальності.

Нові алгоритми SLAM досі потребують активного дослідження і пошуку, і часто обумовлені різними вимогами і припущеннями щодо типів карт, датчиків і моделей. Більшість SLAM систем можна розглядати як комбінації виборів кожного з розглянутих в цьому розділі методів.

# РОЗДІЛ 2 ОГЛЯД РЕАЛІЗАЦІЙ АЛГОРИТМУ SLAM В ДОПОВНЕНІЙ РЕАЛЬНОСТІ

## 2.1 ARKit

ARKit розробляється компанією Apple. Побудований на основі SLAM, спочатку розробки велися німецькою компанією Metaio, яка досить швидко була куплена Apple. ARKit дозволяє відслідковувати переміщення телефону в просторі, визначати кількість і теплоту світла навколо, отримувати інформацію про горизонтальні площини, визначення вертикальних площин, розпізнавання 2D-зображень, розпізнавання 3D-об’єктів, автофокус, відзеркалення на текстурах віртуальних об’єктів фрагментів реальних об’єктів, розпізнавання обличчя, напряму погляду, можливе створення багатокористувацьких мережевих ігор для доповненої реальності.[6]

На сьогоднішній день ARKit являється найпопулярнішим засобом для створення додатків доповненої реальності. Цей фактор зумовлений політикою Apple в регулярних оновленнях програмного забезпечення та підтримки сучасного рівня апаратного забезпечення своїх девайсів. Це дозволяє більшій кількості користувачів мати доступ до таких технологій, як ARKit, що відповідно збільшує кількість замовлень на розробку відповідних програм.

Основною вимогою для будь-якого AR додатку та визначальною функцією ARKit є здатність створювати та відслідковувати відповідність між реальним простором, в якому знаходиться користувач, та віртуальним простором, де можна моделювати візуальний контент. Коли AR додаток показує цей вміст разом із зображенням камери, користувач бачить доповнену реальність: ілюзію того, що віртуальний вміст додатку є частиною реального світу.

У всіх AR реалізаціях, ARKit використовує світову систему та систему координат камери, яка слідує за правилом правої руки: вісь Y вказує вгору, а вісь z (коли відносна) вказує на глядача, вісь x вказує праворуч від глядача.

Конфігурації сеансу можуть змінити походження та орієнтацію системи координат по відношенню до реального світу завдяки змінній worldAlignment, що є частиною класу ARConfiguration з бібліотеки ARKit. Кожен якор у сесії AR визначає свою власну локальну систему координат, також слідуючи правилам правої руки та вісі z до користувача ; наприклад, клас ARFaceAnchor визначає систему координат для визначення рис обличчя.

Для створення відповідності між реальним і віртуальним просторами, ARKit використовує техніку, названу візуально-інерційною одометрією. Цей процес поєднує в собі інформацію з апаратного забезпечення пристрою iOS з аналізом зору комп'ютера на сцені, видимій на камеру пристрою. ARKit визнає помітні особливості зображення, відслідковує відмінності в положеннях цих функцій у відеокадрах і порівнює цю інформацію з даними про рух даних. Результатом є високоточна модель позиції та руху пристрою.

Відстеження простору також аналізує та розуміє вміст сцени. Використовуючи методи хіт-тестування класу ARHitTestResult з бібліотеки ARKit можна знайти реальні поверхні, що відповідають точці зображення камери. Якщо увімкнути параметр planeDetection у налаштуваннях сеансу, ARKit виявляє плоскі поверхні на зображеннях, отриманих з камери, та повідомляє про їх розташування та розміри. Результати хіт-тесту або виявлені площини можна використовувати для розміщення або взаємодії з віртуальним вмістом на сцені.

Гарною практикою є надання часу для виявлення площини, щоб отримати чіткі результати, і після отримання потрібних результатів вимикати процес виявлення площини. Результати виявлення площини залежать від часу – коли площина вперше виявляється, її положення та розміри можуть бути неточними. Поки площина залишається на місці, ARKit удосконалює свою оцінку її положення та розміру. Коли велика плоска поверхня знаходиться на місці, ARKit може продовжувати змінювати позицію, розміри та трансформувати якір площини, коли вона вже використовується для розміщення віртуального вмісту. [20]

Додатки доповненої реальності з ARKit можна писати в SceneKit, частині Xcode – середовищі для розробки нативних додатків під iOS та MacOS на мовах Swift і ObjectiveC, або в Unity – кросплатформенному середовищі для розробки додатків на мові C# як на iOS та MacOS, так і на Android, Windows, Linux, PS і т. д. [21] Загалом, Unity має декілька переваг над Xcode’овським SceneKit: кросплатформність, легкість у використанні в таких аспектах як імпорт 3D моделей, програмування, наявність великою кількості навчальних матеріалів, open source прикладів та документації. З цих причин для подальшого ознайомлення та створення власних додатків буде використано Unity.

Співпрацею Unity та Apple було оперативно створено sdk (від англ. Software Development Kit – набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми за визначеною технологією або для певної платформи) для роботи з ARKit. Плагін ARKit для Unity дає розробникам доступ до всіх ARKit функцій, таких як відстеження рухів, рендеринг трансльованого відеоряду, пошук площин, перевірка попадання, оцінка природного освітлення, необроблені дані хмар точок і так далі. Окрім цього, зручні компоненти Unity полегшують розробку нових AR-додатків та інтеграцію AR-функцій в уже існуючі проекти на Unity.

Для розробки додатку необхідно мати встановленою Unity, в проекті завантаженим Unity ARKit Plugin, та завантажити 3D модель на власний вибір. Для першого прикладу буде використано модель оливкового дерева, і відповідно до неї додаток називатиметься «Olive tree».

В тестових сценах, які йдуть в комплекті з Unity ARKit Plugin, вже є налаштовані приклади, які можна використовувати. З усіх скриптів та об’єктів на тестовій сцені основними для нас являються HitParen – об’єкт, під яким буде знаходитись обрана 3D модель, та класи UnityARHitTestExample і PointCloudParticleExample. Лістинг цих класів наведено в додатку А.

UnityARHitTestExample відопвідає за отримання івенту про реєстрування взаємодії користувача з площиною, що була розпізнана на основі даних з камери. Такою взаємодією є дотик на сенсорному екрані. Від двовимірної точки дотику на екрані смартфона проводиться промінь. Якщо цей промінь перетинає площину, що була створена на основі реальних даних, то викликається метод з класу UnityARHitTestExample і відбуваються дії, запрограмовані розробником – зазвичай це поява моделі на місці зіткнення променя та площини. Також в цьому скріпті можна прописати, щоб модель не можна було перемістити знову після першого ставлення.

PointCloudParticleExample відображує знаходження опорних точок – на кожній точці, яку розпізнає алгоритм ARKit з’являється реальна точка, зазвичай жовта. В процесі сканування оточуючого середовища ці точки показують, де і як гарно розпізнається поверхня. В наведеному прикладі, після поставлення моделі дерева візуалізація опорних точок вимикається, щоб не зменшувати реалістичність отриманої картинки.

До вже існуючих в тестовій сцені класів додаємо один власний – клас для контролю розміру і повороту дерева ModelController. Отриманий результат можна спостерігати на рис. 2.1-2.3.

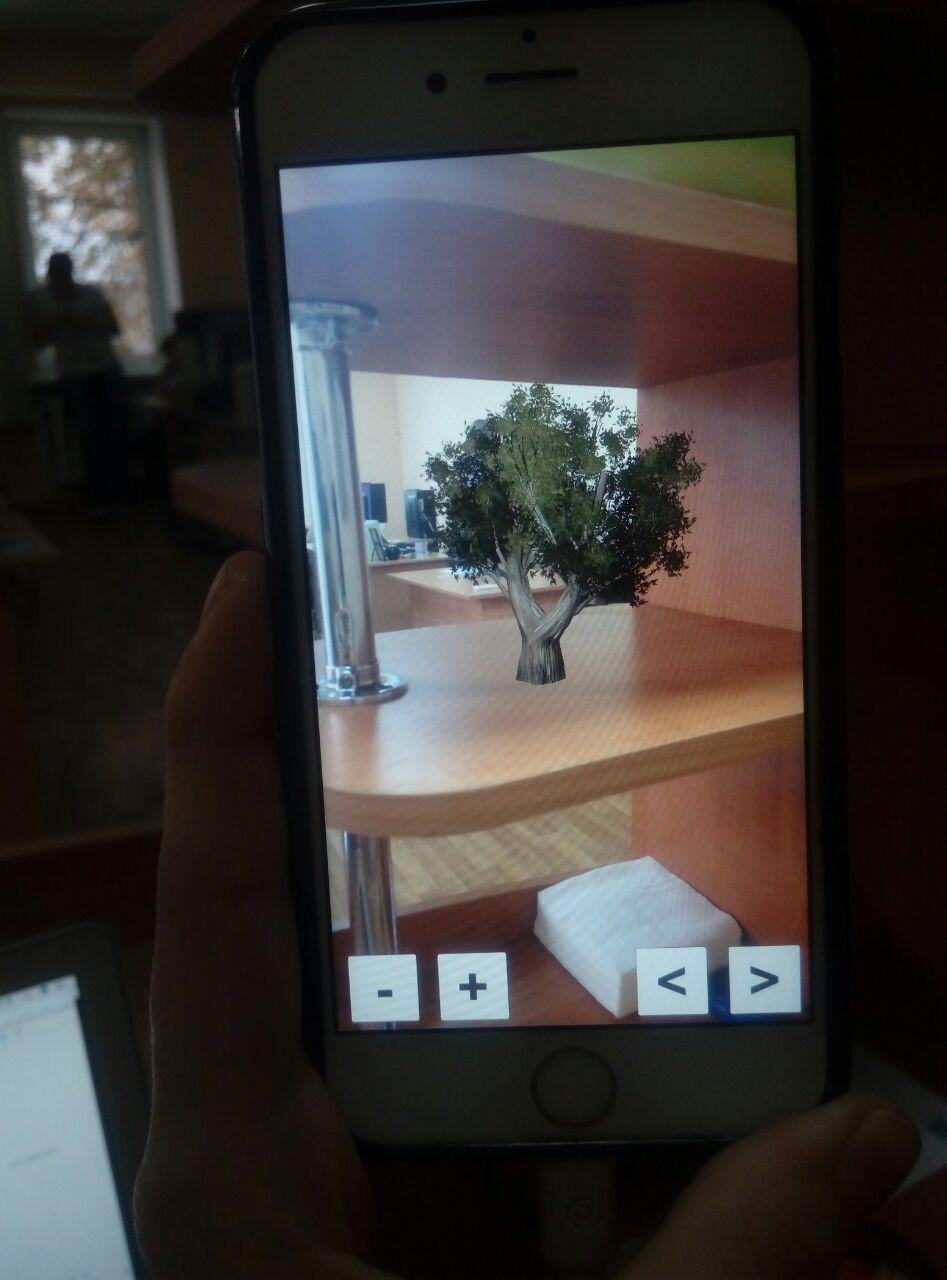


Рисунок 2.1 Додаток «Olive tree», дерево розміщене на поличці



Рисунок 2.2 Додаток «Olive tree», дерево розміщене на ноутбуці



Рисунок 2.3 Додаток «Olive tree», дерево розміщене на підлозі, створюється ефект великого дерева в офісі

Як ще один приклад, створимо додаток з моделлю мерседеса, і додамо більше інтерактивних опцій для користувача. Назвемо цю програму «AR Mersedes». Основа цієї програми та ж сама – Unity ARKit Plugin, тобто класи з нього UnityARHitTestExample і PointCloudParticleExample використовуються таким же чином. Проте, на відміну від попереднього прикладу, після проведення InteractiveHitTest, коли користувач обирає місце, на яке поставити машину, переставляти її на інше місце не можна, а одночасно з її появою з’являються кнопки плюс та мінус для збільшення і зменшення розміру моделі, та набір кнопок у вигляді ключів до машини, що дозволяють відкривати капот, багажник, двері, вмикати фари та змінювати колір машини. За взаємодією з цим набором кнопок відповідає клас ModelController. Лістинги зазначених класів розміщено в додатку Б. Результати роботи програми показано на рис. 2.4-2.6.



Рисунок 2.4 Додаток «AR Mersedes», модель машини поставлено на реальній парковці, капот відчинено



Рисунок 2.5 Додаток «AR Mersedes», відкриті двері машини, вигляд з середини



Рисунок 2.6 Додаток «AR Mersedes», машина пофарбована в синій

В результаті роботи з Unity та ARKit було підтверджено позитивні відгуки про цей плагін. Додатки будувати просто та швидко, продуктивність роботи висока – майже не було так званих «лагів», тобто тимчасова втрата моделі її позиції в просторі. Правильність розпізнання поверхонь також дуже висока.

## 2.2 ARCore

ARCore – комплект розробки програмного забезпечення, створений компанією Google, що дозволяє будувати додатки з доповненою реальністю. ARCore використовує три основні технології для інтеграції віртуального контенту з реальним світом:

1. Відстеження руху дозволяє телефону зрозуміти та відстежувати свою позицію щодо світу.
2. Розуміння навколишнього середовища дає змогу телефону визначити розмір і розташування плоских горизонтальних та вертикальних поверхонь, таких як земля або журнальний столик чи стіна.
3. Оцінка світла дозволяє телефону оцінити поточні умови освітлення середовища.

Samsung оголосила про те, що ARCore буде використовуватися у їх продуктовій лінії.[9]

Основним недоліком ARCore є обмежена кількість девайсів, що підтримують його. На час написання цієї роботи таких моделей є 106 [22], що відносно мале число у порівнянні з різноманітністю смартфонів на ринку. Основною причиною цьому є гіроскоп, адже для роботи ARCore необхідно мати гіроскоп високої якості, але саме на цій деталі апаратного забезпечення частіше всього економлять виробники бюджетних смартфонів.

Для розробки додатків з ARCore також можна використовувати Unity, так як, я і в ситуації з ARKit, існує плагін, який вбудований в Unity. Для того, щоб написати такий додаток, необхідно мати встановленими ndk, jdk, без яких Android програми не зможуть збілдитись, та в налаштуваннях проекту у відділі «XR Setting» поставити відмітку напроти пункту «ARCore Supported». Після цього необхідно завантажити плагін GoogleARCore, в якому є приклади виклику необхідних методів та тестові сцени, на основі яких можна будувати власні додатки. В більшості, процес роботи з ARCore не дуже відрізняється від роботи з ARKit.

На рис. 2.7 показано тестову сцену, що є в комплекті з GoogleARCore. Вона дає змогу розпізнавати поверхні, будувати площини, ставити об’єкти на горизонтальні та вертикальні площини. Особливістю ARCore є те, що при HitTest створюється новий екземпляр моделі і ставиться в місце, в якому промінь від точки дотику до екрану зіткнувся з побудованою площиною.

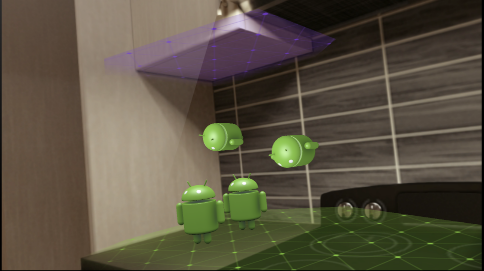


Рисунок 2.7 Тестова сцена від розробників ARCore [9]

На рис. 2.8-2.9 показано, як можна використати оцінку світла – модель має різні анімації залежно від рівня освітленості.



Рисунок 2.8 Анімація моделі за яскравого освітлення [9]



Рисунок 2.9 Анімація моделі за поганого освітлення [9]

## 2.3 Wikitude

Wikitude – це компанія, що програмує основу для додатків доповненої реальності (AR), що базується в Зальцбурзі, Австрія. Заснована у 2008 році, Wikitude спочатку зосереджувала увагу на створенні доповненої реальності на основі місцезнаходження за допомогою програми Wikitude World Browser. У 2012 році компанія реструктуризувала свою пропозицію, запустивши Wikitude SDK, концепцію розвитку, що використовує технологію розпізнавання та відстеження зображення та геолокацію.

Wikitude SDK є основним продуктом компанії. Спочатку запущений у жовтні 2008 р. SDK включає в себе розпізнавання та відстеження зображень, рендеринг 3D-моделей, відео-накладання, AR-локацію та технологію SLAM, яка дозволяє розпізнавати та відстежувати об'єкти, а також відстежувати миттєву позицію без маркеру. Крос-платформний SDK доступний для операційних систем Android і iOS, а також оптимізований для декількох пристроїв для доповненої реальності з окулярами. Програма Wikitude була першою загальнодоступною програмою, яка використовувала підхід, що ґрунтується на розташуванні, до доповненої реальності.

Як приклад однієї з програм, побудованих на основі Wikitude, можна розглянути Augmenting UNESCO Heritage на рис. 2.10, випущений як доповнення до статті в Вашингтон Пост про пам'ятки UNESCO в США. [7]



Рисунок 2.10 Бізон з Єловстоуну на столі в кімнаті читачів Вашингтон Пост [7]

## 2.4 Kudan

Kudan SDK – це двигун доповненої реальності для пристроїв iOS та Android, який дозволяє мобільним додаткам розпізнавати та відстежувати їх фізичне оточення, використовуючи не більше, ніж апаратне забезпечення, яке можна знайти практично в кожному розумному пристрої. Він також пропонує власний 3D-рендеринг. Приклад додатку, що розпізнає поверхні і ставить напільне покриття замість існуючого показано на рис. 2.11.



Рисунок 2.11 Додаток Karndean на Kudan для реклами напільних поверхонь [8]

Цей двигун може бути включений в будь-який проект, тобто існуючі додатки можуть бути покращені з AR так само легко, як і нові програми.[8]

## 2.5 Facebook AR

В 2017 році Facebook оголосив про вихід їх доповненої реальності для мобільних додатків. Особливість Facebook AR полягає в тому, що всі додатки для доповненої реальності працюють всередині додатка для мобільних телефонів Facebook, що дозволяє поширити їх на значно більшу аудиторію. Багато компаній співпрацюють з Facebook щоб мати свою рекламу у вигляді доповненої реальності. Приклади таких колаборацій можна спостерігати на рис. 2.12-2.13.

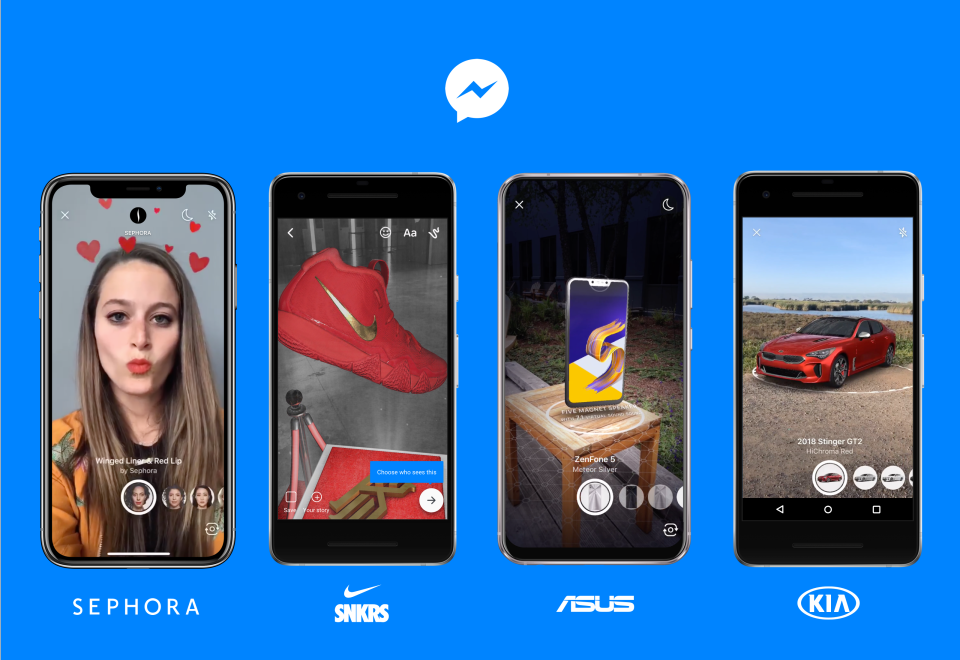


Рисунок 2.12 Партнерство Facebook з компаніями Sephora, Snikers, Asus, Kia [23]



Рисунок 2.13 Рекламна кампанія до виходу фільму «Світ Юрського періоду 2» [24]

Розробники можуть користуватись трьома типами вхідних даних , щоб запускати ефекти доповненої реальності: Face Tracker (але не ідентифікація чи розпізнавання обличчя), дані датчиків, такі як гіроскоп та місцеположення, а також API, щоб залучити дані з інших додатків та реагувати на вхід користувача в реальному часі. [11]

## 2.6 Vuforia 7

Vuforia забезпечує крос-платформну підтримку додаткових рішень для пристроїв Android, iOS та UWP через єдиний API, що дозволяє розробникам писати свої програми одноразово та запускати їх на девайсах з різними операційними системами.

Vuforia 7 – це найбільший і найпотужніший реліз за історію існування Vuforia. Він включає нові можливості для розміщення цифрових даних на поверхні землі, підлоги або на столі – цей спосіб називається Ground Plane. Розробники стверджують, що Ground Plane можливий на великій кількості пристроїв, поширюючись на найширший спектр популярних пристроїв Android і всі пристрої iOS із підтримкою ARKit.

Також в цьому релізі представлено Model Targets – розпізнавання заздалегідь відомих об’єктів в реальному світі та використання їх для відображення цифрових даних. Дана технологія має багато застосувань в заздалегідь відомих середовищах без використання зображень як маркерів.

Починаючи з Unity 2017.2, розробники Unity могли створювати додатки доповненої реальності з підтримкою Vuforia 7, вона стає вбудованою в Unity, дозволяючи розробникам створювати додатки ще швидше та ефективніше.

Головними конкурентами Vuforia 7 на ринку доповненої реальності є, звичайно, ARCore та ARKit. Для порівняння їх роботи було створено простий додаток «AR T-shirt», що розміщує футболку на проскановану поверхню.

Цей додаток розроблений за допомогою Unity. Разом з плагіном, який завантажується при виборі використання Vuforia в налаштуваннях проекту у відділі «XR Setting», додаються тестові сцена та префаби, необхідні для роботи. Префаб – це спеціальний тип асетів, збережений об’єкт з внутрішніми зв’язками, матеріалами, моделями та скриптами, які могли бути в ньому чи в його дочірніх елементах. При роботі з Vuforia відразу впадає в око закритість коду. Всі налаштування відбуваються виключно через редактор. Єдиний клас, який ми можемо редагувати, є DefaultTrackableEventHandler. В ньому вмикаються та вимикаються всі графічні та фізичні складові моделі, як то Renderer, Collider та інші. Лістинг цього класу представлено в додатку В. По аналогії з іншими плагінами, тут також модель футболки потрібно поставити дочірньою до об’єкта «Ground Plane Stage». Що відрізняє Vuforia в позитивну сторону, це наявність стіки, що показує розмір моделі в реальному середовищі (рис. 2.14).

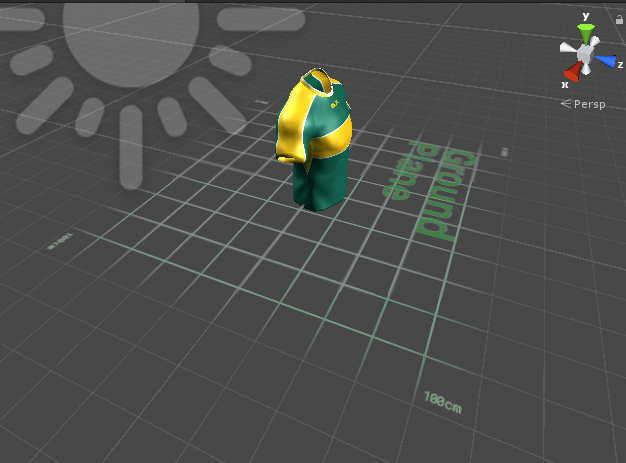


Рисунок 2.14 Скріншот з програми Unity

Результат роботи додатку «AR T-shirt» показано на рис. 2.15. Даний додаток показав нижчі показники, ніж ARKit. Часто спостерігалися стрибки моделі, коли рухалась камера, скоріше всього в цей момент втрачався якір що тримав площину.



Рисунок 2.15 Тестова програма з розміщення футболки в реальному світі н

## 2.7 AR Foundation

Літом 2018 року Unity оголосили про офіційний випуск AR Foundation – плагіна, що дозволяє писати кросплатформні додатки для доповненої реальності. AR Foundation об’єднує ARKit та ARCore в одну сутність, будучи надбудовою над ними. Перший випуск AR Foundation забезпечує підтримку основної функціональності більшості додатків AR:

* виявлення плоскої поверхні;
* глибинні дані представлені у вигляді точкових хмар;
* рендеринг прозорих матеріалів з урахуванням даних з камери;
* орієнтири для допомоги у прив'язці віртуальних об'єктів до фізичного світу;
* оцінки середньої колірної температури та яскравості;
* відстеження положення пристрою та його орієнтація у фізичному просторі;
* утиліти для правильного масштабування вмісту в AR;
* рейкастинг (відбиття променів) від площину та глибині дані.

Unity стверджують, що AR Foundation розвиватиметься, щоб забезпечити доступ до нових функцій платформи та корисних утиліт, зберігаючи при цьому зворотну сумісність. [26]

До офіційного релізу AR Foundation у вільному доступі майже рік був експериментальний проект AR Interface. Цей проект – це було розробки AR Foundation. В описі проекту було написано, що користувачі Unity можуть використовувати ці доробки, але попереджали про нестабільність. З виходом AR Foundation репозиторій для AR Interface закрили. Тобто, до офіційного релізу AR Foundation, програмісти майже рік могли користуватись його перевагами.

Для ознайомлення з AR Foundation було створено додаток «AR Car». Цей додаток є частиною більшої програми, але враховуючи тему дослідження, буде розглянуто лише її частину, пов’язану з доповненою реальністю.

AR Foundation побудований таким чином, що дописувати свій код над його функціоналом дуже просто. Для основного функціоналу, пов’язаного з моделью написано клас ARInterfaceController, який наслідується від класу ARBase, що є частиною AR Foundation.

Цікавою особливістю AR Foundation є функціональне рішення стосовно зміни розміру і положення моделі. Просто змінюючи розмір моделі можна отримати в результаті порушення в фізиці чи роботі системи Particles. Для вирішення цієї проблеми AR Foundation запропонували змінювати положення додаткової камери: віддаляти її щоб візуально зробити об’єкт меншим, наближати щоб зробити більшим, і рухати нею для представлення руху моделі. Ідея ця є дуже розумною, проте через невелику кількість документації, імплементація її в проект є досить важкою. В класі ARInterfaceController такі взаємодії описані, вони виконують жестами на екрані смартфону. Лістинг класу представлено в додатку Г.

Як і аналоги, надбудовою над якими є AR Foundation, в розробників є різні методи відображення просканованих поверхонь та опорних точок. На рис. 2.16 представлено відображення площини синьою рамкою. Всередині цієї рамки модель може вільно переміщатись.



Рисунок 2.16 Модель машини поставлена на робочий стіл з виділеною просканованою поверхнею

Скріншоти машини з відкритими капотом та дверима і ввімкненими фарами показано на рис. 2.17-2.18.

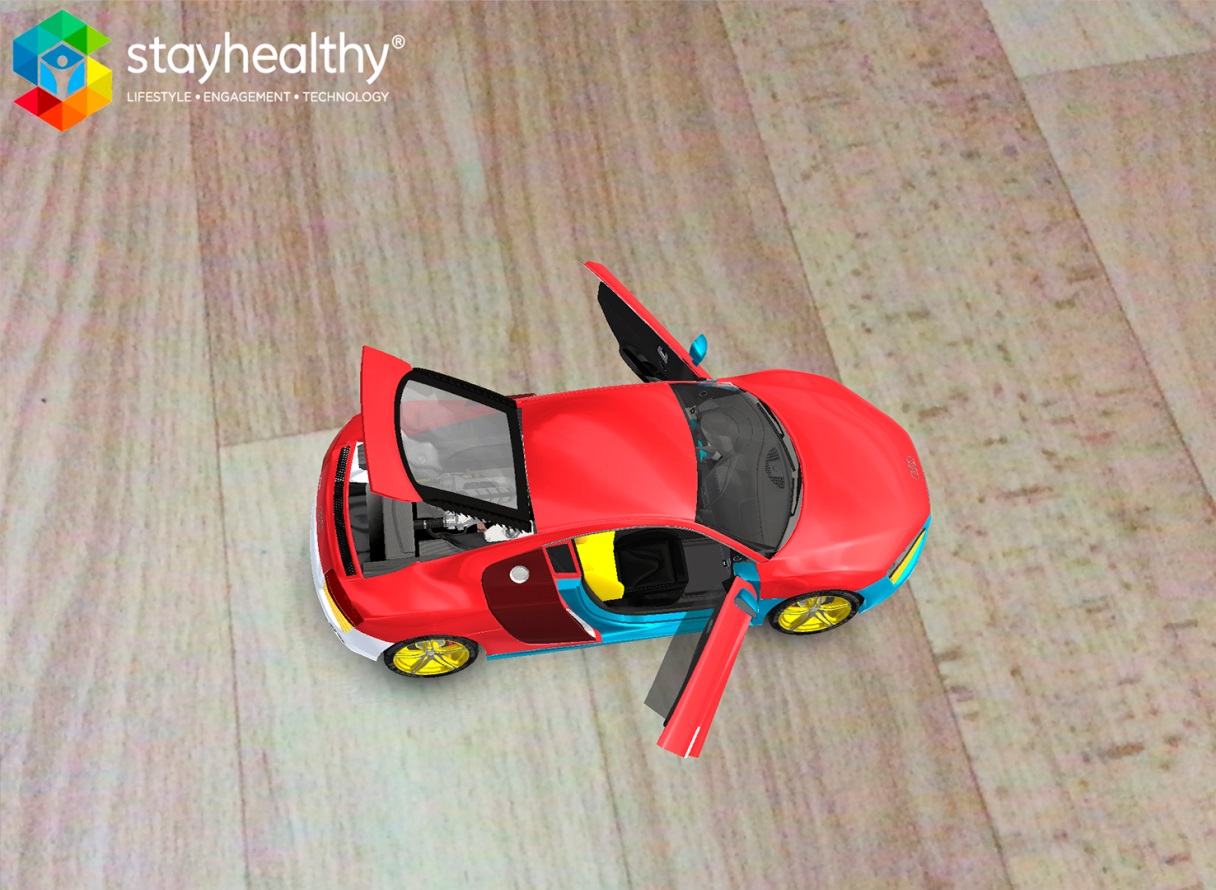


Рисунок 2.17 Модель на підлозі



Рисунок 2.18 Модель на підлозі з ввімкненими фарами

Протягом роботи з цим плагіном було зроблено висновки, що за продуктивністю він не поступається ARKit, проте деколи є важким до розуміння, і необхідно довго шукати рішення до певних задач.

## 2.8 Tango

Tango (раніше відома як Project Tango) – платформа для обчислень доповненої реальності, розроблена Google. Технологія застосовує комп'ютерний зір для мобільних пристроїв, таких як смартфони і планшетні комп'ютери, щоб з'ясовувати своє місцезнаходження відносно навколишнього світу без використання GPS або інших зовнішніх сигналів. Це дозволяє розробникам додатків створювати призначений для користувача інтерфейс, що включає внутрішню навігацію, 3D-картографування (рис. 2.19), вимір фізичного простору, розпізнавання навколишнього середовища, доповнену реальність і вікна в віртуальний світ.

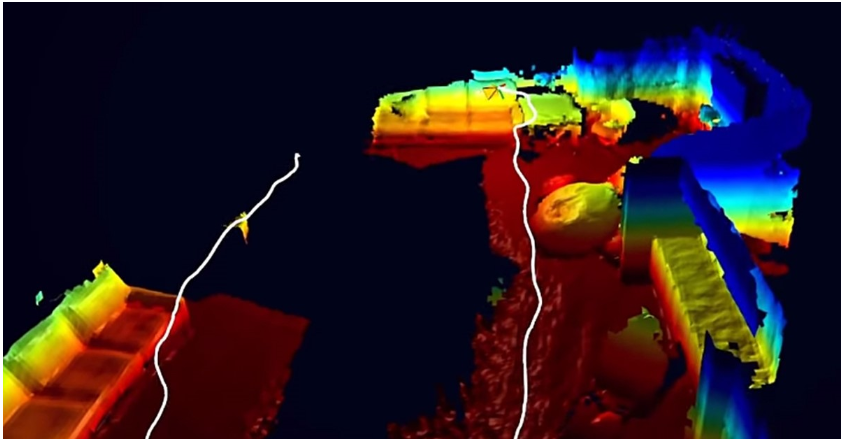


Рисунок 2.19 Побудова 3D карти навколишнього середовища [25]

1 березня 2018 року Google припинила розробку Tango щоб зосередити всі ресурси на ARCore.[10]

## Висновки до розділу 2

В цьому розділі було розглянуто декілька реалізацій алгоритму SLAM для доповненої реальності. Найпопулярнішою серед розробників є ARKit, за нею йде AR Foundation. На третьому місці стоїть Vuforia 7. І хоч Vuforia показує гірші результати ніж ARCore, вона більш поширена завдяки своїм попереднім здобуткам в сфері розпізнавання маркерів.

ARCore має такі низькі показники через кілька причин – в більшості це мала кількість телефонів, на яких він працює. Тому значно кращою альтернативою є AR Foundation, який ґрунтується на ARCore та ARKit, захоплюючи одночасно велику кількість користувачів.

Facebook AR представляє собою дуже цікаву платформу – різні програми для доповненої реальності зібрані в одному місці, тобто користувачу не потрібно завантажувати окремі додатки. Проте це створює і обмеження на те, що можна створити з цією технологією.

Wikitude, Kudan і Tango є найстарішими реалізаціями з розглянутих. Їх важко використовувати, для роботи з ними потрібно сплачувати або за доступ до коду, або за дороге апаратне забезпечення.

# РОЗДІЛ 3 ПРОБЛЕМИ ДОПОВНЕНОЇ РЕАЛЬНОСТІ ТА ПРОПОЗИЦІЇ ЇХ ВИРІШЕННЯ

## 3.1 Проблеми в доповненій реальності

Існуючі реалізації для доповненої реальності, звичайно, не ідеальні.