Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів обробки та проеціювання web-контенту з використанням засобів доповненої реальності
(тема)

Виконав: студент 2 курсу, групи ІПЗм-17-1
спеціальності 121-Інженерія програмного забезпечення
(код і повна назва спеціальності)
Освітньо-наукової програми Інженерія програмного забезпечення

Брижниченко А.В.
(прізвище, ініціали)
Керівник проф. Лєсна Н.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ З.В.Дудар

2019 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук

Кафедра програмної інженерії

Рівень вищої освіти другий (магістерський)

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

Освітньо-наукова програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

Студентові Брижниченку Андрію Володимировичу
(прізвище, ім'я, по батькові)

1.Тема роботи Дослідження методів обробки та проеціювання web-контенту з використанням засобів доповненої реальності затверджена наказом по університету університету від «18» квітня 2019р №546Ст

2. Термін подання студентом роботи до екзаменаційної комісії «20» 06 2019 р.

3. Вихідні дані до роботи *засоби і технології AR розробки, OS MacOS, об'єктно-орієнтовані мови програмування Swift, Python, середовище розробки XCode.*

4. Перелік питань, що потрібно опрацювати в роботі *аналіз предметної галузі і постановка задачі, засоби і технології реалізації доповненої реальності, проектування моделі алгоритму вирішення проблематики роботи, опис програмної системи генерації та конвертації веб контенту та засобів машинного навчання для визначення маркерів доповненої реальності.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій *причини виникнення проблем оновлення компонентів в системах доповненої реальності, генерація даних, методи оптимізації компонентів, засоби конвертації отриманих даних, структура програмної системи дослідження продуктивності, обрані засоби та технології, структура таблиць, які містять порівняльні дані, результати моделювання.*

6 Консультанти розділів роботи

Найменування Розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	проф. Лєсна Н.С.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка*
1.	Аналіз предметної галузі	22 квітня 2019р.	
2.	Опрацювання першоджерел та постановка задачі	10 травня 2019р.	
3.	Розробка методики та моделі	20 травня 2019р.	
4.	Оформлення результатів дослідження	28 травня 2019р.	
5.	Підготовка пояснювальної записки	5 червня 2019р.	
6.	Підготовка презентації та доповіді	15 червня 2019р.	
7.	Перевірка роботи на антиплагіат	16 червня 2019р.	
8.	Нормоконтроль	18 червня 2019р.	
9.	Рецензування	20 червня 2019р.	
10.	Занесення атестаційної роботи в електронний архів	21 червня 2019р.	
11.	Попередній захист	22 червня 2019р.	
12.	Допуск до захисту у зав. кафедри	22 червня 2019р.	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання _____22_квітня_____2019 р.

Студент _____

(підпис)

Керівник роботи _____проф. Лєсна Н.С._____

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до до атестаційної роботи: 90 с., 32 рис., 1 табл., 4 додатки, 24 джерел.

ДОПОВНЕНА РЕАЛЬНІСТЬ, ВЕБ КОНТЕНТ, ПРОЕЦІЮВАННЯ, APPLE, ARKIT

Об’єктом дослідження є існуючі методи, підходи та інструментарій доповненої реальності та використання веб-контенту у поєднанні з цими методами. Метою роботи є дослідження існуючих методів, підходів та інструментів для реалізації застосунків на базі доповненої реальності та використання веб-контенту у поєднанні з цими методами. В рамках переддипломної практики було проаналізовано існуючі типи та підходи використання доповненої реальності, виявлено сферу застосування цього підходу, а також виявлено підходи проеціювання веб-контенту.

AUGMENTED REALITY, WEB CONTENT, PROJECTION, APPLE, ARKIT

The subject of the research is the existing methodological approach and tools for augmented reality and the use of web content in combination with these methods. The aim of tihs work is to explore existing methods, approaches and tools for implementing applications based on augmented reality and the use of web content and methods with these methods. In the course of pre-diploma practice, existing types and approaches of the use of augmented reality, revealed by the scope of this approach were analyzed, as well as the approach of projecting web content.

ЗМІСТ

ВСТУП

Останнім часом завдяки активному розвитку технологій людство все частіше реалізує концепти що були закладені ще наприкінці минулого сторіччя, зокрема такі як реалізація віртуальної та доповненої реальності.

Бурхливий розвиток мобільних технологій дозволяє реалізувати засоби доповненої реальності навіть на смартфоні користувача, утилізуючи як програмні, так і апаратні потужності пристрою. Також останні декілька років між великими корпораціями йде гонка розвитку цього напрямку, з Apple ARKit, Google ARCore та численними менш відомими фреймворками для роботи з доповненою/віртуальною реальністю.

Чому ж доповнена реальність стає такою популярною?

По перше, як уже було сказано вище, через розвиток апаратної складової. Реалізація доповненої реальності передбачає просторові обчислення та аналіз зображення з камери, тобто використовує засоби машинного навчання [1], що раніше не було можливим у не-лабораторних умовах. Тому з розвитком машинного навчання і технології доповненої реальності теж набули активного розвитку, а збільшення потужності та зменшення розміру апаратних комплектуючих надало можливість використання моделей машинного навчання разом із сенсорами смартфонів для досягнення мети доповненої реальності.

По друге, методи доповненої реальності можуть бути використані у широкому спектрі застосувань, дозволяючі візуалізувати програмні об’єкти у проекції на реальний світ через камеру користувача [2], що може бути корисним у медицині, освіті, розважальній та навіть військових сферах. Вже зараз існують численні рішення з використанням AR: інтерактивні гіди, застосунки для тренувань, що показують модель тіла людини відносно тренажеру, медичні застосунки, що проеціюють внутрішні травми на тіло людини, тощо.

Актуальність теми атестаційної роботи обумовлена великим попитом на AR застосунки та стрімким розвитком технологій, потенційні можливості котрих ще не повністю вивчені.

Метою роботи є дослідження існуючих методів доповненої реальності та їх різновидів, а також можливості доставки та взаємодії веб-контенту при використанні цих методів.

Об’єктом дослідження є процес генерації та конвертації компонентів доповненої реальності.

Предметом дослідження є методи, засоби і технології побудови і проеціювання доповненної реальності.

В ході даної роботи задля дослідження ефективності використаних програмних методів, засобів і технологій застосовувались емпіричні методи програмної інженерії, загальнологічні методи наукового пізнання, а також порівняльний метод.

Наукова новизна. Об’єднано методи генерації з удосконаленими методами оптимізації та конвертації нейронних мереж для використання у зв’язці з доповненою реальністю, розроблено методи проеціювання веб-контенту за допомогою програмних об’єктів.

Практичне значення. Результати роботи та розроблена модель системи допоможе зменшити ресурсні (часові, людські, машинні) витрати на зміни компонентів доповненої реальності.

Публікації. Результати дослідження, проведеного в атестаційній роботі, пройшли апробацію в рамках десятої Міжнародної наукової конференції «Perspectives of Science and Education».

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність роботи

Останнім часом завдяки активному розвитку технологій людство все частіше реалізує концепти, що були закладені ще наприкінці минулого сторіччя, зокрема такі як реалізація віртуальної та доповненої реальності.

Особливо бурхливого розвитку набули мобільні технології, що дозволяють використовувати засоби доповненої реальності навіть на смартфоні користувача, утилізуючи як програмні так і апаратні потужності пристрою.

Доповнена реальність (AR) – це інтерактивний досвід середовища реального світу, де об'єкти, що знаходяться в реальному світі, "доповнюються" комп'ютерно-генерованою перцептивною інформацією, іноді через декілька сенсорних модальностей, включаючи зорові, слухові, гаптичні, соматосенсорні і нюхові.

Накладена сенсорна інформація може бути конструктивною (тобто добавкою до природного середовища) або маскуючою (тобто маскуванням природного середовища) і безперешкодно переплітається з фізичним світом так, що вона сприймається як аспект реального середовища. Таким чином, доповнена реальність змінює постійне сприйняття середовища реального світу, на відміну від віртуальної реальності, що повністю замінює реальне середовище користувача імітаційною. Доповнена реальність пов'язана з двома в значній мірі синонімічними термінами: змішаною реальністю і комп'ютерно-опосередкованою реальністю.

Основна цінність розширеної реальності полягає в тому, що вона привносить компоненти цифрового світу до сприйняття людиною реального світу і робить це не як просте відображення даних, а через інтеграцію захоплюючих відчуттів, які сприймаються як природні частини навколишнього середовища [3]. Доповнена реальність використовується для посилення природного середовища або ситуацій і пропонує сприйняття збагаченого досвіду.

Це зробило Augmented Reality (AR) однією з найбільш освітлених технологій 2018 року, що зображено на графіку рисунка 1.1

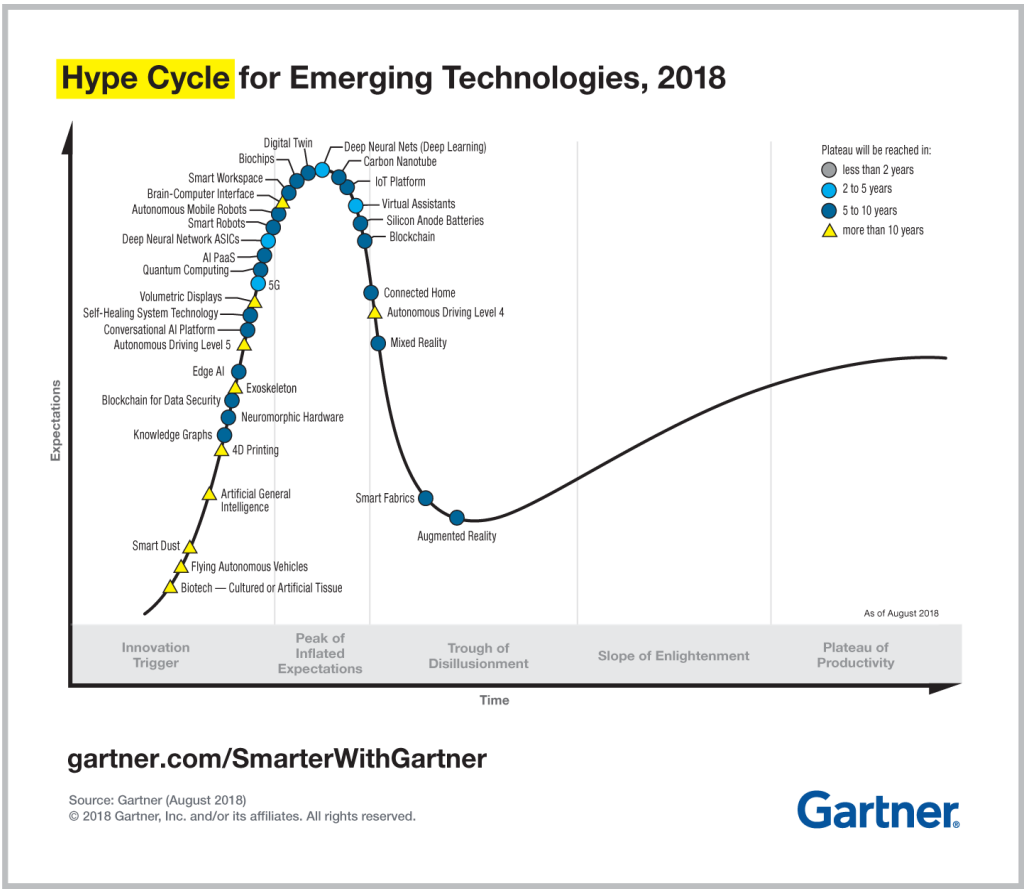

Рисунок 1.1 – Графік популярності технологій за 2018 рік

Перші функціональні системи AR, що забезпечували користувачам вражаючі враження про змішану реальність, були винайдені на початку 1990-х років, починаючи з системи віртуальних світильників, розробленої в 1992 році в Армстронській лабораторії ВПС США[4]. Ігрові підприємства, а тепер і інші галузі також зацікавлені в можливостях AR, наприклад, у обміні знаннями, освіті, управлінні інформаційним потоком і організації дистанційних зустрічей. Доповнена реальність також трансформує світ освіти, де вміст може бути доступний шляхом сканування або перегляду зображення за допомогою мобільного пристрою або шляхом залучення до

класу захоплюючих, без маркерів досвіду AR. Іншим прикладом є шоломи AR для будівельників, які відображають інформацію про будівельні об'єкти.

За допомогою передових технологій AR (наприклад, додавання комп'ютерного зору та розпізнавання об'єктів[5]) інформація про навколишній реальний світ користувача стає інтерактивним та цифровим. Інформація про навколишнє середовище та його об'єкти накладається на реальний світ. Ця інформація може бути віртуальною або реальною, наприклад, бачачи іншу реальну відчутну або виміряну інформацію, таку як електромагнітні радіохвилі, накладені в точному узгодженні з місцем, де вони перебувають у просторі. Доповнена реальність також має величезний потенціал у зборі та обміну мовчазними знаннями. Методи розширення зазвичай виконуються в реальному часі і в семантичному контексті з елементами навколишнього середовища. Імперсивна перцептивна інформація іноді поєднується з додатковою інформацією, подібною оцінці над живим відеострімом спортивної події. Це поєднує в собі переваги технології розширеної реальності та технології відображення інтерфейсу (HUD).

1.2 Виявлення проблем та актуалізація рішень

Зараз існують способи пакування та доставки контента у мобільний застосунок, проте основною проблемою цих способів є те, що зазвичай контент, що проеціюється, знаходиться у основному пакеті системи та обробляється під час компіляції та пакування системи у виконуваний файл, тому необхідність створення системи, що реалізує метод динамічного постачання контента, є доволі високою.

Однією з ключових проблем у створенні досвіду доповненої реальності є належне закріплення віртуального вмісту в реальному світі; процес, який вимагає унікального набору сприйнятливий технологій, здатних відстежувати високу динаміку поверхні реальних об’єктів і належного проеціювання цільової моделі на ці поверхні, для цього необхідні належним чином треновані нейронні мережі, що здатні класифікувати і

визначати шукані об’єкти як якорі для проеціювання контенту. Проблема полягає в тому, що для належного використання моделей нейронних мереж вони мають бути включені у бандл мобільного застосунку [6], що значно обмежує гнучкість розробки таких систем. Тому було вирішено дослідити цю проблему та запропонувати її вирішення шляхом динамічного постачання тренованих моделей до кінцевого користувача з можливістю подальшого використання у встановленому на пристрої користувача додатку.

Також значною проблемою є динамічність моделей нейронних мереж доповненої реальності. Зазвичай це заготовлені моделі що йдуть вже у SDK “з коробки”: обличчя, площини, qr-коди, собаки, кішки, без можливості змінити їх “на льоту”, а при спробі додати свою власну нейронну мережу користувач зустрінеться з проблемами сумісності, високого навантаження на систему та великого розміру цих моделей, тож окрім проблеми динамічності ще існує проблема оптимізації [7].

1.3 Поняття змішаної, доповненої та віртуальної реальності

У 1994 році Пол Мілграм і Фуміо Кішино визначили змішану реальність як "... десь між екстремумами континууму віртуальності" (VC), де континуум віртуальності поширюється від абсолютно реального до повністю віртуального середовища з розширеною реальністю і збільшеною віртуальністю. між ними. Перша повністю занурена система змішаної реальності була платформою Virtual Fixtures, розробленою у ВПС США, лабораторіями Armstrong в 1992 році Луїсом Розенбергом, щоб дозволити користувачам керувати роботами в реальних середовищах, які включали реальні фізичні об'єкти та 3D віртуальні накладки, які називаються "світильниками" які додавалися, підвищують продуктивність виконання маніпуляційних завдань. Опубліковані дослідження показали, що, впроваджуючи віртуальні об'єкти в реальний світ, значні збільшення продуктивності можуть бути досягнуті людиною.

Континуум змішаної реальності є однією з двох осей у концепції Стіва Манна про опосередковану реальність, реалізовану різними зварювальними шоломами та перносними комп'ютерами, а також перносними фотографічними системами, які він створив у 1970-х та на початку 1980-х років, а друга ось – медіальний континуум, наприклад, змішана реальність., схема екстремумів яких зображена на рисунку 1.2.

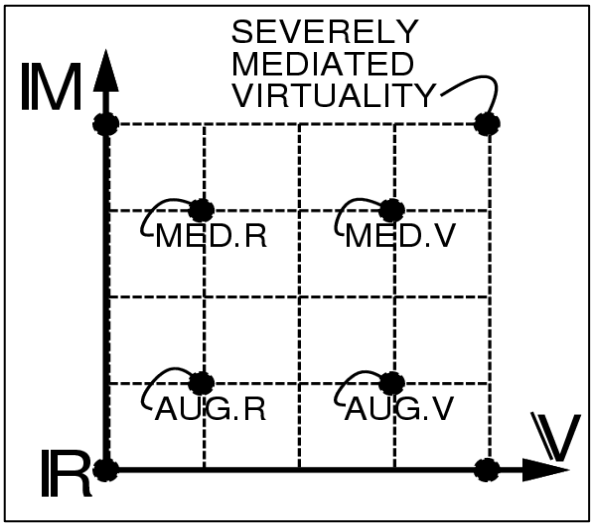

Рисунок 1.2 – Суворо опосередкована віртуальність

Вісь віртуальності (зліва направо) і вісь медіальності (знизу вгору) континууму опосередкованої реальності. Тут показані чотири приклади: доповнена реальність, розширена віртуальність, опосередкована реальність і опосередкована віртуальність на вісі віртуальності і медіальності. Це включає, наприклад, зменшену реальність (наприклад, комп'ютеризовані зварювальні шоломи, які відфільтровують і зменшують певні частини сцени).

Традиційно розглядається середовище віртуальної реальності, в якому учасник-спостерігач повністю занурюється і здатний взаємодіяти з повністю синтетичним світом, який може імітувати властивості деяких реальних середовищ. як існуючих, так і вигаданих, але він також може перевищувати межі фізичної реальності, створюючи світ, в якому фізичні закони, які зазвичай керують простором, часом, механікою, властивостями матеріалу і т.д., більше не тримаються. Однак, те, що VR мітка також

часто використовується у зв'язку з безліччю інших середовищ, до яких повне занурення і повний синтез не обов'язково стосуються, але які потрапляють десь уздовж континууму віртуальності. підклас пов'язаних з ВР технологій, які передбачають злиття реальних і віртуальних світів, які ми називаємо загалом як Змішана реальність (MR).

У контексті фізики термін "система інтерреальності" (див. рис. 1.2) відноситься до системи віртуальної реальності, пов'язаної з її колегою реального світу. Доповідь у випуску Physical Review E від травня 2007 року описує систему інтерреальності, що включає реальний фізичний маятник, приєднаний до маятника, який існує тільки у віртуальній реальності, градацію переходу від реальності до власне віртуальності можна побачити на рисунку 1.3.

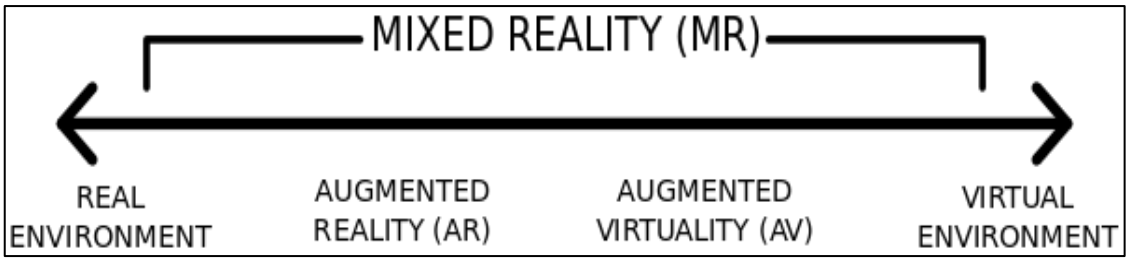

Рисунок 1.3 – Континуум реальності-віртуальності

Ця система, мабуть, має два стабільних станів руху: стан "подвійної реальності", при якому рух двох маятників є некорельованим і станом "змішаної реальності", в якому маятник має стабільний фазово-заблокований рух, який сильно корелює. Використання термінів "змішана реальність" і "інтерреальність" у контексті фізики чітко визначено, але може бути дещо іншим в інших областях.

Важливо чітко розділяти поняття віртуальної, доповненої та змішаної реальностей. Віртуальна реальність вимагає особливого обладнання, такого як шоломи, датчики, джерело струму для повної емуляції реальності, доповнена та змішана ж реальності, як це слідує з назви, використовують оточення та доповнюють його проекціями, що

вимагає набагато менших затрат ресурсів. Саме тому доповнена реальність зараз набагато доступніша, перспективніша, та, як наслідок, актуальніша.

1.4 Постановка задачі

Метою дослідження є дослідження методів обробки та проеціювання web-контенту з використанням засобів доповненої реальності.
У світлі високого зросту можливостей нейронних мереж та полегшення доступу до машинного навчання протягом останніх років, було прийнято рішення досліджувати методи побудування доповненої реальності.

У сфері доповненої реальності доволі гостро стоїть проблема постачання динамічного контенту, тому конкретизація теми зводиться саме до дослідження методів вирішення проблеми проеціювання веб-контенту засобами доповненої реальності, також, враховуючи розповсюдження та доступність мобільних пристроїв, збільшення їх потужностей і покращення роботи їх сенсорів було обрано саме мобільні пристрої як цільову платформу.

Таким чином, для досягнення мети дипломної роботи необхідно дослідити:

– існуючі методи доповненої реальності;

– існуючі засоби доповненої реальності;

– існуючі інструменти доповненої реальності.

А також розробити систему, що може вирішити цей комплекс задач, а саме буде здатна:

– тренувати на обраному датасеті та зберігати моделі доповненої реальності;

– конвертувати ці моделі у необхідний застосунку формат;

– постачати їх на мобільний застосунок;

– зберігати та передавати мобільному застосунку тривимірний, двовимірний або динамічний веб-контент у вигляді 3D моделей, зображень, відео, відеопотоку, анімацій, веб-сторінок;

– динамічно обробляти цей контент для подальшого використання у доповненій реальності;

– побудувати модель відображення отриманого контента та передавати отриманий результат на компонент доповненої реальності для подальшого проеціювання.

Процес передачі, генерації та проеціювання контента має відбуватися в реальному часу у працюючому додатку (runtime) на мобільному пристрої т серверах тренування нейронних мереж.

Для вирішення задачі створення і постачання нейронних мереж необхідно налаштувати алгоритм тренування моделі нейронної мережі на віддаленому сервері із запропонованим датасетом, переведення цієї мережі у формат, що буде здатний працювати з мобільними технологіями доповненої реальності.

При вирішенні другої задачі варто враховувати те, що доповнена реальність оперує просторовими моделями, тому при отриманні веб-контенту і проекції його на реальний світ необхідно спочатку побудувати віртуальний об’єкт, на який цей контент буде проеціюватися, що враховує складну просторову геометрію розташування для будування цього об’єкта.

2 ЗАСОБИ І ТЕХНОЛОГІЇ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

2.1 Компоненти доповненої реальності

Кожен засіб доповненої реальності – це комплекс компонентів, що спрямований на досягнення єдиної цілі – проекції контенту на реальні об’єкти, тому всі відмінності між засобами доповненої реальності зумовлені відмінностями між їх компонентами. В зв’язку з цим перед дослідженням засобів необхідно спочатку дослідити типові компоненти засобу доповненої реальності.

Апаратними компонентами для розширеної реальності є:

– процесор;

– дисплей;

– датчики;

– пристрій введення.

В свою чергу апаратні компоненти потребують програмних засобів для коректної роботи, ними є:

– комп'ютерне бачення: запрограмований додаток або набір програмного забезпечення має об'єднати два типи зображень [8];

– один з фотоапаратів і один з них, створений за допомогою маркерів;

– реєстрація оточення, що складається з піктограм (які користувач не бачить), через який комп'ютеризована частина пристроїв розміщує об'єкт AR в реальному світі. Піктограми або маркери використовують різні орієнтири із зображення реального світу для орієнтування, такі як кут, виконаний між двома стінами, лінії вікна, геометрична форма килима тощо;

– поле зору: звичайне поле зору людини близько 210 градусів по горизонталі і 150 градусів по вертикалі. Окуляри AR не можуть відтворювати ці цифри, насамперед через фізичні обмеження розміщення лінз в гарнітурі. Потім виникає проблема обчислювальної потужності, необхідної для обробки та відображення цифрових

об'єктів з достатньо високою точністю і роздільною здатністю. В даний час 50-градусне поле зору вже вважається великим для окулярів з розширеною реальністю;

Результатом комбінації цих компонентів є те, що ви дійсно бачите через окуляри розширеної реальності.

Прототип пристрою, що складається з усіх цих компонентів можна побачити на рисунку 2.1.

Рисунок 2.1 – Типовий пристрій доповненої реальності

Сучасні мобільні обчислювальні пристрої, такі як смартфони і планшетні комп'ютери, містять такі елементи, які часто включають камери і MEMS-сенсори, такі як акселерометр, GPS і твердотільний компас, що робить їх придатними платформами AR.

AR може відображатися на різних пристроях: екранах, окулярах, портативних пристроях, мобільних телефонах, дисплеях на голові. Вона включає такі технології, як одночасна локалізація і відображення, відстеження глибини (дані датчика, що обчислюють відстань до об'єктів).

Перелік компонентів:

– камери та датчики: збір даних про взаємодії користувача та надсилання його для обробки. Камери на пристроях сканують околиці і за допомогою цієї інформації пристрій виявляє фізичні об'єкти і генерує 3D-моделі. Це можуть бути спеціальні камери, як у Microsoft Hololens, або звичайні смартфонові камери для фотографування / відео;

– обробка: пристрої AR в кінцевому підсумку повинні діяти як маленькі комп'ютери, те, що сучасні смартфони вже роблять. Таким же чином, вони вимагають CPU, GPU, флеш-пам'яті, оперативної пам'яті, Bluetooth / WiFi, GPS, і т.д.

– проекція: це стосується мініатюрного проектора на гарнітурі AR, який приймає дані з датчиків і проектує цифровий контент (результат обробки) на поверхню для перегляду. Фактично, використання прогнозів в AP ще не було повністю винайдене, щоб використовувати його в комерційних продуктах або послугах;

– відображення: деякі пристрої AR мають дзеркала для допомоги людським очам для перегляду віртуальних зображень.

Деякі з них мають «масив невеликих вигнутих дзеркал», а деякі мають двостороннє дзеркало, яке відображає світло на камеру та на очі користувача.

Метою таких шляхів відображення є виконання правильного вирівнювання зображення.

На відміну від концептуальних великих AR пристроїв, мобільні поєднують всі ці компоненти у лаконічному компактному вигляді.

Насправді з появою смартфонів кожен з них вже був комплектований всім необхідним, не вистачало лише потужностей та програмного забезпечення, яке могло б оптимізувати процес.

Проте, як це часто буває, реальність виявилась інакшою, аніж планували у минулому столітті.

Апаратне та програмне забезпечення легко пропускати з уваги на дизайн та досвід. Однак, працюючи в нових і нових технологіях, знаючи, як працюють речі, можна надихнути і навіть внести інновації в цей простір.

Нижче наведено розбиття різних частин високого рівня, що робить роботу програми AR можливою на мобільному пристрої.

Апаратними компонентами для розширеної реальності є:

– процесор;

– дисплей;

– датчики;

– пристрій введення.

Схему сучасного смартфону здатного до проеціювання AR можна побачити на рисунку 2.2.

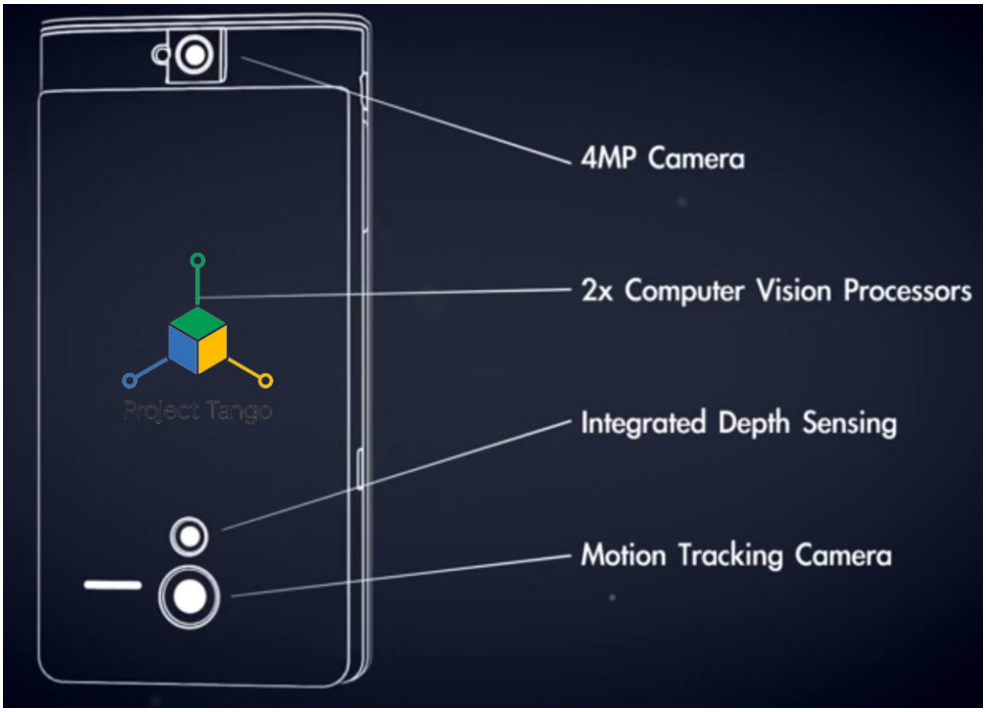

Рисунок 2.2 – Мобільний пристрій з апаратними компонентами AR

Однак вирішальну роль грає саме програмне забезпечення, тому що ключовим показником систем AR є те, наскільки реально вони інтегрують доповнення з реальним світом. Програмне забезпечення повинне отримувати координати реального світу, незалежно від камери, з зображень камери. Цей процес називається реєстрацією зображень, який використовує різні методи комп'ютерного зору, здебільшого пов'язані

з відеоспостереженням. Багато методів комп'ютерного зору розширеної реальності успадковані від візуальної одометрії. Зазвичай ці методи складаються з двох частин.

Спочатку визначення точкок інтересу, або фідуціарні маркери, або оптичний потік у зображеннях камери. Перший етап може використовувати способи виявлення ознак, такі як виявлення кутів, виявлення крапель, виявлення країв або порогове значення та інші способи обробки зображень. Другий етап відновлює систему координат реального світу з даних, отриманих на першому етапі.

Деякі методи передбачають об'єкти з відомою геометрією (або фідуціарними маркерами), присутніми в сцені. У деяких з цих випадків структура 3D сцени повинна бути попередньо розрахована. Якщо частина сцени невідома, одночасна локалізація і відображення (SLAM) може відображати відносні позиції. Якщо немає інформації про геометрію сцени, використовуються структури з методів руху, як налаштування пакетів. Математичні методи, що використовуються на другому етапі, включають в себе проективну (епіполярну) геометрію, геометричну алгебру, представлення обертання з експоненціальним картою, фільтри Калмана, нелінійну оптимізацію, надійну статистику.

Розширена мова розмітки реальності (ARML) є стандартом даних, розробленим в рамках Open Geospatial Consortium (OGC), який складається з граматики XML, що описує розташування і зовнішній вигляд віртуальних об'єктів сцени, а також прив'язки ECMAScript для забезпечення динамічного доступу до властивості віртуальних об'єктів.

2.2 Типові методи доповненої реальності

Розглянемо відомі теоретичні методи проеціювання доповненої реальності, від фізичних з участю спеціально проецийованих променів світла, до повністю програмних,

для кращого розуміння принципу та різновидів роботи цих методів і подальшого використання.

2.2.1 Доповнена реальність на основі маркера.

Доповнена реальність з розширеною реальністю, заснована на маркерах (доповнена реальність на основі маркерного зображення) (яка також називається розпізнаванням зображень), використовує камеру та певний тип візуального маркера, наприклад, QR / 2D-код, для отримання результату тільки тоді, коли маркер відчувається читачем, що зображено на рисунку 2.3.

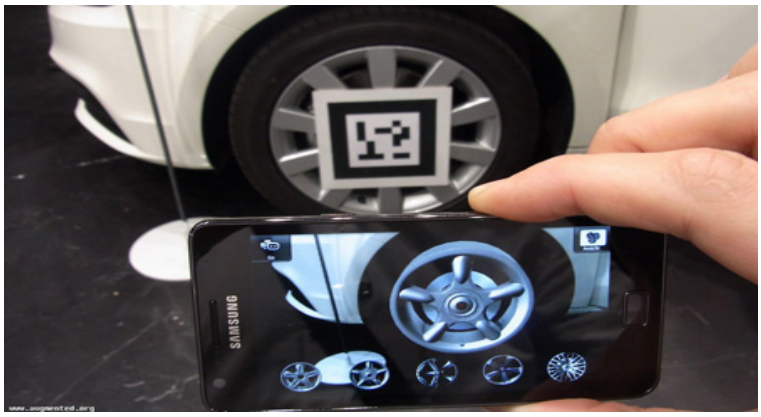

Рисунок 2.3 – Доповнена реальність на основі маркера

Додатки, що базуються на маркерах, використовують камеру на пристрої, щоб відрізнити маркер від будь-якого іншого об'єкта реального світу. Окремі, але прості шаблони (наприклад, QR-код) використовуються як маркери, тому що вони можуть бути легко розпізнані і не вимагають великої потужності обробки для читання. Положення і орієнтація також обчислюються, в яких якийсь тип контенту та / або інформації потім перекривається маркером. Це може бути що завгодно, від друкованого QR-коду до спеціальних знаків. Пристрій AR також обчислює положення

і орієнтацію маркера, щоб позиціонувати вміст, в деяких випадках. Таким чином, маркер ініціює цифрові анімації для перегляду користувачами, і тому зображення в журналі можуть перетворюватися на 3D-моделі.

2.2.2 Markerless доповнена реальність.

Як один з найбільш широко впроваджених додатків реальності, без маркерних (також називається розташування, на основі позицій або GPS) доповненої реальності, використовує GPS, цифровий компас, вимірювач швидкості або акселерометр, який вбудований у пристрій надавати дані на основі вашого місцезнаходження. Сильною силою технології без маркерної доповненої реальності є широка доступність смартфонів та функцій виявлення місцезнаходження, які вони надають. Найчастіше використовується для відображення напрямків, пошуку сусідніх підприємств та інших мобільних додатків, орієнтованих на місцезнаходження.

Системи AR без маркерів можуть використовувати реальні зображення та об'єкти для розміщення камери в 3D-просторі і привертати увагу до реальної картини. Серцевим маркером без AR є алгоритми розпізнавання зображень і виявлення об'єктів.

На відміну від маркерів, форма і внутрішня структура яких фіксовані і відомі, реальні об'єкти не можуть бути визначені таким чином. Також об'єкти можуть мати складну форму і вимагатимуть модифікованих алгоритмів оцінки позиції, щоб знайти їх правильні 3D-перетворення.

З недавнім появою передових систем камер і більш точних датчиків в основних пристроях, таких як iPhone від Apple, Augmented Reality завершила перехід від активації на основі зображень або QR-кодів до немаркованого досвіду з розширеною реальності. Спочатку називається «мертвим розрахунком», Markerless AR використовує комбінацію систем камер, спеціальних датчиків і складної математики

для точного виявлення і відображення реального середовища – наприклад, розташування стін і точок перетину.

За допомогою карти області програма з підтримкою AR дає можливість розміщувати віртуальні об'єкти в реальному контексті і залишати їх на місці без необхідності використання QR-коду або зображення.

Приклад роботи такого методу можна побачити на рисунку 2.4.

Рисунок 2.4 – Доповнена реальність без маркерів

Як можна бачити з рисунка, маркер без AR виконує значні обчислення процесора, тому мобільний пристрій часто не здатний забезпечити плавний FPS.

Markerless Augmented Reality розганяється у суспільну свідомість у великій мірі, багато в чому завдяки деяким гігантським корпораціям, що потрапляють за технологію у великій мірі. Це починається з Apple, яка зробила величезний сплеск ще в червні з оголошенням ARKit, який був негайно названий "зміною гри" для AR.

2.2.3 Доповнена реальність на основі проекції.

Доповнена реальність, заснована на проекції, працює шляхом проектування штучного світла на поверхні реального світу. Проеціювання синтетичного світла на фізичні поверхні, в деяких випадках дозволяє взаємодіяти з ним.

Це голограми, які виявляють взаємодію користувача з проекцією за її змінами. Додатки на основі доповненої реальності на основі проекції дозволяють взаємодіяти з людиною, посилаючи світло на поверхню реального світу, а потім відчуваючи взаємодію людини (тобто дотик) цього проецируемого світла.

Виявлення взаємодії користувача здійснюється шляхом диференціації між очікуваною (або відомою) проекцією та зміненою проекцією (викликаною взаємодією користувача).

Проекційний підхід до побудови AP використовує такі фізичні об'єкти, як стіни, книги, гіпсові орнаменти, і будь-який комп'ютер, на який можна оптично проектувати вміст.

А саме, проекція дозволяє використовувати реальні об'єкти як дисплеї. Ми в основному зосереджені на захопленні та використанні 3D-форми поверхні об'єкта, інформація якої дозволяє системі AR / MR враховувати візуальну узгодженість при об'єднанні фізичних і візуалізованих об'єктів. 3D-дані об'єкта можуть бути використані для компенсації спотворень, викликаних різницею між положеннями проекторів і глядача.

Іншою перевагою є можливість генерувати належну візуальну оклюзію між фізичними і віртуальними об'єктами, щоб вони, здавалося, співіснували перед глядачем.

Паралельна обробка для надання AR представлення може бути досягнута відповідно до динамічного фізичного змінення геометрії, наприклад руху деякої частини тіла, та проекція на неї відповідних фізичних і візуалізованих об'єктів AR або MR.

Це можна побачити на рисунку 2.5.

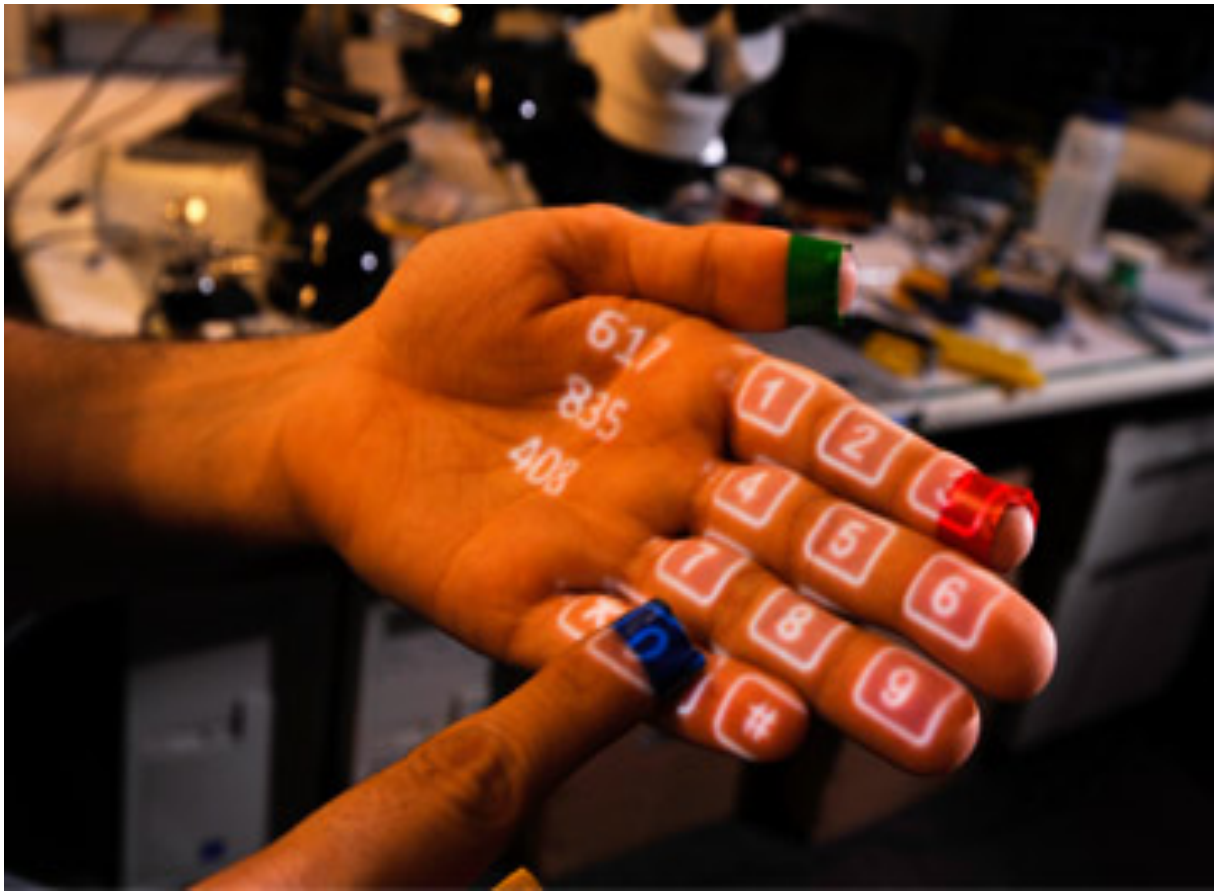

Рисунок 2.5 – Доповнена реальність на проекції

З певних причин цей метод теж неможливо використовувати мобільними пристроями, а отже його не можна вважати достатньо актуальним.

2.2.4 Доповнена реальність на основі накладання.

Доповнена реальність, накладена на суперпозицію, заснована на збільшеній реальності, або частково або повністю замінює оригінальний вигляд об'єкта з новим переглядом того самого об'єкта. У доповнюваній реальності на основі накладання

розпізнавання об'єктів відіграє важливу роль, оскільки програма не може замінити початковий вид доповненим, якщо він не може визначити, що таке об'єкт. У каталозі меблів з розширеною реальністю Ikea можна знайти сильний приклад споживача, наближеного до накладеної реальності. Приклад можна побачити на рисунку 2.6.

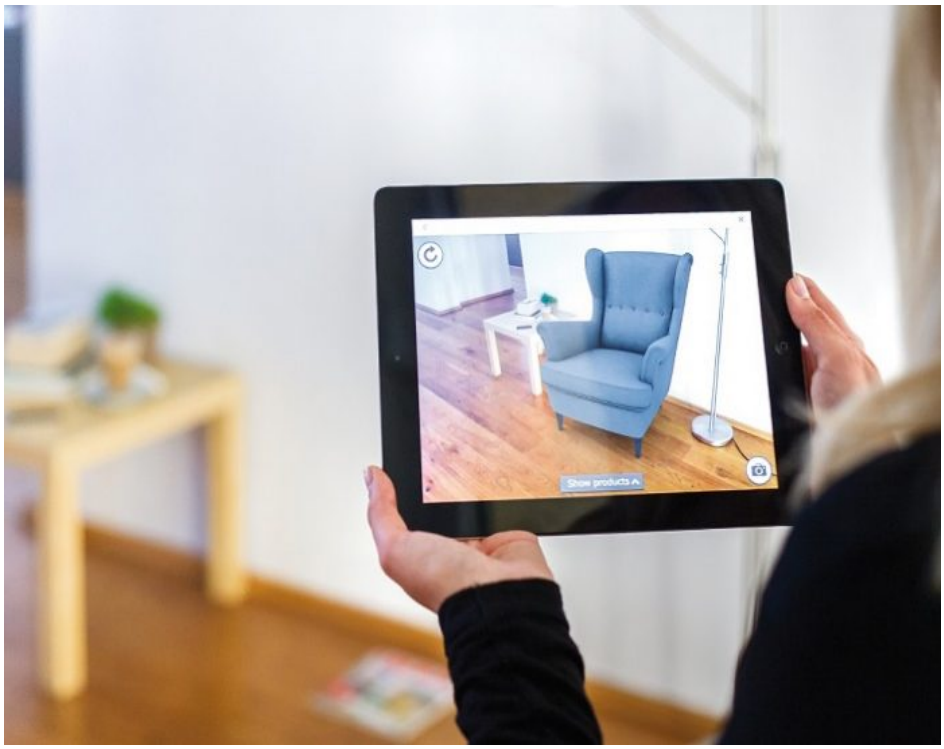

Рисунок 2.6 – Доповнена реальність на основі накладання

Це дійсно один з найважливіших видів технологій, де визнання об'єктів відіграє важливу роль. У цій технології доповнене зображення може замінити оригінальне зображення, частково або повністю. Технологія корисна в медичній сфері. Лікар може ретельно обстежити пацієнта і дати належне лікування. Технологія також є корисною у військовому застосуванні, оскільки AR на основі накладання може запропонувати кілька поглядів на ціль без будь-якого відволікання.

Ця технологія була б корисною для любителів історії, оскільки накладання стародавніх зображень над нинішнім може розкрити багато таємниці про минуле. Це також може допомогти зробити наукове освіту більш цікавим, наприклад, вивчити структуру кістки.

2.3 Аналіз існуючих рішень

Дослідимо існуючі рішення, щоб з’ясувати чи існує вирішення поставлених проблем та щоб дослідити найбільш розповсюджені підходи на ринку.

За останні два роки спостерігався вибух інтересів до технологій VR і AR. Рівні інтересів не демонструють жодних ознак зниження, тому що багато великих технологічних компаній, за чутками, працюють над новими секретними пристроями. Також не дивно, що AR прийшла в Мережу, таким чином представляючи дизайнерам деякі важливі нові області для розгляду для створення контенту.

Майбутнє технологій буде набагато більше, ніж просто розумний, він буде надзвичайно розумним. І ця нова тенденція вже поширена в нашому повсякденному житті.

Від великих корпорацій, таких як Microsoft, які надають технологію AR для військових цілей, і до компанії Apple, яка інвестує не тільки в свої власні рамки AR, але й технології навігації і до різноманітних стартапів та пропрієтарних інструментаріїв.

2.3.1 Огляд існуючих інструментів

Загалом на ринку можна виділити два напрямки розвитку AR інструментарію:

– пропрієтарні засоби, серед яких найкращими є Apple ARKit та Google ARCore;

– засоби з відкритим вихідним кодом, зокрема OpenCV системи.

ARKit: завдяки своїм останнім чіпам A11 і глибоко сприймаючим камерам, Apple зробила крок вперед, випустивши ARKit для розробників iOS. Вона забезпечує постійний досвід AP, який зберігається між сесіями і може бути відновлений пізніше. Крім того, ці сеанси можуть бути спільними для декількох користувачів, кожен зі своїм

пристроєм iOS. Звичайно, ARKit також забезпечує виявлення і відстеження об'єктів і об'єктів: ARKit 1.5 додав підтримку для виявлення 2D зображень, дозволяючи вам запускати досвід AR на основі 2D зображень, таких як плакати, ілюстрації або знаки; ARKit 2 розширює цю підтримку, пропонуючи повне відстеження 2D зображень, так що ви можете включати рухомі об'єкти, такі як коробки для виробів або журнали, у ваш досвід AR; ARKit 2 також додає можливість виявлення відомих 3D-об'єктів, таких як скульптури, іграшки або меблі [9];

ARCore: Google також розробила платформу для створення досвіду розширеної реальності для Android і iOS. Використовуючи різні API, ARCore дозволяє вашому телефону відчувати своє оточення, розуміти світ і взаємодіяти з інформацією. Деякі API доступні для Android та iOS [10], щоб дозволити спільний досвід AR. По суті, ARCore робить дві речі: відстежує позицію мобільного пристрою під час його руху і будує своє власне розуміння реального світу.

ARCore у роботі можна побачити на рисунку 2.7.

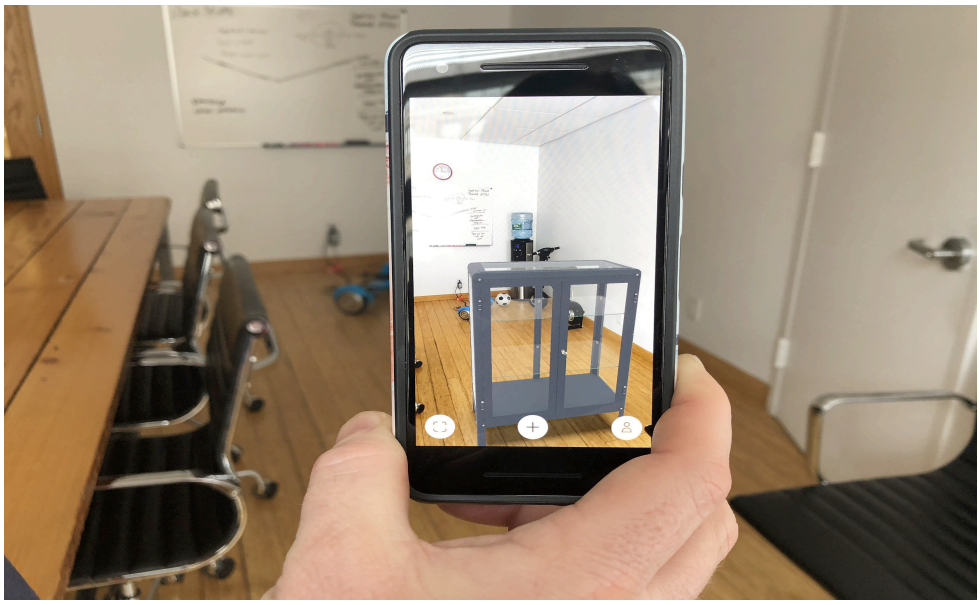

Рисунок 2.7 – Google ARCore

Технологія відстеження руху ARCore використовує камеру телефону для виявлення цікавих точок, які називаються функціями, і відстежує, як ці точки

рухаються з плином часу. Завдяки поєднанню переміщень цих точок і показань з інерційних датчиків телефону, ARCore визначає як положення, так і орієнтацію телефону, коли він рухається по простору. На додаток до ідентифікації ключових точок, ARCore може виявляти плоскі поверхні, такі як стіл або підлога, а також може оцінити середнє освітлення в районі навколо нього. Ці можливості поєднують ARCore, щоб побудувати своє власне розуміння навколишнього світу. ARCore використовує камеру вашого телефону для відстеження руху, дозволяючи йому зрозуміти і відстежувати своє положення щодо світу. Після цього він використовує екологічне розуміння для визначення розміру та розташування всіх типів поверхонь: горизонтальних, вертикальних та кутових поверхонь, таких як земля, журнальний столик або стіни. Нарешті, він виконує оцінку світла для оцінки поточних умов освітлення.

ARToolKit – це була перша широко доступна і відкрита бібліотека, орієнтована на AR.

Одну з перших презентацій ARToolKit можна побачити на рисунку 2.8.

Рисунок 2.8 – ARToolKit

Для того, щоб створити сильну доповнену реальність, він використовує можливості відстеження відео, які обчислюють реальну позицію камери та орієнтацію

щодо квадратних фізичних маркерів або природних маркерів об’єктів у реальному часі. Після того, як реальна позиція камери буде відома, віртуальна камера може бути розташована в одній точці, а моделі 3D комп'ютерної графіки можуть бути намальовані, точно накладені на реальний маркер.

Таким чином, ARToolKit вирішує дві ключові проблеми в розширеній реальності: відстеження точок зору та взаємодії віртуальних об'єктів.

На жаль, при усій перспективності технології, вона застаріла та дуже важка у використанні порівняно з конкурентами;

Wikitude є постачальником мобільних додатків реальності (AR), що базується в Зальцбурзі, Австрія.

Приклад інструментарія WikiTude можна побачити на рисунку 2.9.

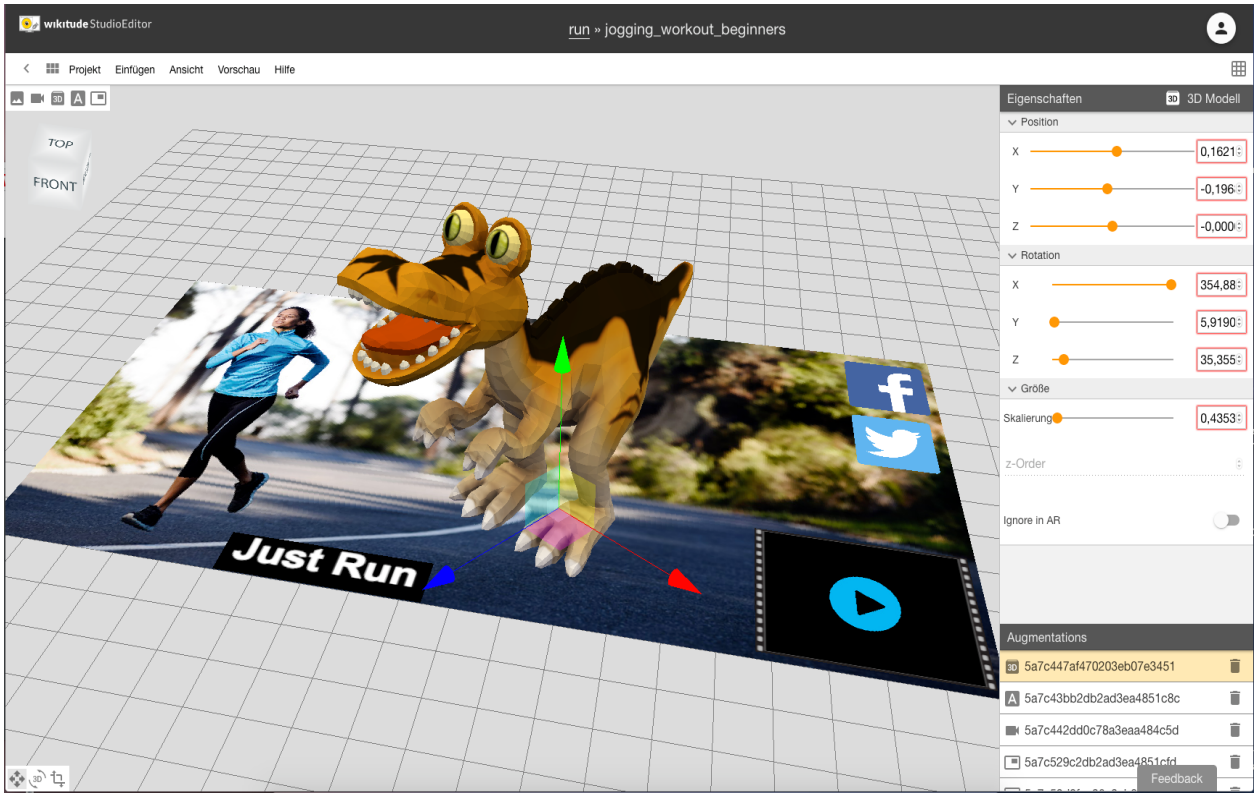

Рисунок 2.9 – WikiTude

Заснована у 2008 році, Wikitude спочатку зосереджувалася на забезпеченні досвіду розширеної реальності на основі розташування за допомогою програми Wikitude World

Browser App. У 2012 році компанія реструктуризувала свою пропозицію, запустивши програму Wikitude SDK, основу розробки, що використовує технології розпізнавання та відстеження зображень, а також технології геолокації.

Основний продукт компанії – Wikitude SDK. Вперше запущений у жовтні 2008 року, SDK включає в себе розпізнавання і відстеження зображень, рендеринг 3D-моделі, накладання відео, розташування на основі AR. У 2017 році Wikitude запустила свою технологію SLAM (Simultaneous Localization And Mapping), яка дозволяє розпізнавання та відстеження об'єктів, а також миттєве відстеження без маркер.

Крос-платформенний SDK доступний для операційних систем Android, iOS і Windows, оптимізований для декількох смарт-пристроїв для окулярів.

Додаток Wikitude був першим загальнодоступним додатком, який використовував підхід, заснований на розташуванні, до розширеної реальності.

Для даної роботи було обрано Apple ARKit як найбільш стабільний, доступний та простий інструмент розробки AR-застосуваннь.

2.3.2. Засіб розробки доповненої реальності ARKit

Принцип роботи ARKit як маркерного AR засобу дуже простий. Фактично ARKit це три фреймворка Apple в одному:

– Vision – фреймворк обробки візуальної інформації;

– CoreML – фреймворк машинного навчання, що дозволяє визначати зображення та об’єкти як маркери;

– CoreMotion – фреймворк що відповідає за взаємодію з сенсорами пристрою, використовуються для трекінгу об’єктів.

Програмне забезпечення аналізує маркер і створює віртуальне накладання зображення на екрані мобільного телефону, прив'язане до положення камери. Це

означає, що програма працює з камерою для інтерпретації кутів і відстані мобільного телефону від маркера.

Схему роботи цих компонентів можна побачити на рисунку 2.10.

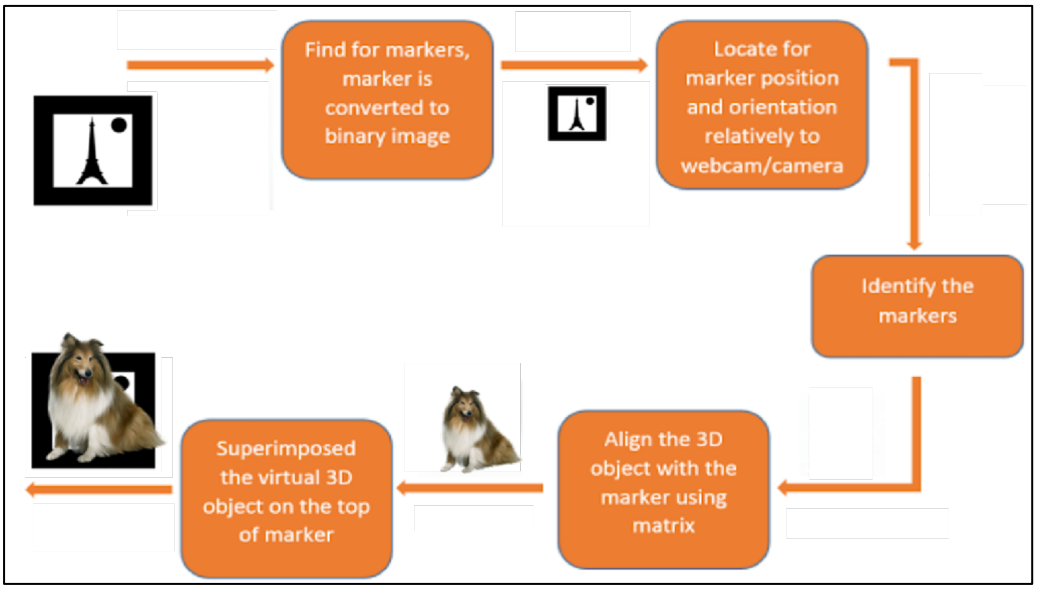

Рисунок 2.10 – Схема роботи ARKit

Маркером може бути що завгодно, друк 2D зображень, об’єкти реального світу, як автомобіль або людське обличчя, загальновідомий як ImageTarget. Після того, як додаток розширеної реальності розпізнає ціль за допомогою камери, він обробляє зображення і збільшує віртуальний вміст, як 3D-модель, відео, зображення або інші засоби масової інформації. Наприклад, ви можете побачити, що плакат фільму оживає і зіграти трейлер фільму. Поки ви дивитеся на плакат через «вікно» дисплея, ви можете бачити розширену реальність замість простої старої типової реальності. Використовуючи інтелектуальні алгоритми та інші датчики, такі як акселерометри та гіроскопи, пристрій може утримувати доповнені елементи у відповідності з зображенням реального світу. Використовуючи мобільний додаток, камера мобільного телефону визначає та інтерпретує маркер (ImageTarget).

3 ОПТИМІЗАЦІЯ НЕЙРОННИХ МЕРЕЖ ТА ПРОСТОРОВЕ СПРИЙНЯТТЯ AR.

3.1 Математична модель нейронних мереж для розпізнавання образів

Для класифікації зображень використовується згортуюча нейронна мережа, яку необхідно тренувати на запропонованому датасеті. Зазвичай це тривіальна задача, однак оскільки ця нейронная мережа буде використовуватися мобільним пристроєм, гостро стоїть проблема розміру цієї мережі. Для вирішення цієї проблеми необхідно врахувати математичну модель мережя для її оптимізації. Обчислювальна складність для сучасних моделей як для навчання, так і для висновків є надзвичайно високою, що вимагає декількох графічних процесорів для навчання протягом сотень годин.

Якщо розглядати шари нейронної мережі як матрицю, що складається з ядер як параметрів, то можна досягти зменшення розміру нейронної мережі шляхом зменшення кількості параметрів матриці. Цього можна досягти шляхом факторизації згортань, а саме зменшення кількості з'єднань / параметрів без зменшення ефективності мережі. Розглянемо вхідні дані I в

$$R^h \times w \times c \tag{3.1}$$

де h – висота,

w –ширина,

c – кількість каналів вхідних характеристичних карт,

Та згорткове ядро K.

$$R^{k \times k \times c \times n} \tag{3.2}$$

де k – розмір згорткового ядра,

n є число ядер.

Робота стандартного згортаного шару

$$O \in R^{h \times w \times n} = K * I \tag{3.3}$$

представлена наступною формулою:

$$O(y, x, j) = \sum_{i=1}^{c} \sum_{u=1}^{k} \sum_{v=1}^{k} K(u, v, i, j) I(y + u - 1, x + v - 1, i) \tag{3.4}$$

Складність згорткового шару за кількістю множень, дорівнює

$$nck^2hw \tag{3.5}$$

Ілюстрація роботи стандартного згорткового шару зображена на рисунку 3.1.

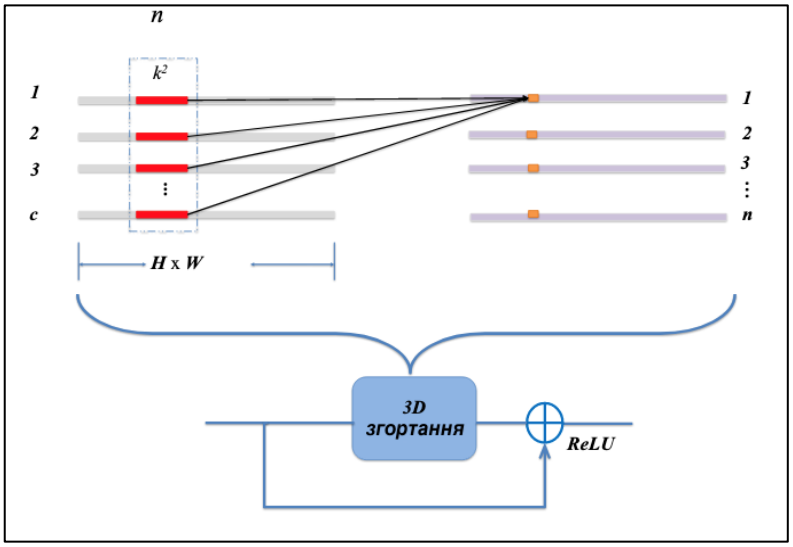

Рисунок 3.1 – Стандартний згортковий шар з залишковим з'єднанням

Оскільки складність квадратична з розміром ядра, в самих останніх моделях CNN, розмір ядра обмежується до 3 × 3 для контролю загального часу роботи.

3.1.1 Факторизація згорткових шарів

У стандартних згорткових шарах вихідні особливості виробляються шляхом згортання групи 3D ядер з вхідними ознаками вздовж просторових вимірів. Таку операцію 3D-згортки можна розглядати як комбінацію 2D просторової згортки всередині кожного каналу (внутрішньоканальна згортка) і лінійної проекції через канали. Для кожного вихідного каналу на кожному вхідному каналі виконується просторова згортка.

Просторова згортка здатна захоплювати локальну структурну інформацію, тоді як лінійна проекція перетворює простір ознак для вивчення необхідної нелінійності в шарах нейронів. Коли кількість вхідних і вихідних каналів велика, зазвичай сотні, такий 3D-згортковий шар вимагає величезної кількості обчислень.

Природна ідея полягає в тому, що 2D просторова згортка і проекція лінійних каналів можуть бути розформовані і виконані окремо.

4D згорткове ядро K може бути розкладено на основні фільтри

$$\mathrm{B} \in R^{k \times k \times b \times c} \tag{3.6}$$

і тензор лінійної проекції

$$P \in R^{b \times c \times n} \tag{3.7}$$

де b – число основ,

$$K(u, v, i, j) = \sum_{t=1}^{b} B(u, v, t, i) P(t, i, j) \tag{3.8}$$

Для кожного вхідного каналу згорткове ядро розкладається вздовж просторових вимірів. Операція згорткового шару в рівнянні перетворюється в

$$J(y,x,t,i) = \sum_{u=1}^{k} \sum_{v=1}^{k} B(u,v,t,i) I(y+u-1, x+v-1, i) \quad (3.9)$$

$$O(y,x,j) = \sum_{i=1}^{c} \sum_{t=1}^{b} P(t,i,j) J(y,x,t,i) \quad (3.10)$$

Кожен канал вхідної карти уявлення спочатку згортається за допомогою b 2D фільтрів (баз), генеруючи проміжну карту J, з якої число каналів становить b разів на вхід. Потім використовується лінійна проекція каналу для проеціювання проміжної карти характеристик на виході.

Вирівнювання цих двох операцій (2D просторова згортання і лінійна проекція) дає нам більшу свободу моделювання за допомогою налаштування кількості 2D баз b і розміру ядра k .

Розглядаючи використовану Inception v3 модель, процес факторизації одного з її модулей зображено на рисунку 3.2.

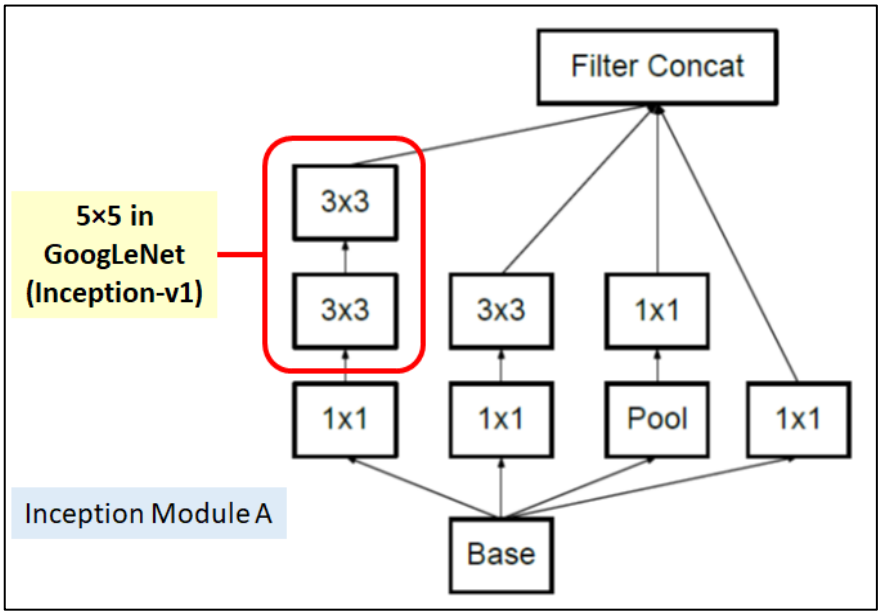

Рисунок 3.2 – Inception Module A з використанням факторизації

Складність такого шару – bck2hw + bcnhw. Перший член відповідає 2D просторовій згортці, а другий – лінійній проекції. Як правило, k2 є набагато меншим,

ніж n, і більшість обчислень споживається лінійною проекцією, яка є лінійною з числом базисного фільтра b. При b = k2 це еквівалентно повнорозмірному розкладанню згорткового ядра для кожного вхідного каналу вздовж просторових вимірів.

Складність лінійної проекції така ж, як і стандартний згортковий шар; Завдяки присвоєнню b <k2 складність нижча, ніж стандартний згортковий шар у низькому порядку.

Ключове спостереження полягає в тому, що замість згортання b 2D фільтрів (баз) з кожним вхідним каналом одночасно і виконують над ними лінійну проекцію, ми можемо послідовно організувати звивини, інтерліруючи з лінійною проекцією.

Вищезгаданий згортковий шар з b 2D фільтрами може бути перетворений в каркас, який має b шари. У кожному шарі кожен вхідний канал спочатку згортається з одним 2D фільтром (основа), тоді лінійна проекція застосовується до всіх каналів.

Таким чином, кількість каналів підтримується так само, як і вхідні дані по всіх слоях. Зауважимо, що така послідовна організація зберігає однакову обчислювальну складність, збільшуючи глибину та можливості навчання. Таким чином, складність кожного шару буде

$$ck^2hw + cnhw \tag{3.11}$$

Що значно пришвидшить роботу моделі та зменшить її розмір.

3.2 Математична модель оточення доповненої реальності

При будуванні доповненої реальності головною задачей є сприйняття сенсорами пристрої оточення та будування його моделі на основі отриманих даних. Хоча у різних системах підходу можуть відрізнятися деталями, основний принцип залишається

незмінним. Мета розширеної реальності полягає в тому, щоб забезпечити сприйняття комп'ютером простору з розумінням людини. У комп'ютерній науці простір просто метафора для загальноприйнятих і науково обгрунтованих концепцій простору, часу і матерії. Комп'ютерне розуміння простору – це не що інше, як математично визначене 3D представлення об'єктів, місцеположення та матерії. Його можна просто зрозуміти за допомогою систем координат без необхідності заплутати жаргон, як гіпер-нерухомість або альтернативні всесвіти. Хоча це, безумовно, цікаві думки експериментів [11].

Те, що сприймається як розташування об'єктів у навколишньому середовищі, є реконструкція світлових моделей на сітківці. Візуальний простір у комп'ютерній графіці можна визначити як сприйманий простір або візуальну сцену 3D-віртуального простору, що відчувається учасником.Віртуальний простір, в якому існує об'єкт, називається простором об'єкта. Це прямий аналог візуального простору.

Три типи різних математичних систем координат використовуються для компонування та програмування додатків віртуальної та розширеної реальності:

– декартові координати;

– сферичні полярні координати;

– циліндричні координати.

Декартові системи координат використовуються в основному за рахунок їх простоти, і більшість віртуальних просторів визначаються нею. Система координат на основі x-y-z є точною для визначення розташування 3D-об'єктів у віртуальному просторі. Три координатні площини перпендикулярні один одному. Відстані та місцеположення визначаються від точки виникнення, яка є точкою, де три площини перетинаються один з одним. Ця система використовується в основному для визначення візуальних координат 3D-об'єктів.

Декартова система визначає положення 3D-об'єктів часто по відношенню до початкової точки. Система розміщення сферичних полярних координат використовується для визначення об'єктів і особливостей по відношенню до положення користувачів. Ця система використовується в основному для відображення

віртуального джерела звуку або відображення сферичного відео у випадку занурення VR першої особи. Сферична система координат заснована на перпендикулярних площинах, що розділяють сферу і складається з трьох елементів: азимут, висота і відстань. Азимут – це кут від початкової точки в горизонтальній / заземленій площині, а висота – кут у вертикальній площині. Відстань – це величина або діапазон від початку, що можна побачити на рисунку 3.3.

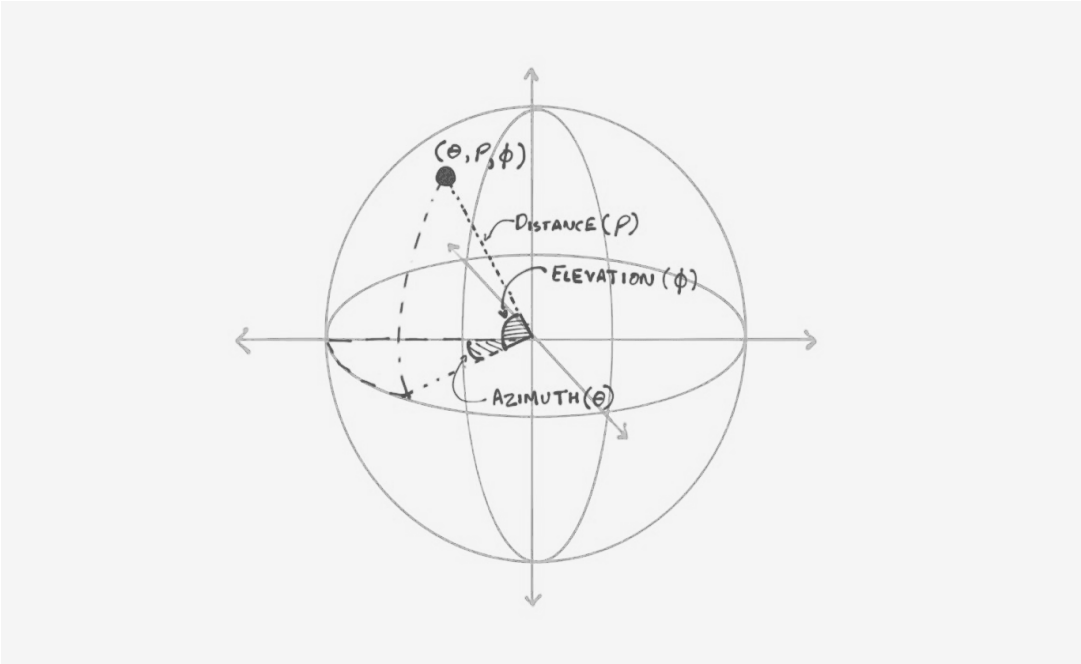

Рисунок 3.3 – Сферичні декартові координати

Циліндрична система система використовується в основному у програмах VR для перегляду панорам 360 градусів. Циліндрична система дає можливість точного відображення та вирівнювання нерухомих зображень для перекриття для зшивання країв у панорам. Система складається з центральної опорної осі (L) з точкою початку (O). Радіальна відстань (ρ) визначається з походження (O). Кутову координату (is) визначають для радіальної відстані (ρ) разом з висотою (z). Хоча ця система хороша для сценаріїв, які вимагають ротаційної симетрії, вона обмежена з точки зору її вертикального вигляду.

Полярні координати можна назвати радіальною відстанню або радіусом і кутом або азимутом, відповідно. Третю координату можна назвати висотою (якщо площина відліку горизонтальна). Лінію, перпендикулярну до площини відліку, яка проходить через центр координат можна назвати циліндричною віссю.

Далі зображення осей циліндричних координат на рисунку 3.4.

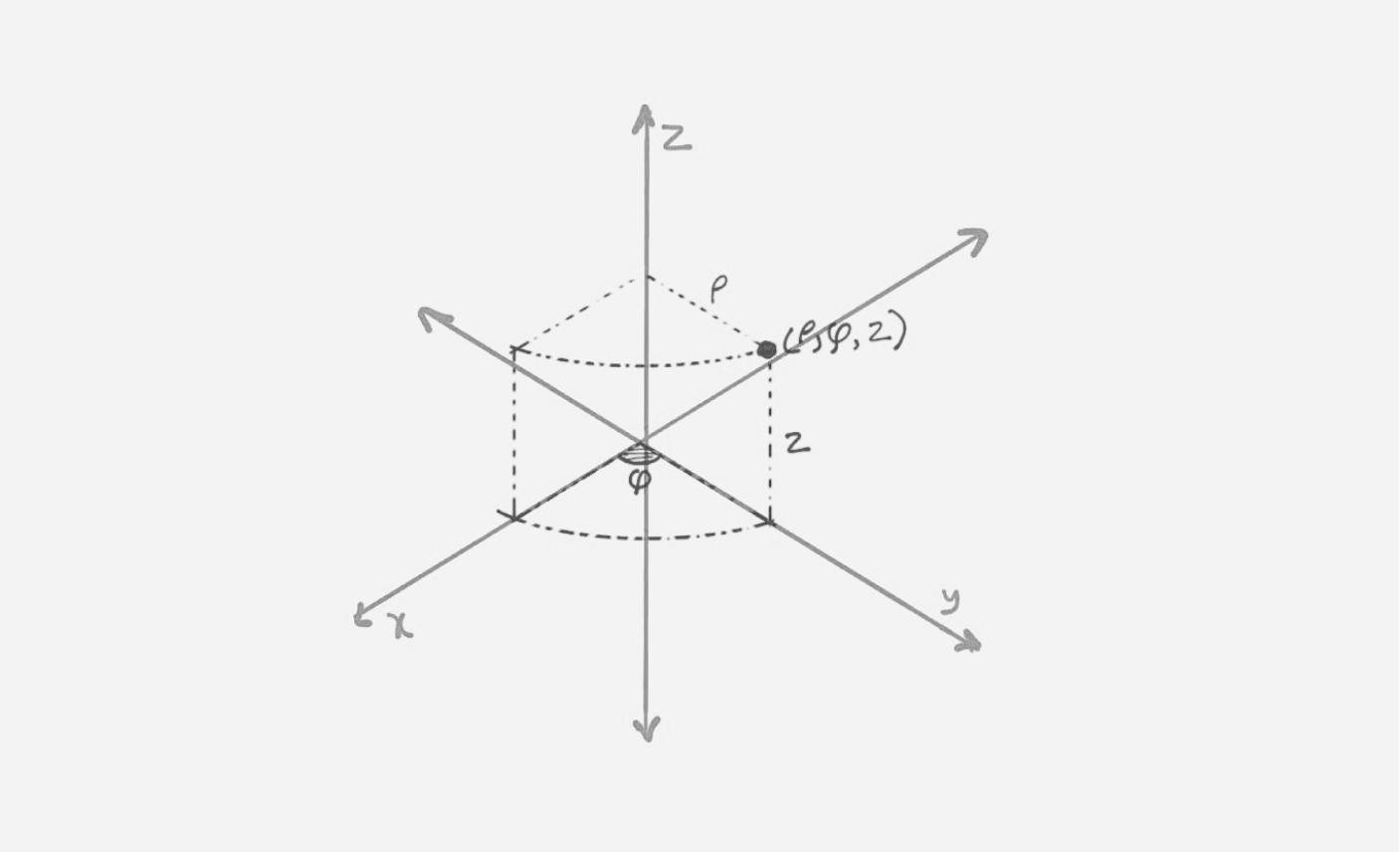

Рисунок 3.4 – Циліндрична система координат

Необхідно визначити орієнтацію і обертання точок зору та об'єктів користувача разом з їх положенням у віртуальному просторі. Знання цієї інформації особливо важливо при відстеженні, де користувач шукає або знає орієнтацію віртуальних об'єктів щодо зорового простору.

Також у віртуальній і доповненій реальності загальноприйнято визначати орієнтацію і обертання з трьома незалежними значеннями.

Вони називаються roll (x), pitch (y) і yaw (z) і відомі як кути Тейта-Баяна. Поєднання положення (x-y-z) і орієнтації (roll-pitch-yaw) називається шістьма осями свободи (6 DOF).

Схематичне зображення осей розташування на рисунку 3.5.

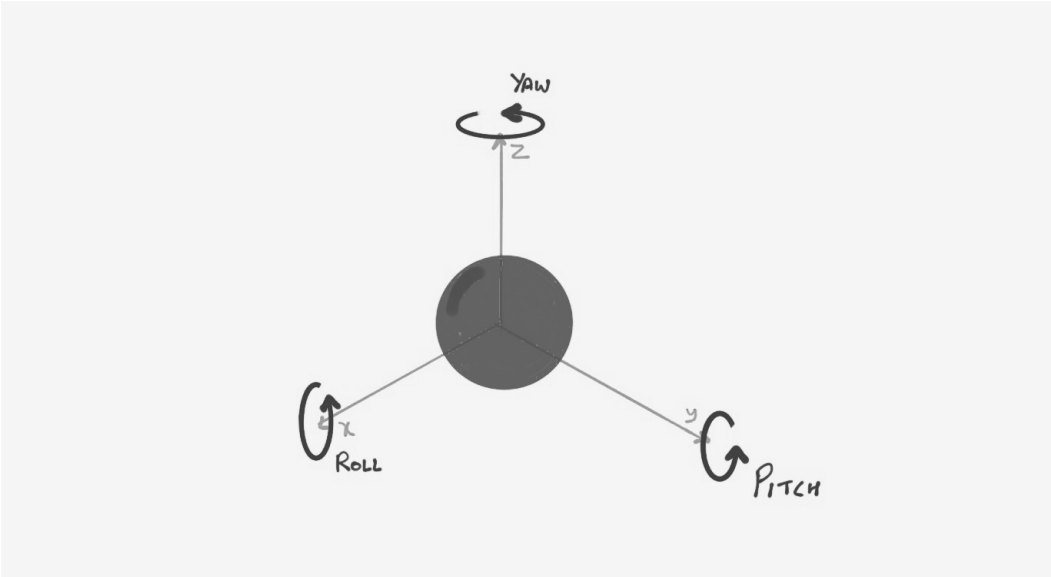

Рисунок 3.5 – Кути Тейта-Баяна

Однак цікавим питанням є сприйняття відстаней органами доповненої реальності, для цього необхідно провести обчислення у реальному світі, використовуючи сферичні координати.Радіус Землі в екваторі становить 3 956,547 кілометрів. Позначимо його через s = 3,956,547 кілометрів.

Перетворення широт в кілометри буде простішим, оскільки ми можемо наблизити широтні лінії до дуги кола. Ми знаємо, як обчислити окружність кола (2πr), і ми також знаємо, що коло має 360∘. Таким чином, якщо ми розділимо окружність Землі на 360 градусів, ми будемо мати наближення до кілометрів на градус широти t. Точність цього наближення є достатньою для нашого застосування.

$$t = \frac{2\pi s}{360^\circ} \tag{3.12}$$

Розрахунок кілометрів на градус довготи g буде не таким простим. Пересування на градус довжини поруч з екватором означає прогулянку на більшу відстань, ніж прогулянка на один градус довготи ближче до полюсів.

Таким чином, отримаємо що кожний градус довготи дорівнює

$$g = t\cos\left(a_y \frac{\pi}{180^\circ}\right) \tag{3.13}$$

де g – кілометри на ступінь довжини 180∘

Наш користувач стоїть біля:

$$a = \begin{bmatrix} a_x \\ a_y \end{bmatrix} = \begin{bmatrix} \text{Широта пристрою} \\ \text{Довгота пристрою} \end{bmatrix} \tag{3.14}$$

Користувач буде спрямовувати пристрій до напрямку, і цей напрямок буде робити кут α з істинним північчю (заголовком).

Тепер цей кут α буде виміряний по лініях широти, які є вертикальними, і еквівалентні осі y, або будь-якій з його паралелей у декартовій площині. Однак у декартовій площині звичайною практикою є вимірювання кутів на осі x, тому кут α необхідно перетворити на кут ψ, що дасть нам напрмок відносно горизонтальної лінії, паралельної осі x.

$$\psi = \frac{\pi}{2} - a \tag{3.15}$$

Оскільки тепер ми знаємо, як перетворити з градусів на кілометри, і навпаки, ми можемо обчислити точки b і c з урахуванням відстані r. Враховуючи те, що ми маємо набір місць, кожне з яких має власну довготу і широту. Наше завдання полягає в тому, щоб визначити, які з них потраплятимуть в область інтересу, тому бути видимими і мати відповідну анотацію на екрані.

Маємо, що проекція вектора ap⃗ на вектор ab⃗ може бути обчислена за формулою:

$$proj \underset{ab}{\rightarrow} ap = |ap| \cos \omega \tag{3.16}$$

Крім того, вектор між ap⃗ і ab⃗ є:

$$\overrightarrow{ab} \cdot \overrightarrow{ap} = |\overrightarrow{ab}||\overrightarrow{ap}| \cos \omega \tag{3.17}$$

$$proj_{\overrightarrow{ab}} \overrightarrow{ap} = \frac{\overrightarrow{ap} \cdot \overrightarrow{ab}}{||\overrightarrow{ab}||} \tag{3.18}$$

Потрібно виразити $\text{proj}_{\overrightarrow{ab}} \overrightarrow{ap}$ як координату вектора ab, або іншими словами,обчислити власне значення для нього. Це досягається діленням його на норму ab. Назвемо цю власну величину λ.

$$\lambda = \frac{proj_{ab} ap}{|ab|} \Rightarrow \lambda = \frac{ap \cdot ab}{|ab|^2} \tag{3.19}$$

Тепер нам потрібно обчислити проекцію вектора ap на вектор ac:

$$proj_{\vec{ac}} ap = \frac{ap \cdot ac}{|ac|} \tag{3.20}$$

Розділення проекції на норму вектора ac⃗ дало нам власне значення і дозволяє виразити його координати в термінах ac⃗. Назвемо це власне значення σ.

$$\sigma = \frac{proj_{ac} \vec{ap}}{|ac|} \Rightarrow \sigma = \frac{ap \cdot ac}{|ac|^2} \tag{3.21}$$

При λ і σ можна перейти до останнього етапу визначення того, чи міститься мітка в межах видимої області, що представляє інтерес.

Дуга окружності, що визначає межу відстані, може бути обчислена за допомогою теореми Піфагора:

$$\left(\vec{\lambda}ab\right)^2 + (\sigma a\vec{c})^2 = r^2 \tag{3.22}$$

Однак довжина векторів $ab^{\rightarrow}$ і $ac^{\rightarrow}$ мають однакову довжину і дорівнюють довжині радіус r. Таким чином, для спрощення рівняння можна розділити обидві сторони на r2.

$$\lambda^2 \frac{\cancel{ab^2}}{\cancel{r^2}} + \sigma^2 \frac{\cancel{ac^2}}{\cancel{r^2}} = \frac{\cancel{r^2}}{\cancel{r^2}} \tag{3.23}$$

Залишаючи нас з:

$$\lambda^2 + \sigma^2 = 1 \tag{3.24}$$

Тепер ми можемо визначити, чи мітка місця p знаходиться в межах інтересу, тому є видимою. Власні значення λ і σ повинні бути позитивними числами (інакше позначка місця буде розташована за спостерігачем), а при об'єднанні, використовуючи теорему Піфагора, вони повинні бути меншими або рівними 1, тому вони не далі, ніж межа задається радіусом сегмента дуги. p (λ, σ) дорівнює:

$$\forall(\lambda > 0) \wedge (\sigma > 0) \wedge (\lambda^2 + \sigma^2 \leq 1) \tag{3.24}$$

Результати справедливі для видимого спектра, та всі інші значення для невидимого.

Наступним кроком є використання отриманого результату у застосунку.

Більш ранні методи AR спиралися на безліч датчиків на додаток до камери. Бібліотеки програмного забезпечення, такі як OpenCV, Vuforia, ARCore, ARKit, MRKit.

4 ПРОЕЦІЮВАННЯ ДИНАМІЧНОГО ВЕБ-КОНТЕНТУ У ДОПОВНЕНІЙ РЕАЛЬНОСТІ ЗАСОБАМИ РОЗРОБЛЕНОЇ ПРОГРАМНОЇ СИСТЕМИ

4.1 Вибір і опис середовища розробки

В якості основного середовища було обрано продукт Xcode, лінійку продуктів Apple, що включає інтегровану середу розробки програмного забезпечення та ряд інших інструментів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, включаючи підтримку технології UIKit, ARKit, CoreML, Vision, для розробки програмного забезпечення для MacOS, iOS, iPadOS, watchOS і tvOS [12].

Основним додатком комплекту є інтегрована середовище розробки (IDE), також названа Xcode. Набір Xcode включає більшу частину документації розробника Apple, а також вбудований інтерфейс Builder – додаток, що використовується для побудови графічних інтерфейсів користувача. До Xcode 4.1, пакет Xcode містив модифіковану версію GNU Compiler Collection. В Xcode 3.1 до Xcode 4.6.3, він включав компілятор LLVM-GCC, з передніми кінцями з колекції компіляторів GNU і генератора коду на основі LLVM. У Xcode 3.2 і пізніших версіях він включав компілятор Clang C / C ++ Objective-C, з новими письмовими кінцями і генератором коду на основі LLVM, а також статичним аналізатором Clang. Починаючи з Xcode 4.2, компілятор Clang став типовим Компілятор, починаючи з Xcode 5.0, Clang був єдиним наданим компілятором.

Всі інструменти встановлені на macOS, це серія графічних операційних систем, розроблених і реалізованих компанією Apple Inc. з 2001 року. Це основна операційна система для комп'ютерів Apple Mac.

На ринку настільних комп'ютерів, ноутбуків і домашніх комп'ютерів, а також за допомогою веб-застосувань, він є другим найбільш широко використовуваним настільним ОС після Microsoft Windows.

macOS – друга велика серія операційних систем Macintosh. Перший розмовно називається "класичний" Mac OS, який був введений в 1984 році, і остаточний випуск

якого був Mac OS 9 в 1999 році. Перша десктопна версія, Mac OS X 10.0, була випущена в березні 2001 року, з її першим. оновлення, 10.1, прибуває пізніше цього року. Після цього Apple почала називати свої випуски після великих кішок, які тривали до OS X 10.8 Mountain Lion. Починаючи з OS X 10.9 Mavericks, релізи отримали назву за місцем розташування в Каліфорнії. Apple скоротила назву на "OS X" у 2012 році, а потім змінила її на "macOS" у 2016 році, прийнявши номенклатуру, яку вони використовували для своїх інших операційних систем, iOS, watchOS і tvOS. Остання версія macOS Mojave, яка була публічно випущена у вересні 2018 року та є найбільш продвинутою операційною системою для розробки AR застосунків

У період з 1999 по 2009 рік Apple продала окрему серію операційних систем Mac OS X Server. Початкова версія, Mac OS X Server 1.0, була випущена в 1999 році з інтерфейсом користувача, схожим на Mac OS 8.5. Після цього нові версії були введені одночасно з настільною версією Mac OS X. Починаючи з Mac OS X 10.7 Lion, функції сервера були доступні як окремий пакет на Mac App Store, на Mac OS X server є можливість розробляти веб-застосунки з використанням мови програмування Swift.

iOS (раніше iPhone OS) – це мобільна операційна система, створена і розроблена компанією Apple Inc. виключно для апаратного забезпечення. Це операційна система, яка в даний час керує багатьма мобільними пристроями компанії, включаючи iPhone, iPad і iPod Touch. Це друга за популярністю мобільна операційна система в усьому світі після Android.

Спочатку презентований у 2007 році для iPhone, iOS був розширений на підтримку інших пристроїв Apple, таких як iPod Touch (вересень 2007 року) і iPad (січень 2010 року). Станом на березень 2018 року, Apple App Store містить більше 2,1 мільйона iOS-додатків, з яких 1 мільйон є рідними для iPad. Ці мобільні програми колективно завантажуються більше 130 мільярдів разів.

Інтерфейс користувача iOS заснований на безпосередній маніпуляції, використовуючи жести з мультитач. Елементи управління інтерфейсом складаються з повзунків, перемикачів і кнопок. Взаємодія з операційною системою включає в себе жести, такі як удари, натискання, затискання та зворотний поштовх, кожна з яких має

певні визначення в контексті операційної системи iOS та її мультитач-інтерфейсу. Внутрішні акселерометри використовуються деякими додатками для реагування на струшування пристрою (один загальний результат – команда скасування) або обертання в трьох вимірах (один загальний результат – перемикання між портретом і пейзажем). Компанія Apple отримала високу оцінку за включення до iOS повноцінних функцій доступності, що дозволяє користувачам з вадами зору та слуху правильно використовувати її продукти.

Основні версії iOS випускаються щорічно. Поточна версія, iOS 12, була випущена 17 вересня 2018 року. Вона доступна для всіх пристроїв iOS з 64-розрядними процесорами; iPhone 5S і пізніші моделі iPhone, iPad (2017), iPad Air, а потім моделі Air Air, всі моделі iPad Pro, iPad Mini 2 і пізніші моделі iPad Mini, а також IPod Touch шостого покоління. На всіх останніх iOS-пристроях iOS регулярно перевіряє наявність оновлення, а якщо є, підкаже користувачеві дозволити автоматичну інсталяцію.

Прийшовши восени 2019 року, iOS 13 буде випущений для громадськості, представляючи незначні налаштування інтерфейсу користувача, а також багато нових функцій, таких як оновлений додаток Reminders, клавіатура swipe, розширене додаток Photos і новий темний режим. iOS 13 завершиться підтримкою декількох пристроїв iOS, включаючи iPhone 5s, iPod Touch (6-е покоління), iPhone 6 і iPhone 6 Plus, які все ще складають понад 10% всіх пристроїв iOS.

Випуск iOS 13 буде виключно для iPhone і iPod touch, оскільки iPad буде відгалужуватися в нову операційну систему на базі iOS під назвою iPadOS. Крім того, iPadOS також зменшить підтримку iPad, які мають менше 2 Гб оперативної пам'яті, такі як iPad Air, iPad mini 2 і iPad mini 3.

Підтримку ARKit було введено з iOS 11, ARKit 2 було випущено на поточну версію iOS 12, наступна версія iOS 13, матиме підтримку ARKit 3.0, найбільш розвинену SDK для розробки AR застосунків.

4.2 Вибір мови програмування і шаблону проектування

Пакети для кодування нейронних мереж існують у більшості популярних мов програмування, включаючи Matlab, Octave, C, C ++, C #, Ruby, Perl, Java, Javascript, і PHP, однак найбільш поширеною мовою для розробки комплексних нейронних мереж є мова програмування Python.

Python – це мова програмування високого рівня, призначена для читаності коду та ефективного синтаксису, що дозволяє виражати поняття в меншій кількості рядків коду, ніж мови C ++ або Java. Двома бібліотеками Python, що мають особливе значення для створення нейронних мереж, є NumPy і Theano. NumPy – це пакет Python, який містить безліч інструментів для наукових обчислень, включаючи об'єкт N-масиву, функції мовлення, лінійну алгебру та можливості випадкових чисел. NumPy пропонує ефективний багатовимірний контейнер з загальними даними з довільним визначенням типів даних, що дозволяє легко інтегруватися з широким спектром баз даних. Функції NumPy слугують фундаментальними будівельними блоками для наукових обчислень, і багато з його особливостей є невід'ємною частиною розробки нейронних мереж у Python.

NumPy – це бібліотечний пакет для мови програмування Python, який можна використовувати для розробки нейронних мереж, серед інших наукових обчислювальних завдань. Theano – бібліотека Python, яка визначає, оптимізує і оцінює математичні вирази, інтегрується з NumPy. Він має модульні тестування та функції самостійної перевірки для виявлення помилок; ефективна символічна диференціація; динамічне генерування коду C для швидкої оцінки експресії; і здатний використовувати блок обробки графіки (GPU) прозоро для дуже швидких обчислень, які інтенсивно використовують дані. Він може моделювати обчислення у вигляді графіків і використовувати ланцюгове правило для обчислення складних градієнтів. Це особливо застосовне до нейронних мереж, оскільки вони легко виражаються у вигляді графіків обчислень. Використовуючи бібліотеку Theano, нейронні мережі можуть бути

написані більш стисло і виконуватися набагато швидше, особливо якщо використовується GPU.

У екосистемі Apple використовуються дві мови програмування: більш старий і традиційний Objective-C, та Swift.

Objective-C – це універсальна, об'єктно-орієнтована мова програмування, яка додає повідомлення в стилі Smalltalk мовою програмування C.

Це була основна мова програмування, підтримувана Apple для операційних систем macOS, iOS і iPadOS, та їх відповідних інтерфейсів прикладного програмування (API) Cocoa і Cocoa Touch до введення Swift. Мова програмування Objective-C була розроблена на початку 1980-х років. Він був обраний як основна мова, яку використовує NeXT для своєї операційної системи NeXTSTEP, з якої виведені macOS і iOS. Портативні програми Objective-C, які не використовують бібліотеки Cocoa або Cocoa Touch, або ті, які використовують частини, які можуть бути перенесені або перевиконані для інших систем, також можуть бути зібрані для будь-якої системи, що підтримується GNU Compiler Collection (GCC) або Clang. Файли програмної реалізації Objective-C, як правило, мають розширення .m filename, тоді як файли "header / interface" Objective-C мають розширення .h, так само як і C-файли заголовків.

Оскільки Objective-C було виведено з C, було випущено Objective-C++ для сумісності з C++ відповідно. Файли Objective-C ++ позначаються розширенням .mm.Objective-C ++Objective-C ++ – це мовний варіант, прийнятий інтерфейсом до GNU Compiler Collection і Clang, який може компілювати вихідні файли, які використовують комбінацію синтаксису C ++ і Objective-C. Objective-C ++ додає до C ++ розширення, які додає Objective-C до C.

Оскільки нічого не робиться для уніфікації семантики за різними функціями мови, застосовуються певні обмеження: Клас C ++ не може бути отриманий з класу Objective-C і навпаки.

Однак Objective-C є потужним інструментом для використання більш низькорівневих API написаних на C та C++, наприклад OpenCV, що є дуже потужним інструментом Computer Vision.

Однак основною мовою розробки AR застосунку було обрано Swift як найбільш сучасну та потужну.

Swift – це універсальна, складна мова програмування, розроблена компанією Apple для iOS, macOS, watchOS, tvOS, і Linux. Swift розроблений для роботи з Cocoa та Cocoa Touch продуктів Apple. Вона побудована з відкритим вихідним кодом LLVM компілятора і була включена в Xcode з версії 6, випущеної в 2014 році. На платформах Apple вона використовує бібліотеку виконання Objective-C, яка дозволяє запускати код C, Objective-C, C ++ і Swift. в межах однієї програми.

Apple призначила Swift підтримати багато основних концепцій, пов'язаних з Objective-C, особливо динамічну диспетчеризацію, широко розповсюджене пізнє зв'язування, розширюване програмування та подібні функції, але "безпечнішим" способом, що полегшує уловлювання програмних помилок; Swift має функції, що стосуються деяких поширених помилок, таких як нульовий покажчик.

Таким чином Swift можна використовувати разом із Objective-C, що в свою чергу дає можливість користуватися Objective-C++ та C++ кодом, що є необхідним для повноцінної розробки комплексної AR системи з використанням Computer Vision, а також повноцінну розробку web-сервера для зберігання та розповсюдження тренованих моделей та контенту.

Зазвичай для розробки мобільних застосунків використовуються архітектури сімейства MVX, де найбільш розповсюдженим є MVC (model-view-controller), або більш продвинутий MVP (model-view-presenter), однак ці архітектури занадто не підходять для комплексної задачі адаптації AR даних, тому у мобільному компоненті системи вирішено використати архітектурний шаблон Model-view-viewmodel (MVVM) як найбільш відповідну для розділення відображення (View) у вигляді AR сутностей, сховища моделей нейронних мереж та веб-контенту (Model) та адаптера, що переводить ці моделі у сприйнятливий для відображення вид (ViewModel).

MVVM полегшує поділ розробки графічного інтерфейсу користувача – мовою розмітки або кодом GUI – від розробки бізнес-логіки або локальної логіки (дані

моделі), у нашому випадку – розділ бізнес-логіки AR компонентів від користувацького інтерфейсу AR та від даних AR у їх звичайному вигляді.

Схему роботи MVVM зображено на рисунку 4.1.

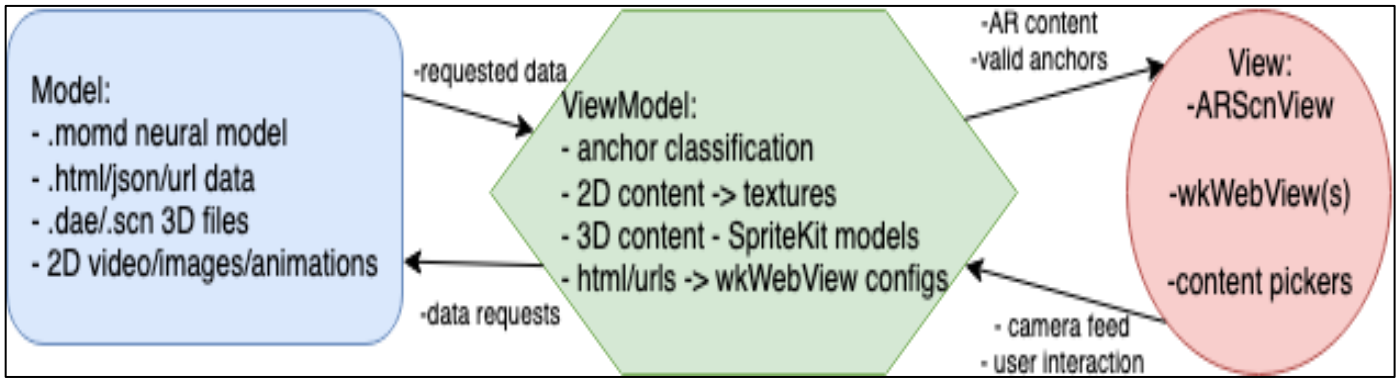

Рисунок 4.1 – MVVM схема AR додатку

Модель перегляду MVVM є перетворювачем значень, тобто модель перегляду відповідає за викриття (перетворення) об'єктів даних з моделі таким чином, що об'єкти легко керуються і подаються. У цьому відношенні модель перегляду є більш моделлю, ніж перегляд, і обробляє більшість, якщо не всю, логіку відображення перегляду. Модель перегляду може реалізовувати шаблон посередника, організовуючи доступ до локальної логіки навколо набору випадків використання, що підтримується видом. \ TMVVM є варіацією шаблону дизайну моделі презентації Мартіна Фаулера.

MVVM аналогічно описує стан і поведінку перегляду, але модель презентації абстрагує уявлення (створює модель перегляду) таким чином, що не залежить від конкретної платформи інтерфейсу користувача. \ TMVVM був винайдений архітекторами Майкрософт Кеном Купером і Тедом Петерсом для спрощення програмування користувальницьких інтерфейсів за допомогою подій. Шаблон був включений у Windows Presentation Foundation (WPF) (Microsoft. NET графічної системи) і Silverlight (WPF в інтернет-додатків похідних). Джон Госсман (John Gossman), один з архітекторів Microsoft WPF і Silverlight, оголосив MVVM у своєму блозі в 2005 році.

Модель-view-viewmodel також називається модель-view-binder, особливо в реалізаціях, які не залучають платформу .NET. ZK (фреймворк веб-додатків, написаний на Java) і KnockoutJS (бібліотека JavaScript) використовують модель-view-binder.

4.3 Програмна реалізація

Як було вже було зазначено, великий внесок у розвиток доповненої реальності було зроблено компанією Apple.

Майже всі кроки алгоритму системи, що розробляється, реалізуються наступними інструментами екосистеми:

– ARKit стек приймає контент у вигляді 2D або 3D об’єктів та проеціює його на точки (anchors), знайдені методами комп’ютерного зору та оброблені нейронною мережею;

– SpriteKit та SceneKit дозволяють створювати 2- та 3-вимірні моделі на основі динамічно отриманого контенту та будування просторових моделей на основі точок знайдених ARKit;

– CoreML фреймворк дозволяє використовувати моделі, отримані ззовні для класифікації оточення, отриманого з Vision [13];

– Vision фреймворк дозволяє використовувати сенсори мобільного пристрою як інструменти машинного зору та введення оточення для класифікації нейронною мережею, OpenCV додатково покращує результати роботи Vision [14];

– Create ML утиліта дає можливість створювати прості класифікатори для подальшого використання CoreML або конвертувати складні існуючі нейронні мережі в необхідний CoreML формат зчитування даних.

Як було вказано в останньому пункті, CreateML при розробці CoreML не може створювати складні моделі [15], а лише базові класифікатори зображень, звуку та

тексту. Тому для складних комплексних ситуацій необхідно використати додатковий алгоритм генерації складних моделей:

– фреймворк для навчання моделі (TuriCreate/TensorFlow);

– платформа для запуску цієї основи для навчальних цілей (Google Colab) [16];

– програми для розмітки фотографій (RectLabel);

– скрипти Python, які будуть використовуватися для перетворення цих анотацій у формат, необхідний фреймворкам;

– скрипт Python для оптимізації отриманої нейронної мережі [17].

Рисунок 4.2 зображає алгоритм роботи системи:

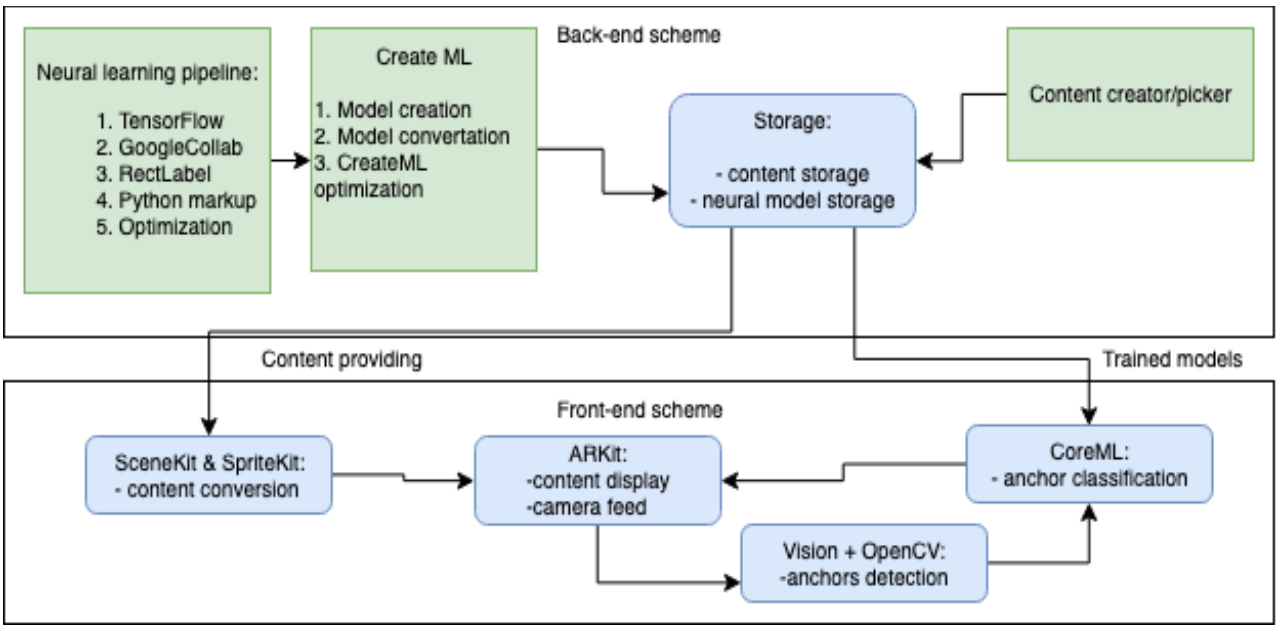

Рисунок 4.2 – Загальна схема системи генерації та проеціювання динамічного веб-контенту.

Таким чином, задача динамічного постачання моделей та контенту до кінцевого застосунку зводиться до алгоритму тренування та адаптації нейронної мережі у необхідний формат та перебудування контенту відповідно. Алгоритм роботи наведеної системи враховує складні випадки та є досить гнучким для переважної більшості випадків [18].

Отже, є чіткий розподіл на генерацію, перетворення та відображення контенту.

4.3.1 Тренування моделі засобами Turi Create

Turi спочатку був стартапом в Сіетлі, відомим своєю продукцією GraphLab, що спрощує загальні завдання машинного навчання. У 2016 році Apple придбала Turi, а пізніше в грудні 2017 року зробила Turi Create проектом з відкритим кодом.

Головним плюсом Turi є те, що система спрощує розробку власних моделей машинного навчання, щоб надавати можливість додавати рекомендації, виявлення об'єктів, класифікацію зображень, подібність зображень або класифікацію діяльності.

Також вона проста у використанні, візуальна, гнучка і експортується в сумісні з CoreML моделі. Отже, створені моделі можна використовувати відразу в iOS, macOS, tvOS і додатках watchOS без додаткового перетворення.

Turi Create фокусується на домені програми, а не на домені моделі. Це означає, що нам не потрібно турбуватися про побудову архітектури моделі та з'єднувальних шарів, а ми маємо справу з API високого рівня для виконання завдань, таких як завантаження зображень, навчання та оцінювання. Це також означає, що не потрібно використовувати фахівців з машинного навчання.

Turi Create автоматично використовує графічні процесори Mac для виконання таких завдань:

– класифікація зображень (macOS 10.13+);

– подібність зображення (macOS 10.13+);

– виявлення об’єктів (macOS 10.14+, лише дискретний GPU);

– класифікація активності (macOS 10.14+, лише дискретні GPU).

Наступний фрагмент коду демонструє процес тренування моделі за допомогою Python скрипта.

```
    import turicreate as tc
    import os
    data = tc.image_analysis.load_images('dataset',
with_path=True)
    data['hero_name'] = data['path'].apply(lambda path:
os.path.basename(os.path.dirname(path)))
```

```
    data.save('turi.sframe')
    data.explore()
    data = tc.SFrame('turi.sframe')
    train_data, test_data = data.random_split(0.8)
    model = tc.image_classifier.create(train_data,
target='hero_name')
    predictions = model.predict(test_data)
    metrics = model.evaluate(test_data)
    print(metrics['accuracy'])
    model.save('turi.model')
    model.export_coreml('model/TuriCreate.mlmodel')
```

На виході отримали треновану модель.

4.3.2 Тренування моделі та конвертація засобами Create ML

У 2018 році Apple представила Core ML: швидкий спосіб імпортувати попередньо навчені моделі машинного навчання в додаток з якомога меншим кодом! У цьому році, за допомогою програми Create ML, Apple надає розробникам можливість створювати власні моделі машинного навчання прямо в Xcode Playgrounds! Нам потрібен лише певний обсяг даних. На даний момент Create ML дозволяє переносити текст, зображення та таблиці як дані.

Однак, оскільки це являє собою більшість додатків ML, це повинно слугувати цілі добре.

Модель є результатом застосування алгоритму машинного навчання до набору навчальних даних. Можна використовувати модель для прогнозування на основі нових вхідних даних. Наприклад, модель, яка пройшла підготовку з історичних цін на житло в регіоні.

Класифікатор зображень – це модель машинного навчання, яка розпізнає зображення. Коли ви надаєте йому зображення, він відповідає міткою для цього зображення.

Діаграма, що показує, як класифікатор зображення передбачає позначку "Жираф" із зображення жирафа.Ви тренуєте класифікатор зображень, показуючи йому багато прикладів зображень, які ви вже позначили. Наприклад, ви можете навчити класифікатор зображень, щоб розпізнати тварин сафарі, показуючи йому різноманітні фотографії слонів, жирафів, левів і так далі.

На рисунку 4.3 зображено принцип роботи Create ML.

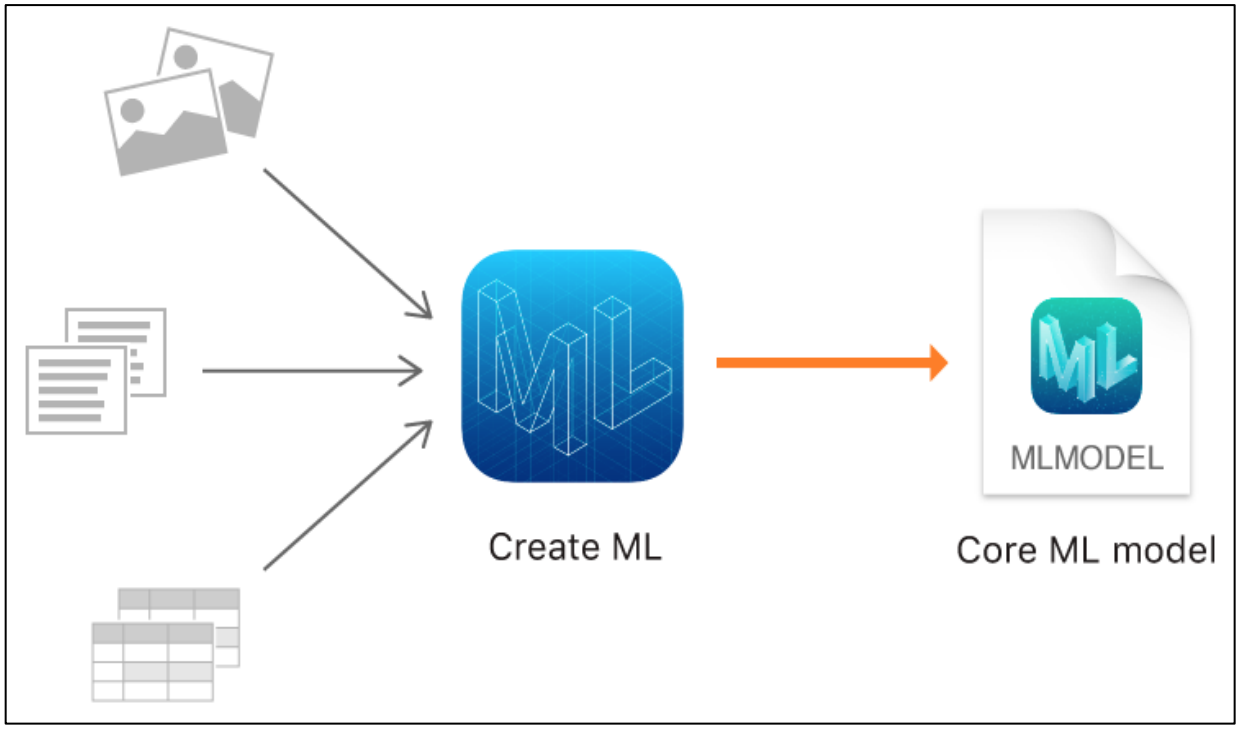

Рисунок 4.3 – Принцип роботи Create ML.

Остаточна модель міститься у файлі .mlmodel. Це новий відкритий формат файлів, який описує шари у вашій моделі, входи та виходи, мітки класів і будь-яку попередню обробку, яку необхідно виконати на даних. Вона також містить всі вивчені параметри (ваги та упередження).

Потім ми можемо просто завантажити .mlmodel в Xcode і почати робити прогнози.

Щоб полегшити ситуацію, припустимо, що у нас вже є джерело моделі, яке навчається з даними. Існують тонни цієї моделі в Інтернеті. Ми можемо зосередитися на частині перетворення між джерелом моделі та MLModel.

На рисунку 4.4 зображено процес конвертації моделі у робочий формат.

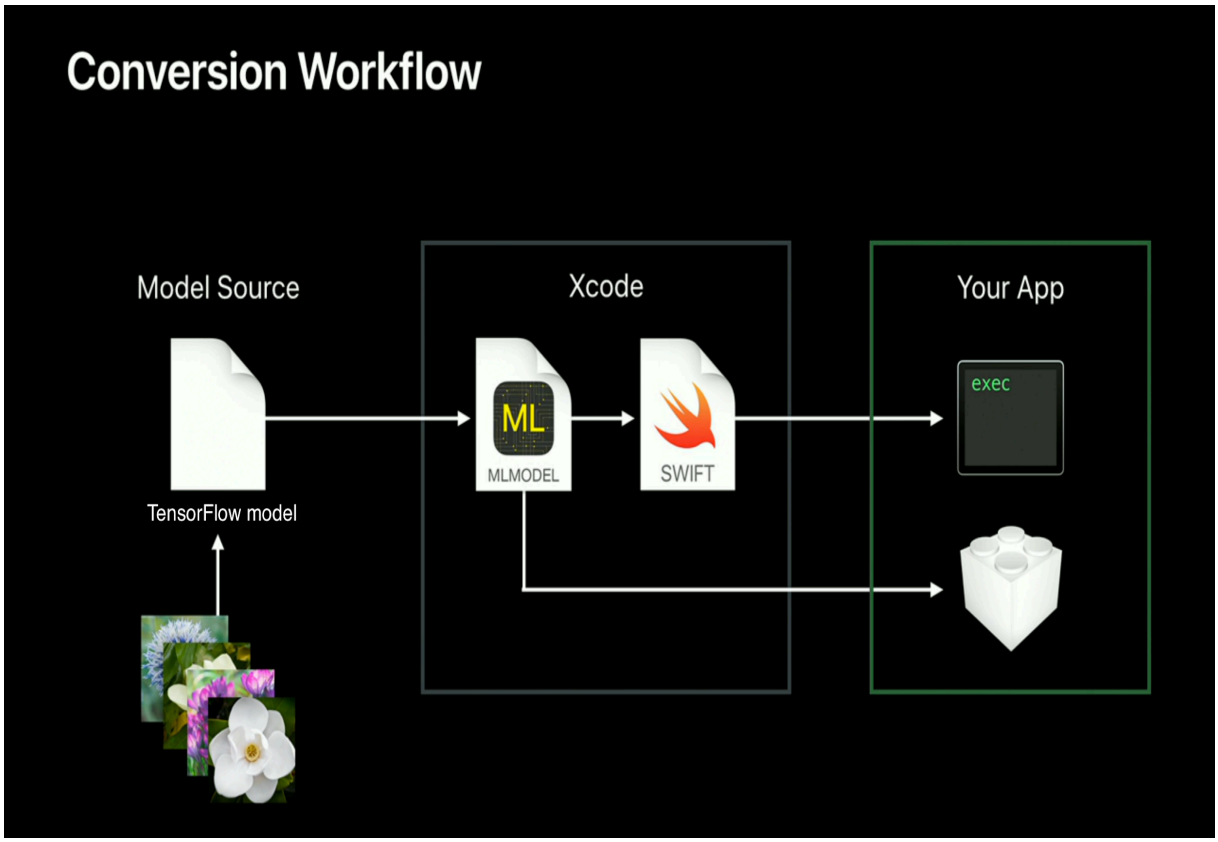

Рисунок 4.4 – Конвертація моделі TensorFlow у .mlmodel формат

Наразі для конверсії підтримуються наступні формати:

– TensorFlow;

– Keras;

– Kaffe;

– scikit-learn;

– XGBoost;

– libsvm.

Далі отримана модель записується до сховища даних.

4.3.3 Використання моделі у застосунку засобами CoreML

Швидкість роботи нейронних мереж на мобільних пристроях забезпечується BNNS та MPSCNN:

– BNNS (Basic Neural Network Subroutines) , Основні підпрограми для нейронних мереж, є частиною рамки прискорення, колекції математичних функцій, які повністю використовують швидкі векторні інструкції процесора;

– MPSCNN є частиною Metal Performance Shaders, бібліотеки оптимізованих обчислювальних ядер, які працюють на GPU, а не на процесорі.

У своєму нинішньому вигляді BNNS і MPSCNN корисні для виконання висновків на згорткових нейронних мережах.

У порівнянні з TuriCreate, де ви створюєте нейронну мережу з нуля шляхом побудови обчислювального графа, BNNS і MPSCNN є API більш високого рівня – математичних обчислень як таких значно менше.

Оскільки дані перетікають з одного шару в інший в нейронній мережі, кожен шар перетворює дані деяким чином. Як частина цього перетворення, шар застосовує функцію активації (activation function). Без цих функцій активації нейронні мережі не зможуть навчатися. Існує широкий вибір функцій активації, і BNNS і MPSCNN підтримують найбільш поширені

До цих функцій належать:

– ReL;

– logistic sigmoid;

– t*anh* та масштабований *tanh.*

BNNS визначає два типи BNNSActivationFunctionRectifiedLinear і BNNSActivationFunctionLeakyRectifiedLinear, але в MPSCNN є лише один MPSCNNNeuronReLU, який має параметр "alpha" для того, щоб зробити ReLU вихідним чи ні. Так само і для tanh і масштабованого tanh.

Це також має недолік: BNNS і MPSCNN набагато менш ефективні, ніж інші фреймворки, такі як TuriCreate. Легше їх запустити, але це обмежує глибоке навчання, яке ви можете зробити.

На рисунку 4.5 зображено рівні роботи з CoreML.

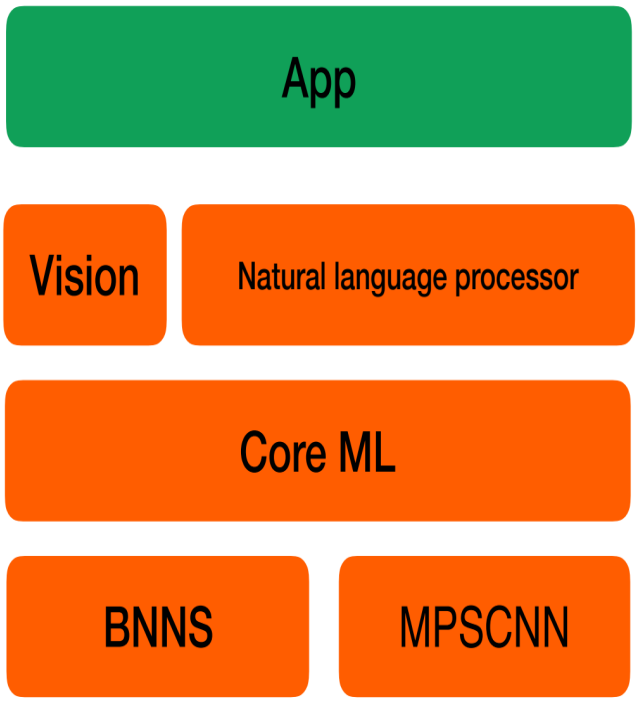

Рисунок 4.5 – Рівні роботи CoreML.

Фрагмент коду використання CoreML:

```
var tempBuffer1: [Float] = . . .
var tempBuffer2: [Float] = . . .

var results: [Float] = . . .

BNNSFilterApply(convLayer, inputData, &tempBuffer1)
BNNSFilterApply(poolLayer, tempBuffer1, &tempBuffer2)

BNNSFilterApply(fcLayer, tempBuffer2, &results)
```

Можна з упевненістю сказати, що MPSCNN набуває більш гнучкого та адаптованого підходу, ніж BNNS. Це те, що відбувається на всьому API.

Властивості обох методів наведені у таблиці 4.1.

Таблиця 4.1 – Порівняння BNNS та MPSCNN.

API	BNNS	MPSCNN
Компонент обробки	На основі ЦП	На основі GPU
Мова розробки	API на основі C	Адаптований для Swift
Ключова властивість	Швидка робота	Функції активації

Хоча BNNS працює швидше, аніж MPSCNN, було вирішено використовувати CoreML з MPSCNN завдяки підтримці Swift, та тому що ARKit використовує Metal framework для рендерінгу.

На рисунку 4.6. Зображено результати замірів швикдості роботи обох API у консолі.

```
Setting up neural network with Metal... Elapsed time: 0.09986683333409019 sec
Setting up neural network with BNNS... Elapsed time: 0.03813095833356783 sec
Performing inference with Metal... Elapsed time: 0.5680807083335822 sec
Performing inference with BNNS... Elapsed time: 0.5057908750004572 sec
```

Рисунок 4.6 – Результати замірів швикдості роботи обох API у консолі

Оскільки дані перетікають з одного шару в інший в нейронній мережі, кожен шар перетворює дані деяким чином. Як частина цього перетворення, шар застосовує функцію активації (activation function). Без цих функцій активації нейронні мережі не зможуть навчатися.

Таким чином можна бути певними, що MPSCNN набуває більш гнучкого та адаптованого підходу, ніж BNNS.

4.3.4 Використання Vision разом з CoreML

Варто розуміти різницю між Vision та CoreML:

– Core ML – спрощує використання навчальних моделей ваших додатків;

– Vision – дає вам легкий доступ до моделей від Apple для розпізнавання осіб, орієнтирів, тексту, прямокутників, штрих-кодів і об'єктів.

VNCoreMLRequest – це запит на аналіз зображення, який використовує модель Core ML для виконання даної роботи. Коли обробка завершилась, ми отримуємо об'єкт запиту і помилку.

VNClassificationObservation має два властивості: ідентифікатор – тип String і впевненість – число між 0 і 1 – це вірогідність того, що називається інформація про класифікацію. У майбутньому при використанні моделі розпізнавання об'єкта, ви, скоріше всього, звернете увагу тільки на ті об'єкти, які будуть довірятись, як дохід, до 30% або більше.

Наочний принцип роботи Vision зображено на рисунку 4.7.

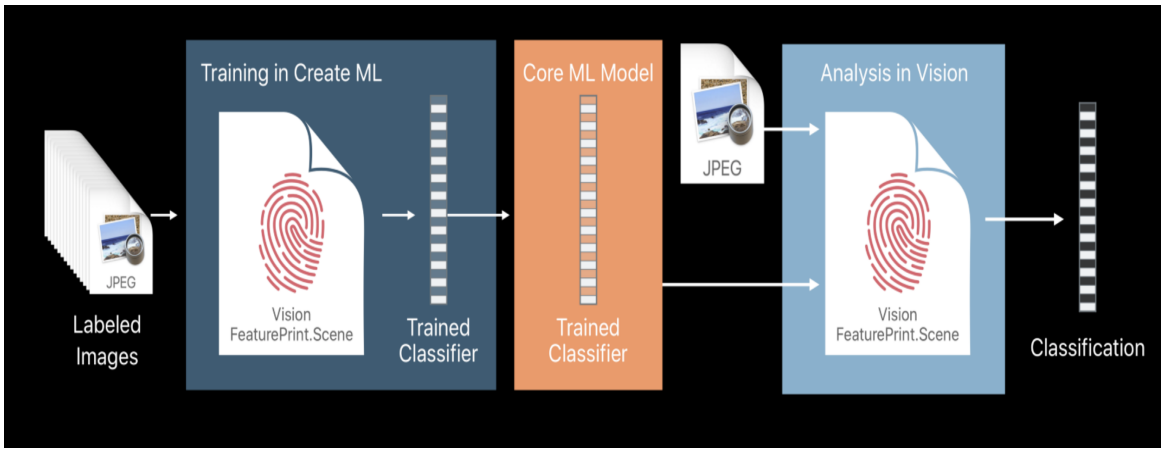

Рисунок 4.7 – Принцип роботи Vision

Потім ми перевіримо, що запити на ресурси представляють собою об'єкти VNClassificationObservation, це те, що повертається до Vision, так як модель Core ML є

скоріше класичним, ніж процесором обробки зображень або прогнозом. GoogLeNetPlaces теж є класифікатором, тому що він передбачає, але тільки одну особливість: класифікацію знімків.

На рисунку 4.8 зображено процес роботи запитів та спостережень у Vision.

Рисунок 4.8 – Процес роботи запитів та спостережень у Vision

Таким чином, Vision як більш високорівнева та швидка похідна CoreML має можливість швидше знаходити зображення, площини та прямокутники, які потім класифікуються CoreML і використовуються як опорні точки AR.

4.3.5 Використання SceneKit та SpriteKit для конвертації веб-контента у AR формат

SceneKit, іноді Scene Kit, являє собою інтерфейс програмування 3D-графіки (API) для платформ Apple Inc., написаних у Objective-C [19]. Це високорівневий фреймворк, розроблений для забезпечення простих у використанні шарів на рівні API нижчого рівня, таких як OpenGL і Metal. SceneKit підтримує об'єктний графік сцени, а також фізичний движок, систему частинок і посилання на основні анімації та інші фреймворки, які легко анімують цей дисплей.

Перегляди SceneKit можуть бути змішані з іншими переглядами, наприклад, дозволяючи відображення SpriteKit 2D [20] відображатися на поверхні об'єкта в SceneKit, або UIBezierPath від Core Graphics, щоб визначити геометрію об'єкта SceneKit. SceneKit поєднує в собі високопродуктивний механізм рендеринга з описовим API для імпорту, маніпулювання та рендеринга 3D-активів.

На відміну від API нижчого рівня, таких як Metal і OpenGL, які вимагають чіткої реалізації алгоритмів візуалізації, які відображають сцену, SceneKit вимагає лише описів вмісту сцени та дій або анімацій для виконання.

Наприклад, на рисунку 4.9 зображено об’єкт розроблений у Blender.

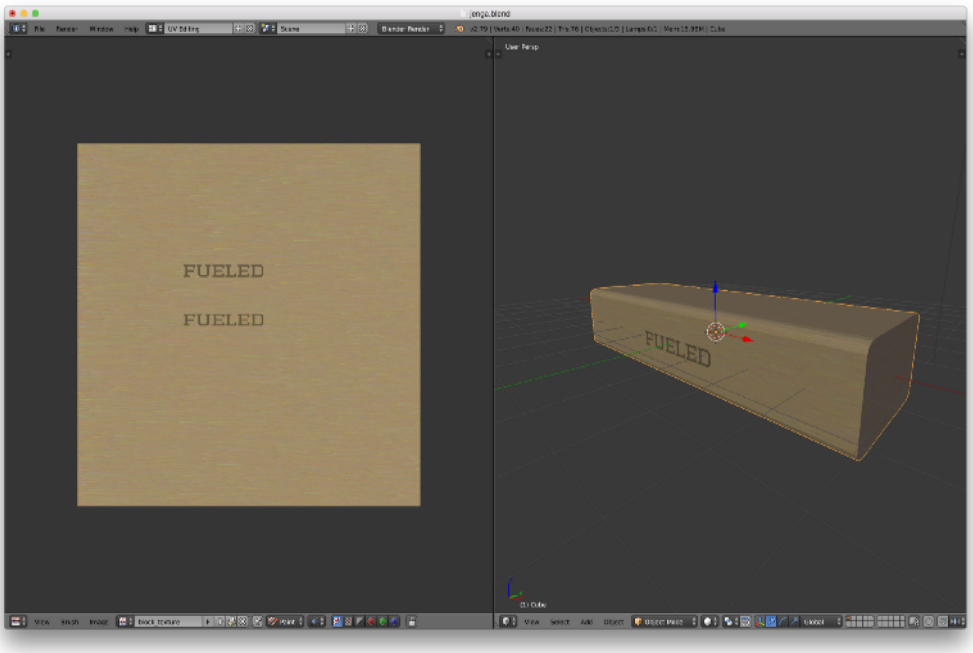

Рисунок 4.9 – Об’єкт розроблений у Blender

Можна відтворити цей об’єкт у редакторі SceneKit, проста сцена може бути створена шляхом створення одного об'єкта SCNGeometry, як правило, з одним з класів конструкторів, таких як SCNBox, однієї SCNCamera, одного або декількох SCNLights, а потім призначення всіх цих об'єктів окремим вузлам. Потім створюється один додатковий загальний вузол, який призначається кореневому вузлу об'єкта SCNScene,

а потім всі об'єкти додаються як діти цього кореневого вузла. SCNScene також містить декілька вбудованих елементів керування інтерфейсом користувача та бібліотеки вводу / виводу, які значно полегшують реалізацію простих глядачів і подібних завдань.

На рисунку 4.10 зображено об’єкт (див. рис. 2.9) у редакторі SceneKit.

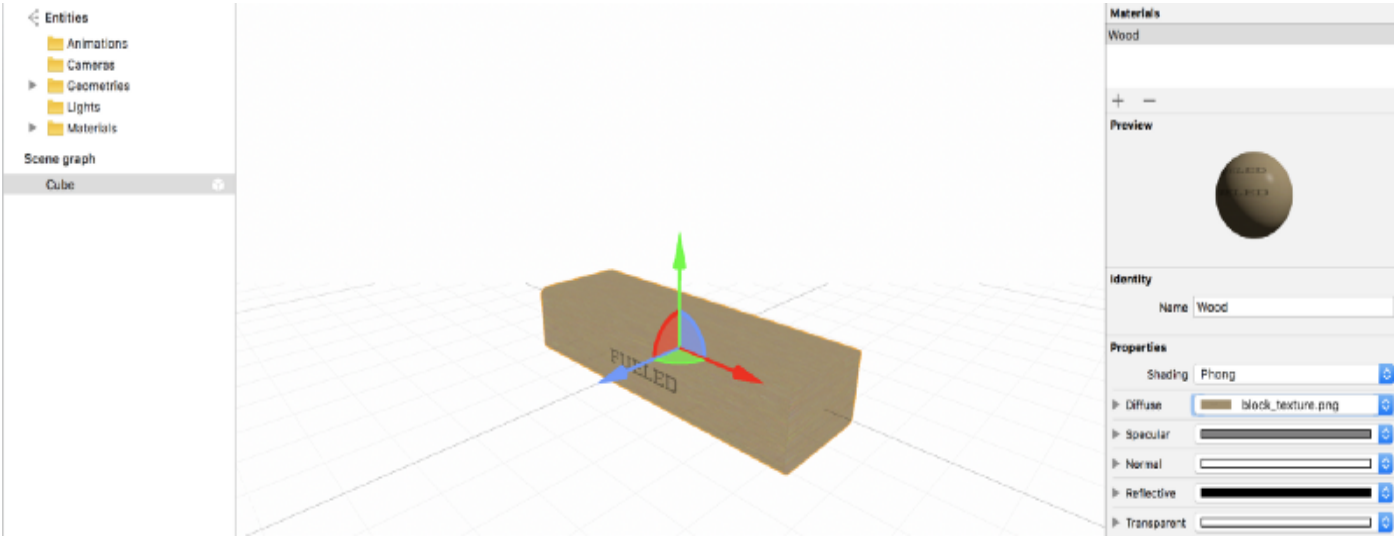

Рисунок 4.10 – Об’єкт у редакторі SceneKit

Проте це було виконано у самому Xcode, що не задовольняє цілям системи.

Для конвертації 3D контенту у рантаймі було використано SCNSceneSource, об'єкт, який керує завданнями зчитування даних, пов'язаних з завантаженням вмісту сцени з файлу. Xcode надає можливість конвертації більшості форматів у SceneKit формат.

Фрагмент коду завантаження наведено нижче.

```
    class func loadObjectNamed(_ objectName: String,
fromSceneNamed sceneName: String) -> CAAnimation? {
          let url = FileManager(urlForResource: sceneName,
ofType: “DAE”)
          #if os(OSX)
               let options: [SCNSceneSource.LoadingOption:
Any] = [SCNSceneSource.LoadingOption.convertToYUp: true]
          #else
```

```
            let options: [SCNSceneSource.LoadingOption:
Any] = [:]
        #endif
        let sceneSource = SCNSceneSource(url: url, options:
options)
        let animation =
sceneSource?.entryWithIdentifier(animationName, withClass:
CAAnimation.self)
        // Blend animations for smoother transitions
        animation?.fadeInDuration = 0.3
        animation?.fadeOutDuration = 0.3
        return animation
    }
```

Як результат маємо конвертований об’єкт.

На рисунку 4.11 зображено той же об’єкт завантажений у рантаймі у AR застосунок.

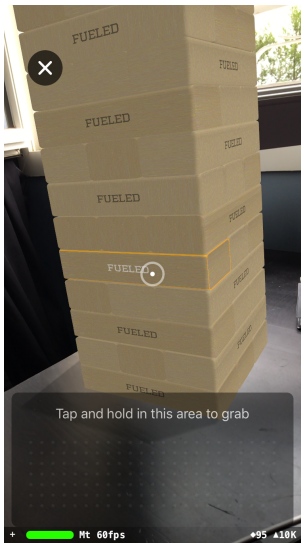

Рисунок 4.11 – Об’єкт завантажений у рантаймі у AR застосунок

Для проеціювання html контенту було реалізовано ARWebView, побудований на наступному:

– Екземпляр WKWebView. WKWebView – це клас iOS, який дозволяє розробникам вбудовувати веб-перегляди в свої рідні програми, а також відкривати можливості пристрою для веб-контенту за допомогою спеціальних API. У нашому випадку ми

використовуємо WKWebView, щоб розкрити функціональність ARKit веб-контенту. Структури вітчизняних / веб-програм, такі як Cordova, використовують аналогічний підхід;

– ARWebView вводить скрипт (ARWebView.js), як тільки сторінка завантажується в WKWebView. Цей скрипт, серед іншого, поліффікує WebVR 1.1 API і обробляє всі комунікації між рідним і веб-контентом.

На рисунку 4.12 зображено скріншот ARWebView що розпізнається як звичайний WebView у AR.

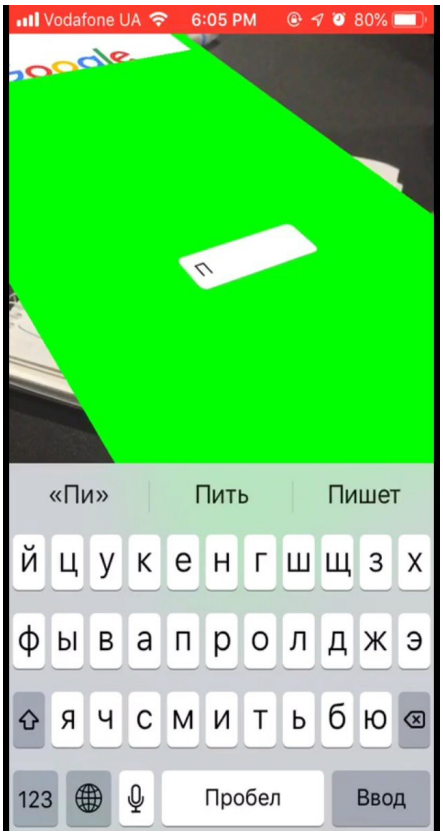

Рисунок 4.12 – Скріншот ARWebView

При запуску ARWebView шари повноекранного каналу подаються у фоновому режимі з прозорим WKWebView зверху. Таке розташування створює досить простий результат між "реальним світом" і веб-контентом, але має двонаправлений міст зв'язку між рідною стороною і стороною JavaScript який завжди асинхронний. ARWebView

намагається максимально вирішити це обмеження, використовуючи різні методики (наприклад, для hitTest, що має бути синхронним).

Таким чином відбувається конвертація об’єктів отриманих з веб-серверу до AR-сприйнятливого формату.

4.3.6 Побудова AR застосунку

ARKit використовує візуальну інерційну одометрію (VIO) для відстеження руху пристрою та навколишнього світу. VIO запобігає входженню на основі AVFoundation з датчика камери з даними руху пристрою, знятими за допомогою CoreMotion [21].

На рисунку 4.13 зображено стек технологій, на яких реалізовано ARKit.

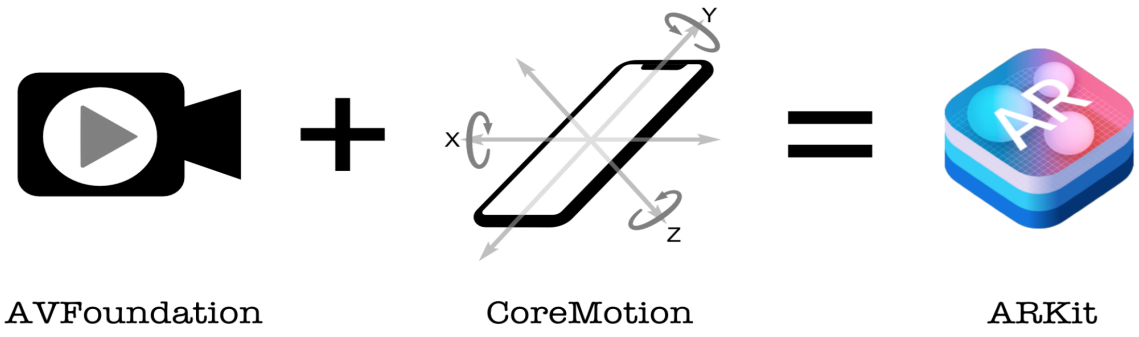

Рисунок 4.13 – Стек технологій, на яких реалізовано ARKit

Клас ARViewController керує сеансом AR і відображає вміст AR накладання у вигляді SpriteKit. ARKit захоплює відеокадри з камери і подає їх до контролера подання в методі session (_: didUpdate :), який потім викликає метод classifyCurrentImage () для запуску класифікатора зображень Vision.

Далі наведено фрагмент коду ініціалізації AR стеку.

```
    func session(_ session: ARSession, didUpdate frame:
ARFrame) {
```

```
            guard currentBuffer == nil, case .normal =
frame.camera.trackingState else {
          return
      }
          self.currentBuffer = frame.capturedImage
      classifyCurrentImage()}
```

Метод classifyCurrentImage () використовує властивість currentBuffer контролера перегляду, щоб відстежувати, чи Vision зараз обробляє зображення, перш ніж починати інше завдання Vision.

Більшість завдань комп'ютерного зору не є агностичними для обертання, тому важливо передати орієнтацію зображення відносно пристрою.

Фрагмент коду з’ясування орієнтації зображення.

```
    let orientation =
CGImagePropertyOrientation(UIDevice.current.orientation)

    let requestHandler = VNImageRequestHandler(cvPixelBuffer:
currentBuffer!, orientation: orientation)
    visionQueue.async {
      do {
                  defer { self.currentBuffer = nil }
          try
requestHandler.perform([self.classificationRequest])
      } catch {
          print("Error: Vision request failed with error
\"\(error)\"")
      }
    }
```

Обмежено обробку на один буфер одночасно для виконання. Камера переробляє кінцевий пул буферів пікселів, тому збереження занадто великої кількості буферів для обробки може призвести до уповільнення камери та вимкнення сеансу зйомки [22].

Передача декількох буферів Vision для обробки призведе до уповільнення обробки кожного зображення, додавання затримки і зменшення кількості процесорів і накладних витрат GPU для візуалізації AR.

Крім того, додаток дозволяє налаштування usesCPUOnly для свого запиту Vision, звільняючи GPU для використання у візуалізації.

Метод processClassifications (для: error :) зберігає мітку результату найкращого відповідності, створену класифікатором зображення, і відображає її в кутку екрана. Потім користувач може натиснути на сцену AR, щоб помістити цю позначку в реальне положення. Розміщення мітки вимагає двох основних кроків.

По-перше, розпізнавач жестів торкання спрацьовує дію placeLabelAtLocation (відправника :). Цей метод використовує метод ARKit hitTest (_: types :), щоб оцінити позицію 3D у реальному світі, що відповідає екрану, і додає прив'язку до AR сесії в цій позиції [23].

Фрагмент коду прив’язки об’єкта до маркера.

```
    @IBAction func placeLabelAtLocation(sender:
UITapGestureRecognizer) {
      let hitLocationInView = sender.location(in: sceneView)
      let hitTestResults =
sceneView.hitTest(hitLocationInView, types: [.featurePoint,
.estimatedHorizontalPlane])
      if let result = hitTestResults.first {

                  let anchor = ARAnchor(transform:
result.worldTransform)
          sceneView.session.add(anchor: anchor)

                  anchorLabels[anchor.identifier] =
identifierString
      }
    }
```

Далі, після того, як ARKit автоматично створить вузол SpriteKit для доданого якоря, метод перегляду (_: didAdd: for :) надає вміст для цього вузла [24]. У цьому випадку клас ARLabelNode створює стилізовану текстову мітку, використовуючи рядок, наданий класифікатором зображень.

Фрагмент коду створення ще одного об’єкта і додавання до маркера.

```
    func view(_ view: ARSKView, didAdd node: SKNode, for
anchor: ARAnchor) {
      guard let labelText = anchorLabels[anchor.identifier]
else {
          fatalError("missing expected associated label for
anchor")
```

```
    }
    let label = TemplateLabelNode(text: labelText)
    node.addChild(label)
}
```

Таким чином будуємо AR об’єкти на розпізнаних маркерах.

4.4 Тестування програмної системи

Зазвичай у мобільних застосунках використовується Unit Testing та UI testing, що підходить для перевірки роботи даного AR застосунку.

Unit тестування – це рівень тестування програмного забезпечення, в якому перевіряються окремі одиниці / компоненти програмного забезпечення. Мета полягає в тому, щоб перевірити, що кожна одиниця програмного забезпечення виконується так, як вона була розроблена.

Пристрій є найменшою частиною будь-якого програмного забезпечення, що перевіряється. Зазвичай він має один або декілька входів і зазвичай єдиний вихід. У процедурному програмуванні одиниця може бути окремою програмою, функцією, процедурою тощо. У об'єктно-орієнтованому програмуванні найменша одиниця – це метод, який може належати базовому / супер-класу, абстрактному класу або похідному / дочірньому класу.

Деякі з них розглядають модуль програми як одиницями. Це не рекомендується, оскільки в цьому модулі, ймовірно, буде багато окремих одиниць.) Для полегшення тестування використовуються модульні тестові рамки, драйвери, заглушки та макетні / підроблені об'єкти. .

Unit тести реалізуються за допомогою QUnit в якості тестової основи. Тестова папка містить автоматизовані тестові набори.

Метод модульного тестування повинен відповідати узгодженому формату, щоб полегшити його розуміння іншими розробниками. Наступним форматом, як правило, вважається найкраща практика:

– Arrange;

– Act;

– Assert.

По-перше, для створення екземпляра класу, що тестується, а також будь-яких залежностей, які він може мати, може бути написаний деякий код установки. Це розділ "Впорядкувати" тестового модуля.

Після того, як блок тестування встановить стадію для фактичного випробування, метод може бути виконаний або властивість. Це розділ Акту тесту. Також він може бути виконаний з урахуванням параметрів (якщо застосовуються), встановлених під час розділу "Розташування".

Нарешті, при виконанні даного методу або власності тест повинен перевірити, чи метод чи властивість зробили саме те, що він повинен був зробити. Це розділ Assert тесту. Під час фази верифікації, методи затвердження викликаються для порівняння фактичних результатів з очікуваними результатами. Якщо фактичні результати є такими, як очікується, проходить модульний тест. Якщо ні, тест провалиться. Можна запустити програму в Xcode і вставити нерухоме зображення.

Це допомагає перевірити можливість виявлення зображень. Цей підхід використовується для розміщення нерухомого зображення і надає йому декілька різних варіантів, наприклад, чорно-білу або розмитої версії, і перевіряє, чи правильно інтерпретувати зображення.

ВИСНОВКИ

Було розглянуто актуальну тему проеціювання та обробки веб-контенту засобами доповненої реальності, бо їх потенційні можливості котрих ще не повністю вивчені та існує ряд проблем, що потребують вирішення.

Мету роботи було досягнено, а саме досліджено існуючі методів доповненої реальності та їх різновиди, а також можливості доставки та взаємодії веб-контенту при використанні цих методів, було розроблено алгоритм генерації і доставки компонентів доповненої реальності, що використовує ці методи, а також програмну систему, що реалізує цей алгоритм.

Було досліджено:

– існуючі методи доповненої реальності;

– існуючі засоби доповненої реальності;

– існуючі інструменти доповненої реальності.

Можна зробити висновок, що для задачі обробки і проеціювання веб-контенту краще всього підходять маркерний метод проеціювання для мобільних пристроїв та фреймфорк Apple ARKit, на яких було розроблену систему, що може:

– тренувати на обраному датасеті та зберігати моделі доповненої реальності;

– конвертувати ці моделі у необхідний застосунку формат;

– постачати їх на мобільний застосунок;

– зберігати та передавати мобільному застосунку веб-контент;

– динамічно обробляти цей;

– будувати модель відображення отриманого контента.

Для вирішення задачі створення і постачання нейронних мереж було реалізовано алгоритм тренування моделі нейронної мережі на віддаленому сервері із тестовими датасетами, переведення цієї мережі у формат, що буде здатний працювати з мобільними технологіями доповненої реальності.

Задачу доставки і обробки саме веб-контенту було реалізовано шляхом деконструкії вихідних об’єктів і подальше використання у Apple SceneKit.

Також було досліджено методи ручної оптимізації нейронних мереж засобами Python для використання мобільними пристроями, а також оптимізацю використання цих нейронних мереж технологіями Apple CoreML та CreateML.

В ході роботи застосовувались емпіричні методи програмної інженерії, загальнологічні методи наукового пізнання, а також порівняльний метод.

Було доведено наукову новизну роботи, а саме об’єднано методи генерації з удосконаленими методами оптимізації та конвертації нейронних мереж для використання у зв’язці з доповненою реальністю, розроблено методи проеціювання веб-контенту за допомогою програмних об’єктів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ: Про ОБЧІСЛЕННЯ, М.А. Новотарській, Б.Б. Нестеренко [Текст] – Інститут математики НАН України, Київ 2004 – 127 с.

2. Drummond T., Rosten E. Machine Learning for high-speed corner detection – 9th European Conference on Computer Vision (ECCV), p. 430-443, 2006.

3. Lowe D.G. Distinctive Image Features from Scale-Invariant Keypoints [Текст] – International Journal of Computer Vision, p. 91-110, 2004.

4. Ray Wanderlich, ARKit by Tutorials// Razeware LLC – p. 32-37, 2019

5. Ian Goodfellow, Yoshua Bengio, Deep Learning (Adaptive Computation and Machine Learning series) // The MIT Press – p. 621-638, 2016.

6. Frank Millstein, Deep Learning: 2 Manuscripts – Deep Learning With Keras And Convolutional Neural Networks In Python // Paperback – p.117-129, 2018

7. Joshua Newman, Machine Learning with Core ML: An iOS developer's guide to implementing machine learning in mobile apps // Packt Publishing – p. 45-48, 2018

8. Apple Developer Documentation: Creating Core ML [Електронний ресурс]. — Режим доступу: https://developer.apple.com/documentation/coreml (дата звернення: 09.05.2019).

9. Apple Developer Documentation: Creating Core ML [Електронний ресурс]. — Режим доступу: https://developer.apple.com/machine-learning/ (дата звернення: 01.02.2019).

10. Apple Developer Documentation: Creating Core ML [Електронний ресурс]. — Режим доступу: https://developer.apple.com/augmented-reality/ (дата звернення: 04.03.2019).

11. Apple Developer Documentation: Creating Core ML [Електронний ресурс]. — Режим доступу: https://developer.apple.com/documentation/arkit (дата звернення: 03.06.2019).

12.Apple Developer Documentation: Creating Core ML [Електронний ресурс]. — Режим доступу: https://developer.apple.com/documentation/vision (дата звернення: 10.12.2018).

13.OpenCV Developer Documentation: Getting started [Електронний ресурс]. — Режим доступу: https://docs.opencv.org/master/d9/df8/tutorial_root.html (дата звернення: 05.05.2019).

14.OpenCV Homepage [Електронний ресурс]. — Режим доступу: https://opencv.org/ (дата звернення: 01.04.2019).

15.Medium: Using Machine Learning and CoreML to control ARKit [Електронний ресурс]. — Режим доступу: https://medium.com/s23nyc-tech/using-machine-learning-and-coreml-to-control-arkit-24241c894e3b (дата звернення: 02.01.2019).

16.Medium: Using inception v3 for image classigifcation [Електронний ресурс]. — Режим доступу: https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c (дата звернення: 10.05.2019).

17.Habrahabr: Еволюція нейронних мереж у Google Inception v3 [Електронний ресурс]. — Режим доступу: https://habr.com/ru/post/302242/ (дата звернення: 03.02.2019).

18.Inception with CoreML on GitHub [Електронний ресурс]. — Режим доступу: https://github.com/hollance/Inception-CoreML (дата звернення: 12.02.2019).

19.Apple Developer Documentation: SceneKit [Електронний ресурс]. — Режим доступу: https://developer.apple.com/documentation/scenekit (дата звернення: 23.04.2019).

20.Apple Developer Documentation: SpriteKit [Електронний ресурс]. — Режим доступу: https://developer.apple.com/spritekit/ (дата звернення: 30.02.2019).

21.Julie Carmigniani, Borko Furht, Marco Anisetti, Paolo Ceravolo, Ernesto Damiani, Misa Ivkovic Augmented reality technologies, systems and applications // Multimed Tools Appl 2011 C. 377. — Режим доступу: https://www.csd.uoc.gr/~hy469/files/panels/Augmented_reality_technologies_systems_and_applications.pdf (дата звернення: 12.01.2019).

22.Jessica H. Robin Augmented reality technologies // PIRE 2014 С. 12. — Режим доступу:http://pire.fiu.edu/publications/Augmented.pdf (дата звернення: 04.04.2019).

23.Jonathan Linowes, Krystian Babilinski Augmented Reality for Developers — Packt Publishing Ltd, 9 жовт. 2017 р. — 441 с.

24.Apple Developer Documentation: ARKit: tracking and visualizing planes [Електронний ресурс]. — Режим доступу: https://developer.apple.com/documentation/arkit/tracking_and_visualizing_planes (дата звернення: 20.05.2019).