МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

СЛОБОДЯН Владислав Романович

Алгоритми управління звуком комп'ютера за допомогою жестів на основі бібліотеки MediaPipe / Algorithms of computer sound management by using gestures based on the MediaPipe library

спеціальність: 123 – комп'ютерної інженерії
освітньо-професійна програма – Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи Кім-21
В.Р. Слободян
_____

Науковий керівник:
О.Й. Піцун
_____

Кваліфікаційну роботу

допущено до захисту
«__»_____ 20__ р.

Завідувач кафедри
_____ Л.О. Дубчак

ТЕРНОПІЛЬ - 2022

ЗМІСТ

ВСТУП

Розпізнавання жестів допомагає комп'ютерам розуміти мову людського тіла. Це допомагає створити міцніший зв'язок між людьми та машинами, окрім простих текстових або графічних інтерфейсів користувача (GUI). У цьому проекті з розпізнавання жестів рухи людини зчитуються камерою комп'ютера. Потім комп'ютер використовує ці дані як вхідні дані для програм обробки. Метою цього проекту є розробка інтерфейсу, який динамічно фіксує рухи рук людини та контролює рівні гучності.

Технологія управління жестами дозволяє користувачу більш інтуїтивно використовувати свої пристрої, керувати та адмініструвати їх. Також дана технологія є незамінимою для людей із певними вадами. У наш управління жестами, як комп'ютерна технологія, впроваджується у багатьох сучасних індустріях, від інтерактивних розваг до автомобільної промисловості.

У майбутньому, у міру розвитку технології, розпізнавання жестів вийде за рамки інформаційно-розважальних систем і дозволить користувачам керувати більш широким спектром систем і структур, іншими системами автомобіля, такими як опалення та охолодження, а також підключатися до систем розумного будинку. Наприклад, уявіть, що ви можете перевірити камери відеоспостереження свого будинку одним рухом, коли ви їдете додому. Жести також можна поєднувати з телематичними системами, відкриваючи ширший спектр застосувань у майбутньому.

Метою роботи є розроблення адаптивного алгоритму управління звуком комп'ютера за допомогою жестів із використанням бібліотеки MediaPipe а також дослідження можливостей технології управління комп'ютерними системами за допомогою жестів.

Для розв'язання поставлених задач у кваліфікаційній роботі використано: технологію комп'ютерного навчання та комп'ютерного зору для попереднього оброблення зображень; об'єктно-орієнтованого програмування; бібліотека MediaPipe (для проектування програмних засобів аналізу зображень).

Науковою новизною даної роботи є те що запропонований алгоритм є більш гнучким, універсальним, а також завдяки бібліотеці MediaPipe витрачає в рази менше потужностей пристрою. Перевагою даної розробки є те що, створюваний алгоритм може працювати із будь якими домашніми пристроями передачі зображення.

Орієнтовні напрямки розвитку досліджень: розроблення спеціалізованого алгоритму управління комп'ютерними системами на основі, дослідження бібліотеки MediaPipe, розроблення алгоритму розпізнавання зображення на основі MediaPipe Hands, створення програмного застосунку для реалізації управління звуком комп'ютера за допомогою жестів.

Кваліфікаційна робота складається із трьох розділів, висновків, списку використаної літератури та додатків. У першому розділі було проаналізовано технологію управління комп'ютерними системами на основі жестів, вивчено історію розвитку даної технологію. А також детально досліджені основні характеристики бібліотеки MediaPipe.

У другому розділі було проаналізовано основні алгоритми розпізнавання зображень, а також проведено детальний аналіз технології та алгоритми комп'ютерного навчання, саме до яких відноситься бібліотека MediaPipe. Також було створено узагальнений алгоритм опрацювання вхідного зображення(жестів користувача) а також алгоритм опрацювання жестів бібліотекою MediaPipe.

У третьому розділі детально описана архітектура проекту, а також програмно реалізований алгоритм управління комп'ютерним звуком на основі жестів з допомогою бібліотеки MediaPipe.

Практична цінність одержаних результатів полягає в тому, що: розроблено та проведено експериментальне технології управління комп'ютерними системи на основі жестів. Розроблено структуру алгоритму опрацювання і інтерпретації жестів користувача бібліотекою MediaPipe, створено програмний застосунок для управління комп'ютерним звуком жестами, на основі бібліотеки MediaPipe.

Публікації та апробація до випускної кваліфікаційної роботи. За результатами наукових досліджень, проведених у випускній кваліфікаційні

роботі, підготовлено тези доповіді «Особливості технології згорткових нейронних мереж для розпізнавання образів» обсягом 1 сторінка та «Огляд інструментів для розпізнавання жестів з відеокамери» обсягом 1 сторінка на Науково-практичній конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі».

1. АНАЛІЗ ТЕХНОЛОГІЇ УПРАВЛІННЯ ЖЕСТАМИ

1.1 Технологія управління комп'ютерними системами за допомогою жестів

Управління жестами — це здатність розпізнавати та інтерпретувати рухи людини для взаємодії та керування комп’ютерними системами без прямого фізичного контакту. Термін «природний інтерфейс користувача» стає загальновживаним для опису цих систем інтерфейсу, що відображає загальну відсутність будь-яких проміжних пристроїв між користувачем і системою.

З початку комп’ютерної революції докладалися зусилля для покращення взаємодії людини з комп’ютером. Сьогодні комп’ютери є настільки невід’ємною частиною нашого життя, що користуватися ними має бути так само просто, як спілкуватися з людьми. Раніше люди використовували клавіатуру або мишу для взаємодії з цією розумною машиною. Але зараз робляться спроби зробити взаємодію між людьми і машинами максимально природними. Протягом десятиліть клавіатура та миша були основними пристроями введення для комп’ютерів. Однак із появою повсюдних зовнішніх пристроїв і апаратного забезпечення, які дозволяють користувачам охоплювати віртуальні об’єкти, жести руками або тілом стали важливими: управління жестами стало ядром систем взаємодії людини з комп’ютером.

Жести людини, безсумнівно, природні. Вони часто ефективніші та потужніші, ніж різні інші способи взаємодії. Відстеження положення або конфігурації голови/руки чи тіла є дуже цінним для керування об’єктами/системами або передачі вхідних параметрів системі. Жести також можна використовувати для самовираження. Наприклад, кивок головою може означати згоду або згоду, підняття пальця — бажання перервати, а «так» — «я з вами згоден, продовжуйте». Розпізнавання жестів передбачає відстеження позиції, орієнтації або руху людини та остаточну інтерпретацію їх для розпізнавання семантично послідовних жестів.

Жести використовуються для передачі інформації різними способами. Смуток можна передати через вираз обличчя, схилені голови, опущені плечі та мляві рухи.

Подібним чином жест «Стій!» може позначатися піднятою рукою, долонею вперед, або перебільшеним помахом руками над головою. Оскільки існує відображення «багато-до-одного» від понять до жестів, жести часто є неоднозначними; водночас існує також відображення «багато-до-одного» від жестів до понять, тому жести не є повністю визначеними. Оскільки мова та почерк у кожної людини різні, жести теж суб’єктивні. Вони відрізняються від людини до людини.

Хоча жестове спілкування багате, воно однаково складне. Дослідники диференціюють їх різними способами. Зазвичай вирізняють 3 основних типів жестів: Жестикуляція(Мимовільні рухи кистей і рук, що супроводжуються промовою), мовні жести(Жестикуляція інтегрована в мову, замінюючи певне вимовлене слово), пантоніми(Жести, які зображують предмети або дії, з супроводом мови або без нього).

Спілкування мовою жестів, незважаючи на багате, але однаково складне. Дослідники розрізняють їх по-різному. Загалом жести поділяються на 3 основні типи: жести (мимовільні рухи рук і рук, що супроводжуються промовою), розмовні жести (жести, інтегровані в мову, які замінюють вимовлене слово), повні назви (жести, що описують об’єкти чи дії, з супроводом або без нього).

Спонтанні жести складають приблизно 90% людських жестів. Люди навіть використовують жести руками, коли розмовляють по телефону, а незрячі люди часто використовують жести руками, коли розмовляють по телефону. Жести, пов'язані з промовою, є природними і поширеними в різних культурах. Тим не менш, жести підпису та мова жестів, хоча, можливо, менш спонтанні та природні, мають чіткішу семантику та можуть краще підходити для командно-контрольної взаємодії.

Розпізнавання жестів — це процес, у якому система розпізнає жести, зроблені користувачем.

Його також можна інтерпретувати як математичну інтерпретацію руху людини обчислювальними пристроями. В даний час використовуються різні типи технологій розпізнавання жестів.

Контактний тип - тип контакту, який передбачає жести на основі дотику за допомогою сенсорної панелі або сенсорного екрана. Розпізнавання жестів на сенсорній панелі чи сенсорному екрані здійснюється шляхом виявлення фізичного контакту на звичайній сенсорній панелі чи сенсорному екрані. Сенсорні панелі та сенсорні екрани в основному використовуються для керування курсором на ПК чи мобільному телефоні та стають дедалі популярнішими серед користувачів торгових терміналів, КПК, різноманітних промислових і автомобільних програм. Вони використовувалися в автомобільних додатках і КПК. Прийняття користувачами технологій автомобільних систем на основі жестів було відносно легким для широкої громадськості, оскільки вони зберігають фізичний інтерфейс користувача.

Технологія жестів на основі пристрою – це технологія на основі пристрою, яка використовує рукавичку, стилус або інший пристрій відстеження положення, рух якого надсилає сигнали, які використовуються системою для розпізнавання жестів. Звичайна техніка розпізнавання жестів передбачає надягання на руку рукавички; рукавичка оснащена різними датчиками, які надають інформацію про положення руки, орієнтацію та згинання пальців. Перший комерційний ручний трекер Dataglove використовує тонкі волоконно-оптичні кабелі, які проходять уздовж тильної сторони кожної руки, з невеликим отвором у кожній руці. Світло спрямоване вниз по кабелю так, що при згинанні пальця світло просочується через щілину. Вимірювання втрати світла дозволяє точно визначити положення руки. Подібні методи використовуються для одягу, що використовується у віртуальних середовищах. Хоча рукавички можуть забезпечити точні вимірювання руки, їх важко надягати та підключати до проводів. У літературі повідомлялося про різні типи систем для розпізнавання нав’язливих жестів. Деякі використовують датчик

згинання на вказівному пальці та акселерометр на руці з мікроперемикачем для активації.

Стилус поєднується з технологією відображення для запису та інтерпретації жестів, наприклад написання тексту. Щоб зменшити фізичні обмеження кабелів, була використана альтернативна техніка: ультразвуковий передавач одягається на вказівний палець.

Приймач, здатний відстежувати положення передавача, встановлений на головному дисплеї (HMD). Технологія на основі комп'ютерного зору. Існує два підходи до розпізнавання жестів на основі зору. Перший – модельний. Цей підхід намагається створити 3D-модель руки користувача та використовувати її для розпізнавання. Деякі системи відстежують рух жестів через набір ключових позицій. Система розпізнає жест, коли він переміщається повз ту саму клавішу, що й збережений жест. Інші системи відстежують рухомі частини тіла, обчислюють характер руху, а потім визначають жести. Системи зазвичай роблять це шляхом застосування статистичного моделювання до набору рухів. Другий спосіб заснований на зображеннях. Методи на основі зображень виявляють жести шляхом захоплення рухомих зображень користувача під час виконання жесту. Система надсилає ці зображення програмі комп’ютерного зору, яка відстежує їх і розпізнає жести. Ці методи зазвичай виділяють колір шкіри з фонових зображень, щоб знайти руки, а потім намагаються витягти такі елементи, як кінчики пальців, краї рук або загальну геометрію руки для розпізнавання жестів.

Хоча технологія розпізнавання жестів спочатку була призначена для покращення взаємодії людини з комп’ютером, вона знайшла багато застосувань із поширенням використання комп’ютера. Люди з вадами зору або іншими складними руховими функціями можуть використовувати пристрої введення на основі жестів, щоб уникнути дискомфорту під час доступу до комп’ютера. Крім того, сучасні електричні інвалідні візки оснащені системами на основі жестів. Все, що тут потрібно зробити від користувача, це злегка провести рукою вздовж панелі на підлокотнику крісла. Рухи рук

виконуватимуть функцію контролерів, дозволяючи легко контролювати швидкість і напрямок.

Технологія розпізнавання жестів набирає популярності майже в усіх сферах, де використовуються розумні машини. У системах управління рухом літаків ця технологія може допомогти деталізувати кожну інформацію про місцезнаходження літаків поблизу аеропорту.

У кранах це можна використовувати замість пульта дистанційного керування, щоб легко піднімати та опускати вантажі у складних положеннях. Сьогодні смарт-телевізори оснащені цією технологією, що дозволяє користувачам перемикати канали або гучність обома руками, не турбуючись про пульт дистанційного керування. Qualcomm нещодавно випустила розумні камери та планшети на основі цієї технології. Камера розпізнає відстань до об’єкта перед зйомкою та відповідно регулює її. Коли користувачеві потрібно зробити демо або змінити пісні на своєму музичному автоматі, планшет із цією технологією допоможе впоратися з цим завданням. Лише помахом руки він може переглянути всі дані. Різні смартфони з сенсорним екраном також використовують цю технологію для забезпечення легкого доступу. Технологію розпізнавання жестів також можна використовувати, щоб дозволити роботам розуміти людські жести та працювати відповідно.

З технологією розпізнавання зображень також багато проблем.

- Затримка. Однією з ключових проблем із розпізнаванням жестів є те, що обробка зображень може бути набагато повільнішою, спричиняючи неприйнятні затримки для відеоігор та інших подібних програм.

- Немає мови жестів. Оскільки універсальної мови жестів не існує, різні користувачі жестикулюють по-різному. Якщо користувач жестикулює так, як вважає за потрібне, системі розпізнавання жестів буде важко розпізнати дію за допомогою імовірнісних методів, які зараз використовуються.

- Міцність. Багато систем розпізнавання жестів не зчитують жести точно або оптимально через такі фактори, як недостатнє фонове освітлення, високий фоновий шум тощо.

-Продуктивність. Обробка зображень, пов’язана з розпізнаванням жестів, потребує ресурсів, і запуск програм на пристроях з обмеженими ресурсами, наприклад КПК, може бути складним завданням.

Жести є потужною формою спілкування для взаємодії людини з комп’ютером.

Зараз існує багато основних пристроїв для взаємодії з комп’ютерами, таких як клавіатури, миші, джойстики та сенсорні екрани, але ці пристрої не забезпечують простіший спосіб спілкування. Через цю проблему запропонована система складається з настільного комп’ютера та ноутбука або інтерфейсу настільного комп’ютера, використовуючи веб-камеру або окрему камеру для захоплення жестів для запису жестів. Система розпізнавання жестів полягає в реалізації відстеження руки користувача. Інші датчики, які використовуються в системі, також оброблятимуть конфігурацію руки та її рух. Методи на основі зору вимагають веб-камери, яка забезпечує природну взаємодію людини з комп’ютером без використання додаткового обладнання. Складною частиною в цих системах є фон зображень або відео, які записуються або фіксуються під час прийому вхідних даних, тобто жести рукою користувача, іноді з ефектом блискавки, який безпосередньо впливає на якість вхідного сигналу, який у свою чергу викликає розпізнавання жестів та інші проблеми оцифрування. Процес пошуку релевантних областей на зображенні з деякими основними властивостями, такими як колір вхідного зображення, інтенсивність і зв’язність пікселів у зображенні, тобто шаблон, називається сегментацією.

1.2 - Історія розвитку технології управління жестами

Технологія керування жестами стрімко розвивається та змінює кожен аспект нашого життя. Пристрої з керуванням жестами варіюються від дуже

простих пристроїв введення до детального розпізнавання. Ці пристрої використовуються в широкому діапазоні застосувань від дослідницьких експериментів і прототипів до повсякденних продуктів.

Однак технологія пройшла досить тривалу еволюцію. Люди користуються цією формою невербального спілкування більше мільйона років, і навіть деякі з її основних сигналів стали звичайними жестами: рух головою, щоб підтвердити або спростувати щось, насупитися, щоб показати роздратування, і знизати плечима, щоб показати, що щось не так.

Правильно розуміти, навіть ті жести, успадковані від тваринного світу, такі як оголення зубів для вираження агресії. Спосіб розвитку невербальної комунікації полягає в тому, що певна мова створюється з серії жестів. Деякими прикладами є мова жестів, якою користуються глухі, або універсальні знаки, що використовуються для авіації, дайвінгу чи надання першої допомоги.

Як зазначалося вище, з точки зору взаємодії людини з комп'ютером, розпізнавання жестів вважається найбільш інтуїтивно зрозумілим і природним, тому його розвиток постійно розвивається, залежно від удосконалення датчиків, які використовуються для захоплення жестів. У цьому сенсі розпізнавання жестів має істотне значення. Еволюція від інтуїтивної ідентифікації до більш формальної ідентифікації, заснованої на вдосконаленні сенсорних експериментів, що використовуються для цієї мети. Спочатку, у 1960-х роках, дослідники почали використовувати планшети та спеціальні олівці для запису написів, сенсорних інтерфейсів або вказівних пристроїв. Потім, у 1969 році, інженер Майрон Крюгер почав працювати над прототипом віртуальної реальності, уявляючи майбутнє без екранів, у якому люди безпосередньо взаємодіють із навколишнім середовищем. Цей підхід відтепер називатиметься природною взаємодією, а ці інтерактивні інтерфейси – природними інтерфейсами користувача.

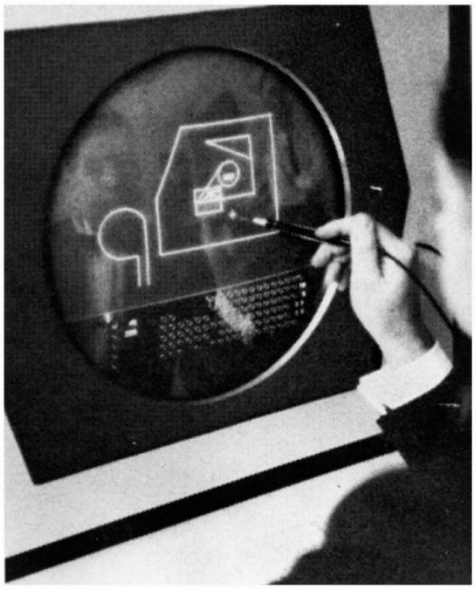

Рисунок 1.2.1 – Малювання на дисплеї в 1963 році

Захоплення руху не почало використовувати рукавички з датчиками згинання та положення до 1980-х років. У 1990-х роках робота над розпізнаванням жестів зображень і відео за допомогою комп'ютерного зору збільшилась і продовжує вдосконалюватися до сьогодні, коли доступні системи відстеження тіла в реальному часі. Крім комп'ютерного зору, існують електромагнітні системи, які визначають місцезнаходження об'єктів шляхом вимірювання електромагнітних полів, створюваних передавачами, наприклад радіочастот.

Ці пристрої носять на руках для вимірювання їх положення та руху. Зазвичай вони мають тактильні або інші типи вбудованих датчиків, які забезпечують точне відображення рухів усіх різних фаланг і суглобів зап'ястя. Це має велику перевагу, оскільки для отримання дескрипторів не потрібно жодних етапів обробки даних, наприклад, у випадку зображень, отриманих камерою. Їх недолік в тому, що вони дорогі і незручні для користувачів. Їх можна розділити на дві категорії: активні та пасивні. З одного боку, активні системи включають всі подібні рукавички, які містять певний тип датчика або акселерометра. У минулому ці типи рукавичок для передачі даних

підключалися до комп’ютера за допомогою кабелю, тепер ця функція зникла завдяки бездротовим технологіям.

З іншого боку, пасивні або ненав’язливі рукавички відносяться до неелектронних пристроїв, які містять певні кольорові позначки для полегшення ідентифікації зображення.

Рисунок 1.2.2 - Пасивні рукавички, які допомагають розрізняти положення пальців

Першою, у 1977 році, була рукавичка Sayre, заснована на гнучкій трубці (а не на оптичному волокні), яка вимірює кривизну пальця, випускаючи світловий промінь на одному кінці, тоді як датчик (фотодіод) виявляє кривизну пальця. пальця на основі кривизни та положення пальця Інтенсивність випромінюваного променя.

У 1980 році Медіалабораторія Массачусетського технологічного інституту почала використовувати рукавички, схожі на рукавички Сейра, але вони були більше зацікавлені в розробці пристроїв для захоплення руху, ніж в інструментах керування.

На жаль, технологія на той час була недостатньо розвиненою, щоб створити справді ефективний пристрій для введення, тому використання таких рукавичок незабаром припинилося.

У 1983 році була запатентована перша рукавичка, яка фактично розпізнавала положення руки для створення буквено-цифрових символів, а лише через чотири роки була випущена перша рукавичка віртуальної реальності. Фізично DataGlove складається з легкої рукавички з оптичними волокнами, прикріпленими до тильної сторони пальців. У цій моделі згинання пальців згинає волокна, погіршуючи їх світлопроникність. Потужність сигналу кожного волокна належним чином надсилається до процесора, який потім визначає кут з'єднання на основі попереднього калібрування кожного користувача.

Пізніше, у 1989 і 1995 роках, з'явилися рукавички Power Glove і Super Glove, в обох використовувалися матеріали зі змінним опором для вимірювання згинання пальців. Вони дешевші, але в той же час менш точні. З додаванням магнітних датчиків і акселерометрів почали з'являтися численні приклади цих пристроїв як для комерційних, так і для наукових цілей. На даний момент CyberGlove є найвідомішою моделлю, розробленою для графічної анімації та кіноіндустрії. Крім того, MIT AcceleGlove широко використовується у рамках віртуальної реальності.

Рисунок 1.2.3 - CyberGlove III в основному використовується для керування роботами.

Розробка експериментів із використанням рукавичок для даних дозволяє використовувати їх у різних сферах, від символічного розуміння чи класичних програм, таких як розваги та віртуальна реальність, до інших, таких як медицина, робототехніка чи промислове виробництво.

Подібно до активних рукавичок, браслети також намагаються уникнути голосової взаємодії або відволікаючих факторів, з якими можуть зіткнутися системи комп'ютерного зору. Однак його робота базується не на акселерометрах чи оптичних волокнах, а на акселерометрах чи оптичних волокнах. Він ґрунтується на датчиках електроміографії (ЕМГ). Це означає, що він намагається виміряти електричний потенціал, створений активністю м'язових клітин. Серед цих технологій розпізнавання ведуться активні дослідження на основі технологій розпізнавання біосигналів: сигнали електроокулограми (ЕЕГ: електрична активність рухів очей або «положення погляду», зареєстрована навколо очей), сигнали електроенцефалограми (ЕЕГ: електрична активність мозку). Запис із шкіри голови), сигнал електрокардіограми (ЕКГ: електрична активність серця), ЕМГ тощо.

Електроміографія заснована на вивченні нервово-м'язової системи, тому вона відповідає за виявлення, аналіз і обробку електричних сигналів, що генеруються м'язами та нервами за допомогою електродів.

М'язи складаються з групи спеціалізованих клітин, які можуть скорочуватися та розслаблятися, тому вони відповідають за такі функції, як створення руху, транспортування речовин у тілі або забезпечення адекватної стабільності та температури. Крім того, можна виділити три типи м'язової тканини на основі їх будови, скорочувальних властивостей і механізмів контролю: (а) скелетний м'яз, (б) гладкий м'яз і (в) серцевий м'яз. Як випливає з назви, скелетний м'яз прикріплюється до кісток і допомагає рухати тіло, гладка мускулатура розташована в кишечнику і відповідає за транспортування речовин, і, нарешті, серцевий м'яз відповідає за биття серця. Сигнали ЕМГ надходять від скелетних м'язів.

Слід зазначити, що біопотенціал, який генерують живі організми, насправді є іонним потенціалом, який створюється потоком іонного струму. Тому для ефективного вимірювання цих іонних потенціалів їх потрібно перетворити на електричні, перш ніж їх можна буде виміряти звичайними методами. Пристроями, відповідальними за цей перехід, є електроди, які сприймають внутрішньом'язову або поверхневу напругу. Електроди ЕМГ здатні реєструвати потенціали від усіх м'язів у межах їх досяжності. Це означає, що при спробі виміряти ЕМГ малих м'язів потенціали сусідніх великих м'язів можуть заважати, навіть якщо електроди розміщені безпосередньо на цих малих м'язах. Таким чином, у цих випадках утворюваний сигнал зазвичай є результатом суми всіх цих індивідуальних потенціалів, вироблених волокнами, які складають досліджуваний м'яз. Якщо це викликає проблеми, потрібні голчасті електроди, вставлені безпосередньо в м'яз, оскільки вони можуть контактувати з м'язовими волокнами окремо. Таким чином, внутрішньом'язова або голкова ЕМГ використовується для вивчення фізіології або патології рухових одиниць, тоді як поверхневий тип (SEMG) більше підходить для вивчення типів поведінки м'язів, моделей тимчасової активності або м'язової втоми.

Такі фактори, як відстань між електродами, площа м'язової активності, властивості шкіри, обробка сигналу, є важливими для аналізу ЕМГ оскільки амплітуда та її властивості безпосередньо залежать від усього цього.

Як правило, сигнали, отримані електродами, повинні бути посилені та оброблені. При цьому амплітуда сигналу і ширина смуги частот можуть змінюватися в залежності від розміру і типу електродів і відстані між ними.

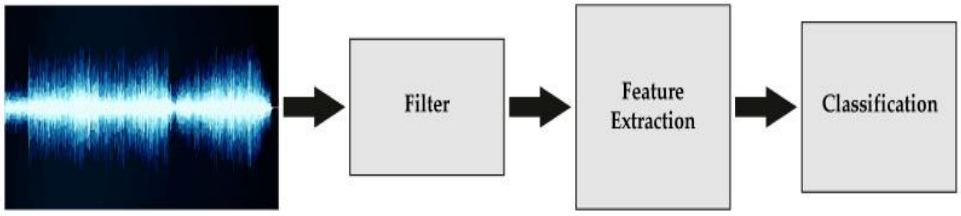

Рисунок 1.2.3 - Різні фази обробки сигналів електроміографії (ЕМГ).

Амплітуда ЕМГ-сигналу змінюється від 10 мкВ до 5 мВ, а його смуга пропускання змінюється від 10 Гц до 10 кГц, залежно від конфігурації електрода. Комерційно він використовує електроди з хлориду срібла (Ag–Ag - Cl) завдяки їх стабільності та зменшенню шуму. Що стосується кількості необхідних електродів, то потрібно не менше трьох. Два з них утворюють диференціальну вхідну пару, а третій відповідає землі. Інколи додають активні або активовані електроди для створення ланцюга керування зворотним зв'язком між датчиком і тілом, таким чином усуваючи потребу в електропровідному гелі між електродами та шкірою (його використання зменшує електричний опір). Традиційно технологія ЕМГ використовується в медичній діагностиці, протезному контролі та техніці медичної реабілітації. У контексті взаємодії людини з комп'ютером (HCI) з'явилося кілька досліджень, які зосереджуються на використанні вхідних даних ЕМГ як альтернативного каналу для користувачів з обмеженими фізичними можливостями, оскільки діяльність одного або кількох м'язів можна графічно пояснити за допомогою інтерфейсу користувача. Інтерфейс (GUI) є як постійним (як заміна миші), так і статичним із використанням розпізнавання жестів. Інші приклади додатків EMG від HCI включають роботизоване керування, розпізнавання немовлення, розпізнавання емоційного стану, інтерфейси музичної експресії та загальне розпізнавання жестів, використовуючи їх для керування музичними плеєрами, як покажчики GUI або цифрові клавіатури.

Що стосується існуючих продуктів на ринку, завдяки останнім розробкам у сфері бездротового зв'язку та інтегрованих обчислень ми можемо знайти портативні пристрої, які широко використовуються для отримання даних ЕМГ за допомогою браслетів. Вони містять ряд датчиків ЕМГ, радіально розподілених по колу гнучкої стрічки, що забезпечує відносно щільне прилягання. Яскравим прикладом є браслет Myo виробництва Thalmic Labs, який використовувався в багатьох програмах і експериментах.

Наступним метод який я розгляну є ультразвук. Можна виділити дві методики захоплення жестів. В одному з них використовуються ультразвукові зображення (контрастна сонографія).

В іншому використовується ефект Доплера, що забезпечує ультразвукове випромінювання в будь-якому приміщенні. Технологія передбачає використання ультразвукових зображень, які дозволяють у реальному часі переглядати м'язи всередині тіла. Це досягається завдяки тому, що наші тканини мають різний акустичний опір, тому, коли звукові хвилі проходять від однієї тканини з різним акустичним опором до іншої, різна кількість енергії відбивається для формування ультразвукового зображення. Це навряд чи забезпечить більшу точність, ніж інші системи, які безпосередньо вимірюють частини тіла в русі, такі як камери або рукавички для обробки даних, але, як і у випадку з методом ЕМГ, існує потенційна перевага з точки зору втрати інформації через оклюзію. У цьому випадку жодна частина тіла не буде прихована від інших. Крім того, на відміну від ЕМГ, ехоміографія не досліджує труднощі розрізнення окремих м'язів (для неінвазивних методів) від м'язів, розташованих глибоко в передпліччі, що обмежує точність, з якою методи ЕМГ можуть фіксувати різні рухи руки.

Ще один метод фіксації жестів за допомогою сигналів ультразвукової частоти заснований на управлінні ефектом Доплера. В основному він складається з передавача, який випромінює безперервний ультразвуковий тон, який відбивається від об'єктів у межах діапазону виявлення (руки та руки під час розпізнавання жестів). Потім сигнал, відбитий датчиком, фіксується для належного аналізу.

У статичному середовищі зворотні сигнали мають однакову частоту, але різні фази й амплітуди. Однак, якщо об'єкт рухається, його відлуння змінюватиме свою частоту (доплерівський зсув), створюючи компоненти на інших частотах, пропорційних його швидкості відносно датчика.

У випадку кількох об'єктів, що рухаються з різними швидкостями, сигнал зворотного зв'язку міститиме кілька частот, по одній для кожного

рухомого об’єкта. Це той самий принцип, який використовує радар поліції для контролю швидкості.

Однак тут замість РЧ використовується ультразвук, і ультразвуковий датчик спостерігає повний розподіл швидкості в полі зору. Навпаки, поліцейський радар може визначити лише швидкість об’єкта.

Хоча ультразвукові далекоміри широко використовуються на ринку, дослідження щодо застосування цієї технології для ультразвукового розпізнавання жестів широко не повідомлялося.

Наступний спосіб заснований на системах Wi-Fi. Для повторного використання існуючої інфраструктури Wi-Fi, економії відповідного використання апаратного забезпечення та можливості легкого масштабного розгортання системи розпочато пошук розпізнавання жестів на основі мереж Wi-Fi. Ще однією перевагою цієї технології є можливість забезпечити розпізнавання жестів навіть у сценаріях поза межами прямої видимості (NLOS) (без прямої видимості), про що більш детально описано в розділі про камеру.

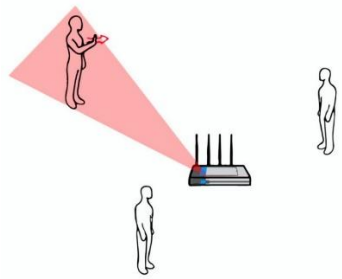

Рисунок 1.2.5 - WFI розпізнавання різних положень рук.

Також в таких системах використовуються різні індикатори для отримання всієї інформації, необхідної для розпізнавання жестів. Наприклад, деякі дослідження базуються на індикаторі потужності прийнятого сигналу (RSSI), часу польоту (ToF) або доплерівському зсуві. Однак ці системи або вимагали спеціального обладнання, або були модифіковані за допомогою наявного комерційного обладнання, деякі з яких були надто чутливими до перешкод. Тому поточні дослідження більше зосереджуються на спостереженні інформації про стан каналу (CSI).

Завдяки ортогональному поділу частот (OFDM) ця інформація надходить з одного потоку сигналу, що забезпечує точніші показання. Таким чином, CSI виявився більш надійним показником навіть за наявності перешкод у приміщенні.

На основі використання RRSI Абдельнассер створив систему розпізнавання жестів за допомогою Wi-Fi. Система може розпізнавати кілька жестів і відображати їх для моніторингу різних рухів з точністю від 87,5% до 96%. Для виконання цього завдання він використовує просту точку доступу та три антени.

Системи RFID зазвичай складаються з надвисокочастотних (UHF) комерційних зчитувачів, здатних виявляти мітки на відстані навіть десятків метрів, залежно від потужності передачі. Ідентифікаційні мітки можуть мати або не мати, залежно від застосування RFID, і можуть працювати з акумуляторами (активними чи пасивними) або без них. Як правило, RFID-мітки використовуються для управління ланцюгом поставок і автоматичної ідентифікації об’єктів. Вони також використовуються для розширення цифрових інформаційних середовищ, моніторингу людської діяльності в приміщеннях або виявлення взаємодії людини з об’єктами, позначеними RFID. Як і у випадку з сигналами Wi-Fi, відповідну інформацію, таку як зміни фази, RSSI та доплерівські зрушення, можна отримати з сигналів УВЧ, надаючи потенційні можливості для розпізнавання жестів. Перевагою використання пасивних технологій RFID є низька вартість, але логічно, оскільки вони не використовують власне джерело енергії, дальність виявлення зменшується до сантиметрів.

Через цю властиву помилку пов’язані дослідження виявлення жестів зосереджені на виявленні взаємодії людини з комп’ютером або тактильних інтерфейсів. Активні теги зазвичай використовуються в додатках відстеження, визначення місцезнаходження та ідентифікації через їх великий радіус дії.

Найпоширенішими пристроями захоплення жестами сьогодні, безумовно, є камери RGB і RGB-d.

Сучасні цифрові камери вловлюють світло через три основні канали червоного, зеленого та синього світла (звідси скорочення червоний (R), зелений (G) і синій (B)). Отримане зображення організоване як масив пікселів, кожен піксель має три власні значення RGB. Отже, будь-яка спроба розпізнати об’єкти на зображенні призведе до створення алгоритму комп’ютерного зору, де, по суті, порядок і інтенсивність кольорів різних пікселів надають значення. Камера глибини, також відома як RGB-d, є традиційною камерою. Вони мають четвертий канал для відображення інформації про глибину, відстані між фокусною точкою та об’єктом зйомки. Для отримання інформації про глибину для створення тривимірного зображення використовуються три різні методи.

Наступний метод базується на камерах із технологією Time of Flight. Ці камери можуть безпосередньо оцінювати 3D-структуру без допомоги будь-яких алгоритмів штучного зору. Його робота заснована на модулюванні випромінювання джерела світла, зазвичай в інфрачервоному (ІЧ) спектральному діапазоні, для освітлення сцени. Відстань визначається шляхом вимірювання часу проходження сигналу від джерела світла до сцени та назад до сенсорного елемента. Проблема проектування системи, здатної розпізнавати жести для керування додатком, зазвичай виникає на наступних етапах: збір і обробка даних (зйомка камерою, обробка та сегментація), моделювання знятих об’єктів (вибір і керування функціями об’єкта), визначення жестів (статика та динаміка), їх виявлення (розділення пози та виявлення руху) та контроль переходу (автомат стану).

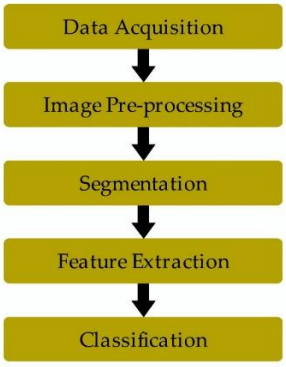

Рисунок 1.2.5 – Етапи роботи систем на основі зору.

Кожен із попередніх розділів містить низку досліджень, залежно від підходу, обраної сегментації та класифікації та, можливо, використання кількох камер для розгляду оклюзії об’єкта (руки та руки у випадку захоплення жестами). Кілька яскравих прикладів — Коен і Лі, які класифікують пози тіла за допомогою методів опорного вектора (SVM) для гарнітури 3D-бачення, створеної з набору вхідних даних. Система повертає класифіковане положення тіла як мініатюрне зображення. Крім того, ChengMo вказує на системи, які розпізнають положення людини, такі як ходьба, нахил або сидіння, щоб аналізувати поведінку людини. Крім того, Corradini може аналізувати та розпізнавати місцезнаходження за допомогою гібридної нейронної мережі.

Також, наприклад, Раптіс запропонував систему класифікації, яка розпізнавала танцювальні жести з точністю 96,9 відсотка на основі руху шести суглобів, зібраних у реальному часі камерою Kinect.

Хоча ці 3D-камери використовувалися в системах розпізнавання жестів, вони традиційно були дорогими. З моменту появи пристрою Microsoft Kinect вартість різко впала, що робить його дуже привабливим варіантом для багатьох дослідників. Система повертає класифіковане положення тіла як мініатюрне зображення.

Microsoft нещодавно випустила нову версію пристрою Kinect під назвою Azure Kinect DK. Якщо попередня версія була в основному орієнтована на ігри, цей новий пристрій призначений для професійного використання, охоплюючи такі ринки, як логістика, робототехніка та охорона здоров’я. Azure Kinect DK — це набір для розробки та периферійний пристрій для ПК, який використовує розширений штучний інтелект (ШІ), датчики для вдосконаленого комп’ютерного зору та моделі мовлення. Він поєднує в собі датчик глибини та просторову мікрофонну матрицю з камерою 4K RGB та інерційним вимірювальним блоком. Весь пристрій використовує кілька режимів, параметрів, SDK і підтримує хмарні служби Azure. Microsoft також пропонує

новий набір інструментів для відстеження скелета, який може визначати 3D-координати людських суглобів.

Функція відстеження тіла забезпечує сегментацію тіла, анатомічний скелет кожної частини або всього тіла, унікальну ідентифікацію кожного тіла та можливість відстежувати тіла з часом. Це дорогий пристрій з дуже низькими вимогами. Крім того, це новий пристрій, і наукових публікацій, пов’язаних з мовою жестів, не знайдено, а деякі фірмові рішення від приватних компаній.

Наступним розглянемо портативний пристрій для захоплення жестів користувача під назвою Leap Motion Leapt Motion — це невеликий і доступний пристрій для розпізнавання жестів у реальному часі, який може відстежувати пальці, руки та передпліччя в 3D.

Його найкраще використовувати для інтерактивних програм і він відносно простий у використанні. Leap Motion також доступний, компактний і його легко носити з собою.

Рисунок 1.2.6 – Приклад роботи Leap Motion.

Він містить дві інфрачервоні камери 120° і три інфрачервоні світлодіоди, тому його можна класифікувати як камеру ToF відповідно до попереднього розділу про типи камер. Ці датчики працюють на довжині хвилі 850 нанометрів, тобто вони працюють у спектрі, невидимому для людського ока, і використовують частоту дискретизації до 200 кадрів в секунду (кадрів за секунду), адаптуючи своє освітлення до виявленого світла, таким чином прояснюючи потік. Визначайте регіони та забезпечуйте постійну роздільну здатність зображення.

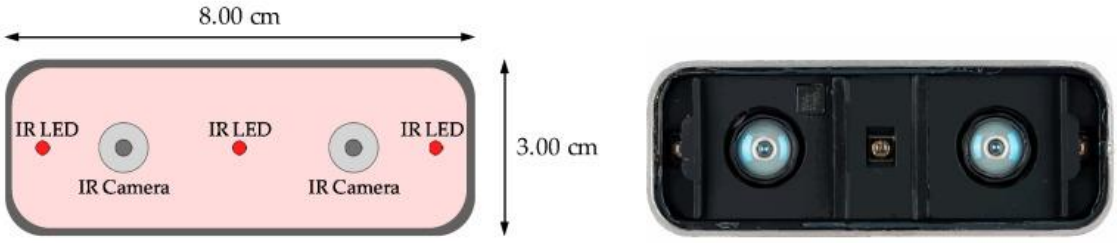

Рисунок 1.2.7 – Різні компоненти апаратної системи Leap Motion.

Неважко уявити можливості пристрою для полегшення дослідження за межами VR та ігор. Про це свідчать численні публікації з використанням Leap Motion для статичного та динамічного розпізнавання жестів. Наприклад, у 2014 році Мохандес успішно зафіксував 28 символів арабського алфавіту з показником успіху 99% за допомогою багатошарової нейронної мережі (MLP) із класифікатором Наве Байєса. Логічно, що, як і інші описані вище технології, використання Leap Motion поширилося на інші сфери, такі як робототехніка, медична реабілітація, домашня автоматизація, ідентифікація та автентифікація, музика чи освіта.

З моменту появи Sketchpad у 1960-х роках і до сьогодні багато досліджень і зусиль було витрачено на створення пристроїв, які розпізнають жести. Найважливіша еволюція спостерігається з 1980-х років, коли вдосконалення напівпровідникових технологій дозволило створити датчики, які розпізнають різні рухи рук, передаючи їх системам керування.

Досить поширеним прикладом є використання різних типів рукавичок (DataGlove, PowerGlove, SuperGlove, CyberGlove, AcceleGlove). Що стосується обладнання, слід зазначити, що ці зразки спочатку були інвазивними і не дуже універсальними, хоча їх якість і точність, безумовно, покращилися з роками. Це вдосконалення досягається не тільки типом використовуваних датчиків, але й розробкою програмного забезпечення для управління інформацією.

У другому десятилітті 21 століття був випущений Kinect. Ця віха є важливою революцією в області розпізнавання 3D-жестів. Датчик Kinect — це група камер ToF, яка виявляє рух людини та передає силует людини та її рух у реальному часі.

Пристрій не розроблений спеціально для визначення рухів рук, тому він не ідеальний для використання з мовою жестів. Крім того, їх розмір значно зменшує їх універсальність як мобільних пристроїв.

1.3 - Бібліотека MediaPipe. Характеристика

MediaPipe — це платформа з відкритим вихідним кодом для створення конвеєрів комп’ютерного зору на довільних сенсорних даних, таких як відео чи аудіо. Використовуючи MediaPipe, цей перцептивний конвеєр можна побудувати як граф модульних компонентів. Наразі MediaPipe знаходиться в альфа-версії 0.7, і в API ще можуть бути критичні зміни. Для версії 1.0 потрібен стабільний API.

MediaPipe пропонує революційні продукти та послуги, якими ми користуємося щодня. На відміну від енергоємних фреймворків машинного навчання, MediaPipe вимагає мінімальних ресурсів. Він настільки малий і ефективний, що його можуть запускати навіть вбудовані пристрої IoT.

У 2019 році публічний випуск MediaPipe відкрив новий світ можливостей для дослідників і розробників. MediaPipe розроблений для

команд машинного навчання (ML) і розробників програмного забезпечення, які впроваджують готові до виробництва програми ML, або для студентів і дослідників, які випускають код і прототипи як частину своєї дослідницької роботи.

Фреймворк MediaPipe в основному призначений для швидкого створення прототипів конвеєрів виведення з використанням моделей AI для виведення та інших компонентів, які можна повторно використовувати. Це також полегшує розгортання додатків комп'ютерного зору в демонстраційних версіях і додатків на різних апаратних платформах. Мова конфігурації та інструменти оцінки дозволяють команді поступово покращувати конвеєр комп'ютерного зору. Набір інструментів MediaPipe складається з фреймворків і рішень. На наступній діаграмі показано компоненти набору інструментів MediaPipe.

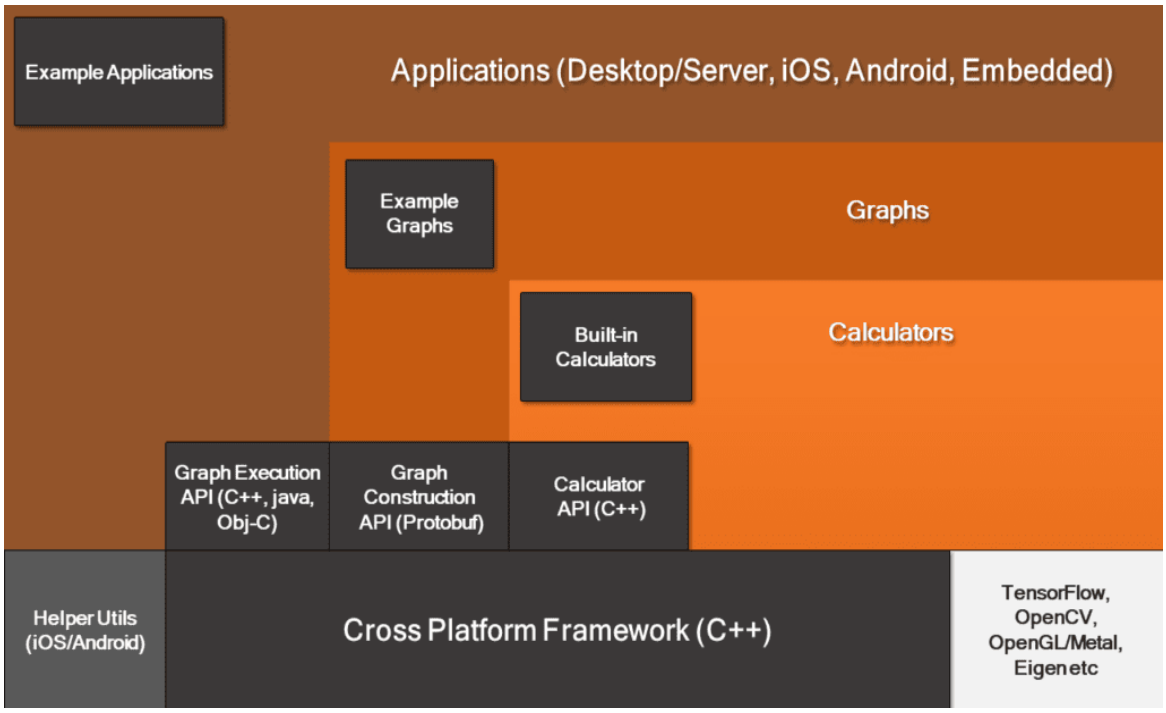

Рисунок 1.3.1 – Framework MediaPipe.

У MediaPipe також додано калькулятор. Це незалежні обчислювальні пристрої, написані мовою C++ із призначеними завданнями для виконання. Пакети даних (відеокадри або аудіосегменти) входять і виходять з порту калькулятора. Коли калькулятор ініціалізовано, він оголошує тип корисного

навантаження пакетів, які проходитимуть через порт. Фреймворк реалізує методи Open, Process і Close на калькуляторі кожного разу, коли виконується графік. Відкрийте калькулятор завантаження; повторіть процес у міру надходження пакетів. Процес закривається після повного виконання графіка.

Калькулятор ImageTransform отримує зображення на вхідний порт і повертає перетворене зображення на вихідний порт. З іншого боку, другий калькулятор, ImageToTensor, приймає зображення як вхідні дані та виводить тензор. Усі калькулятори вбудовані в MediaPipe, і їх можна розділити на чотири категорії. Калькулятори попередньої обробки — це набір калькуляторів обробки зображень і медіа.

ImageTransform і ImageToTensors на зображенні вище належать до цієї категорії. Logical Reasoning Calculator забезпечує вбудовану інтеграцію з Tensorflow і Tensorflow Lite для логічних міркувань. Калькулятори постобробки виконують такі завдання пост обробки ML, як виявлення, сегментація та класифікація.

TensorToLandmark — це калькулятор постобробки. Допоміжні калькулятори — це сімейство калькуляторів, які виконують кінцеві завдання, наприклад анотації зображень. Інтерфейси API калькулятора дозволяють вам написати свій власний калькулятор.

MediaPipe підтримує мультимодальні графіки. Щоб прискорити обробку, різні калькулятори працюють у різних потоках. Багато вбудованих калькуляторів мають параметри прискорення GPU для оптимізації продуктивності. Робота з даними часових рядів повинна бути правильно синхронізована, інакше система зависне. Розклади забезпечують коректну обробку потоків на основі часових позначок пакетів. Фреймворк забезпечує синхронізацію, спільний доступ до контексту та взаємодію з калькуляторами ЦП.

MediaPipe покладається на OpenCV для обробки відео та FFMPEG для обробки аудіо. Він також має інші залежності, такі як OpenGL/Metal, Tensorflow, Eigen тощо.

MediaPipe Visualizer забезпечує простий спосіб випробувати всі рішення. У MediaPipe текстовий файл protobuf (.pbtxt) визначає графік. Сторінка привітання MediaPipe Visualizer вітає вас файлом protobuf, що містить порожні графічні клітинки. У ньому є готові діаграми для різних рішень, які можна завантажити за допомогою кнопки «Новий» у верхньому правому куті.

Рішення MediaPipe не завжди працюють у реальному часі. Рішення побудовано на основі MediaPipe, яка надає API калькулятора (C++), API креслення (Protobuf) і API креслення (C++, Java, Obj-C). За допомогою API ми можемо будувати наші графіки та писати власні калькулятори. Набір інструментів чудовий, але вона залежить від основного обладнання.

1.4 Аналіз завдання магістерської роботи та постановка задач дослідження

Метою роботи є розроблення адаптивного алгоритму управління звуком комп’ютера за допомогою жестів із використанням бібліотеки MediaPipe а також дослідження можливостей технології управління комп’ютерними системами за допомогою жестів.

Для реалізації поставленої мети потрібно виконати наступні завдання:

- проаналізувати технологію управління жестами;
- проаналізувати бібліотеку MediaPipe;
- розробити алгоритм управління звуком за допомогою жестів на основі бібліотеки MediaPipe;
- здійснити програмну реалізацію розробленого алгоритму;

1.5 Висновки до розділу

На основі аналітичного підходу проведено аналіз технології управління комп’ютерними системами на основі жестів, що дало змогу виділити їх переваги та недоліки на етапі опрацювання.

Проведено аналіз історії розвитку технології управління комп’ютерними системами на основі зчитування жестів, що дозволили проаналізувати основні етапи імплементації даної технології з часом.

Проведено дослідження бібліотеки MediaPipe, що дозволило дослідити її основні характеристики і можливості, а також історію її розвитку.

2. АЛГОРИТМИ УПРАВЛІННЯ КОМП'ЮТЕРНИМ ЗВУКОМ З ДОПОМГОЮ ЖЕСТІВ

2.1 - Алгоритми розпізнавання зображень

Розпізнавання зображень — це завдання ідентифікації цікавих об'єктів на зображенні та розпізнавання, до якої категорії належить зображення. Розпізнавання зображень, розпізнавання фотографій і розпізнавання зображень — це терміни, які використовуються як синоніми. Коли ми візуально бачимо об'єкт або сцену, ми автоматично розпізнаємо ці об'єкти як різні екземпляри та пов'язуємо їх з різними визначеннями. Однак візуальне розпізнавання є дуже складним завданням для машин, які потребують великої обчислювальної потужності. Розпізнавання зображень за допомогою штучного інтелекту є давньою темою досліджень у галузі комп'ютерного зору. Хоча методи імітації людського зору змінювалися з часом, спільною метою розпізнавання зображень є класифікація виявлених об'єктів у різні класи (визначення, до якого класу належить зображення). Тому його ще називають розпізнаванням об'єктів. В останні роки машинне навчання, особливо методи глибокого навчання, досягли великих успіхів у багатьох задачах комп'ютерного зору та розуміння зображень. Таким чином, методи глибокого навчання для розпізнавання зображень досягають найкращих результатів з точки зору продуктивності (обчислення кадрів за секунду/FPS) і гнучкості.

У комп'ютерному зорі такі терміни, як сегментація, класифікація, розпізнавання та виявлення, часто використовуються як синоніми, а різні завдання збігаються. Хоча здебільшого це не проблема, але якщо ваш робочий процес вимагає від вас виконання певних завдань, все може стати безладним. Терміни розпізнавання зображень і комп'ютерний зір часто використовуються як синоніми, але насправді вони різні. Насправді розпізнавання зображень — це програма комп'ютерного зору.

Вона часто потребує виконання кількох завдань комп'ютерного зору, таких як виявлення об'єктів, розпізнавання зображень, класифікація зображень тощо.

Загальний підхід комп'ютерного зору до розпізнавання зображень — це послідовність фільтрації зображень, сегментації, виділення ознак і класифікації на основі правил. Однак традиційні методи комп'ютерного зору вимагають високого рівня досвіду, значного часу на розробку та містять багато параметрів, які необхідно визначати вручну, з дуже обмеженою можливістю перенесення на інші завдання.

Розпізнавання зображень за допомогою машинного навчання, з іншого боку, використовує алгоритми для вилучення прихованих знань із наборів даних хороших і поганих прикладів (див. Контрольоване та неконтрольоване навчання). Найпопулярнішим методом машинного навчання є глибоке навчання, де в моделі використовуються кілька прихованих шарів нейронних мереж. Впровадження глибокого навчання в поєднанні з потужним апаратним забезпеченням штучного інтелекту та графічним процесором уможливило великий прорив у розпізнаванні зображень. Досягніть надлюдської продуктивності та виявляйте об'єкти в реальному часі за допомогою глибокого навчання, класифікації зображень і алгоритмів розпізнавання обличь.

Існує кілька етапів, які складають основу роботи системи розпізнавання зображень. Для моделей розпізнавання зображень потрібні навчальні дані (відео, зображення, фотографії тощо). Нейронні мережі потрібно навчати на зображеннях зі згенерованих наборів даних, щоб створити уявлення про те, як виглядають певні класи. Наприклад, модель розпізнавання зображення (модель оцінки пози), яка розпізнає різні пози, вимагатиме кількох екземплярів різних поз людини, щоб зрозуміти, чим ці пози відрізняються одна від одної. Навчання нейронної мережі для розпізнавання зображень Зображення зі створеного набору даних подаються в алгоритм нейронної мережі. Це аспект глибокого або машинного навчання створення моделей розпізнавання зображень.

Навчальні алгоритми розпізнавання зображень можуть використовувати згорточні нейронні мережі для розпізнавання певних категорій розпізнавання зображень. Сьогодні існує кілька добре перевірених фреймворків, які широко використовуються для цих цілей. Тестування моделей штучного інтелекту Навчену модель потрібно перевірити на зображеннях, які не є частиною навчального набору даних. Це використовується для визначення зручності використання, продуктивності та точності моделі.

Таким чином, приблизно 80-90% повного набору даних зображень використовується для навчання моделі, а решта даних зберігається для тестування моделі.

Ефективність моделі вимірюється за набором параметрів, які представляють відсоток впевненості в точності тестового зображення, помилкової ідентифікації тощо. Поки GPU (блоки обробки графіки) не стали достатньо потужними, щоб підтримувати масові паралельні обчислювальні завдання нейронних мереж, традиційні алгоритми машинного навчання були золотим стандартом для розпізнавання зображень.

Рисунок 2.1.1 – Приклад комп’ютерного бачення в авіації

Розглянемо три найпопулярніші моделі машинного навчання для розпізнавання зображень. Векторні машини підтримки SVM створюють гістограми зображень, які містять і не містять об’єкт інтересу. Потім алгоритм бере тестове зображення та порівнює навчені значення гістограми зі значеннями з різних частин зображення, щоб перевірити збіг. Моделі набору функцій Моделі набору функцій, такі як Scale Invariant Feature Transform (SIFT) і Maximum Stable Extremal Region (MSER), працюють, беручи за посилання зображення, яке потрібно відсканувати.

Зразок фотографії об’єкта, який потрібно знайти. Потім модель намагається зіставити характеристики зразка фотографії з різними частинами цільового зображення піксель за пікселем, щоб побачити, чи знайде збіг.

Алгоритм Віоли-Джонса є широко використовуваним алгоритмом розпізнавання облич, який навіть передував CNN. Він працює шляхом сканування облич і виділення ознак, які потім передаються до розширеного класифікатора. Це, у свою чергу, створює низку доповнених класифікаторів, які використовуються для перевірки тестових зображень. Щоб знайти успішний збіг, тестове зображення має генерувати позитивний результат від кожного з цих класифікаторів.

У розпізнаванні зображень використання згорткових нейронних мереж (CNN) також відоме як глибоке розпізнавання зображень. CNN не поступаються традиційним методам машинного навчання. CNN не тільки швидші та дають найкращі результати виявлення в розпізнаванні зображень за допомогою машинного навчання, але вони також можуть виявляти кілька екземплярів об’єкта на зображенні, навіть якщо зображення трохи спотворене, розтягнуте або іншим чином змінено. У глибокому розпізнаванні зображень згорткові нейронні мережі навіть перевершили людей у таких завданнях, як класифікація об’єктів за невеликими категоріями, такими як певні породи собак чи птахів. Найпопулярніші моделі глибокого навчання, такі як YOLO, SSD і RCNN, використовують згорткові шари для аналізу цифрових зображень або фотографій. Під час навчання кожен згортковий шар діє як фільтр,

навчаючись розпізнавати деякі аспекти зображення перед тим, як передавати їх наступному. Один шар обробляє колір, інший — форму і так далі. Зрештою, при визначенні відповідності враховуються сукупні результати всіх цих рівнів.

Для розпізнавання зображень або фотографій є кілька кращих алгоритмів. Хоча всі ці алгоритми глибокого навчання відрізняються фундаментальним підходом до розпізнавання різних категорій об’єктів. Faster RCNN (Region-based Convolutional Neural Network) є найкращим із алгоритмів розпізнавання зображень сімейства R-CNN, включаючи R-CNN і Fast R-CNN. Він використовує регіональну пропозиційну мережу (RPN) для виявлення функцій і швидкий RCNN для розпізнавання зображень, що значно покращує його попередника (примітка: швидкий RCNN проти швидшого RCNN). Швидший RCNN може обробляти зображення менш ніж за 200 мілісекунд, а Faster RCNN займає 2 секунди або більше.

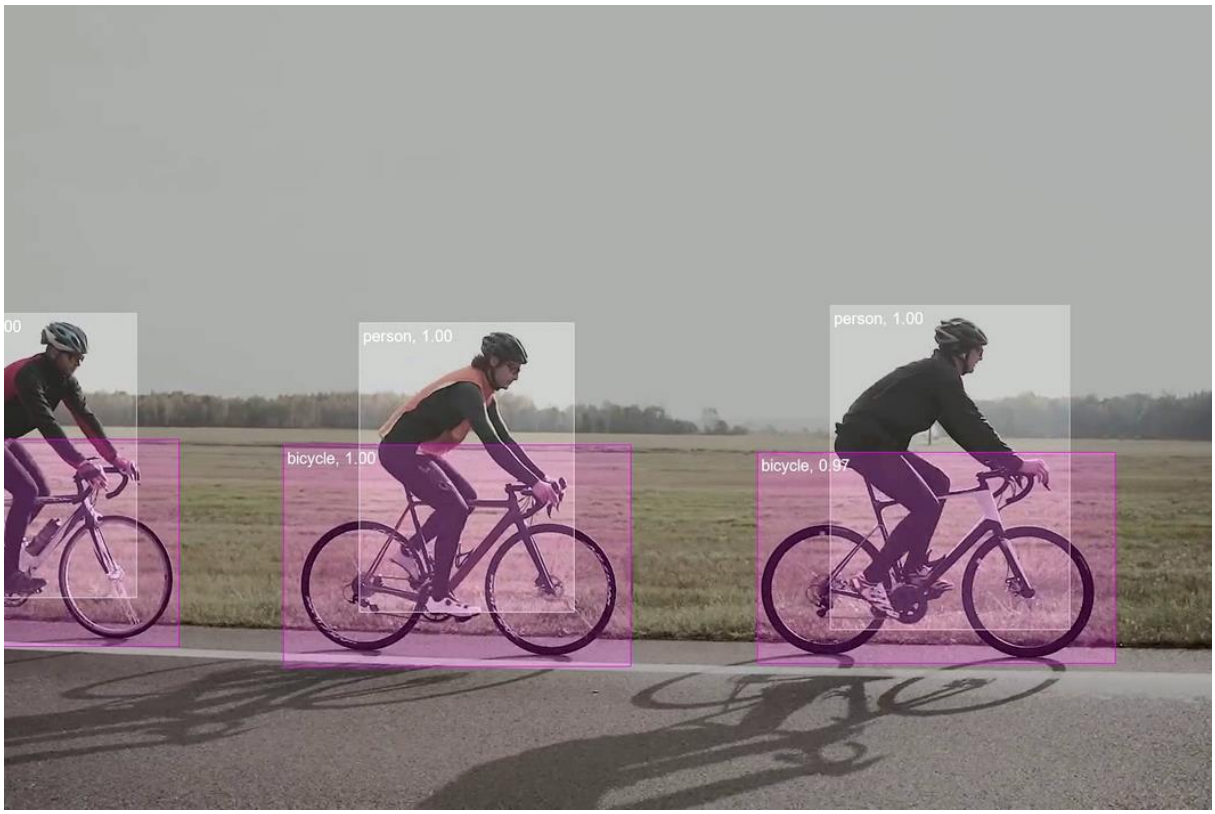

Рисунок 2.1.2 – AI-розпізнавання зображень із виявленням і класифікацією об’єктів за допомогою глибокого навчання

RCNN малює рамку навколо набору запропонованих точок на зображенні, деякі з яких можуть перекриватися. Одноразові детектори (SSD) дискретизують цю концепцію, розділяючи зображення на рамки за замовчуванням у формі сітки з різними пропорціями. Потім він об'єднує карти функцій, отримані в результаті обробки зображень із різними співвідношеннями сторін, щоб природним чином обробляти об'єкти різних розмірів. Це робить твердотільні накопичувачі дуже гнучкими, точними та простими в освоєнні. Реалізація SSD може обробляти зображення за 125 мілісекунд.

YOLO означає You Only Look Once. Як випливає з назви, алгоритм використовує фіксований розмір сітки для обробки кадру лише один раз, а потім визначає, чи містить кадр сітки зображення. Для цього алгоритми виявлення об'єктів використовують міри достовірності та кілька обмежувальних рамок у кожному кадрі сітки.

Однак він не включає численні співвідношення сторін або складність карт функцій, тому, хоча він може давати результати швидше, вони можуть бути дещо менш точними, ніж SSD. Однією з найпопулярніших моделей YOLO є його третя версія під назвою YOLOv3. Найтонша версія YOLO під назвою Tiny YOLO може обробляти відео зі швидкістю до 244 кадрів на секунду або 1 зображення кожні 4 мілісекунди.

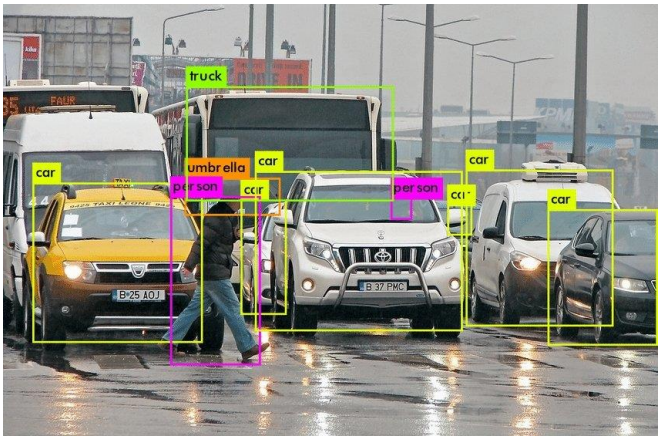

Рисунок 2.1.3 – Алгоритм розпізнавання зображень YOLOv3, застосований до фотографії щільної сцени.

Коли справа доходить до розпізнавання зображень, Python є мовою програмування, яку вибирають більшість спеціалістів із обробки даних та інженерів комп'ютерного зору. Він підтримує велику кількість бібліотек, розроблених для робочих процесів ШІ, включаючи виявлення та розпізнавання зображень. Щоб налаштувати комп'ютер для виконання завдань розпізнавання зображень Python, вам потрібно завантажити Python і інсталювати пакети, необхідні для виконання завдань розпізнавання зображень, включаючи Keras. Keras — це високорівневий API глибокого навчання для програм AI.

Він працює на TensorFlow/Python і допомагає кінцевим користувачам розгортати програми машинного навчання та ШІ за допомогою легкого для розуміння коду.

Якщо ваш комп'ютер не має відеокарти, ви можете скористатися безкоштовним екземпляром графічного процесора онлайн на Google Colab. Для цілей класифікації тварин ви можете знайти добре позначений набір даних під назвою «Animals-10» на Kaggle. Набір даних можна завантажити абсолютно безкоштовно. Отримавши онлайн-набір даних від Kaggle за допомогою маркера API, ви можете розпочати кодування на Python після повторного завантаження необхідних файлів на Диск Google.

Спеціальна модель для розпізнавання зображень — це модель ML, спеціально розроблена для конкретного завдання розпізнавання зображень. Це може включати використання спеціальних алгоритмів або модифікацію існуючих алгоритмів для покращення їх продуктивності на зображеннях (наприклад, перенавчання моделі). Хоча попередньо підготовлені моделі забезпечують потужні алгоритми, навчені на мільйонах точок даних, є багато причин, чому ви можете створити власні моделі для розпізнавання зображень. Наприклад, ваш набір даних зображення може сильно відрізнятися від стандартних наборів даних, на яких навчаються поточні моделі розпізнавання зображень. У таких випадках користувацькі моделі можна використовувати

для кращого розуміння характеристик даних і підвищення продуктивності. Або доведеться розробляти нову програму.

У якій поточні моделі розпізнавання зображень не можуть досягти необхідної точності чи продуктивності. Створення спеціальної моделі на певному наборі даних може бути складним завданням і вимагає збору високоякісних даних і анотацій зображень. Для цього потрібно добре розуміти як машинне навчання, так і комп'ютерний зір. Перегляньте нашу статтю про те, як оцінити продуктивність моделі машинного навчання.

API забезпечує простий спосіб розпізнавання зображень за допомогою виклику хмарних служб API, таких як Amazon Rekognition.

Подібним чином ви можете легко використовувати API Google Vision (Google Cloud) API для розпізнавання об'єктів у зображеннях для виконання таких завдань, як розпізнавання об'єктів або облич, розпізнавання тексту чи розпізнавання рукописного тексту. Якщо сценарій використання дозволяє розвантажувати дані (надсилати візуальні елементи на хмарний сервер), API розпізнавання зображень, такі як API виявлення об'єктів TensorFlow, є потужними інструментами для розробників, щоб швидко створювати та розгортати програмне забезпечення для розпізнавання зображень. Використовуйте API розпізнавання зображень, щоб отримати інформацію про саме зображення (класифікація зображення або розпізнавання зображення) або об'єкти, які воно містить (виявлення об'єкта). Чисто хмарні API комп'ютерного зору корисні для створення прототипів і невеликих рішень, які дозволяють розвантажувати дані (конфіденційність, безпека, законність), некритичні (з'єднання, пропускна здатність, надійність) і не в реальному часі (затримка, дані) громіздкі і дорого). Щоб подолати ці обмеження суто хмарних рішень, останні тенденції в розпізнаванні зображень зосереджуються на розширенні хмари шляхом використання периферійних обчислень і машинного навчання на пристрої. Щоб зрозуміти, як працюють API розпізнавання зображень, який з них вибрати та які обмеження API для завдань розпізнавання, я рекомендую вам переглянути наш огляд найкращих платних і безкоштовних API

комп’ютерного зору. У той час як API комп’ютерного бачення можна використовувати для обробки окремих зображень.

Системи периферійного штучного інтелекту використовуються для виконання завдань розпізнавання відео в реальному часі шляхом переміщення машинного навчання ближче до джерела даних (граничний інтелект). Це забезпечує обробку зображень штучним інтелектом у режимі реального часу, оскільки немає необхідності розвантажувати дані (завантажувати дані в хмару) під час обробки даних зору, забезпечуючи вищу продуктивність логічного висновку та надійність, необхідні для систем виробничого рівня.

2.2 - Використання алгоритмів машинного навчання для розпізнавання зображень

У 1959 році комп’ютерний науковець Артур Семюел, піонер у дослідженні штучного інтелекту, описав машинне навчання як «дослідження, яке дозволяє комп’ютерам навчатися без явного програмування». Фундаментальна стаття Алана Тюрінга (Turing, 1950) встановила еталонний стандарт для демонстрації машинного інтелекту, згідно з яким машини мають бути розумними та чуйними, що не відрізняються від людського інтелекту. Машинне навчання зазвичай поділяється на три типи: контрольоване навчання, неконтрольоване навчання та навчання з підкріпленням. Під час навчання під наглядом машина вивчає приклади та мітку або ціль для кожного прикладу. Мітки в даних допомагають алгоритму пов’язувати ознаки. Двома найпоширенішими завданнями машинного навчання під наглядом є класифікація та регресія.

Перш ніж модель можна буде проаналізувати за допомогою алгоритмів машинного навчання, необхідно правильно обробити зображення, які можна

розділити на кілька етапів. Сегментація є першою діяльністю, яка виконується під час машинного навчання. Вчити розпізнавати предмети.

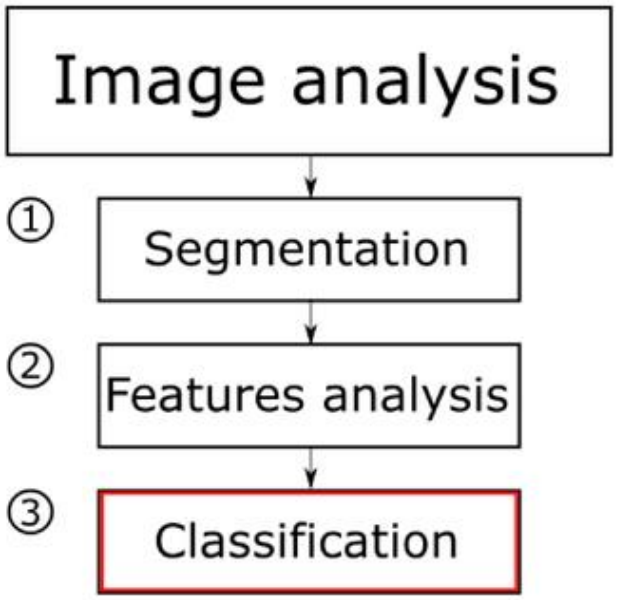

Рисунок 2.2.1 - Діаграма аналізу зображення

Тут зображення розділене на частини, які певним чином з’єднані між собою. Це для того, щоб спочатку ізолювати області, що належать цим об'єктам, межі яких утворюють або обмежують надсилання непотрібної інформації в пам'ять комп'ютера на наступному етапі. Наступним етапом є аналіз ознак зображення, які дають змогу ідентифікувати та описати атрибути об’єктів, які зазвичай помічає лише око. Характеристики об’єктів можна розділити на кілька основних груп, таких як геометричні, негеометричні та топологічні. Вибір особливостей зображення для аналізу є особистою справою і залежить від кінцевого результату. Однак для цілей цього алгоритму надзвичайно важливо мати справу з математичними аспектами ознак аналізу зображення, з яких формуються так звані сигнатури та скелети, тобто об’єкти, що представляють одновимірні функції контурів. Після виконання цих попередніх операцій процес розпізнавання об'єктів буде здійснюватися за допомогою штучних нейронних мереж, в яких можуть використовуватися методи глибокого навчання.

ANN — це назва математичної структури та її програмної або апаратної реалізації, що складається з окремих елементів, званих нейронами.

Нейрони здатні виконувати базові операції на своїх вхідних даних. Принцип роботи нейронів можна виразити наступним чином.

$$e = \sum_{i=1}^{n} x_i w_i$$

У цій функції xi — значення i-го вхідного сигналу, wi — ваговий коефіцієнт i-го сигналу, n — кількість нейронів у вхідному сигналі, e — загальне значення нейронної стимуляції, y є i-м значенням сигналу, який видає нейрон, а f є функцією активації.

Після обчислення суми e з використанням вагових коефіцієнтів wi і вхідного сигналу xi результат множиться на функцію активації f, яка повинна задовольняти наступним умовам: безперервність зміни між мінімальним значенням і максимальним значенням, безперервність похідна (похідну також не повинно бути важко обчислити). Використовується функція активації ReLU. Цей тип функції активації на сьогоднішній день є найпоширенішою функцією активації для навчання нейронних мереж, призначених для розпізнавання об’єктів на зображеннях. Це пов’язано з його нескінченним бажанням реагувати на дійсні сигнали та обнуленням значення нейронів для негативних сигналів. Такий підхід означає, що не всі нейрони використовуються в мережі, що захищає її від переобладнання та прискорює процес навчання мережі. Крім того, ReLU дуже просто обчислює похідні.

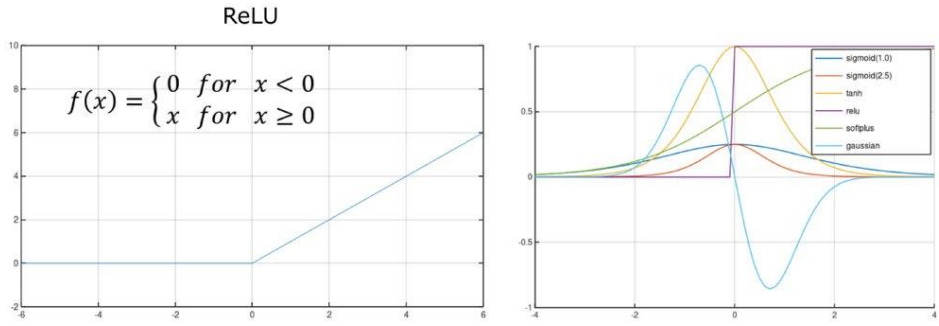

Рисунок 2.2.2 - Презентація функції активації ReLu та порівняння її першої похідної (справа) з похідними інших функцій активації

Оскільки для обробки та класифікації потрібні великі обсяги даних, створення програм для розпізнавання об’єктів на зображеннях потребує розробки математичних моделей та комплексних методів. Завдяки високій продуктивності одним із можливих рішень для розпізнавання об’єктів на зображеннях є використання ШНМ, де буде реалізований правильний алгоритм машинного навчання.

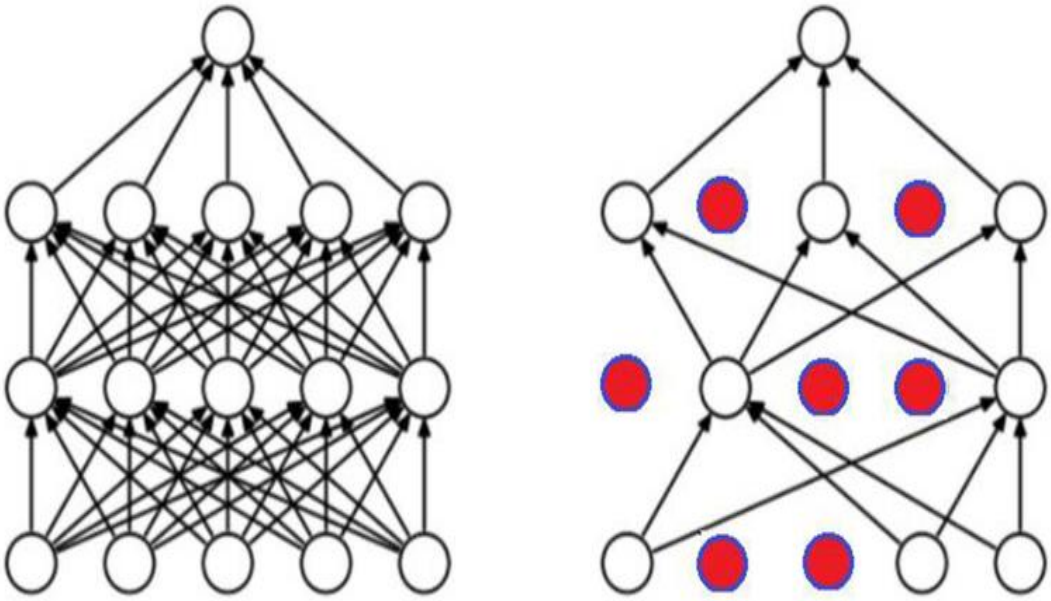

Рисунок 2.2.3 - Мережа з повністю пов’язаними шарами (ліворуч) і регуляризована мережа (праворуч)

Спочатку три рівні мережі (включаючи вхідний рівень) зменшують «розмірність» зображення завдяки використанню масок фільтрів, адаптованих до очікуваних функцій, а потім використовують ієрархічний шаблон ознак, що містяться в даних (декілька функцій відповідають за розпізнавання форм, а не всього зображення, оскільки деякі пікселі несуть більше інформації, ніж інші), що дозволяє використовувати меншу кількість нейронів і зв’язків між ними для досягнення того самого ефекту. Це особливо важливо у випадках, коли багаторівневі мережі чутливі до перевантаження даними. Повністю пов’язані шари між нейронами (ця мережа все ще односпрямована), це останні три шари обробки даних.

А також два шари з 4096 нейронами, що відповідають за математичні розрахунки. Використання цього алгоритму може прискорити роботу нейронної мережі, оскільки програміст виступає в ролі вчителя, знаючи, яке значення він очікує від нейронної мережі, і таким чином може визначити правильність результату алгоритму. Коли різниця надто велика, вона надає пропоновану зміну значення, яка дає змогу правильно ідентифікувати об’єкт і дає вказівку алгоритму виконати наступну ітерацію.

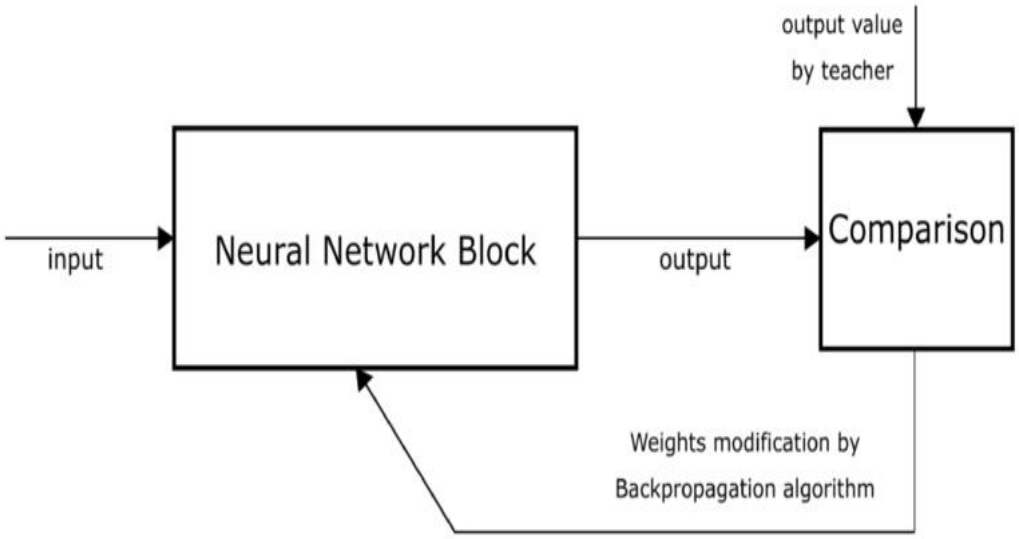

Рисунок 2.2.4 - Ідея навчання алгоритму методом зворотного поширення

Цей метод використовується для обчислення значення нейрона – для цього використовується математична формула. В основі його функціональності лежить використання знань про результати, які необхідно отримати на виході мережі.

Потім обчислюється помилка між запропонованим значенням і значенням, отриманим мережею, і помилка виправляється шляхом зміни значення на нейроні. Це просте завдання ускладняється багаторівневими мережами, основним типом яких використовується в більш просунутих алгоритмах штучного інтелекту. У цьому випадку алгоритм зворотного поширення підсумовує помилки нейронів прихованого шару перед останнім модифікованим шаром і лише потім коригує значення та ваги між ними.

Роботу алгоритму можна виразити наступним чином: Рандомізація вагових значень і їх призначення нейронам. Завантажте вхідні дані (вже в математичній формі). Введіть запропоноване вихідне значення та на його основі обчисліть значення нейронного виходу та порівняйте. Розрахунок похибки виходу мережі. Помилка в прихованому шарі обчислюється з вихідного шару з урахуванням суми помилок у вихідному шарі та помилки в прихованому шарі (зберігаючи вагові коефіцієнти між нейронами). Повторіть процес для наступного прихованого шару, використовуючи його загальну помилку та початкову помилку перед прихованим значенням, обчисленим на попередньому кроці.

Повторіть процес для наступного прихованого шару, використовуючи його загальну помилку та вихідну помилку попереднього прихованого шару, обчислену на попередньому кроці. Після обчислення похибок усіх прихованих шарів алгоритм таким же чином змінює значення нейронів вхідного шару. Змінено всі ваги в мережі. Помилка зменшується з кожним повним виконанням алгоритму, поки не досягне прийнятного рівня.

Модель нейронної мережі складається з кількох основних елементів. На додаток до використання типових рівнів нейронної мережі, що складається з нейронів і функцій активації.

Вона має елементи, відповідальні за оптимізацію роботи мережі, що дозволяє правильні обчислення та запобігає можливості повторного навчання, незважаючи на дуже великий обсяг даних. можлива нормалізація, кожна з яких має тенденцію посилювати збуджуючий нейрон і гальмувати сусідні нейрони в межах обмеженого діапазону значень.

Метод шукає найсильнішу реакцію нейрона та нормалізує відповіді сусідніх нейронів, роблячи вибрані нейрони більш чутливими до характеристик об’єкта. Якщо всі відповіді сусідніх нейронів достатньо великі, функція обмежить значення у всіх нейронах із заданого каналу, оскільки вони не ідентифікують жодного з них як особливо чутливого. Окрім нормалізації, ця функція також обмежує кількість нейронів, які використовуються для навчання.

Існує два типи нормалізації: у межах каналу (групи суміжних нейронів) і між каналами, що передбачає розгляд околиць у трьох вимірах замість двох.

Після нормалізації дані надсилаються до статистичного фільтра під назвою max pooling, який витягує максимальне значення з маски та зменшує кількість обчислень для наступних шарів. Застосування цього фільтра до субрегіонів, що не перекриваються, призводить до передачі лише максимальних значень, що значно зменшує обсяг даних, необхідних для подальшої обробки, без зниження ефективності алгоритму.

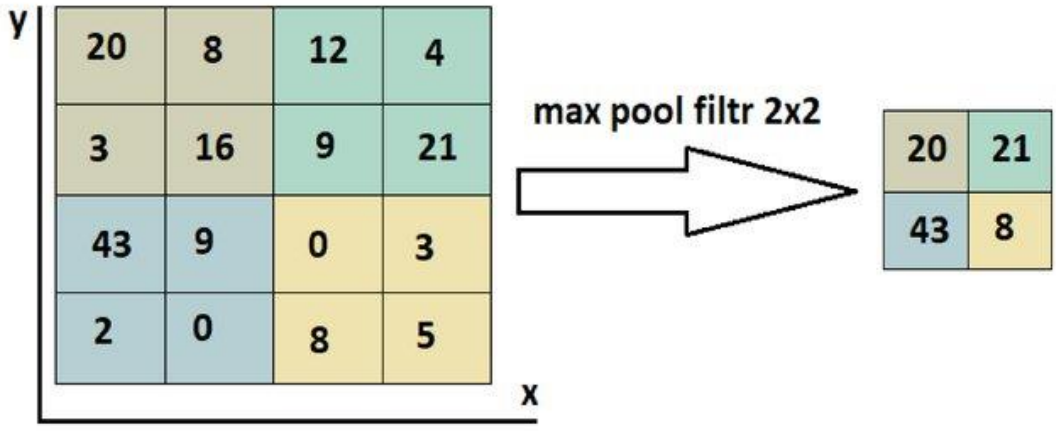

Рисунок 2.2.5 - Робота фільтра максимального об'єднання

Перш ніж інформація досягне мережевого виходу, вона все ще повинна пройти через функцію Softmax. Його завданням, як передавальною функцією, є зміна вхідного вектора з отриманого значення на вихідну інформацію, вихідна інформація має форму вектора з нормалізованим значенням між 0 і 1, так що вихідний рівень отримує певну інформацію які можна інтерпретувати як певну ймовірність, це допоможе визначити відсоток точності мережі під час аналізу.

Дуже важливо встановити темп навчання на курсі на належному рівні. Цей фактор визначає максимальне значення, яке може змінити вага нейрона.

Коли це значення занадто низьке, процес навчання мережі займає непропорційно багато часу, оскільки вимагає більше ітерацій для внесення виправлень, указаних викладачем. Однак, коли він занадто високий, корекція ваги, яка виконується на нейроні, настільки велика, що перешкоджає правильному процесу навчання. Фіксація ваг з великими значеннями означає, що обчислені значення нейронів, незважаючи на наступні ітерації, не зможуть адаптуватися до схеми мережі, що призведе до того, що вони будуть завжди випадковими, наприклад, на початкових етапах процес навчання. При виборі правильної швидкості навчання мережа навчається правильно і через кілька епох вже досягає потрібної точності розпізнавання.

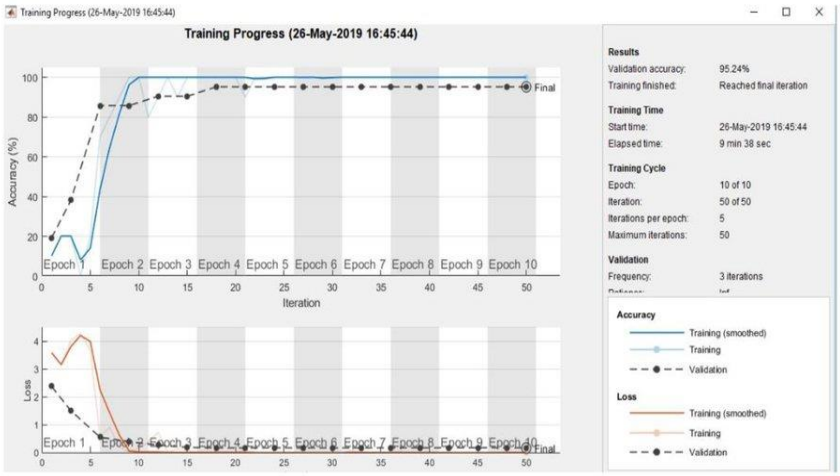

Рисунок 2.2.5 - Мережевий тест на швидкість навчання 1

Якщо значення цього фактора встановлено занадто високим, процес навчання мережі відбувається дуже швидко, що не завжди є позитивним явищем, оскільки мережа може зупинятися на локальних мінімумах і таким чином не досягати максимальної точності розпізнавання. Неправильний вибір ваг під час навчання мережі також може збільшити час її навчання.

З наведених вище результатів розрахунку ми бачимо, що найгірше, що може трапитися з мережею, це встановити занадто високу поправку ваги, що може спричинити випадкове встановлення середньоквадратичної помилки нижче цього значення, навіть якщо розподіл ваги не є в нейронах Забезпечити коректну роботу мережі. Щоб запобігти цьому, програмне забезпечення було розроблено з використанням правила імпульсу. Він працює шляхом поєднання попередніх і поточних значень коригування ваги. Використання цього імпульсу призводить до того, що корекція ваги (швидкість навчання) починається з деякого високого рівня, а потім зменшується, коли кожна ітерація наближається до значення, указаного вчителем у вихідних даних. Цей підхід масштабує та покращує процес навчання, одночасно запобігаючи неочікуваному завершенню програми через великі корекції ваги. Навчання нейронної мережі вимагає обробки великої кількості вхідних даних.

Нейронна мережа не може розрізнити один і той же об’єкт у різних положеннях відносно площини, оскільки для неї це різні об’єкти, тому під час навчання кожен об’єкт повинен бути представлений мережею під різними кутами та різними налаштуваннями. Ця необхідність випливає з того факту, що мережа спочатку шукає найпростіші ознаки в об'єктах, які вона розпізнає. Якщо ви досліджуєте набір об’єктів, які повернуті праворуч, наприклад стільці, а потім набір подібних об’єктів, але тільки обличчям ліворуч, то під час тестування мережі, коли для класифікації дається стілець обличчям ліворуч, мережа відповість що саме Тумбочки – адже найвідмітнішою рисою тумбочок буде їх повернення. Тому виникає необхідність «чергувати» предмети під час навчального процесу.

Після того, як зображення завантажені та навчені в мережу, він здатний з високою точністю ідентифікувати об’єкти на зображеннях і призначати їм правильні мітки. Мережа порівнює свої результати розпізнавання з реальними зображеннями та відображає на цьому зображенні зелену мітку, якщо розпізнавання правильне, і червону мітку в іншому випадку. Додавання масштабування даних дає змогу використовувати ту саму базу даних нейронної мережі з дещо іншою структурою.

Завдяки первинному тестуванню мережі на зображеннях з низькою роздільною здатністю мережа працює швидше та витрачає менше часу на визначення своєї структури та вибір відповідних факторів навчання.

Таким чином, представлений вище матеріал дозволяє ідентифікувати різні об’єкти на зображеннях, застосовуючи алгоритми машинного навчання для класифікації за допомогою штучних нейронних мереж. Нейронні мережі — чудовий інструмент для розпізнавання об’єктів на зображеннях, але слід пам’ятати про правильний вибір її моделі. Також дуже важливий правильний вибір кількості і типу шарів, числа нейронів, функцій активації і значень факторів навчання. Інтерфейс комп’ютерного додатку заснований на використанні Deep Learning Toolbox.

Це дозволяє легко завантажувати в попередньо розроблені нейронні мережі, бази даних або вибирати відсоток даних для навчання мережі, розпізнавання об’єктів і перевірки. v Представлені експериментальні результати демонструють переваги програмного забезпечення, яке використовується для обробки ШНМ.Знання архітектури нейронної мережі дозволяє скоротити час навчання та розпізнавати об’єкти на зображеннях у кінцевій системі в режимі реального часу. Щоб мати можливість використовувати штучні нейронні мережі для розпізнавання в реальних системах, потрібна велика кількість відомих еталонних об’єктів, які можна використовувати під час навчання зображень. Представлену тут модель розпізнавання можна адаптувати до будь-яких потреб розпізнавання об’єктів.

2.3 - Принцип роботи бібліотеки MediaPipe

Здатність відчувати форму та рухи руки може бути важливою частиною покращення взаємодії з користувачем у різноманітних технологічних областях і платформах.

Наприклад, це може бути основою для розуміння мови жестів і керування жестами, а також для накладання цифрового контенту та інформації на поверхні у фізичному світі в доповненій реальності. Хоча це природно для людини, надійне сприйняття руки в реальному часі є надзвичайно складним завданням для комп'ютерного зору, оскільки руки часто закривають одна одну (наприклад, оклюзія пальців/долоні та тремтіння рук) і не мають висококонтрастної моделі.

MediaPipe Hands — це високоточне рішення для відстеження рук і пальців. Він використовує машинне навчання (ML), щоб визначити 21 3D орієнтир руки лише з одного кадру. У той час як поточні найсучасніші підходи покладаються насамперед на потужні настільні середовища для висновків, наш метод забезпечує продуктивність у режимі реального часу на мобільному телефоні та навіть масштабується для кількох рук. Зроблення цієї сенсорної здатності доступною для широкого дослідницького співтовариства призведе до творчих варіантів використання, стимулюючи нові застосування.

MediaPipe Hands використовує конвеєр ML, що складається з кількох моделей, що працюють разом: модель виявлення долоні, яка працює з повним зображенням і повертає орієнтовану обмежувальну рамку для руки. Модель орієнтира руки, яка обробляє обрізані області зображення, визначені детектором долоні, і повертає високоточні тривимірні ключові точки руки. Ця стратегія подібна до тієї, що використовується в нашому рішенні MediaPipe Face Mesh, яке використовує детектор обличчя та еталонну модель обличчя.

Надання моделі орієнтира руки з точно обрізаними зображеннями руки значно зменшує потребу в доповненні даних, наприклад обертанні, трансляції та масштабуванні, і натомість дозволяє мережі присвятити більшу частину своїх можливостей скоординованій точності передбачення. Крім того, у нашому конвеєрі вирізки також створюються з орієнтирів руки, визначених у попередньому кадрі, а виявлення долоні викликається для зміни положення руки лише тоді, коли модель орієнтира більше не може розпізнати присутність руки.

Конвеєр реалізовано як графік MediaPipe, який використовує підграф відстеження орієнтирів рук із модуля орієнтирів рук і візуалізує за допомогою спеціального підграфа рендерера рук. Субграф відстеження орієнтирів руки внутрішньо використовує субграф орієнтирів руки з того самого модуля та субграф виявлення долоні з модуля виявлення долоні.

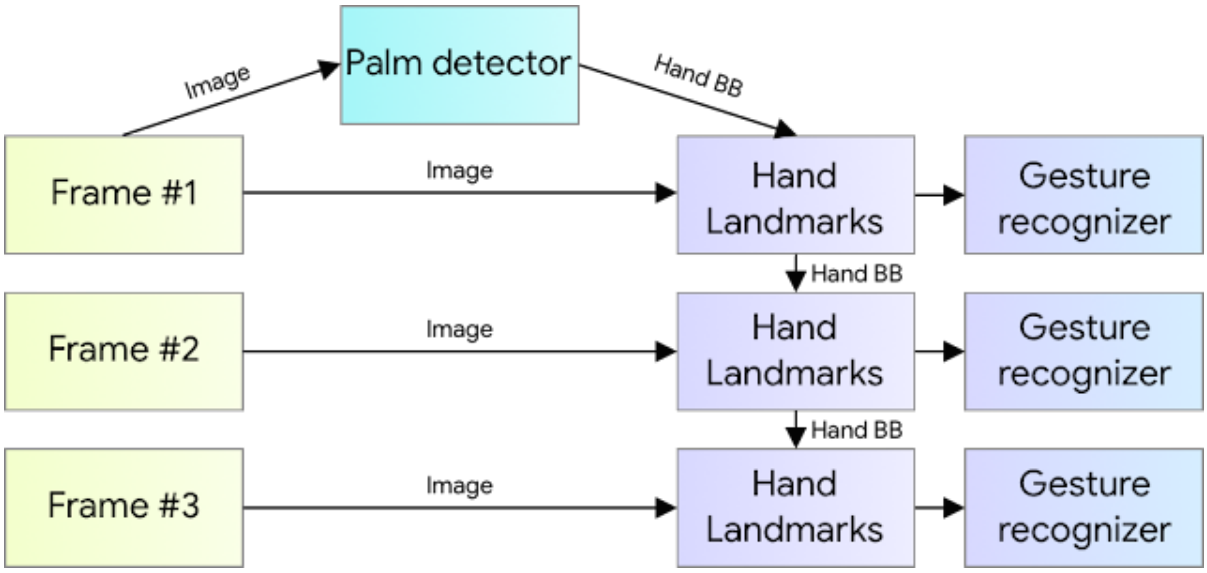

Рисунок 2.3.1 – Огляд принципу сприйняття руки

Щоб визначити початкове положення руки, MediaPipe розробила одноразову модель детектора, оптимізовану для мобільного використання в режимі реального часу, подібну до моделі розпізнавання обличчя в MediaPipe Face Mesh.

Виявлення руки є надзвичайно складним завданням: наші спрощені та повні моделі повинні працювати з руками різного розміру, мати великий діапазон масштабування (~20x) відносно рамки зображення та бути здатними виявляти оклюзію та самооккклюзію руки. Хоча обличчя мають висококонтрастні візерунки, наприклад, в області очей і рота, відсутність таких ознак на руках ускладнює їх надійне виявлення лише на основі візуальних ознак. Натомість надання додаткового контексту, наприклад рук, тіла чи рис обличчя, допомагає точно визначити місцезнаходження рук. Цей метод використовує різні стратегії для вирішення вищезазначених проблем. По-перше, MediaPipe тренує детектори долоні замість детекторів рук, тому що оцінити обмежувальні рамки твердих предметів, таких як долоні та кулаки, набагато легше, ніж виявити руки з шарнірними пальцями.

Крім того, оскільки долоні є невеликими об’єктами, алгоритми немаксимального придушення добре працюють навіть у ситуаціях, коли руки самозакриваються, наприклад, рукостискання. Крім того, долоню можна моделювати за допомогою квадратного обмежувального прямокутника (якір у термінології ML), ігноруючи інші співвідношення сторін.

Таким чином зменшуючи кількість якорів у 3-5 разів. По-друге, засіб виділення функцій кодера-декодера використовується для кращого визначення контексту сцени, навіть для малих об’єктів (подібно до підходу RetinaNet). Нарешті, ми мінімізуємо фокусні втрати під час навчання, щоб підтримувати велику кількість прив’язок, спричинених високою дисперсією масштабу. Використовуючи вищевказаний метод, ми досягли середньої точності 95,7% у виявленні долоні. Використання звичайної крос-ентропійної втрати без декодера дає базову лінію лише 86,22%. Однак чисто синтетичні дані погано узагальнюють дикий домен.

Щоб подолати цю проблему, ми використовуємо гібридну схему навчання. Схема навчання моделі високого рівня показана на малюнку нижче.

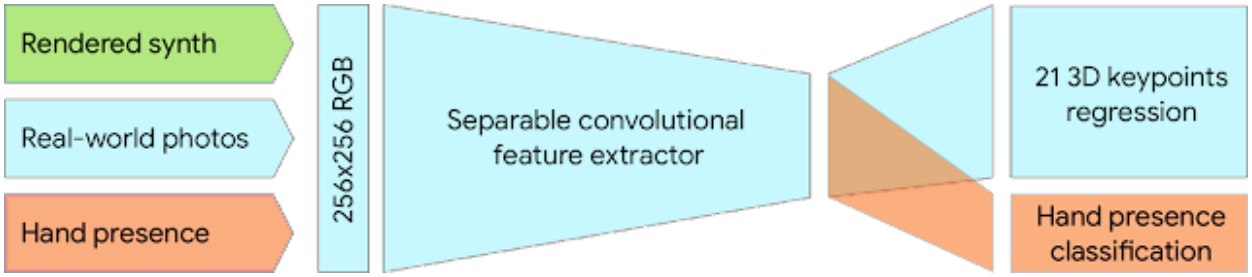

Рисунок 2.3.1 - Змішана схема навчання для мережі відстеження рук

За допомогою MediaPipe цей перцептивний конвеєр можна побудувати як орієнтований граф модульних компонентів, які називаються калькуляторами.

MediaPipe поставляється з розширюваним набором калькуляторів для таких завдань, як визначення моделі, алгоритми обробки медіа та перетворення даних на різних пристроях і платформах. Спеціальні калькулятори, такі як відсікання, візуалізація та обчислення нейронної мережі, можуть працювати виключно на GPU. Наприклад, ви можете використовувати TFLite GPU Inference на більшості сучасних телефонів. Наш графік MediaPipe для відстеження руки показаний нижче. Графік складається з двох підграфів: один для виявлення рук і інший для обчислення ключових точок рук (тобто орієнтирів). Ключовою оптимізацією MediaPipe є те, що детектор долоні запускається лише за потреби, заощаджуючи багато часу на обчислення.

Ми робимо це, визначаючи положення руки в наступних відеокадрах на основі обчислених ключових точок руки в поточному кадрі, усуваючи необхідність покадрово запускати детектор долоні. Для надійності модель відстеження руки виводить додатковий скаляр, який фіксує впевненість у тому, що рука присутня та достатньо вирівняна у вхідному кадрі. Лише коли впевненість падає нижче певного порогу, модель виявлення руки повторно застосовується до всього кадру.

Рисунок 2.3.1 – Узагальнений алгоритм зміни звуку комп’ютера за допомого жестів

2.4 Висновки до розділу

У даному розділі наведено опис сучасних алгоритмів розпізнавання зображень, які дозволяють швидко та якісно обробляти жести користувача на вході і інтерпретувати їх для управління комп’ютерними системами.

Задля кращого розуміння технології управління жестами проведено покроковий аналіз різних методів алгоритмів комп’ютерного зору та комп’ютерного навчання, саме до яких і відноситься бібліотека MediaPipe.

На основі експериментальних досліджень наведено основний алгоритм роботи із жестами в MediaPipe Hands. У вигляді блок-схеми наведено узагальнену структуру опрацювання зображення за допомогою MediaPipe.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ РЕГУЛЮВАННЯ ЗВУКУ КОМП'ЮТЕРА З ДОПОМГОЮ ЖЕСТІВ

3.1 - Архітектура системи.

Для розробки цієї системи, як було зазначено вище, ми будемо використовувати бібліотеку MediaPipe як основу. Першим кроком у створенні архітектури нашої системи є розпізнавання руки користувача та створення її моделі.

Крім того, оскільки долоні є невеликими об'єктами, алгоритми немаксимального придушення (NMS) підходять для самозакривання рук, наприклад рукостискання. Крім того, долоню можна моделювати, використовуючи лише квадратну рамку та ігноруючи інші пропорції, таким чином зменшуючи кількість ригів у 3-5 разів. По-друге, ми також використовуємо екстрактор функцій кодера-декодера, схожий на FPN, щоб покращити розпізнавання контексту сцени навіть для невеликих об'єктів. Нарешті, ми мінімізуємо фокусні втрати під час навчання, щоб підтримувати велику кількість прив'язок (жорстких блоків), спричинених високою дисперсією масштабу.

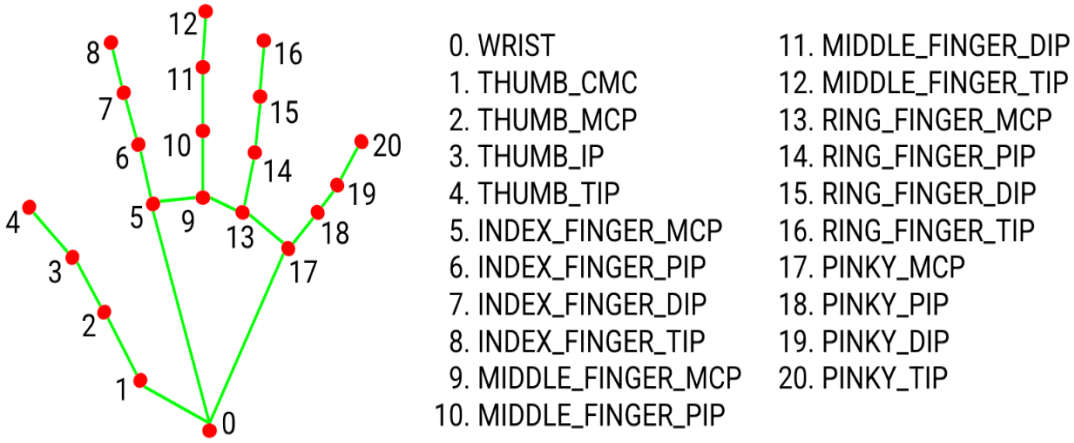

Рисунок 3.1.1 – Основна модель індексування лодоні в Mediapipe.

На першому кроці ми розглянемо лише координати x і y детектора, де кожне зображення в наборі даних фактично перевіряється, щоб зібрати всі точки даних в один файл.

Ми робимо це, визначаючи положення руки в наступних відеокадрах на основі обчислених ключових точок руки в поточному кадрі, усуваючи необхідність покадрово запускати детектор долоні. Для надійності модель відстеження руки виводить додатковий скаляр, який фіксує впевненість у тому, що рука присутня та достатньо вирівняна у вхідному кадрі. Лише коли впевненість падає нижче певного порогу, модель виявлення руки повторно застосовується до всього кадру.

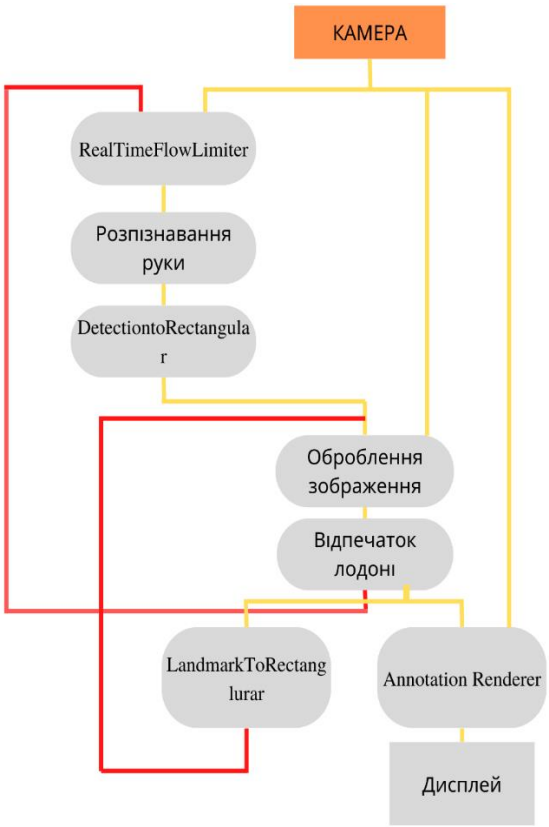

Рисунок 3.1.2 - Вихід моделі ручного орієнтира

Високопродуктивне ML-рішення, яке працює в реальному часі та на кількох платформах і форм-факторах, складніше, ніж те, що зображено у спрощеному описі вище.

Потім файл очищається, щоб перевірити наявність порожніх записів за допомогою функції в бібліотеці pandas. Іноді детектору не вдається виявити руку через розмиті зображення.

Це призводить до нульових записів у наборі даних. Нам потрібно видалити ці записи в наборі даних. Тому ці точки або порожні записи потрібно очистити, інакше це призведе до зміщення прогнозної моделі. Рядки, що містять ці пусті записи, знаходяться за їх індексами. Після видалення непотрібних точок ми нормалізуємо координати x і y відповідно до нашої системи. Потім підготуйтеся до розділення файлу даних на дві частини, навчальний набір і набір перевірки. 80% даних зарезервовано для навчання нашої моделі з використанням різних функцій оптимізації та втрат, а 20% даних зарезервовано для перевірки моделі.

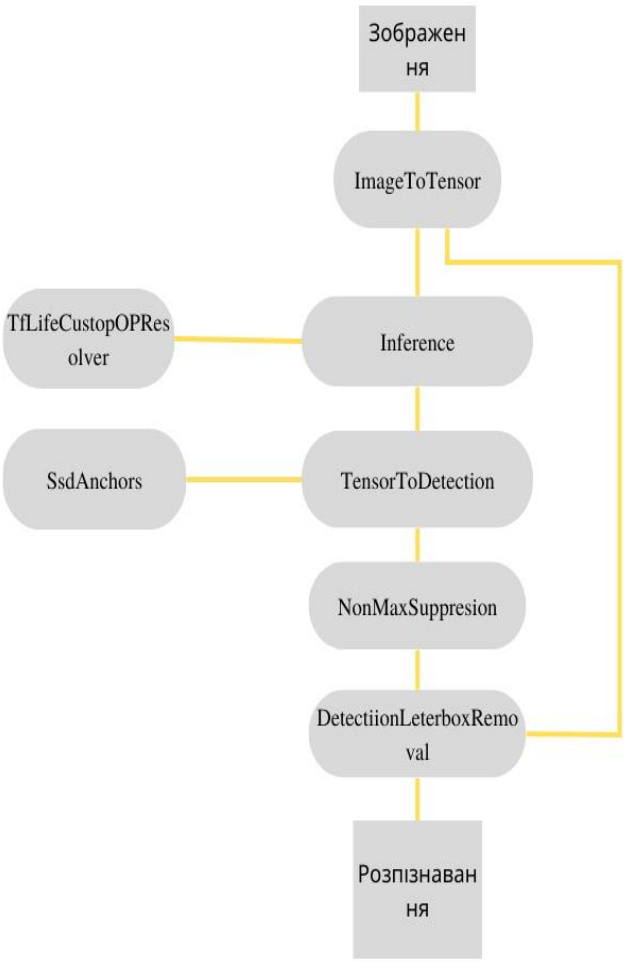

Рисунок 3.1.3 – Процес обробки зображення в MediaPipe

Після виявлення долоні на всьому зображенні модель орієнтира руки використовує регресію (тобто пряме передбачення координат), щоб точно локалізувати 3D-координати ключової точки 21 руки та пальців у виявленій області руки.

Модель вивчає послідовне внутрішнє представлення поз рук, яке є надійним навіть для частково видимих рук і самоокклюзій. Щоб отримати реальні дані, ми створили наступні набори даних, які стосуються різних аспектів проблеми. Набір даних про дику природу: цей набір даних містить 6K різних зображень, таких як географічне різноманіття, різні умови освітлення та зовнішній вигляд рук. Обмеження цього набору даних полягає в тому, що він не містить складних артикуляцій рук. Внутрішній набір даних жестів: цей набір даних містить 10 000 зображень, що охоплюють різні кути всіх фізично можливих жестів. Обмеженням цього набору даних є те, що він був зібраний лише від 30 осіб із обмеженою варіацією фону. Внутрішні та власні набори даних ідеально доповнюють один одного для підвищення надійності. Синтетичний набір даних: щоб додатково зафіксувати можливі пози рук і забезпечити додатковий глибокий контроль, ми візуалізуємо високоякісні синтетичні моделі рук на різних фонах і наносимо їх на відповідні 3D-координати. Ми використали комерційну 3D-модель руки, що складається з 24 кісток, включаючи 36 змішаних форм, які контролюють товщину пальців і долоні. Макет також пропонує 5 текстур з різними відтінками шкіри. Ми створили відеоряди переходів поз рук і вибрали 100 000 зображень із відео. Ми фіксуємо кожну позу, використовуючи випадкове освітлення з високим динамічним діапазоном і три різні камери. Для детектора долонь ми використовуємо лише ті набори даних, які є достатніми для локалізації рук і забезпечують найрізноманітніший вигляд. Однак усі набори даних використовуються для навчання моделей рук орієнтирів. Ми додаємо до зображень реального світу 21 орієнтир і використовуємо спроектовані 3D-з’єднання для синтезу зображень. Для наявності рук ми вибираємо підмножину

зображень реального світу як позитивні приклади та область, яка не включає анотовану область руки як негативні приклади.

Для зручності ми додали підмножину реальних зображень за допомогою пера, щоб надати такі дані

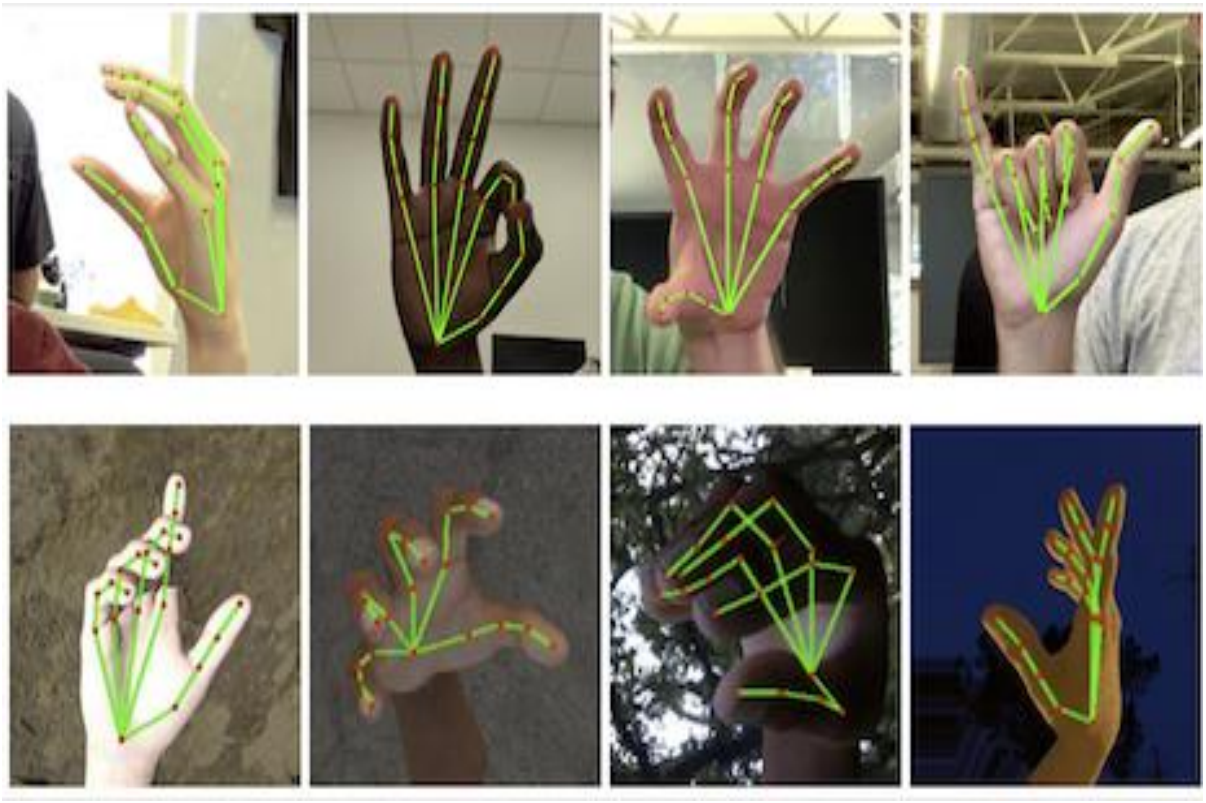

Рисунок 3.1.4 – Приклад бази даних

Проведені експерименти показують, що для еталонних моделей рук поєднання реальних і синтетичних наборів даних забезпечує найкращі результати. Ми оцінюємо лише реальні зображення. Окрім покращення якості, навчання з великими синтетичними наборами даних зменшує візуальне тремтіння між кадрами. Це спостереження змушує нас повірити, що наш реальний набір даних можна розширити для досягнення кращого узагальнення.

Наша мета — забезпечити продуктивність у реальному часі на мобільних пристроях. Ми проекспериментували з різними розмірами моделей і виявили, що повна модель забезпечує гарний компроміс між якістю та швидкістю.

Збільшення потужності моделі приносить лише невеликі поліпшення якості, але істотне зниження швидкості.

Використовуючи MediaPipe, наш трасований конвеєр можна побудувати як орієнтований граф модульних компонентів, які називаються калькуляторами.

Mediapipe постачається з розширюваним набором калькуляторів для таких завдань, як визначення моделі, обробка медіа та перетворення даних на різних пристроях і платформах.

Спеціальні калькулятори, такі як відсікання, візуалізація та обчислення нейронної мережі, були додатково оптимізовані для використання переваг прискорення GPU. Наприклад, ми використовуємо графічні процесори TFLite для більшості сучасних пристроїв.

Графік складається з двох підграфів - одного для виявлення рук і одного для обчислення орієнтирів. Ключова оптимізація, яку забезпечує MediaPipe, полягає в тому, що детектор долоні спрацьовує лише за потреби (рідко), що економить багато обчислень. Ми робимо це шляхом визначення положення руки в поточному кадрі відео на основі орієнтирів рук, обчислених у попередньому кадрі, усуваючи необхідність застосовувати детектор долоні до кожного кадру. Для надійності модель відстеження руки також виводить додатковий скаляр, який фіксує впевненість у тому, що рука присутня та достатньо вирівняна у вхідному кадрі. Лише коли впевненість падає нижче певного порогу, модель виявлення руки повторно застосовується до наступного кадру.

Після розробки моделі захоплення та обробки зображень долоні ми зможемо розробити програмне забезпечення для реалізації алгоритму за допомогою мови програмування Python. Python — це інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Його розширені вбудовані структури даних у поєднанні з динамічним набором тексту та динамічним зв’язуванням роблять його дуже привабливим для швидкої розробки додатків і для використання як мови сценаріїв або як мови

зв’язування для з’єднання існуючих компонентів. Простий у вивченні синтаксис Python підкреслює читабельність.

Таким чином зменшуючи витрати на обслуговування програми. Python підтримує модулі та пакунки, які заохочують модульність програми та повторне використання коду. Інтерпретатор Python і обширна стандартна бібліотека вільно доступні у вихідному або двійковому вигляді для всіх основних платформ і вільно поширюються.

Python зазвичай використовується для розробки веб-сайтів і програмного забезпечення, автоматизації завдань, аналізу даних і візуалізації. Оскільки його відносно легко вивчити, Python використовувався багатьма непрограмістами, такими як бухгалтери та вчені, для різноманітних повсякденних завдань, таких як організація фінансів. Python став основним продуктом науки про дані, що дозволяє аналітикам даних та іншим професіоналам використовувати цю мову для виконання складних статистичних обчислень, створення візуалізацій даних, створення алгоритмів машинного навчання, обробки й аналізу даних, а також виконання інших завдань, пов’язаних із даними.

Python може створювати різні візуалізації даних, такі як лінійні та стовпчасті діаграми, секторні діаграми, гістограми та 3D-графіки. Python також має багато бібліотек, які дозволяють програмістам писати програми інтелектуального аналізу даних і машинного навчання швидше й ефективніше, наприклад TensorFlow і Keras. Щоб жести, які ми пояснюємо, могли контролювати рівень гучності на нашому пристрої, ми будемо використовувати бібліотеку Pycaw.

Pycaw — це досить проста у використанні бібліотека, функції якої дозволяють дуже швидко і без обмежень взаємодіяти з параметрами звуку на різних пристроях. Ця бібліотека дозволяє змінювати рівень гучності параметрів звуку, регулювати фон і висоту параметрів звуку.

Для програмної реалізації цієї роботи нам знадобляться: mediapipe 0.8.1, OpenCV 3.4.2 або вище, Tensorflow 2.3.0 або вище, tf-nightly 2.5.0.dev або вище (тільки якщо створюється TFLite для моделі LSTM) , scikit-learn 0.23.2 або

пізнішої версії (тільки якщо ви хочете відобразити матрицю помилок), matplotlib 3.3.2 або пізнішої версії (лише якщо ви бажаєте відобразити матрицю помилок)

3.2 - Програмна реалізація компонентів і алгоритмів

Перш ніж використовувати бібліотеку MediaPipe у Python, її потрібно встановити:

```
pip install mediapipe
```

Наступним кроком є створення класу обслуговування, щоб зробити наш проект модульним. Імпортуємо необхідні пакети.

```
import mediapipe as mp
import cv2
```

Створимо екземпляр ручного модуля, наданого Mediapipe, а потім екземпляр утиліти малювання Mediapipe. Утиліта для малювання допоможе вам намалювати ці 21 орієнтир і лінії, що їх з'єднують, на вашому зображенні чи рамці. Цей крок є майже постійним і його слід виконувати в кожному проекті, який використовує MediaPipe.

```
mpHands = mp.solutions.hands
mpDraw = mp.solutions.drawing_utils
```

Наступний крок — почати писати наш перший клас.

```
class HandDetector:
def __init__(self, max_num_hands=2, min_detection_confidence=0.5,
min_tracking_confidence=0.5):
self.hands = mpHands.Hands(max_num_hands=max_num_hands,
min_detection_confidence=min_detection_confidence,
min_tracking_confidence=min_tracking_confidence)
```

Max_num_hands: кількість роздач, яку має виявити Mediapipe. Mediapipe поверне масив лотів, кожен елемент масиву (або лотів) у свою чергу має 21 опорну точку.

Min_detection_confidence, min_tracking_confidence: Коли Mediapipe запускається вперше, руки виявляються. Після цього він намагається відстежити руку, оскільки виявлення займає більше часу, ніж відстеження. Він повернеться до виявлення, якщо впевненість відстеження впаде нижче вказаного значення. Клас Hands() потребує всіх цих параметрів, тому ми передаємо їх класу Hands у наступному рядку.

Далі ми визначаємо функцію findHandLandMarks() цього класу, основні речі, які тут відбуваються:

```
def findHandLandMarks(self, image, handNumber=0, draw=False):
                originalImage = image
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # mediapipe needs RGB
            results = self.hands.process(image)
                landMarkList = []

if results.multi_hand_landmarks: # returns None if hand is not found
            hand = results.multi_hand_landmarks[handNumber]
  #results.multi_hand_landmarks returns landMarks for all the hands

              for id, landMark in enumerate(hand.landmark):
            # landMark holds x,y,z ratios of single landmark
imgH, imgW, imgC = originalImage.shape # height, width, channel for image
       xPos, yPos = int(landMark.x * imgW), int(landMark.y * imgH)
                landMarkList.append([id, xPos, yPos])

                            if draw:
mpDraw.draw_landmarks(originalImage, hand, mpHands.HAND_CONNECTIONS)

                            return landMarkList
```

Функціональний параметр: зображення, на якому будуть виявлені орієнтири рук. HandNumber у випадку, якщо на зображенні кілька рук, тому наша функція повертатиме орієнтири лише з указаною кількістю рук. Логічне

значення малювання визначає, чи хочемо ми, щоб MediaPipe намалював ці орієнтири на нашому зображенні.

Наступний рядок виконує все. Цей маленький рядок насправді робить багато речей за лаштунками та дає нам усі вказівки

```
results = self.hands.process(image)
```

Потім ми створюємо порожній landMarkList, який міститиме остаточний результат, повернутий функцією. Results.multi_hand_landmarks повертає None, якщо руки не виявлено, тому ви повинні використовувати це як безвідмовний стан. Results.multi_hand_landmarks повертає орієнтири для всіх виявлених роздач, тому передача йому handNumber дасть вам дані для правильної роздачі, а hand.landmark повертає 21 орієнтир для вибраного гравця. Тому ми повторюємо ці 21 точки, де id містить ідентифікатор для кожного орієнтира. results.multi_hand_landmarks[0].landmark поверне координати всіх 21 орієнтирів на 3 осях, як показано нижче. Індекс 0 означає, що він визначатиме лише орієнтири першої виявленої руки.

```
, x: 0.7789771
y: 0.02958417
z: -0.33826008
, x: 0.86682624
y: 0.5056988
z: -0.14933816
, x: 0.8733816
y: 0.31712145
z: -0.22050717
, x: 0.878462
y: 0.19470873
z: -0.2739978
, x: 0.8840521
y: 0.07972583
z: -0.31275642
, x: 0.94578373
y: 0.5453321
z: -0.1611221
```

Рисунок 3.2.1 – Обраховані Координати

Слід зазначити, що інформація про орієнтир, яку повертає Mediapipe, не є положенням орієнтира в пікселях. Натомість це співвідношення сторін зображення. Отже, щоб отримати точні координати x і y орієнтирних пікселів, ми виконуємо наступне просте обчислення:

```
xPos, yPos = int(landMark.x * imgW), int(landMark.y * imgH)
landMarkList.append([id, xPos, yPos])
```

Потім важливо додати ідентифікатор орієнтира (0…21) і відповідні координати x і y до порожнього списку, який ми створили раніше. Ми повертаємо цей список до функції виклику. Остання частина малює орієнтири на зображенні, якщо це визначено логічною змінною draw.

```
mpDraw.draw_landmarks(originalImage, hand,
mpHands.HAND_CONNECTIONS)
```

Отже, все, що нам потрібно зробити, це передати зображення руки в findHandLandMarks(), і ми отримаємо список з інформацією про всі 21 орієнтир.

На наступному кроці ми множимо ці масштабовані значення на висоту та ширину зображення, щоб отримати координати в піксельних значеннях. Потім ми додаємо їх до списку з індексами орієнтирів і відповідними координатами x,y.

```
for id, lm in enumerate(results.multi_hand_landmarks[0].landmark):
                    h, w, _ = image.shape
            xc, yc = int(lm.x * w), int(lm.y * h)
                lml.append([id, xc, yc])
                    xl.append(xc)
                    yl.append(yc)
```

```
[[0, 191, 392], [1, 180, 342], [2, 176, 312], [3, 178, 294], [4, 179, 282], [5, 164, 332], [6, 163, 320],
```

Рисунок 3.2.2 - Список координат орієнтирів у форматі [індекс, x, y]

Для регулювання гучності будемо використовувати відстань між кінчиком вказівного та великого пальців.

По-перше, ми можемо отримати координати кінчиків пальців великого та вказівного пальців зі списку орієнтирів відповідно до індексу орієнтирів.

У цьому випадку великий палець вказівного пальця дорівнює 4, а кінчик вказівного пальця — 8. Потім використовуйте отримані координати, щоб намалювати коло на кінчику пальця і великому пальці з лінією між ними. Далі обчисліть відстань між двома підказками.

```
x1, y1 = lml[4][1], lml[4][2]
x2, y2 = lml[8][1], lml[8][2]

cv2.circle(image, (x1, y1), 10, (255, 0, 128), cv2.FILLED)
cv2.circle(image, (x2, y2), 10, (255, 0, 128), cv2.FILLED)
cv2.line(image, (x1, y1), (x2, y2), (255, 0, 128), 3)distance =
math.hypot(x2 - x1, y2 - y1)cv2.putText(image, str(int(distance)),
(cx+30, cy), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 128), 3)
```

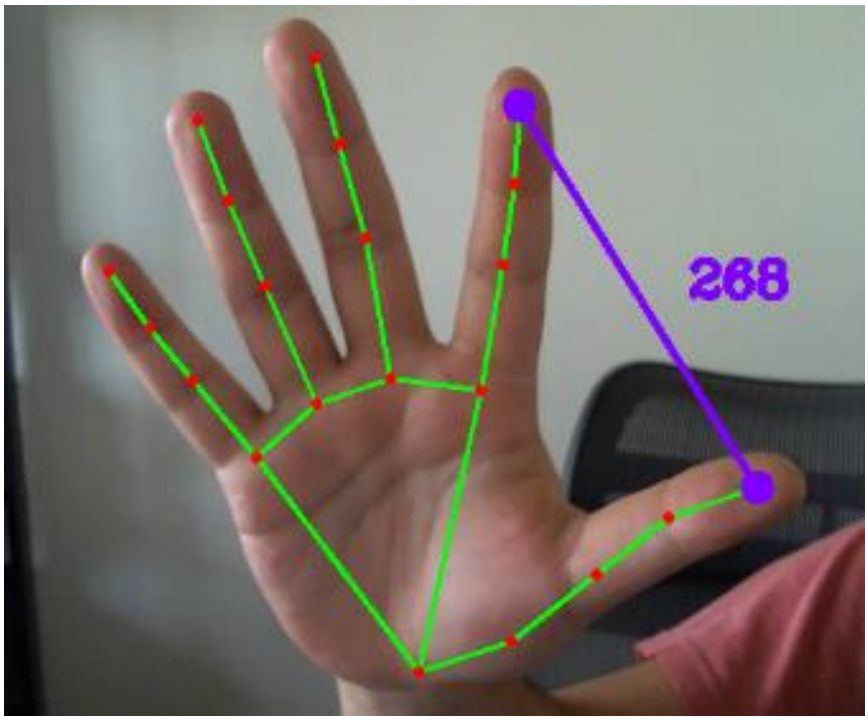

Рисунок 3.2.3 - Відстань (у пікселях) між кінчиками великого та вказівного пальців

Гучність буде регулюватися відповідно до значення відстані, діапазон робочої відстані фіксований.

Оскільки відстань обчислюється на основі пікселів між великим і кінчиком пальців, це значення буде пов’язане з відстанню руки від веб-камери. Тому нам потрібно створити функцію, яка вмикає керування жестами, лише якщо ми перебуваємо на певній відстані від веб-камери, яка повертатиме дійсні значення в межах діапазону робочої відстані.

Наприклад, якщо рука знаходиться надто далеко від веб-камери, ми не зможемо збільшити гучність до 100%, тому що ми збільшуємо відстань між великим і вказівним пальцями до максимуму, який ми можемо отримати, тобто менше ніж фіксоване значення volumeMax. Шляхом експериментів було встановлено, що оптимальне значення площі для людини, яка сидить перед комп’ютером, становить від 300 до 1000.

```
if 300 < area < 1000:
cv2.putText(image, 'GestureControl On', (0, 30), cv2.FONT_HERSHEY_COMPLEX,
                            1, (0, 255, 0), 2)
        cv2.putText(image, str(int(area)), (box[1] + 50, box[1]),
                cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
```

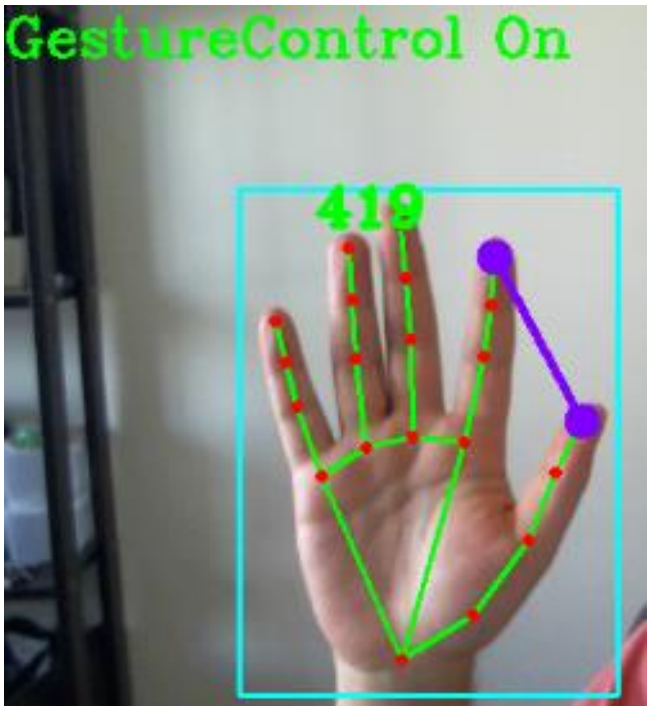

Рисунок 3.2.4 - Управління жестами активується, якщо виявлена область більша 300

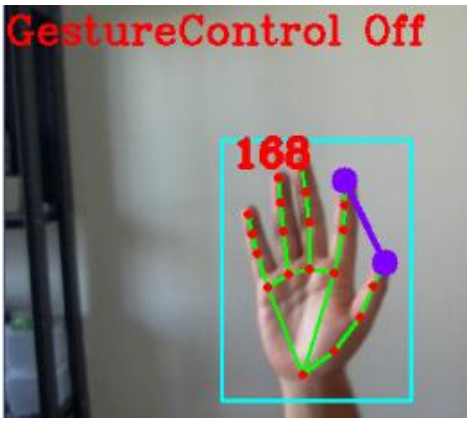

Рисунок 3.2.5 - Управління жестами деактивується, якщо область менша за 300

Наступним кроком буде реалізація можливості управління гучністю медіа на нашому пристрої. Перш ніж ми зможемо написати будь-який власний код, нам потрібно встановити зовнішній пакет Python pycaw. Ця бібліотека керуватиме системним томом. Ми починаємо з імпорту власних класів, які ми створили раніше, та інших необхідних пакетів.

```
from handDetector import HandDetector
import cv2
import math
import numpy as np
```

Також імпортуємо пакети та клас, пов’язані з pycaw:

```
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
```

Потім ми створюємо екземпляр нашого спеціального класу :

```
handDetector = HandDetector(min_detection_confidence=0.7)
```

Тоді стандартний код для ініціалізації нашої веб-камери:

```
webcamFeed = cv2.VideoCapture(0)
```

Потім стандартна ініціалізація для нашого регулятора гучності.

```
#Volume related initializations
devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(
IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
print(volume.GetVolumeRange()) #(-65.25, 0.0)
```

Volume.GetVolumeRange() повертає діапазон гучності, який підтримує система, -65,25 — мінімальне значення, а 0,0 — максимальне значення. Немає пояснення того, як ці значення були отримані або що вони означають, це лише мінімальні та максимальні значення. Нам знадобляться ці значення для останньої частини цієї розробки. Далі ми розглянемо основні правила регулювання гучності вашого пристрою.

```
while True:
status, image = webcamFeed.read()
handLandmarks = handDetector.findHandLandMarks(image=image, draw=True)

if(len(handLandmarks) != 0):
#for volume control we need 4th and 8th landmark
#details: https://google.github.io/mediapipe/solutions/hands
x1, y1 = handLandmarks[4][1], handLandmarks[4][2]
x2, y2 = handLandmarks[8][1], handLandmarks[8][2]
length = math.hypot(x2-x1, y2-y1)
print(length)

#Hand range(length): 50-250
#Volume Range: (-65.25, 0.0)

volumeValue = np.interp(length, [50, 250], [-65.25, 0.0]) #coverting
length to proportionate to volume range
volume.SetMasterVolumeLevel(volumeValue, None)


cv2.circle(image, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
cv2.circle(image, (x2, y2), 15, (255, 0, 255), cv2.FILLED)
cv2.line(image, (x1, y1), (x2, y2), (255, 0, 255), 3)
```

```
cv2.imshow("Volume", image)
cv2.waitKey(1)
```

Спочатку ми зчитуємо кадр із веб-камери, по кадру за раз, і надсилаємо зображення кадру до нашого findHandLandMarks().

```
handLandmarks = handDetector.findHandLandMarks(image=image, draw=True)
```

Потім ми витягуємо координати x і y кінчика великого та вказівного пальців, тобто ми посилаємося на 4-е посилання, яке є координатою x у першому індексі кінчика нашого великого пальця, тобто ми хочемо отримати відповідь, надану findHandLandMarks() Решту можете розібратися самі. Потім обчислюємо довжину лінії, що з'єднує орієнтир 4 і орієнтир 8.

```
x1, y1 = handLandmarks[4][1], handLandmarks[4][2]
length = math.hypot(x2-x1, y2-y1)
```

Коли кінчики вказівного та великого пальців торкаються один одного, буде значення довжини (наприклад, L1, що означає, що ми хочемо встановити гучність системи на 0%). Коли ми розведемо вказівний і великий палець, буде інше значення довжини (наприклад, L2, що означає, що ми хочемо встановити гучність системи на 100%). Давайте спробуємо запустити наш код на цьому етапі та отримати довжину. Давайте запишемо значення довжини L1 і L2, щоб нашим пальцям було зручно. Для мене L1 — 50, а L2 — 250. Раніше ми помітили мінімальне і максимальне значення гучності, назвемо їх V1 і V2. Тепер ми маємо L1=50, L2=250, V1=-65,25, V2=0, тому наступний рядок коду перетворює нашу довжину (L1, L2) на пропорційну рівень гучності (V1, V2)

```
volumeValue = np.interp(length, [50, 250], [-65.25, 0.0])
```

Отримавши правильне значення гучності, ми просто регулюємо гучність системи за допомогою

```
volume.SetMasterVolumeLevel(volumeValue, None)
```

Ми обчислимо volumeBar і volumePercent, щоб намалювати рухому смугу гучності та налаштувати гучність шляхом лінійної інтерполяції значення

відстані. Праворуч від зображення буде намальована панель гучності, яка показуватиме виявлений обсяг у відсотках, збережений у volumePercent. Для кращої взаємодії з користувачем ми включили список правил для відображення смуг гучності різних кольорів для позначення рівнів гучності.

```
volumeBar = int(np.interp(distance, [50, 200], [400, 150]))
volumePercent = int(np.interp(distance, [50, 200], [0, 100]))

cv2.rectangle(image, (w - 50, 150), (w - 80, 400), (255, 255, 255), 2)
if 21 < volumePercent < 50:
cv2.rectangle(image, (w - 50, int(volumeBar)), (w - 80, 400), (0, 255, 0),
cv2.FILLED)
cv2.putText(image, f'{int(volumePercent)} %', (w - 100, 450),
cv2.FONT_HERSHEY_COMPLEX,
1, (0, 255, 0), 2)
elif 51 < volumePercent < 80:
cv2.rectangle(image, (w - 50, int(volumeBar)), (w - 80, 400), (0, 255,
255), cv2.FILLED)
cv2.putText(image, f'{int(volumePercent)} %', (w - 100, 450),
cv2.FONT_HERSHEY_COMPLEX,
1, (0, 255, 255), 2)
elif volumePercent > 81:
cv2.rectangle(image, (w - 50, int(volumeBar)), (w - 80, 400), (0, 0, 255),
cv2.FILLED)
cv2.putText(image, f'{int(volumePercent)} %', (w - 100, 450),
cv2.FONT_HERSHEY_COMPLEX,
1, (0, 0, 255), 2)
elif volumePercent < 20:
cv2.rectangle(image, (w - 50, int(volumeBar)), (w - 80, 400), (255, 255,
0), cv2.FILLED)
cv2.putText(image, f'{int(volumePercent)} %', (w - 100, 450),
cv2.FONT_HERSHEY_COMPLEX,
1, (255, 255, 0), 2)cVol = int(volume.GetMasterVolumeLevelScalar() * 100)
cv2.putText(image, f'Current Volume: {int(cVol)}', (0, 60),
cv2.FONT_HERSHEY_COMPLEX,
1, (255, 255, 255), 2)
```

Отже, об’єднавши ці дві бібліотеки, ми отримаємо шаблонний код, наданий MediaPipe у поєднанні з pycaw. Цей код захопить відеопотік з вашої веб-камери кадр за кадром. Він перетворює зображення на RGB, а потім запускає на ньому модель MediaPipe Hands. Він визначає вашу руку (якщо є), потім оцінює 21 орієнтир на вашій руці та малює орієнтири (використовуючи метод mp_drawing) у вигляді червоних кіл і зелених ліній між орієнтирами.

```
import                                                    cv2
import                mediapipe                as              mp
from          ctypes          import          cast,          POINTER
from             comtypes             import             CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolumedevices
=                                      AudioUtilities.GetSpeakers()
interface                        =                  devices.Activate(
    IAudioEndpointVolume._iid_,          CLSCTX_ALL,          None)
volume     =     cast(interface,     POINTER(IAudioEndpointVolume))
volume.GetMute()
volume.GetMasterVolumeLevel()
volumeRange       =       volume.GetVolumeRange()mp_drawing       =
mp.solutions.drawing_utils
mp_hands    =    mp.solutions.handscap    =    cv2.VideoCapture(0)
with                                              mp_hands.Hands(
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5)             as             hands:
  while                                             cap.isOpened():
    success,             image             =             cap.read()
    if                         not                         success:
      print("Ignoring         empty         camera         frame.")
      continue          image   =   cv2.cvtColor(cv2.flip(image,   1),
cv2.COLOR_BGR2RGB)
    image.flags.writeable                  =                  False
    results                 =                 hands.process(image)

    #    Draw    the    hand    annotations    on    the    image.
    image.flags.writeable                  =                   True
    image       =       cv2.cvtColor(image,       cv2.COLOR_RGB2BGR)
```

Ми можемо запустити код, щоб перевірити, чи орієнтири та анотації рук правильно намальовані на ваших руках. Ми повинні побачити результат, як показано на зображенні нижче.

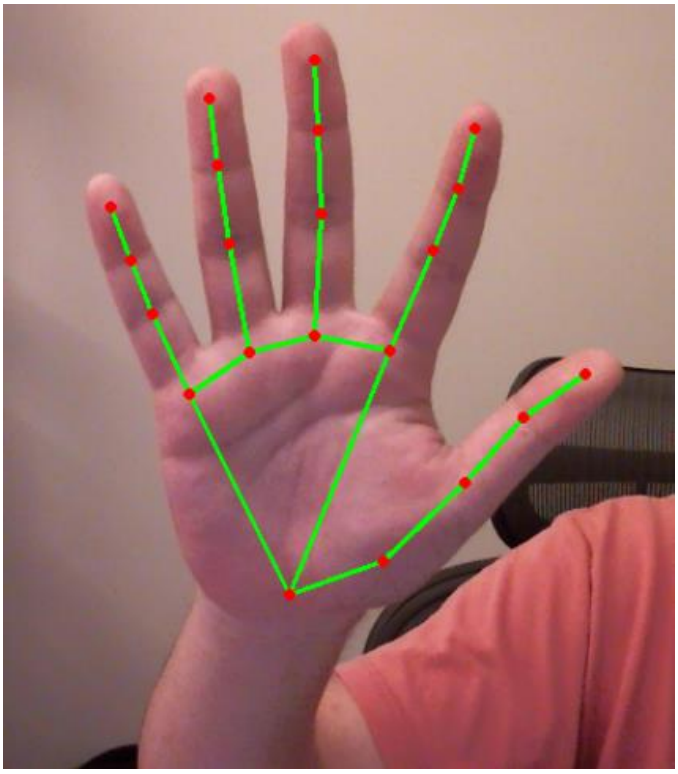

Рисунок 3.2.6 – Приклад орієнтирів на виході

Далі ми створимо функцію, яка перевіряє, чи пальці відкриті чи зімкнуті відносно долоні. Ця інформація дозволить нам налаштувати деякі додаткові функції, такі як регулювання гучності, вимкнення та ввімкнення звуку відповідно до положення пальця.

```
fCount = []
for fid in range(8, 21, 4):
if lml[fid][2] < lml[fid- 2][2]:
fCount.append(1)
else:
fCount.append(0)
```

На цьому останньому кроці ми створимо остаточну функціональність, необхідну для цієї програми. Ця функція виконуватиме дії на основі жестів. Він використовує наведену вище інформацію для виявлення та розпізнавання жестів на основі положення пальців. Статус вимкнення звуку та активація функції буде намальовано у верхньому лівому куті зображення.

```
if fCount[3] == 0 and fCount[2] == 1 and fCount[1] == 1 and fCount[0] == 1:
volume.SetMasterVolumeLevelScalar(volumePercent / 100, None)
cv2.putText(image, 'Volume Set', (0, 90), cv2.FONT_HERSHEY_COMPLEX, 1, (0,
0, 255), 2)
colorVol = (0, 255, 0)
elif fCount[3] == 1 and fCount[2] == 0 and fCount[1] == 0 and muteStatus ==
False:
volume.SetMute(1, None)
cv2.putText(image, 'Muted', (0, 90), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0,
255), 2)
muteStatus = True
```

Ми також можемо використовувати наступний код для відображення лічильника FPS, щоб зрозуміти продуктивність цієї програми.

```
currentTime = time.time()
fps = 1 / (currentTime - previousTime)
previousTime = currentTime
cv2.putText(image, f'FPS: {int(fps)}', (w-150, 50),
cv2.FONT_HERSHEY_COMPLEX,
1, (255, 255, 255), 2)
```

3.4 Висновки до розділу

У даному розділі створено та проаналізовану основну архітектуру проекту а також, визначені основні компоненти та етапи розробки програмного застосунку. Наведено основні алгоритми та схеми, які використовувались при проектуванні проекту.

Створено та проаналізовано кожен крок програмного застосунок для управління звуком комп’ютера на основі бібліотеки MediaPipe.

ВИСНОВКИ

На основі аналітичного підходу проведено аналіз технології управління комп’ютерними системами за допомогою жестів, що дало змогу виділити їх переваги та недоліки на етапі опрацювання на низькому рівні комп’ютерного зору.

Проведено аналіз розвитку технології управління жестами, що дозволило виділити переваги й недоліки існуючих систем для подальшої розробки.

З допомогою порівняльного аналізу виділено сучасні програмні засоби опрацювання зображень, а також детальний аналіз бібліотеки MediaPipe

Наведено опис сучасних алгоритмів машинного зору та машинного навчання для опрацювання зображень, які активно використовуються в сучасних системах.

На основі експериментальних досліджень наведено алгоритм опрацювання зображення на основі бібліотеки MediaPipe.

Було модифіковано основний алгоритм розпізнавання жестів в MediaPipe Hands, а також реалізовано програмний алгоритм контролю медіа засобів пристрою за допомогою жестів.

Створено та проаналізовано кожен крок програмного застосунок для управління звуком комп’ютера на основі бібліотеки MediaPipe

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bolt RA. Put that there: voice and gesture. ACM SIGGRAPH Computer Graphics. 1980(ISSN:0097-8930,1980, ACM, USA):P. 262 - 70.

2. V.L.Hanson JpB, S.Crayne et.al. Improving Web accessibility through an enhanced open-source browse. IBM System Journal,. 2005;44(3).

3. Maribeth Gandy TS, Jake Auxier, Daniel Ashbrook. The Gesture Pendant: A Self-illuminating, Wearable, Infrared Computer Vision System for Home Automation Control and Medical Monitoring. Proceeding ISWC '00 Proceedings of the 4th IEEE International Symposium on Wearable Computer. 2000:87.

4. Shahzad Malik JL. Visual touchpad: a two-handed gestural input device. Proceeding ICMI '04 Proceedings of the 6th international conference on Multimodal interfaces. 2004:Pages 289-96

5. Sanna K. JK, Jani M. and Johan M. Visualization of Hand Gestures for Pervasive Computing Environments. Proceedings of the working conference on Advanced visual interfaces, ACM, Italy. 2006:p480-3.

6. Jia PaHHH. Head gesture recognition for hands-free control of an intelligent wheelchair. Industrial Robot: An International Journal,Emerald. 2007:p60-8.

7. Jacob O. Wobbrock MRM, Andrew D. Wilson. User-defined gestures for surface computing. CHI '09 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2009:Pages 1083-92

8. Andrew Bragdon EN, Yang Li and Ken Hinckley. Experimental Analysis of Touch-Screen Gesture Designs in Mobile Environments. CHI 2011 Session: Gestures, Vancouver, BC, Canada. 2011.

9. Yao-Jen Chang S-FC, An-Fu Chuang A gesture recognition system to transition autonomously through vocational tasks for individuals with cognitive impairments. Research in Developmental Disabilities 2011;32:2064–8

10. Julien Pauchot LDT, Ahmed Lounis, Mourad Benassarou, Pierre Mathieu, Dominique Bernot, Sébastien Aubry. Leap Motion Gesture Control With Carestrea Software in the Operating Room to Control Imaging: Installation Guide and Discussion. Surgical Innovation 2015. 2015;Vol. 22(6) 615–20.

11. Weichert F BD, Rudak B, Fisseler D. Analysis of the accuracy and robustness of the Leap Motion controller. Sensors (Basel). 2013;13:6380-93.

12. Guna J JG, Pogačnik M, Tomažič S, Sodnik J. An analysis of the precision and reliability of the Leap Motion sensor and its suitability for static and dynamic tracking. Sensors (Basel). 2014;14:3702-20.

13. MyoCraft: Logging IMU and Raw EMG Data. 14th March, 2015.

14. J. Han LS, D. Xu, and J. Shotton. Enhanced Computer Vision with Microsoft Kinect Sensor: A Review. IEEE Trans Cybern. Oct. 2013;vol. 43(no. 5):pp. 1318-34.

15. Hsu H-MJ. The Potential of Kinect in Education. International Journal of Information and Education Technology. 2011;Vol.1(No.5).

16. Prindle D. Myo Gesture Control Armband Review. 2015.

17. Yao-Jen Chang S-FC, Jun-Da Huang A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. Research in Developmental Disabilities 2011;32:2566–70.

18. Ling Shao CS, Jiebo Luo, Minoru Etoh. Multimedia Interaction and Intelligent User Interfaces: Principles, Methods and Applications 2010.

19. Zimmerman eA. A hand gesture interface device. Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface. 1987(ISBN:0-89791-213-6,ACM, Canada):P.189 - 92.

20. KinectEDucation (2011) 5 benefits of using Kinect in education. Available at:

http://www.kinecteducation.com/blog/2011/07/02/5-benefits-of-using-kinect-in-education/ (Accessed: 2016).