АНОТАЦІЯ

Олексійко Ю. Р. Розробка програмної системи керування гучністю звуку на основі відео потоку жестів. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня магістр за спеціальністю 121 — Інженерія програмного забезпечення. – Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПм-62 // м.Тернопіль, 2023 // С.62 , рис. – 13, додат. – 6, бібліогр. – 10

Мета кваліфікаційної роботи полягає в використанні сучасних інформаційних технологій для розробки системи зчитування потоку жестів руки за допомогою пристроїв ведення інформацій і контролю функції машини таких як гучність звуку з використанням сучасних методів.

Практичне застосування – розроблено надійна програмна система, що дозволить підвищити ефективність та інноваційний підхід для керуванням систем машини який є сучасним та відповідає поставлені задачі.

Технічні вимоги – методи розробки базуються на технології та високорівневій мові програмування Python, а також принципі відстеження відео потоку жестів.

Ключові слова: ПОТІК ЖЕСТІВ, ОБРОБКА ВІДЕО ПОТОКУ, АНАЛІЗ ЖЕСТІВ, ЧАСТОТНИЙ АНАЛІЗ, НЕЙРОННА МЕРЕЖА, КОМП’ЮТЕРНИЙ ЗІР

ABSTRACTS.

Oleksiiko Y. R. Development of a software system for controlling the volume of sound based on a video stream of gestures.

Qualification work for the degree of Master in speciality 121 - Software Engineering - Ternopil National Technical University named after Ivan Puluj, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, group SPm-62 // Ternopil, 2023 // P.62 , Fig. - 13, Appendix - 6, Bibliography - 10.

The purpose of the qualification work is to use modern information technologies to develop a system for reading the flow of hand gestures using information storage devices and controlling machine functions such as sound volume using modern methods.

Practical application - a reliable software system has been developed that will increase efficiency and provide an innovative approach to controlling machine systems that is modern and appropriate to the task.

Technical requirements - the development methods are based on technology and the high-level programming language Python, as well as the principle of tracking the video stream of gestures.

Keywords: GESTURE STREAM, VIDEO STREAM PROCESSING, GESTURE ANALYSIS, FREQUENCY ANALYSIS, NEURAL NETWORK, COMPUTER VISION

ЗМІСТ

ВСТУП..... 7
1 АНАЛІТИЧНИЙ ОГЛЯД В ОБЛАСТІ ДОСЛІДЖЕНЬ..... 9
1.1Аналіз та огляд предметної області 11
1.2 Основні алгоритми та принципи аналізу відео потоку жестів..... 16
2 ОБҐРУНТУВАННЯ ВИБОРУ НАПРЯМКУ ДОСЛІДЖЕНЬ..... 26
2.1 Вибір технологій розробки 26
2.2 Обґрунтування вибору середовища розробки програмної системи 33
3 РОЗРОБКА СКЛАДОВИХ ПРОГРАМНОГО КОМПЛЕКСУ 36
3.1 Базові операції використовуючи OpenCV і MediaPipe 36
3.2 Клас HandDetector 39
3.3 Функція main 44
3.4 Файл VolumeControl..... 49
4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ 59
4.1 Охорона праці..... 59
4.2 Фактори, що впливають на функціональний стан користувачів комп’ютерів. 63
ВИСНОВОК..... 69

ВСТУП

Нові і сучасні технології дають багато можливостей для вдосконалення і покращення життя, люди шукають нові можливості використати їх для свого блага. Різкий технологічний прогрес збільшив обчислювальні швидкості та можливості для появи нових високоінтелектуальних програмних систем, деякі з них можуть змінити або розширити людські послуги. Розвиток комп'ютерного зору є одним із найкращих прикладів, коли розумні інформаційні системи готові змінити світ робототехніки. Використовуючи згорткові нейронні мережі та OpenCV, комп'ютерний зір дозволяє роботам розпізнавати та інтерпретувати візуальну інформацію, що відкриває нові можливості для взаємодії роботів з людьми та їхнім оточенням.

OpenCV є однією з найцікавіших галузей комп'ютерного зору, яка вже породила такі технології, як розпізнавання обличчя, слідкування за об'єктами, виявлення руху та багато інших інструментів. Завдяки OpenCV, системи можуть "бачити" та інтерпретувати візуальний світ навколо нас.

У сучасному світі, де технології розвиваються зі швидкістю світла, компанії та розробники створюють безліч програм та систем для керування систем машин . Проте виклик полягає в тому, що людям складно обробити великі обсяги відеоданих вручну без помилок або упередженості. Саме тут на допомогу приходить аналіз відео потоку жестів, який дає відповіді на найважливіші проблеми. Оскільки аналіз жестів можна автоматизувати, рішення можна приймати на основі значної кількості даних, а не простої інтуїції, яка не завжди вірна. Таким чином, системи керування звуком на основі відео потоку жестів можуть надати більш точне та ефективне керування звуком.

Для реалізації цього проекту та врахування всіх особливостей предметної області та принципів обробки відео, було використано мову

програмування Python. Ця мова програмування підтримує велику кількість доступних бібліотек для розв’язання завдань комп’ютерного зору та аналізу відео потоку, що полегшує роботу для розробників і дає змогу зосередитися на реалізації самої системи. Завдяки цьому, розробники можуть створювати ефективні системи для керування звуком на основі відео потоку жестів.

Враховуючи те, що обсяг відеоданих зростає експоненціальна, ця тема є надзвичайно актуальною та має великий потенціал. Це, в свою чергу, привертає увагу багатьох компаній, які зацікавлені в подальших дослідженнях і розробках в цій області.

Наукову новизну можна розглядати через використання сучасних технологій машинного навчання для створення унікального нового програмного забезпечення для аналізу відео потоку жестів. Враховуючи вимоги та специфіку конкретної галузі, це може відкрити нові можливості для керування системами машини на основі відео потоку жестів такими як звук.

Практичне застосування результатів наукових досліджень може бути спрямоване на конкретну сферу діяльності, і продукт має постійно вдосконалюватися та пристосовуватися до сучасних вимог.

Виконання та оформлення роботи будуть проводитися відповідно до всіх стандартів та вимог, встановлених методичними рекомендаціями та Державним стандартом України щодо термінології, оформлення звітів та специфічної документації.

1 АНАЛІТИЧНИЙ ОГЛЯД В ОБЛАСТІ ДОСЛІДЖЕНЬ

Розробка програмної системи керування гучністю звуку на основі відео потоку жестів використовує різні технології та методи. В основі цього процесу лежить використання комп’ютерного зору для ідентифікації, розпізнавання та класифікації жестів користувача виявлених з відео потоку.

Ці системи надають зручний та інтуїтивно зрозумілий спосіб взаємодії з комп’ютерними системами. Вони відкривають нові можливості для користувачів, оскільки вони не вимагають використання традиційних пристроїв введення, таких як клавіатура або миша. Замість цього, користувачі можуть взаємодіяти з системами за допомогою своїх рук, використовуючи жести для керування різними функціями, такими як гучність звуку. Це не тільки збільшує зручність використання, але й робить взаємодію з комп’ютерними системами більш природньою та інтуїтивно зрозумілою.

Комп’ютерний зір - це галузь штучного інтелекту, яка надає комп’ютерам здатність “бачити” та інтерпретувати візуальну інформацію в реальному часі. Він використовує алгоритми та моделі машинного навчання для аналізу відео та зображень, щоб визначити об’єкти, особи, текст, дії, а також контекст візуальних даних.

Використання комп’ютерного зору в системах керування гучністю звуку на основі відео потоку жестів включає в себе кілька етапів. По-перше, система повинна визначити присутність руки на відео. Це може бути досягнуто за допомогою алгоритмів виявлення об’єктів, які можуть ідентифікувати форму руки та відокремлювати її від фону.

Ці жести потім інтерпретуються як команди для зміни рівня гучності звуку. Різноманітні технології можуть бути застосовані для цієї задачі, включаючи OpenCV для обробки відео, MediaPipe для детекції та оцінювання

орієнтації рук, PyCaw для регулювання системної гучності, а також NumPy для виконання обчислень та візуалізації.

Ці системи надають зручний та інтуїтивно зрозумілий спосіб взаємодії з комп'ютерними системами. Вони відкривають нові можливості для користувачів, оскільки вони не вимагають використання традиційних пристроїв введення, таких як клавіатура або миша. Замість цього, користувачі можуть взаємодіяти з системами за допомогою своїх рук, використовуючи жести для керування різними функціями, такими як гучність звуку. Це не тільки збільшує зручність використання, але й робить взаємодію з комп'ютерними системами більш природньою та інтуїтивно зрозумілою.

Використання жестів для керування гучністю звуку - це лише один з багатьох можливих способів використання цих технологій. Вони можуть бути використані для керування різними аспектами комп'ютерної системи, включаючи навігацію, вибір опцій, введення тексту та багато іншого. Це відкриває нові можливості для взаємодії з комп'ютерами та іншими цифровими пристроями.

Наприклад, в медичних системах для людей з обмеженими можливостями, в автомобільних системах для безпечного керування автомобілем, в системах доповненої реальності для більш природного способу взаємодії з віртуальним середовищем, а також в системах розваг для створення більш імерсійного досвіду.

Ці системи також можуть бути використані для підвищення доступності комп'ютерних систем для людей з обмеженими можливостями. Наприклад, люди з обмеженою рухливістю або ті, хто не може використовувати традиційні пристрої введення, можуть використовувати жести для керування комп'ютером. Це може значно підвищити їхню самостійність та якість життя.

Використання жестів для керування гучністю звуку - це лише початок. Можливості для використання жестів для взаємодії з комп'ютерами та іншими

цифровими пристроями майже необмежені. В майбутньому ми можемо очікувати бачити все більше і більше застосувань цих технологій в різних областях життя.

1.1Аналіз та огляд предметної області

Темою магістерської роботи є обрано «Розробка програмної системи керування гучністю звуку на основі відео потоку жестів». Обрана тема є на даний момент актуально. Через те, що в наш час у світі виникай потреба у розвитку інноваційних систем керування системами машини такими як звук і інші. Через складність і важкість обробки даних відео потоку в ручну тому і правильна класифікація та розподілення є важливими, як ніколи раніше.

Розвиток системи керування гучністю звуку на основі відео потоку жестів можна описати в термінах трьох основних хвиль: раціоналізм, емпіризм і глибоке навчання. У першій хвилі раціоналістичні підходи відстоювали розробку правил, створених вручну, щоб включати знання в системи керування гучністю звуку, припускаючи, що знання мови жестів в людському розумі заздалегідь фіксується з допомогою вже наявних знань. У другій хвилі емпіричні підходи припускають, що багатий сенсорний вхід і спостережувані дані жестів у поверхневій формі необхідні і достатні для того, щоб розум міг вивчити детальну структуру жестів. У результаті були розроблені ймовірнісні моделі для виявлення закономірностей жестів із великих корпусів. У третій хвилі глибоке навчання використовує ієрархічні моделі нелінійної обробки, натхненні біологічними нейронними системами для вивчення внутрішніх репрезентацій з даних жестів, таким чином, щоб імітувати когнітивні здібності людини.

Глибоке навчання та обробка відео потоку дозволили створити ефективні рішення та зручні для розпізнавання жестів користувача. Розпізнавання жестів - це новий спосіб взаємодії з комп'ютерами, який використовує відео потік з камери, щоб аналізувати рухи рук користувача. За допомогою глибоких нейронних мереж можна навчити систему розпізнавати різні жести і рухи, які відповідають певним командам, наприклад, збільшити або зменшити гучність звуку. Така система може бути корисною для людей з обмеженими можливостями, які не можуть використовувати клавіатуру або мишу, або для забезпечення більш природного та зручного способу керування машиною. Багато інших застосувань обробки відео потоку жестів, таких як взаємодія між машиною і людиною, полегшення користування компютером, також використовують глибоке навчання для покращення своєї продуктивності та якості. Сьогодні глибоке навчання є основним методом, який застосовується майже до всіх завдань обробки.

Нейронні мережі - це тип обчислювальних моделей, які намагаються наслідувати роботу нервової системи людини. Нервова система людини складається з клітин, які називаються нейронами. Біологічні нейрони мають зв'язки один з одним у точках контакту, які називаються синапсами. Навчання в живих організмах відбувається шляхом адаптації сили синапсичних зв'язків між нейронами. Зазвичай, сила цих зв'язків залежить від зовнішніх стимулів. Нейронні мережі можна вважати симуляцією цього біологічного процесу.

Так само, як і в біологічних мережах, окремі елементи в штучних нейронних мережах називаються нейронами. Ці нейрони є обчислювальними вузлами, які приймають вхідні дані від інших нейронів, застосовують обчислювальну функцію до цих входів і передають їх іншим нейронам.

На обчислення в нейроні впливають вагові коефіцієнти вхідних зв'язків з цим нейроном, оскільки вхідні дані нейрона множаться на вагу. Цю вагу можна розглядати як аналогію сили синаптичного зв'язку. Змінюючи ці ваги відповідним чином, можна навчити штучну нейронну мережу виконувати

загальну обчислювальну функцію, подібно до навчання синаптичної сили в біологічних нейронних мережах. “Зовнішній стимул” у штучних нейронних мережах для навчання цих ваг надають навчальні дані. Ідея полягає в тому, щоб поступово коригувати вагові коефіцієнти, коли поточний набір вагових коефіцієнтів дає невірні прогнози. Існує багато різних архітектур нейронних мереж, які варіюються від простого персептрона до складних багатошарових мереж. Використання багатьох шарів називається глибоким навчанням.

Модель персептрона є найпростішою формою нейронної мережі, що містить лише один вхідний і вихідний шари. Оскільки вхідні шари лише передають значення атрибутів, не застосовуючи жодної математичної функції до вхідних даних, функція, яку навчається модель персептрона, є лише простою лінійною моделлю, заснованою на одному вихідному вузлі. На практиці може знадобитися навчати більш складні моделі за допомогою багатошарових нейронних мереж.

Багатошарові нейронні мережі мають прихований шар, крім вхідного та вихідного шарів. Вузли в прихованому шарі можуть бути пов’язані різними способами. Наприклад, сам прихований шар може складатися з кількох шарів, і вузли в одному шарі можуть подаватися на вузли наступного шару. Це називається багатошаровою мережею прямого зв’язку. Також припускається, що вузли в одному шарі повністю з’єднані з вузлами наступного шару. Таким чином, топологія багаторівневої мережі з прямим зв’язком визначається автоматично після того, як кількість шарів і кількість/тип вузлів у кожному шарі були визначені розробником, хоча вибір функції втрат також є важливим. Базовий персептрон можна розглядати як одношарову мережу прямого зв’язку. Популярною моделлю є модель, в якій багатошарова мережа прямого зв’язку містить лише один прихований шар. Таку мережу можна вважати двошаровою мережею прямого зв’язку. Приклад багатошарової мережі прямого зв’язку проілюстрований на Рисунок 1.1.1. Варто зауважити, що

кількість шарів належить до кількості обчислювальних шарів і не включає вхідний рівень (який лише передає дані наступному шару).

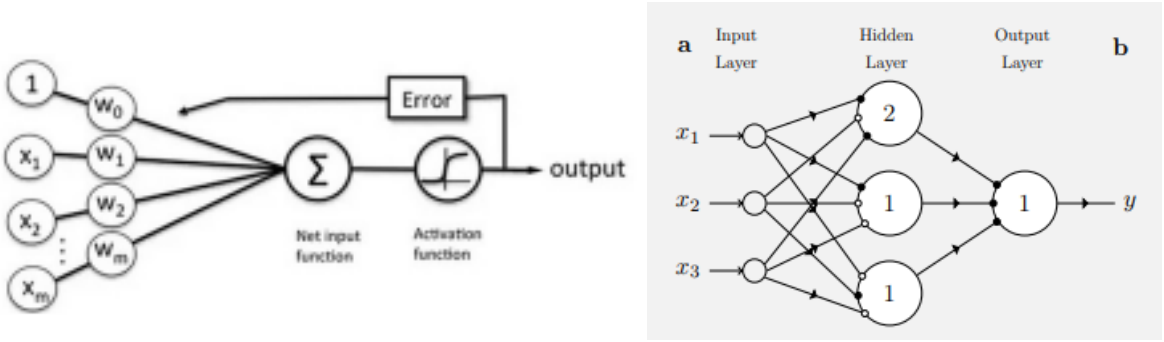

Рисунок 1.1 - Одно- та багатошарові нейронні мережі

Нейронна мережа - це складна обчислювальна модель, яка створюється шляхом поєднання багатьох простих параметричних моделей, які були розглянуті в попередніх розділах. Нейронні мережі можуть вивчати більш складні функції, ніж їх складові частини, оскільки параметри цих моделей навчаються разом для побудови оптимальної функції комбінації цих моделей.

Термін “персептрон” часто використовується для позначення базового модулю нейронної мережі, але це не зовсім вірно, оскільки багатошарова мережа може використовувати моделі різних типів (які часто мають нелінійну активацію), щоб досягти більшої потужності. У одношаровій нейронній мережі процес навчання є досить простим, оскільки помилку (або функцію втрати) можна обчислити як безпосередню функцію ваг, що дозволяє легко обчислити градієнт. У випадку багатошарових мереж проблема полягає в тому, що втрата є складною функцією комбінації ваг у попередніх шарах. Градієнт функції комбінації обчислюється за допомогою алгоритму зворотного поширення.

Алгоритм зворотного поширення використовує ланцюгове правило диференціального обчислення, яке обчислює градієнти помилки в термінах суми добутків локального градієнта за різними шляхами від вузла до виходу.

Хоча ця сума має експоненційну кількість складових (шляхів), її можна ефективно обчислити за допомогою динамічного програмування.

Алгоритм зворотного поширення є прямим застосуванням динамічного програмування. Він складається з двох основних фаз, які називаються фазами вперед і назад відповідно. Фаза вперед полягає в обчисленні виходу нейронної мережі для даного вхідного вектора, передаючи його через всі шари мережі. Фаза назад полягає в обчисленні градієнтів втрати по вагах мережі, використовуючи ланцюгове правило, і оновленні ваг за допомогою алгоритму оптимізації, наприклад, стохастичного градієнтного спуску.

Нейронні мережі можуть мати різні типи активаційних функцій, які визначають, як вихід нейрона залежить від його входу. Деякі приклади активаційних функцій - це сигмоїда, гіперболічний тангенс, ReLU, softmax тощо. Активаційні функції додають нелінійність до нейронної мережі, що дозволяє їй вивчати складніші функції, ніж лінійні моделі.

Нейронні мережі можуть мати різні архітектури, залежно від завдання, яке вони виконують. Деякі приклади архітектур - це згорткові нейронні мережі, рекурентні нейронні мережі, трансформери, генеративні змагальні мережі тощо. Кожна архітектура має свої переваги та недоліки, і їх вибір залежить від характеристик даних та цілей навчання.

Одним із поширених підходів є розробка програмної системи керування гучністю звуку на основі відео потоку жестів. Ідея полягає в тому, що користувач може регулювати гучність звуку свого пристрою за допомогою простих рухів рук, таких як підняття, опускання, поворот або змахування. Для реалізації цього проекту потрібно виконати наступні кроки:

- За допомогою камери захопити відео потік жестів користувача.
- За допомогою алгоритму розпізнавання об’єктів визначити положення та орієнтацію руки користувача на кожному кадрі відео.

- За допомогою вбудовування слів перетворити кожний кадр відео в вектор, який відображає характеристики жесту, такі як напрямок, швидкість, форма тощо.
- За допомогою класифікатора визначити, який жест відповідає якій команді керування гучністю, наприклад, розширити відстань між пальцями - збільшити гучність, звузити відстань між пальцями - зменшити гучність, поворот руки - перемкнути канал тощо.
- За допомогою інтерфейсу передати команду керування гучністю до пристрою користувача і виконати її.

1.2 Основні алгоритми та принципи аналізу відео потоку жестів

Mediapipe який відповідає за визначення та аналіз HandTraking для отримання даних відео потоку рух руки за допомогою комп’ютерного зору. Основний функціонал реалізується у модулі MediaPipe Hands.

Відстеження руху - це процес визначення положення та орієнтації руки користувача в просторі на основі виявлених ключових точок. Це дозволяє інтерпретувати жести руки як команди керування гучністю звуку. Для відстеження руху використовуються наступні кроки:

- Обчислення векторів напрямку між сусідніми ключовими точками руки, такими як кінчики пальців, долоня, зап’ясток тощо. Це дозволяє визначити відстань, кут і напрямок руху руки відносно початкового положення.
- Застосування фільтрів калмана або середнього рухомого для зменшення шуму та поліпшення точності відстеження. Це допомагає усунути

помилки, спричинені низькою якістю відео, освітленням, перешкодами тощо.

- Використання алгоритмів, що базуються на оптичному потоці, для визначення швидкості та прискорення руху руки. Оптичний потік - це розподіл очікуваного зсуву кожного пікселя між двома послідовними кадрами відео. Це дозволяє визначити, наскільки швидко та як рухається рука в просторі.
- Класифікація руху руки на основі визначених параметрів, таких як напрямок, швидкість, прискорення, форма тощо. Це дозволяє розрізняти різні типи жестів, такі як підняття, опускання, поворот, змахування тощо.

Виявлення обличчя та рук - це процес розпізнавання положення та розміру людського обличчя та руки на цифрових зображеннях. Це є необхідним першим кроком для всіх алгоритмів аналізу обличчя та руки, таких як вирівнювання, розпізнавання, верифікація та розбір. Також, виявлення обличчя та руки використовується в багатьох областях, таких як засноване на змісті відновлення зображень, кодування відео, відеоконференції, відеоспостереження за натовпом та інтелектуальні інтерфейси людина-комп’ютер. Для виявлення обличчя та руки використовуються наступні кроки:

- За допомогою камери захопити відеопотік жестів користувача.
- За допомогою глибоких нейронних мереж визначити ключові точки на обличчі та руках. Глибокі нейронні мережі - це моделі машинного навчання, які навчаються виявляти та класифікувати об’єкти на зображеннях за допомогою великих наборів даних. Для обличчя це включає виявлення 468 ключових точок, що представляють різні частини обличчя, такі як очі, ніс, рот, брови тощо. Для рук це включає виявлення 21 ключової точки, що представляє відомі частини руки, такі як кінчики пальців, долоня, зап’ясток тощо.

- За допомогою алгоритмів обробки зображень визначити межі (bounding box) кожного виявленого обличчя та руки. Межі - це прямокутні рамки, які охоплюють обличчя та руки на зображенні. Вони використовуються для визначення розміру та положення обличчя та руки відносно інших об’єктів на зображенні.

Виявлення обличчя та руки є складною задачею комп’ютерного зору, оскільки обличчя та руки є динамічними об’єктами, які мають високий ступінь змінності у своєму вигляді. У останні роки методи виявлення обличчя та руки досягли значного прогресу. Однак високоякісне виявлення обличчя та руки залишається викликаючою проблемою, особливо коли є багато дрібних обличчя та рук. Існують два типи підходів до виявлення обличчя та рук, (1) засновані на ознаках (feature-based) та (2) засновані на зображеннях (image-based) підходи.

1. Засновані на ознаках підходи намагаються знайти незмінні ознаки обличчя та рук для виявлення. Основна ідея базується на спостереженнях, що людське зору може легко виявляти обличчя та руки в різних позах та умовах освітлення, тому мають бути властивості або ознаки, які є послідовними, незважаючи на ці змінності. Було запропоновано широкий спектр методів для виявлення ознак обличчя та рук, а потім виведення наявності обличчя та рук.
2. Засновані на зображеннях підходи намагаються виявити обличчя та руки безпосередньо з зображень. Основна ідея полягає в тому, що обличчя та руки мають певні статистичні властивості, які можна використовувати для класифікації. Було запропоновано багато методів, які використовують глибокі нейронні мережі, які навчаються виявляти та класифікувати обличчя та руки з великих наборів даних.

Моделювання руки - це процес створення тривимірної моделі руки на основі виявлених ключових точок. Це дозволяє отримати інформацію про положення та орієнтацію руки в просторі, а також візуалізувати руку в різних проекціях. Для моделювання руки використовуються наступні кроки:

Вибір моделі руки. Модель руки - це математична абстракція, яка описує форму та структуру руки. Існує багато типів моделей руки, таких як сферичні, циліндричні, еліпсоїдні, полігональні, скелетні, м’які тощо. Кожна модель має свої переваги та недоліки, і її вибір залежить від цілей моделювання. Однією з найпоширеніших моделей руки є MANO (Model-based Approximation of Natural hand shape and pose), яка базується на скелетній структурі руки та використовує нейронну мережу для генерації мешу руки з 778 вершинами та 1538 трикутниками.

Відповідність ключових точок та моделі руки. Відповідність - це процес встановлення зв’язків між ключовими точками руки та вершинами моделі руки. Це дозволяє визначити параметри моделі руки, такі як кути суглобів, довжини кісток, форма долоні тощо. Існує багато методів для відповідності ключових точок та моделі руки, таких як оптимізація, регресія, класифікація тощо. Одним з прикладів є метод, який використовує глибоку нейронну мережу для регресії параметрів моделі руки з 21 ключової точки Рисунок 1.2.1.

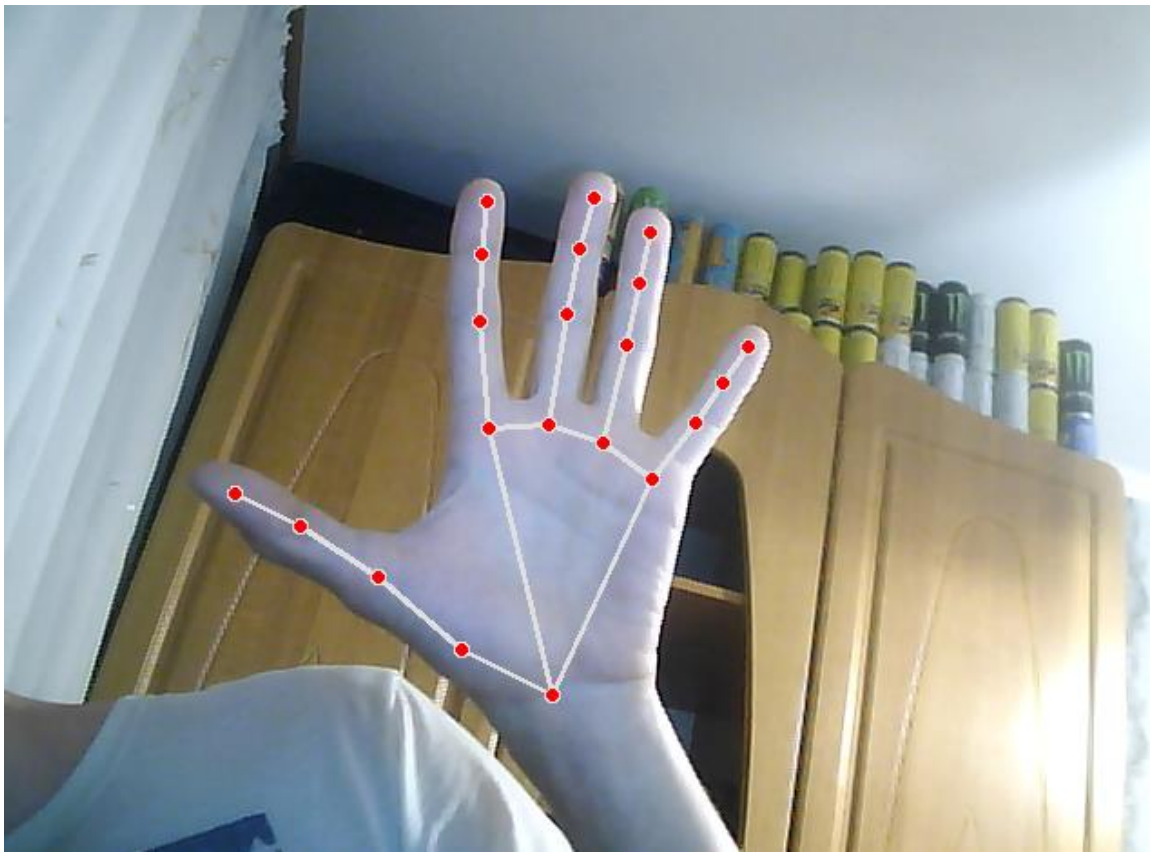

Рисунок 2.1 Модель руки з 21 точкою

Рендеринг моделі руки. Рендеринг - це процес генерації двовимірного зображення моделі руки з певної точки зору. Це дозволяє візуалізувати модель руки в різних проекціях, таких як перспектива, ортогональна, ізометрична тощо. Існує багато методів для рендерингу моделі руки, таких як растеризація, трасування променів, трасування шляхів тощо. Одним з прикладів є метод, який використовує OpenGL для рендерингу моделі руки з текстурами, освітленням та тінями.

Інтеграція з іншими додатками процес зв’язування результатів виявлення та розпізнавання жестів руки з іншими програмами або додатками, які можуть використовувати ці результати для виконання певних дій або функцій. Це дозволяє розширити можливості взаємодії з різними пристроями та сервісами за допомогою жестів руки. Для інтеграції з іншими додатками використовуються наступні кроки:

Вибір інтерфейсу для передачі результатів. Інтерфейс - це спосіб обміну даними між різними програмами або додатками. Інтерфейс повинен бути стандартизованим, надійним, безпечним та швидким. Існує багато типів інтерфейсів, таких як API, MQTT, WebSocket, Bluetooth, USB тощо. Кожен інтерфейс має свої переваги та недоліки, і їх вибір залежить від цілей інтеграції. Одним з прикладів є використання MQTT протоколу для передачі результатів до симульованого IoT пристрою за допомогою Home Assistant додатку[1].

Вибір формату для представлення результатів. Формат - це спосіб організації даних у структурованій та зрозумілій формі. Формат повинен бути сумісним, ефективним, гнучким та зручним. Існує багато форматів для представлення результатів, таких як JSON, XML, CSV, HDF5 тощо. Кожен формат має свої переваги та недоліки, і їх вибір залежить від типу та обсягу даних. Одним з прикладів є використання JSON формату для представлення координат ключових точок руки[2].

Вибір додатків для використання результатів. Додатки - це програми або сервіси, які можуть використовувати результати для виконання певних дій або функцій. Додатки повинні бути релевантними, корисними, інтерактивними та інтуїтивними. Існує багато додатків, які можуть використовувати результати, таких як віртуальна реальність, ігри, відеоредактори, робототехніка тощо. Кожен додаток має свої цілі та функції, і їх вибір залежить від потреб та інтересів користувачів. Одним з прикладів є використання додатку для керування гучністю звуку за допомогою жестів руки[3].

Інтеграція з іншими додатками є важливим етапом для розробки програмної системи керування гучністю звуку на основі відео потоку жестів, оскільки воно дозволяє розширити можливості взаємодії з різними пристроями та сервісами за допомогою жестів руки. Для інтеграції з іншими додатками використовуються методи, які використовують стандартизовані

інтерфейси, формати та додатки, які можуть приймати та обробляти результати в реальному часі.

Моделювання руки є важливим етапом для розробки програмної системи керування гучністю звуку на основі відео потоку жестів, оскільки воно дозволяє отримати інформацію про положення та орієнтацію руки в просторі, а також візуалізувати руку в різних проекціях. Для моделювання руки використовуються методи, які використовують глибокі нейронні мережі, які навчаються виявляти та аналізувати ключові точки руки з великих наборів даних. Моделювання руки вимагає високої точності та реалістичності, а також здатності працювати в реальному часі. Виявлення рук воно дозволяє розпізнавати та інтерпретувати жести руки як команди керування. Для виявлення руки використовується MediaPipe, який є бібліотекою для роботи з відстеженням рук за допомогою комп'ютерного зору. Відстеження руху дозволяє розпізнавати та інтерпретувати жести руки як команди керування. Для відстеження руху використовуються глибокі нейронні мережі, які навчаються виявляти та аналізувати ключові точки руки з великих наборів даних. Відстеження руху вимагає високої точності та швидкості обчислень, а також здатності працювати в реальному часі.

Для комп'ютерного бачення використовується функція cv::cornerHarris приймає на вхід зображення, яке перетворюється в сірі тони, і повертає матрицю, яка містить значення кутового відповідника для кожного пікселя. Кутовий відповідник визначає, наскільки сильно змінюється інтенсивність пікселів в різних напрямках. Якщо значення кутового відповідника велике для всіх напрямків, то це означає, що піксель належить куту.

Матриця Гарріса - це спосіб обчислення кутового відповідника. Вона використовує поняття градієнта, який є вектором, що показує напрямок і величину зміни інтенсивності. Градієнт можна розкласти на дві складові: x-компоненту і y-компоненту. Матриця Гарріса складається з чотирьох елементів, які обчислюються за формулою:

$$M = \begin{bmatrix} \sum_w I\frac{2}{x} & \sum_w I_x I_y \\ \sum_w I_x I_y & \sum_w I\frac{2}{y} \end{bmatrix}$$

де Ix і Iy - це x-компонента і y-компонента градієнта, а ∑w означає сумування по вікну w, яке є невеликою областю навколо пікселя.

З матриці Гарріса можна отримати кутовий відповідник за допомогою наступної формули:

$$R = det(M) - k \cdot trace(M)2$$

де det(M) - це визначник матриці, trace(M) - це сума діагональних елементів матриці, а k - це деяка константа, яка зазвичай дорівнює 0.04 або 0.06.

Якщо R додатне і велике, то піксель належить куту. Якщо R від'ємне, то піксель належить краю. Якщо R близьке до нуля, то піксель належить рівномірній області.

Теорема Пітo-Рунге-Кутта - це теорема, яка стверджує, що якщо диференціальне рівняння має гладкий розв'язок, то метод Рунге-Кутта четвертого порядку збігається до цього розв'язку з похибкою O(h4), де h - це крок інтегрування[4]. Метод Рунге-Кутта - це чисельний метод, який дозволяє знайти наближене значення розв'язку диференціального рівняння за допомогою ітераційної формули, яка використовує чотири проміжні значення, називані коефіцієнтами Рунге-Кутта[5]. Цей метод широко застосовується для розв'язування задач динаміки, таких як відстеження руху об'єктів на зображенні[6].

Формула методу Рунге-Кутта четвертого порядку має наступний вигляд:

$$y_{n+1} = y_n + \frac{h}{6}(k1 + 2k2 + 2k3 + k4)$$

де yn - це наближене значення розв’язку в точці xn, h - це крок інтегрування, а коефіцієнти Рунге-Кутта обчислюються таким чином:

$$k_1 = f(x_n y_n)$$
$$k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$$
$$k_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right)$$
$$k_3 = f(x_n + h, y_n + hk_3)$$

де f(x,y) - це права частина диференціального рівняння dxdy=f(x,y).

Моменти контуру - це скалярні величини, які характеризують форму та розташування контуру на зображенні. Контур - це крива, яка обмежує деяку область на зображенні. Моменти контуру можуть бути використані для розпізнавання, класифікації та порівняння об’єктів на зображенні[7].

Формула для обчислення моментів контуру використовує інтеграли Гріна, які дозволяють обчислити площу та центр маси області, обмеженої контуром. Інтеграли Гріна мають наступний вигляд[8]:

$$\int_C P(x,y)dx + Q(x,y)dy = \iint_D (\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y})dxdy$$

де C - це контур, D - це область, обмежена контуром, а P(x,y) та Q(x,y) - це деякі функції.

За допомогою інтегралів Гріна можна обчислити такі моменти контуру:

- Площа області D:

$$A = \frac{1}{2}\int_C xdy - ydx$$

- Координати центру маси області D:

$$x = \frac{1}{6A}\int_c x(xdy - ydx)$$

$$y = -\frac{1}{6A}\int_c y(xdy - ydx)$$

- Центральні моменти області D:

$$\mu_{pq} = \iint_D (x - x)^p (y - y)^q dxdy$$

Центральні моменти не залежать від положення об’єкта на зображенні, а тільки від його форми. Вони можуть бути нормалізовані за допомогою площі області, щоб отримати нормалізовані центральні моменти:

$$\eta_{pq} = \frac{\mu_{pq}}{A^1 + \frac{p+q}{2}}$$

Нормалізовані центральні моменти не залежать від масштабу об’єкта на зображенні, а тільки від його форми. Вони можуть бути використані для обчислення неваріантних моментів, які не залежать від повороту об’єкта на зображенні, а тільки від його форми. Наприклад, один з неваріантних моментів має наступну формулу:

$$\phi_1 = \eta_{20} + \eta_{02}$$

2 ОБҐРУНТУВАННЯ ВИБОРУ НАПРЯМКУ ДОСЛІДЖЕНЬ

2.1 Вибір технологій розробки

Вибрана була тема яка має великий потенціал в майбутніх система які можливо інтегрувати у наше повсякденне життя і покращить його. Тема програмна система керування гучністю звуку на основі відео потоку жестів

Штучний інтелект (ШІ) вимагає іншого підходу до програмування, ніж звичайні програмні проекти. Це пов’язано з використанням спеціальних технологій, необхідністю мати високий рівень навичок і здатністю проводити глибокі аналізи. Якщо потрібно реалізувати свої ідеї з використанням ШІ, потрібна мова програмування, яка буде стабільною, гнучкою та має багато інструментів для роботи з ШІ.

Python є такою мовою, тому не дивно, що багато проектів Python AI з’являються сьогодні. Python допомагає розробникам на всіх етапах роботи з проектами на основі ШІ: від розробки до розгортання та підтримки. Python має багато переваг для машинного навчання (ML) та проектів ШІ, таких як простота та зрозумілість, наявність великої кількості бібліотек і фреймворків для ШІ та ML, гнучкість, незалежність від платформи та активна спільнота. Python є однією з найпопулярніших мов програмування в світі. За даними опитування розробників Stack Overflow 2020, Python увійшов до топ-5 найбільш вживаних мов програмування. Це означає, що можливо легко знайти і найняти розробників, які мають необхідні навички для створення вашого проекту на основі ШІ. Python дозволяє писати короткий і зрозумілий код. Це дуже важливо для машинного навчання та ШІ, де алгоритми та робочі процеси можуть бути дуже складними. Python спрощує розробку надійних систем, дозволяючи розробникам фокусуватися на рішенні проблеми машинного навчання, а не на технічних деталях мови. Також Python є привабливим для

багатьох розробників, тому що його легко навчитися. Код Python схожий на людську мову, що полегшує створення моделей для машинного навчання.

Python не тільки підходить для машинного навчання та ШІ, але й має багато інших застосувань. Python використовується для веб-розробки, наукових обчислень, автоматизації, аналізу даних, розробки ігор, створення графіки та багато іншого. Python є універсальною мовою, яка може задовольнити різні потреби розробників.

Python також має велику кількість ресурсів для навчання та підтримки. Існує багато книг, курсів, відео, статей, блогів, підкастів, форумів, конференцій та інших матеріалів, які допоможуть вам вивчити Python або покращити свої навички. Python також має велику і дружню спільноту, яка готова ділитися своїми знаннями, досвідом, порадами та допомогою.

Розробка алгоритмів ШІ та ML складний і часомісткий процес. Розробникам потрібно мати добре організоване і надійне середовище, де вони можуть знаходити оптимальні рішення для свого коду. Python надає таке середовище, оскільки він є незалежним від платформи. Це означає, що код Python можна запускати на різних операційних системах, таких як Linux, Windows і macOS, без потреби вносити зміни. Код Python також можна компілювати в окремі виконувані файли, які можна легко розповсюджувати і використовувати на цих операційних системах без встановлення Python.

Крім того, розробники часто користуються сервісами, такими як Google або Amazon, для обробки своїх даних. Але іноді компанії та дослідники використовують свої власні машини з потужними графічними процесорами (GPU) для тренування своїх моделей машинного навчання. А те, що Python не залежить від платформи, робить це тренування більш доступним і простим. Завдяки своєму характеру мови програмування, яка не залежить від конкретної платформи, стає важливим інструментом для розробників у цьому контексті. Завдяки цій незалежності від платформи, розробники можуть легко

переміщати свій код між різними середовищами, що робить тренування моделей машинного навчання більш доступним і простим. Python також відомий своєю широкою підтримкою бібліотек для машинного навчання, таких як TensorFlow та PyTorch, які роблять розробку та тренування моделей ефективнішими.

Python також широко використовується для веб-розробки. За даними опитування розробників Python 2020, веб-розробка є найпоширенішим випадком використання Python, який складає понад 26% від загальної кількості. Але якщо об’єднати науку про дані та машинне навчання, вони становлять більше половини випадків використання Python. ШІ та ML дозволяють створювати програми, які можуть імітувати людську поведінку та інтелект. Такі програми використовуються для фільтрації спаму, рекомендації продуктів, пошуку інформації, особистої асистенції та виявлення шахрайства. І це тільки деякі приклади того, що можна зробити з ШІ та ML. Власники продуктів хочуть створювати ефективні програми, які можуть обробляти дані розумно і діяти як людина. Python допомагає їм у цьому, надаючи простий і потужний інструмент для розробки алгоритмів ШІ та ML.

Провівши аналіз всього перерахованого і з кількістю доступних інструментів для роботи з ШІ, перш за все було обрано бібліотеки які допоможуть з розробкою і підійшли б найкраще для неї. Ця система використовує комп’ютерний зір для виявлення, відстеження, моделювання та розпізнавання жестів руки, а також інтегрується з іншими додатками для виконання команд керування гучністю. Ця система розроблена з використанням технологій OpenCV та MediaPipe які підходять під цю мету найераще. Розглянемо детальніше кожен етап цієї системи:

Виявлення обличчя та рук. На цьому етапі за допомогою камери захоплюється відеопотік жестів руки користувача. За допомогою OpenCV виконується попередня обробка зображення, така як зменшення розміру, перетворення в сірий колір, застосування фільтрів тощо. За допомогою

MediaPipe визначаються ключові точки на обличчі та руках користувача. Ключові точки - це точки, які представляють різні частини обличчя та руки, такі як очі, ніс, рот, брови, кінчики пальців, долоня, зап’ясток тощо. MediaPipe використовує глибокі нейронні мережі для виявлення та відстеження ключових точок з високою точністю та швидкістю і яке підходить під наші потреби у розробці найкраще. За допомогою сучасних алгоритмів обробки зображень визначаються межі кожного виявленого обличчя та руки. Межі - це прямокутні рамки, які охоплюють обличчя та руки людини на зображенні. Вони використовуються для визначення розміру та положення у просторі обличчя та руки відносно інших об’єктів на зображенні.

Відстеження руху. На цьому етапі на основі виявлених ключових точок визначається положення та орієнтація руки користувача в просторі. Це дозволяє наші системі інтерпретувати жести руки як команди керування гучністю звуку. Для відстеження руху обчислюються вектори напрямку між сусідніми ключовими точками руки, такими як кінчики пальців, долоня, зап’ясток або те що нам потрібно у даний момент для керування системи. Це дозволяє визначити відстань, кут і напрямок руху руки відносно початкового положення. Застосовуються фільтри калмана або середнього рухомого для зменшення шуму та поліпшення точності відстеження з відео потоку. Використовуються алгоритми, що базуються на оптичному потоці які надходять від пристрою зчитування такі як камера , для визначення швидкості та прискорення руху руки.

Оптичний потік - це розподіл очікуваного зсуву кожного пікселя між двома послідовними кадрами відео. Це дозволяє системі визначити, наскільки швидко та як рухається рука в просторі. Класифікується рух руки на основі визначених параметрів, таких як напрямок, швидкість, прискорення, форма тощо. Це дозволяє розрізняти різні типи жестів для присвоєння їм певних команд які потім ітнтепритуються у код і розпізнаються систему для зміни тих

чи інших параметрів або керуванням машиною , такі як підняття, опускання, поворот, змахування тощо.

Зчитування та збереження зображень в OpenCV відбувається за допомогою функцій cv2.imread () та cv2.imwrite () відповідно. Ці функції приймають як параметри шлях до файлу зображення та режим читання або запису. Режим читання визначає, як OpenCV інтерпретує кольори зображення, наприклад, як сіре, RGB або BGR. Режим запису визначає, в якому форматі OpenCV зберігає зображення, наприклад, JPEG, PNG або TIFF. Формат зображення можна вказати за допомогою розширення файлу або за допомогою додаткового параметра, який вказує якість стиснення. Теоретично, OpenCV використовує бібліотеки, такі як libjpeg, libpng, libtiff тощо, для кодування та декодування зображень у різних форматах. Кожен формат має свої переваги та недоліки, такі як розмір файлу, якість зображення, підтримка кольорів, прозорість тощо. Наприклад, JPEG - це формат з втратами, який стискає зображення за рахунок деякої втрати деталей, але займає менше місця на диску. PNG - це формат без втрат, який зберігає зображення з високою якістю, але займає більше місця на диску. TIFF - це формат, який підтримує багатосторінкові зображення, але не підтримується багатьма програмами.

OpenCV надає функції для обробки та зміни розмірів зображень, що включає в себе зміну контрастності, яскравості, обрізку та інші операції. Як це відбувається. Зміна контрастності та яскравості:

- Контрастність - це різниця між найтемнішими та найсвітлішими пікселями на зображенні.
- Яскравість - це загальний рівень освітлення зображення. OpenCV дозволяє змінювати контрастність та яскравість зображення за допомогою наступної формули:

$$g(x) = \alpha f(x) + \beta$$

де g(x) - це вихідне зображення, f(x) - це вхідне зображення, α - це коефіцієнт контрастності, а β - це коефіцієнт яскравості. Ця формула застосовується до кожного пікселя на зображенні. Збільшення α підвищує контрастність, а збільшення β підвищує яскравість.

Обрізка зображення:

- Обрізка зображення - це процес видалення частини зображення, які не потрібні або не бажані. OpenCV дозволяє обрізати зображення за допомогою наступної формули:

$$g(x, y) = f(x + x0, y + y0)$$

де g(x,y) - це вихідне зображення, f(x,y) - це вхідне зображення, а (x0,y0) - це координати верхнього лівого кута області обрізки. Ця формула застосовується до кожного пікселя на зображенні.

Фільтри та операції обробки, такі як розмиття, розшарювання, згортання тощо. Розмиття зображення:

- Розмиття зображення - це процес зменшення різкості зображення, що призводить до розмивання деталей та контурів. Розмиття зображення може бути корисним для підвищення якості зображення, видалення шуму, зменшення розміру файлу тощо. OpenCV надає різні функції для розмиття зображення, такі як cv2.blur (), cv2.GaussianBlur (), cv2.medianBlur () та cv2.bilateralFilter (). Кожна з цих функцій використовує різні методи розмиття, які базуються на згортанні зображення з ядром розмиття.

- Згортання - це математична операція, яка застосовує ядро (або маску) до кожного пікселя зображення та обчислює середнє значення пікселів, які відповідають ядру.

- Ядро розмиття - це матриця, яка містить ваги для кожного пікселя. Наприклад, наступне ядро розмиття 3x3 застосовує однакову вагу 1/9 до кожного пікселя:

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Різні методи розмиття використовують різні ядра розмиття, які можуть мати різні розміри та ваги. Наприклад, гауссівське розмиття використовує ядро, яке має нормальний розподіл ваг, тобто центральний піксель має найбільшу вагу, а крайні пікселі мають найменшу вагу. Медіанне розмиття використовує ядро, яке вибирає медіанне значення пікселів, які відповідають ядру, замість середнього. Білатеральне розмиття використовує ядро, яке враховує не тільки просторову відстань між пікселями, але й різницю в інтенсивності, що дозволяє зберігати краї зображення.

Розшарювання зображення:

- Розшарювання зображення - це процес підвищення різкості зображення, що призводить до виділення деталей.

Розшарювання зображення може бути корисним для покращення якості зображення, підсилення країв, підвищення контрасту тощо. OpenCV надає функцію cv2.filter2D () для застосування довільного лінійного фільтру до зображення за допомогою згортання з ядром розшарювання. Ядро розшарювання - це матриця, яка містить ваги для кожного пікселя, який входить до згортання. Наприклад, наступне ядро розшарювання 3x3 застосовує від'ємну вагу -1 до сусідніх пікселів і позитивну вагу 5 до центрального пікселя:

$$\begin{bmatrix}-1 & -1 & -1\\ -1 & 5 & -1\\ -1 & -1 & -1\end{bmatrix}$$

2.2 Обґрунтування вибору середовища розробки програмної системи

Цей проект показує, як можна створити систему, яка дозволяє регулювати гучність звуку за допомогою жестів руки, зафіксованих відеокамерою. Для цього використовуються бібліотеки OpenCV і MediaPipe, які дозволяють обробляти відео потік і розпізнавати положення руки та пальців. За допомогою алгоритмів розраховується відстань між пальцями і відповідно до неї змінюється рівень гучності. Система може працювати в реальному часі і адаптуватися до різних розмірів і форм рук.

Комп’ютерний зір та розпізнавання образів - це галузі, де глибоке навчання показало високу ефективність. У сфері НЛП також спостерігається зростання застосування нових методів глибокого навчання. Раніше підходи машинного навчання для рішення проблем НЛП базувалися на неглибоких моделях (наприклад, SVM і логістичній регресії), які використовували великі розміри і розріджені ознаки. Однак за останні роки нейронні мережі, які працюють з щільними векторними представленнями, демонструють відмінні результати в різних задачах НЛП. Це пов’язано з успіхом вбудовування слів і методів глибокого навчання. Глибоке навчання дозволяє автоматично навчатися багаторівневим представленням функцій. На відміну від традиційних систем НЛП, заснованих на машинному навчанні, які сильно залежать від ручної роботи з функціями. Така ручна робота займає багато часу і часто неповна.

Для того щоб почати розробку на потрібно скористатись менеджером пакетів для Python і встановити потрібні пакети. PIP - це інструмент для керування пакетами Python, які не входять до стандартної бібліотеки Python, але доступні в індексі пакетів Python. Це альтернатива для easy_install. Якщо у вас є Python 2.7.9 (або новіший) або Python 3.4 (або новіший), то PIP вже

встановлений разом з Python, в іншому випадку вам потрібно встановити його самостійно.

PIP - це скорочення від “PIP Installs Python” або “PIP Installs Packages”. Це утиліта, яка працює з командного рядка, і дозволяє встановлювати, оновлювати або видаляти пакети PyPI за допомогою однієї команди: pip. Якщо ви працювали з іншими мовами, то ви, можливо, знайомі з терміном менеджер пакетів, наприклад, Ruby використовує Gem, JavaScript використовує npm, а .NET використовує NuGet. Pip є стандартним менеджером пакетів для Python.

Python вже має PIP вбудованим, якщо ви встановили новішу версію Python. Ви також можете перевірити, чи є PIP у вашій системі, виконавши наступну команду яка на русунку 2.1.

```
$ pip --version
pip 23.2.1 from C:\Users\54314\AppData\Roaming\Python\Python311\site-packages\pip (python 3.11)
```

Рисунок 2.1 – Команда перевірки версії PIP

Якщо ви отримаєте вивід, який містить номер версії PIP, наприклад, pip 23.2.1 або інший, це означає, що PIP встановлений і готовий до використання.

Якщо ви отримаєте помилку, це означає, що вам потрібно встановити PIP. Для встановлення PIP ви можете завантажити файл get-pip.py з [офіційного сайту PIP] і запустити його з командного рядка:

- python get-pip.py

Це встановить PIP і всі необхідні залежності. Після цього ви можете використовувати PIP для встановлення будь-яких пакетів Python, які вам потрібні. Наприклад, якщо ви хочете встановити бібліотеку requests, ви можете виконати наступну команду:

- pip install requests

Це завантажить і встановить пакет requests і всі його залежності. Ви також можете використовувати PIP для оновлення або видалення пакетів. Наприклад, якщо ви хочете оновити пакет requests до останньої версії, ви можете виконати наступну команду:

- pip install --upgrade requests

3 РОЗРОБКА СКЛАДОВИХ ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Базові операції використовуючи OpenCV і MediaPipe

OpenCV і MediaPipe - це дві потужні бібліотеки для комп’ютерного зору, які дозволяють виконувати різноманітні завдання, пов’язані з обробкою зображень та відео. OpenCV надає реалізовану на реальному часі бібліотеку комп’ютерного зору, інструменти та апаратне забезпечення. MediaPipe пропонує кросплатформні, налаштовуванні рішення ML для живих та потокових медіа.

Зчитування, запис та відображення зображень та відео з використанням функцій OpenCV, таких як

- cv2.imread
- cv2.imwrite
- cv2.VideoCapture
- cv2.VideoWriter
- cv2.imshow.

Застосування різних перетворень та фільтрів до зображень та відео, таких як зміна розміру, обертання, обрізання, розмиття, гострота, контраст, колір та інше, з використанням функцій OpenCV, таких як

- cv2.resize
- cv2.rotate
- cv2.warpAffine
- cv2.GaussianBlur
- cv2.Laplacian
- cv2.equalizeHist
- cv2.cvtColor

Виявлення, розпізнавання та відстеження облич, рук, тіла, об’єктів, жестів тощо з використанням рішень MediaPipe, таких як

- mp_face_detection,
- mp_hands,
- mp_pose,
- mp_objectron,
- mp_holistic

Ці рішення повертають набори ключових точок, які відповідають різним частинам об’єктів, і можуть бути відображені на зображеннях або відео з використанням функцій OpenCV, таких як

- cv2.circle,
- cv2.line,
- cv2.putText

Створення та використання власних графів MediaPipe, які складаються з калькуляторів, що виконують специфічні обчислення, і потоків, які переносять пакети даних між калькуляторами. Графи MediaPipe можуть бути створені за допомогою API конструювання графів (Protobuf) та виконані за допомогою API виконання графів (C++, Java, Obj-C).

Система, яку ми розробляємо, має свої переваги і недоліки, пов’язані з тим, що вона використовує CPU для всіх обчислень. З одного боку, це робить систему більш універсальною і доступною, оскільки вона не потребує наявності потужного GPU, який може бути дорогим або відсутнім на деяких машинах. Таким чином, система може працювати на різних машинах і задовольняти потреби більшої кількості користувачів. З іншого боку, це призводить до того, що CPU перевантажується і споживає багато ресурсів, що може погіршити продуктивність машини в цілому, особливо якщо система працює в фоновому режимі постійно. Це може створювати проблеми для

роботи з іншими системами, які також потребують великих обчислювальних ресурсів.

Для інсталяція MediaPipe потрібно зробити наступні кроки. Для цього потрібно скористатися pip.

```
54314@Admin MINGW64 /e/Mate/test_python
$ pip install mediapipe
Requirement already satisfied: mediapipe in c:\python311\lib\site-packages (0.10.7)
Requirement already satisfied: absl-py in c:\python311\lib\site-packages (from mediapipe) (2.0.0)
Requirement already satisfied: attrs>=19.1.0 in c:\python311\lib\site-packages (from mediapipe) (23.1.0)
Requirement already satisfied: flatbuffers>=2.0 in c:\python311\lib\site-packages (from mediapipe) (23.5.26)
Requirement already satisfied: matplotlib in c:\python311\lib\site-packages (from mediapipe) (3.8.0)
Requirement already satisfied: numpy in c:\python311\lib\site-packages (from mediapipe) (1.26.0)
Requirement already satisfied: opencv-contrib-python in c:\python311\lib\site-packages (from mediapipe) (4.8.1.78)
Requirement already satisfied: protobuf<4,>=3.11 in c:\python311\lib\site-packages (from mediapipe) (3.20.3)
Requirement already satisfied: sounddevice>=0.4.4 in c:\python311\lib\site-packages (from mediapipe) (0.4.6)
Requirement already satisfied: CFFI>=1.0 in c:\python311\lib\site-packages (from sounddevice>=0.4.4->mediapipe) (1.16.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\python311\lib\site-packages (from matplotlib->mediapipe) (1.1.1)
Requirement already satisfied: cycler>=0.10 in c:\python311\lib\site-packages (from matplotlib->mediapipe) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\python311\lib\site-packages (from matplotlib->mediapipe) (4.43.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\python311\lib\site-packages (from matplotlib->mediapipe) (1.4.5)
Requirement already satisfied: python-dateutil>=2.7 in c:\python311\lib\site-packages (from matplotlib->mediapipe) (2.8.2)
Requirement already satisfied: pycparser in c:\python311\lib\site-packages (from CFFI>=1.0->sounddevice>=0.4.4->mediapipe) (2.21)
Requirement already satisfied: six>=1.5 in c:\python311\lib\site-packages (from python-dateutil>=2.7->matplotlib->mediapipe) (1.16.0)

[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Рисунок 3.1.1 – Процес встановлення MediaPipe

Щоб інсталювати OpenCV ми теж використаємо pip

```
54314@Admin MINGW64 /e/Mate/test_python
$ pip install opencv-python
Requirement already satisfied: opencv-python in c:\python311\lib\site-packages (4.8.1.78)
Requirement already satisfied: numpy>=1.21.2 in c:\python311\lib\site-packages (from opencv-python) (1.26.0)

[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Рисунок 3.1.2 – Процес встановлення OpenCV

Для створення програмної системи, яка дозволить керувати звуком за допомогою жестів руки, зафіксованих відеокамерою, нам потрібно встановити деякі додаткові бібліотеки, які не входять до стандартної бібліотеки Python. Для цього ми можемо скористатися інструментом PIP, який дозволяє легко встановлювати, оновлювати або видаляти пакети Python, які доступні в індексі пакетів Python.

Ось всі бібліотеки які нам потрібні для встановлення.

- pip install opencv-python
- pip install mediapipe
- pip install numpy
- pip install pycraw
- pip install ctypes
- pip install comtypes
- pip install screen_brightness_control

Це завантажить і встановить пакети всі їх залежності. Після цього ми зможемо імпортувати ці бібліотеки в нашому коді Python і використовувати їх для реалізації нашої системи.

3.2 Клас HandDetector

Клас handDetector - це інструмент, який допомагає виявляти наявність рук на зображенні та визначати їхню структуру. Приймає на вхід зображення, яке може бути статичним або динамічним (відео потік). Використовує бібліотеку MediaPipe, яка містить модель машинного навчання, що навчена розпізнавати руки на зображенні та виділяти їхні ключові точки (landmarks). Використовує бібліотеку OpenCV, яка дозволяє обробляти зображення та відео, накладати на них різні ефекти та візуалізувати результати.

Клас handDetector виконує наступні кроки:

- Зчитує зображення або відео потік і перетворює їх у формат, який сумісний з MediaPipe.

- Використовує модель MediaPipe для виявлення рук на зображенні та отримання їхніх ключових точок (21 точка для кожної руки).
- Використовує OpenCV для відображення зображення та відео потоку з накладеними на них контурами рук та ключовими точками.
- Використовує OpenCV для визначення відстані між пальцями руки та відповідної зміни рівня гучності звуку.
- Повертає зображення або відео потік з візуалізованими результатами та рівнем гучності звуку.

Для розробки системи ми використовували методи які допоможуть нам створити стабільну роботу і легкий для писання і читання код.

__init__: Цей метод ініціалізує об’єкт handDetector. Він приймає декілька параметрів, які визначають режим роботи моделі MediaPipe Hands, максимальну кількість рук для виявлення, складність моделі, пороги виявлення та відстеження. Він також ініціалізує модель MediaPipe Hands та об’єкт для малювання кісток рук.

```
def __init__(self, mode=False, maxHands=1, modelComplexity=1, detectionCon=0.5, trackCon=0.5):

    self.mode = mode
    self.maxHands = maxHands
    self.detectionCon = detectionCon
    self.modelComplex = modelComplexity
    self.trackCon = trackCon
    self.mpHands = mp.solutions.hands
    self.hands = self.mpHands.Hands(self.mode, self.maxHands,self.modelComplex,self.detectionCon, self.trackCon)
    self.mpDraw = mp.solutions.drawing_utils
```

Рисунок 3.2.1 Метод __init__

mode: Якщо встановлено значення False, то буде відстежуватися всього 1 рука (найбільш помітна). Якщо True, то будуть відстежуватися всі руки в кадрі. В нашій системі ми вибрали значення False для параметра, який визначає, скільки рук треба відстежувати. Це означає, що наша система буде відстежувати тільки одну руку на зображенні. Ми зробили це для того, щоб зменшити навантаження на CPU і збільшити швидкість обробки. Це дозволить

нашій системі працювати на більшості сучасних машин і виконувати задачу регулювання гучності звуку за допомогою жестів руки. Ми вважаємо, що для цієї задачі достатньо використовувати одну руку, а не дві.

maxHands: Максимальна кількість рук, які будуть відстежуватися.

detectionCon: Поріг виявлення для моделі MediaPipe Hands.

trackCon: Поріг відстеження для моделі MediaPipe Hands.

modelComplex: Складність моделі MediaPipe Hands. Чим вище значення, тим складніша модель, але і точніше виявлення.

findHands: Цей метод приймає зображення та параметр draw, який визначає, чи слід малювати кістки рук на зображенні. Він перетворює зображення в формат RGB, обробляє його за допомогою моделі MediaPipe Hands та зберігає результати. Якщо на зображенні виявлено руки, він малює кістки рук на зображенні.

```
def findHands(self, img, draw=True):
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    self.results = self.hands.process(imgRGB)
    # print(results.multi_hand_landmarks)

    if self.results.multi_hand_landmarks:
        for handLms in self.results.multi_hand_landmarks:
            if draw:
                self.mpDraw.draw_landmarks(img, handLms,
                                           self.mpHands.HAND_CONNECTIONS)
    return img
```

Рисунок 3.2.2 Метод findHands

Перетворення зображення в формат RGB: OpenCV, бібліотека для обробки зображень, яку використовує цей код, за замовчуванням читає зображення в форматі BGR (синій, зелений, червоний). Однак модель MediaPipe Hands працює з зображеннями в форматі RGB (червоний, зелений, синій), тому зображення перетворюється з BGR в RGB за допомогою функції cv2.cvtColor.

Обробка зображення моделлю MediaPipe Hands: Модель MediaPipe Hands - це машинне навчання, яке було навчено виявляти руки на зображеннях. Вона приймає зображення в якості вхідних даних і повертає координати ключових точок (або “орієнтирів”) рук на зображенні. Цей процес використовує алгоритми машинного навчання, які були навчені виявляти руки на зображеннях. Ці алгоритми можуть бути досить складними і включати в себе багато різних кроків, включаючи різні види нейронних мереж, такі як згорткові нейронні мережі (CNN), які часто використовуються для обробки зображень. Малювання кісток рук на зображенні: Якщо на зображенні було виявлено одно або декілька руки, і параметр draw встановлено як True, метод findHands використовує функцію draw_landmarks з MediaPipe для малювання кісток рук на зображенні.

findPosition: Цей метод приймає зображення, номер руки та параметр draw, який визначає, чи слід малювати кістки рук на зображенні. Він створює список lmList для зберігання положення кісток рук. Якщо на зображенні виявлено руки, він визначає положення кожної кістки руки та додає його до списку. Якщо параметр draw встановлено в True, він також малює кістки рук на зображенні.

```
def findPosition(self, img, handNo=0, draw=True):

    lmList = []
    if self.results.multi_hand_landmarks:
        myHand = self.results.multi_hand_landmarks[handNo]
        for id, lm in enumerate(myHand.landmark):
            # print(id, lm)
            h, w, c = img.shape
            cx, cy = int(lm.x * w), int(lm.y * h)
            # print(id, cx, cy)
            lmList.append([id, cx, cy])
            if draw:
                cv2.circle(img, (cx, cy), 15, (255, 0, 255), cv2.FILLED)

    return lmList
```

Рисунок 3.2.3 Метод findPosition

Створення списку lmList: Цей список буде використовуватися для зберігання положення кісток рук. Кожен елемент списку - це список, що містить ідентифікатор кістки та її координати x та y на зображенні.

Перевірка наявності рук на зображенні: Якщо на зображенні було виявлено руки (тобто self.results.multi_hand_landmarks не є пустим), метод переходить до наступного кроку. В іншому випадку він повертає пустий список.

Визначення положення кісток рук: Для кожної кістки руки метод визначає її положення на зображенні. Він робить це, множачи координати x та y кістки (які представлені в нормалізованій формі, тобто в діапазоні від 0 до 1) на ширину та висоту зображення відповідно. Результат додається до списку lmList.

Малювання кісток рук на зображенні: Якщо параметр draw встановлено в True, метод також малює кістки рук на зображенні. Він робить це, малюючи коло на зображенні в положенні кожної кістки.

Цей процес не використовує жодних специфічних формул або теорем. Замість цього він використовує результати моделі MediaPipe Hands, яка використовує машинне навчання для виявлення рук на зображеннях. Ці результати включають координати кісток рук, які визначаються моделлю на основі шаблонів, які вона вивчила під час тренування. За допомогою цих координат метод findPosition може визначити положення кісток рук на зображенні.

Цей клас відкриває безліч можливостей для виявлення рук та визначення їхнього положення та кісток на зображеннях або відео в реальному часі. Використання цього класу може бути вкрай корисним в різних сферах, таких як віртуальна реальність, доповнена реальність, системи взаємодії з користувачем та багато інших. Однією з ключових переваг є можливість створювати системи, які взаємодіють з рухами користувача. Це може бути

використано для відтворення рухів користувача віртуальним або розширеним середовищем, що створює захопливий досвід для користувача. Також ця технологія відкриває можливості для розробки систем взаємодії з реальними об'єктами, де рухи користувача визначають дії системи.

Крім того, клас може бути використаний для аналізу жестів рук. Розпізнавання конкретних жестів відкриває можливості для створення інтерактивних систем, де рухи рук користувача слугують відправною точкою для виконання різних функцій чи взаємодії з віртуальним оточенням. Загалом, цей клас є потужним інструментом для розробки інтерактивних та захопливих застосувань у світі комп'ютерної взаємодії.

3.3 Функція main

Функція main є основною функцією програми, яка виконує всі необхідні операції для реалізації системи керування гучністю звуку за допомогою жестів руки. Для цього функція main використовує клас handDetector, який є частиною бібліотеки HandDectector[10]. Клас handDetector має методи для виявлення рук на відео зображенні, отримання їхніх ключових точок та кісток, а також визначення стану пальців (згорнуті або розгорнуті). Завдяки цьому функція main може отримувати відео потік з камери, обробляти його у реальному часі і визначати положення рук на кожному кадрі. На основі положення рук функція main може визначати, який жест руки виконує користувач, і відповідно змінювати гучність звуку на комп’ютері. Таким чином, функція main реалізує інноваційну систему керування гучністю звуку на основі відео потоку жестів.

```python
def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList[4])

        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime

        cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 255), 3)

        cv2.imshow("Image", img)
        cv2.waitKey(1)
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

cv2.destroyAllWindows()
if __name__ == "__main__":
    main()
```

Рисунок 3.3.1 Функція main

Для початку роботи з системою ми створюємо два об’єкти: cap і detector. Об’єкт cap використовує функцію cv2.VideoCapture, яка дозволяє отримувати відео зображення з веб-камери, підключеної до нашого комп’ютера. Об’єкт detector належить до класу handDetector, який містить методи для виявлення рук на відео та отримання їхніх ключових точок. Ці об’єкти допоможуть нам реалізувати функціонал нашої системи.

- Створення об’єкта cap за допомогою cv2.VideoCapture: OpenCV (Open Source Computer Vision Library) - це бібліотека алгоритмів комп’ютерного зору та машинного навчання. cv2.VideoCapture - це клас в OpenCV, який використовується для захоплення відео з веб-камери або файлу. У цьому випадку він використовується для захоплення відео з веб-камери. Параметр 0 вказує на те, що використовується веб-камера за замовчуванням.

- Створення об’єкта detector класу handDetector: handDetector - це клас, який було створено для виявлення рук на зображеннях або відео. Він використовує модель MediaPipe Hands для виявлення рук та визначення положення кісток рук. Об’єкт detector створюється без параметрів, що означає, що використовуються значення за замовчуванням.

Ці два об’єкти потім використовуються в циклі обробки відео для захоплення кадрів відео, виявлення рук на кожному кадрі та визначення положення кісток рук. За допомогою цієї ініціалізації можна створювати системи, які можуть відстежувати рухи рук користувача та використовувати цю інформацію для взаємодії з віртуальним середовищем.

Цикл обробки відео в цьому коді використовує конструкцію while True, яка створює безкінечний цикл, що продовжується до тих пір, поки його не перерветься. Розглянемо кроки цього циклу:

- Читання кадру: На кожній ітерації циклу викликається cap.read(), щоб отримати наступний кадр з відеопотоку. Зчитаний кадр зберігається в змінній img.
- Виявлення рук: Зображення кадру передається у метод detector.findHands(img), який використовує алгоритм виявлення рук для знаходження рук на кадрі відео.
- Визначення положення кісток рук: Після виявлення рук викликається метод detector.findPosition(img), який визначає положення кісток рук на кадрі.
- Відображення кадру: Отриманий кадр відео відображається за допомогою cv2.imshow("Image", img).
- Переривання циклу: Цикл продовжується, поки користувач не натисне клавішу 'q'. Коли це відбувається, викликається break, що призводить до переривання безкінечного циклу.

Цей підхід дозволяє обробляти кадри в реальному часі та виявляти руки на них, використовуючи бібліотеку OpenCV та детектор рук.

Метод cap.read() використовується для отримання кадрів з відеопотоку за допомогою бібліотеки OpenCV. Цей метод використовується для захоплення наступного кадру з відео, яке відтворюється за допомогою об'єкта cap, який створений за допомогою cv2.VideoCapture.

Кожен виклик cap.read() читає наступний кадр з відеопотоку. Результатом є два значення: success та img. Змінна success – це булеве значення, яке вказує на успішність читання кадру. Якщо відео завершилося або виникла яка-небудь проблема, success буде False. Змінна img містить сам кадр у вигляді зображення.

Цей підхід дозволяє робити обробку кадрів з відеопотоку в реальному часі, використовуючи OpenCV для подальшого аналізу чи візуалізації.

Метод detector.findHands(img) використовується для виявлення рук на кадрі відео та є складовою частиною класу handDetector. Описана функціональність створена для виявлення рук та визначення положення кісток рук на зображеннях або відео.

Коли кадр відео (зображення) передається у цей метод, він використовує модель MediaPipe Hands, щоб ефективно визначити руки на зображенні. Модель MediaPipe Hands є продуктом машинного навчання, навченою виявляти руки на зображеннях. Застосовуючи алгоритми комп'ютерного зору та техніки машинного навчання, ця модель надає високу точність виявлення рук на вхідних зображеннях.

Після успішного виявлення рук на зображенні метод findHands використовує внутрішню модель MediaPipe Hands для визначення положення кісток рук. Він використовує отримані координати для точного визначення положення кожної кістки руки на вхідному зображенні. Враховуючи цей підхід, метод findHands надає детальну інформацію про структуру та

положення рук на кадрі відео, що може бути використано для подальшого аналізу або взаємодії в реальному часі.

Метод detector.findPosition(img) використовується для визначення положення кісток рук на кадрі відео. Цей метод належить класу handDetector, створеного для виявлення рук та визначення положення кісток рук на зображеннях або відео. При передачі кадру відео (зображення) у цей метод, він використовує модель MediaPipe Hands для виявлення рук на зображенні. MediaPipe Hands - це модель машинного навчання, яка була навчена розпізнавати руки на зображеннях, використовуючи алгоритми комп’ютерного зору та машинного навчання. Після виявлення рук на зображенні, метод findPosition використовує модель MediaPipe Hands для визначення положення кісток рук. Він отримує координати кісток від моделі і додає їх до списку lmList, який містить координати всіх кісток руки.

Результатом цього методу є список lmList з координатами кісток руки. Цей список можна використовувати для різноманітних застосувань, таких як відстеження рухів рук, визначення жестів та інших взаємодій з комп’ютером чи віртуальним середовищем.

Метод cv2.imshow("Image", img) використовується для відображення обробленого кадру відео. OpenCV (Open Source Computer Vision Library) - це бібліотека, яка містить різноманітні алгоритми комп'ютерного зору та машинного навчання. Функція cv2.imshow призначена для відображення зображення в вікні. При виклику цієї функції передаються два аргументи: назва вікна і зображення для відображення. У цьому випадку назва вікна встановлена як "Image", а img є обробленим кадром відео.

Під час виклику цієї функції відбувається відображення зображення в новому вікні або оновлення вже існуючого вікна з вказаною назвою. Це дозволяє користувачу спостерігати за результатами виявлення рук та визначення положення кісток рук в реальному часі. Цей метод

використовується в циклі обробки відео, забезпечуючи відображення кожного обробленого кадру для спостереження за процесом в реальному часі.

Цикл обробки відео в даному коді безперервно виконується завдяки конструкції while True. Тобто, цей цикл буде тривати нескінченно довго, поки не буде виконана умова для його припинення.

Умова для припинення циклу перевіряється за допомогою виразу cv2.waitKey(10) & 0xFF == ord('q'). Функція cv2.waitKey(10) затримує виконання коду на 10 мілісекунд та повертає код натисканої клавіші. Оператор & 0xFF використовується для отримання останніх 8 біт коду клавіші, а ord('q') повертає ASCII-код символу 'q'. Умова стає істинною, якщо користувач натискає клавішу 'q'.

Якщо умова стає істинною, виконується оператор break, який негайно припиняє виконання циклу. Це означає, що обробка відео завершується, і програма переходить до наступного рядка коду після циклу. Ця можливість припинення циклу дає користувачеві контроль над тим, коли він хоче завершити обробку відео.

3.4 Файл VolumeControl

Цей файл є частиною нашої системи, яка дозволяє регулювати гучність звуку машини за допомогою жестів руки, зафіксованих веб-камерою. Для цього ми використовуємо різні функції і методи з модуля HandModule, який містить клас handDetector та інші корисні інструменти. За допомогою цих функцій і методів ми можемо отримувати відео потік з веб-камери, виявляти руки на ньому, отримувати ключові точки рук, визначати відстань між

пальцями і змінювати рівень гучності відповідно до неї. Все це відбувається у реальному часі, тому ми можемо керувати гучністю звуку динамічно і інтуїтивно.

```
devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(
    IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
volRange = volume.GetVolumeRange()
minVol = volRange[0]
maxVol = volRange[1]
```

Рисунок 3.4.1 Код для взаємодії системи з Windows

Код, який видно на рисунку 3.4.1, допомагає регулювати гучність системи за допомогою жестів руки, які програма визначає на відео. Цей код є складовою більшого проекту, а не окремою програмою. Змінна volume, яку цей код вираховує, потім передається до іншої частини проекту, яка змінює гучність системи.

Цей код дозволяє контролювати звукові налаштування Windows за допомогою бібліотеки pycaw (Python Core Audio Windows Library). Бібліотека pycaw надає функції для доступу до аудіо пристроїв, сесій, гучності та інших параметрів аудіо системи Windows. За допомогою цього коду ми можемо змінювати рівень гучності системи або окремих програм, які використовують аудіо.

Команда devices = AudioUtilities.GetSpeakers() використовує бібліотеку pycaw (Python Core Audio Windows Library) для взаємодії з аудіосистемою Windows.

AudioUtilities.GetSpeakers() є методом, який повертає об’єкт “гучномовців” (або пристроїв відтворення звуку), які використовуються системою за замовчуванням. Це можуть бути вбудовані гучномовці ноутбука,

зовнішні гучномовці, навушники або будь-який інший пристрій, який використовується для відтворення звуку на вашому комп’ютері.

Команда interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None) використовується для активації інтерфейсу контролю гучності звуку на пристроях відтворення.

devices.Activate() це метод, який активує інтерфейс (або службу) на пристрої відтворення звуку. Цей метод приймає три аргументи:

- IAudioEndpointVolume._iid_: Це ідентифікатор інтерфейсу, який потрібно активувати. В даному випадку, ми активуємо інтерфейс IAudioEndpointVolume, який дозволяє контролювати гучність звуку.
- CLSCTX_ALL: Це константа, яка вказує, що інтерфейс може бути активований в будь-якому контексті класу.
- None: Це місце для об’єкта, який використовується для управління контекстом активації. В даному випадку, ми не використовуємо жодного об’єкта, тому передаємо None.

Після активації інтерфейсу, він повертається методом Activate() і зберігається в змінній interface для подальшого використання. Зокрема, цей інтерфейс потім використовується для отримання та зміни рівня гучності системи.

Команда volume = cast (interface, POINTER (IAudioEndpointVolume) використовується для перетворення інтерфейсу, який було активовано раніше, на вказівник до цього інтерфейсу. Це дозволяє використовувати цей інтерфейс для контролю гучності.

cast() це функція з бібліотеки ctypes, яка використовується для безпечного перетворення об’єкта в інший тип об’єкта. Вона приймає два аргументи:

- interface: це об’єкт, який потрібно перетворити. В даному випадку, це інтерфейс, який було активовано раніше.
- POINTER(IAudioEndpointVolume): це тип об’єкта, в який потрібно перетворити. В даному випадку, ми перетворюємо інтерфейс в вказівник на інтерфейс IAudioEndpointVolume.

Після перетворення, вказівник на інтерфейс зберігається в змінній volume для подальшого використання.

Команда volRange = volume.GetVolumeRange() використовується для отримання діапазону гучності, який підтримується пристроєм відтворення звуку.

GetVolumeRange() це метод інтерфейсу IAudioEndpointVolume, який повертає мінімальне та максимальне значення гучності, які підтримуються пристроєм. Ці значення вимірюються в децибелах, де 0 дБ відповідає максимальному рівню гучності, а мінімальне значення є негативним числом, яке відповідає мінімальному рівню гучності.

Ці значення потім зберігаються в змінній volRange, яка є кортежем з двох елементів. Перший елемент (volRange[0]) - це мінімальне значення гучності, а другий елемент (volRange[1]) - це максимальне значення гучності. Вони будуть дорівнювати -65 для мінімум звуку і 0 для максимуму.

Команди minVol = volRange[0] та maxVol = volRange[1] використовуються для зберігання мінімального та максимального значень гучності, які підтримуються пристроєм відтворення звуку. volRange Ці значення були отримані від методу GetVolumeRange() інтерфейсу IAudioEndpointVolume.

Команда minVol = volRange[0] зберігає мінімальне значення гучності в змінній minVol.

Команда maxVol = volRange[1] зберігає максимальне значення гучності в змінній maxVol.

```
while True:
    success, img = cap.read()
    img = detector.findHands(img)
    lmList = detector.findPosition(img, draw=False)
    if len(lmList) != 0:
        #print(lmList[4], lmList[8])

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

        cv2.circle(img, (x1, y1), 15, (255, 255, 255), cv2.FILLED)
        cv2.circle(img, (x2, y2), 15, (255, 255, 255), cv2.FILLED)
        cv2.line(img, (x1, y1), (x2, y2), (255, 255, 255), 3)


        length = math.hypot(x2 - x1, y2 - y1)

        length = math.hypot(x2 - x1, y2 - y1)

        vol = np.interp(length, [50, 300], [minVol, maxVol])
        volBar = np.interp(length, [50, 300], [400, 150])
        volPer = np.interp(length, [50, 300], [0, 100])

        print(int(length), vol)
        volume.SetMasterVolumeLevel(vol, None)

    cv2.rectangle(img, (50, 150), (85, 400), (255, 0, 0), 3)
    cv2.rectangle(img, (50, int(volBar)), (85, 400), (255, 0, 0), cv2.FILLED)
    cv2.putText(img, f'{int(volPer)} ', (40, 450), cv2.FONT_HERSHEY_COMPLEX,
                1, (255, 0, 0), 3)

    cTime = time.time()
    fps = 1 / (cTime - pTime)
    pTime = cTime
    cv2.putText(img, f'FPS: {int(fps)}', (40, 50), cv2.FONT_HERSHEY_COMPLEX,
                1, (255, 0, 0), 3)

    cv2.imshow("Img", img)
    #cv2.waitKey(1)
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break
cv2.destroyAllWindows()
```

Рисунок 3.4.2 Цикл для зміні гучності

На рисунку 3.4.2 показано код, який створює безкінечний цикл, який буде адаптувати гучність звуку в залежності від вхідного відео потоку. Цей цикл буде виконувати наступні дії:

Команда success, img = cap.read() дозволяє отримати поточний кадр з веб-камери за допомогою об’єкта cap, який ми створили раніше за допомогою cv2.VideoCapture(0).

- cap.read() це метод, який читає наступний кадр з відеопотоку. Він повертає два значення:
- success: це булеве значення, яке показує, чи було зчитано кадр без помилок. Якщо кадр було зчитано правильно, success буде True. Якщо сталася помилка або якщо відеопотік скінчився, success буде False.
- img: це зображення кадру, яке було зчитано. Якщо success є True, img буде містити зображення кадру. Якщо success є False, img може бути None або містити некоректні дані.

Команда img = detector.findHands(img) застосовує об’єкт detector для пошуку руки на зображенні.

- detector це екземпляр класу handDetector, який ми створили раніше. Цей клас використовує модель машинного навчання для пошуку рук на зображеннях.
- findHands() це метод класу handDetector, який отримує зображення як аргумент і повертає зображення з пошуком рук. Він використовує модель машинного навчання для пошуку рук на зображенні, а потім додає на зображення ключові точки та скелет руки.

Результат цього методу (img з пошуком рук) потім зберігається в тій самій змінній img

Команда lmList = detector.findPosition(img, draw=False) застосовує об’єкт detector для отримання положення ключових точок руки на зображенні.

findPosition() це метод класу handDetector, який отримує зображення та параметр draw як аргументи. Параметр draw може бути True або False. Якщо draw=True, метод додає на зображення ключові точки та скелет руки. Якщо

draw=False, метод лише повертає координати ключових точок без додавання їх на зображення.

Цей метод повертає список lmList, який складається з координат ключових точок руки. Кожна ключова точка має три елементи: її номер та координати x та y на зображенні. if len(lmList) != 0:: Ця команда перевіряє, чи були виявлені ключові точки. Якщо так, то виконуються наступні команди.

Команди x1, y1 = lmList[4][1], lmList[4][2] та x2, y2 = lmList[8][1], lmList[8][2] дозволяють отримати координати двох ключових точок на руці.

- lmList - це список, який складається з координат ключових точок руки. Кожна ключова точка має три елементи: її номер та координати x та y на зображенні.

В цьому випадку, lmList[4] та lmList[8] відносяться до вказівного пальця та великого пальця руки відповідно. lmList[4][1] та lmList[8][1] - це координати x цих точок, а lmList[4][2] та lmList[8][2] - це координати y.

Ці координати потім присвоюються змінним x1, y1, x2, y2 відповідно.

- cv2.circle(img, (x1, y1), 15, (255, 255, 255), cv2.FILLED) та cv2.circle(img, (x2, y2), 15, (255, 255, 255), cv2.FILLED): Ці команди додають на зображення білі круги, які відповідають двом ключовим точкам на руці.
- cv2.line(img, (x1, y1), (x2, y2), (255, 255, 255), 3): Ця команда додає на зображення білу лінію, яка з’єднує дві ключові точки на руці.

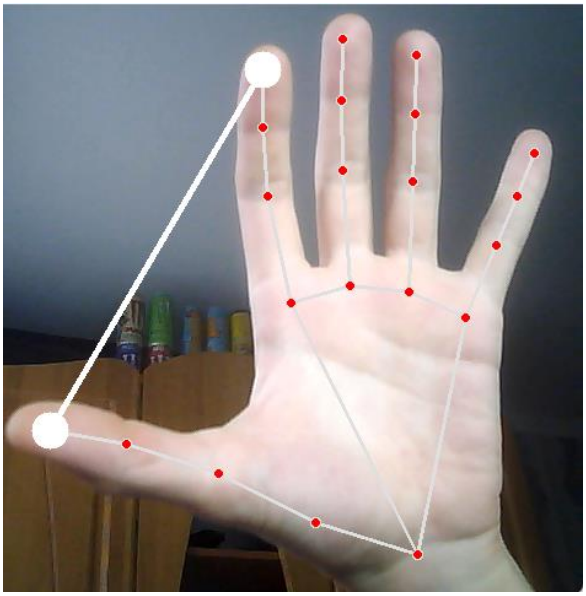

Рисунок 3.4.3 Відображення кругів і ліній

Команда length = math.hypot(x2 - x1, y2 - y1) дозволяє виміряти відстань між двома ключовими точками на руці. math.hypot() це функція з бібліотеки math, яка вираховує довжину гіпотенузи прямокутного трикутника за формулою Піфагора.

Вона отримує два аргументи, які є довжинами катетів трикутника, і повертає довжину гіпотенузи. В цьому випадку, x2 - x1 та y2 - y1 є довжинами катетів трикутника. Ці значення вираховуються як різниця між координатами x та y двох ключових точок на руці (зазвичай вказівного пальця та великого пальця). Тому, length = math.hypot(x2 - x1, y2 - y1) вимірює відстань між цими двома ключовими точками на руці. Ця відстань потім використовується для встановлення рівня гучності системи.

Команда vol = np.interp(length, [50, 300], [minVol, maxVol]) використовує функцію np.interp() з бібліотеки numpy для інтерполяції відстані між двома ключовими точками руки до відповідного рівня гучності.

np.interp() це функція, яка приймає три аргументи:

1. length: це значення, яке потрібно інтерполювати. В даному випадку, це відстань між двома ключовими точками руки.
2. [50, 300]: це діапазон вхідних значень для інтерполяції. В даному випадку, він вказує, що відстань між ключовими точками може варіюватися від 50 до 300 пікселів.
3. [minVol, maxVol]: це діапазон вихідних значень для інтерполяції. В даному випадку, він вказує, що рівень гучності може варіюватися від minVol до maxVol.

Отже, vol = np.interp(length, [50, 300], [minVol, maxVol]) обчислює рівень гучності, який відповідає відстані між двома ключовими точками руки. Цей рівень гучності потім використовується для встановлення гучності системи.

Команда volume.SetMasterVolumeLevel(vol, None) використовує об’єкт volume для встановлення рівня гучності системи на обчислене значення.

SetMasterVolumeLevel() це метод інтерфейсу IAudioEndpointVolume, який приймає два аргументи:

1. vol: це рівень гучності, який потрібно встановити. В даному випадку, це значення, яке було обчислено за допомогою функції np.interp() на основі відстані між двома ключовими точками руки.
2. None: це місце для контексту події, який може бути використаний для визначення, чи була зміна гучності викликана користувачем або програмою. В даному випадку, ми не використовуємо жодного контексту події, тому передаємо None.

```
cv2.rectangle(img, (50, 150), (85, 400), (255, 0, 0), 3)
cv2.rectangle(img, (50, int(volBar)), (85, 400), (255, 0, 0), cv2.FILLED)
cv2.putText(img, f'{int(volPer)} ', (40, 450), cv2.FONT_HERSHEY_COMPLEX,
            1, (255, 0, 0), 3)

cTime = time.time()
fps = 1 / (cTime - pTime)
pTime = cTime
cv2.putText(img, f'FPS: {int(fps)}', (40, 50), cv2.FONT_HERSHEY_COMPLEX,
            1, (255, 0, 0), 3)

cv2.imshow("Img", img)
#cv2.waitKey(1)
if cv2.waitKey(10) & 0xFF == ord('q'):
    break
```

Рисунок 3.4.4 Додаткові модулі системи

На рисунку 3.4.4 зображений код з додатковими елементами для зручності користувача. Це фрагмент коду для створення звукової панелі у вигляді прямокутника, який показує рівень звуку у цифрах від 0 до 100, кількість кадрів у секунду та умову зупинки програми. Можливий перефразування та розширення цього тексту такі:

Це частина коду, яка використовує бібліотеку cv2 для малювання прямокутної звукової панелі на зображенні. Звукова панель потрібна і вказує відсоток гучності від 0 до 100 що відповідає рівню який буде встановлений в операційні системі, який обчислюється зі змінної volBar. Код також має можливість для відображення кількість кадрів у секунду (fps) у верхньому лівому куті зображення, використовуючи модуль time для вимірювання часової різниці між кадрами, ці дані допомагають користувачу зрозуміти затримку у відгуку між відео і системою. Код показує зображення у вікні розмір якого задано в коді системи, використовуючи cv2.imshow, і чекає 10 мілісекунд на натискання клавіші, використовуючи cv2.waitKey. Якщо користувач натискає клавішу ‘q’, програма виходить з циклу і завершується це потрібно для переривання виконання щоб користувач мав змогу закрити вікно.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Створення програмного забезпечення повинно завжди відбуватись в умовах нанесення мінімальної шкоди здоров'ю розробника. Метою кваліфікаційної роботи магістра є розробка інформаційної системи для аналізу тональності тексту з використанням технології глибокого машинного навчання та мови програмування Python, що означає виконання великого обсягу роботи за персональним комп'ютером протягом тривалого часу, тому є доцільним відзначити важливість дотримання норм охорони праці та техніки безпеки під час роботи з електронно-обчислювальними машинами. Найбільш повним нормативним документом щодо забезпечення охорони праці користувачів ПК є "Державні санітарні норми і правила роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин" ДСанПіН 3.3.2.007-98. Даний документ описує: вимоги до виробничих приміщень для експлуатації ВДТ ЕОМ та ПЕОМ, гігієнічні вимоги до параметрів виробничого середовища приміщень, гігієнічні вимоги до організації і обладнання робочих місць, вимоги до режимів праці і відпочинку, вимоги до профілактичних медичних оглядів. Значне зниження наслідків несприятливої дії на програмістів шкідливих та небезпечних факторів можна досягти за допомогою дотримання цих вимог.

Природне освітлення в приміщеннях з ВДТ має здійснюватися через вікна, орієнтовані переважно на північ або північний схід і забезпечувати коефіцієнт природної освітленості не нижче ніж 1,5 %. Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски з поверхні екранів ПК і клавіатури повинні бути передбачені сонцезахисні пристрої, вікна повинні мати жалюзі або штори.

Основні вимоги до виробничого приміщення для експлуатації ВДТ:

– воно не може бути розміщено у підвалах та цокольних поверхах;

– площа на одне робоче місце в такому приміщенні повинна становити не менше 6,0 кв.м, а об’єм не менше 20,0 куб.м;

– воно повинно мати природне та штучне освітлення відповідно до ДБН В.2.5-28:2018;

– в ньому мають бути шафи для зберігання документів, магнітних дисків, полиці, стелажі, тумби тощо, з урахуванням вимог до площі приміщення;

– щоденно проводити вологе прибирання;

– поруч з приміщенням для роботи з ВДТ мають бути обладнані:

– побутова кімната для відпочинку під час роботи;

– кімната психологічного розвантаження.

Штучне освітлення в приміщеннях з робочим місцем, обладнаним ВДТ, має здійснюватися системою загального рівномірного освітлення. Як джерело штучного освітлення мають застосовуватись люмінесцентні лампи ЛБ.

Вимоги до освітлення приміщень та робочих місць під час роботи з ВДТ:

– освітленість на робочому місці повинна відповідати характеру зорової роботи, який визначається трьома параметрами: об’єктом розрізнення – найменшим розміром об’єкта, що розглядається на моніторі ПК; фоном, який характеризується коефіцієнтом відбиття; контрастом об’єкта і фону;

– необхідно забезпечити достатньо рівномірне розподілення яскравості на робочій поверхні монітора, а також в межах навколишнього простору;

– на робочій поверхні повинні бути відсутні різкі тіні;

– в полі зору не повинно бути відблисків (підвищеної яскравості поверхонь, які світяться та викликають осліплення);

– величина освітленості повинна бути постійною під час роботи;

– слід обирати оптимальну спрямованість світлового потоку і необхідний склад світла. Застосування світильників без розсіювачів та екрануючих гратів заборонено.

Гігієнічні норми до організації і обладнання робочих місць з ВДТ. При розташуванні елементів робочого місця користувача ВДТ слід враховувати:

– робочу позу користувача; – простір для розміщення користувача;

– можливість огляду елементів робочого місця;

– можливість ведення захистів;

– розміщення документації і матеріалів, що використовуються.

Конструкція робочого місця користувача ВДТ має забезпечити підтримання оптимальної робочої пози. Робочі місця з ВДТ слід так розташувати відносно вікон, щоб природне світло падало збоку, переважно зліва.

Робочі місця з ВДТ повинні бути розташовані від стіни з вікнами на відстані не менше 1,5м, від інших стін — на відстані 1 м, відстань між собою – не менше ніж 1,5 м.

Принтер повинен бути розміщений у зручному для користувача положенні, так, що максимальна відстань від користувача до клавіш управління принтером не перевищувала довжину витягнутої руки користувача.

Конструкція робочого стола повинна забезпечувати можливість оптимального роміщення на робочій поверхні обладнання, що використовується, з урахуванням його кількості та конструктивних

особливостей (розмір монітора, клавіатури, принтера, ПК та ін.) і документів, а також враховувати характер роботи, що виконується.

Вимоги до режимів праці і відпочинку при роботі з ВДТ. Під час роботи з ВДТ для збереження здоров'я працівників, запобігання профзахворюванням і підтримки працездатності встановлюються внутрішньо змінні регламентовані перерви для відпочинку.

Тривалість регламентованих перерв під час роботи з ЕОМ за 8-годинної денної робочої зміни залежно від характеру праці: 15 хвилин через кожну годину роботи – для розробників програм зі застосуванням ЕОМ; 15 хвилин через кожні дві години – операторів із застосуванням ЕОМ; 10 хвилин після кожної години роботи за ВДТ для операторів комп'ютерного набору.

У випадках, коли виробничі обставини не дозволяють застосовувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 годин.

Для зниження нервово-емоційного напруження, втомленості зорового аналізатора, для поліпшення мозкового кровообігу і запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ, які передбачені ДСанПіН 3.3.2.007-98, в тому числі і для сеансів психологічного розвантаження у кімнаті з відповідним інтер'єром та кольоровим оформленням.

Виконання вимог ДСанПіН 3.3.2.007-98 повинне стати нормою всіх користувачів, які працюють над даним проектом.

4.2 Фактори, що впливають на функціональний стан користувачів комп’ютерів.

Надійність системи "людина – комп’ютер" значною мірою визначається функціональним станом людини. Психофізіологічні та емоційні перенапруження, втома людини-оператора можуть призвести в комп’ютеризованих системах керування до помилок і як наслідок – до значних економічних втрат.

Згідно зі статистичними даними від 40 до 75% аварій літаків зумовлено людським фактором. Відмови комп’ютеризованої системи керування рухом 57 залізничного транспорту, на гірничо-збагачувальних комбінатах з вини операторів становлять понад 50% їх загальної кількості, причому значна їх частина спричинена невідповідністю функціонального стану оператора складності виконуваної роботи.

Трудова діяльність користувачів комп’ютерів відбувається у певному виробничому середовищі, яке впливає на їх функціональний стан. Найбільш значимі – фізичні фактори виробничого середовища, до яких належать електромагнітні хвилі різних частотних діапазонів, електростатичні поля, шум, параметри мікроклімату та ціла низка світлотехнічних показників.

Трудовий процес суттєво впливає на психофізіологічні можливості користувачів комп’ютерів, оскільки їх діяльність характеризується значними статичними фізичними навантаженнями; недостатньою руховою активністю; напруженнями сенсорного апарату, вищих нервових центрів, які забезпечують функції уваги, мислення, регуляції рухів. Окрім того, трудовий процес користувачів комп’ютерів відзначається значними інформаційними навантаженнями.

Професійні якості та виробничий досвід, які визначають внутрішні засоби діяльності, обумовлюють надійну та безпомилкову діяльність користувачів комп’ютерів, дозволяють знаходити безпечні методи розв’язання виробничих завдань навіть у нестандартних ситуаціях.

Зовнішні засоби діяльності, які в основному визначаються ергономічними показниками щодо організації робочого місця, формою та параметрами його елементів, просторового розташування основного і допоміжного устаткування, можуть суттєво знизити фізичні та психофізіологічні навантаження, що діють на користувачів комп’ютерів.

Оскільки робота користувачів комп’ютерів найчастіше проходить за активної взаємодії з іншими людьми, то виникають питання раціоналізації міжособистісних стосунків. Цей комплекс питань порушує як психологічні, так і соціально- 58 психологічні аспекти трудових взаємовідносин, які також є факторами "ризику", що відчутно впливають на функціональний стан користувачів комп’ютерів.

Визначення та вивчення факторів, що впливають на функціональний стан користувачів комп'ютерів дозволить виділити основні причини виникнення станів напруженості, стомлення, стресу і здійснити відповідні профілактичні заходи.

Отже, до основних факторів, що впливають на функціональний стан користувачів комп’ютера належать:

1. середовище – характеризується такими шкідливими факторами:

1.1. фізичні: електромагнітні хвилі різних частотних діапазонів, електростатичні поля, шум, параметри мікроклімату та ряд світлотехнічних показників;

1.2. хімічні: пил, шкідливі хімічні речовини, які виділяються при роботі принтера і копіювальної техніки;

1.3. біологічні: підвищений вміст в повітрі патогенних мікроорганізмів, особливо у приміщенні з великою кількістю працюючих, при недостатній вентиляції, особливо у період епідемій;

1.4. психофізіологічні: напруження зору та уваги, інтелектуальні та емоційні навантаження, тривалі статичні навантаження і монотонність праці.

2. трудовий процес - характеризується значними статичними фізичними навантаженнями; недостатньою руховою активністю; напруженнями сенсорного апарату, вищих нервових центрів, які забезпечують функції уваги, мислення, регуляції рухів. Окрім того, трудовий процес користувачів комп'ютерів відзначається значними інформаційними навантаженнями;

3. внутрішні засоби діяльності – це професійні риси та виробничий досвід, які обумовлюють надійну та безпомилкову діяльність користувачів комп'ютерів, дозволяють знаходити безпечні методи розв'язання виробничих завдань навіть у нестандартних ситуаціях; 59

4. зовнішні засоби діяльності - визначаються ергономічними показниками щодо організації робочого місця, форми та параметрів його елементів, просторового розташування основного і допоміжного устаткування, які можуть суттєво знизити фізичні та психофізіологічні навантаження, що діють на користувачів комп'ютерів;

5. соціально-психологічні фактори трудових взаємовідносин. У професійних операторів частіше зустрічаються порушення органів зору, опорно-рухового апарату, центральної нервової, серцево-судинної, імунної та статевої систем, захворювання шкіри. Зафіксована значна кількість скарг операторського персоналу на загальне недомагання, передчасне стомлювання, головний біль, порушення функцій органів зору, які здійснювали несприятливий психофізіологічний вплив на самопочуття та працездатність операторів.

Сучасна професія користувача ВДТ належить до розумової праці, яка характеризується: високою напруженістю зорових функцій; одноманітною позою; великою кількістю стереотипних високо координованих рухів, що виконуються лише м'язами кистей рук на фоні малої загальної рухової активності; значним нервово-емоційним компонентом, особливо в умовах дефіциту часу; роботою з великими масивами інформації, що викликає активізацію уваги та інших вищих психічних функцій. Крім того, при роботі з дисплеями на електронно-променевих трубках виникає вплив на користувача цілої низки факторів фізичної природи — електростатичні поля, радіочастотне та рентгенівське випромінювання тощо.

Діяльність професіоналів можна поділити на три групи:

1. Діяльність, яка пов'язана з виконанням нескладних багаторазово повторюваних операцій, що не вимагають великого розумового напруження. Наприклад, робота операторів комп'ютерного набору, працівників довідкових служб. 60

2. Діяльність, яка пов'язана із здійсненням логічних операцій, що постійно повторюються. Це робота інженера-економіста, інженера-проектувальника, оператора автоматизованого виробництва.

3. Діяльність, коли в процесі роботи необхідно приймати рішення за відсутності заздалегідь відомого алгоритму. Наприклад, робота інженера програміста, диспетчерів руху залізничного транспорту, аеропортів тощо.

У користувачів, які інтенсивна використовують комп'ютер в умовах значних розумових напружень досить часто (40—70%) виникають психологічні та поведінкові порушення (нервозність, роздратування, тривога, нерішучість, замкнутість тощо). Серед користувачів ВДТ в США і Європі значного поширення набуло специфічне захворювання, яке отримало назву синдром комп'ютерного стресу (СКС). СКС супроводжується головним болем, запаленням очей, алергією, роздратованістю, млявістю і депресією.

Інформаційне перевантаження користувачів ВДТ супроводжується низкою специфічних захворювань, які називають інформаційними. Першим симптомом їх є головний біль. Дослідження, проведені в США, Німеччині, Швейцарії та інших країнах, показали, що робота з обслуговування ВДТ супроводжується підвищеним напруженням зору, інтенсивністю і монотонністю праці, збільшенням статичних навантажень, нервово-психічним напруженням, впливом різного виду випромінювана та ін. Внаслідок цього серед операторів ВДТ, як зазначають фахівці Всесвітньої організації охорони здоров'я, частіше, ніж в інших групах працюючих, трапляються такі професійні захворювання, як передчасна стомлюваність, погіршення зору, м'язові і головні болі, психічні й нервові розлади, хвороби серцево-судинної системи, онкологічні захворювання та ін. Вважається, що стан організму операторів ВДТ визначається комплексним впливом факторів трудового процесу і середовища, значення яких є неоднаковим. На операторів з малим стажем роботи на ВДТ домінуючий вплив чинять фактори середовища, а на операторів зі стажем понад 5 років - фактори трудового процесу.

Комп'ютерний зоровий синдром (КЗС) - комплекс порушень здоров'я, який може виникати у користувачів персональних комп'ютерів (ПК). У користувачів ПК дуже поширені кон'юнктивіти і блефаріті, патогенетична пов'язані з КЗС. Синдром розвивається при умові, що робоче місце організовано неправильно - у користувача незручне крісло, відсутні пюпітри для паперів, підставки для ніг та кистей рук, не встановлена висота і нахил монітора відносно очей, відстань від очей до екрана. За таких умов тіло людини при роботі займає вимушене положення: спина статично напружена, шия витягнута, плечі жорстко фіксовані. Напружені м'язи погіршують кровоток у сонних артеріях, а недостатнє кров забезпечення головного мозку веде до очманіння, появи головного болю. На фоні шийного остеохондрозу з'являється відчуття випирання очних яблук, туману в очах, мушок та райдужних кіл у полі зору. Розвитку КЗС сприяє поганий мікроклімат

приміщення, значна загальна іонізація та мікробне забруднення, а також куріння.

Національною радою з наукових досліджень США для стану зорового дискомфорту був уведений термін "астенопія", який означає "будь-які суб'єктивні зорові симптоми чи емоційний дискомфорт, що є результатом зорової діяльності". Симптоми астенопії були класифіковані на "очні" (біль, печія та різь в очах, почервоніння повік та очних яблук, ломота у надбрівній частині тощо) та "зорові" (пелена перед очима, мерехтіння, швидка втома під час зорової роботи та ін.).

ВИСНОВОК

Ця магістерська робота присвячена розробці та реалізації інформаційної системи, яка дозволяє користувачеві керувати гучністю звуку за допомогою жестів руки, визначених з відеопотоку. Основною перевагою цієї системи є те, що вона не потребує додаткового обладнання, а лише веб-камеру та доступ до API, який надає функціонал для аналізу відеоданих та взаємодії з машинними командами. Система розроблена на мові програмування Python з використанням бібліотек OpenCV та MediaPipe, які демонструють високу ефективність та точність у розпізнаванні жестів руки.

У процесі проектування системи було проведено глибокий аналіз предметної області, визначено основні вимоги та завдання, обрано оптимальні технічні засоби та розроблено алгоритми для реалізації функцій системи. У процесі розробки системи було створено вихідний код, який забезпечує зчитування відеопотоку, виявлення та класифікацію жестів руки, виконання відповідних команд для регулювання гучності звуку. Також було проведено підготовку даних для навчання та тестування системи. У процесі тестування системи було перевірено її функціональність, точність, швидкодію та стійкість до різних умов роботи. Результати тестування показали, що система працює без помилок та відповідає всім поставленим вимогам.

Готова інформаційна система є повноцінним продуктом, який може бути використаний для зручного керування гучністю звуку за допомогою жестів руки. Система має інтуїтивний інтерфейс, який надає користувачеві зрозумілий вихідний результат роботи системи. Система також має високу гнучкість та масштабованість, оскільки може бути встановлена на будь-який пристрій, який підтримує API.

У частині роботи «Охорона праці та безпека в надзвичайних ситуаціях» було розглянуто основні аспекти безпечної роботи, описано правила охорони праці, які слід дотримуватися під час розробки та використання системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vision projects | computer science & engineering. University of Nevada, Reno. [Електронний ресурс] / David Feil-Seifer – Режим доступу до ресурсу: https://www.unr.edu/cse/research/intelligent-systems/vision-projects
2. Учасники проектів Вікімедіа. Метод рунге – кутти – вікіпедія. Вікіпедія. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Метод_Рунге_–_Кутти.
3. AI technologies used in robotics - datasciencecentral.com. Data Science Central. [Електронний ресурс] – Режим доступу до ресурсу: https://www.datasciencecentral.com/ai-technologies-used-in-robotics/
4. Creating a hand tracking module using python, opencv, and mediapipe. Engineering Education (EngEd) Program | Section. [Електронний ресурс] – Режим доступу до ресурсу: https://www.section.io/engineering-education/creating-a-hand-tracking-module/
5. Hand landmarks detection guide | MediaPipe | Google for Developers. Google for Developers. [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker
6. Учасники проектів Вікімедіа. Гаррісів афінний виявляч областей – Вікіпедія. Вікіпедія. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.m.wikipedia.org/wiki/Гаррісів_афінний_виявляч_областей

7. 3D hand pose with mediapipe and tensorflow.js. The TensorFlow Blog. [Електронний ресурс] – Режим доступу до ресурсу: https://blog.tensorflow.org/2021/11/3D-handpose.html
8. Utrecht University Student Theses Repository Home. [Електронний ресурс] – Режим доступу до ресурсу: https://studenttheses.uu.nl/bitstream/handle/20.500.12932/30508/thesis.pdf?sequence=2
9. Стручок В. С. БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ / Володимир Сергійович Стручок., 2022. – 154 с.
10. Петрик М.Р. Проєктування програмного забезпечення на основі аналізу вимог та інструментальних засобів розробки IBM Rational Software Architect (від Вимог до коду) Науково-методичний посібник. Тернопіль: Вид-во ТНТУ ім. Івана Пулюя.-2022.- 560с.

ДОДАТКИ