ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА
(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем
(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття рівня вищої освіти «магістр»
(рівень вищої освіти)

Інформаційна технологія генерації персоналізованих рекомендацій у спорті
Information technology for generating personalized recommendations in sports

Виконав: студент денної форми навчання
спеціальності 126 – Інформаційні системи та технології .
(шифр і назва напряму підготовки, спеціальності)
Освітня програма «Інформаційні системи та технології»
(назва освітньої програми)
Гальчинський Максим Вячеславович
(прізвище, ім'я, по-батькові)

Керівник к.ф.-м.н., доц. Петрушина Т. І.
(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент д.т.н., проф. Малахов Є. В.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент д.т.н., проф. Арсірій О. О.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№___ від «___»_____ 2024 р.
Завідувач кафедри
_____ _Євгеній МАЛАХОВ_
(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___
протокол № __ від «___» ____ 2024 р.
Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)
Голова ЕК
_____ _Володимир ВИЧУЖАНІН_
(підпис) (ім'я, прізвище)

Одеса - 2024

АНОТАЦІЯ

У роботі представлено результати дослідження, спрямованого на покращення аналізу спортивної техніки, зокрема техніки плавання, за допомогою сучасних методів комп’ютерного зору та машинного навчання. Розроблено інформаційну систему, яка здатна визначати ключові точки тіла спортсмена у тривимірному просторі на основі відео послідовності, виділяти патерни руху та порівнювати їх.

Метою даної роботи є підвищення ефективності надання рекомендацій в плаванні шляхом розробки інформаційної технології виявлення та візуалізації помилок на основі порівняння відео послідовності у техніці плавання.

Об'єктом дослідження є процес виділення патерну руху спортсмена з відео послідовності та його порівняння з іншим патерном.

Предметом дослідження є методи та алгоритми для розпізнавання правильної техніки на основі патернів руху спортсмена.

Для реалізації системи створено Docker середовище. Проєкт виконано мовою програмування Python із застосуванням низки спеціалізованих бібліотек, зокрема: PyQt6 для створення користувацького інтерфейсу, cv2 для обробки зображень, MMPose для роботи з ключовими точками тіла, json_tricks для ефективного збереження та обміну даними, matplotlib для візуалізації, а також Open3D для побудови тривимірних моделей та візуалізації рухів.

Результати дослідження демонструють нові можливості в аналізі спортивної техніки та визначення її недоліків. Розроблена технологія сприяє підвищенню якості тренувального процесу та є першим кроком для побудови автоматизованої рекомендаційної системи. Крім того це є важливим етапом у побудові кіберфізичних систем, які базуються на фізичних моделях рухів живих організмів, що відкриває нові перспективи для інтеграції біологічних і фізичних процесів у єдину інформаційну інфраструктуру.

ABSTRACT

This paper presents the results of a study aimed at improving the analysis of sports techniques, in particular swimming techniques, using modern computer vision and machine learning methods. An information system has been developed that is capable of identifying key points of an athlete's body in three-dimensional space based on a video sequence, identifying movement patterns, and comparing them.

The aim of this work is to increase the efficiency of providing recommendations in swimming by developing information technology for detecting and visualizing errors based on the comparison of video sequences in swimming techniques.

The object of research is the process of extracting a sportsman's movement pattern from a video sequence and comparing it with another pattern.

The subject of the study is methods and algorithms for recognizing the correct technique based on the athlete's movement patterns.

A Docker environment was created to implement the system. The project was implemented in the Python programming language with the use of a number of specialized libraries, in particular: PyQt6 for creating a user interface, cv2 for image processing, MMPose for working with key points of the body, json_tricks for efficient data storage and exchange, matplotlib for visualization, and Open3D for building three-dimensional models and visualizing movements.

The results of the study demonstrate new possibilities in analyzing sports techniques and identifying its shortcomings. The developed technology contributes to improving the quality of the training process and is the first step in building an automated recommendation system. In addition, it is an important step in the construction of cyber-physical systems based on physical models of the movements of living organisms, which opens up new prospects for the integration of biological and physical processes into a single information infrastructure.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

CNN – Convolutional Neural Networks

GCN – Graph Convolutional Network

HRNet – High-resolution network

LSTM – Long Short-Term Memory

PCA – Principal Component Analysis

RNN – Recurrent Neural Networks

SMPL – Skinned Multi-Person Linear Model

TCN – Temporal Convolutional Networks

КФС – Кіберфізичні Системи

ВСТУП

У багатьох видах спорту, зокрема в плаванні, техніка розглядається як один з ключових факторів успіху в змаганнях і одна з основних характеристик, що відзначає кращих спортсменів [1].

Техніка плавання – це патерн руху, засвоєний спортсменом впродовж тривалого часу в м'язовій пам'яті, який вирішує задачу переміщення за найменший час на певній дистанції у воді. Цього можна досягнути за рахунок мінімізації впливу зовнішніх сил на тіло людини.

Для покращення власної техніки спортсмени звертаються до спеціалістів. Традиційно тренер аналізує техніку через призму власного досвіду та набутих технічних знань. На більш високому рівні необхідна допомога вчених. Вони моделюють рух спортсмена в 3D просторі і надають рекомендації щодо покращення техніки. Для цього використовується спеціальне обладнання, таке як сенсори, прикріплені до тіла, та різні методи порівнянь.

Фахівці, які досліджують техніку окремих спортсменів, зазвичай стикаються з кількома проблемами.

Перша з них є виділення патерну руху, особливо складним стає це питання в тих видах спорту, де рухи не є циклічними.

Наступна проблема – кількісно оцінити техніку таким чином, щоб стало можливим об'єктивне порівняння технік спортсменів.

Крім того, у більшості видів спорту техніка є координаційною схемою всього тіла, отже, ретельний аналіз техніки вимагає запису, аналізу та інтерпретації великих наборів параметрів. Зосередження уваги на кількох ключових змінних може становити ризик упередженості.

Ще одне завдання полягає в тому, щоб визначити, яка зміна техніки може бути корисною для конкретного спортсмена [2, 3].

Якщо вийти за рамки розгляду виключно техніки плавання, і, розглянути дану тему в більш широкому ракурсі, а саме як прояв фізичних характеристик у живих організмах. Можна вважати вирішення перелічених проблем як один з етапів при побудові кіберфізичних систем.

Термін кіберфізичні системи (КФС) був запропонований Хелен Гілл для опису інтеграції обчислень з фізичними процесами [4].

Хоча живі організми зазвичай вважаються суто біологічними, вони також демонструють фізичні характеристики, які є важливими для їхньої функціональності. Ці прояви одночасно демонструють цілу низку фізичних явищ, включаючи механічні, хімічні, електромагнітні та оптичні. Водночас вони часто цілком здатні до обчислень і регулярної комунікації. Живі системи можуть бути чудовим джерелом натхнення для розробки нових КФС, і, аналогічно, досягнення в галузі КФС можуть надати нам кращі інструменти для покращення нашого розуміння життя і самих себе [5].

В такому випадку виділення патерну руху можна застосувати при побудові ідеальної моделі. А методи порівняння патернів – для наближення КФС до цієї ідеальної моделі.

Отже, ціллю даної роботи є підвищення ефективності надання рекомендацій в плаванні шляхом розробки інформаційної технології виявлення та візуалізації помилок на основі порівняння відео послідовності у техніці плавання.

Об'єктом дослідження є процес виділення патерну руху спортсмена з відео послідовності та його порівняння з іншим патерном.

Предметом дослідження є методи та алгоритми для розпізнавання правильної техніки на основі патернів руху спортсмена.

Для досягнення поставленої цілі необхідно виконати наступні задачі:

1) Дослідити методи комп’ютерного зору та машинного навчання для обробки відеоматеріалів;

2) Розробити алгоритм виділення патерну руху на основі обробленої відео послідовності;

3) Розробити алгоритм порівняння техніки спортсмена з натренованою моделлю ідеальної техніки;

4) Реалізувати розроблені алгоритми та моделі у вигляді інформаційної системи;

5) Провести аналіз ефективності розробленої інформаційної системи з урахуванням вищезазначених критеріїв на декількох відео послідовностях.

Для вирішення цих задач запропоновано використання методів машинного навчання та комп’ютерного зору в єдиній інформаційній системі.

Результатом даного дослідження є інформаційна система, здатна візуалізувати ключові точки тіла людини у трьохвимірному просторі, виокремлювати патерни руху та порівнювати їх між собою. На основі такого порівняння експерт може надати рекомендації по покращенню техніки плавання.

Вона наближає до нового підходу в побудові КФС, де система калібрується в залежності від ідеальної моделі, основаної на фізичних процесах живих істот. А також робить більш доступним отримання рекомендацій щодо техніки спортсменів, так як в процесі аналізу виключено використання спеціального обладнання.

1 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ

В цьому розділі розглядаються методи та технології отримання даних з відео матеріалів, їх подальшої обробки та візуалізації. Так як відео складається з послідовності кадрів, то для деяких задач, таких як візуалізація, достатньо дослідити методи та технології, які працюють із зображеннями.

1.1 Методи комп’ютерного зору

1.1.1 Методи на основі LSTM

Мережі довготривалої короткочасної пам'яті (LSTM) (рис. 1.1) є спеціалізованим типом рекурентних нейронних мереж (RNN), призначених для збереження інформації протягом тривалих періодів і навчання довгострокових залежностей [6].

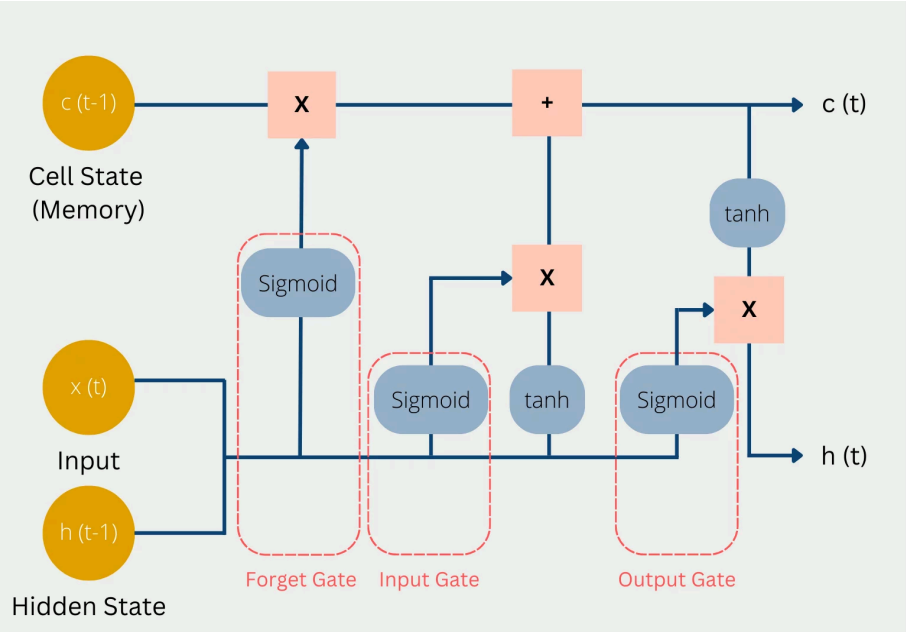

Рисунок 1.1 – Архітектура LSTM [7]

Однією з ключових концепцій в LSTM є прихований стан (hidden state), який містить короткострокову пам'ять. Разом з ним LSTM використовує стан клітини (cell state), що представляє собою довгострокову пам'ять. Ці стани оновлюються на кожному кроці обробки послідовності даних.

Також LSTM містить спеціальні блоки пам'яті, які дозволяють зберігати та вибірково оновлювати інформацію протягом тривалого часу. Кожен блок пам'яті складається з трьох основних частин:

1. Вхідний вентиль (input gate) – визначає, яка нова інформація повинна бути збережена в блоці пам'яті.
2. Вентиль забування (forget gate) – вирішує, яку інформацію слід забути або видалити з блоку пам'яті.
3. Вихідний вентиль (output gate) – визначає, яку інформацію з блоку пам'яті слід використовувати в даний момент [8].

Завдяки такій структурі застосування LSTM в комп'ютерному зорі є ефективним інструментом для вирішення задач відеоаналізу дій людини. Так, ключові точки скелету, отримані з кожного кадру відео, подаються на вхід LSTM, де кожна точка представляє певний суглоб. Далі моделюються часові взаємозв'язки між послідовними кадрами, враховуючи як поточну інформацію, так і попередні рухи. Це в результаті дозволяє аналізуючи зміни положення ключових точок у часі.

1.1.2 Методи, засновані на TCN

Тимчасові згорткові мережі (TCN) є різновидом нейронних мереж, призначених для обробки послідовних даних.

Основна ідея TCN полягає у використанні згорткових шарів для обробки послідовностей даних. На відміну від LSTM, послідовні дані обробляються паралельно, що пришвидшує процес навчання.

Так як TCN здатний обробляти послідовності, то цілком доречно використовувати його для моделювання часових залежностей між різними кадрами відео. Проте в такій ситуації є ряд недоліків.

По-перше, TCN зосереджений на часовій обробці і може не повною мірою враховувати просторові залежності між ключовими точками в кожному кадрі.

По-друге, на відміну від рекурентних мереж, які обробляють дані послідовно, TCN не мають природної ієрархічної структури для розуміння послідовності подій.

Це може призвести до втрати важливої інформації про взаємозв'язки між подіями, що відбуваються у різних частинах кадру.

Щоб компенсувати ці недоліки, TCN часто комбінують з іншими архітектурами нейронних мереж, такими як згорткові нейронні мережі (CNN). Наприклад, CNN використовується для екстракції просторових ознак з кожного кадру, а TCN потім моделює часові залежності між цими ознаками. Це поєднання дозволяє отримати більш повну картину подій як у просторі, так і в часі.

1.1.3 Методи на основі GCN

Згорткові мережі графів (GCN) – це тип нейронних мереж, призначених для роботи з графовою структурою даних. З назви видно, що GCN походить від CNN. Головна відмінність між цими мережами полягає в тому, що кількість вузлів може варіюватися, а вузли не впорядковані (рис. 1.2). Кожен вузол графа передає інформацію своїм сусідам через згорткові операції, що дозволяє враховувати структуру графа та зв'язки між вузлами.

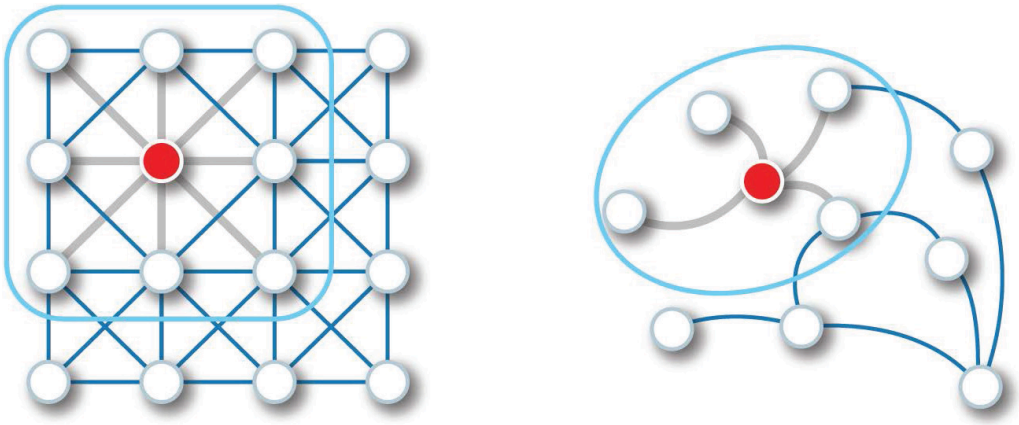

Рисунок 1.2 – Порівняння CNN та GCN [9]

Така структура дозволяє ефективно обробляти нерегулярні дані, такі як тривимірні об’єкти. Таким чином цей метод можна використовувати для аналізу переміщення людини у відео шляхом моделювання скелетних ключових точок як вузлів графа, а зв'язків між ними як ребер. Кожен кадр відео представлений графом, де GCN агрегують інформацію з сусідніх вузлів для врахування просторових залежностей. Для обробки часових залежностей результати GCN передаються на вхід RNN або TCN, що дозволяє моделі прогнозувати дії людини на основі аналізу послідовності кадрів у відео.

1.1.4 Методи на основі HRNet

HRNet (High-Resolution Network) – це архітектура глибокого навчання, призначена для вирішення проблем оцінки поз людини. Ключова особливість HRNet полягає в здатності підтримувати високороздільні представлення на всіх етапах, що є важливим для точної оцінки поз як у 2D, так і в 3D контекстах.

HRNet працює за новою архітектурною схемою, яка усуває проблему поступового зменшення роздільної здатності зображень і потім їх збільшення. Замість цього підтримуються кілька роздільних здатностей паралельно, що дає змогу зберігти високороздільні представлення на кожному етапі. Ця архітектура

виявилася надзвичайно ефективною для завдань оцінки поз, оскільки вона дозволяє мережі зберігати точну просторову інформацію та захоплювати багатошарові ознаки. Використання високороздільних представлень гарантує, що деталі людського тіла, такі як суглоби та кінцівки, залишаються чіткими, що особливо важливо для точної детекції поз в складних зображеннях, таких як ті, що містять перекриття або складні фони.

Архітектура HRNet складається з кількох ключових компонентів, зокрема високороздільних модулів, які побудовані на принципі паралельних згорткових мереж. Ці модулі взаємопов'язані, і за допомогою ряду стратегій злиття модель може комбінувати інформацію для отримання якісного представлення ознак.

Однією з основних переваг HRNet є здатність покращувати точність оцінки поз в різних контекстах. В трьохвимірному просторі HRNet виграє завдяки своїм високороздільним ознакам, що дозволяють точніше розуміти просторові характеристики і глибину. Це особливо важливо при застосуванні захоплення рухів.

Крім того, HRNet адаптована для інших завдань комп'ютерного зору. Її архітектура використовується для завдань, таких як семантична сегментація та виявлення об'єктів, що демонструє універсальність підходу до навчання з високою роздільною здатністю. Ця гнучкість робить HRNet цінним інструментом для широкого спектра застосувань в комп'ютерному зорі.

Однак, попри переваги в точності, HRNet має деякі недоліки. Використання високороздільних представлень вимагає більше пам'яті та обчислювальних ресурсів порівняно з традиційними архітектурами, що може бути проблемою в умовах обмежених ресурсів. Крім того, залежність моделі від високороздільних ознак може призвести до повільного часу обробки, особливо при роботі з великими наборами даних або в режимі реального часу [10, 11].

1.1.5 Методи на основі трансформерів

Методи, засновані на трансформерах, складають категорію архітектур глибокого навчання, які використовують механізми самоуваги (self-attention) для обробки послідовних даних. Цей механізм дозволяє кожному елементу послідовності враховувати всі інші елементи під час обробки, що забезпечує контекстуальне розуміння, оскільки кожен елемент «уважно стежить» за всіма іншими елементами.

Трансформери спочатку призначалися для заміни рекурентних шарів в умовах машинного перекладу. Його мета полягала в тому, щоб виправити обмеження архітектури моделювання послідовностей шляхом обробки цілих послідовностей одночасно (на відміну від RNN, які є послідовними за своєю суттю), дозволяючи подальше розпаралелювання. Крім того, він усуває локальне упередження традиційних архітектур, таких як CNN, і натомість вивчає взаємодію нелокальних контекстів вхідних даних.

В результаті багато варіацій трансформерів стали звичайним явищем у комп’ютерному зорі і, зокрема, у відео [12].

Використання CLS токену може допомогти зібрати всю послідовність відеокадрів у один вектор представлення, що може стати в нагоді для класифікації загальної техніки плавання або виявлення відхилень від ідеальної форми.

За допомогою MSK токенів, які призначені для маскування певних кадрів або частин кадрів, можна змусити модель навчитися контексту від оточуючих кадрів. Такий підхід дає змогу зрозуміти положення тіла плавця та безперервність руху.

1.2 Фреймворки комп’ютерного зору

1.2.1 OpenPose

OpenPose – це потужна бібліотека з відкритим вихідним кодом, призначена для оцінки поз людей за допомогою комп’ютерного зору. Вона здатна визначати ключові точки тіла, обличчя, рук і ніг як для окремих осіб, так і для груп людей на зображеннях або відео. OpenPose підтримує одночасну обробку кількох людей у кадрі, дозволяючи будувати точні скелетні моделі навіть у складних умовах.

Головним досягненням OpenPose є можливість обробляти зображення в реальному часі. Це стало можливим завдяки інноваційній архітектурі нейронних мереж, яка поєднує розпізнавання й регресію ключових точок. OpenPose використовує двоступеневий підхід: спочатку знаходяться можливі ключові точки на основі теплових карт, а потім між ними встановлюються зв’язки, формуючи скелет. Цей метод є надзвичайно ефективним для точного визначення поз навіть у динамічних сценах.

OpenPose легко інтегрувати в проекти завдяки підтримці таких мов програмування, як Python і C++. Крім того, бібліотека підтримує апаратне прискорення за допомогою GPU, що забезпечує її високу продуктивність навіть для великих обсягів даних.

Однією з важливих особливостей OpenPose є її модульність. Наприклад, користувачі можуть обирати, які частини тіла аналізувати (тільки тіло, обличчя чи руки), що дозволяє адаптувати бібліотеку до конкретних задач.

Головним недоліком фреймворку є те, що для отримання 3D координат ключових точок тіла людини необхідно мати кілька камер і точно відкалібровану систему для створення стереопари. Якщо стереозображення недоступне або має низьку якість, точність реконструкції значно погіршується.

Окрім цього, хоча OpenPose добре працює з 2D оцінкою поз, в динамічних умовах, як у спорті, швидкі рухи можуть призводити до втрати точності при визначенні ключових точок, що впливає на 3D реконструкцію [13].

1.2.2 MMPose

MMPose є сучасним фреймворком для оцінки поз людини, розробленим як універсальний інструмент для дослідження, моделювання та аналізу поз у дво- та тривимірному просторі. Його модульна архітектура дозволяє адаптувати різні компоненти моделі. Основною метою MMPose є забезпечення високої точності прогнозування ключових точок тіла, що досягається завдяки використанню HRNet і підтримці багатьох сучасних підходів до моделювання, таких як top-down і bottom-up методики.

Фреймворк розроблений для інтеграції з великим числом наборів даних, включаючи COCO, MPII та інші, що значно полегшує навчання та оцінювання моделей, забезпечуючи їх відповідність міжнародним стандартам. Зокрема, використання глибоких нейронних мереж дозволяє MMPose вирішувати завдання від простого визначення ключових точок до комплексного тривимірного моделювання руху.

Інтеграція SMPL (Skinned Multi-Person Linear Model) з HRNet використовується у MMPose для оцінки тривимірних поз. Це метод реконструкції тривимірної форми і пози людини безпосередньо з двовимірного зображення. Такий підхід дозволяє об'єднувати етапи оцінки пози та форми в єдиний процес, що не потребує попередньої ідентифікації ключових точок. Використання SMPL забезпечує фізично правдоподібні результати, враховуючи анатомічні та кінематичні обмеження людського тіла [14].

Завдяки широким можливостям інтеграції з PyTorch, MMPose дозволяє ефективно працювати із завданнями обробки відеоданих, включаючи аналіз

рухів. Додатково, фреймворк пропонує інструменти для обробки часових рядів, що є критично важливим для задач моніторингу рухів у реальному часі.

Проте є і недоліки. MMPose дуже залежний від високоякісних вхідних даних для точної оцінки пози. Що може заважати правильно визначати ключові точки людини в складних умовах, таких як водне середовище, та вносити шум в дані [15].

1.2.3 TransFusion

TransFusion – це рішення для об'єднання інформації з LiDAR-камер в контексті виявлення 3D-об'єктів. Воно вирішує проблему обробки зображень в поганих умовах, таких як погане освітлення і неспіввісність датчика.

TransFusion використовує CNN і Transformers. Transformers забезпечує адаптивне злиття інформації, визначаючи, де і які ознаки потрібно виділити із зображення, а CNN закріплює ці залежності.

На відміну від традиційних методів, які покладаються на жорсткі асоціації між точками LiDAR і пікселями зображення за допомогою калібрувальних матриць, TransFusion використовує механізм м'яких асоціацій. Цей підхід дозволяє йому більш ефективно обробляти складні умови зображення [16].

1.3 Метод порівняння рухів

Аналіз головних компонент (PCA) використовується для кількісного аналізу складних рухових патернів шляхом їх розкладу на основні компоненти руху. Цей підхід дозволяє об'єктивно та кількісно оцінити рухи, подібно до того, як це роблять тренери, але з використанням математичних моделей.

Процес роботи алгоритму головних компонент (PCA) включає кілька ключових етапів. Спершу збираються дані, що описують техніку рухів, у вигляді

високовимірних векторів, де кожен вектор представляє координати ключових точок людини у різні моменти часу. Потім ці дані центруються, тобто від кожного виміру віднімається його середнє значення. Далі обчислюється коваріаційна матриця даних, яка відображає взаємозалежність між різними вимірами. З цієї матриці отримуються власні вектори і власні значення, що визначають напрямки головних компонент. Власні вектори ранжуються за відповідними власними значеннями від найбільшого до найменшого, що дозволяє ідентифікувати компоненти, які пояснюють найбільшу частку варіації в даних. Нарешті, початкові дані проектуються на ці компоненти, що дозволяє зменшити їх розмірність і отримати новий набір змінних, які компактно і лаконічно описують основні характеристики рухів.

Використання цього методу дозволяє об'єктивно оцінити складні рухові патерни, зводячи багатовимірні дані до кількох основних рухів. Однак деякі основні рухи представляють собою комбінацію двох або більше рухів, які людина-спостерігач розглядає як незалежні. Якщо, наприклад, вертикальний рух і поворот верхньої частини тіла відбуваються по фазі під час циклу руху, то PCA може бути не в змозі їх розділити [17].

1.4 Технології візуалізації даних

1.4.1 Point cloud representation

Хмара точок – це множина 3D-точок, де кожна точка представлена кортежем з координатами x, y та z. Окрім координат точки можуть мати додаткові атрибути, такі як колір або інтенсивність (рис. 1.3).

Рисунок 1.3 – Приклад візуалізації хмари точок

Зазвичай хмари точок можна отримати при використанні камер глибини, наприклад, Lidar-камер, або стереоскопічних камер, де камера отримує декілька RGB-зображень з різних камер приблизно в один і той самий час.

Основним недоліком цієї технології є те, що неможливо отримати явної інформації про топологію об'єкта. Тобто, хоча точки можуть точно описувати форму поверхні, вони не містять інформації про те, як ці точки з'єднані між собою. Це робить обробку та аналіз точкових хмар складнішими і вимагає додаткових алгоритмів для реконструкції поверхні або пошуку структурних зв'язків між точками.

Іншим недоліком є складність поєднання даних з різних джерел та їх інтеграція в єдину модель. Адже дані є невпорядковані та нерегулярні, що унеможливлює використання методу CNN при навчанні. А різнорідність призводить до того, що для одного навчального набору різні хмари точок можуть містити різну кількість точок.

1.4.2 Mesh representation

Mesh representation являє собою технологію візуалізації даних, за якої об'єкти відображаються у вигляді сітки, що складається з вершин, ребер і граней. Кожна вершина в сітці визначається своїми координатами, а ребра описують, як ці вершини з'єднані між собою для формування граней, які зазвичай мають трикутну форму. Така форма дозволяє швидко трансформувати та відтворювати об’єкт.

Основна перевага mesh representation полягає в її здатності точно і компактно описувати геометрію об'єктів. З іншого боку, створення і обробка сіток є досить ресурсоємними процесом. Зазвичай вони є результатом постобробки необроблених хмар точок і створюються вручну.

Одним із поширених форматів для зберігання сіток данних є PLY-файл. Кожен такий файл містить секцію заголовка та секцію даних. Перший рядок містить ключове слово ply, що вказує на те, що це PLY-файл. Другий рядок format ascii 1.0 вказує, що файл має кодування ASCII з версією 1. Далі в кожному рядку оголошується типи даних та кількість рядків цього типу. Так, в даному випадку оголошено 8 вершин, які мають властивості x, y і z типу float32. Таким чином, кожна вершина представляє одну 3D-точку. Рядок element face 12 означає, що другим типом даних є грані, а наступний вираз property list uint8 int32 vertex_indices вказує, що кожна грань буде списком індексів вершин. Для завершення заголовка використовується рядок end_header.

Результат візуалізації хмари точок для простої геометричної фігури зображено на рис. 1.4.

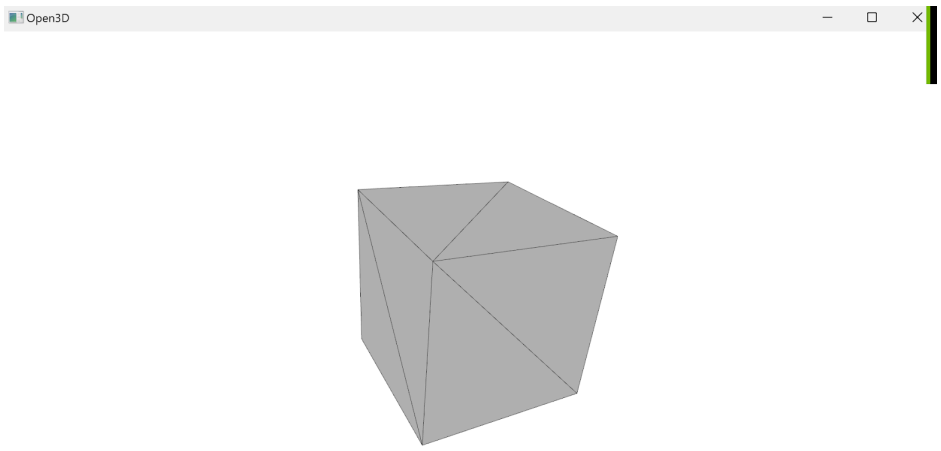

Рисунок 1.4 – Візуалізація PLY-файлу

1.4.3 Voxel representation

Воксель – тривимірний аналог двовимірного пікселя, який являє собою невеликий кубик. В результаті такі кубики заповнюють простір об'єкта як зображено на рис. 1.5.

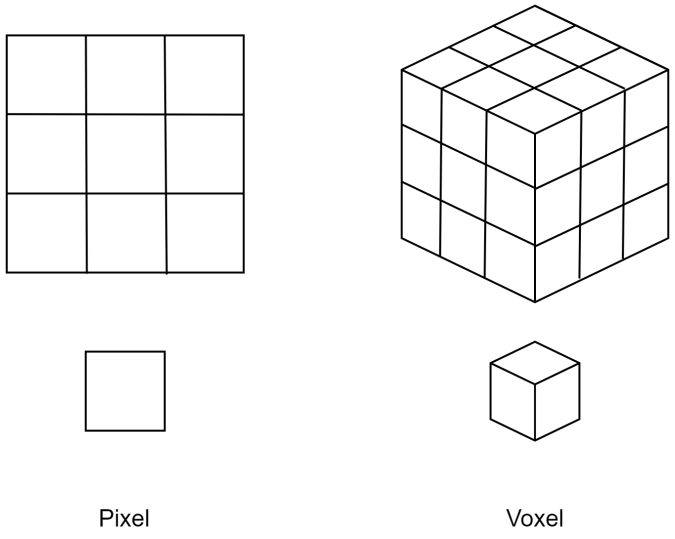

Рисунок 1.5 – Порівняння пікселя та вокселя

Основна перевага вокселя полягає в його здатності точно описувати внутрішню структуру об'єктів. Вокселі дають змогу моделювати не тільки поверхні, а й внутрішні об'єми, що робить цей метод особливо корисним для завдань, які потребують детального аналізу внутрішньої структури.

На відміну від хмар точок і полігональних сіток уявлення у вигляді вокселів упорядковане і є регулярним. Ця властивість схожа на пікселі в зображеннях і дає змогу використовувати згорткові фільтри в моделях глибокого навчання. Однак є і недоліки – воксельні моделі часто вимагають великого обсягу пам'яті для зберігання даних, особливо при високій роздільній здатності. Окрім цього рендеринг воксельних моделей може бути складнішим і повільнішим порівняно з рендерингом полігональних сіток, що обмежує застосування цього методу в реальному часі [18].

2 РОЗРОБЛЕНІ АЛГОРИТМИ ТА МЕТОДИ

2.1 Метод визначення патерну руху

2.1.1 Огляд вхідних даних

Розглянемо оброблене відео як послідовність кадрів відеоряду (рис. 2.1).

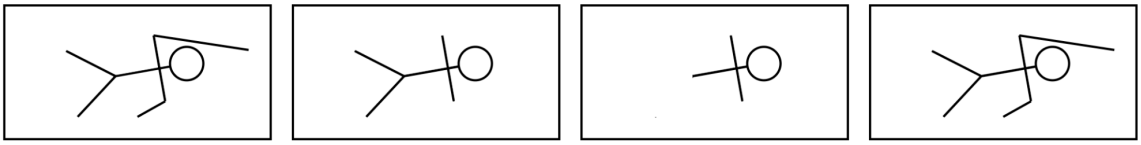

Рисунок 2.1 – Послідовність кадрів відеоряду F

Позначимо

$$F \in \{F_1, F_2, \ldots, F_n\}, \tag{2.1}$$

де F_i – кадр відеоряду в певний момент часу.

Кожен F_i – кадр відеоряду (формула 2.2) може бути представлений трійкою об'єктів,

$$F_i \in \{k_i, s_i, d_i\} \tag{2.2}$$

та, в свою чергу, містить інформацію про:

1) Ключові точки тіла людини k_i (формула 2.3), при чому індекси цієї послідовності відповідають номерам зображених на рис. 2.2;

$$k_i \in \{k_{i,1}, k_{i,2}, \ldots, k_{i,n} \mid n = 16\} \quad (2.3)$$

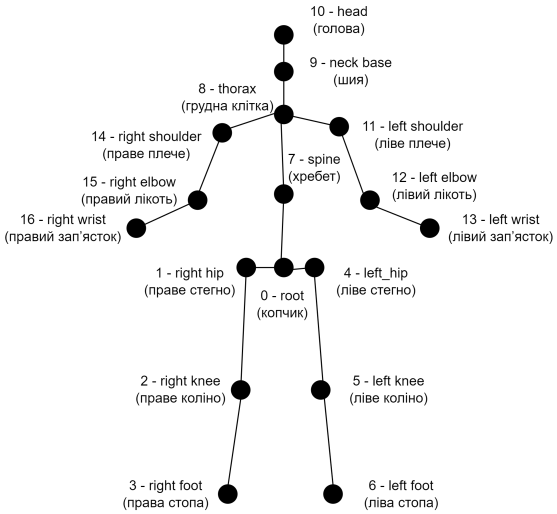

Рисунок 2.2 – Ключові точки тіла людини

2) Коефіцієнти точності s_i (формула 2.4), з якою визначено кожну ключову точку k_i;

$$s \in \{s_1, s_2, \ldots, s_n \mid n = 16\} \quad (2.4)$$

3) Довжину тіла d_i (формула 2.5), що являє собою евклідову відстань між середніми точками $a_{i,1}$ та $a_{i,2}$ (формули 2.6 - 2.7).

$$d_i = \left\| a_{i,2} - a_{i,1} \right\|, \quad (2.5)$$

$$a_{i,1} = \frac{k_{i,13} + k_{i,16}}{2}, \quad (2.6)$$

$$a_{i,2} = \frac{k_{i,3} + k_{i,6}}{2}. \quad (2.7)$$

Приклад обчислення евклідової відстані наведено на рисунку 2.3.

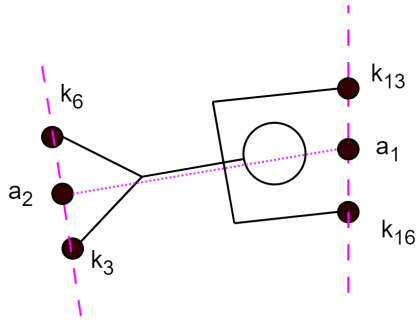

Рисунок 2.3 – Евклідових відстань між середніми точками

2.1.2 Визначення повторних послідовностей

Якщо побудувати графік залежності d_i (формула 2.5) від кадру відеоряду F_i (рис. 2.4), можна побачити періодичність у зростанні та спаданні величин d_i як зображено на рис. 2.5. Ця періодичність відповідає повторенню одного й того ж руху. Причому локальна точка максимуму у кожному з періодів є моментом ковзання плавця – тобто тіло повністю витягнуте в пряму лінію. Саме ці точки і будуть відокремлювати повторні рухи.

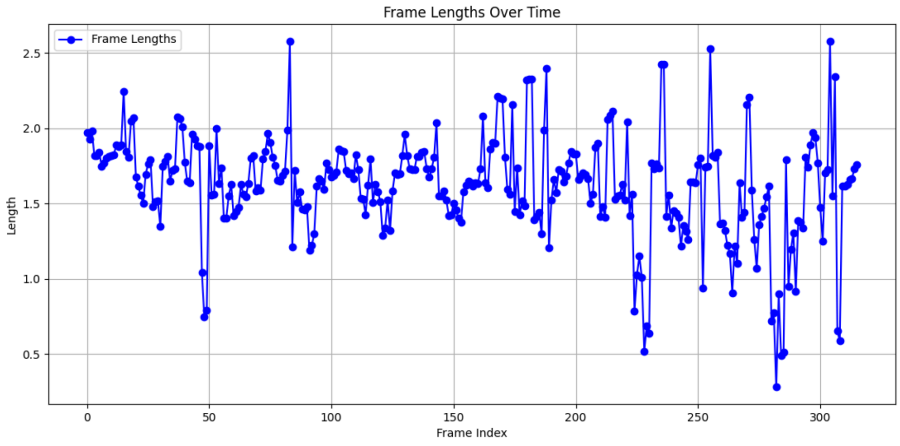

Рисунок 2.4 – Графік залежності d_i від F_i

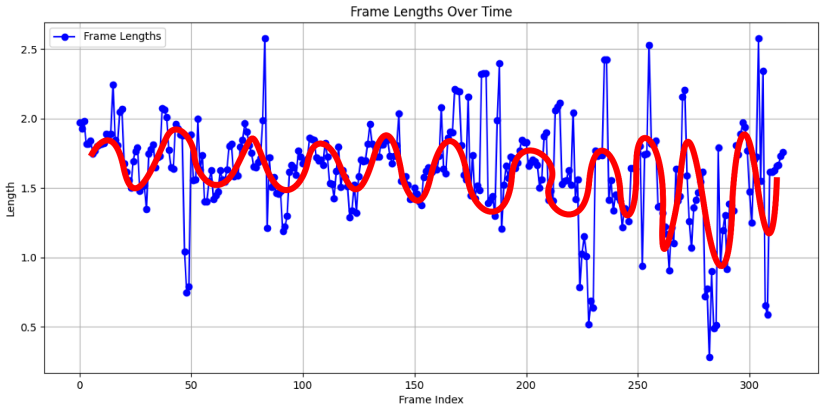

Рисунок 2.5 – Періодична зміна величини d_i

Отже можна визначити множину індексів локальних максимумів як:

$$b \in \{b_j \mid b_j = i, d_{i-1} < d_i \text{ та } d_i > d_{i+1}, i = 1, 2, \ldots, n, j \in N\}, \qquad (2.8)$$

де n – загальна кількість кадрів відеоряду.

Слід зауважити, що в реальності, після отримання даних за допомогою комп'ютерного зору точність розташування 3D точок може бути дуже низькою, що спотворює знайдені значення d. Тому спробуємо відфільтрувати їх виставивши поріг на мінімальне значення s (рис. 2.6).

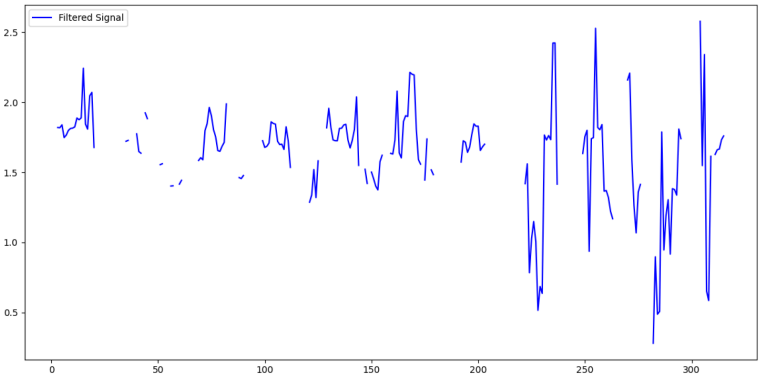

Рисунок 2.6 – Відфільтровані значення d_i за мінімальним значенням s

Так як для визначення локального максимуму функція повинна бути безперервна, то екстраполюємо та інтерполюємо дані для заповнення пропусків (рис. 2.7).

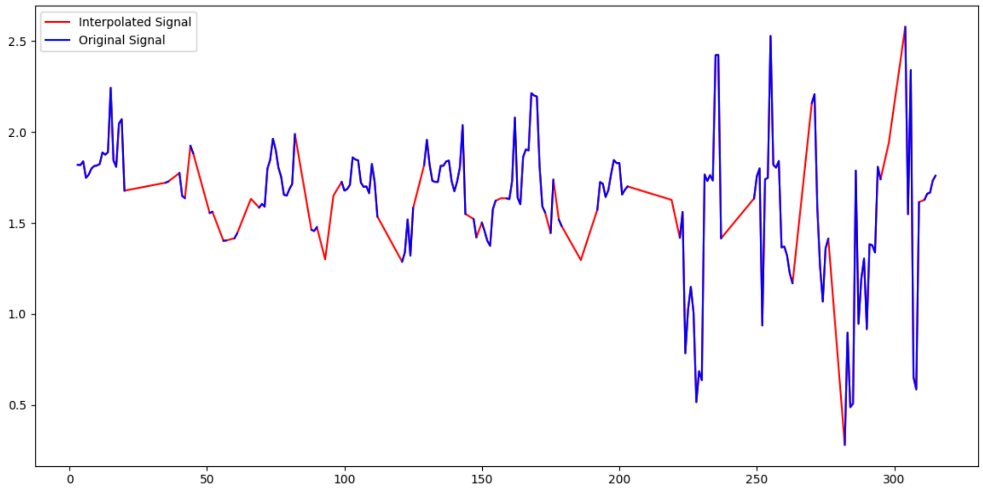

Рисунок 2.7 – Інтерпольовані значення d_i

І наостанок, використаємо гауссов фільтр для згладження значних відхилень (рис.2.8).

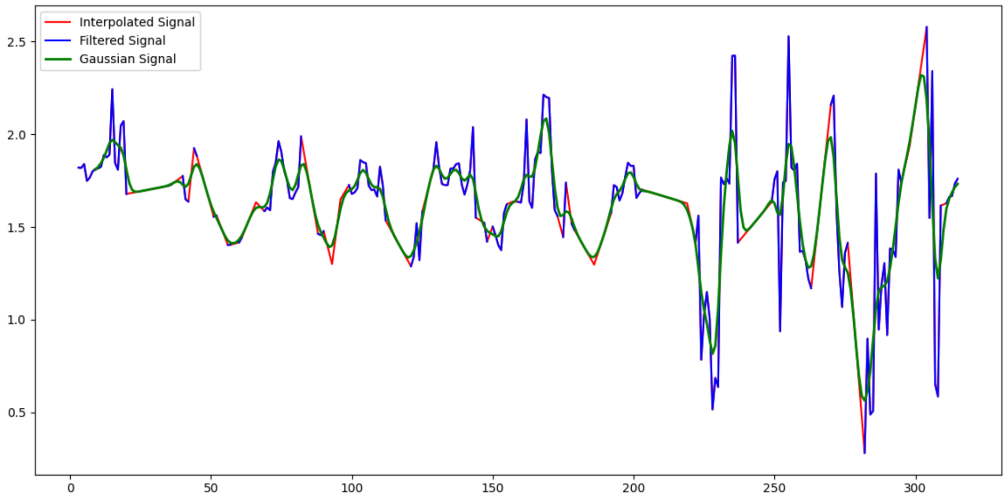

Рисунок 2.8 – Використання гауссового фільтру

В результаті стає можливим визначити програмно граничні точки для кожного циклу руху, як наведено в додатку Г.

На наступному етапі розіб'ємо послідовність F на серії рухів S, де граничними елементами будуть кадри під індексами з множини b як зображено на рис. 2.9:

$$S \in \{S_i | i = 1, 2, \ldots, n\}, \tag{2.9}$$

де n – кількість індексів елементів локальних максимумів

$$S_i \in \{F_{b_j}, F_{b_j+1}, \ldots, F_{b_{j+1}}\} \tag{2.10}$$

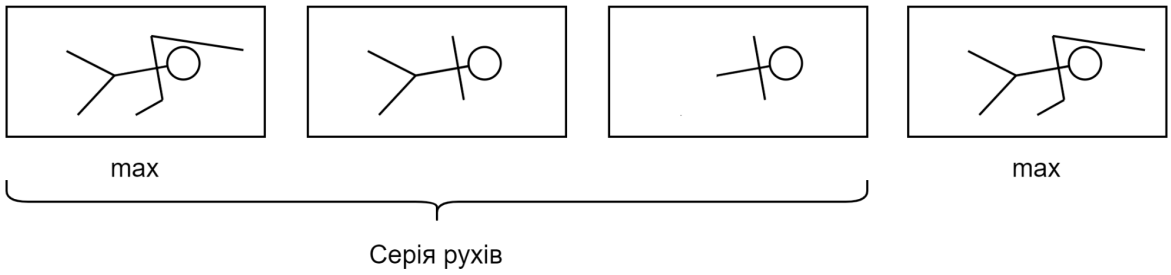

Рисунок 2.9 – Розбиття послідовності на серії рухів

2.1.3 Об'єднання повторень

Не зважаючи на те, що патерн руху постійно повторюється, швидкість виконання може бути різною, тому цілком ймовірно, що і кількість фреймів у кожній серії буде теж різною. З цієї причини необхідно узгодити серії за кількістю фреймів.

Для цього оберемо середнє значення кількості кадрів з кожної серії рухів:

$$S_a = \frac{\sum_{j=2}^{n}(b_j - b_{j-1})}{n-1}, \tag{2.11}$$

де n – кількість граничних індексів.

А далі за допомогою рівномірного розподілення видалимо або додамо фрейми для кожної серії таким чином, щоб отримати S_a фреймів в кожній серії рухів.

Таке узгодження надає можливість розглядати серії рухів як матрицю

$$S = \begin{pmatrix} S_{1,1} & S_{1,2} & \dots & S_{1,m} \\ S_{2,1} & S_{2,2} & \dots & S_{2,m} \\ \dots & \dots & \dots & \dots \\ S_{n,1} & S_{n,2} & \dots & S_{n,m} \end{pmatrix}, \tag{2.12}$$

де $S_{i,j}$ – j-ий фрейм для і-ої серії.

Тепер необхідно об’єднати фрейми в єдину серію.

Щоб отримати результуючий j-ий фрейм складемо всі j-ті фрейми з кожної серії

$$R_j = \begin{pmatrix} S_{1,j} \\ S_{2,j} \\ \dots \\ S_{n,j} \end{pmatrix} \tag{2.13}$$

Із формули 2.2 і 2.3 отримуємо

$$R_j = \begin{pmatrix} k_{1,j,1} & k_{1,j,2} & \dots & k_{1,j,p} \\ k_{2,j,1} & k_{2,j,2} & \dots & k_{2,j,p} \\ \dots & \dots & \dots & \dots \\ k_{n,j,1} & k_{n,j,2} & \dots & k_{n,j,p} \end{pmatrix} \tag{2.14}$$

Ну і для отримання і-ої ключової точки результуючого фрейму скористаємося формулою

$$k_i = \frac{\sum_{p=1}^{n} k_{p,j,i}}{n-1} \quad (2.15)$$

Результуюча серія являє собою набір фреймів для одного гребка або одного циклічного руху.

2.1.4 Оцінка методу

За рахунок таких перетворень вдалось визначити патерн руху плавця, фрагмент якого зображено на рис. 2.10 та стабілізувати визначені ключові точки до результату подібного зображенню на рис. 2.11.

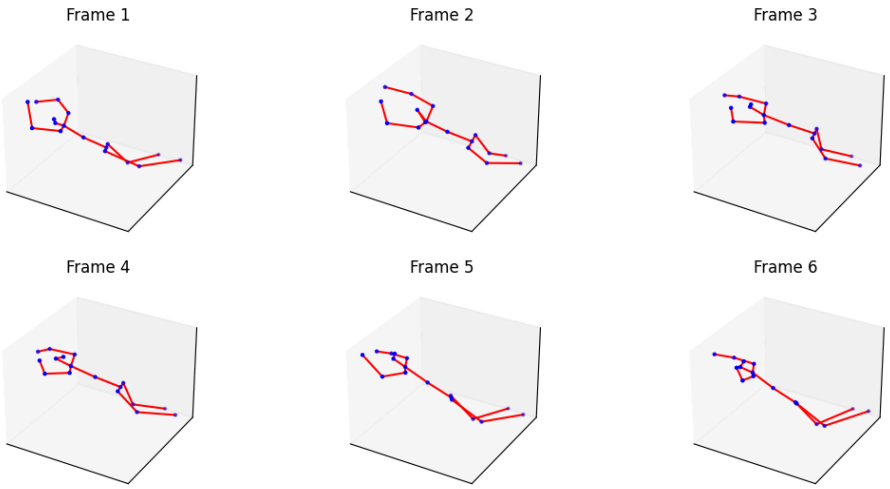

Рисунок 2.10 – Фрагмент отриманої послідовності кадрів патерну руху

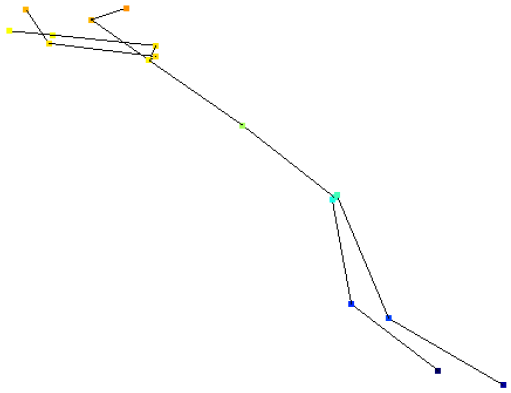

Рисунок 2.11 – Ключові точки одного з фреймів результуючої серії

2.2 Метод порівняння двох патернів руху

В якості основного алгоритму для порівняння двох патернів руху обрано PCA, проте при використанні вже існуючої реалізації із бібліотеки sklearn.decomposition [19] не вдалося отримати очікуваних результатів. Тому запропоновано використати лише деяку частину цього алгоритму, щоб визначити вклад кожної ключової точки людини у весь процес руху.

Для того, щоб ключові точки тіла могли розглядатися як головні компоненти руху, необхідно визначені координати інтерпретувати як вектори. Для цього визначимо довжину вектора від точки початку координат до кожної ключової точки тіла людини (див. рис. 2.12)

$$l_{k_i} = \left|\overline{k_i}\right|, \tag{2.16}$$

де i – індекс ключової точки.

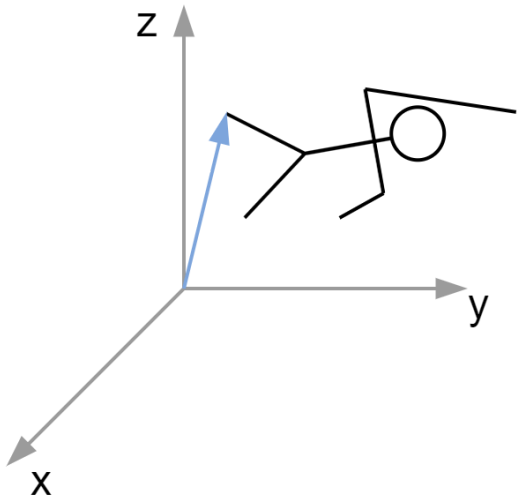

Рисунок 2.12 – Вектор ключової точки

Розглянемо серію фреймів, де кожен фрейм містить набір довжин векторів для кожної ключової точки (формули 2.17 - 2.19).

$$F_j^l = \{l_{k_i} \mid i = \overline{0,16}\}, \tag{2.17}$$

де j – індекс фрейму.

$$S^l = \{F_j^l \mid j \in N\}, \tag{2.18}$$

$$S^l = \begin{pmatrix} l_{1,k_0} & l_{1,k_1} & \dots & l_{1,k_{16}} \\ l_{2,k_0} & l_{2,k_1} & \dots & l_{2,k_{16}} \\ \dots & \dots & \dots & \dots \\ l_{n,k_0} & l_{n,k_1} & \dots & l_{n,k_{16}} \end{pmatrix}. \tag{2.19}$$

Транспонуємо матрицю S^l, щоб отримати довжини певної ключової точки в одному рядку. І знайдемо дисперсію для кожного рядка або ж ключової точки

$$D_i = \frac{\sum_{j=1}^{n} (l_j - \frac{\sum_{p=1}^{n} l_j}{n})^2}{n}, \quad (2.20)$$

де i – індекс ключової точки

n – кількість фреймів.

Значення дисперсії буде вказувати з якою інтенсивністю змінюється відстань до ключової точки. Проте для кращого розуміння інтенсивності при порівнянні двох патернів рухів слід зробити це значення відносним. Для цього порахуємо суму дисперсій і знайдемо відсоток вкладу для кожної ключової точки

$$c_i = \frac{D_i}{\sum_{j=1}^{n} D_j} \quad (2.21)$$

Результатом даного метода буде відсотковий вклад кожної ключової точки в порівнянні з іншими.

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

В даному розділі розглядається повних шлях по розробці системи.

Спочатку виконується налаштування середовища з можливістю працювати з усіма запропонованими технологіями на комп’ютері під операційною системою Windows 11 та графічною відеокартою NVIDIA.

Далі демонструються можливості визначення ключових точок тіла людини з використанням фреймворку MMPose. Визначаються переваги і недоліки демоваріантів роботи з цим фреймворкам. А також зазначаються шляхи покращення запропонованих прикладів.

Після цього розглядається реалізація запропонованих методів для визначення патерну руху та порівняння декількох патернів.

І, врешті-решт, демонструється графічний застосунок з інтерфейсом для взаємодії користувача з системою.

3.1 Налаштування середовища

З метою кращого контролю системи вирішено її контейнеризувати за допомогою Docker.

Для того щоб Docker мав доступ до відеокарти необхідно встановити nvidia-docker – це образ, який містить драйвера NVIDIA CUDA Toolkit (див. лістинг 3.1 та рис. 3.1).

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/libnvidia-container/gpgkey |
sudo gpg --dearmor -o
/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg
curl -s -L
https://nvidia.github.io/libnvidia-container/$distribution/libnvidi
a-container.list | \
sed 's#deb https://#deb
[signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg
] https://#g' | \
sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
sudo apt-get update
sudo apt-get install -y nvidia-container-toolkit
sudo nvidia-ctk runtime configure --runtime=docker
sudo systemctl restart docker
```

Лістинг 3.1 – Встановлення NVIDIA Container Toolkit

```
max@OgYrKo:~$ sudo docker run --rm --gpus all nvidia/cuda:11.0.3-base-ubuntu20.04 nvidia-smi
Unable to find image 'nvidia/cuda:11.0.3-base-ubuntu20.04' locally
11.0.3-base-ubuntu20.04: Pulling from nvidia/cuda
96d54c3075c9: Pull complete
59f6381879f6: Pull complete
655ed0df26cf: Pull complete
848b95ad96b5: Pull complete
e43c2058e496: Pull complete
Digest: sha256:c8269d6967e10940c368ea24fb8086cb21471cb8fefc66861d72f74f0c67e904
Status: Downloaded newer image for nvidia/cuda:11.0.3-base-ubuntu20.04
Sat Jun  8 18:01:53 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 555.52.01              Driver Version: 555.99         CUDA Version: 12.5     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 4060 ...    On  |   00000000:01:00.0 Off |                  N/A |
| N/A   45C    P8             1W /   50W |     111MiB /   8188MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

Рисунок 3.1 – З’єднання Docker та GPU

Для роботи програми створено Docker-образ наведений у додатку А. В образі налаштовується середа для роботи з бібліотекою mmpose та фреймворком PyQt5.

Для сборки образу виконується команда наведена в лістингу 3.2. Для запуску створеного контейнеру використовується команда наведена в лістингу 3.3. Атрибут -v вказує точку монтування контейнеру з робочим комп'ютером. Атрибут -e необхідний при використанні WSL, він налаштовує сервер візуалізації, який знаходиться за ip-адресою WSL адаптеру. В рамках даної роботи $DISPLAY є 172.26.32.1:0.0.

```
docker build -t mmpose .
```

Лістинг 3.2 – Сборка Docker-образу

```
docker run --gpus all -it --rm -v /home/max/code:/workspace/code -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix mmpose
```

Лістинг 3.3 – Запуск Docker-контейнеру

3.2 Визначення розташування ключових точок тіла людини у трьохвимірному просторі

Визначення координат ключових точок тіла людини у трьохвимірному просторі відбувається за допомогою фреймворку MMPose. В налаштованому Docker-середовищі окрім необхідних бібліотек також містяться приклади застосування цього фреймворку.

Одним з таких прикладів є python-файл body3d_pose_lifter_demo.py, який дає змогу одразу побачити як працює MMPose при визначенні ключових точок з відео послідовності.

При запуску прикладу в якості одного з аргументу надано відео файл з переміщенням плавця у воді стилем брас. В результаті отримано нову відео послідовність, яка містить візуальне накладання знайдених ключових точок на вхідну послідовність та трьох просторове відображення (рис. 3.2).

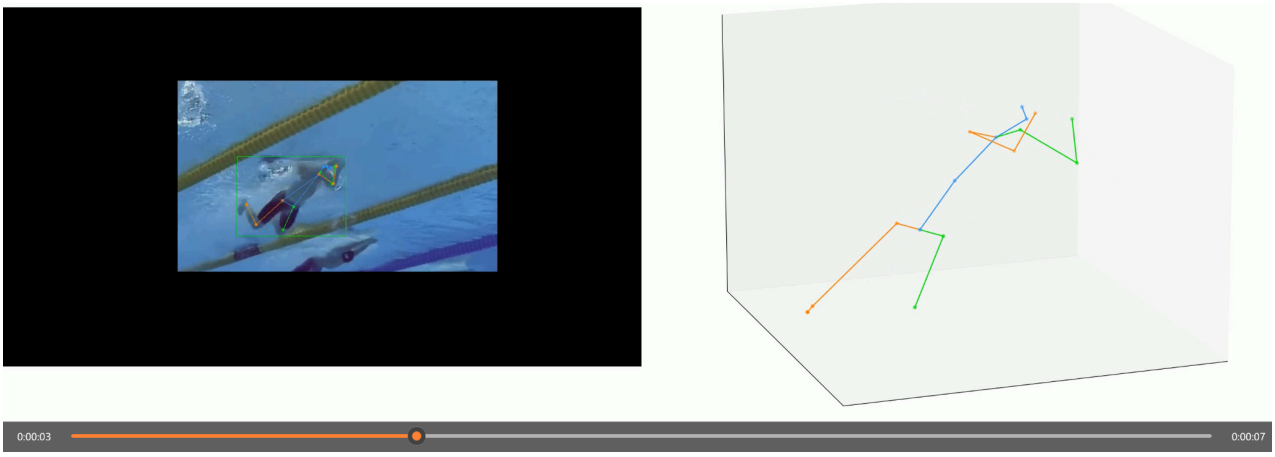

Рисунок 3.2 – Візуалізація в 3D просторі поєднаних між собою ключових точок тіла плавця

Проте цей фреймворк не завжди видає ідеальний результат і можна отримати дані не сумісні з реальністю, як зображено на рис. 3.3. Хоча й вдалося визначити тіло людини і деякі його частини, проте багато ключових точок втрачено, незважаючи на те, що людське око може ідентифікувати відповідні частини тіла.

Рисунок 3.3 – Візуалізація хибного передбачення ключових точок

Беручи за основу той факт, що спортсмен використовує певний патерн руху для переміщення у воді розроблено метод визначення цього патерну (див. підрозділ 2.2), який усуває проблему хибних даних для певних кадрів відеоряду.

Щоб інтегрувати MMPose у власну систему розроблено клас Body3D (додаток В). В його основі лежить демо варіант використання фреймворку, який адаптовано під вимоги визначення ключових точок тіла з будь-якого відео. А саме: перебудовано відповідно об’єктно-орієнтованій структурі та видалено зайві компоненти.

За рахунок таких перетворень стало можливим використання багатьох компонентів окремо один від одного за межами демо-файлу. Наприклад, отримати ключові точки тіла за допомогою методу get_predictions, або такі самі прогнози, але з додатковою інформацією про взаємне розташування точок тіла за допомогою методу get_predictions_with_metadata.

Також оптимізовано швидкість виконання програмного застосунку. Одним з найдовших процесів при визначенні ключових точок є ініціалізація конфігураційних моделей, на базі яких потім визначаються положення тіла в просторі. Для оптимізації часу виконання достатньо ініціалізувати конфігурацію лише 1 раз, але в демо варіанті такої можливості не було.

3.3 Визначення патерну руху

Для реалізації методу визначення патерну руху, наведеного в підрозділі 2.2 створено класи MovementPatternBuilder, VideoSequence, Sequence, Frame та Keypoint. Їх взаємовідношення зображено на рис. 3.4 за допомогою UML-діаграми.

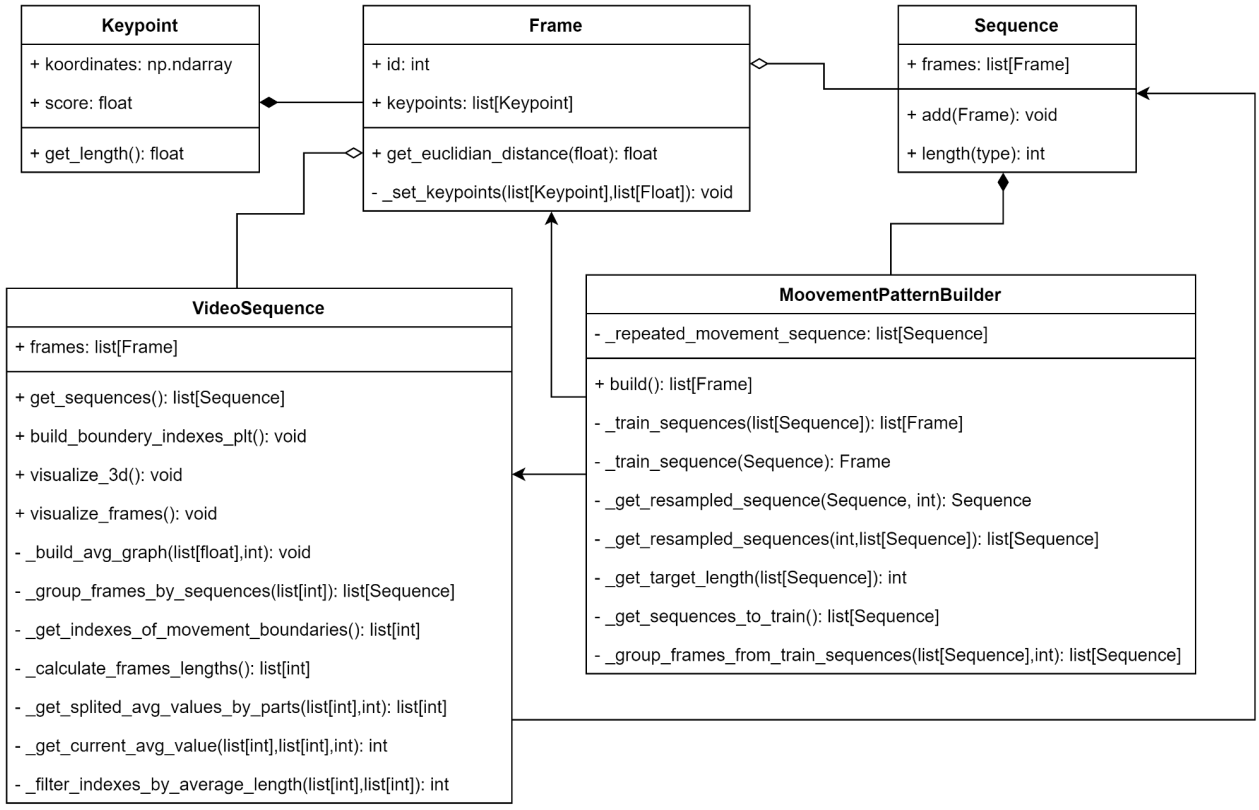

Рисунок 3.4 – UML-діаграма класів, застосованих для визначення патерну руху

Клас Keypoint відіграє роль ключової точки тіла людини. Цей клас містить абсолютні координати точки у тривимірному просторі та точність, з якою цю точку визначено. Окрім цього в цьому класі є метод get_length, який необхідний для визначення довжини вектора від початку системи координат до точки.

Наступний клас – Frame. Він містить в собі інформацію про кадр відео послідовності і складається з порядкового індексу id та виявлених ключових точок тіла keypoints. Також цей клас має два методи get_euclidian_distance та _set_keypoints. Перший визначає довжину тіла за формулою 2.5, але лише в тому разі, коли точність розрахунку всіх необхідних ключових точок перевищує заданий поріг. А другий перетворює масив координат ключових точок у клас Keypoint.

Клас Sequence виступає у ролі допоміжного класу і є по своїй суті обгорткою до колекції. Його основна задача згрупувати набори кадрів в одному об’єкті.

Більш цікавим є клас VideoSequence. Як і попередній він теж об’єднує набір кадрів в єдиному об’єкті, проте колекція загалом формує послідовну відео модель. За допомогою таких методів як visualize_3d та visualize_frames цю послідовність можна візуалізувати (див. підрозділ 3.5).

Проте основна задача даного класу – це розбиття відео послідовності на набір фреймів циклічних рухів. Робиться це в методі get_sequences, в якому визначаються граничні фрейми, після чого послідовність відеокадрів розбивається на цих визначених фреймах.

Розглянемо більш детально код визначення граничних фреймів (див. Лістинг 3.4). Алгоритм цього методу відповідає опису наведеного в підрозділах 2.2.1 та 2.2.2. Спочатку за допомогою методу _calculate_frames_lengths визначаються довжини тіла на різних кадрах. Далі використовується функція interpolate з python-бібліотеки pandas з визначенням параметру limit_direction='both', що означає, що передана функція буде і екстраполюватись, і інтерполюватись одночасно [20]. Після цього за допомогою функції gaussian_filter1d з python-бібліотеки scipy [21] функція згладжується і в результаті можна більш якісно визначити локальні максимуми за допомогою функції find_peaks, знов таки, з бібліотеки scipy. Знайдені вершини ще раз

перевіряються в методі _filter_indexes_by_average_length. Це необхідно, так як функція find_peaks чутлива до локальних максимумів, які мають приблизно однакове значення. В такому разі воно буде знаходити обидві вершини, проте при визначенні граничних фреймів така поведінка недопустима, тому обирається середнє значення між вершинами, як зображено на рис. 3.5 зеленим хрестиком.

```
def _get_indexes_of_movement_boundaries(self):
        lengths = self._calculate_frames_lengths()
        interpolated_lengths =
pd.Series(lengths).interpolate(limit_direction='both').to_numpy()
        gaussian_filtered_lengths =
gaussian_filter1d(interpolated_lengths, sigma=1.5)
        indexes, _ = signal.find_peaks(gaussian_filtered_lengths,
prominence=0.1)
        return
self._filter_indexes_by_average_length(gaussian_filtered_lengths,
indexes)
```

Лістинг 3.4 – Метод визначення індексів граничних фреймів

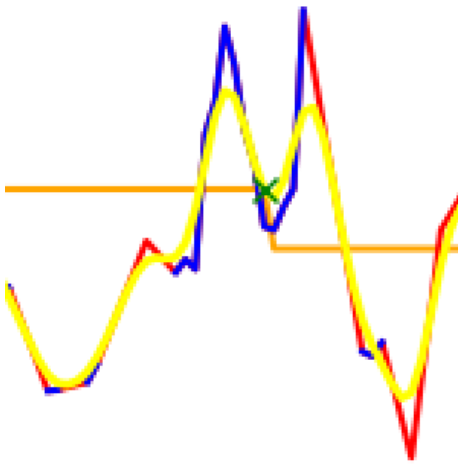

Рисунок 3.5 – Ідентифікація граничного фрейму між двома локальними максимумами, які знаходяться поруч

Останній клас – MovementPatternBuilder. В конструктор передається екземпляр VideoSequence, після чого одразу визначаються циклічні серії рухів. Основним методом класу є build (Лістинг 3.5), який повертає набір кадрів побудованого патерну руху.

```
    def build(self)->list:
        sequences_to_train = self._get_sequences_to_train()
        return self._train_sequences(sequences_to_train)
```

Лістинг 3.5 – Реалізація методу build

Метод _get_sequences_to_train реалізує логіку формул 2.11 та 2.12. І як результат повертає масив значень відповідно формулі 2.13. Формули 2.14 та 2.15 реалізуються вже у методі _train_sequences: для кожного отриманого масиву застосовується метод _train_sequence (Лістинг 3.6), де і визначається значення ключових точок для результуючого фрейму.

```
def _train_sequence(self,sequence:Sequence)->Frame:
        # Array for storing the average coordinates of each key
point
        averaged_keypoints = []
        for i in range(KEYPOINTS_COUNT):
            # Collect coordinates of the i-th key point from all
frames
            keypoint_coords = [frame.keypoints[i].koordinates for
frame in sequence.frames]
            keypoint_scores = [frame.keypoints[i].score for frame
in sequence.frames]
            # Convert the list of coordinates to numpy array for
convenience
            keypoint_coords = np.array(keypoint_coords)
```

Лістинг 3.6 – Реалізація методу _train_sequence

```
            # Averaging coordinates over all frames
            averaged_coords = np.mean(keypoint_coords, axis=0)
            averaged_score =
sum(keypoint_scores)/len(keypoint_scores)
            # Write the average result to a new Keypoint
            average_keypoint = Keypoint(averaged_coords,
score=averaged_score)
            # Add the key point to the list
            averaged_keypoints.append(average_keypoint)
        # Create a new frame based on the averaged key points
        new_frame = Frame(id=-1, keypoints=[kp.koordinates for kp
in averaged_keypoints], scores=[kp.score for kp in
averaged_keypoints])
        return new_frame
```

Лістинг 3.6, аркуш 2

Дану реалізацію методу визначення патерну руху можна покращити шляхом винесення деяких методів у нові класи, щоб кожен клас мав свою зону відповідальності і загальна структура застосунку краще відповідала об’єктно-орієнтованій структурі.

3.4 Порівняння патернів руху

Для порівняння патернів руху і візуалізації результатів створено набір функцій наведених в додатку Г. Ключова функція – це compare_sequences (Лістинг 3.7). Спочатку визначається довжина векторів до ключових точок тіла за допомогою функції extract_keypoint_lengths, яка повертає масив еквівалентний матриці з формули 2.19. Далі отриманий масив перебудовується, або ж матриця транспонується і за формулою 2.20 визначається вклад кожної ключової точки в функції calculate_variance. Після чого цей вклад перераховується у відсотковому відношенні відносно всіх вкладів (формула 2.21) у функції calculate_ration_of_sum.

```
def compare_sequences(sequence1: list, sequence2: list):
    lengths1 = extract_keypoint_lengths(sequence1)
    lengths2 = extract_keypoint_lengths(sequence2)

    reorganized_lengths1 = reorganize_lengths(lengths1)
    reorganized_lengths2 = reorganize_lengths(lengths2)


    # Comparing the contribution of each key point
    var1 = calculate_variance(reorganized_lengths1)
    var2 = calculate_variance(reorganized_lengths2)

    ratio1 = calculate_ration_of_sum(var1)
    ratio2 = calculate_ration_of_sum(var2)
    recommendations = []
    for i in range(len(var1)):
        recommendations.append((i, ratio1[i], ratio2[i]))
    return recommendations
```

Лістинг 3.7 – Реалізація функції порівняння патернів руху

Отриманий результат візуалізується у вигляді діаграми за допомогою функції plot_keypoint_contributions як на рис 3.6.

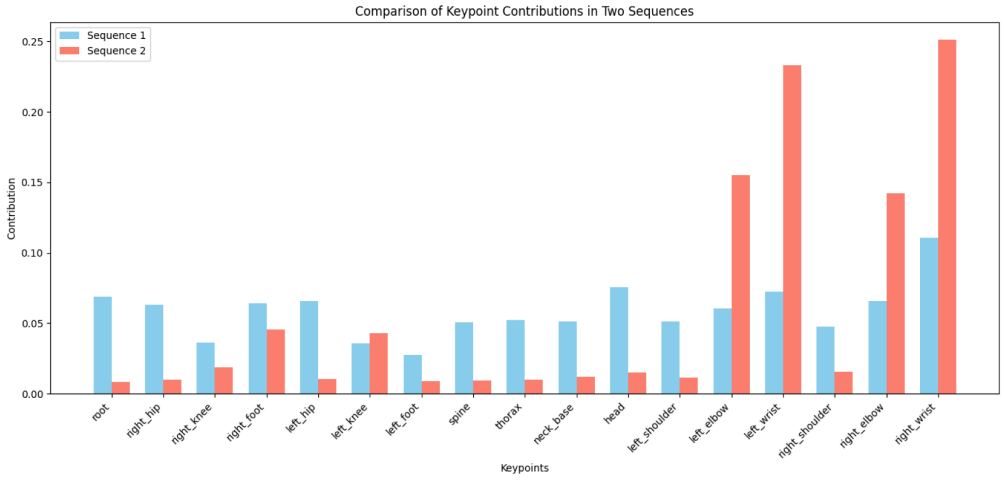

Рисунок 3.6 – Діаграма порівняння вкладу кожної ключової точки для двох патернів руху

3.5 Користувацький інтерфейс

Для взаємодії користувача з системою створено інтерфейс, який дозволяє візуалізовувати 3D моделі завантаженого руху та виявленого патерну руху для двох завантажених файлів. Окрім цього є можливість переглянути як саме виявляється патерн, за допомогою візуалізації у вигляді графіку змін довжин тіла протягом всієї відео послідовності. І звісно присутній графік порівняння вкладу кожної ключової точки для двох послідовностей.

При запуску застосунку викликається вікно, де необхідно завантажити два файли для порівняння (рис. 3.7). Без виконання цієї дії буде неможливе продовження роботи з програмою і кнопка «Continue» залишиться неактивною.

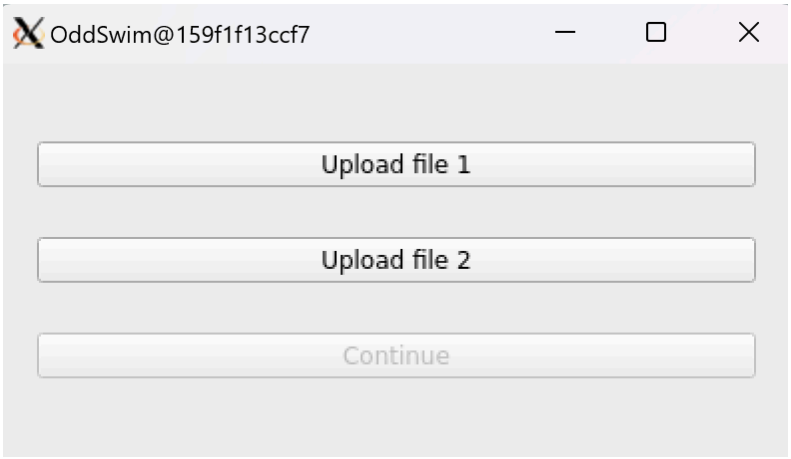

Рисунок 3.7 – Початкове вікно вибору файлів для порівняння

На вхід приймаються файли двох типів: mp4 та json. Завантажений json файл має відповідати структурі як наведено в додатку Б. Фактично це попередньо оброблений mp4 файл. В свою чергу, при завантаженні mp4 файлу застосунок обробить відео з використанням фреймворку MMPose (див. підрозділ 3.2) і сформує новий json-файл, який автоматично буде застосовуватись далі.

При переході на наступну сторінку відображається меню вибору можливих дій над відео послідовністю (рис. 3.8).

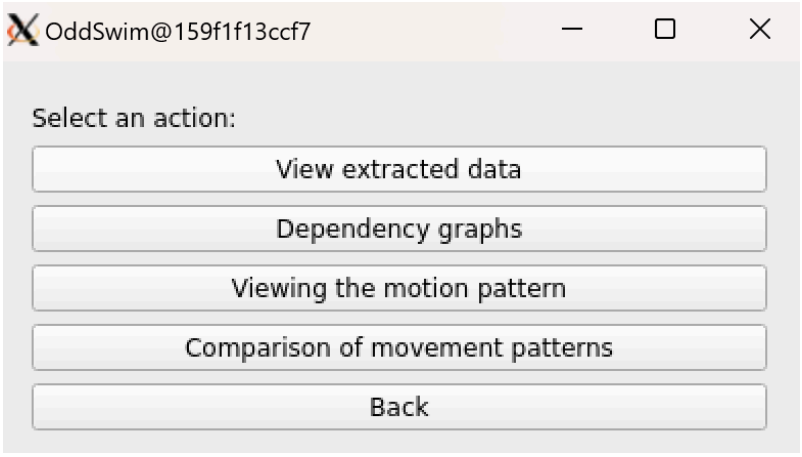

Рисунок 3.8 – Головне меню вибору дії над обробленою відео послідовністю

Розглянемо можливі дії більш детально.

При виборі пункту «View extracted data» з’являється ще одне меню, але на цей раз з можливістю вибору методу візуалізації для окремої моделі руху (див. рис. 3.9). Також є можливість зберегти оброблену відео послідовність у json файлі за допомогою кнопки «Save».

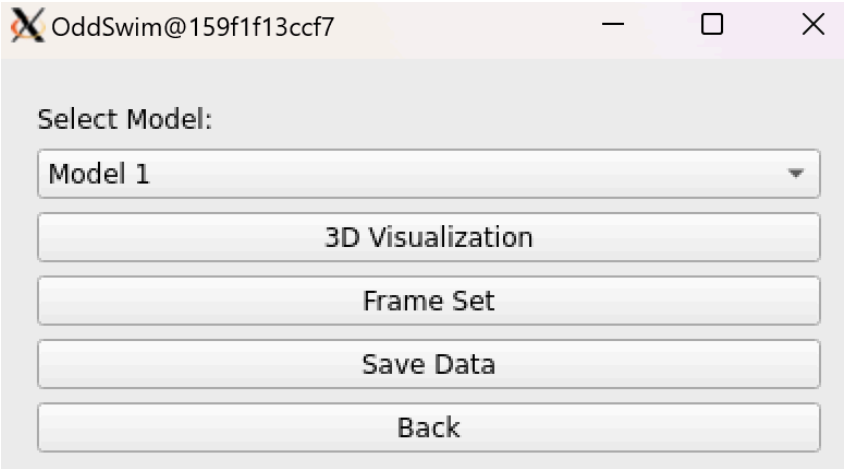

Рисунок 3.9 – Меню з вибором методу візуалізації моделі

При виборі «3D Visualization» з’являється додаткове вікно з візуалізацією моделі у трьохвимірному просторі (рис. 3.10). В такому режимі відображається лише 1 кадр з відео послідовності. Щоб переключити кадри достатньо використати клавіши зі стрілками вправо або вліво. Окрім цього, для більш детального аналізу отриманих даних є можливість змінити ракурс огляду моделі, а також відстань до неї.

Рисунок 3.10 – 3D візуалізація руху для окремого кадру обробленої відео послідовності

При виборі іншого методу візуалізації – «Frame Set» відображається серія кадрів з одного ракурсу (рис. 3.11). Такий підхід надає краще уявлення про зміни розташування ключових точок для кожного кадру.

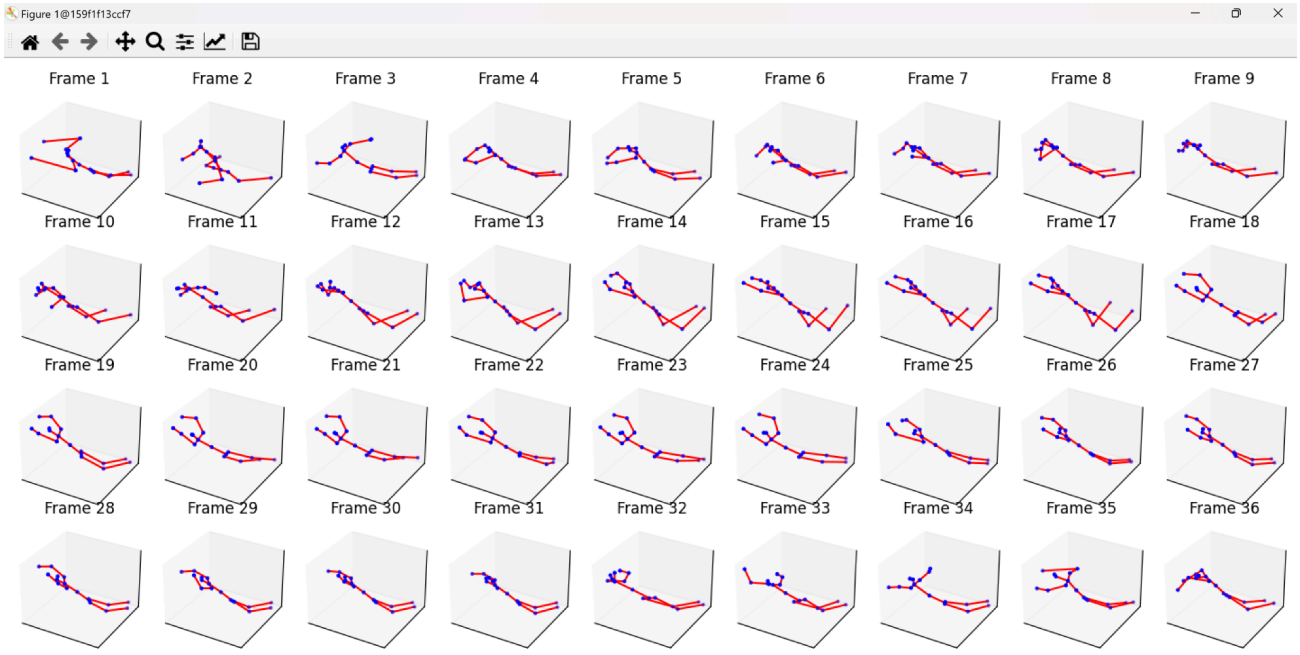

Рисунок 3.11 – Візуалізація набору кадрів обробленої відео послідовності

Для повернення до головного меню слід натиснути кнопку «Back».

Аналогічно дії «View extracted data» працює і дія «View the motion pattern». Єдина різниця в тому, що візуалізується не вся відео-послідовність, що подавалася на вході, а виділений патерн руху.

Для перевірки роботи виділення патерну руху присутня дія «Dependency Graphs». Після натискання цієї кнопки є можливість вибору моделі, з якою буде пов’язано наступні дії (рис. 3.12). Після обрання моделі візуалізується графік із функцією залежності довжини тіла від індексу кадру, графіки подальшої обробка цієї функції та маркеруються результуючі кадри, де визначено локальні максимуми функції (рис. 3.13).

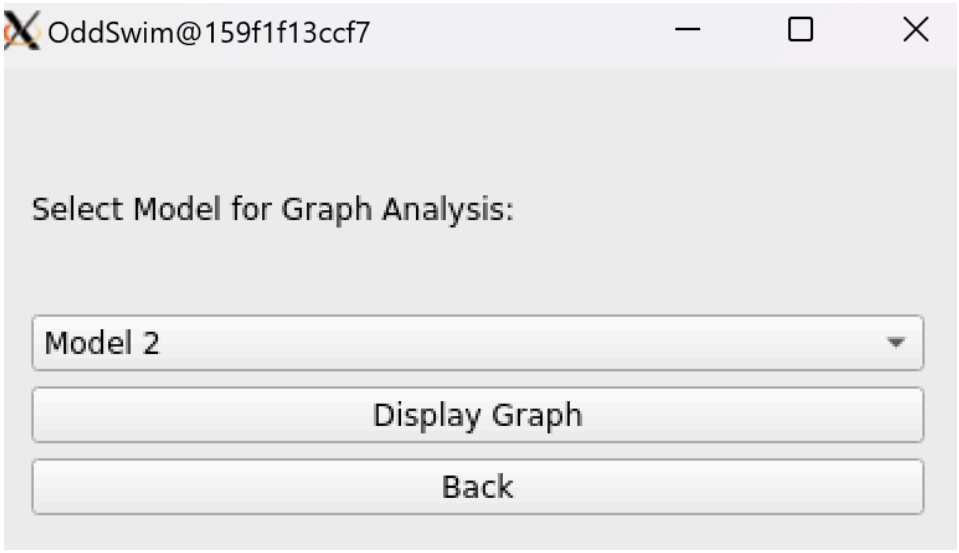

Рисунок 3.12 – Вибір моделі для візуалізації графіку залежності

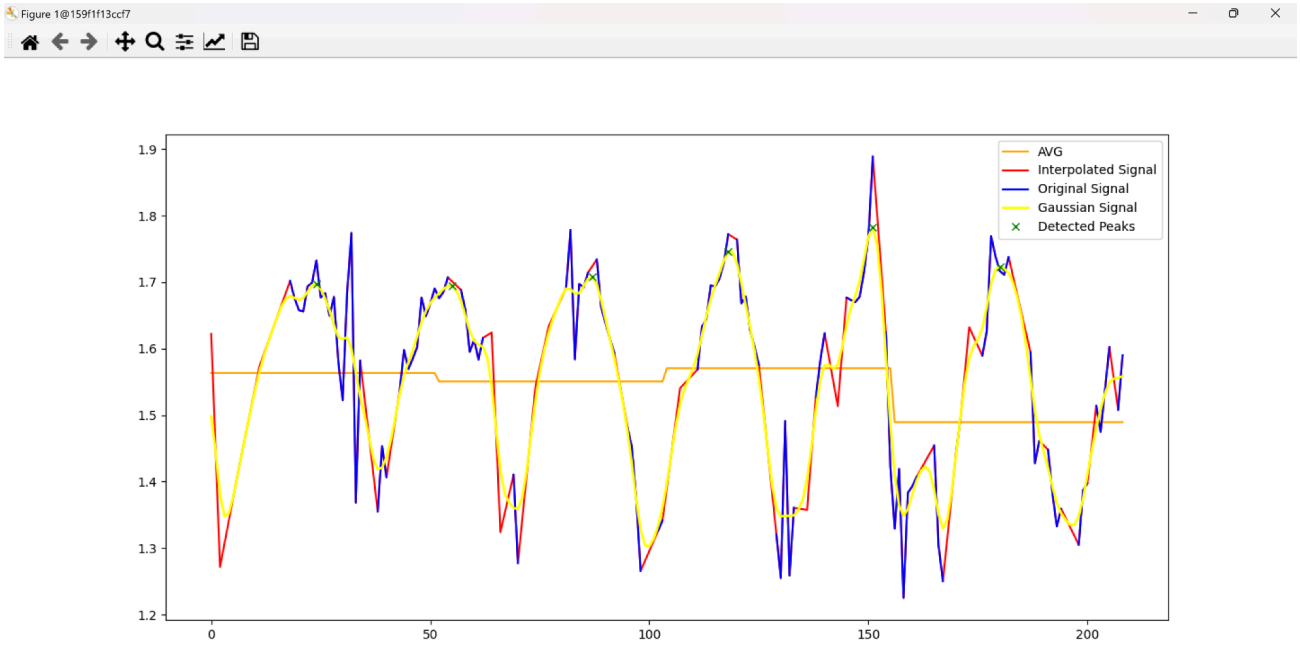

Рисунок 3.13 – Графіки функції залежності довжини тіла від індексу кадру, її подальша обробка та визначення локальних максимумів

І остання можлива дія – «Comparison of movement patterns», яка відповідає за візуалізацію діаграми порівняння вкладу кожної ключової точки для отриманих патернів руху з двох моделей (рис. 3.6).

4 ТЕСТУВАННЯ РОЗРОБЛЕНИХ ТЕХНОЛОГІЙ

В рамках даної роботи завантажуються два mp4 файли (рис. 4.1 та 4.2), де спортсмени роблять 10 та 5 циклічних рухів стилем брас відповідно. Даний стиль обрано через постійне перебування всіх частин тіла окрім голови під водою. Це означає, що при зйомці з підводного ракурсу всі ключові точки можна буде ідентифікувати та обробити для створення тривимірної моделі.

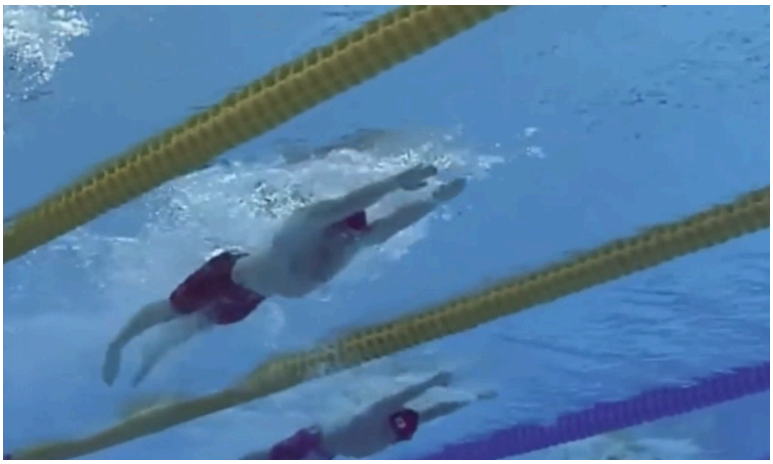

Рисунок 4.1 – Один з кадрів першої відео послідовності

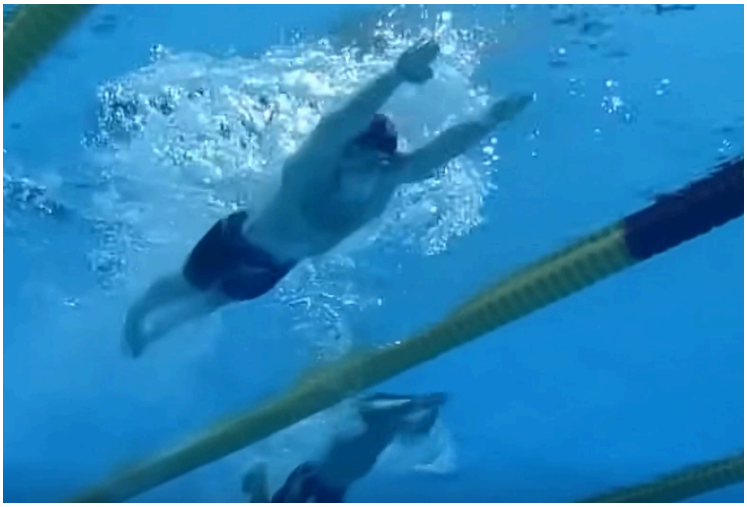

Рисунок 4.2 – Один з кадрів другої відео послідовності

Перший етап – виділення ключових точок тіла людини для кожного кадру відео послідовності. На цьому етапі отримано 3D моделі кожного кадру, для кожного відеофайлу. Фрагмент результуючих послідовностей кадрів зображено на рис. 4.3 та 4.4 для першої і другої послідовності відповідно.

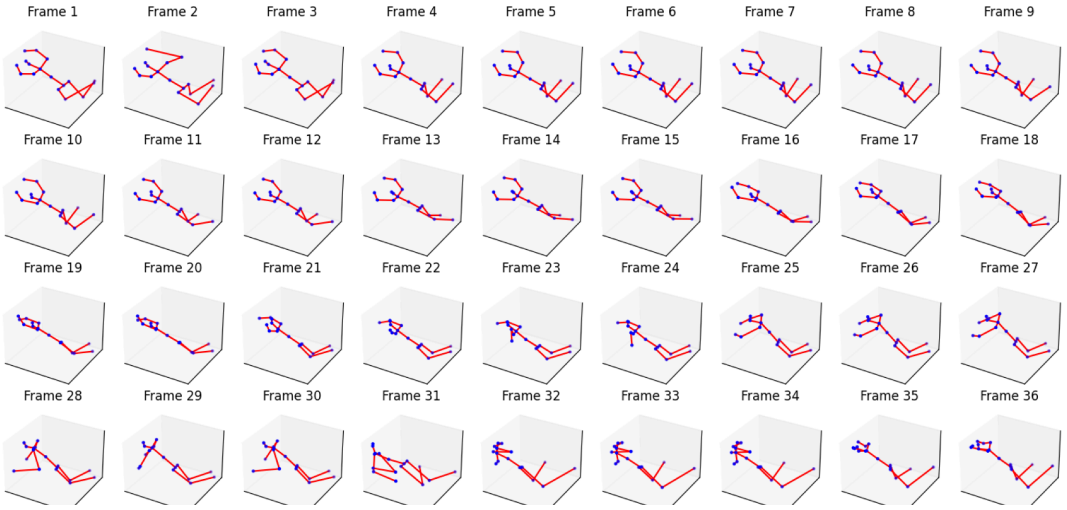

Рисунок 4.3 – Фрагмент отриманих 3D моделей кадрів для першої відео послідовності

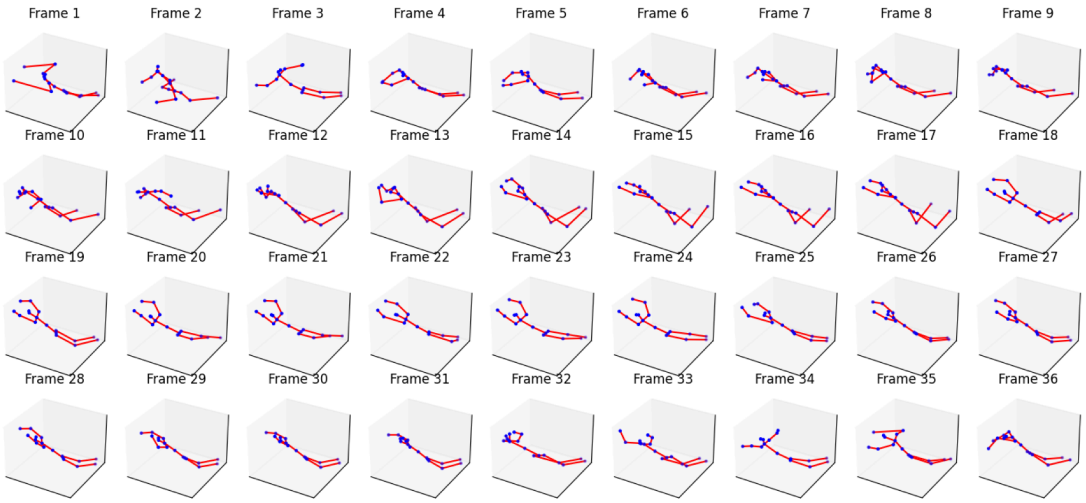

Рисунок 4.4 – Фрагмент отриманих 3D моделей кадрів для другої відео послідовності

Зазвичай на цьому етапі використовують спеціальні датчики, які прикріплюють до тіла. Такий підхід може побудувати ідеальну 3D модель. В дослідженні ж отримано приблизну модель. Як можна побачити існують фрейми, де результати несумісні з реальністю. Проте кількість таких кадрів менше 10%, тому вони не сильно впливають на остаточний результат.

В цілому модель змогла непогано ідентифікувати спортсмена і визначити його ключові точки тіла в трьохвимірному просторі. Не дивлячись на те, що відео знімалось з нижнього ракурсу, візуалізація фреймів відбувається з верхнього. При цьому більшість фреймів правильно візуалізує людину із рухами схожими на плавальні. Це вказує на вдале використання даного методу визначення ключових точок.

Наступний етап – виділення патерну руху або ж техніки спортсмена. В першу чергу візуалізуємо, як саме визначено початок і кінець кожного циклічного руху (рис. 4.5 та рис. 4.6). Як видно з наведених прикладів для першої моделі визначено 11 граничних кадрів, а для другої – 6, що відповідає початковим розрахункам кількості циклічних рухів.

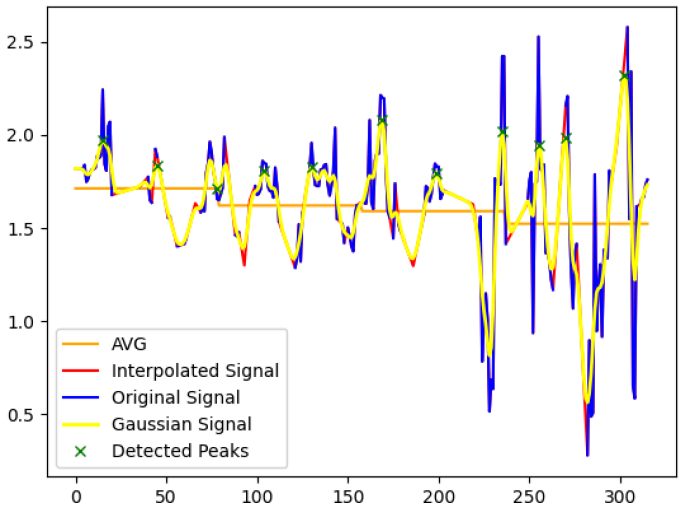

Рисунок 4.5 – Визначення граничних кадрів циклічних рухів для першої моделі

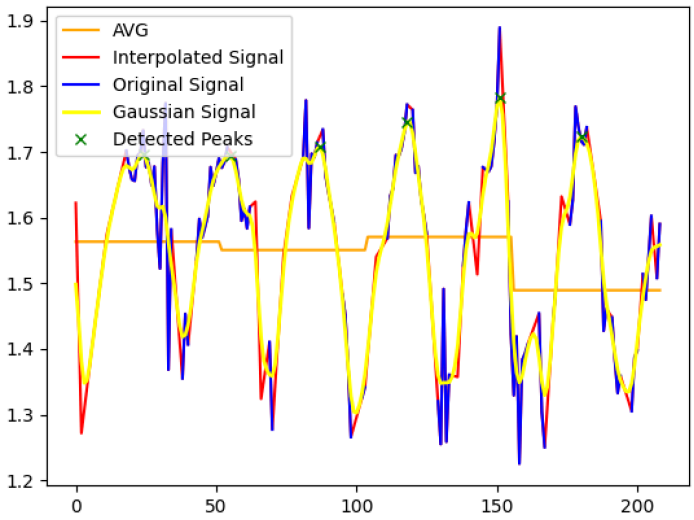

Рисунок 4.6 – Визначення граничних кадрів циклічних рухів для другої моделі

Після визначення циклічних рухів вони об’єднуються в єдиний патерн (рис. 4.7 та 4.8). Візуалізувавши техніку можна побачити зміни рухів від етапу ковзання, далі рухи руками та ногами, і, врешті-решт, повернення до ковзання. Проте якість отриманих даних для кожної моделі відрізняється. В першої моделі на деяких кадрах присутні шуми. Проте друга модель їх не містить, що вказує на ефективність методу визначення патерну руху.

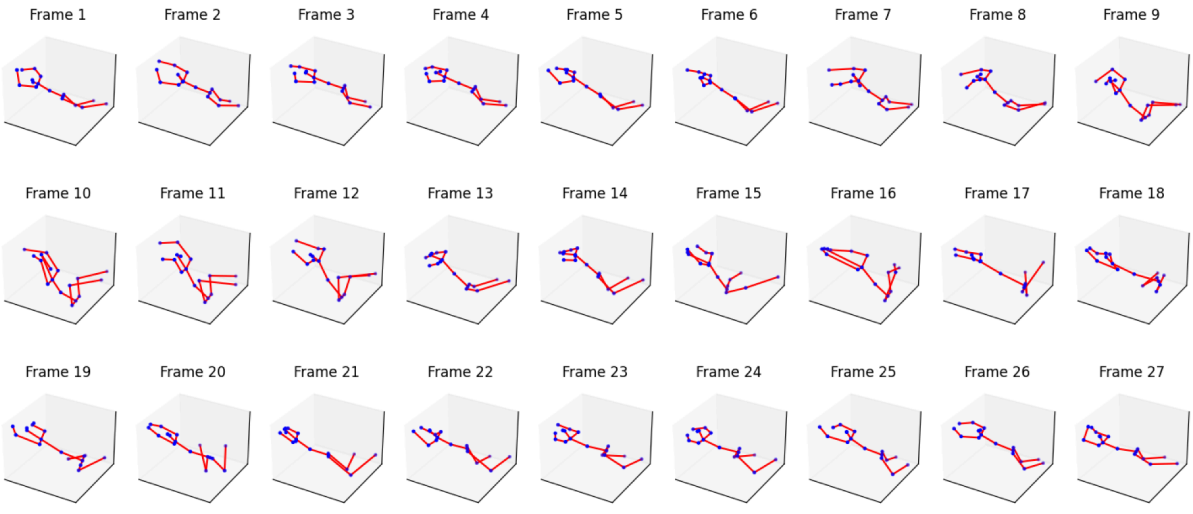

Рисунок 4.7 – Візуалізація патерну руху першої моделі

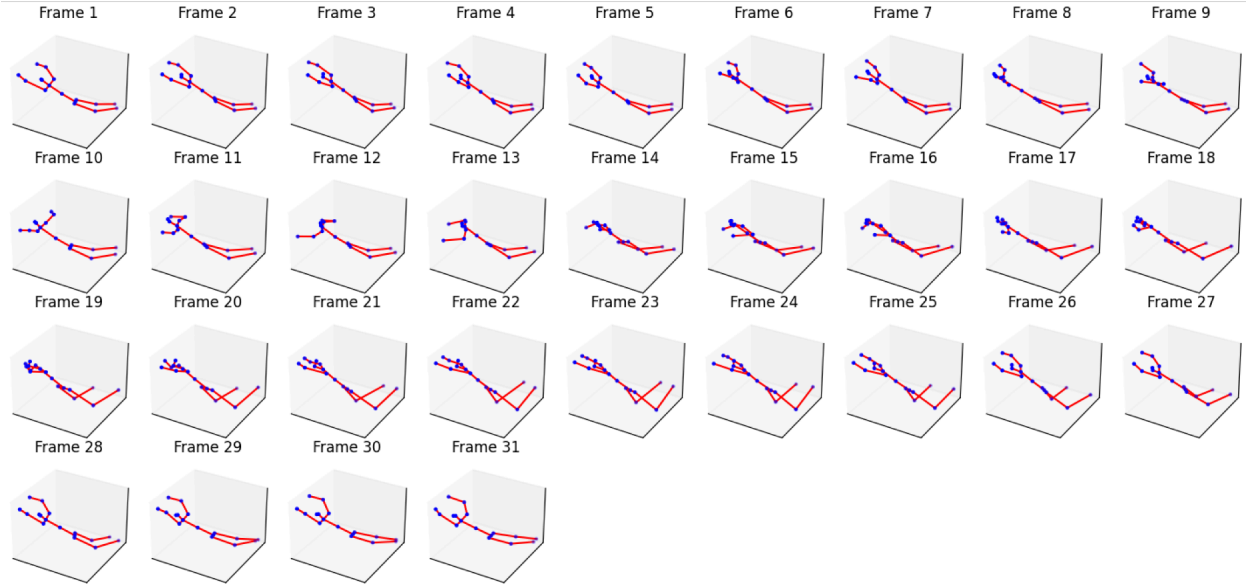

Рисунок 4.8 – Візуалізація патерну руху другої моделі

І останній етап – порівняння рухів. Для оцінки результатів візуалізуємо інформацію про вклад кожної ключової точки в двох патернах руху (рис. 4.9).

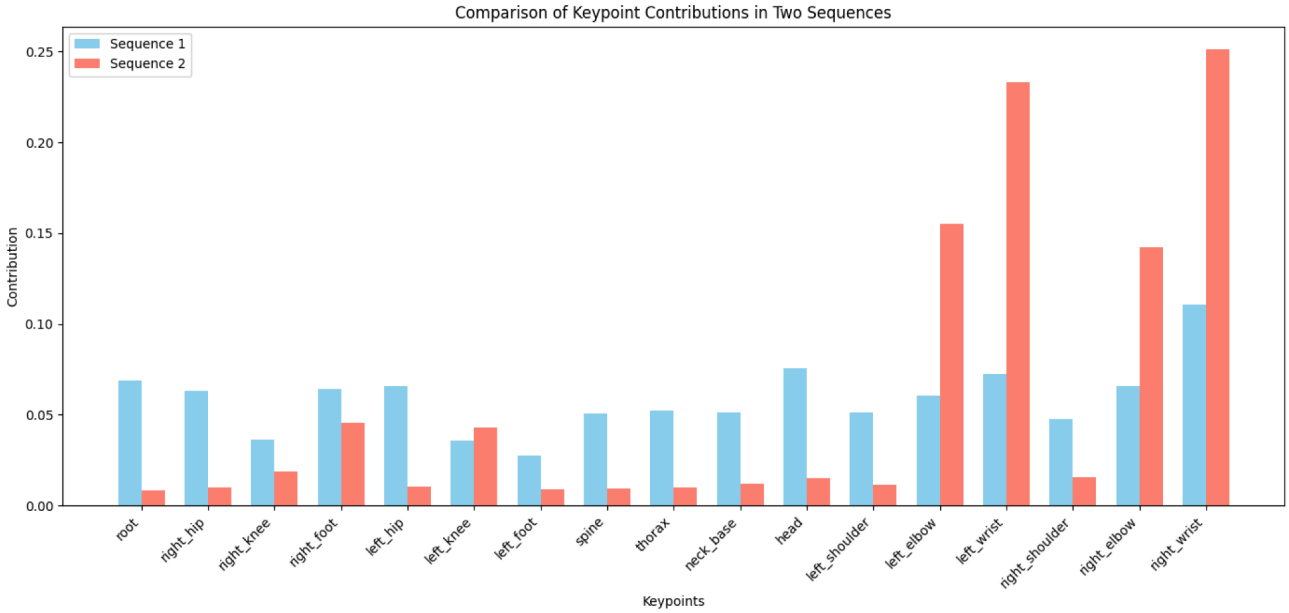

Рисунок 4.9 – Діаграма порівняння вкладу кожної ключової точки для двох патернів руху

Аналізуючи діаграму можна визначити, наскільки вклад певної ключової точки одного патерну руху відрізняється від іншого. Проведемо такий аналіз. Наприклад, другий плавець набагато активніше гребе своїми руками ніж перший, приділяючи менше увагу іншим частинам тіла, таких як ноги та тулуб. Перший спортсмен в свою чергу плаває більш збалансовано. Всі його частини тіла мають приблизно однаковий вклад.

Слід також зауважити, що вклад парних частин тіла, наприклад, як лівий та правий лікоть, мають приблизно однакові значення, що свідчить про правильну роботу методу.

ВИСНОВКИ

Під час виконання роботи визначено проблематику предметної області та сформовано задачі для її вирішення.

В результаті аналізу предметної області обрано методи та технології для виконання поставлених задач.

Результати дослідження, представлені в роботі, розвивають сучасні підходи до аналізу спортивної техніки за допомогою комп'ютерного зору та машинного навчання. Побудована інформаційна система, здатна визначати ключові точки тіла спортсмена у тривимірному просторі, виділяти патерни руху з відеоматеріалів і порівнювати їх з іншими зразками. Запропоновані методи дозволяють виділити об'єктивну характеристику плавання конкретного спортсмена без використання складного спеціалізованого обладнання, що робить систему доступною для широкого впровадження в тренувальний процес.

Інноваційність роботи полягає в інтеграції сучасних фреймворків, таких як MMPose, з власноруч розробленими алгоритмами обробки відео, що забезпечило підвищення точності визначення ключових точок тіла навіть у складних умовах, таких як зйомка у воді. Розроблена система використовує модульний підхід, який передбачає гнучкість у налаштуванні для різних завдань та можливість масштабування під інші види спорту чи форми рухів.

Основним досягненням є створення методології визначення патернів руху, яка враховує циклічність рухів спортсмена і дозволяє ідентифікувати ключові фази техніки. Це забезпечує не лише виявлення помилок у техніці, а й створення універсальних моделей для порівняння рухів, на основі яких можуть формуватися рекомендації для вдосконалення техніки.

Ефективність системи оцінювалася за кількома критеріями, серед яких точність визначення ключових точок, здатність виділяти циклічні рухи та

зручність візуалізації результатів. Проведене тестування показало, що запропоновані методи досягають візуально прийнятних показників точності навіть у складних умовах, таких як зйомка під водою. Система також демонструє здатність адаптуватися до різних відеоматеріалів без попереднього калібрування камер, що значно підвищує її практичність. Окрім цього система забезпечує суб’єктивно точне визначення патернів рухів із мінімальними помилками.

Робота має перспективи подальшого розвитку в напрямках підвищення точності визначення 3D координат ключових точок, інтеграції в режим реального часу для використання в тренувальних комплексах, а також адаптації методів під інші види спорту або інші сфери людської діяльності. Запропоновані підходи сприяють розробці кіберфізичних систем, які моделюють рухи на основі біомеханічних принципів, що є важливим для інтеграції фізичних і обчислювальних процесів у спортивній практиці та наукових дослідженнях.

Результати дослідження докладено на міжнародній науковій конференції «51 років наукового співробітництва» Штутгартського Університету та Донецького Національного Технічного Університету, яка відбулася 07/08-го листопада 2024. За результатами цієї конференції прийнято статтю до публікації до Всеукраїнського наукового збірнику Донецького національного технічного університету [22]. Окрім цього вони опубліковані на XXI Всеукраїнській конференції студентів і молодих науковців [3].

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lees A. Technique analysis in sports: a critical review [Електронний ресурс] / Adrian Lees // Journal of Sports Sciences. – 2002. – Т. 20, № 10. – С. 813–828. – DOI: https://doi.org/10.1080/026404102320675657

2. Technique analysis in elite athletes using principal component analysis [Електронний ресурс] / Øyvind Gløersen [та ін.] // Journal of Sports Sciences. – 2017. – Т. 36, № 2. – С. 229–237. – DOI: https://doi.org/10.1080/02640414.2017.1298826

3. Гальчинський М. В. Аналіз та рекомендації у техніці плавання / М. В. Гальчинський, Т. І. Петрушина // Інформатика, інформаційні системи та технології: XXI Всеукраїнська конференція студентів і молодих науковців. 26 квітня, 2024, Одеса – С. 61-63.

4. Lee E. A., Seshia S. A. Introduction to Embedded Systems: A Cyber-Physical Systems Approach. MIT Press, 2017. 568 p.

5. Taha W. M., Taha A. E. M., Thunberg J. Cyber-Physical Systems: A Model-Based Approach. Cham : Springer International Publishing, 2021. 187 p.

6. Hochreiter S. Long Short-Term Memory [Електронний ресурс] / Sepp Hochreiter, Jürgen Schmidhuber // Neural Computation. – 1997. – Т. 9, № 8. – С. 1735–1780. – DOI: https://doi.org/10.1162/neco.1997.9.8.1735

7. Long Short-Term Memory Networks (LSTM). Data Basecamp. [Електронний ресурс] – Режим доступу: https://databasecamp.de/en/ml/lstms – 28.11.2024.

8. Howard J. Deep Learning for Coders with Fastai and Pytorch / Jeremy Howard, Sylvain Gugger. – [Б. м.] : O'Reilly Media, Incorporated, 2020.

9. A Comprehensive Survey on Graph Neural Networks [Електронний ресурс] / Zonghan Wu [та ін.] // IEEE Transactions on Neural Networks and

Learning Systems. – 2021. – Т. 32, № 1. – С. 4–24. – DOI: https://doi.org/10.1109/tnnls.2020.2978386

10. Deep High-Resolution Representation Learning for Visual Recognition / J. Wang та ін. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020. С. 1. [Електронний ресурс] – DOI: https://doi.org/10.1109/tpami.2020.2983686.

11. Deep High-Resolution Representation Learning for Human Pose Estimation / K. Sun та ін. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), м. Long Beach, CA, USA, 15–20 черв. 2019 р. [Електронний ресурс] – DOI: https://doi.org/10.1109/cvpr.2019.00584.

12. Video Transformers: A Survey [Електронний ресурс] / Javier Selva [та ін.] // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2023. – С. 1–20. – DOI: https://doi.org/10.1109/tpami.2023.3243465.

13. Demarcq G. Complete Guide to OpenPose. Build and deploy AI with your own API – Custom AI solutions. [Електронний ресурс] – Режим доступу: https://www.ikomia.ai/blog/complete-openpose-guide#compare-openpose-to-other-human-pose-estimation-algorithms – 28.11.2024.

14. End-to-End Recovery of Human Shape and Pose / A. Kanazawa та ін. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), м. Salt Lake City, UT, 18–23 черв. 2018 р. [Електронний ресурс] – DOI: https://doi.org/10.1109/cvpr.2018.00744.

15. MMPose 1.3.2 documentation. [Електронний ресурс] – Режим доступу: https://mmpose.readthedocs.io/en/latest/ – 28.11.2024.

16. El Kaid A. A Systematic Review of Recent Deep Learning Approaches for 3D Human Pose Estimation [Електронний ресурс] / Amal El Kaid, Karim Baïna // Journal of Imaging. – 2023. – Т. 9, № 12. – С. 275. – DOI: https://doi.org/10.3390/jimaging9120275.

17. The application of principal component analysis to quantify technique in sports [Електронний ресурс] / P. Federolf [та ін.] // Scandinavian Journal of

Medicine & Science in Sports. – 2012. – Т. 24, № 3. – С. 491–499. – DOI: https://doi.org/10.1111/j.1600-0838.2012.01455.x

18. Ma X. 3D Deep Learning with Python: Design and Develop Your Computer Vision Model with 3D Data Using PyTorch3D and More / Xudong Ma, Vishakh Hegde, Lilit Yolyan. – [Б. м.] : Packt Publishing, Limited, 2022.

19. PCA. scikit-learn. [Електронний ресурс] – Режим доступу: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html – 28.11.2024.

20. pandas.Series.interpolate. pandas - Python Data Analysis Library. [Електронний ресурс] – Режим доступу: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.interpolate.html – 28.11.2024.

21. gaussian_filter1d. Numpy and Scipy Documentation. [Електронний ресурс] – Режим доступу: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter1d.html – 28.11.2024.

22. Galchynskyi M., Petrushina T., Malakhov E. Intellectual video analysis of swimming techniques to develop a cyber-physical movement pattern. Problems of Modeling and Design Automatization. – pp. 5, Прийнята до публикації у 2024 р

ДОДАТОК А

Dockerfile

```
ARG PYTORCH="1.9.0"
ARG CUDA="11.1"
ARG CUDNN="8"

FROM pytorch/pytorch:${PYTORCH}-cuda${CUDA}-cudnn${CUDNN}-devel

ENV TORCH_CUDA_ARCH_LIST="6.0 6.1 7.0 7.5 8.0 8.6+PTX" \
    TORCH_NVCC_FLAGS="-Xfatbin -compress-all" \
    CMAKE_PREFIX_PATH="$(dirname $(which conda))/../" \
    FORCE_CUDA="1"

# Avoid Public GPG key error and install system dependencies
RUN rm /etc/apt/sources.list.d/cuda.list \
    && rm /etc/apt/sources.list.d/nvidia-ml.list \
    && apt-key del 7fa2af80 \
    && apt-key adv --fetch-keys
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804
/x86_64/3bf863cc.pub \
    && apt-key adv --fetch-keys
https://developer.download.nvidia.com/compute/machine-learning/repo
s/ubuntu1804/x86_64/7fa2af80.pub \
    && apt-get update \
    && apt-get install -y --no-install-recommends \
        ffmpeg \
        libsm6 \
        libxext6 \
        git \
        ninja-build \
        libglib2.0-0 \
        libxrender-dev \
        libgl1-mesa-glx \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*
```

```
# Install MMEngine, MMCV, and xtcocotools
RUN pip install openmim \
    && mim install "mmengine>=0.7.1" \
    && pip install cython \
    && pip install xtcocotools

# Install MMDetection
RUN conda clean --all \
    && git clone https://github.com/open-mmlab/mmdetection.git
/workspace/mmdetection \
    && pip install --no-cache-dir -e /workspace/mmdetection

# Install compatible version of MMCV
RUN pip install mmcv==2.0.0rc4 -f
https://download.openmmlab.com/mmcv/dist/cu111/torch1.9/index.html

# Install MMPose
RUN git clone https://github.com/open-mmlab/mmpose.git
/workspace/mmpose \
    && pip install -r /workspace/mmpose/requirements/build.txt \
    && pip install --no-cache-dir -e /workspace/mmpose

# Install desktop app dependencies
RUN pip install -U fbs requests

RUN apt-get update && apt-get install -y --no-install-recommends \
    libxkbcommon-x11-0 \
    libxcb-xinerama0 \
    libxcb-xinput0 \
    libxcb-xfixes0 \
    libxcb-keysyms1 \
    libxcb-image0 \
    libxcb-icccm4 \
    libxcb-render-util0 \
    libxcb-shape0 \
    libxcb-xkb1 \
    libx11-xcb1 \
    libfontconfig1 \
    libfreetype6 \
    libxrender1 \
    libx11-dev \
    libdbus-1-3 \
```

```
    libxtst6 \
    libxcomposite1 \
    libxcursor1 \
    libxi6 \
    libxrandr2 \
    libxss1 \
    libxcb-randr0 \
    libgl1-mesa-dri \
    libxt6

RUN pip install open3d==0.16.0
RUN pip install PyQt5==5.15.9
# Удалить текущие версии OpenCV
RUN pip uninstall -y opencv-python opencv-python-headless

# Установить opencv-python-headless
RUN pip install opencv-python-headless


ENV
QT_QPA_PLATFORM_PLUGIN_PATH=/opt/conda/lib/python3.7/site-packages/
PyQt5/Qt5/plugins
ENV XDG_RUNTIME_DIR=/workspace/tmp
ENV LIBGL_ALWAYS_INDIRECT=0

# Make port 80 available to the world outside this container
EXPOSE 80

WORKDIR /workspace
```

ДОДАТОК Б

Приклад json з передбаченням ключових точок тіла людини


```
{
    "meta_info": {
        "dataset_name": "h36m",
        "num_keypoints": 17,
        "keypoint_id2name": {
            "0": "root",
            "1": "right_hip",
            "2": "right_knee",
            "3": "right_foot",
            "4": "left_hip",
            "5": "left_knee",
            "6": "left_foot",
            "7": "spine",
            "8": "thorax",
            "9": "neck_base",
            "10": "head",
            "11": "left_shoulder",
            "12": "left_elbow",
            "13": "left_wrist",
            "14": "right_shoulder",
            "15": "right_elbow",
            "16": "right_wrist"
        },
        "keypoint_name2id": {
            "root": 0,
            "right_hip": 1,
            "right_knee": 2,
            "right_foot": 3,
            "left_hip": 4,
            "left_knee": 5,
            "left_foot": 6,
            "spine": 7,
            "thorax": 8,
            "neck_base": 9,
            "head": 10,
            "left_shoulder": 11,
```

```
            "left_elbow": 12,
            "left_wrist": 13,
            "right_shoulder": 14,
            "right_elbow": 15,
            "right_wrist": 16
        },
"upper_body_ids": [7, 8, 9, 10, 11, 12, 13, 14, 15, 16],
"lower_body_ids": [0, 1, 2, 3, 4, 5, 6],
"flip_indices": [0, 4, 5, 6, 1, 2, 3, 7, 8, 9, 10, 14, 15, 16, 11,
12, 13],
"flip_pairs": [[4, 1], [5, 2], [6, 3], [1, 4], [2, 5], [3, 6], [14,
11], [15, 12], [16, 13], [11, 14], [12, 15], [13, 16]],
"keypoint_colors": {"__ndarray__": [[51, 153, 255], [255, 128, 0],
[255, 128, 0], [255, 128, 0], [0, 255, 0], [0, 255, 0], [0, 255,
0], [51, 153, 255], [51, 153, 255], [51, 153, 255], [51, 153, 255],
[0, 255, 0], [0, 255, 0], [0, 255, 0], [255, 128, 0], [255, 128,
0], [255, 128, 0]], "dtype": "uint8", "shape": [17, 3], "Corder":
true},
"num_skeleton_links": 16, "skeleton_links": [[0, 4], [4, 5], [5,
6], [0, 1], [1, 2], [2, 3], [0, 7], [7, 8], [8, 9], [9, 10], [8,
11], [11, 12], [12, 13], [8, 14], [14, 15], [15, 16]],
"skeleton_link_colors": {"__ndarray__": [[0, 255, 0], [0, 255, 0],
[0, 255, 0], [255, 128, 0], [255, 128, 0], [255, 128, 0], [51, 153,
255], [51, 153, 255], [51, 153, 255], [51, 153, 255], [0, 255, 0],
[0, 255, 0], [0, 255, 0], [255, 128, 0], [255, 128, 0], [255, 128,
0]], "dtype": "uint8", "shape": [16, 3], "Corder": true},
"dataset_keypoint_weights": {"__ndarray__": [1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
            "dtype": "float32",
            "shape": [
                17
            ]
        },
        "fps": 30.0
    },
    "instance_info": [
        {
            "frame_id": 1,
```

```
            "keypoints": [
[0.0007917770999483764,-0.0001913963642437011,0.5789320468902588],[
-0.046536628156900406,-0.1407005935907364,0.5370981097221375],[-0.0
36861080676317215,0.12584608793258667,0.0],[0.48669761419296265,0.5
64525842666626,0.3967871367931366],[0.04703128710389137,0.141004532
5756073,0.6206258535385132],[0.3544907867908478,0.30605724453926086
,0.06891220808029175],[0.42429596185684204,0.6920239925384521,0.350
61895847320557],[-0.24070113897323608,-0.1583109498023986 8,0.896225
5120277405],[-0.5026196837425232,-0.22939765453338623,1.12423551082
61108],[-0.8127425909042358,-0.24064038693904877,1.1934218406677246
],[-0.8857938051223755,-0.2623131573200226,1.2780468463897705],[-0.
4568904638290405,-0.02199520915746689,1.315810203552246],[-0.943257
4510574341,0.1899578720331192,1.304337978363037],[-1.12152588367462
16,-0.01585179939866066,1.3704712390899658],[-0.5059794187545776,-0
.4532080888748169,1.1254767179489136],[-0.8106499314308167,-0.53921
59819602966,1.0912714004516602],[-1.043130874633789,-0.400281518697
73865,1.1925218105316162]
            ],
            "keypoint_scores": [
[0.4569229483604431, 0.31911611557006836, 0.3782806396484375,
0.2326609194278717, 0.3816274404525757, 0.32331526279449463,
0.3717804551124573, 0.155861034989357, 0.3054439425468445,
0.1852579265832901, 0.41669291257858276, 0.4429534673690796,
0.44877806305885315, 0.399034708738327, 0.3920191824436188,
0.385187566280365, 0.49607568979263306]
            ]
        },{...},...]}
```

ДОДАТОК В

Програмна реалізація визначення ключових точок тіла людини

```
import mimetypes
import os
from argparse import ArgumentParser
from functools import partial
import cv2
import json_tricks as json
import mmcv
import numpy as np
from mmpose.apis import (_track_by_iou, _track_by_oks,
    convert_keypoint_definition, extract_pose_sequence,
    inference_pose_lifter_model, inference_topdown, init_model)
from mmpose.models.pose_estimators import PoseLifter
from mmpose.models.pose_estimators.topdown import
TopdownPoseEstimator
from mmpose.structures import (PoseDataSample, merge_data_samples,
                               split_instances)
from mmpose.utils import adapt_mmdet_pipeline
try:
    from mmdet.apis import inference_detector, init_detector
    has_mmdet = True
except (ImportError, ModuleNotFoundError):
    has_mmdet = False
class Body3D:
    def __init__(self):
        self._init_models_path()
        self._init_device()
        self._init_detector()
        self._init_pose_estimator()
        self._init_pose_lifter()
```

```
    def _init_models_path(self):
        base_path = "/workspace"
        self.det_config = os.path.join(base_path,
"mmpose/demo/mmdetection_cfg/rtmdet_m_640-8xb32_coco-person.py")
        self.det_checkpoint =
"data/models/rtmdet_m_8xb32-100e_coco-obj365-person-235e8209.pth"
        self.pose_estimator_config = os.path.join(base_path,
"mmpose/configs/body_2d_keypoint/rtmpose/body8/rtmpose-m_8xb256-420
e_body8-256x192.py")
        self.pose_estimator_checkpoint =
"data/models/rtmpose-m_simcc-body7_pt-body7_420e-256x192-e48f03d0_2
0230504.pth"
        self.pose_lifter_config = os.path.join(base_path,
"mmpose/configs/body_3d_keypoint/video_pose_lift/h36m/video-pose-li
ft_tcn-243frm-supv-cpn-ft_8xb128-200e_h36m.py")
        self.pose_lifter_checkpoint =
"data/models/videopose_h36m_243frames_fullconv_supervised_cpn_ft-88
f5abbb_20210527.pth"

    def _init_device(self):
        self.device = 'cuda:0'.lower()

    def _init_detector(self):
        self.detector = init_detector(self.det_config,
self.det_checkpoint, device=self.device)
        self.detector.cfg = adapt_mmdet_pipeline(self.detector.cfg)

    def _init_pose_estimator(self):
        self.pose_estimator = init_model(
            self.pose_estimator_config,
            self.pose_estimator_checkpoint,
            device=self.device)
        assert isinstance(self.pose_estimator,
TopdownPoseEstimator), 'Only "TopDown"' \
            'model is supported for the 1st stage (2D pose
detection)'

    def _init_pose_lifter(self):
        self.pose_lifter = init_model(
            self.pose_lifter_config,
            self.pose_lifter_checkpoint,
            device=self.device)
        assert isinstance(self.pose_lifter, PoseLifter), \
```

```
            'Only "PoseLifter" model is supported for the 2nd stage
' \
            '(2D-to-3D lifting)'

    def
_get_estimated_pose_results(self,frame,bbox_thr=0.3,det_cat_id=0):

        # use detector to obtain person bounding boxes
        det_result = inference_detector(self.detector, frame)
        pred_instance = det_result.pred_instances.cpu().numpy()

        # filter out the person instances with category and bbox
threshold
        # e.g. 0 for person in COCO
        bboxes = pred_instance.bboxes
        bboxes = bboxes[np.logical_and(pred_instance.labels ==
det_cat_id,
                                       pred_instance.scores >
bbox_thr)]

        return inference_topdown(self.pose_estimator, frame,
bboxes)

    def _convert_est_pose_res_to_pose_lifting_format(self,
pose_est_results,pose_est_results_last,next_id,

use_oks_tracking = False, tracking_thr = 0.3):

        pose_est_results_converted = []
        for i, data_sample in enumerate(pose_est_results):
            pred_instances =
data_sample.pred_instances.cpu().numpy()
            keypoints = pred_instances.keypoints
            # calculate area and bbox
            if 'bboxes' in pred_instances:
                areas = np.array([(bbox[2] - bbox[0]) * (bbox[3] -
bbox[1])
                                  for bbox in pred_instances.bboxes])
                pose_est_results[i].pred_instances.set_field(areas,
'areas')
            else:
                areas, bboxes = [], []
                for keypoint in keypoints:
```

```
                    xmin = np.min(keypoint[:, 0][keypoint[:, 0] >
0], initial=1e10)
                    xmax = np.max(keypoint[:, 0])
                    ymin = np.min(keypoint[:, 1][keypoint[:, 1] >
0], initial=1e10)
                    ymax = np.max(keypoint[:, 1])
                    areas.append((xmax - xmin) * (ymax - ymin))
                    bboxes.append([xmin, ymin, xmax, ymax])
                pose_est_results[i].pred_instances.areas =
np.array(areas)
                pose_est_results[i].pred_instances.bboxes =
np.array(bboxes)

            # track id
            if use_oks_tracking:
                _track = partial(_track_by_oks)
            else:
                _track = _track_by_iou
            track_id, pose_est_results_last, _ =
_track(data_sample,

pose_est_results_last,

tracking_thr)
            if track_id == -1:
                if np.count_nonzero(keypoints[:, :, 1]) >= 3:
                    track_id = next_id
                    next_id += 1
                else:
                    # If the number of keypoints detected is small,
                    # delete that person instance.
                    keypoints[:, :, 1] = -10
                    pose_est_results[i].pred_instances.set_field(
                        keypoints, 'keypoints')
                    pose_est_results[i].pred_instances.set_field(
                        pred_instances.bboxes * 0, 'bboxes')
                    pose_est_results[i].set_field(pred_instances,
'pred_instances')
                    track_id = -1
            pose_est_results[i].set_field(track_id, 'track_id')

            # convert keypoints for pose-lifting
            pose_est_result_converted = PoseDataSample()
```

```
            pose_est_result_converted.set_field(
                pose_est_results[i].pred_instances.clone(),
'pred_instances')
            pose_est_result_converted.set_field(
                pose_est_results[i].gt_instances.clone(),
'gt_instances')
            pose_lift_dataset_name =
self.pose_lifter.dataset_meta['dataset_name']
            pose_det_dataset_name =
self.pose_estimator.dataset_meta['dataset_name']
            keypoints = convert_keypoint_definition(keypoints,

pose_det_dataset_name,

pose_lift_dataset_name)
            pose_est_result_converted.pred_instances.set_field(
                keypoints, 'keypoints')

pose_est_result_converted.set_field(pose_est_results[i].track_id,
                                                    'track_id')

pose_est_results_converted.append(pose_est_result_converted)


        return pose_est_results_converted.copy()

    def _process_one_image(self,frame,frame_idx,
        pose_est_results_last, pose_est_results_list, next_id,
        disable_rebase_keypoint = False, disable_norm_pose_2d =
False):

        # First stage: conduct 2D pose detection in a Topdown
manner

        # estimate pose results for current image
        pose_est_results=self._get_estimated_pose_results(frame)

        pose_est_results_list.append(

self._convert_est_pose_res_to_pose_lifting_format(pose_est_results,
pose_est_results_last,next_id))

        # Second stage: Pose lifting
```

```
        # extract and pad input pose2d sequence

        pose_lift_dataset =
self.pose_lifter.cfg.test_dataloader.dataset
        pose_seq_2d = extract_pose_sequence(
            pose_est_results_list,
            frame_idx=frame_idx,
            causal=pose_lift_dataset.get('causal', False),
            seq_len=pose_lift_dataset.get('seq_len', 1),
            step=pose_lift_dataset.get('seq_step', 1))

        # conduct 2D-to-3D pose lifting
        norm_pose_2d = not disable_norm_pose_2d
        pose_lift_results = inference_pose_lifter_model(
            self.pose_lifter,
            pose_seq_2d,
            image_size=frame.shape[:2],
            norm_pose_2d=norm_pose_2d)

        # post-processing
        for idx, pose_lift_result in enumerate(pose_lift_results):
            pose_lift_result.track_id =
pose_est_results[idx].get('track_id', 1e4)

            pred_instances = pose_lift_result.pred_instances
            keypoints = pred_instances.keypoints
            keypoint_scores = pred_instances.keypoint_scores
            if keypoint_scores.ndim == 3:
                keypoint_scores = np.squeeze(keypoint_scores,
axis=1)
                pose_lift_results[
                    idx].pred_instances.keypoint_scores =
keypoint_scores
            if keypoints.ndim == 4:
                keypoints = np.squeeze(keypoints, axis=1)

            keypoints = keypoints[..., [0, 2, 1]]
            keypoints[..., 0] = -keypoints[..., 0]
            keypoints[..., 2] = -keypoints[..., 2]

            # rebase height (z-axis)
            if not disable_rebase_keypoint:
                keypoints[..., 2] -= np.min(
```

```
                    keypoints[..., 2], axis=-1, keepdims=True)

            pose_lift_results[idx].pred_instances.keypoints =
keypoints

        pose_lift_results = sorted(
            pose_lift_results, key=lambda x: x.get('track_id',
1e4))

        pred_3d_data_samples =
merge_data_samples(pose_lift_results)
        det_data_sample = merge_data_samples(pose_est_results)
        pred_3d_instances =
pred_3d_data_samples.get('pred_instances', None)

        return pose_est_results, pose_est_results_list,
pred_3d_instances, next_id

    def get_predictions(self,mmcv_frames):
        pose_est_results_list = []
        pred_instances_list = []
        next_id = 0
        pose_est_results = []
        frame_idx = 0

        for frame in mmcv_frames:
            frame_idx += 1

            pose_est_results_last = pose_est_results

            # First stage: 2D pose detection
            # make person results for current image
            (pose_est_results, pose_est_results_list,
pred_3d_instances,
             next_id) = self._process_one_image(
                 frame=mmcv.bgr2rgb(frame),
                 frame_idx=frame_idx,
                 pose_est_results_last=pose_est_results_last,
                 pose_est_results_list=pose_est_results_list,
                 next_id=next_id)

            for i in range(len(pred_3d_instances)):
```

```
                np.copyto(pred_3d_instances[i].keypoint_scores,
pose_est_results[i].pred_instances.keypoint_scores)



            # save prediction results
            pred_instances_list.append(dict(
                        frame_id=frame_idx,
                        keypoints =
pred_3d_instances[0].keypoints[0].tolist(),
                        keypoint_scores =
pred_3d_instances[0].keypoint_scores[0].tolist()
                        ))


        return pred_instances_list

    def get_predictions_with_metadata(self,frames,fps):
        pred_instances_list = self.get_predictions(frames)
        result_dict = dict(
            meta_info={
        **self.pose_lifter.dataset_meta,
        "fps": fps
        },
            instance_info=pred_instances_list
        )
        return result_dict

    def get_json_predictions(self,frames,fps):
        predictions =
self.get_predictions_with_metadata(frames,fps)
        return json.dumps(predictions, indent='\t')
```

ДОДАТОК Г

Програмна реалізація порівняння патерну руху

```
from sklearn.decomposition import PCA
import numpy as np
import json_manager as jm
from movement_pattern_builder import MovementPatternBuilder
from video_sequence import VideoSequence
import matplotlib.pyplot as plt

keypoint_id2name = {
    0: "root",
    1: "right_hip",
    2: "right_knee",
    3: "right_foot",
    4: "left_hip",
    5: "left_knee",
    6: "left_foot",
    7: "spine",
    8: "thorax",
    9: "neck_base",
    10: "head",
    11: "left_shoulder",
    12: "left_elbow",
    13: "left_wrist",
    14: "right_shoulder",
    15: "right_elbow",
    16: "right_wrist"
}

def extract_keypoint_lengths(sequence: list):
    lengths = []
    for frame in sequence:
        #lenghts of each keypoint in the frame
        frame_lengths = [keypoint.get_length() for keypoint in
frame.keypoints]
        lengths.append(frame_lengths)
    return np.array(lengths)
```

```
def reorganize_lengths(lengths: list):
    reorganized_lengths = [[] for _ in range(len(lengths[0]))]
    for frame_lengths in lengths:
        for i in range(len(frame_lengths)):
            reorganized_lengths[i].append(frame_lengths[i])
    return np.array(reorganized_lengths)

def calculate_variance(lengths: list):
    result = []
    for i in range(len(lengths)):
        result.append(np.var(lengths[i]))
    return result

def calculate_ration_of_sum(variance):
    return variance/sum(variance)

def calculate_ration_of_body(variance,body):
    return variance/body

def compare_sequences(sequence1: list, sequence2: list):
    lengths1 = extract_keypoint_lengths(sequence1)
    lengths2 = extract_keypoint_lengths(sequence2)

    reorganized_lengths1 = reorganize_lengths(lengths1)
    reorganized_lengths2 = reorganize_lengths(lengths2)

    # Comparing the contribution of each key point
    var1 = calculate_variance(reorganized_lengths1)
    var2 = calculate_variance(reorganized_lengths2)

    ratio1 = calculate_ration_of_sum(var1)
    ratio2 = calculate_ration_of_sum(var2)
    recommendations = []
    for i in range(len(var1)):
        recommendations.append((i, ratio1[i], ratio2[i]))

    return recommendations

def compare_sequences_pca(sequence1: list, sequence2: list):
    lengths1 = extract_keypoint_lengths(sequence1)
    lengths2 = extract_keypoint_lengths(sequence2)

    pca1 = PCA()
```

```
    pca1.fit(lengths1)
    explained_variance_ratios1 = pca1.explained_variance_ratio_

    pca2 = PCA()
    pca2.fit(lengths2)
    explained_variance_ratios2 = pca2.explained_variance_ratio_

    # Comparing the contribution of each key point
    recommendations = []
    for i in range(len(explained_variance_ratios1)):
        recommendations.append((i, explained_variance_ratios1[i],
explained_variance_ratios2[i]))

    return recommendations

def get_movement_pattern(filename):
    predictions = jm.get_data_from_json_file(filename)
    frames = jm.get_frames_from_json(predictions)
    video = VideoSequence(frames)
    builder = MovementPatternBuilder(video)
    return builder.build()

def plot_keypoint_contributions(result):

    keypoint_indices = [idx for idx, _, _ in result]
    var1_values = [var1 for _, var1, _ in result]
    var2_values = [var2 for _, _, var2 in result]
    keypoint_names = [keypoint_id2name[idx] for idx in
keypoint_indices]

    x = np.arange(len(keypoint_names))
    width = 0.35

    fig, ax = plt.subplots(figsize=(10, 6))
    ax.bar(x - width / 2, var1_values, width, label='Sequence 1',
color='skyblue')
    ax.bar(x + width / 2, var2_values, width, label='Sequence 2',
color='salmon')

    ax.set_xlabel('Keypoints')
    ax.set_ylabel('Contribution')
    ax.set_title('Comparison of Keypoint Contributions in Two
Sequences')
```

```
    ax.set_xticks(x)
    ax.set_xticklabels(keypoint_names, rotation=45, ha="right")
    ax.legend()

    plt.tight_layout()
    plt.show()

def build_comparison_plt(pattern1, pattern2):
    result = compare_sequences(pattern1, pattern2)
    plot_keypoint_contributions(result)


def main():
    filename1 = '/home/max/code/data/vis_results/results_1.json'
    filename2 = '/home/max/code/data/vis_results/results_2.json'
    pattern1 = get_movement_pattern(filename1)
    pattern2 = get_movement_pattern(filename2)
    build_comparison_plt(pattern1, pattern2)

if __name__ == '__main__':
    main()
```

Людмила ВОЛОЩУК
Подписано цифровой подписью: Людмила ВОЛОЩУК
Дата: 2024.12.05 13:09:59 +02'00'