МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп’ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ГАДЕВИЧ Володимир Юрійович

Програмний модуль розпізнавання пози людини для системи комп'ютерного зору / Human pose recognition software module for computer vision system

Спеціальність 122 – Комп’ютерні науки
Освітньо-професійна програма – Комп’ютерні науки

Дипломний проект

Виконав студент групи КН-42
В.Ю. Гадевич

Науковий керівник:
к.т.н., доцент Д.І. Загородня

Дипломний проект допущено до
захисту
«___»_____ 2023 р.

Завідувач кафедри
_____ М.П. Комар

Тернопіль – 2023

Факультет комп’ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
Спеціальність 122 – Комп’ютерні науки
Освітньо-професійна програма – Комп’ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2022р.

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Гадевичу Володимиру Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема проекту: Програмний модуль розпізнавання пози людини для системи комп'ютерного зору / Human pose recognition software module for computer vision system
керівник проекту к.т.н., доцент Д.І. Загородня

затверджені наказом по університету від 08 грудня 2022 р. № 491.

2. Строк подання студентом закінченого проекту 01 червня 2023 р.

3. Вихідні дані до проекту: технічне завдання.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- провести аналіз предметної області;
- провести аналіз існуючих систем розпізнавання пози людини;
- зробити постановку задач дослідження;
- розробити структуру модуля;
- розробити алгоритмічне забезпечення модуля;
- провести реалізацію програмного забезпечення;
- розробити та реалізувати інтерфейс користувача;
- провести тестування програмного модуля.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

- структура програмного модуля;
- алгоритм роботи програмного модуля.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Н. контроль	к.т.н., доцент Д.І. Загородня		

7. Дата видачі завдання 08 грудня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1	Аналіз систем комп'ютерного зору	30.12.2022	
2	Алгоритмічне та інформаційне забезпечення модуля	24.03.2023	
3	Програмно-технологічне забезпечення модуля	12.05.2023	
4	Повне завершення та оформлення дипломного проекту	01.06.2023	

Студент_____ В.Ю. Гадевич
(підпис)

Керівник проекту_____ Д.І. Загородня
(підпис)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту: 65 с., 33 рис., 3 додатки, 18 джерел.

Метою дипломного проекту є розробка програмного модуля, який здатний виявляти та розпізнавати пози людини на зображеннях або відео на основі моделей глибокого навчання

Об’єкт дослідження – система комп'ютерного зору та процеси розпізнавання пози людини.

Предмет дослідження – методи та засоби розпізнавати пози людини.

Розроблено та досліджено програмний модуль розпізнавання пози людини для системи комп'ютерного зору.

Запропонований програмний модуль розпізнає позу людини на зображенні та будує її скелет. Результати програмного модуля можуть бути використані в системах комп’ютерного зору.

КОМП’ЮТЕРНИЙ ЗІР, ПОЗА ЛЮДИНИ, АЛГОРИТМ, МОДЕЛІ ГЛИБОКОГО НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ФРЕЙМВОРК TENSORFLOW, БІБЛІОТЕКА KERAS.

ABSTRACT

The bachelor's thesis report: 65 pages, 33 figures, 3 appendices, 18 sources.

The aim of the diploma project is to develop a software module capable of detecting and recognizing human poses in images or videos based on deep learning models

The object of the research is the computer vision system and human posture recognition processes.

The subject of research is methods and means of recognizing human postures.

A human pose recognition software module for the computer vision system was developed and researched.

The proposed software module recognizes the pose of a person in the image and builds its skeleton. The results of the software module can be used in computer vision systems.

COMPUTER VISION, NON-HUMAN, ALGORITHM, DEEP LEARNING MODELS, NEURAL NETWORKS, TENSORFLOW FRAMEWORK, KERAS LIBRARY.

ТЕХНІЧНЕ ЗАВДАННЯ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

1.1 Програмний модуль розпізнавання пози людини для системи комп'ютерного зору.

1.2 Область застосування – комп’ютерний зір.

2. ОСНОВА ДЛЯ РОЗРОБЛЕННЯ

Основою для розроблення є завдання на дипломний проект, затверджене кафедрою інформаційно-обчислювальних систем і управління факультету комп’ютерних інформаційних технологій Західноукраїнського національного університету.

3. ПРИЗНАЧЕННЯ РОЗРОБЛЕНОГО КОМПЛЕКСУ

Метою дипломного проекту є розробка програмного модуля, який здатний виявляти та розпізнавати пози людини на зображеннях на основі моделей глибокого навчання

4. ДЖЕРЕЛА РОЗРОБЛЕННЯ

Джерелами даної розробки є матеріали навчальної і реферативної літератури, технічна документація, науково-дослідні статті, журнали, Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Основні функціональні вимоги до програмної системи:

– зручність та зрозумілість представлених результатів для користувача;

– можливість передачі даних та отриманих результатів роботи програмного модуля у файл – можливість експорту даних;

– здатність програмного модуля до подальших змін та доповнень в майбутньому, тобто дотримання принципу модульності;

– здатність програми інтегруватись в систему комп’ютерного зору;

– вхідна інформація отримується шляхом завантаження зображення із бази або відеокамери.

5.2 Вимоги до апаратних засобів:

– модуль повинен працювати на IBM-сумісних робочих станціях;

– мінімальні вимоги до робочих станцій: процесор від 2 ГГц, оперативна пам’ять від 1 Гб, відеокарта від 512 Мб, об’єм пам’яті на жорсткому диску до 1 Гб, клавіатура, маніпулятор «миша», відеокамера.

5.3 Вимоги до програмних засобів:

– для розробки програмне забезпечення - Python 3.7;

– для реалізації моделей глибокого навчання – фреймворк Tensorflow, бібліотека Keras.

– операційна система сімейства Windows та браузер для роботи з Colab.

6. ПОРЯДОК КОНТРОЛЮ

6.1 Представлення дипломного проекту на попередній захист.

6.2 Представлення дипломного проекту на захист.

Завдання прийняв до виконання _____ В.Ю. Гадевич

(підпис) (прізвище та ініціали)

Керівник дипломного проекту_____ Д.І. Загородня

(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Комп’ютерний зір – це сфера, де комп'ютери та системи отримують цінну інформацію з цифрових зображень, відео та інших візуальних даних і використовують цю інформацію для прийняття рішень або надання рекомендацій.

Одним з важливих аспектів систем комп'ютерного зору є розпізнавання людської пози, яке має великий потенціал у таких областях, як безпека, здоров'я, робототехніка та спортивний аналіз. Оцінка пози — це одне із завдань комп’ютерного зору та штучного інтелекту, яке включає виявлення та відстеження положення та орієнтації частин тіла людини на зображеннях або відео.

Розробка програмного модуля розпізнавання пози людини відкриє широкі перспективи для автоматизації процесів, поліпшення безпеки та ефективності у різних областях. Наприклад, в системах відеоспостереження модуль зможе автоматично виявляти небезпечні або підозрілі пози людей, сприяючи вчасному реагуванню та запобіганню можливим негативним подіям. У спортивному аналізі модуль може допомогти відстежувати та аналізувати рухи спортсменів, надаючи цінні дані для тренувань та покращення результатів. Окрім того, програмний модуль розпізнавання пози людини може бути використаний у медичній діагностиці для оцінки постави пацієнтів або в реабілітаційних системах для контролю рухів та вправ. Також він може знайти застосування у віртуальній реальності та ігровій індустрії, де реалістичне відтворення рухів людини є важливою складовою.

Мета дипломного проекту полягає в розробці програмного модуля, який здатний виявляти та розпізнавати пози людини на зображеннях. Модуль буде здатен аналізувати структуру людського тіла, визначати положення та орієнтацію окремих частин, таких як руки, ноги, голова тощо. Це дозволить отримати цінну інформацію про стан людини, її рухи та позу.

Об'єктом дослідження є система комп'ютерного зору що використовує модуль розпізнавання пози людини.

Предметом дослідження є розробка та впровадження програмного модуля, який здатний визначати та розпізнавати позу людини на основі вхідних зображень, отриманих від системи комп'ютерного зору.

РОЗДІЛ 1 АНАЛІЗ СИСТЕМ КОМП’ЮТЕРНОГО ЗОРУ

1.1 Коротка характеристика систем комп’ютерного зору

Сьогодні величезні обсяги візуальних даних фіксуються в цифрових зображеннях, відео та 3D-даних. Завдання полягає в тому, як ефективно та економічно ефективно оптимізувати ці дані та витягнути з них цінність — чи це отримання інформації про поведінку клієнтів для оптимізації розміщення продукту в роздрібних торговельних середовищах, виробництво продуктів вищої якості шляхом виявлення дефектів на виробничій лінії, виявлення шахрайства на самоконтроль або виявлення загроз у громадських місцях [1].

Комп’ютерний зір використовується в різних галузях промисловості, від енергетики та комунальних послуг до виробництва та автомобілебудування – і його ринок продовжує зростати. В 2022 року він досяг 48,6 млрд доларів США [2].

Деякі з найпоширеніших сфер застосування комп’ютерного зору:

- автомобілебудування (системи комп'ютерного зору використовуються для моніторингу перехрестя на предмет інцидентів, надання допомоги автономним транспортним засобам у навігації та сприйнятті об'єктів. Вони також використовуються для розробки систем автопілота);
- виробництво. Системи комп'ютерного зору застосовуються для підрахунку та класифікації товарів на конвеєрній стрічці, аналізу штрих-кодів, управління запасами та автоматизованого вирізання.
- роздрібна торгівля (вони використовуються для моніторингу якості та кількості товарів на полицях, допомоги управлінню запасами, відстеження поведінки клієнтів та виявлення спрямованого погляду);
- оборона та безпека. Системи комп'ютерного зору використовуються для виявлення мін, розпізнавання обличчя, виявлення дефектів зброї та розробки безпілотних військових автомобілів.

– сільське господарство (застосовуються для моніторингу посівів за допомогою дронів та виявлення пошкоджених культур, автоматизованого обприскування пестицидами, фенотипування, виявлення та підрахунку худоби, інтелектуальних систем сортування та аналіз посівів).

– охорона здоров'я. Системи комп'ютерного зору використовуються для своєчасного виявлення захворювань, точної діагностики, вимірювання крововтрати, медичної візуалізації з більшою точністю, тактичної медицини та інших медичних додатків.

– робототехніка. Системи комп'ютерного зору допомагають в моніторингу та імітації руху, вивченні навколишнього середовища, визначенні швидкості рухомого об'єкта та навчанні конкретним маніпуляціям у робототехніці.

– страхування. Використання систем комп'ютерного зору у страхуванні включає аналіз активів, розрахунок суми збитків, аналіз даних документообігу та зменшення шахрайства, мінімізацію страхових спорів та надання правильної оцінки поданих вимог.

– мода. Використання систем комп'ютерного зору у модній індустрії включає розробку рішень "приміряти перед покупкою", аналіз одягу на основі іміджу, прогнозування модних тенденцій, визначення дизайнерських брендів та надання індивідуальних модних рекомендацій.

– медіа. Системи комп'ютерного зору використовуються для фільтрації та класифікації засобів масової інформації, ідентифікації фейкових новин, моніторингу впливу бренду, аналізу ефективності розміщення реклами та вимірювання уваги на екрані.

Використання розпізнавання об'єктів в цих сценаріях має на меті автоматизацію процесів, підвищення рівня безпеки і ефективності, а також отримання цінної інформації для прийняття рішень.

До загальних завдань систем комп’ютерного зору належать:

– виявлення об'єктів – це здатність розпізнавати, класифікувати та точно визначати просторове положення цікавих об’єктів на зображенні.

Положення, а також розширення об’єкта зазвичай фіксується за допомогою обмежувальних рамок (або інших фігур, багатокутників або еліпсоїдів).

– класифікація зображень – відноситься до процесу маркування кадру. На відміну від виявлення об’єктів, класифікація зображень спрямована на тегування зображення в цілому, а не його окремих компонентів. Цей клас визначає клас, до якого належить об’єкт.

– візуальне виявлення зв'язків. Це інше завдання комп’ютерного зору, яке виявляє зв’язок між двома чи більше об’єктами, щоб семантично описати зображення. На рисунку 1.1 два об’єкти поділяють «цикли» зв’язку.

– розпізнавання обличчя. Полягає у розпізнаванні облич і приписуванню їх певній людині. Є одним із найпоширеніших завдань комп’ютерного зору.

Рисунок 1.1 – Приклад виявлення зв’язків на зображенні.

– семантична сегментація – це процес ідентифікації подібних об'єктів, що належать до одного класу на піксельному рівні. Наприклад, на рисунку 1.2 модель намагалася ідентифікувати подібні об’єкти, в даному випадку людей, і позначила їх одним кольором, щоб виразити їхню приналежність до одного класу.

Рисунок 1.2 – Приклад семантичної сегментації

1.2 Опис предметної області систем комп’ютерного зору

Виявлення об'єктів є основним завданням для процесу розуміння зображень. Завдання виявлення об'єктів можна розділити на два послідовні підзавдання. Перше - це завдання пропозиції області, яке полягає в визначенні області, в якій може міститися об'єкт. Інше - це завдання класифікації зображень, яке полягає в визначенні, чи містить запропонована область певний об'єкт.

Найважливішим викликом, з яким стикається завдання виявлення об'єктів, є те, що об'єкти на зображеннях мають велику варіацію в масштабі, освітленні, кутах огляду, текстурі та позах. Метод, який може виявляти один вид об'єктів, може не впоратися з виявленням інших видів об'єктів. Крім того, метод, який підходить для виявлення великих об'єктів, не обов'язково добре працюватиме з виявленням малих об'єктів. Наприклад, область, яка містить об'єкт, може або охоплювати більше 90% всього зображення, або менше 10% зображення, що призводить до невдачі методів виявлення великих об'єктів у виявленні малих об'єктів.

Традиційно, методи виявлення об'єктів сканують велику кількість областей на зображенні. Потім з кожної області вилучають ручні ознаки, такі як SIFT, HOG і Haar-подібні ознаки. Нарешті, до отриманих ознак застосовується класифікатор, такий як SVM, AdaBoost або DPM щоб отримати результати класифікації [2].

З розвитком глибинного навчання, останнім часом область виявлення об'єктів була заселена методами на основі глибинного навчання. Методи виявлення об'єктів, засновані на глибинному навчанні, вимагають наявності наборів даних, що містять велику кількість точно позначених зображень для навчання. Основні набори даних для виявлення об'єктів, що використовуються сьогодні, це набір даних PASCAL VOC 2012, набір даних ImageNet і набір даних COCO.

Багато методів виявлення об'єктів на основі глибинного навчання досягли значного успіху. Структура цих методів може бути поділена на два види: фреймворк на основі пропозицій регіонів та фреймворк на основі регресії/класифікації [2].

Фреймворк на основі пропозицій регіонів може бути поділений на два модулі: модуль пропозицій регіонів для локалізації об'єктів та класифікатор для класифікації об'єктів. Перед пропозицією методу Region-based Convolutional Network (RCNN) методи виявлення об'єктів сканували всю картинку за допомогою ковзного вікна як пропозицій регіонів, і завдання класифікації мало проводитись у кожному ковзному вікні. Обчислення великої кількості ковзних вікон призводить до великих обчислювальних навантажень і повільного процесу.

SPP-Net поліпшує метод RCNN. Він обчислює характеристики CNN лише один раз на всьому зображенні. Застосовуючи модуль просторового пірамідального пулінгу (SPP), щоб нормалізувати масштаб отриманих ознак в межах регіональних пропозицій, SPP-Net заощаджує багато часу обчислень. Однак, як і в RCNN, модуль вибіркового пошуку, що використовується в SPP-Net, все ще досить повільний і займає великий об'єм дискового простору.

Регресійно-класифікаційний фреймворк використовує все-в-одному підхід для регресії глобального місцезнаходження об'єкта та його класифікації, на відміну від двомодульного підходу в системі на основі вибіркового пошуку пропозицій регіонів. Цей метод набагато простіший у навчанні та швидший у обробці. До значних робіт в цій системі належать YOLO та SSD.

Метод YOLO бере вхідне зображення і розбиває його на 7*7 підрегіонів. У кожному підрегіоні він використовує повністю зв'язаний шар для регресії двох прямокутних обмежуючих рамок, разом з координатами центроїда і оцінкою впевненості. Ця оцінка впевненості є добутком ймовірності наявності об'єкта та показника перетину над об'єднанням (IoU) між передбаченими рамками та дійсними даними [3].

Однією із задач комп’ютерного зору є розпізнавання пози людини (рис 1.3). Оцінка пози людини та відстеження — це завдання комп’ютерного зору, яке включає виявлення, асоціювання та відстеження семантичних ключових точок Алгоритми комп’ютерного зору виконують обробку зображень у відео в реальному часі, щоб виявити конкретні пози людини. Оцінка пози людини використовує семантичне відстеження ключових точок у відеокадрах. . Прикладами семантичних ключових моментів є «праві плечі», «ліві коліна» [4].

Рисунок 1.3 – Приклад розпізнавання пози людини

Розпізнавання пози в галузі комп'ютерного зору відноситься до завдання визначення положення ключових точок людського тіла на зображеннях або відео. Як показано на рисунку 1.4, шляхом визначення цих ключових точок можна адекватно описати скелетну інформацію людини, що далі може бути використана для інших завдань комп'ютерного зору, таких як визначення активності, відстеження об'єктів, анімація персонажів людини. Ключові точки, які зазвичай визначаються, включають: голову, шию, зап'ястя, плечі, лікті, коліна, щиколотки та стегна [5].

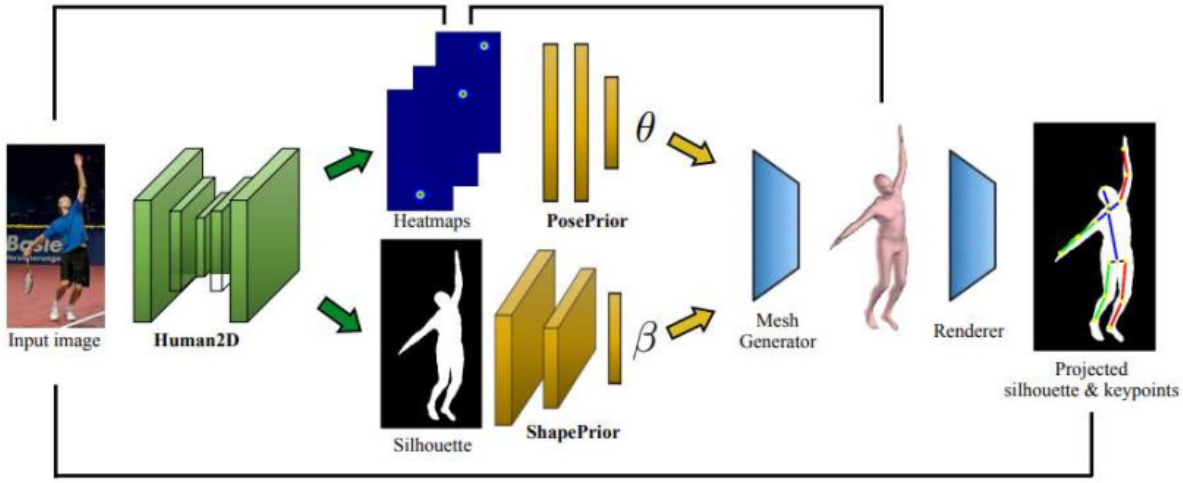


Рисунок 1.4 - Приклад завдання визначення пози.

Виявлення та відстеження людських позицій як широкий спектр варіантів використання, які забезпечують економічну цінність за рахунок автоматизації та оцифрування людської діяльності. Використання звичайних камер є неінвазивним, фізичні датчики не потрібні. Обробка візуальних зображень на пристрої забезпечує обробку зображень із збереженням конфіденційності (без вивантаження даних у хмару). Автоматично виявляти певні події та підозрілу поведінку в програмах безпеки та спостереження. Програма безпеки робочої сили для автономного виявлення небезпечних ситуацій. Що дозволяє заощаджувати витрати на ручне спостереження, отримуючи вищу точність і повторюваність. А також дозволяє проводити

кількісну оцінку виникнення конкретних подій на основі розпізнавання пози [6].

Розпізнавання та моніторинг поз людини за допомогою систем камер є високоефективним методом оцінки ергономічного ризику. Виявлення та відстеження людських позицій як широкий спектр випадків використання забезпечують економічну цінність завдяки автоматизації та оцифровці людської діяльності [7].

Комп’ютерний зір протягом багатьох років використовувався для відстеження рухів спортсменів, щоб покращити їх продуктивність. Наприклад, IBM розробила когнітивну систему коучингу для жіночої національної збірної США з футболу. Система відстежує відеозйомку виступів гравців і забезпечує зворотній зв'язок у режимі реального часу. Відстежуючи рухи спортсменів, тренери можуть визначити сфери, де вони можуть покращитися, а спортсмени можуть побачити, на чому їм потрібно зосередити свою підготовку [7].

Високоефективні системи глибокого навчання працюють шляхом визначення положення тіла та руху суглобів рук і ніг за допомогою ключових точок і оцінки пози людини (рисунок 1. 5).

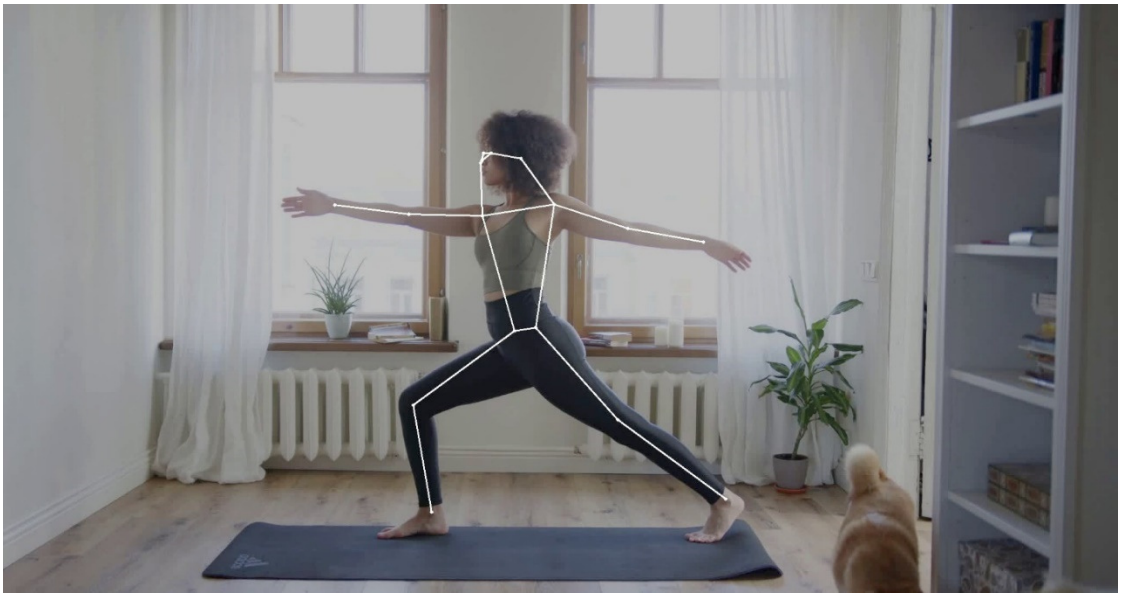


Рисунок 1.5 - Відстеження людського скелета вправ йоги в спорті (створено за допомогою Viso Suite).

Окрім виявлення травм, комп’ютерний зір також використовується в спортивній індустрії для моніторингу процесу реабілітації спортсменів.

Рисунок 1.6 - Приклади передбачених поз на спортивних, професійних і повсякденних фотографіях із набору CrowdPose.

1.3 Огляд існуючих систем розпізнавання пози людини

Існує кілька готових систем розпізнавання пози людини, які можуть бути використані в різних ситуаціях.

Viso.ai – наскрізна платформа комп’ютерного зору Viso Suite, яка значно спрощує комп’ютерне бачення для організацій. Команди можуть використовувати провідні в галузі функції розпізнавання зображень і

розпізнавання людей або об’єктів на основі штучного інтелекту, сегментацію зображення, визначення ключових точок, оцінку пози, розпізнавання облич і проведення аналізу. В роздрібній торгівлі використовують цю технологію для автоматизації створення, доставки та масштабування своїх програм комп’ютерного бачення. На сайті компанії можна отримати демонстрацію для своєї компанії.

Viso Suite [5] — це платформа розробки, яка є гнучкою, розширюваною та розробленою відповідно до вимог підприємства. З її допомогою можна створювати програми, інтегруючи потоки камери з найпотужнішими алгоритмами глибокого навчання. Можна збирати анотації до власних даних, навчати моделі штучного інтелекту або вибрати з понад 55 попередньо навчених моделей штучного інтелекту. На рисунку 1.7 представлено головне вікно Viso Suite.

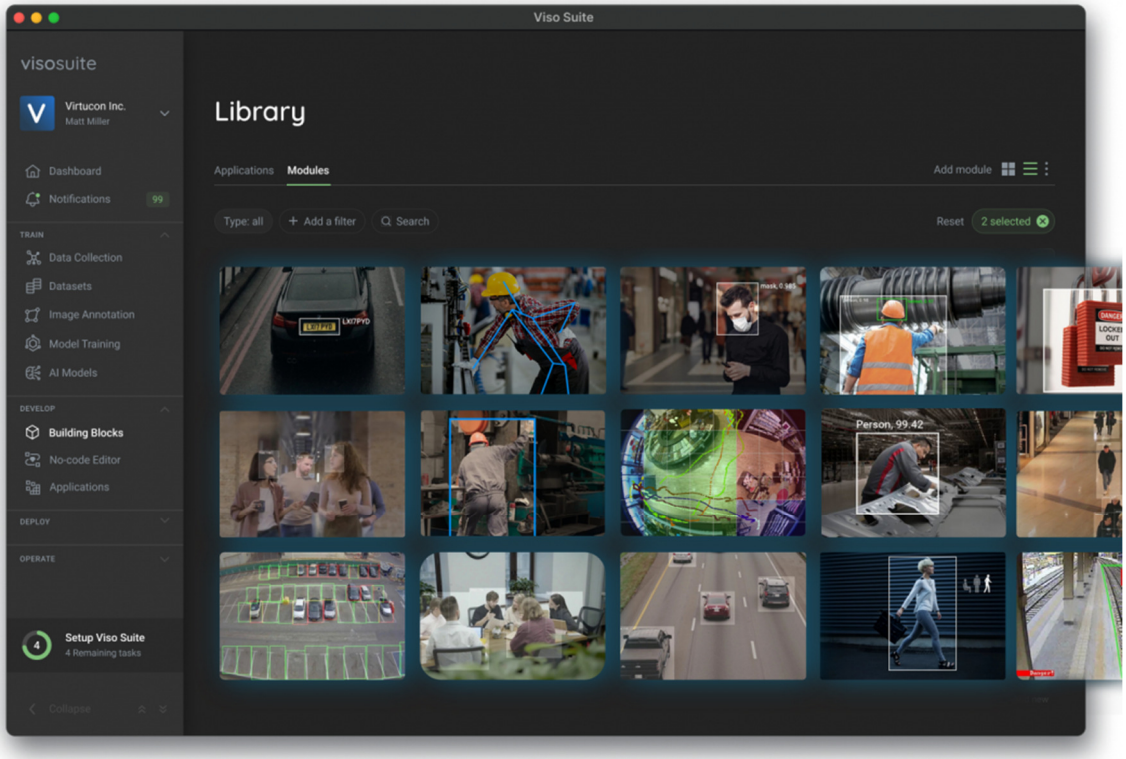


Рисунок 1.7 – Головне вікно Viso Suite

Наука про дані та комп'ютерний зір стають все більш важливими для програм виявлення та моніторингу факторів ризику травм. «Детройт Пістонс» використовує комп'ютерне бачення для відстеження рухів гравців під час ігор і тренувань. Дані про продуктивність використовуються для покращення продуктивності гравців і запобігання травмам. Наприклад, комп'ютерний зір використовується для аналізу ризику розтягнення щиколотки під час занять аеробікою. Травми коліна становлять найбільший відсоток важких травм, пов'язаних зі спортом. Понад 50% випадків пов'язані з розривом передньої хрестоподібної зв'язки, що щороку вражає 200 000 осіб у Сполучених Штатах. Таким чином, дослідники використовували багаторакурсний відеоаналіз, щоб оцінити ризик травми нижньої частини тіла від сценарію спортивних рухів.

Корпорація Strong є провідним постачальником спеціального програмного забезпечення та рішень для машинного навчання, яка складається із науковців та інженерів з машинного навчання та надає багатий міжгалузевий досвід створення та впровадження рішень машинного навчання для вирішення найскладніших проблем організацій. У 2021 команда Strong створила додаток для відстеження бігунів у прямому ефірі. На рисунку 1.8 представлений скрін з програми [7].

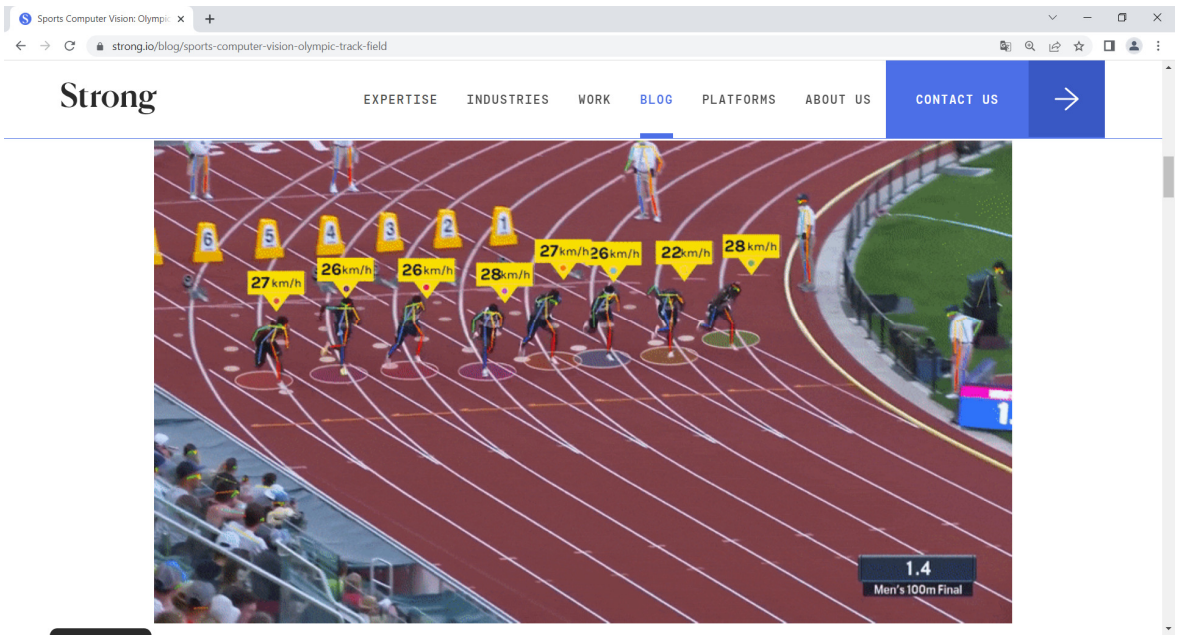


Рисунок 1.8 – Приклад розпізнавання програмою розробленою командою Strong.

SuperAnnotate – платформа, яка заснована в серпні 2018 року двома братами та пропонує рішення на основі комп'ютерного зору в різних галузях: сільське господарство, охорона здоров'я, страхування, спорт, автопілот, робототехніка, аерофотознімки, обробка документів, охорона та спостереження [8]. Однією з програм для спорту є визначення пози та дії (рисунок 1.9).

Їхнє програмне рішення дозволяє відстежувати й оцінювати пози гравців, виявляти удари в таких іграх, як бейсбол і теніс, щоб рекомендувати рухи та покращувати досвід тренувань. Аналізуючи кожен міні-рух гравців, визначення пози людини надає моделям статистику на основі даних, на основі якої можна робити прогнози. А на рисунку 1.9 представлено результати розпізнавання пози гравця у теніс.

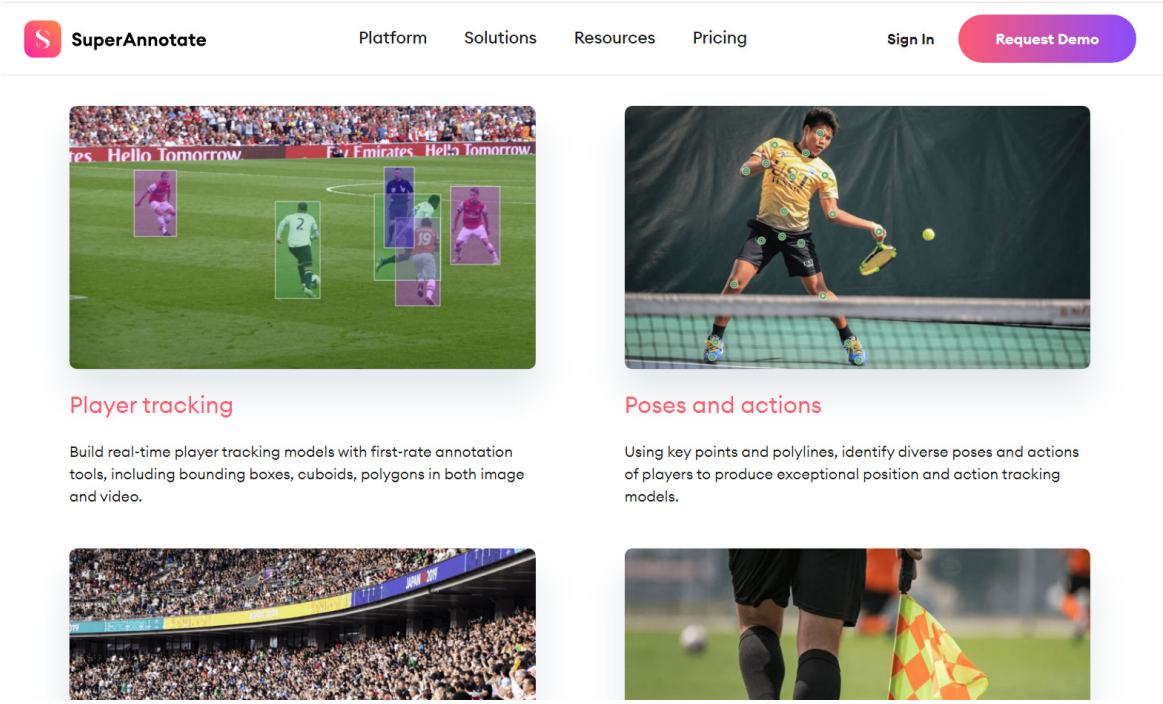


Рисунок 1.9 – Пропозиції SuperAnnotate для спорту

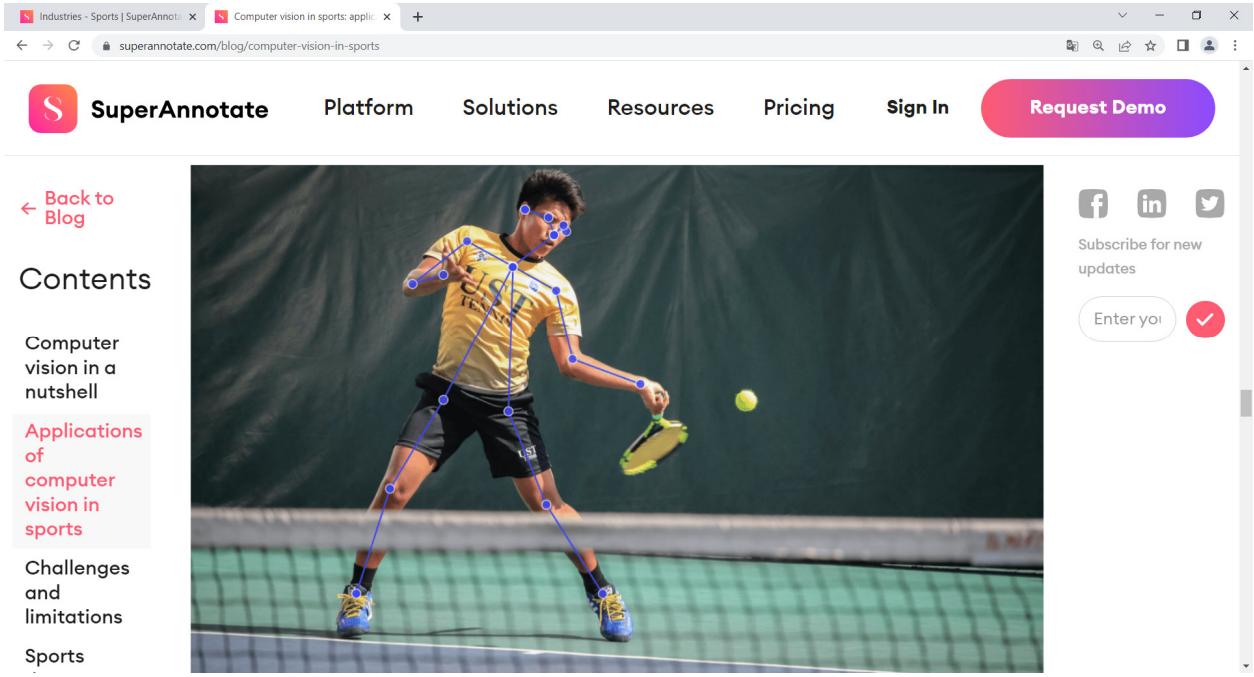


Рисунок 1.10 – Приклад розпізнавання пози гравця у теніс.

1.4 Постанова задачі дослідження

Мета даної роботи полягає у створенні програмного модуля, який здатний виявляти та розпізнавати пози людини на зображеннях. Розроблений модуль буде здатний проводити аналіз структури людського тіла, визначати положення та орієнтацію окремих його частин, таких як руки, ноги, голова та інші. Це відкриє можливості отримання цінної інформації про стан людини, її рухи та позу.

Для вирішення задачі створення програмного модуля розпізнавання пози людини необхідно виконати наступні кроки:

– провести огляд систем комп’ютерного зору;

– провести огляд систем розпізнавання пози людини;

– провести аналіз методів розпізнавання пози;

– здійснити аналіз існуючих методів розпізнавання пози;

– розробити алгоритмічно-математичну модель модуля розпізнавання пози людини;

– розробити модуль розпізнавання пози людини;

– провести експериментальні дослідження.

Створення програмного модуля розпізнавання пози людини буде корисним у різних сферах, включаючи системи відеоспостереження, розробку ігор та віртуальної реальності, спорті, а також медичних дослідженнях та реабілітації.

РОЗДІЛ 2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

2.1 Структура модулів розпізнавання поз людини

Методи, які використовуються для завдання визначення пози, головним чином базуються на згорткових нейронних мережах (CNN). Основними аспектами методу CNN є проектування архітектури мережі, обробка даних для навчання та оцінки нейронної мережі, а також методи навчання CNN, такі як вибір функцій втрати та методи оптимізації [9].

Рисунок 2.1 показує структуру одгого з підходів для вирішення поставленої задачі. Сині блоки посередині є основною конвеєрною системою визначення 3D пози, зелений блок - це метод додаткової обробки даних, застосовуваний на різних етапах основного конвеєра. Червоний блок це додаткова мережа hourglass для покращення характеристик.

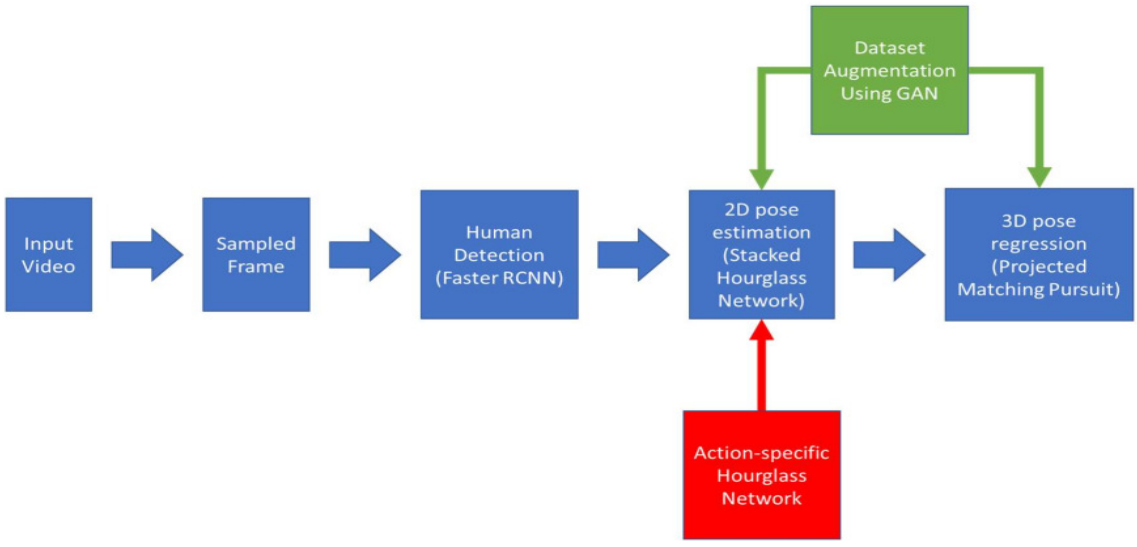


Рисунок 2.1 – Структура підходу визначення пози.

Традиційні методи визначення пози використовують людський шаблон для співставлення зображень. Людський шаблон базується на попередній інформації. Пікторіальні структури, як один з традиційних методів визначення пози, складаються з двох частин: унарний шаблон, який представляє окремі частини тіла людини, та парні пружини, які базуються на попередній інформації для обмеження просторового відношення моделі людини [10]. Шаблон Пікторіальних структур показано на рисунку 2.2.

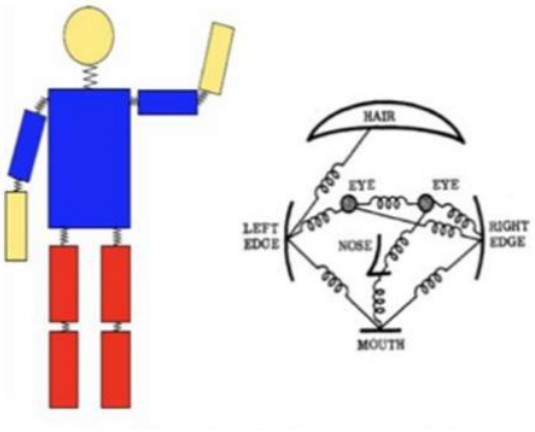


Рисунок 2.2 – Шаблон пікторіальних структур.

Проте цей тип методу не може охопити різноманіття та складність поз людини. Коли в галузі визначення пози вводиться машинне навчання, воно стає універсальним методом розробки. В загальному розглядають наступні методи визначення поз на основі машинного навчання:

Визначення 2D пози однієї особи починається з використання згорткової нейронної мережі для регресії координат ключових точок тіла людини з одного зображення, наприклад, у методі DeepPose, який використовує грубий до дрібний підхід для безпосереднього виведення координат. Також широко застосовується метод теплових карт (Heatmaps Net). Різниця між методами регресії координат та методами регресії теплових карт показана на рисунку 2.3.

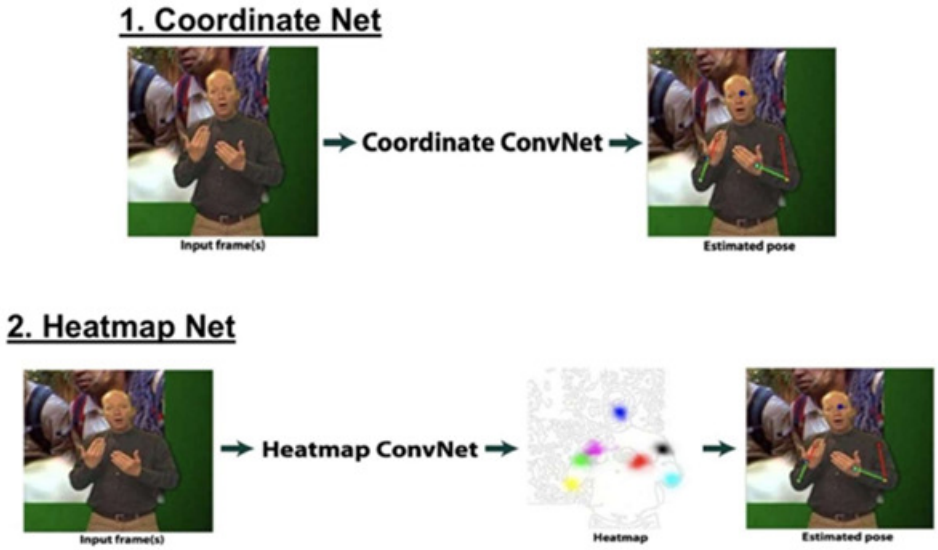


Рисунок 2.3 – Використання теплових карт методом Heatmap Net.

На відміну від методів Coordinates Net, методи Heatmap Net регресують набір теплових карт, які представляють ймовірнісну карту виявлення сполучень. У цих теплових картах кожне значення пікселя показує ймовірність того, що конкретне сполучення виявлено на цьому пікселі. Тому, природно, чим ближче значення пікселя до 1, тим більш ймовірно, що сполучення знаходиться на цьому пікселі, і навпаки. Для тренування нейронної мережі цього методу регресії теплових карт, як правило, використовується набір гаусових теплових карт з відомими позиціями сполучень для обчислення функції втрат. У такому типі тренування надається значно більше інформації, ніж у тренуванні координатних мереж, що прискорює процес тренування і також покращує точність визначення [11].

Convolutional Pose Machines (CPM) і Stacked Hourglass Network є двома важливими еволюціями методів теплових карт. CPM використовує багатоетапну проміжну стратегію навчання, як показано на рисунку 2.4.

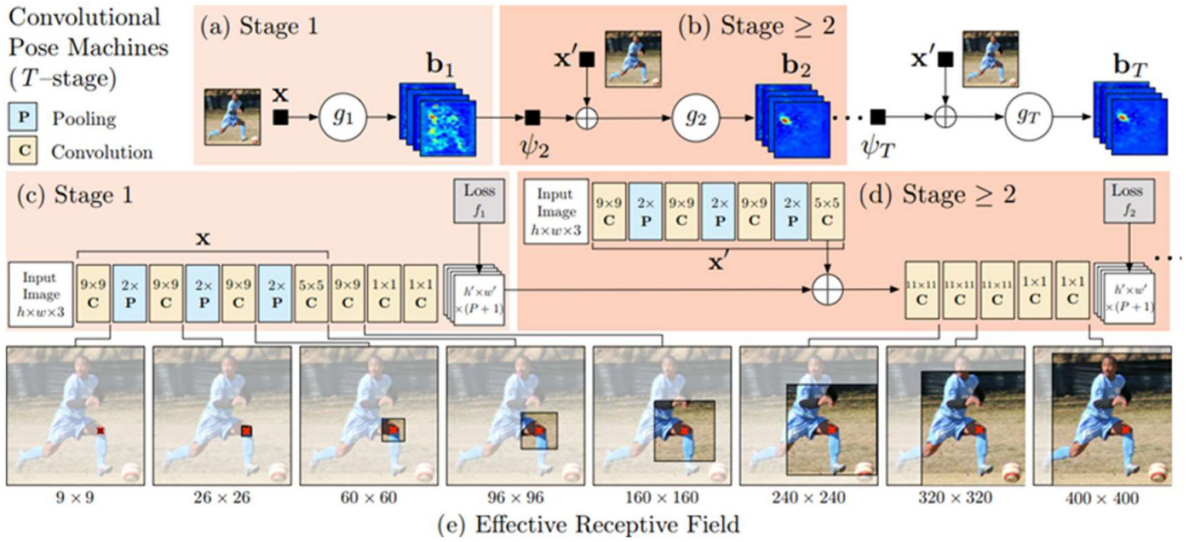


Рисунок 2.4 – Архітектура та рецептивні поля CPM.

На першому етапі відтворюються тільки RGB-зображення для регресії теплових карт. На наступних етапах до вхідних даних додаються теплові карти, отримані на попередньому етапі, та оригінальні RGB-зображення. На кожному етапі обчислюється проміжна функція втрат, що допомагає оновити глибоку згорткову мережу. Крім того, як показано на вставці (e), ефективне рецептивне поле на зображенні збільшується, що дозволяє мережі вивчити довгострокові просторові обмеження між різними ключовими сполученнями та дозволяє моделі вирішувати проблему перешкод. Однак через багатоетапну архітектуру цей метод відносно повільний [12].

Багатоособова оцінка 2D пози працює з зображеннями, на яких зображено більше двох осіб. Вона повинна не тільки визначити ключові точки на зображенні, але й встановити, яка ключова точка належить кожній особі. Існують два типи методів для вирішення цієї проблеми: методи зверху-вниз і методи знизу-вгору.

Методи зверху-вниз розбивають задачу оцінки пози багатоособових на два підзадачі. По-перше, вони виявляють всіх людей на зображенні і локалізують їх за допомогою методу виявлення об'єктів. По-друге, для кожної виявленої особи застосовується оцінка пози одної людини для кінцевого передбачення. До таких підходів входять RMPE, Mask RCNN, G-RMI,

Cascaded Pyramid. Недолік цих підходів полягає в тому, що помилкове виявлення людей може вплинути на точність кінцевих результатів пози.

Методи знизу-вгору використовують частково засновану структуру. Спочатку вони локалізують всі ключові точки на зображенні, а потім визначають, які точки належать кожній особі. OpenPose (Zhe Cao, et al., 2017), Associative Embedding (Newell et al., 2017) і другі методи належать до цього підходу. Перевага цих підходів полягає в тому, що вони здатні виявляти ключові точки навіть у випадках, коли особи перекриваються або знаходяться в складних позах [13].

Двоетапні підходи розбивають завдання оцінки 3D пози на дві підзадачі: оцінку 2D пози і регресію глибини. Структура фреймворку двохетапного підходу представлена на рисунку 2.5.

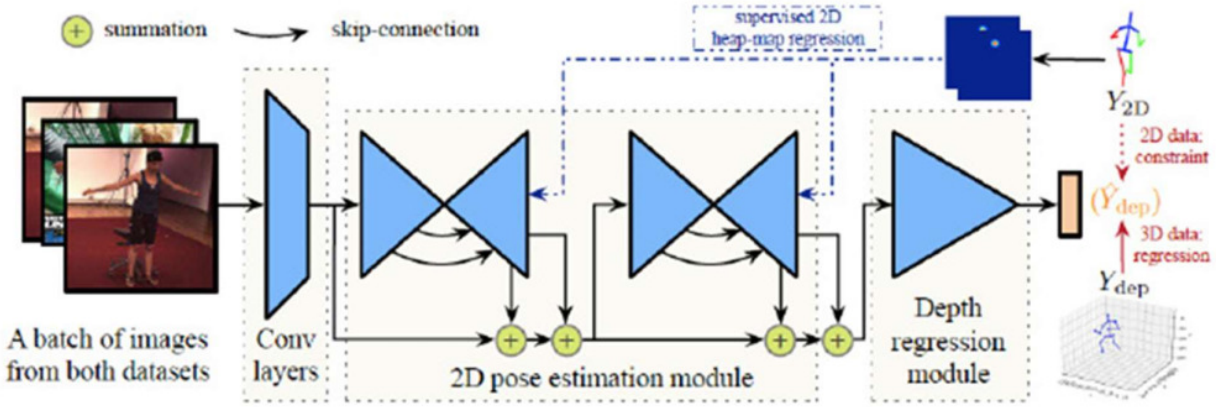


Рисунок 2.5 – Структура фреймворку двохетапного підходу.

Як показано на рисунку, модуль оцінки 2D пози навчається як за допомогою 3D даних з наборів даних Human3.6M, так і за допомогою інших наборів даних з 2D позами. Модуль регресії глибини 3D навчається виключно на 3D даних поз. Цей дизайн дозволяє моделі використовувати дані з різних джерел для оцінки 3D поз без обмежень.

Розширенням даного фреймворку є багатоджерельний дискримінатор для навчання модуля регресії глибини для отримання більш антропометрично

валідних поз. Джерелами для дискримінатора є 2D-теплові карти, картинки глибини, геометричні описи та RGB-зображення. Багатоджерельний дискримінатор показано на рисунку 2.6.

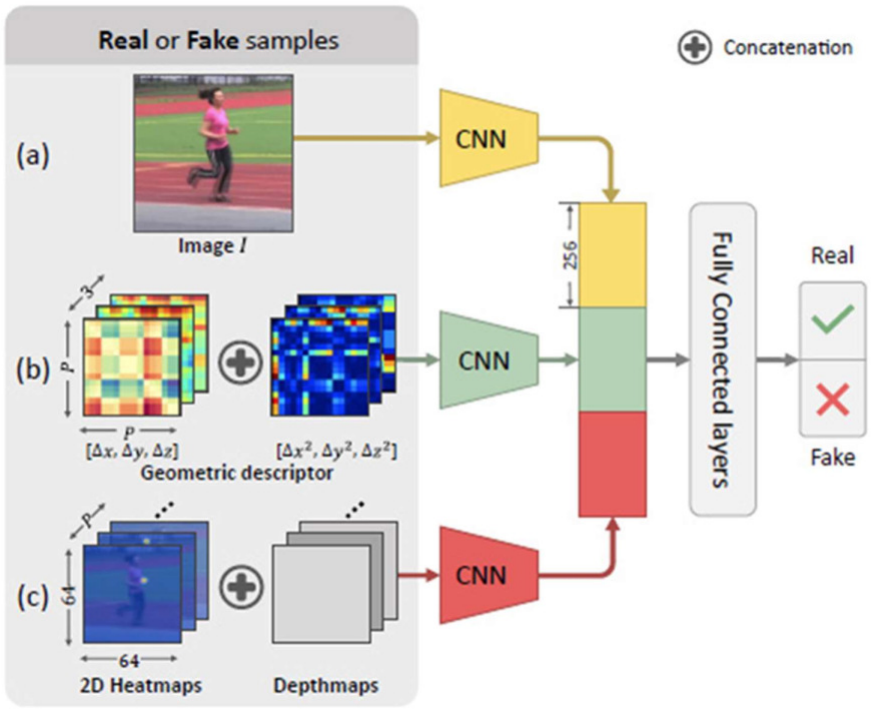


Рисунок 2.6 - Структура багатоджерельного дискримінатора.

Coarse-to-Fine (Грубе до дрібного) об'ємне передбачення є ще одним підходом з двома етапами. Він використовує об'ємне представлення для визначення пози людини в тривимірному просторі. Для кожного з'єднання він дискретизує простір навколо об'єкта і застосовує згорткову мережу для регресії ймовірності на кожному вокселі. Великою перевагою об'ємного представлення є те, що воно перетворює високоякісну нелінійну проблему безпосередньої регресії тривимірних координат в більш управліну форму передбачення в дискретизованому просторі. В цьому методі також використовується стратегія від грубого до дрібного.

Для оцінки 3D-поз у відео був запропонований метод Detect-and-Track для оцінювання тривимірних поз з використанням 3D Mask R-CNN. Він замінює 2D згорткові шари в Mask R-CNN на 3D згорткові шари для обробки

відео. Отже, RPN в 2D Mask R-CNN стає 3D мережею пропозицій труб (TPN) для виявлення пропозицій просторово-часових труб. Потім ці труби використовуються для отримання тривимірних ознак за допомогою методу RoIAlign. Ці ознаки подаються на вхід класифікаторної мережі для визначення їх категорій і також на вхід мережі оцінки поз для регресії пози з'єднань у кожному кадрі відео. Пози з'єднань у кожному кадрі пов'язуються глибокою рекурентною нейромережею (RNN), Таким чином, метод Detect-and-Track може визначити, які з'єднання належать кожній людині [14].

Структура програмного модуля розпізнавання пози людини для системи комп'ютерного зору представлена в додатку А . На вхід подається зображення, на котрому виділяються люди. З кожної виділеної людини створюється спеціальна теплова карта. По кожній карті здійснюється прогнозування, застосовується метод м'якого максимуму (soft-argmax) та масштабування за допомогою константного множника, в кінці відбувається реконструкція абсолютної пози за допомогою диференційованого модуля.

2.2 Алгоритмічне забезпечення

Серед найпопулярніших моделей машинного навчання (ML) котрі використовуються в системах комп'ютерного зору для оцінки пози людини виділяють: OmniPose, RSN, DarkPose, OpenPose, AlphaPose (RMPE), DeepCut, MediaPipe і HRNet, та ін [15]. Проте постійно з’являються і нові моделі, оскільки обчислювальна потужність і алгоритми ML/AI стають все більш точними, а оскільки науковці постійно вдосконалюють їх.

1. OmniPose - це платформа для наскрізного оцінювання пози кількох людей, яку можна навчити за один прохід. Він використовує гнучкий метод водоспаду з архітектурою, яка використовує багатомасштабні представлення функцій, які підвищують точність і одночасно зменшують потребу в постобробці. Включається контекстна інформація зі спільною

локалізацією за допомогою модуляції теплової карти Гауса. OmniPose розроблено для досягнення найсучасніших результатів, особливо в поєднанні з HRNet (докладніше про це нижче).

2. Мережа залишкових кроків (Residual Steps Network, RSN) - це інноваційний метод, який «ефективно об'єднує об'єкти однакового просторового розміру (внутрішньорівневі об'єкти) для отримання делікатних локальних представлень, які зберігають багату просторову інформацію низького рівня та призводять до точної локалізації ключових точок». Цей підхід використовує Pose Refine Machine (PRM), який врівноважує компроміс між «локальними та глобальними представленнями у вихідних функціях», таким чином уточнюючи функції ключових точок. RSN виграв COCO Keypoint Challenge 2019 і архівує найсучасніші результати за тестами COCO та MPII.

3. DARKPose — Представлення опорних координат із урахуванням розподілу (DARK) — це новий підхід до покращення традиційних теплових карт. DARKPose декодує «прогнозовані теплові карти в остаточні координати з'єднань у вихідному просторі зображення» та реалізує «більш принциповий метод декодування з урахуванням розподілу». Створюються точніші розподіли теплової карти, що покращує результати моделі оцінки пози людини.

4. OpenPoseце популярна модель машинного навчання знизу вгору для відстеження, оцінки та анотацій у режимі реального часу за участю кількох осіб. Це алгоритм із відкритим вихідним кодом, який ідеально підходить для визначення ключових точок обличчя, тіла, ніг і рук.

5. OpenPose — це API, який легко інтегрується з широким спектром камер і систем відеоспостереження, а полегшена версія ідеально підходить для пристроїв Edge.

6. AlphaPose (RMPE) - Регіональна оцінка пози кількох осіб (Regional Multi-Person Pose Estimation - RMPE) це модель машинного навчання зверху вниз для анотації оцінки пози. Він точніше визначає пози та моделі

рухів людей у межах обмежувальних рамок. Архітектура застосовується як до пози однієї людини, так і для кількох осіб на зображеннях і відео.

7. DeepCut - це ще один висхідний підхід для виявлення кількох людей і точного визначення суглобів і приблизного руху цих суглобів на зображенні чи відео. Він був розроблений для визначення поз і рухів кількох людей і часто використовується в спортивному секторі.

8. MediaPipe - це «міжплатформне, настроюване рішення ML з відкритим кодом для живих і потокових медіа», розроблене та підтримуване Google. MediaPipe — це потужна модель машинного навчання, розроблена для визначення обличчя, рук, поз, відстеження очей у реальному часі та комплексного використання. Google надає численніdокладні випадки використання в блогах Google AI і Google Developers, і навіть кілька зустрічей MediaPipe відбулися в 2019 і 2020 роках.

9. Мережа високої роздільної здатності (High-Resolution Net, HRNet) — це нейронна мережа для оцінки пози, створена для більш точного пошуку ключових точок (суглобів людини) на зображенні чи відео. HRNet підтримує представлення високої роздільної здатності для оцінки поз і рухів людини порівняно з іншими алгоритмічними моделями. Отже, це корисна модель машинного навчання під час коментування відео, знятих для телевізійних спортивних ігор.

На основі підходу MeTRAbs [16] запропонований алгоритм роботи (добаток Б) програмного модуля розпізнавання пози людини для системи комп'ютерного зору. Він використовує готові інструменти для виділення людей на зображенні (YOLOv3), і для кожної із виділеної людини створюється спеціальна теплова карта, яка показує, де знаходяться різні частини людського тіла на зображенні. Застосовуючи певні обчислення до цих карт отримуються точні координати частин людського тіла .

У метрично-масштабних теплових картах кожна ключова точка представлена у вигляді теплового значення або інтенсивності, яка вказує на ймовірність присутності цієї точки в даних координатах зображення. Інтенсивність може бути представлена числом від 0 до 1, де ближче до 1 вказує на більшу ймовірність присутності ключової точки. Такі карти дозволяють точно виявляти і відстежувати позу людини на зображенні або відео, що є важливим для багатьох додатків, таких як системи відеоспостереження, віртуальна реальність, фітнес-додатки та багато інших. Цей метод дозволяє отримувати інформацію про положення та орієнтацію різних частин тіла людини і використовувати ці дані для подальшого аналізу та обробки.

Прогнозування об'ємних теплових карт здійснюється за допомогою готової повністю конволюційної нейронної мережі. Застосування м'якого максимуму (soft-argmax) до цих теплових карт та масштабування за допомогою константного множника, незалежного від зображення, дає координати суглобів в метричному просторі з точністю до зсуву.

Застосування soft-argmax до цих теплових карт і масштабування за допомогою постійного фактора, незалежного від зображення, дає спільні координати в метричному просторі з точністю до трансляції.

Організовуються відносні кореневі втрати L1. Зосереджуючись на простоті, параметри, які можна вивчати, не вводяться за межі стандартної магістралі, за винятком одного 1×1 згортка. За бажанням, якщо потрібна абсолютна (некоренева відносна) оцінка пози, розширення MeTRAbs також оцінює класичні 2D-теплові карти простору зображень за допомогою іншої 1×1 згортки.

Потім відбувається реконструкція абсолютної пози за допомогою диференційованого модуля реконструкції. Він використовує лінійне формулюванні найменших квадратів, отриманому з моделі камери-обскури. Контроль застосовується як на виходах окремих головок прогнозування, так і на кінцевому комбінованому виході.

Для 2D-мічених прикладів коренева відносна втрата замінюється масштабною та трансляційно-інваріантною 2D втратою та абсолютна втрата 3D не використовується.

Як це зазвичай буває в підходах на основі теплових карт, ми застосовуємо повністю згорточну магістральну мережу з ефективним кроком s для створення масиву з d. Джпросторові вихідні канали. Тут d — кількість дінок дискретизації вздовж осі глибини обсягу передбачення. Потім ми розбиваємо масив уздовж осі каналу на J об'ємів, кожен має форму (ш/ш) x (в/с) x г. Тривимірний просторовий softmax застосовується до кожного з них, що призводить до активацій об'ємної теплової карти V(j)ЄR(w/s)x(h/s)xd.

До цього моменту процес подібний до інших підходів об'ємної теплової карти [13], [14]. Різниця полягає в тому, як осі теплової карти інтерпретуються для отримання координат у метричному масштабі. Зокрема, тривимірні стикові координати декодуються за допомогою soft-argmax із фіксованими коефіцієнтами масштабування:

$$\begin{bmatrix} \Delta X_j \\ \Delta Y_j \\ \Delta Z_j \end{bmatrix} = \sum_{p,q,r} V^{(j)}_{p,q,r} \cdot \begin{bmatrix} p \cdot s/w \cdot W \\ q \cdot s/h \cdot H \\ r \cdot 1/d \cdot D \end{bmatrix},$$

де p, q, r є цілими індексами об'єму від нуля, W,H,D — це фіксовані межі метричної ширини, висоти та глибини повного прогнозованого об'єму.

Задані розміри 2,2 метра дозволяють знімати людей звичайного зросту навіть у розтягнутому стані. Залежно від логіки кроку, рівняння. 1 потрібно трохи відрегулювати, наприклад, розмір гучності може змінитися з більш щільним кроком (рис. 2.7). Остаточний відносний кореневий прогноз отримують шляхом віднімання прогнозованих кореневих координат із усіх положень з'єднання. Контроль застосовується до цих кореневих відносних координат. Це означає, що положення передбаченого кореневого суглоба в обсязі явно не контролюється, і мережа може розмістити скелет будь-де в межах передбаченого обсягу. Градієнти поширюються в зворотному напрямку

через операцію віднімання кореневого з’єднання. Немає зворотної проекції на основі калібрування камери, для цього відносного кореневого прогнозу не потрібне змінення масштабу на основі розміру кістки чи скелета. Мережа навчена виконувати ці операції неявно в межах магістралі.

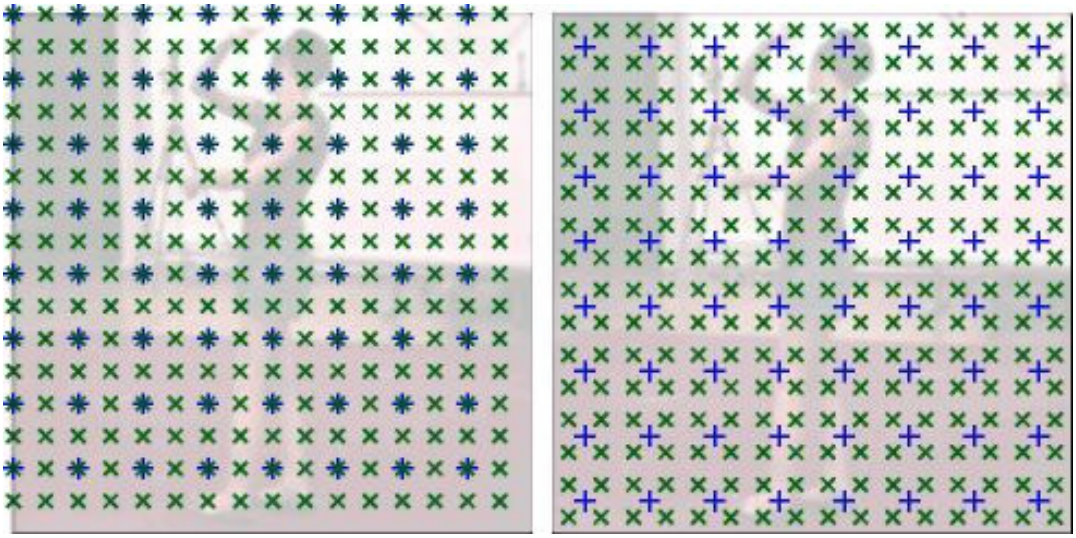


Рисунок 2.7 – Рецептивні польові центри вихідних нейронів

На рисунку представлені рецептивні польові центри вихідних нейронів у покроковому FCN, що працює на зображенні 256x256 (+: крок 32, x: крок 16). ліворуч: звичайна логіка кроків, де верхній лівий результат зберігається на блок 2x2. Щільні кроки зміщують щільність вибірки донизу та праворуч у прикордонних областях, праворуч: змінюючи логіку кроків на останньому шарі кроків (тобто отриманий нижній правий результат замість верхнього лівого), зразки центруються, а підвищена щільність кроку розподіляється рівномірно.

На рисунку 2.8 показано, як розпізнавана позиція людини залишається стійкою до змін масштабування та обрізання. Відображено метричні теплові карти для трьох суглобів з повністю визначеною скелетною структурою для порівняння. Прогнозований скелет приблизно незмінний при зміні масштабування та обрізання. Оскільки метричний розмір людини залишається

незмінним при зміні масштабування зображення, базова модель навчається виводити теплові карти зі схожим центром мас, незалежно від масштабу зображення. Варто зазначити, що теплові карти не вирівнюються з простором зображення, і це зроблено навмисно (широкі піки є результатом навчання моделі на низькій роздільній здатності теплових карт розміром 8x8 пікселів).

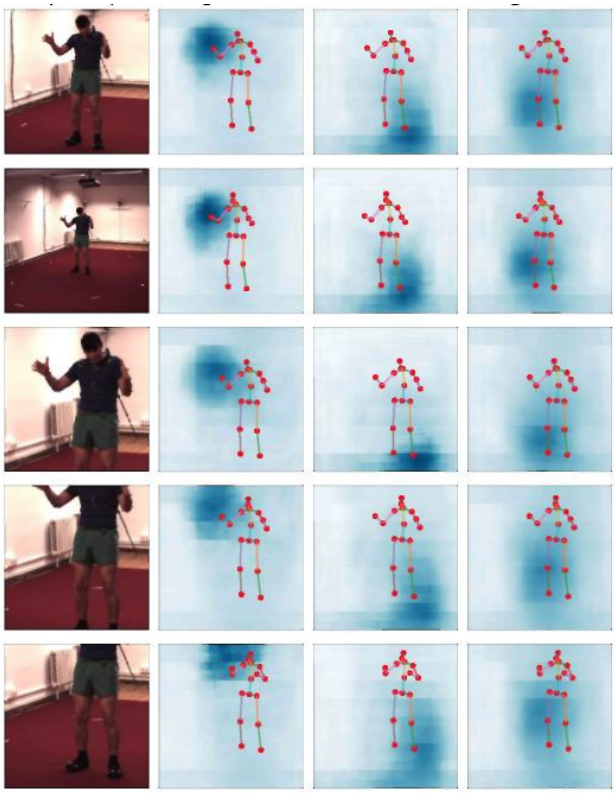


Рисунок 2.8 – Стійкість до змін масштабування та обрізання.

Щоб збалансувати втрати, обчислені на 3D- і 2D-анотованих прикладах, ми використовуємо фіксований ваговий коефіцієнт $\lambda = 0.1$, налаштований на окремому наборі перевірки Human3.6M, що дає загальні втрати як

$$\mathcal{L} = \mathcal{L}_{\text{ann3D}} + \lambda \mathcal{L}_{\text{ann2D}}.$$

Далі ініціалізується мережа за допомогою попередньо навчених вагових коефіцієнтів ImageNet і використанням оптимізатора Адама з розпадом ваги і розміром пакета 64. Мережа навчена виводити повні скелети в тому самому

метричному масштабі, незалежно від масштабування та скорочення зображення.

Як видно з рис. 2.8, модель здатна досягти приблизно масштабу зображення та інваріантних прогнозів. Зокрема, ми бачимо, що результат soft-argmax не обов'язково знаходиться в середині найвиразнішого піку теплової карти. Оскільки soft-argmax визначає центр мас теплової карти, впливають навіть віддалені значення теплової карти. Інтуїтивно зрозуміло, що це дозволяє мережі переміщати результат передбачення в різні місця на тепловій карті, додаючи врівноважувальні коригуючі ваги, наприклад, з боків зображення або в центрі людини..

Прогнозовані теплові карти будуються в метричному масштабі для 3 суглобів із повним скелетом soft-argmax. Прогнозований скелет є приблизно інваріантним до зміни масштабу та скорочення. Оскільки метричний розмір людини не змінюється при масштабуванні зображення, магістраль навчається виводити теплові карти з подібним центром маси, незалежно від масштабу зображення.

Також важливим аспектом є калібрування камери та 3D-реконструкція. Функції в цьому розділі використовують так звану модель пінхол-камери (камери-обскури, камера без об'єктива, роль якого виконує малий отвір). У цій моделі вид сцени формується шляхом проектування 3D-точок на площину зображення за допомогою перспективного перетворення.

$$s\, m' = A[R|t]M'$$

або

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

де:

– (X, Y, Z) є координати тривимірної точки у світовому координатному просторі

– (u, v) – координати точки проекції в пікселях

– A - матриця камери, або матриця внутрішніх параметрів

– (cx, cy) - головна точка, яка зазвичай знаходиться в центрі зображення

– fx, fy — фокусні відстані, виражені в одиницях пікселів.

Таким чином, якщо зображення з камери масштабується на коефіцієнт, усі ці параметри слід масштабувати (відповідно помножити/ділити) на той самий коефіцієнт. Матриця внутрішніх параметрів не залежить від сцени, що переглядається. Отже, після оцінки його можна використовувати повторно, якщо фокусна відстань фіксована (у випадку об’єктива зі змінною фокусною відстанню). Спільна матриця обертання-трансляції $[R|t]$ називається матрицею зовнішніх параметрів. Вона використовується для опису руху камери навколо статичної сцени або, навпаки, жорсткого руху об'єкта перед нерухомою камерою. Тобто, $[R|t]$ переводить координати точки (X, Y, Z) до системи координат, фіксованої відносно камери. Наведене вище перетворення еквівалентне наступному (коли $z \neq 0$):

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z$$
$$y' = y/z$$
$$u = f_x * x' + c_x$$
$$v = f_y * y' + c_y$$

На рисунку 2. 9 показано модель пінхол-камери. Справжні лінзи зазвичай мають певні спотворення, переважно радіальні та невеликі тангенціальні. Таким чином, наведена вище модель розширена як:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z$$

$$y' = y/z$$

$$x'' = x' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + 2p_1 x' y' + p_2(r^2+2x'^2)$$

$$y'' = y' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + p_1(r^2+2y'^2) + 2p_2 x' y'$$

$$\text{where} \quad r^2 = x'^2 + y'^2$$

$$u = f_x * x'' + c_x$$

$$v = f_y * y'' + c_y$$

де, k_1, k_2, k_3, k_4, k_5, і k_6– коефіцієнти радіальних спотворень. p_1 і p_2– коефіцієнти тангенціальних спотворень. Коефіцієнти вищого порядку не враховуються в OpenCV.

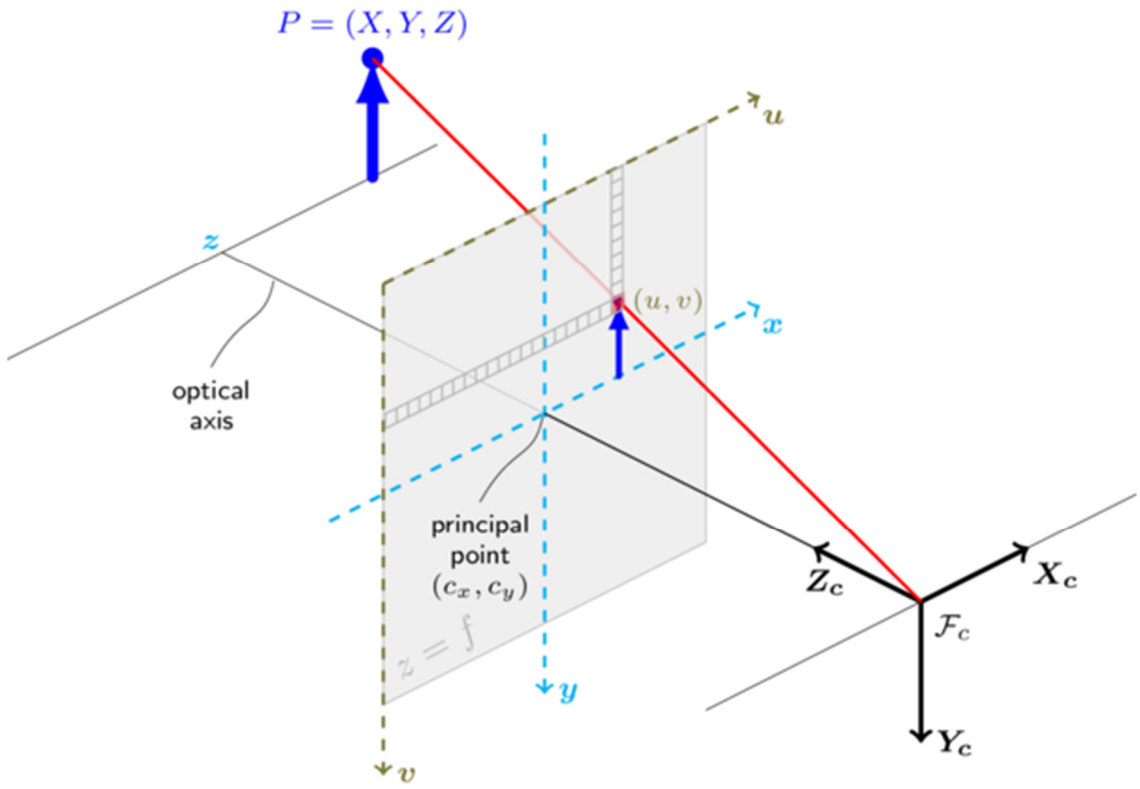


Рисунок 2.9 – Модель пінхол-камери.

На рисунку 2.10 показано два поширені типи радіальних спотворень: бочкоподібне спотворення (зазвичай $k_1 > 0$ і подушкоподібне спотворення (як правило $k_1 < 0$).

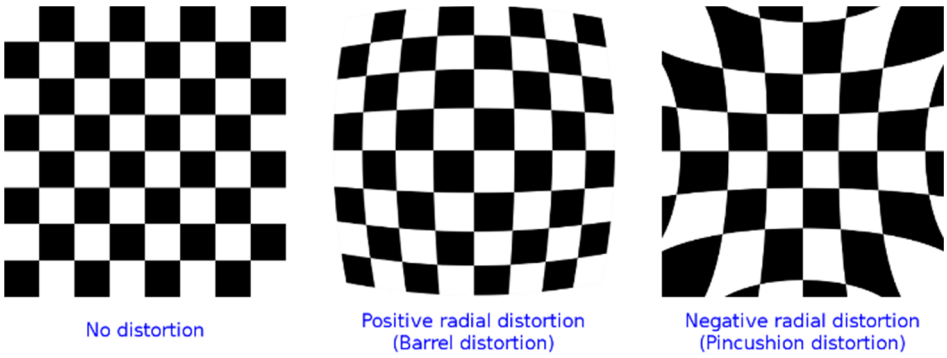


Рисунок 2.10 - типи радіальних спотворень.

У наведених нижче функціях коефіцієнти передаються або повертаються як вектор:

$$(k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6]])$$

Тобто, якщо вектор містить чотири елементи, це означає, що $k_3 = 0$. Коефіцієнти спотворення не залежать від сцени, що переглядається. Таким чином, вони також належать до власних параметрів камери. І вони залишаються незмінними незалежно від роздільної здатності знятого зображення.

Якщо, наприклад, камеру було відкалібровано за зображеннями 320 x 240 роздільної здатності, можна використовувати абсолютно однакові коефіцієнти спотворення 640 x 480 зображення з тієї ж камери f_x,f_y,c_x, іc_y потрібно відповідним чином масштабувати.

Наведені нижче функції використовують наведену вище модель для виконання наступних дій:

- Проект 3D вказує на площину зображення із заданими внутрішніми та зовнішніми параметрами.
- Обчисліть зовнішні параметри з урахуванням внутрішніх параметрів, кількох 3D-точок та їхніх проекцій.
- Оцініть внутрішні та зовнішні параметри камери за кількома видами відомого шаблону калібрування (кожне зображення описується декількома відповідностями 3D-2D).

- Оцініть відносне положення та орієнтацію «голів» стереокамери та обчисліть випрямляючу трансформацію, яка робить оптичні осі камери паралельними.

2.3 Інформаційне забезпечення

Інформаційне забезпечення програмного модуля розпізнавання пози людини системою комп'ютерного зору забезпечується набором вхідних даних, відповідних алгоритмів і моделей розпізнавання, бази даних різного типу зразків, різними способами представлення вихідної інформації.

Безкоштовні набори даних із відкритим кодом (зображення, відео, анотації та мітки) є швидким і бюджетним способом навчити алгоритмічно згенеровану модель. Хоча вони і не завжди представляють найширший діапазон даних необхідних для точного навчання, проте забезпечують гідний рівень для подальшого їх використання дослідниками та містять уже застосовані анотації та мітки. Крім того, при потребі отримання більшої кількості даних, можна використати кілька наборів даних із відкритим кодом. Серед таких можна виділити MPII Human Pose, 3D Poses in the Wild, Leeds Sports Pose [11].

Набір даних MPII Human Pose (рис. 2.11)— це «сучасний еталон для оцінки артикульованої оцінки пози людини». Він містить понад 25 000 зображень із 40 000 із анотаціями, що охоплюють 410 рухів і дій людини. Зображення були витягнуті з відео YouTube, і набір даних включає кадри без анотацій, що передують і слідують за рухом у межах анотованих кадрів. Дані тестового набору включають багатші дані анотацій, оклюзії рухів тіла та 3D-орієнтацію голови та тулуба.

Рисунок 2.11 - Набір даних MPII Human Pose

Набір даних «3D Poses in the Wild» (3DPW, рис. 2.12) використовував IMU та відеоматеріали з камери телефону, що рухається, щоб точно зафіксувати рухи людей у громадських місцях. Цей набір даних включає 60 відеопослідовностей, 3D- і 2D-пози, 3D-сканування тіла та моделі людей, які можна змінювати в позі та формувати.

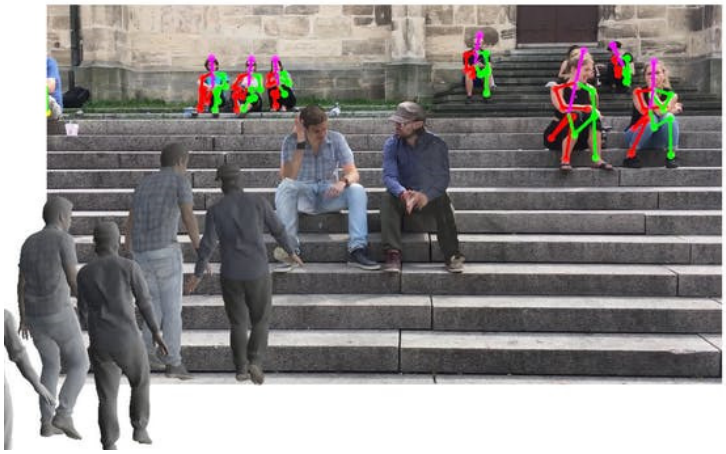


Рисунок 2.12 - Набір даних «3D Poses in the Wild»

Розширений набір даних «Leeds Sports Pose» (LSPe) містить 10 000 зображень, пов’язаних зі спортом, де кожне зображення містить до 14 анотацій спільного розташування. Це корисна потенційна відправна точка для тих, хто тренується за моделлю комп’ютерного зору на основі спортивних рухів.

DensePose-COCO — це набір даних «Оцінка щільної пози людини в дикій природі», що містить 50 000 анотованих вручну COCO зображеннях на основі (Загальні об'єкти в контексті). Анотації в цьому наборі даних вирівнюють щільні «відповідності від 2D-зображень до зображень людського

тіла на поверхні» за допомогою Модель SMPLiCЮРРЕАЛІСТ текстури для анотацій і маркування вручну (рис. 2.13).

Рисунок 2.13 - набір даних DensePose-COCO

AMASS — це велика та різноманітна база даних про рух людини, яка об’єднує 15 різних наборів даних mocap на основі оптичних маркерів, представляючи їх у загальній структурі та параметризації. Він містить 3D- і 4D-сканування тіла, 40 годин даних про рух, 300 людей і понад 11 000 людських рухів і рухів, усі ретельно анотовані.

Набори даних оцінки пози людини VGG включають низку анотованих відео та кадрів із YouTube, BBC та ChatLearn. Він містить понад 70 відео, що містять мільйони кадрів, із вручну анотованими мітками, що охоплюють широкий спектр поз верхньої частини тіла людини.

SURREAL — це «великомасштабний набір даних про людину для генерування глибини, частин тіла, оптичного потоку, 2D/3D-пози, нормалей поверхні, правдивості для відеовходу RGB». Тепер він містить дані оптичного потоку.

Рисунок 2.14 - набір даних про людину SURREAL

Набір даних Panoptic Studio — це «масова багаторакурсна система для захоплення соціального руху». Це набір даних, призначений для фіксації та точного коментування соціальних взаємодій людей. Анотації та мітки були застосовані до понад 480 синхронізованих відеопотоків, фіксуючи та позначаючи тривимірну структуру руху людей, залучених у соціальні взаємодії.

Можна самому додавати анотації до зображень чи відео. Цей процес є досить складним і трудомістким і зазвичай використовують інструмент анотацій. Такі інструменти дають змогу:

- Легко визначити примітиви об’єктів для відеоанотацій оцінки пози.Наприклад, анотатори можуть чітко визначати та малювати ключові точки на людському тілі, де вони потрібні, наприклад суглоби чи риси обличчя.
- Визначте свій шаблон скелета з ключовими точками в позі. Створивши каркасний шаблон, його можна редагувати та застосовувати до моделей машинного навчання. Використовуючи функції відстеження об’єктів Encord для оцінки пози, ви можете значно зменшити навантаження на анотації вручну на початку будь-якого проекту. Завдяки потужному та гнучкому пакету редагування ви можете зменшити обсяг даних, необхідних моделі, а також мати можливість вручну редагувати мітки та ключові точки під час розробки проекту.
- Відстеження об'єктів і оцінка пози. Завдяки потужному алгоритму відстеження об’єктів правильний інструмент може підтримувати широкий спектр модальностей комп’ютерного зору.

Дослідники та анотатори можуть застосовувати цей набір даних для виявлення ключових моментів у русі людини та більш точного розрізнення об’єктів у відео в контексті. Окрім відстеження рухів людини та анотацій, COCO також виявляється корисним під час аналізу відео.

РОЗДІЛ 3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

3.1 Реалізація програмного забезпечення

У даному дипломному проекті було розроблено програмний модуль за допомогою мови програмування Python, це високорівневий інтерфейс – Keras для побудови, тренування та використання моделей глибокого навчання. Він є частиною TensorFlow і надає зручну абстракцію для розробки моделей на основі нейронних мереж.

Завдяки TensorFlow і Keras можна побудувати модель глибокого навчання, яка може аналізувати зображення та точно визначати положення різних частин тіла людини, таких як голова, руки, ноги тощо. TensorFlow надає ресурси для побудови та тренування таких моделей, використовуючи різні типи шарів нейронних мереж, включаючи згорткові нейронні мережі, рекурентні нейронні мережі та розгалужені мережі. Перевагою використання TensorFlow і Keras є можливість використовувати готові моделі, які вже навчені на великих наборах даних для виявлення пози людини. Це дозволяє швидко отримати результати без необхідності тренування моделі з нуля. Крім того, можна створювати власні моделі і навчити її виконувати завдання виявлення пози людини, використовуючи власний набір даних.

Використано підхід MeTRAbs [16], що в основному базується на оцінці пози однієї особи і розширюється до застосувань з кількома особами шляхом виявлення усіх людей на зображенні, а потім запуску оцінки пози для кожної з виявленої особи. Ця так звана стратегія мультиособового верхнього рівня вже вбудована в самостійну модель для зручності. Це розширення для кількох осіб також враховує неявне обертання, яке викликає обрізання. Замість простого обрізання, ця модель автоматично забезпечує застосування відповідного перетворення гомографії для перспективного відновлення (опціонально також радіального і тангенціального спотворення лінзи), з анти-альясінгом, і повертає пози в тривимірній світовій системі координат (або у системі координат камери, якщо зовнішню калібрацію не надано).

Моделі також містять вбудовану можливість аугментації під час тестування (перетворення кожного обрізу кілька разів і усереднення результатів) та підтримку пригнічення неправдоподібного виводу.

Моделі представляються за допомогою TensorFlow і завантажуться за допомогою наступного коду:

```
import tensorflow as tf
model = tf.saved_model.load('path_to_model')
```

Виявлення особи здійснюється за допомогою методу model.detect_poses, де здійснюється абсолютна 3D-оцінка пози людини для декількох осіб, фільтрацією правдоподібності вихідної пози та немаксимальним придушенням на одному зображенні.

```
model.detect_poses(
    image, intrinsic_matrix=UNKNOWN,
    distortion_coeffs=(0, 0, 0, 0, 0), extrinsic_matrix=eye(4),
    world_up_vector=(0, -1, 0), default_fov_degrees=55, internal_batch_size=64,
    antialias_factor=1, num_aug=5, average_aug=True, skeleton='',
    detector_threshold=0.3, detector_nms_iou_threshold=0.7,
    max_detections=-1, detector_flip_aug=False,
    suppress_implausible_poses=True)
```

Аргументи даного методу:

– image: Tensor типу uint8 з формою [H, W, 3], що містить RGB зображення;

– intrinsic_matrix: Tensor типу float32 з формою [3, 3], матриця внутрішньої камери. Якщо залишити значення за замовчуванням, матриця внутрішньої камери визначається з аргументу default_fov_degrees;

– distortion_coeffs: Tensor типу float32 з формою [5], п'ять коефіцієнтів дисторсії лінз згідно з моделлю спотворення OpenCV і у тому ж порядку (k1, k2, p1, p2, k3);

– extrinsic_matrix: Tensor типу float32 з формою [4, 4], матриця зовнішньої камери, з міліметрами як одиницею вектора переміщення;

– world_up_vector: Tensor типу float32 з формою [3], вектор, що вказує вгору в системі координат світу. Використовується для передачі

"вертикальних" обрізок до моделі оцінки пози, навіть якщо кут нахилу камери відмінний від нуля;

– default_fov_degrees: у разі, якщо не вказана intrinsic_matrix, ця скалярна величина типу float вказує поле огляду в градусах вздовж більшої сторони вхідного зображення;

– internal_batch_size: ціле число, розмір партій обрізків, які надсилаються внутрішній моделі обрізків. Обрізки зображень групуються внутрішньо, тому в разі багатьох виявлень вони обробляються порціями, щоб уникнути вичерпання пам'яті GPU. Придатне значення залежить від доступної пам'яті GPU та розміру базової моделі;

– antialias_factor: ціле число, коефіцієнт суперсемплінгу для уникнення ефектів аліасінгу, наприклад, якщо встановити 2, то спочатку ми внутрішньо створюємо обрізки розміром 512x512 пікселів і зменшуємо результат до 256x256 для передбачення;

– num_aug: кількість аугментацій, які виконуються для кожного екземпляра людини під час тестування. Виконуються різні повороти, відображення та зміни яскравості для кожного обрізку, які подаються через модель обрізків, і результати усереднюються після перетворення назад до системи координат камери;

– average_aug: булеве значення, чи повинен результат містити середні значення для різних результатів аугментації під час тестування. Якщо False, то результати будуть містити всі результати аугментації окремо;

– skeleton: рядок, який вказує, яку конвенцію скелету використовувати. Дивіться вкінці сторінки для доступних варіантів;

– detector_threshold: значення float для порогової фільтрації внутрішнього детектора людини;

– detector_nms_iou_threshold: значення float для використання non-max suppression всередині детектора. Занадто низькі значення можуть пригнічувати пози, які знаходяться близько до інших у зображенні, тоді як

занадто високі значення можуть призводити до дублікатів (менш ймовірно виникне проблема, якщо suppress_implausible_poses=True);

– max_detections: ціле число, обмежує кількість виявлень найкращими значеннями max_detections. Це може прискорити передбачення, оскільки до оцінювача поз передається лише обмежена кількість виявлень. Встановіть -1, щоб використовувати всі виявлення;

– detector_flip_aug: булеве значення, яке вказує, чи запускати зображення через детектор з горизонтальним перевертанням, а також агрегувати результати (перед кроком NMS детектора);

– suppress_implausible_poses: булеве значення, яке вказує, чи підтримувати (тобто виключити з результату) пози з неправдоподібними довжинами кісток, неоднорідними результатами аугментації під час тестування та виконувати non-maximum suppression на рівні пози (це додається до NMS детектора).

Метод повертає словник з трьома ключами і відповідними значеннями тензорів:

– boxes: [left, top, width, height, confidence] для кожної коробки виявлення. Форма - [num_detections, 5];

– poses3d: 3D-пози, відповідні виявленим людям. Кожна пози має форму [num_joints, 3] і виражена в тривимірній світовій системі координат у міліметрах (або в системі координат камери, якщо не вказано extrinsic_matrix). Кількість суглобів залежить від вибраного скелета як вхідного аргументу. Форма - [num_detections, num_joints, 3], якщо average_aug=True, в іншому випадку [num_detections, num_aug, num_joints, 3];

– poses2d: Аналогічно poses3d, але у двовимірних піксельних координатах (адресація пікселів вхідного зображення), тому вона має форму [num_detections, num_joints, 2], якщо average_aug=True, в іншому випадку [num_detections, num_aug, num_joints, 2].

Слід увагу, що num_detections - це фактична кількість виявлень людей, яка може бути менша або рівна max_detections, залежно від результатів фільтрації і придатності.

Модель model.estimate_poses забезпечує абсолютну тривимірну оцінку пози людини з використанням декількох осіб і тестової аугментації на одному зображенні. Аналогічно до моделі detect_poses, але прямокутні рамки надаються користувачем, а не вбудованим детектором. Виклик методу представлений наступним кодом:

```
model.estimate_poses(
    image, boxes, intrinsic_matrix=UNKNOWN,
    distortion_coeffs=(0, 0, 0, 0, 0), extrinsic_matrix=eye(4),
    world_up_vector=(0, -1, 0), default_fov_degrees=55, internal_batch_size=64,
    antialias_factor=1, num_aug=5, average_aug=True, skeleton='')
```

Аргументи:

– image: Tensor типу uint8 форми [H, W, 3], що містить зображення RGB.

– boxes: [ліворуч, верх, ширина, висота] для кожної прямокутної рамки осіб. Тензор форми [кількість_рамок, 4] типу tf.float32.

– решта аргументів мають таку ж форму і значення, як у методі detect_poses.

Моделі підтримують кілька різних конвенцій скелета, для цього потрібно вказати одну з наступних назв конвенцій як аргумент skeleton до функцій розпізнавання пози:

– smpl_24: Модель тіла SMPL

– coco_19: Суглоби COCO, включаючи таз в середині стеген та шию в середині плечей, як у CMU-Panoptic.

– h36m_17: Найпоширеніша конвенція суглобів Human3.6M

– h36m_25: Розширений набір суглобів Human3.6M

– mpi_inf_3dhp_17: Основні суглоби MPI-INF-3DHP (ті ж самі, що й суглоби MuPoTS)

– mpi_inf_3dhp_28: Повний набір суглобів MPI-INF-3DHP

– smpl+head_30: Суглоби SMPL разом з ключовими точками обличчя від COCO та верхньою точкою голови від MPI-INF-3DHP (рекомендується для візуалізації, оскільки SMPL_24 не має ключових точок обличчя).

– (порожній рядок): Всі суглоби, на яких була навчена модель.

Щоб отримати назви суглобів та з'єднання між ними використовуються атрибути per_skeleton_joint_names та per_skeleton_edges:

```
pred = model.detect_poses(image, skeleton='smpl_24')
for i, j in model.per_skeleton_edges['smpl_24']:
    draw_line(pred['poses2d'][i], pred['poses2d'][j])
```

Усе вищезазначене також працює для багатьох вхідних зображень (пакетний режим) за допомогою методу detect_poses_batched. У цьому випадку перший аргумент повинен мати форму [batch_size, height, width, 3]:

```
# Для прикладу копіюється одне й те ж зображення двічі.
# Можна використати  різні кадри відео.
images = tf.stack([image, image], axis=0)

# Одна матриця внутрішньої проекції. Вона буде застосована до всіх вхідних зображень у пакеті.
intrinsic_matrix = tf.constant([[[1962, 0, 540], [0, 1969, 960], [0, 0, 1]]], dtype=tf.float32)

# Результати є tf.RaggedTensors, оскільки кожне вхідне зображення може мати різну кількість
# осіб на ньому.
pred = model.detect_poses_batched(image, intrinsic_matrix=intrinsic_matrix)
pred['boxes'], pred['poses2d'], pred['poses3d']

# Різні матриці внутрішньої проекції для кожного вхідного зображення у пакеті:
intrinsic_matrix = tf.constant([
    [[1962, 0, 540], [0, 1969, 960], [0, 0, 1]],
    [[2521, 0, 530], [0, 2501, 970], [0, 0, 1]]], dtype=tf.float32)
pred = model.detect_poses_batched(image, intrinsic_matrix=intrinsic_matrix)
pred['boxes'], pred['poses2d'], pred['poses3d']
```

Оскільки користувач не контролює обмежувальні прямокутники в цій моделі, можна виконати перевірку правдоподібності для кожної передбаченої пози та відхилити ті, які не є дійсними позами (на основі довжини кісток і узгодженості збільшення). Крім того, дублікати пригнічуються за допомогою немаксимального придушення на основі 3D-пози. Це дозволяє встановити нижчий поріг детектора, щоб зменшити кількість помилкових негативів і в той же час усунути більшість помилкових позитивних результатів.

Для встановлення обмежувальних рамок можна також використати метод estimate_poses для одного та estimate_poses_batched для багатьох зображень.

```
import tensorflow as tf

model = tf.saved_model.load('metrabs_eff2l_y4')
image = tf.image.decode_jpeg(tf.io.read_file('test_image_3dpw.jpg'))
intrinsic_matrix = tf.constant([[1962, 0, 540], [0, 1969, 960], [0, 0, 1]], dtype=tf.float32)

# Блоки представлені в порядку [ліворуч, зверху, ширина, висота].
person_boxes = tf.constant(
    [[0, 626, 367, 896], [524, 707, 475, 841], [588, 512, 54, 198]], tf.float32)
pred = model.estimate_poses(image, boxes=person_boxes, intrinsic_matrix=intrinsic_matrix)
pred['poses2d'], pred['poses3d']
```

Щоб виявляти пози на декількох зображеннях одночасно, обмежувальні прямокутники мають бути надані в tf.RaggedTensor, оскільки кожне зображення в групі може містити різну кількість людей.

```
images = tf.stack([image, image], axis=0)
intrinsic_matrix = tf.constant([
    [[1962, 0, 540], [0, 1969, 960], [0, 0, 1]],
    [[2521, 0, 530], [0, 2501, 970], [0, 0, 1]]], dtype=tf.float32)

# Блоки представлені в порядку [ліворуч, зверху, ширина, висота].
person_boxes = tf.ragged.constant([
    [[0, 626, 367, 896], [524, 707, 475, 841], [588, 512, 54, 198]],  # 3 блоки для першого
    [[52, 514, 125, 741], [5, 160, 290, 414]],  # 2 блоки для другого
], tf.float32, ragged_rank=1, inner_shape=(4,))

# Повернені значення є tf.RaggedTensors, які відповідають підрахункам поля введення
pred = model.estimate_poses_batched(image, boxes=person_boxes, intrinsic_matrix=intrinsic_matrix)
pred['poses2d'], pred['poses3d']
```

3.2 Інтерфейс користувача

Google Colab є безкоштовним сервісом для виконання коду Python у хмарі. Він надає можливість створювати та виконувати Jupyter-блокноти безпосередньо у браузері без необхідності установки та конфігурування середовища.

У Google Colab багато бібліотек, включаючи TensorFlow і Keras, котрі вже встановлені за замовчуванням. Однак, якщо потрібна певна версія

TensorFlow або додаткові бібліотеки, їх можна також додатково встановити для даного блокноту. Для роботи з модулем в Google Colab потрібно:

1. Відкрити веб-браузер і перейти на сайт Google Colab за адресою https://colab.research.google.com.
2. У верхньому меню натиснути на кнопку "File" (Файл) і вибрати опцію "New notebook" (Новий блокнот). Завантажиться новий блокнот – текстовий інтерфейс для вводу колу програми, готовий для виконання коду Python.
3. Добавити код модуля розпізнавання пози людини у комірку блокнота (рисунок 3.1).

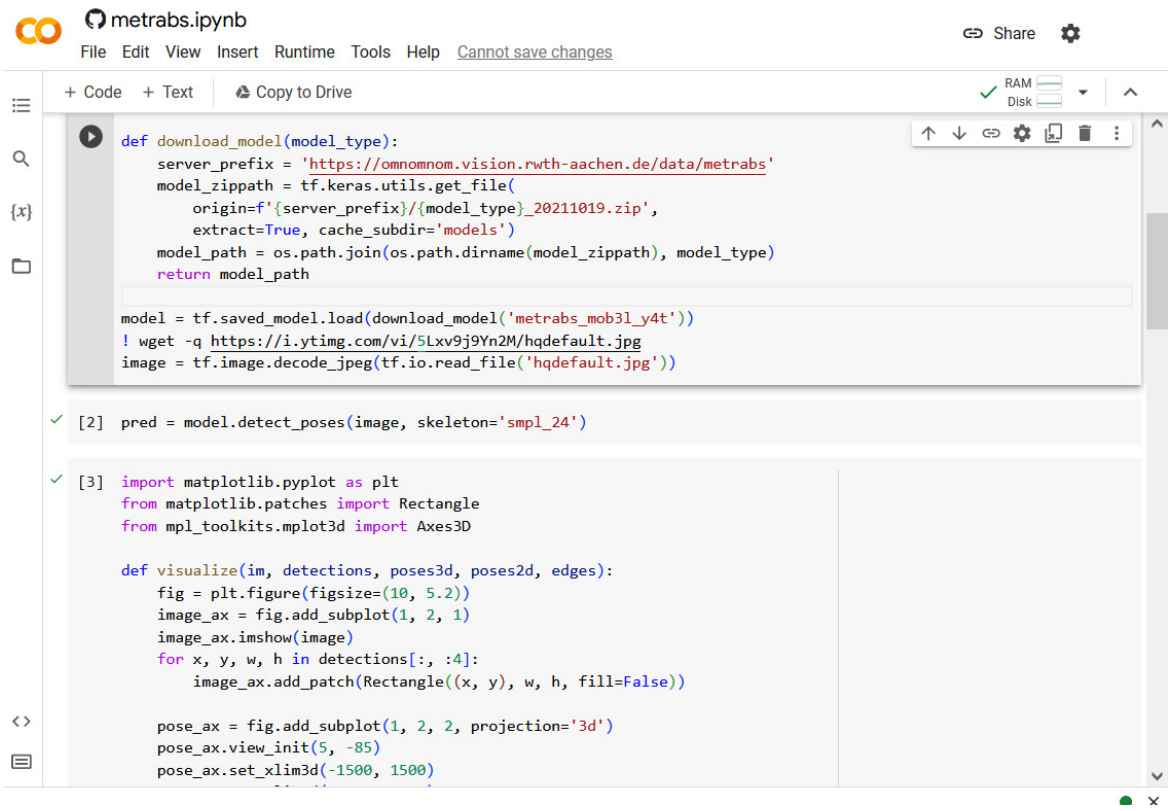


Рисунок 3.1 - Робота з модулем в Google Colab.

4. Натиснути кнопку "Runtime" (Виконання) у верхньому меню і обрати опцію "Run all" (Виконати все), що запустить виконання всього коду юлокноту.

5. Після успішного виконання коду результат виведеться у тому ж вікні блокнота у веб-браузері (рис 3.2).

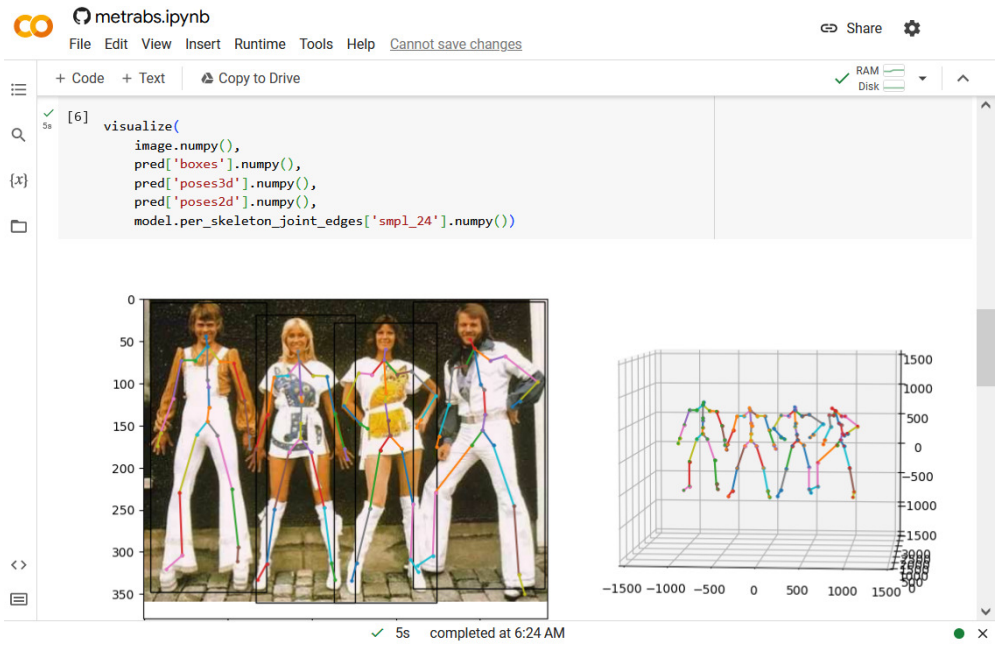


Рисунок 3.2 – Виконання програми в Google Colab.

6. Весь проект внесений у блокнот та результати виконання можна зберегти на комп'ютер або на Google Drive (рис. 3.3).

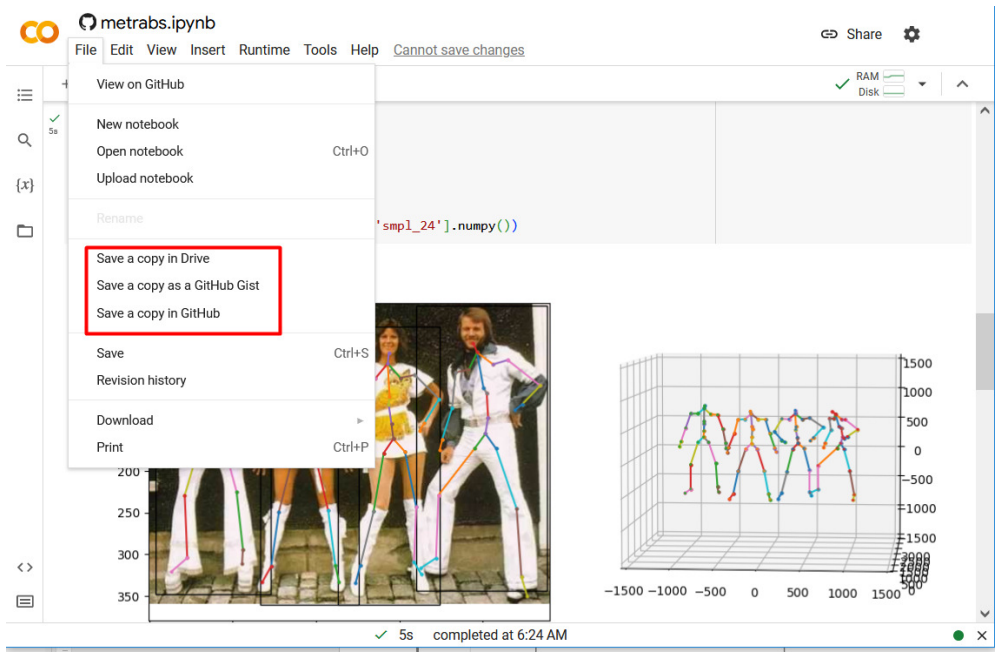


Рисунок 3.3 – Збереження результатів

3.3 Тестування програмного забезпечення

В даному розділі було детально перевірено роботу модуля на зображеннях де пози частково накладаються та з неповним зображенням.

Першим було здійснено тестування на реальних зображеннях, де користувач може підготувати зображення, які містять людей з різними позами, і використати їх для тестування модуля. Зображення передається в модуль і перевіряється чи він правильно розпізнає пози людей.

Для цього варіанту тестування були використані зображення з власної бібліотеки (рис. 3.4). Для розпізнавання пози і побудови скелету зображення попередньо завантажуються у директорію проекту блокноту, також в коді прописується назва відповідного файлу (рис. 3.5).

а) б)

Рисунок 3.4 - Зображення з власної бібліотеки

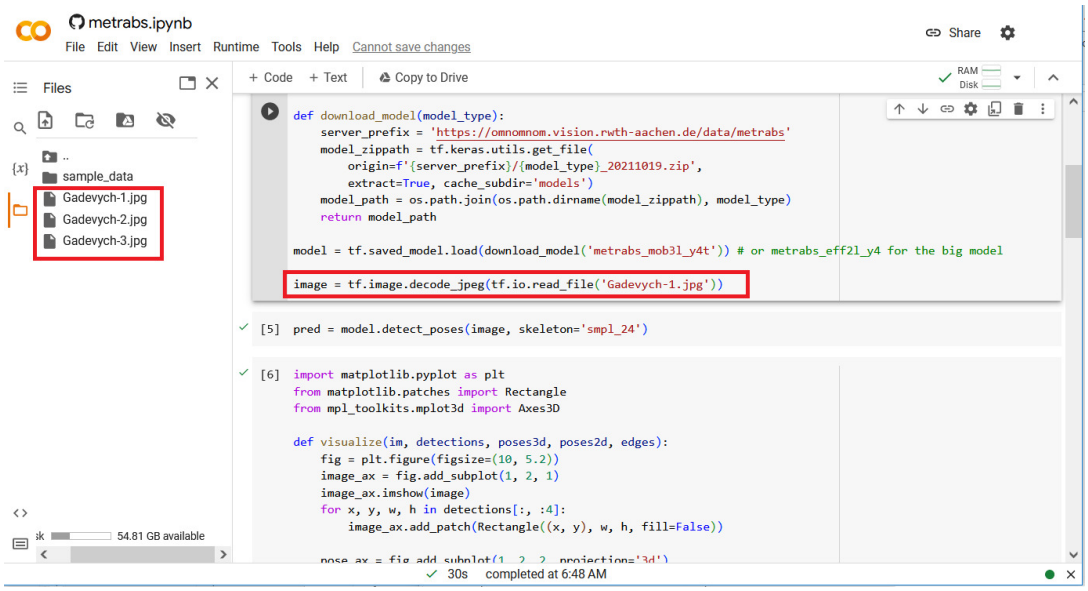


Рисунок 3.5 – Завантаження локальних зображень у директорію модуля

Після виконання модуля видно, що кількість людей правильно визначена – про що свідчать прямокутники чорного кольору, також видно що скелети вірно побудовані, навіть для накладених зображень людей (рис. 3.6).

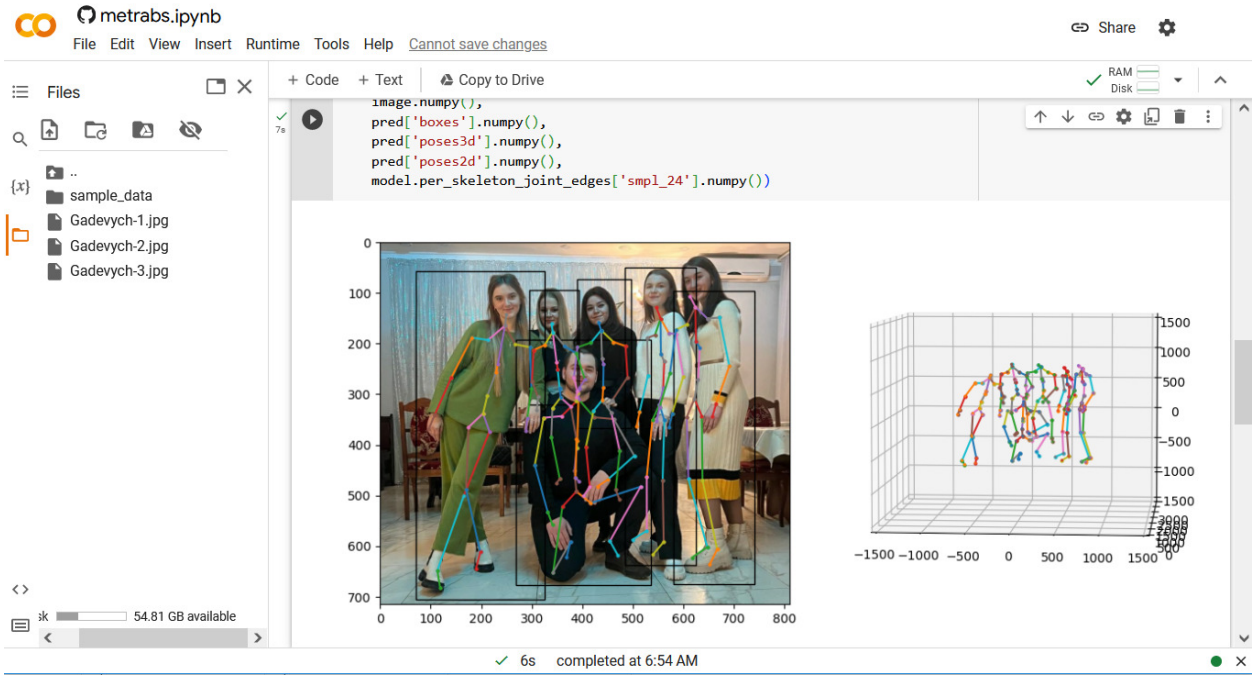


Рисунок 3.6 - Результат виконання модуля на зображенні 3.4 (а).

Завантаживши неповне зображення (рис. 3.7) модуль все одно успішно виявив правильну кількість людей та добудував їх віртуальні скелети.

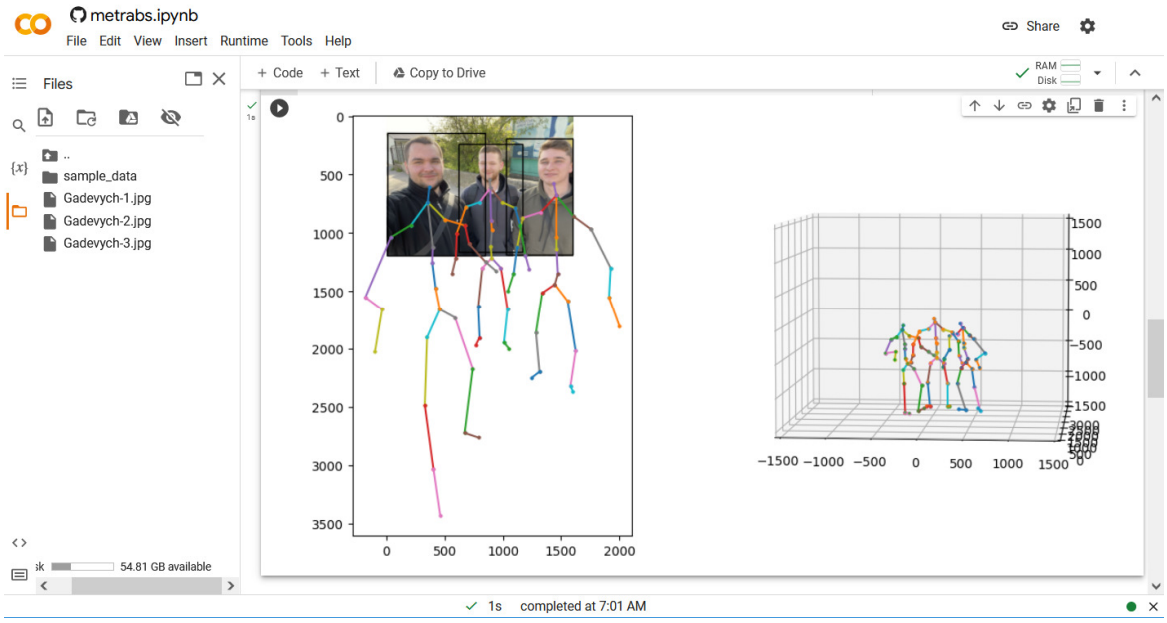


Рисунок 3.7 –- Результат виконання модуля на зображенні 3.4 (б).

Інший підхід до тестування модуля передбачає валідацію з використанням даних з розміченою позою, де завантажується зображеннях з публічних наборів даних, для якого була вже розмічена поза і порівнюються з отриманими даними. В літературі [17] представлені результати оцінки пози гравців гандболу (рис. 3.8). Результат представлений на рис. 3.9.

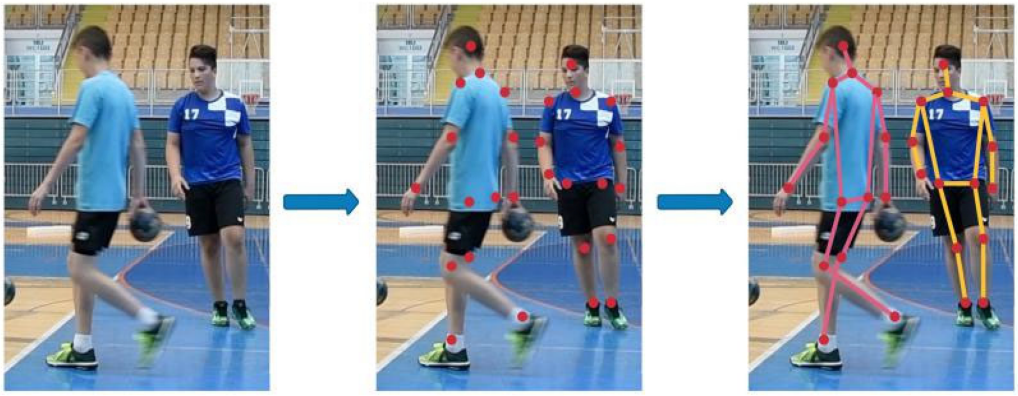


Рисунок 3.8 – Приклад оцінки пози гравців гандболу іншими системами.

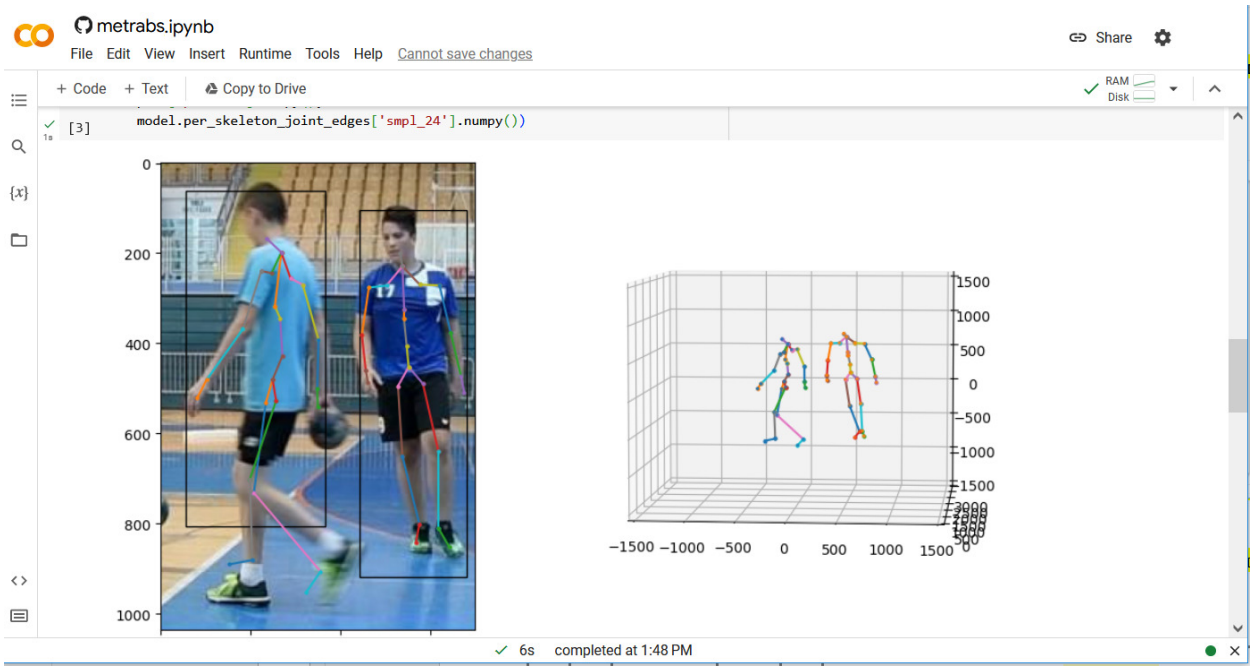


Рисунок 3.9 - Оцінка пози гравців гандболу запропонованим методом

Як видно з рисунку модуль виявив двох гравців та успішно побдував для них скелет оцінки пози.

ВИСНОВКИ

Мета дипломного проекту полягала у розробці та реалізації програмного модуля, який буде розпізнавати позу людини.

Проаналізувавши предметну область, існуючі системи розпізнавання пози людини, було розроблено структуру модуля та алгоритмічне забезпечення.

Проаналізувавши інструментарій та мови програмування, для розробки програмного продукту було обрано мову програмування Python, відкриту нейромережну бібліотеку Keras для побудови, тренування та використання моделей глибокого навчання. Ця бібліотека є частиною TensorFlow і надає зручну абстракцію для розробки моделей на основі нейронних мереж.

В процесі розробки дипломного проекту було вивчено та порівняно декілька методів розпізнавання пози. Для реалізації обрано метод який найкраще відповідає вимогам та поставленим задачам. Також проведено тестування модуля наступними способами: тестування на реальних зображеннях, де користувач може підготувати зображення, які містять людей з різними позами, та тестування з використанням даних з розміченою позою, де завантажується зображеннях з публічних наборів даних, для якого була вже розмічена поза і порівнюються з отриманими даними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. End-to-end image recognition solutions to automate, analyze and interpret visual data [Електронний ресурс]. – Режим доступу: https://www.fujitsu.com/global/services/business-services/computer-vision/

2. Introduction to computer vision: History and applications [Електронний ресурс]. – Режим доступу: https://www.superannotate.com/blog/introduction-to-computer-vision#key-insights

3. Redmon, J., & Farhadi, A. YOLO9000: Better, Faster, Stronger. The 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

4. Yang, L., Qin, Y. & Zhang, X. Lightweight densely connected residual network for human pose estimation // J Real-Time Image – 2021. – Proc. 18, P. 825–837.

5. Ergonomic Risk Analysis [Електронний ресурс]. – Режим доступу: https://viso.ai/application/ergonomics-analysis/

6. How Machine Learning is Shaping the Future of Sports [Електронний ресурс]. – Режим доступу: https://www.strong.io/

7. Build innovative AI products by managing top-quality training data— SuperData [Електронний ресурс]. – Режим доступу: https://www.superannotate.com/

8. George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler and Kevin Murphy. Real-time Human Pose Estimation in the Wild Using DenseNet. - arXiv:1907.00837v1, 1 Jul 2019 [Електронний ресурс]. – Режим доступу: https://pure.mpg.de/rest/items/item_3093760/component/file_3093762/content

9. Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1812.08008

10. Q. Dang, J. Yin, B. Wang and W. Zheng. Deep learning based 2D human pose estimation: A survey // Tsinghua Science and Technology, vol. 24, no. 6, pp. 663-676, Dec. 2019.

11. Yucheng Chena, Yingli Tianb, Mingyi Hea. Monocular Human Pose Estimation: A Survey of Deep Learning-based Methods [Електронний ресурс]. – Режим доступу: https://arxiv.org/pdf/2006.01423.pdf

12. X. Ren. A Novel Pose Recognition Algorithm based on Multi-Feature Point Fusion for Training Images// 2023 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 2023, pp. 83-87.

13. Ce Zheng, Sijie Zhu, Matias Mendieta, Taojiannan Yang, Chen Chen, Zhengming Ding. 3d human pose estimation with spatial and temporal transformers // Proceedings of the IEEE/CVF International Conference on Computer Vision – 2021, P. 11656-11665.

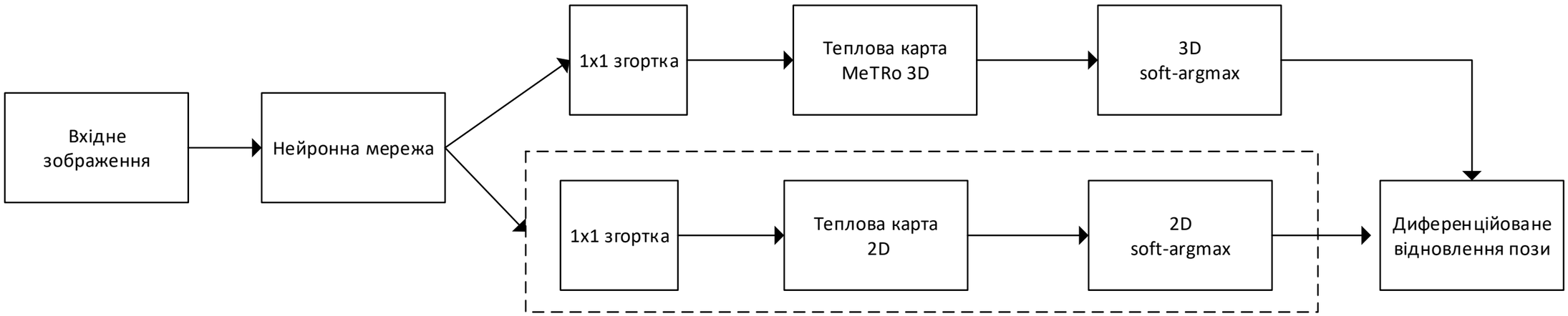
14. Yonghao Dang, Jianqin Yin, Shaojie Zhang. Relation-Based Associative Joint Location for Human Pose Estimation in Videos // IEEE Transactions on Image Processing – 2022. - Vol.31, P.3973-3986.

15. Chao-Lung Yang, Wen-Ting Li, Shang-Che Hsu. Skeleton-based Hand Gesture Recognition for Assembly Line Operation // 2020 International Conference on Advanced Robotics and Intelligent Systems (ARIS) – 2020 – P. 1-6.

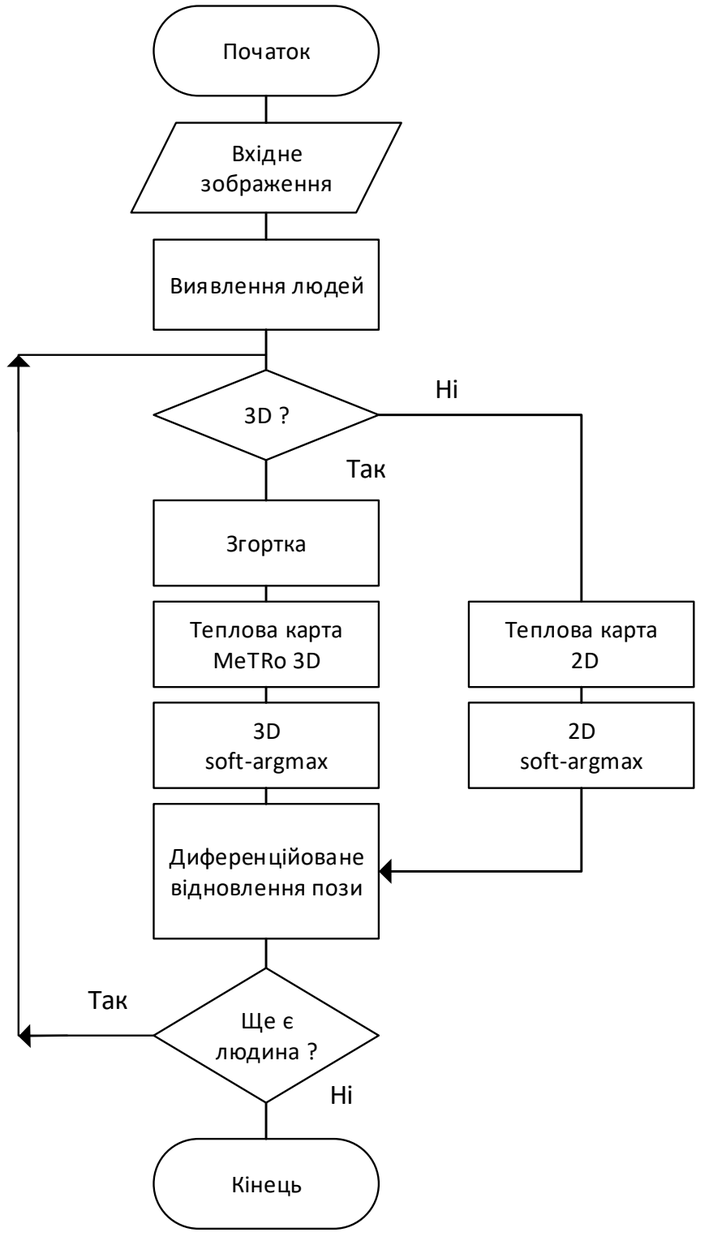
16. I. Sárándi, T. Linder, K. O. Arras and B. Leibe, "MeTRAbs: Metric-Scale Truncation-Robust Heatmaps for Absolute 3D Human Pose Estimation," in IEEE Transactions on Biometrics, Behavior, and Identity Science, vol. 3, no. 1, pp. 16-30, Jan. 2021, doi: 10.1109/TBIOM.2020.3037257.

17. Šajina, R.; Ivašić-Kos, M. 3D Pose Estimation and Tracking in Handball Actions Using a Monocular Camera. J. Imaging 2022, 8, 308. https://doi.org/10.3390/jimaging8110308

18. Методичні вказівки до виконання дипломного проекту освітнього ступеня «бакалавр» спеціальності 122 «Комп'ютерні науки» / Укл. А.О. Саченко, М.П. Комар, Н.М. Васильків, Г.М. Гладій, В.С. Коваль. – Тернопіль: ЗУНУ, 2021. – 56 с.



					ДП.КН.9500009.065 А1			
						Літера	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата	Структура програмного модуля			
Розроб.		Гадевич В.Ю.				Аркуш 1	Аркушів 1	
Перевір.		Загородня Д.І.						
Консультант						ЗУНУ, ФКІТ, КН-42		
Т. Контр.								
Н. Контр.		Загородня Д.І.						
Затверд.		Комар М.П.						



					ДП.КН.9500009.065 А1			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Гадевич В.Ю.			Алгоритм роботи програмного модуля	Літера	Маса	Масштаб
Перевір.		Загородня Д.І.						
Консультант						Аркуш 1	Аркушів 1	
Т. Контр.								
Н. Контр.		Загородня Д.І				ЗУНУ, ФКІТ, КН-42		
Затверд.		Комар М.П.						

Додаток В

Основні частини коду модуля

```
import tensorflow as tf
import os

def download_model(model_type):
    server_prefix = 'https://omnomnom.vision.rwth-aachen.de/data/metrabs'
    model_zippath = tf.keras.utils.get_file(
        origin=f'{server_prefix}/{model_type}_20211019.zip',
        extract=True, cache_subdir='models')
    model_path = os.path.join(os.path.dirname(model_zippath), model_type)
    return model_path

model = tf.saved_model.load(download_model('metrabs_mob3l_y4t')) # or metrabs_eff2l_y4 for the big model
! wget -q https://raw.githubusercontent.com/isarandi/metrabs/master/img/test_image_3dpw.jpg
image = tf.image.decode_jpeg(tf.io.read_file('test_image_3dpw.jpg'))


pred = model.detect_poses(image, skeleton='smpl_24')


import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
from mpl_toolkits.mplot3d import Axes3D


    def visualize(im, detections, poses3d, poses2d, edges):
        fig = plt.figure(figsize=(10, 5.2))
        image_ax = fig.add_subplot(1, 2, 1)
        image_ax.imshow(image)
        for x, y, w, h in detections[:, :4]:
            image_ax.add_patch(Rectangle((x, y), w, h, fill=False))

        pose_ax = fig.add_subplot(1, 2, 2, projection='3d')
        pose_ax.view_init(5, -85)
        pose_ax.set_xlim3d(-1500, 1500)
        pose_ax.set_zlim3d(-1500, 1500)
        pose_ax.set_ylim3d(0, 3000)

        # Matplotlib plots the Z axis as vertical, but our poses have Y as the vertical axis.
        # Therefore, we do a 90° rotation around the X axis:
        poses3d[..., 1], poses3d[..., 2] = poses3d[..., 2], -poses3d[..., 1]
        for pose3d, pose2d in zip(poses3d, poses2d):
            for i_start, i_end in edges:
                image_ax.plot(*zip(pose2d[i_start], pose2d[i_end]), marker='o', markersize=2)
                pose_ax.plot(*zip(pose3d[i_start], pose3d[i_end]), marker='o', markersize=2)
            image_ax.scatter(*pose2d.T, s=2)
            pose_ax.scatter(*pose3d.T, s=2)


                fig.tight_layout()
                plt.show()

          visualize(
              image.numpy(),
              pred['boxes'].numpy(),
              pred['poses3d'].numpy(),
              pred['poses2d'].numpy(),
              model.per_skeleton_joint_edges['smpl_24'].numpy())
```

```
intrinsic_matrix = tf.constant([[1962, 0, 540], [0, 1969, 960], [0, 0 Loading... pe=tf.float32)
boxes = tf.constant([[0, 626, 367, 896], [524, 707, 475, 841], [588, 512, 54, 198]], tf.float32)
pred = model.estimate_poses(image, intrinsic_matrix=intrinsic_matrix, boxes=boxes, skeleton='smpl_24')
visualize(
    image.numpy(),
    boxes.numpy(),
    pred['poses3d'].numpy(),
    pred['poses2d'].numpy(),
    model.per_skeleton_joint_edges['smpl_24'].numpy())
```

Робота з камерою

```
import os

import cv2
import tensorflow as tf

import poseviz


def main():
    model = tf.saved_model.load(download_model('metrabs_rn18_y4'))
    skeleton = 'smpl+head_30'
    joint_names = model.per_skeleton_joint_names[skeleton].numpy().astype(str)
    joint_edges = model.per_skeleton_joint_edges[skeleton].numpy()
    viz = poseviz.PoseViz(joint_names, joint_edges)

    for frame in frames_from_webcam():
        pred = model.detect_poses(
            frame, skeleton=skeleton, default_fov_degrees=55, detector_threshold=0.5)
        camera = poseviz.Camera.from_fov(55, frame.shape[:2])
        viz.update(frame, pred['boxes'], pred['poses3d'], camera)




def frames_from_webcam():
    cap = cv2.VideoCapture(0)
    while (frame_bgr := cap.read()[1]) is not None:
        yield frame_bgr[..., ::-1]


def download_model(model_type):
    server_prefix = 'https://omnomnom.vision.rwth-aachen.de/data/metrabs'
    model_zippath = tf.keras.utils.get_file(
        origin=f'{server_prefix}/{model_type}.zip',
        extract=True, cache_subdir='models')
    model_path = os.path.join(os.path.dirname(model_zippath), model_type)
    return model_path


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

```
import os

import tensorflow as tf


def main():
    model = tf.saved_model.load(download_model('metrabs_eff2l_y4'))
    image = tf.image.decode_jpeg(tf.io.read_file('img/test_image_3dpw.jpg'))
    skeleton = 'smpl_24'

    pred = model.detect_poses(image, default_fov_degrees=55, skeleton=skeleton)
    pred = tf.nest.map_structure(lambda x: x.numpy(), pred)  # convert tensors to numpy arrays
    print(pred['boxes'], pred['poses3d'], pred['poses2d'])

    joint_names = model.per_skeleton_joint_names[skeleton].numpy().astype(str)
    joint_edges = model.per_skeleton_joint_edges[skeleton].numpy()
    visualize(image.numpy(), pred, joint_names, joint_edges)


def download_model(model_type):
    server_prefix = 'https://omnomnom.vision.rwth-aachen.de/data/metrabs'
    model_zippath = tf.keras.utils.get_file(
        origin=f'{server_prefix}/{model_type}.zip',
        extract=True, cache_subdir='models')
    model_path = os.path.join(os.path.dirname(model_zippath), model_type)
    return model_path


def visualize(image, pred, joint_names, joint_edges):
    try:
        visualize_poseviz(image, pred, joint_names, joint_edges)
    except ImportError:
        print(
            'Install PoseViz from https://github.com/isarandi/poseviz to get a nicer 3D'
            'visualization.')
        visualize_matplotlib(image, pred, joint_names, joint_edges)


def visualize_poseviz(image, pred, joint_names, joint_edges):
    # Install PoseViz from https://github.com/isarandi/poseviz
    import poseviz
    camera = poseviz.Camera.from_fov(55, image.shape)
    viz = poseviz.PoseViz(joint_names, joint_edges)
    viz.update(frame=image, boxes=pred['boxes'], poses=pred['poses3d'], camera=camera)
```

```
def visualize_matplotlib(image, pred, joint_names, joint_edges):
    detections, poses3d, poses2d = pred['boxes'], pred['poses3d'], pred['poses2d']

    import matplotlib.pyplot as plt
    # noinspection PyUnresolvedReferences
    from mpl_toolkits.mplot3d import Axes3D
    from matplotlib.patches import Rectangle
    plt.switch_backend('TkAgg')

    fig = plt.figure(figsize=(10, 5.2))
    image_ax = fig.add_subplot(1, 2, 1)
    image_ax.imshow(image)
    for x, y, w, h in detections[:, :4]:
        image_ax.add_patch(Rectangle((x, y), w, h, fill=False))


    pose_ax = fig.add_subplot(1, 2, 2, projection='3d')
    pose_ax.view_init(5, -85)
    pose_ax.set_xlim3d(-1500, 1500)
    pose_ax.set_zlim3d(-1500, 1500)
    pose_ax.set_ylim3d(0, 3000)
    pose_ax.set_box_aspect((1, 1, 1))

    # Matplotlib plots the Z axis as vertical, but our poses have Y as the vertical axis.
    # Therefore, we do a 90° rotation around the X axis:
    poses3d[..., 1], poses3d[..., 2] = poses3d[..., 2], -poses3d[..., 1]
    for pose3d, pose2d in zip(poses3d, poses2d):
        for i_start, i_end in joint_edges:
            image_ax.plot(*zip(pose2d[i_start], pose2d[i_end]), marker='o', markersize=2)
            pose_ax.plot(*zip(pose3d[i_start], pose3d[i_end]), marker='o', markersize=2)
        image_ax.scatter(*pose2d.T, s=2)
        pose_ax.scatter(*pose3d.T, s=2)

    fig.tight_layout()
    plt.show()


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