КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Курсова робота

заспеціальністю 122 Комп'ютерні науки
на тему:

РОЗПІЗНАВАННЯ РУХІВ ЛЮДИНИ НА ВІДЕО ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

Виконав студент 3 курсу
Красіков Антон Ігорович _____
(підпис)

Науковий керівник:
професор, доктор фіз.-мат. наук
Терещенко Василь Миколайович _____
(підпис)

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2024

РЕФЕРАТ

Обсяг роботи 25 сторінок, 8 ілюстрацій, 1 таблиця, 20 джерел посилань.

Ключові слова: КОМП’ЮТЕРНИЙ ЗІР, КАРТА ГЛИБИНИ, ЕКСПОРТ ДАНИХ, ПОБУДОВА 3-D СКЕЛЕТОНУ, КАНАЛИ ЗАПИСУ.

Об’єкт роботи: комп’ютерна система для побудови 3-D скелетону по ключовим точкам тіла людини на відео для подальшого аналізу положення та виконуваних рухів.

Мета роботи: ознайомлення аудиторії з технологіями запису та експорту необхідних даних для роботи (колір + глибина). Подальше створення 3-D моделі положення людини за допомогою нейромереж задля широкого використання в подальшому у сфері медицини та інших.

Методи дослідження: аналіз існуючих підходів до знаходження рішень до подібних задач; визначення найбільш ефективних способів для досягнення цілей з урахуванням технічних можливостей. Розробка власних рішень. що реалізують поставлені задачі.

Інструменти: камера Microsoft Kinect 2.0 для запису даних з картою глибини, безкоштовне програмне забезпечення до камери Kinect for Windows SDK, безкоштовний редактор вихідного коду Visual Studio, мова програмування C++, бібліотека wxWidgets.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки

На сьогоднішній день технологія скелетонізації людини на зображенні та створення 3-D моделей є досить розповсюдженою, та швидко розвивається. Подібні алгоритми знаходять своє застосування у багатьох сферах, як-то в спорті (аналіз рухів спортсменів для запобігання травм), сфері безпеки (ідентифікація підозрілих рухів людей у публічних місцях), ігровій сфері (створення більш реалістичних моделей людини) і т.д.

Станом на 2024 рік існує досить велика кількість бібліотек та моделей, які здатні будувати як 2-D, так і 3-D моделі людей на зображеннях. Ось лише деякі найпопулярніші з них:

- **OpenPose [1]** - це перша й одночасно найпопулярніша модель для виявлення ключових точок людського тіла, розроблена в Університеті Карнегі-Меллона. Вона підтримує роботу в режимі реального часу та здатна розрізняти декілька людей одночасно. Основний режим роботи – 2-D, але можлива побудова 3-D моделей з різних ракурсів.
- **MMPose [2]** - це набір інструментів із відкритим кодом для оцінки пози на основі Pytorch [3]. Він здатен розрізняти ключові точки тіла людей та тварин, точки на обличчі та руках. Також моделює 2-D скелетони.
- **Mediapipe [4]** – фреймворк під мову програмування Python [5], розроблений командою Google. Є найпростішим у використанні, працює в режимі реального часу та повертає 3-D моделі, щоправда, з дещо нижчою точністю.

Актуальність роботи і підстави для виконання

Не дивлячись на те, що існує досить багато готових рішень для оцінки пози людини, включаючи й моделювання в реальному часі, більшість із них або працюють лише з 2-D моделями, або дають погану точність в 3-D, або

містять доволі складний та неінтуїтивний алгоритм побудови 3-D поз із застосуванням багатьох ракурсів.

Дана робота потрібна в першу чергу для того, щоб спростити цей процес та зробити зрозумілішими кроки задля отримання необхідних даних і їх подальшої обробки за допомогою нейромережі з метою побудови 3-D моделі.

Виконання даної роботи дозволить як ознайомитися з технологіями отримання різних структур даних та їх подальшої обробки, так і зробити цей процес простішим для користувача, що створить програмну базу для майбутнього застосування у різних сферах.

Мета й завдання роботи

Метою роботи є ознайомлення з технологіями запису та експорту необхідних даних для роботи. Подальше створення 3-D моделі положення людини за допомогою нейромереж задля широкого використання в подальшому у сфері медицини та інших.

Завдання для досягнення мети:

- Опанувати технології скелетонізації та запису даних, обрати найбільш актуальний спосіб серед наявних.
- Визначити структуру даних, необхідних для отримання високої точності.
- Розробити систему для експорту даних та перевірити їх шляхом практичного застосування.

Можливі сфери застосування

Технологія запису даних та розпізнавання рухів людини широко використовується у різних сферах, що частково були згадані у пункті оцінки сучасного стану об’єкту розробки.

Дана ж робота має основні переваги саме за рахунок спрощення процесу запису даних та своєї точності у 3-D просторі, і тому найкраще розкриє свій потенціал у тих сферах, де немає змоги докладати багато зусиль та необхідна велика точність розпізнавання положення людини.

Такою сферою може стати, наприклад, медицина. Дані напрацювання можуть бути використані для покращення процесу реабілітації людей і запобігання їх травмування. Маючи простий спосіб знімати дані та отримувати високоточну 3-D модель положення людини, можна забезпечити правильне виконання вправ навіть без постійної присутності наглядача.

Окрім цього, використання даної розробки може також допомогти побачити на ранніх стадіях будь-які зміни в рухах пацієнта і зрозуміти вплив тих чи інших вправ на його стан, бачити поступовий прогрес у виконанні.

РОЗДІЛ 1. ОГЛЯД ВИКОРИСТАНИХ ІНСТРУМЕНТІВ

1.1 Microsoft Kinect v2

На сьогодні практично всі відомі нам камери, з якими ми постійно маємо справу, знімають фото та відео у форматі RGB [6], тобто стандартному кольоровому, де кожний піксель містить інформацію лише про інтенсивність червоного, зеленого та синього кольору, в результаті утворюючи звичні для нас зображення. Цього цілком достатньо для того, щоб продемонструвати оточуючу нас реальність на екрані телефону чи ноутбуку, але такий підхід дещо відрізняється від звичного нашого сприйняття реальності і є менш повним.

Важливою відмінністю людського ока від камери RGB формату є здатність оцінювати відстань до об’єктів за допомогою порівняння зображень з двох очей та оцінювання їх різниці. І якщо закрити одне око, то можна одразу побачити, наскільки важче вдається орієнтуватися в просторі.

Саме через нездатність RGB формату надавати дані про глибину зображення він є недостатнім для того, щоб будувати модель пози людини у тривимірному просторі. Деякі моделі, як-то Mediapipe, намагаються оцінювати відстань до точок на тілі людини від сенсору, маючи лише дані про колір, але це вдається лише з дуже малою точністю, і тому залежно від того, з якої перспективи дивиться камера, можуть давати різні дані про положення.

Саме тому для того, щоб отримати точну модель положення людини у тривимірному просторі, недостатньо мати лише кольорове зображення, а ще й необхідно мати дані про глибину для кожного пікселя.

На сьогоднішній день існує досить багато різних камер, що можуть вимірювати глибину зображень, і користуються вони різними технологіями для цього.

Ось деякі найвідоміші з них [7]:

- **Стереоскопічне бачення.** Технологія, за якою фактично працює людський зір. Дві або більше камери на відстані одна від одної,

глибина оцінюється на основі різниці зображень. Цей спосіб є досить ресурсозатратним та повільним, і тому не бажаний в умовах даної роботи.

- **Структуроване світло.** Для визначення глибини використовує лазерне джерело світла для проектування світлових візерунків (переважно смугастих) на цільовий об’єкт. За отриманими спотвореннями розраховує відстань до об’єкта. Дає гарну точність на коротких відстанях, але погано працює з блискучими поверхнями та є досить дорогою технологією.
- **Time-of-Flight (ToF).** Камера випромінює короткі імпульси інфрачервоного світла та вираховує час, який потрібен світлу, щоб дійти до об’єкта, відбитися і повернутися назад. Має високу точність та відносно проста у використанні.

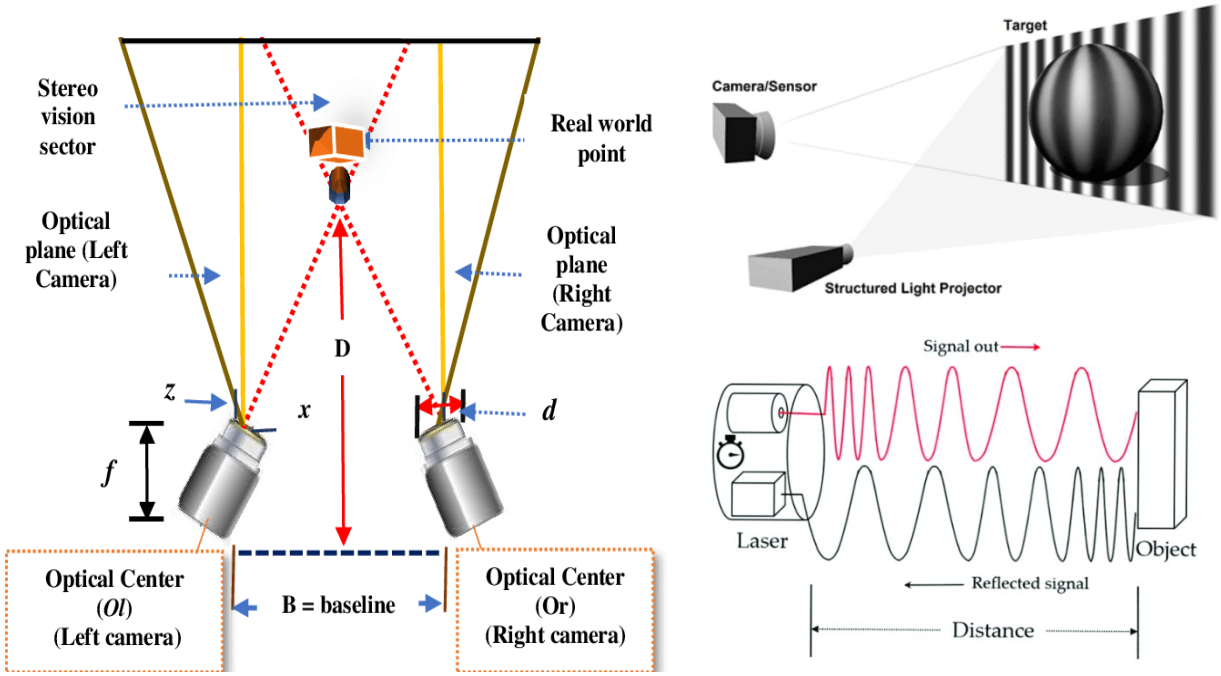

Рисунок 1. Порівняння технологій визначення глибини.

На основі перерахованих технологій було обрано технологію Time-of-Flight. Вона дає гарну точність при невеликому навантаженні на програмне забезпечення та недорога у використанні. Усе це робить її найбільш бажаним варіантом, враховуючи цілі роботи.

Наступним кроком був вибір самої камери з даною технологією. З-проміж великої кількості існуючих я зупинився на Microsoft Kinect v2 [8].

Microsoft Kinect v2 [9] — це сенсорний пристрій, розроблений Microsoft, який здебільшого використовувався для ігрової консолі Xbox One, але також підтримується й на Windows. Він є вдосконаленою версією першого Kinect, що був випущений у 2010 році, пропонуючи значно покращені технічні характеристики та можливості [10].

Дана камера використовує технологію ToF для точного визначення відстані до об’єктів, має як кольоровий сенсор для запису RGB кадрів, так і інфрачервоний, що й дозволяє вимірювати глибину, а також працювати в умовах поганого освітлення.

При цьому даний пристрій є досить недорогим, простим у використанні на більшості пристроїв з Windows та не перенавантажує систему складними розрахунками, завдяки останньому досягається досить пристойна частота кадрів навіть на не найбільш потужних пристроях.

Усе це дозволяє йому бути гарним варіантом для застосування широким колом людей для різноманітних завдань.

Також перевагою саме цього рішення є той факт, що Microsoft Kinect v2 був дуже популярним раніше, і тому має досить гарну документацію та покрокову інструкцію з експлуатації від Microsoft. Це дозволить значно спростити користування з ним як під час виконання даної роботи, так і в подальшому на практиці.

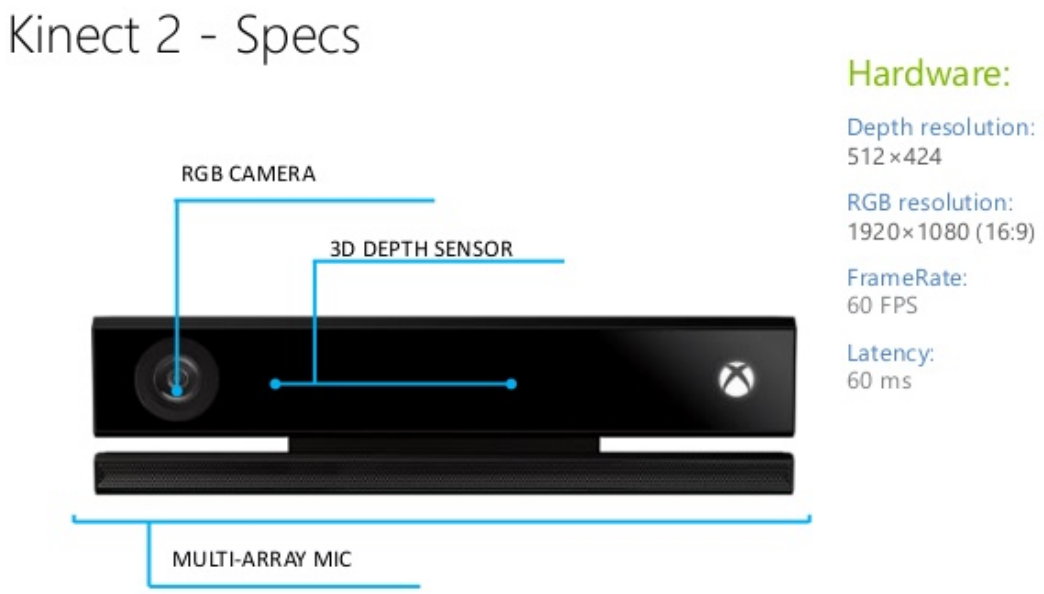

Рисунок 2. Вигляд та будова Microsoft Kinect v2

1.2 Мова програмування

Для збору та перетворення даних використовувалася компільована мова програмування загального призначення C++ [11]. Основними перевагами C++ при використанні в даній роботі є:

- **Висока продуктивність.** Завдяки тому, що C++ є компільованою мовою програмування, програми, написані на C++, працюють швидше, не втрачаючи час на інтерпретацію під час виконання. Також високій продуктивності сприяє низькорівневий контроль над ресурсами. Усе це є досить важливим при роботі з даними та прямо впливає на кількість кадрів на секунду.
- **Наявність документації.** Оскільки мова C++ є популярною та постійно підтримуваною, то вона має досить багато бібліотек, з якими можна працювати. У тому числі під C++ досить добре написані й бібліотеки [12] для роботи з Microsoft Kinect v2, що й є визначальним фактором вибору саме цієї мови програмування.

1.3 Бібліотека wxWidgets

wxWidgets [13] – це безкоштовна бібліотека C++ з відкритим вихідним кодом, яка дозволяє розробникам створювати програми для Windows, macOS, Linux та інших платформ за допомогою єдиного коду. Також має реалізації для Python, Ruby та деяких інших мов програмування.

Основним застосуванням wxWidgets є побудова графічного інтерфейсу користувача, хоча ця бібліотека має й інші можливості.

Серед плюсів використання саме цієї бібліотеки можна відзначити те, що вона:

- має гарну документацію, а також відкритий вихідний код та активну спільноту розробників, що гарантує її актуальність.
- wxWidgets легко підлаштовувати під свої потреби, завдяки можливості створення власних компонентів і розширень.

1.4 Технології для скелетонізації

1.4.1 Модель для визначення ключових точок

Для подальшої побудови ключових точок людини у тривимірному просторі мною була обрана модель комп’ютерного зору RefiNet [14][15].

RefiNet – це глибока модель комп’ютерного зору, що складається з декількох різних етапів та може будувати 3-D скелетон людини з дуже високою точністю. Вона отримує на вхід ключові точки людини у 2-D координатах, визначені за допомогою інших зовнішніх технологій, та карту глибини до зображення, на основі якого ці точки були побудовані.

RefiNet складається з трьох основних модулів, перший з яких аналізує карту глибин та вдосконалює 2-D координати ключових точок, другий об’єднує інформацію про глибину та отримані координати точок, і на основі цього будує 3-D скелетон, а третій модуль вдосконалює результати другого, використовуючи глибоку нейромережу.

Таким чином RefiNet значно підвищує точність отриманої 3-D моделі, при цьому будучи достатньо легкою з точки зору використовуваної пам’яті та обчислювальних ресурсів.

Model	Parameters (M)	Inference (ms)	VRAM (GB)
OpenPose	52.311	44.859	1.175
HRNet	28.536	43.385	1.107
Module A	0.828	1,872	0.669
Module B	4.302	0.824	0.665
Module C	2.935	13.806	1.681
RefiNet Pipeline	8.064	16.473	1.705

Таблиця 1. Порівняння розмірів моделей.

До того ж модель RefiNet досить зручно буде порівнювати з результатами побудованої 3-D моделі через Kinect, різницю в підході яких продемонстровано на наступному зображенні.

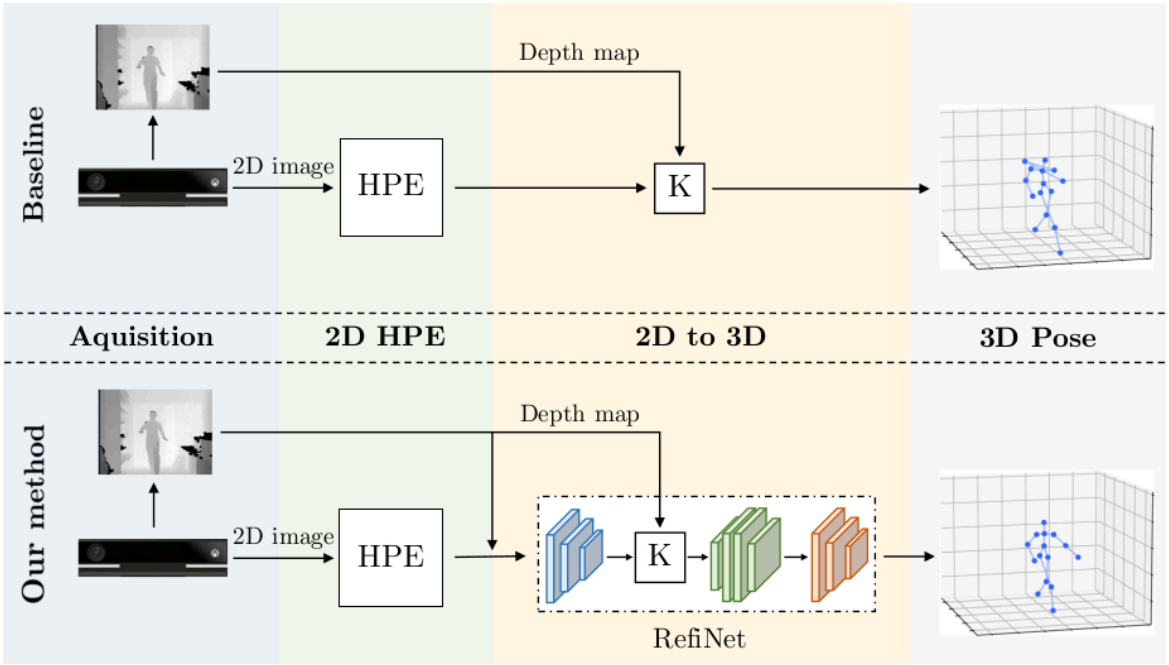

Рисунок 3. Суть методу роботи RefiNet.

Можна побачити, що метод, який використовує ця модель, є набагато більш потужним, аніж вбудоване оцінювання Kinect, і гарантує більш високу точність прогнозів.

1.4.2 Датасет для тренування моделі

Оскільки модель RefiNet є досить великою за кількістю параметрів, то для її тренування необхідна значна кількість якісно підібраних даних. Самому назбирати й розмітити стільки даних не є реально можливим через технічні можливості та час, і тому я скористаюся вже натренованими вагами на датасетах Itop [16] та Baracca [17].

Обидва датасети є достатньо великими та містять різні зображення людей з картами глибини та вказаними координатами ключових точок.

РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ

2.1 Підготовка камери

Для того, щоб почати роботу з записом даних, необхідно в першу чергу налаштувати пристрій для запису. В нашому випадку це сенсор Microsoft Kinect v2.

2.1.1 Підключення до пристрою

Спочатку під’єдную сенсор до пристрою. Для цього в комплекті йде адаптер, за допомогою якого Kinect одночасно під’єднується до мережі з одного боку та до ноутбука через USB-вхід з іншого. далі необхідно впевнитися, що пристрій розпізнає новий сенсор та після цього приступати до наступних кроків.

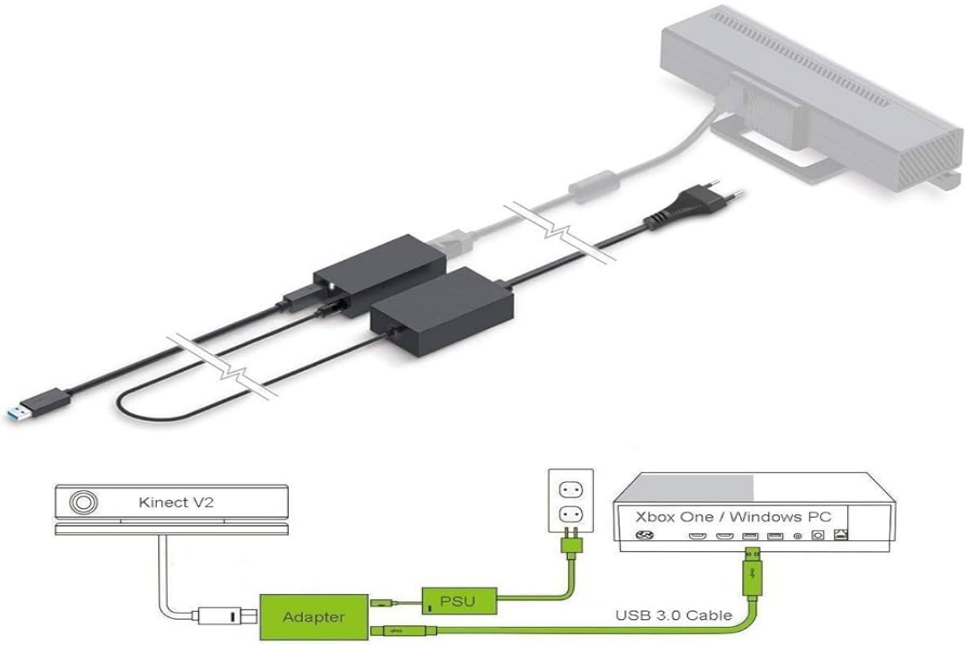

Рисунок 4. Комплектація Microsoft Kinect v2.

2.1.2 Налаштування програмного забезпечення

Далі, для того, щоб Kinect запрацював коректно, необхідно завантажити та встановити стандартне програмне забезпечення до нього, слідуючи офіційним інструкціям. Завантажую програму “Kinect for Windows SDK” [18], яка надає усі інструменти та встановлює драйвери, необхідні для розробки програм із підтримкою Kinect для Microsoft Windows.

Після цього впевнююся, що камера тепер працює коректно, для чого відображаю декілька різних режимів роботи через щойно встановлений застосунок. Нижче послідовно наведені режими роботи в RGB, IR та Depth.

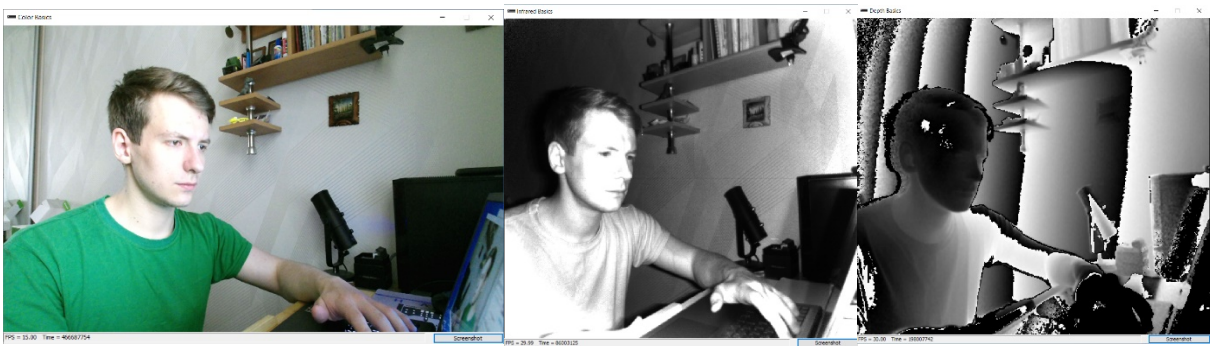

Рисунок 5. Різні режими роботи Kinect.

Як можна побачити, Kinect дійсно підтримує усі необхідні для подальшої роботи режими, а саме: RGB, depth та IR (інфрачервоний). Щоправда, стандартний застосунок дозволяє лише експорт кадрів у звичному форматі при натисненні на поле «Screenshot», і тому не підходить для запису послідовних кадрів з достатньою частотою.

2.2 Налаштування середовища для роботи

Для написання коду програми, що працюватиме з записом та експортом даних у правильному форматі, необхідно налаштувати середовище. Я використовував інтегроване середовище розробки Visual Studio [19].

Для того, щоб почати працювати з експортом даних, необхідно підключити усі потрібні бібліотеки. Для цього переходжу в каталог завантаженого Kinect for Windows SDK, копіюю звідти папки «Lib» та «inc» та вставляю їх у каталог з робочим проектом для зручності. Далі підключаю ці папки у Visual Studio через налаштування проекту project -> properties -> configuration properties -> VC++ directories. Шлях до каталогу «inc» додаю у «Include directories» та шлях до «Lib» у «Library directories» відповідно.

Аналогічно додаю шляхи до бібліотеки wxWidgets, щоб використовувати її у проекті для задач, пов’язаних з графікою.

2.3 Запис відео та експорт даних

2.3.1 Запис та експорт трьох каналів даних

Як уже згадувалося до цього, сенсор Kinect дозволяє одночасно отримувати кольорове зображення, інфрачервоне зображення та дані щодо глибини завдяки різним датчикам, що можуть працювати паралельно.

Взагалі-то, для коректної роботи моделі скелетонізації необхідні лише дані про колір та глибину, але запис даних у вигляді інфрачервоних зображень теоретично може бути корисним також у випадках поганого освітлення чи інших дефектів. Саме тому в даній роботі я експортую усі три канали одночасно, при цьому вимкнути один є тривіальною задачею, якщо це буде необхідно, скажімо, для збільшення швидкості запису чи економії пам’яті.

Оскільки для нас є необхідним отримання даних для роботи моделі у вигляді лише файлів з інформацією, то для реалізації задачі отримання цих файлів цілком достатньо простого консольного застосунку без використання особливого користувацького інтерфейсу (віконного чи якогось іще). Даний застосунок буде знімати дані з сенсорів камери з заданою частотою та записувати їх у каталог у заданому форматі. Такий застосунок спочатку має ініціалізувати камеру (з використанням вказівника на тип `IKinectSensor`)

```
// Initialize camera
if (FAILED(GetDefaultKinectSensor(&pSensor)) || !pSensor) {
    std::cerr << "Kinect sensor initialization failed!" << std::endl;
    return -1;
}
if (FAILED(pSensor->Open())) {
    std::cerr << "Failed to open Kinect sensor!" << std::endl;
    return -1;
}
```

а потім ініціалізувати по черзі всі три сенсори камери з використанням функцій інтерфейсу `IKinectSensor`). Нижче наведена ініціалізація сенсору глибини; сенсори отримання кольорового та інфрачервоного зображень ініціалізуються подібним чином:

```
IDepthFrameSource* pDepthFrameSource = nullptr;
IDepthFrameReader* pDepthFrameReader = nullptr;
if (FAILED(pSensor->get_DepthFrameSource(&pDepthFrameSource))) {
   std::cerr << "Failed to get depth frame source!" << std::endl;
```

```
   return -1;
}
if (FAILED(pDepthFrameSource->OpenReader(&pDepthFrameReader))) {
   std::cerr << "Failed to open depth frame reader!" << std::endl;
   return -1;
}
```

Тобто спочатку ініціалізується джерело кадру, а потім, з його допомогою — його зчитувач.

Після цього розпочинається основний цикл програми. Для забезпечення потрібної частоти кадрів використовується функція `this_thread::sleep_until()` та змінна `wait`, що відстежує стан читання кадрів (до початку зчитування кадрів може минути декілька невдалих спроб):

```
for(int wait = 0;;)
{
    if (wait == 1 || wait == 3)
    {
        awake = awake_time();
        wait = 2;
    }
    if (/* зчитування вдале */)
    {
        if (wait == 0) wait = 3; else wait = 1;
    }
    if (_kbhit() && _getch() == 0x1B) break;
    if (wait == 1)
    {
        this_thread::sleep_until(awake);
    }
}
```

Як видно з наведеного фрагменту псевдокоду, цикл завершує роботу при натисканні клавіші <Esc>.

Саме зчитування виконується наступним чином. Спочатку ми отримуємо кадри відразу з трьох сенсорів за допомогою раніше ініціалізованих зчитувачів:

```
if (  // Need ALL frames
    SUCCEEDED(pColorFrameReader->AcquireLatestFrame(&pColorFrame)) &&
    SUCCEEDED(pDepthFrameReader->AcquireLatestFrame(&pDepthFrame)) &&
    SUCCEEDED(pInfraredFrameReader->AcquireLatestFrame(&pInfraredFrame)))
```

Якщо кадри отримано успішно, то ми зчитуємо відповідну інформацію про розмір кадру, вказуємо його формат, і далі готуємо буфер для кадру (4

байти на піксель для кольорового зображення, та по 2 байти на піксель для інфрачервоного сенсора та сенсора глибини).

Після чого робимо запис кадрів у файли в raw-форматі для збільшення швидкості роботи з використанням відповідних інтерфейсів.

Далі отримані кадри за необхідності можуть бути конвертовані в будь-який поширений графічний формат на кшталт JPEG, TIFF тощо.

Ось деякі приклади даних, які були отримані шляхом запису послідовних кадрів із заданою частотою:

Рисунок 6. Фрагмент експортованих даних.

Варто зазначити, що, на відміну від кольорових зображень, які досить просто продемонструвати через стандартний RGB-формат 24 бітами (по 8 бітів = 256 значень) на кожний колір, з картами глибин все трохи складніше. Значення кожного пікселя лежить у діапазоні від 0 до 2^16, а стандартне представлення кольору складає 8 бітів на канал, тобто фізично не може відобразити увесь діапазон. Саме тому останнє зображення глибини виглядає так плоско.

Тим не менш, щоб показати, що насправді дані про глибину зображення є досить точними, скористаюся налаштуванням більш вузького діапазону значень та розподіленням значень між трьома каналами.

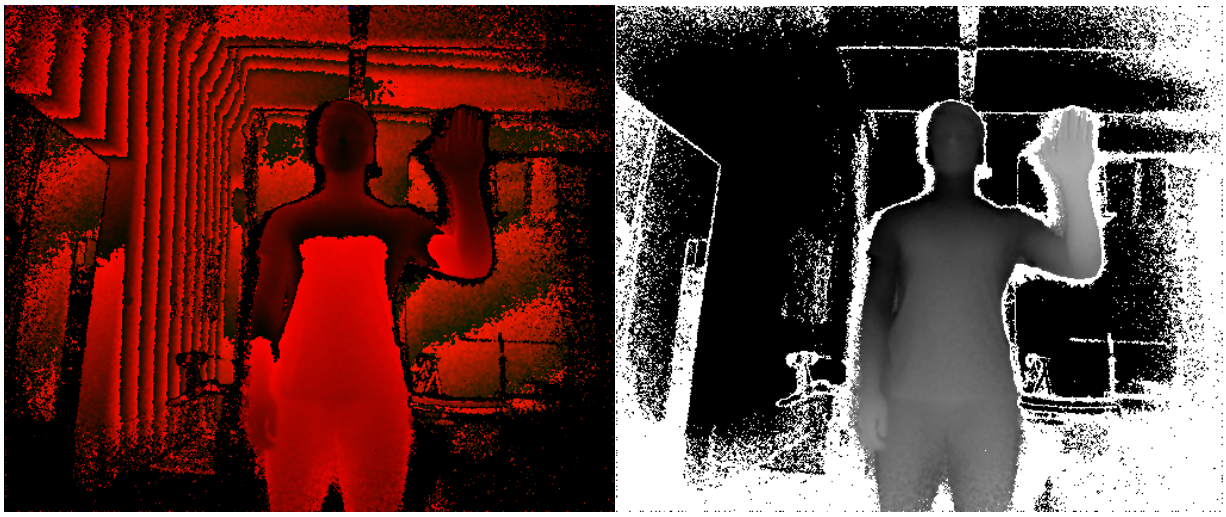

Рисунок 7. Способи відображення глибини.

На рисунку 7 видно, що таким чином дані про глибину стають більш інтерпретованими та наглядними.

2.3.2 Додаткові можливості Kinect

Насправді Microsoft Kinect v2 має набагато більше можливостей, окрім тих, що використовуються у даній роботі. В першу чергу, коли мова заходить про можливості Kinect, варто згадати те, для чого він взагалі був створений. І, як уже згадувалося раніше, дана технологія від Microsoft першочергово задумувалася як ігровий контролер, і саме цим і пояснюються всі її можливості.

Для того, щоб можна було користуватися Kinect як ігровим контролером, необхідно, щоб він розрізняв людину, її положення та рухи задля зчитування команд користувача та відображення їх у грі. Саме тому він містить датчик глибини, інфрачервоне та кольорове бачення. Окрім цього, для того, щоб з високою точністю розпізнавати людину в режимі реального часу, в Kinect вбудована легка модель комп’ютерного зору, яка й будує скелетон людини в тривимірному просторі [20].

Щоправда, побудовані кінектом 3-D моделі людини насправді містять достатньо великі похибки, і, хоча їх показників достатньо для того, щоб розпізнавати більшість рухів, які слід відобразити у грі, дані результати аж ніяк не можуть бути застосовані в медицині, де потрібна набагато більша точність.

Незважаючи на це, отримання даних про скелетон людини з кінекту є також важливим етапом даної роботи і потрібне з двох основних причин. Перша – це щоб порівнювати показники стандартної моделі та тої, яка буде побудована в подальшому в рамках цього дослідження. Друга ж – задля можливої передачі даних про 2-D скелетон на модель RefiNet задля подальшого вдосконалення результатів.

Тут можливі декілька різних рішень для отримання 2-D скелетону, і треба перевірити різні задля порівняння їх точності. Можна як використовувати відомі моделі для двовимірної скелетонізації, наприклад OpenPose, так і скелетон, побудований через Kinect. Саме тому його експорт буде доволі корисною функцією в даній роботі.

Для експорту даних про скелетон людини з кінекту було створено окремий проект у Visual Studio та окремий застосунок, що працює за тим самим консольним принципом виконання. Дані про скелетон в необробленому форматі мають вигляд набору ключових точок людини з координатами по осям x, y та z (ось z за потреби можна не використовувати). Для того, щоб якимось чином їх візуалізувати, я скористувався бібліотекою wxWidgets.

За її допомогою ключові точки будувалися, накладаючись на отримані кольорові зображення з Kinect і результуючі файли надалі експортувалися у форматі JPG у загальний каталог.

Результати того, який це мало вигляд, можна побачити нижче:

Рисунок 8. Ключові точки, визначені Kinect.

РОЗДІЛ 3. ПЛАН МАЙБУТНЬОЇ РОБОТИ

Описана в минулих розділах робота є лише початком для побудови реальної системи для реабілітації пацієнтів та будь-яких інших видів застосувань.

Оскільки застосування моделей побудови 3-D скелетонів людей у медицині потребує високої точності та стабільних і передбачуваних результатів, необхідно зробити велику кількість тестів та провести прискіпливий аналіз роботи моделі перед тим, як починати практичне використання.

Тому подальші кроки в розробці системи реабілітації будуть приблизно такими:

- Завантажити натреновані ваги моделі з офіційного сайту, запустити модель на експортованих даних. Впевнитися, що не виникає жодних помилок у процесі, і що отримані дані відповідають потрібному нам формату.
- Визначитися з метриками оцінювання точності моделі та оцінити за ними прогнози, які дає отримана модель. Перевірити її на загальнодоступних датасетах та порівняти з іншими моделями, як-то отриманими з Kinect тривимірними скелетонами. Зробити перевірку також на деяких експортованих даних.
- Налаштувати зручний процес для завантаження даних у модель, її запуску та запису отриманих результатів. Об’єднати все в єдиний пайплайн, використовуючи мову програмування Python та дотримуючись принципів об’єктно-орієнтованого програмування.
- Спробувати самостійно зібрати датасет з наявних відкритих у мережі та, можливо, з додаванням деякої кількості власних експортованих та розмічених даних. Натренувати на ньому модель, знайти найкращі конфігурації для цього. Протестувати точність прогнозів після цього та, за її покращення, замінити ваги моделі.

- Також можливі деякі зміни в архітектурі моделі, якщо вони дадуть користь.
- Для подальшого застосування в сфері медицини планується розробка зручного додатку зі зрозумілим юзер інтерфейсом, який надасть змогу легко записувати необхідні дані, отримувати 3-D модель ключових точок та, повертаючи значення необхідних кутів та положень суглобів, надавати користувачу поради щодо техніки тощо. Також необхідно налаштувати експорт історії виконання вправ задля наглядного відслідковування прогресу пацієнта.

Але, не дивлячись на те, що попереду ще залишилося досить багато роботи, та наразі система ще не готова настільки, щоб бути запущеною в користування, уже зроблено достатньо для того, щоб розуміти і отримувати базову структуру результатів з використанням вбудованої моделі записуючого сенсору та візуалізувати їх.

ВИСНОВКИ

Були розглянуті основні підходи до побудови ключових точок людини та наведено приклади, для чого це потрібно. Проаналізовані основні технології та ключові пункти на шляху до розв’язку цієї задачі.

У даній роботі ми продемонстрували свою власну послідовність кроків для побудови 3-D моделі ключових точок людини для подальшого застосування у сфері медицини. Пояснили, чому саме цей підхід є доречним та гарантує високу точність розрахунків.

Було налаштовано експорт та демонстрацію коректних даних з сенсору Microsoft Kinect v2 у три канали: кольоровий, глибинний та інфрачервоний. Також була проведена побудова базових скелетонів за допомогою вбудованих методів Kinect та продемонстрована їх точність.

Для подальшої побудови більш точних моделей було проаналізовано наявні рішення та обрано оптимальний варіант на основі даних про точність, простоту та швидкодію.

Виконання даної роботи допомогло покращити розуміння такої фундаментальної задачі комп’ютерного зору, як побудова скелетону, та на практиці попрацювати з новими форматами даних, як-то глибинна карта та інфрачервоні зображення.

Також ми сформували подальший план розвитку даної системи, список необхідних кроків та технологій, які необхідно буде застосувати для розробки повноцінної системи реабілітації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OpenPose [Електронний ресурс] - Режим доступу до ресурсу: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/index.html
2. MMPose [Електронний ресурс] - Режим доступу до ресурсу: https://mmpose.readthedocs.io/en/latest/overview.html
3. Pytorch Wikipedia [Електронний ресурс] - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/PyTorch
4. Mediapipe [Електронний ресурс] - Режим доступу до ресурсу: https://ai.google.dev/edge/mediapipe/solutions/guide
5. Python Wikipedia [Електронний ресурс] - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Python_(programming_language)
6. RGB color model Wikipedia [Електронний ресурс] - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/RGB_color_model
7. Types of Depth Cameras [Електронний ресурс] - Режим доступу до ресурсу: https://www.e-consystems.com/blog/camera/technology/what-are-depth-sensing-cameras-how-do-they-work/
8. Kinect Xbox Wikipedia [Електронний ресурс] - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Kinect
9. Microsoft Kinect for Windows official page [Електронний ресурс] - Режим доступу до ресурсу: https://learn.microsoft.com/en-us/windows/apps/design/devices/kinect-for-windows
10. RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments [Text] / [Henry, P.; Krainin, M.; Herbst, E.; Ren, X.; Fox, D.]. Int. J. Robot. Res. – 2012. – P. 647–663
11. C++ Wikipedia [Електронний ресурс] - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/C%2B%2B
12. Kinect v2 SDK C++ Tutorial [Електронний ресурс] - Режим доступу до ресурсу: https://ed.ilogues.com/Tutorials/kinect2/kinect1.html

13. wxWidgets [Електронний ресурс] - Режим доступу до ресурсу: https://www.wxwidgets.org/
14. RefiNet: 3D Human Pose Refinement with Depth Maps [Text] / [D'Eusanio, Andrea; Pini, Stefano; Borghi, Guido; Vezzani, Roberto; Cucchiara, Rita.]. - 2021. - P. 2320-2327.
15. RefiNet source code [Електронний ресурс] - Режим доступу до ресурсу: https://github.com/aimagelab/RefiNet
16. Itop Dataset [Електронний ресурс] - Режим доступу до ресурсу: https://zenodo.org/records/3932973 - .X5vuVYhKguU
17. Baracca Dataset [Електронний ресурс] - Режим доступу до ресурсу: https://aimagelab.ing.unimore.it/imagelab/page.asp?IdPage=37
18. Kinect for Windows SDK [Електронний ресурс] - Режим доступу до ресурсу: https://learn.microsoft.com/en-us/previous-versions/windows/kinect/dn799271(v=ieb.10)
19. Visual Studio IDE [Електронний ресурс] - Режим доступу до ресурсу: https://visualstudio.microsoft.com/
20. A Survey of Applications and Human Motion Recognition with Microsoft Kinect [Text] / [Roanna Lun; Wenbing Zhao]. – 2015. – 49с.