МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА
Факультет прикладної математики та інформатики
Кафедра програмування

Дипломна робота

Використання методів комп'ютерного зору і машинного навчання для розробки безконтактних інтерфейсів

Виконав:
студент 4 курсу, групи ПМІ-41
спеціальності 122 Комп'ютерні науки та
інформаційні технології (інформатика)
Мацьків П. А.
(підпис) (призвіще та ініціали)

Керівник
доцент Музичук А.О.
(підпис) (призвіще та ініціали)

Рецензент

(підпис) (призвіще та ініціали)

Львів – 2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра програмування

Спеціальність 122 Комп'ютерні науки та інформаційні технології (інформатика)
(шифр і назва)

«ЗАТВЕРДЖУЮ»

Завідувач кафедри

_____

" " 20 року

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Мацьківа Павла Анатолійовича
(прізвище, ім'я, по батькові)

1. Тема роботи Використання методів комп'ютерного зору і машинного навчання для розробки безконтактних інтерфейсів.

керівник роботи Музичук Анатолій Омелянович, кандидат фізико-математичних наук, доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені Вченою радою факультету від "13" вересня 2022 року №15

2. Строк подання студентом роботи 13 червня 2023 року_____

3. Вихідні дані до роботи
Програмні засоби бібліотек вільного доступу OpenCV та MediaPipe для розпізнавання відеозображень

_____

4. Зміст дипломної роботи (перелік питань, які потрібно розробити)

1) Визначити актуальність та проблематику безконтактних інтерфейсів.

2) Дослідити та обрати технології та методи, які будуть використовуватися для реалізації безконтактних інтерфейсів.

3) Розробити та продемонструвати перший метод безконтактного керування - метод керування жестами.

4) Розробити та продемонструвати другий метод безконтактного керування - метод керування рухами.

5) Розробити графічний інтерфейс для демонстрації та тестування обох методів.

__6) Провести тестування. Удосконалити методи керування та інтерфейс згідно з результатами тестування.

__7) Порівняти обидва методи керування. Визначити їхні переваги та недоліки.

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

__Діаграма роботи програми, таблиці матриць невідповідності, таблиця команд, таблиця порівнянь, GUI

6. Консультанти розділів роботи

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7. Дата видачі завдання ____13 вересня 2022 року_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	*Визначення проблеми та вибір технологій*		*Виконано*
2	*Розробка першого методу керування – методу керування жестами*		*Виконано*
3	*Розробка другого методу керування – методу керування рухами*		*Виконано*
4	*Розробка графічного інтерфейсу для демонстрації та тестування обох методів*		*Виконано*
5	*Проведення тестування*		*Виконано*
6	*Вдосконалення програми згідно з результатами тестування*		*Виконано*
7	*Порівняння методів керування*		*Виконано*
8	*Оформлення дипломної роботи*		*Виконано*

Студент _____ Мацьків П.А.

(підпис)

Керівник роботи _____ Музичук А.О.

(підпис)

Автореферат

Назва: Використання методів комп'ютерного зору і машинного навчання для розробки безконтактних інтерфейсів

Актуальність теми: Безконтактні інтерфейси вирішують проблеми доступності та гігієнічності користування комп'ютерами. Вони особливо важливі для користувачів з обмеженими можливостями, оскільки можуть служити кращою альтернативою звичайній периферії. Також вони дають змогу запобігти розповсюдженню інфекцій в медичних закладах, оскільки інфекції часто передаються через поверхні периферії.

Мета: Реалізація безконтактних методів керування комп'ютером, заснованих на зчитуванні жестів та рухів користувача, а також налаштування графічного інтерфейсу для дослідження та тестування цих методів.

Методи: В дипломній роботі представлено два безконтактних інтерфейса користувача, що не потребують спеціального пристрою для роботи, достатньо лише вебкамери. Перший метод – метод керування жестами розпізнає 6 жестів та виконує команду відповідно до показаного жесту. Другий метод – метод керування рухами відслідковує позицію руки у просторі та перетягує курсор миші відповідно до позиції руки.

Результати: Було розроблено два безконтактних інтерфейса для керування комп'ютером, що використовують вебкамеру для відслідковування рухів користувача. Ці інтерфейси були протестовані та вдосконалені згідно з результатами тестування.

Посилання на програму: https://github.com/Deainsi/tui-navigation-model

ЗМІСТ

ВСТУП

Актуальність теми. Безконтактні інтерфейси вирішують проблеми доступності та гігієнічності користування комп'ютерами. Для користувачів з обмеженими можливостями такі інтерфейси можуть бути кращою альтернативою за звичну периферію. Для людей з обмеженою мобільністю або вадами зору технології голосового контролю дають змогу виконувати прості задачі інтерфейсу, зокрема доступ до інформації онлайн і використання електронної пошти. Також люди з обмеженою мобільністю можуть використовувати технології відслідковування руху очей та розпізнавання жестів. Вони отримають змогу керувати комп'ютерами за допомогою малих та простих рухів, а не комбінацій, які можуть спричинити проблеми для людей з гіршою моторикою. Щодо гігієни, то у медичних закладах такі інтерфейси можуть запобігти розповсюдженню інфекції, оскільки вони передаються через поверхні периферії. Це також стосується і повсякденного життя, оскільки люди взаємодіють з різними терміналами.

Існуючі реалізації. Безконтактні інтерфейси – це не нова ідея. Залежно від визначення, "безконтактний інтерфейс" Nintendo Power Glove, випущений у 1989 році, можна вважати методом управління безконтакними інтерфейсами. Також існують і новіші та вдаліші програмні продукти, які базуються на понятті "безконактного інтерфейсу":

- Microsoft kinect[13] – це невдала спроба Майкрософт революціонувати ігрову індустрію. Хоч майбутнього у сфері геймінгу у нього немає, він знайшов своє місце у медицині та навіть освіті. Як девайс кінект відслідковує рухи та жести користувача за допомогою комп'ютерного зору та дає змогу керувати системою Xbox (сучасні версії мають sdk для windows).

- Google Home[14] – розумний динамік, що використовує NLP для взаємодії з користувачами. За допомогою цього пристрою користувачі можуть отримувати інформацію, контролювати елементи розумного будинку, здійснювати покупки тощо.

- Leap Motion Controller[15] – невеличкий пристрій, який відслідковує рухи руки користувача за допомогою інфрачервоних камер, а програма будує тривимірний скелет руки. Використовуючи Leap Motion Controller, можна грати в ігри або взаємодіяти з віртуальними тривимірними об'єктами.

- Девайси для відслідковування руху очей – це пристрої, що відслідковують рухи очей користувачів й дають їм змогу керувати комп'ютером лише поглядом.

- VR шоломи – це шоломи віртуальної реальності, які відслідковують рухи рук за допомогою контролерів, або камер у тривимірному просторі. Завдяки цьому користувачі можуть взаємодіти з віртуальним середовищем без фізичного контакту.

У даній роботі представлено два безконтактних інтерфейса користувача, що не потребують спеціального пристрою для роботи, достатньо лише вебкамери. Перший метод – метод керування жестами розпізнає 6 жестів та виконує команду відповідно до показаного жесту. Другий метод – метод керування рухами відслідковує позицію руки у просторі та перетягує курсор миші відповідно до позиції руки.

Метою цієї роботи є реалізація безконтаних методів керування комп'ютером, занованих на зчитуванні жестів та рухів користувача. Також мета даної праці полягає у налаштуванні графічного інтерфейсу для дослідження та тестування цих методів. Відповідно до мети потрібно виконати наступні завдання:

1. Реалізувати методи безконтактного керування ПК.
2. Створити GUI.
3. Провести тестування цих методів на інтерфейсі.
4. Вдосконалити методи керування та інтерфейс згідно з результатами тестування.
5. Порівняти методи.

1 Огляд використаних бібліотек

1.1 MediaPipe Hands

Розпізнавання руху рук – це доволі складне завдання для комп'ютерного зору, оскільки руки можуть часто загороджувати одна одну. Також трапляється, що комп'ютер "не бачить" всіх частин навіть однієї руки. До того ж на руці відсутня висока контрастність кольорів.

Для розпізнавання кисті руки у цій курсовій роботі використовується бібліотека MediaPipe Hands[8] від Google. Ця бібліотека дозволяє розпізнавати форму руки, не потребуючи багато ресурсів, тож її можна використовувати навіть у програмах для смартфонів. Дана бібліотека використовує 2 моделі для розпізнавання руки: модель розпізнавання долоні та модель для локалізації лендмарок руки.

Модель розпізнавання долоні використовується для того, щоб знайти руку на зображенні. Оскільки просто розпізнати руку доволі тяжко, спочатку використовується ця модель, що знаходить долоню та повертає приблизні межі, в яких знаходиться рука.

Модель лендмарок руки отримує обрізане зображення з рукою та розставляє 21 3D лендмарку (див. рис. 1). Також отримати приблизні межі, в яких знаходиться рука, можна з лендмарок з попереднього кадра. Тож модель розпізнавання долоні викликається тільки за потреби.

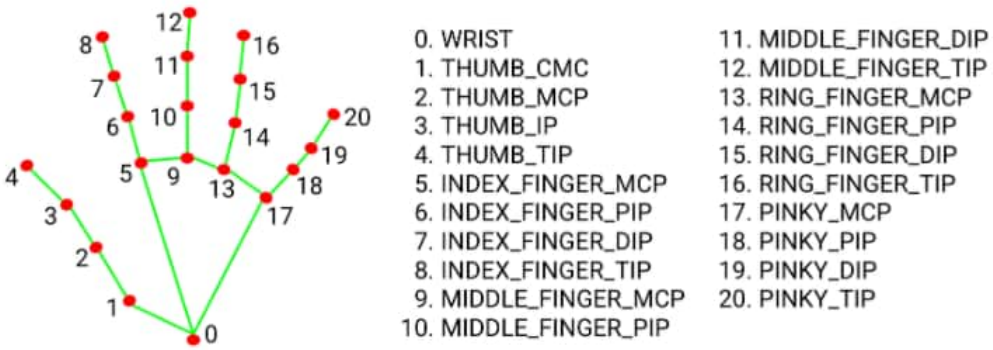

Рис. 1: Модель кисті руки

1.2 OpenCV

Для зчитування зображення з камери використовується бібліотека OpenCV [7]. Відео з камери зчитується покадрово у форматі BGR як масив NumPy [5]. Кадри передаються моделі mediapipe для отримання масиву лендмарок. Модель очікує вхідне зображення у форматі RGB. Отже, для кращого результату кадр, отриманий за допомогою OpenCV, потрбіно

конвертувати.

1.3 MLPclassifier

Класифікація – це одна з найвагоміших проблем машинного навчання, уміння комп'ютерів класифікувати та категоризувати зображення, символи, будь-який інший вид інформації, що є цінним у сучасному світі. MLP(Multi-layer perceptron (багатошаровий персептрон)) класифікатор, який згідно своєї власної назви, пов'язаний з нейронними мережами. Він складається з кількох шарів, включаючи вхідний шар, один або більше прихованих шарів і вихідний шар. Вхідний шар отримує характеристики, приховані шари витягують відповідну інформацію, а вихідний шар забезпечує остаточне рішення щодо класифікації. Модель коригує свої ваги та зміщення під час навчання за допомогою зворотного поширення, мінімізуючи різницю між прогнозованими та справжніми мітками.

2 Програмна реалізація інтерфейсів

Програмний код інтерфейсів реалізовано мовою Python з використанням бібліотек Pandas[2], NumPy, Scikit[6], OpenCV, Mediapipe, PyAutoGUI[9]. Програма для тренування моделі складається з двох частин: модуль для збору тренувальних даних та модуль для тренування моделі.

У першій програмі навігація відбувається шляхом використання жестів. За основу взято код програми для безконтактних інтерфейсів[10]. У цій програмі є 2 моделі, перша розпізнає жести для навігації меню, друга – для введення числових значень.

У другій програмі навігація відбувається у спосіб, при якому відслідковується позиція долоні. Модель цієї програми розпізнає лише 2 жести – відкрита та закрита долоня. Якщо вона закрита, то ми імітуємо натискання лівої кнопки миші. Для цієї програми було створено інтерфейс, використовуючи React, аби продемонструвати можливості даного методу в більш сприятливих умовах.

2.1 Збір тренувальних даних

У програмі реалізовано 2 способи збору тренувальних даних: збір даних зі зображення, або наживо. У першій реалізації потрібно задати шлях до папки зі зображеннями та жест, яким ми хочемо класифікувати ці зображення.

Програма ітеративно проходиться по усіх зображеннях у папці, виконуючи такі кроки:

1. Зображення зчитується за допомогою функції бібліотеки OpenCV - imread.
2. Зображення обробляється функцією **process** бібліотеки mediapipe.
3. З результату [2] ми беремо лендмарки й записуємо їх у файл з тренувальними даними.

У цій реалізації зображення можна класифікувати будь-яким жестом.

Друга реалізація є більш обмеженою у кількості можливих класів, їх лише шість: 0, 1, 2, 3, 4, 5. Зображення у цій реалізації надходять з відеокамери користувача, використовуючи функцію бібліотеки OpenCV – VideoCapture. Зображення знову обробляється функцією **process** бібліотеки mediapipe. За допомогою лендмарок для кожного пальця на одній руці, ми перевіряємо, чи він зігнутий у функції **is_finger_up**.

На кожен палець припадає по 4 лендмарки. Ми знаходимо їх у масиві лендмарок довжиною 21. Нульовим елементом у цьому масиві є лендмарка зап'ястя. Алгоритм збору тренувальних даних такий:

1. Обраховуємо відстань від зап'ястя до пальця, довжину пальця та відстань від зап'ястя до кінчика пальця.

2. Маючи 3 довжини з [1], ми можемо порахувати кут згину пальця, використовуючи теорему косинусів.

3. Якщо кут більше 160°, або 150° для великого пальця, то ми вважаємо що палець розігнутий.

4. Рахуємо кількість розігнутих пальців та класифікуємо зображення відповідним жестом.

5. Рахуємо кількість кадрів на секунду.

6. Отримані дані ми записуємо у файл для тренувальних даних, а також зберігаємо зображення.

Для того, щоб закрити програму, користувачу потрібно натиснути *Esc*. Приклад роботи продемонстровано на рис. 2.

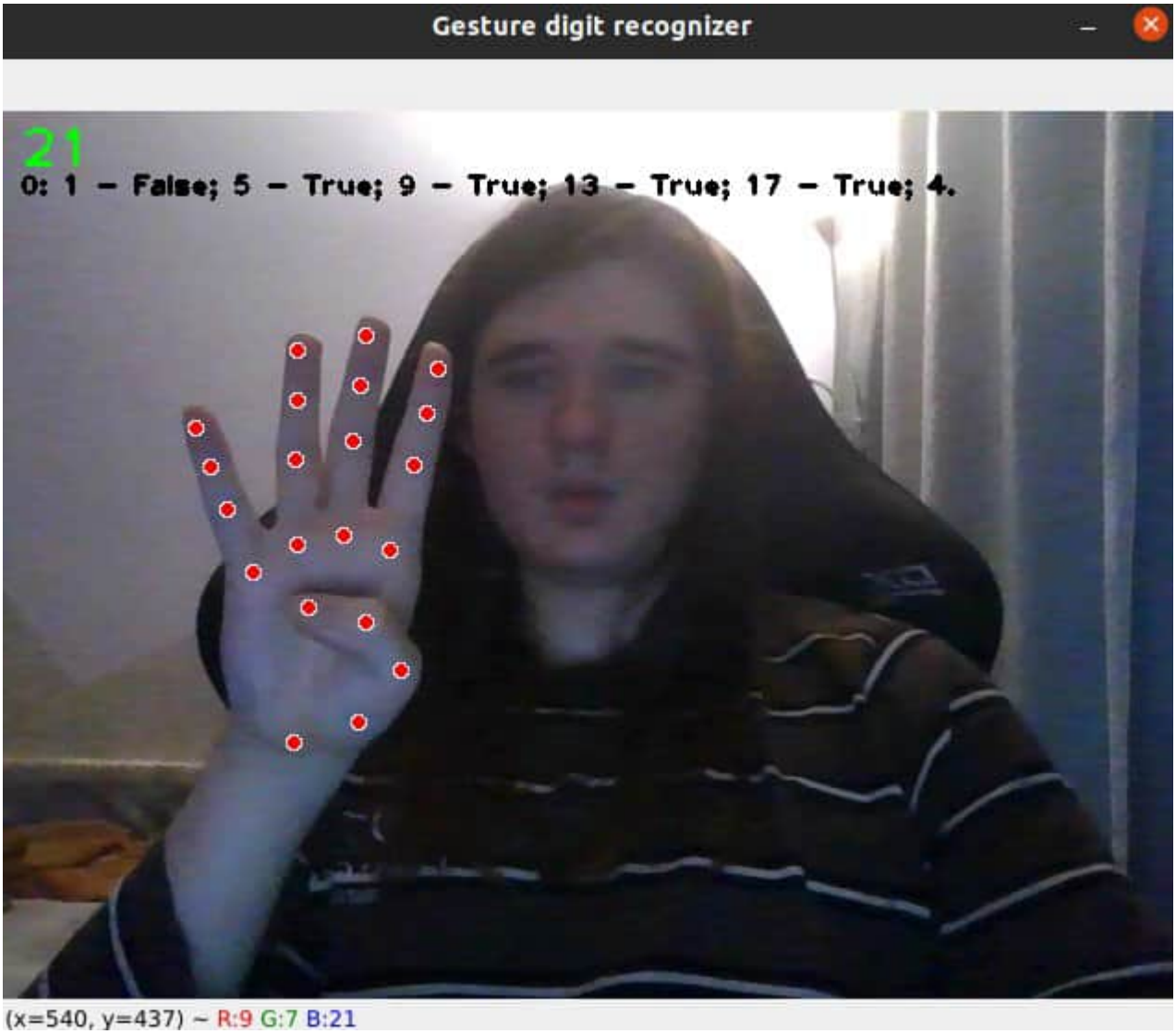

Рис. 2: Приклад роботи програми збору тренувальних даних

Після того, як користувач натиснув *Esc*, викликається функція **clean_train_data**. У цій функції для кожного рядка з файлу з тренувальними даними ми видаляємо наступні **N** рядків, де N – це число, вказане в колонці кадрів. Нам потрібно видалити ці дані, оскільки за 1 секунду програма зчитує $\tilde{2}5$ кадрів. Ці дані є дуже схожими між собою і, відповідно, можуть негативно вплинути на модель, під час її тренування.

Зібравши дані, користувачу потрібно їх перевірити. Це можна зробити за допомогою функції **check_images**. Ця функція показує усі зображення у папці по черзі. Якщо зображення неправильно класифіковане, на ньому невірно розставлені лендмарки, або користувач вважає, що його краще видалити, тоді потрібно натиснути **Y**. В протилежному випадку треба натиснути **N**. Це позначить рядок у файлі та зображення на видалення. Після цього користувачу потрібно запустити функцію **clean_checked_data**. Ця функція видалить усі позначенні зображення, а ті, що залишилися, обріже по периметру руки.

2.2 Тренування моделей

Для тренування обох моделей керування використовувався один модуль, тренувальні дані різні, проте алгоритм їхньої оборбки той самий. Спочатку ми зчитуємо файл у дата фрейм. Потім для кожного рядка виконуємо наступні операції: перетворюємо масив лендмарок, зчитаних з файлу, у масив координат, знаходимо мінімальні та максимальні x, y, z координати, нормалізовуємо координати згідно максимальних та мінімальних значень. У масив вхідних даних додаємо масив лендмарок, а в масив класів – клас. Також ми аугментуємо дані для того, щоб отримати їхню більшу кількість для тренування. Для аугментації ми використовуємо 3 функції: *mirror, flip, rotate*. mirror повертає дзеркальне відображення масиву лендмарок. flip повертає масив лендмарок, обернутих навколо центру руки. rotate повертає масив лендмарок на n градусів. Під час аугментації ми використовуємо функцію rotate, щоб повернути руку на n° де $n \in [-10, 0) \bigcup (0, 10]$

Матриця невідповідностей – це таблиця, що демонструє ефективність алгоритму класифікації. У цій таблиці перший рядок – це класи передбачень, а перша колонка – дійсні класи. По діагоналі ми маємо кількість правильно класифікованих об'єктів, усі інші – класифіковані не правильно.

Class	0	1	2	3	4	5
0	96	2	5	20	1	7
1	4	124	23	30	3	1
2	6	18	92	26	4	1
3	15	14	52	448	52	18
4	3	1	5	24	118	1
5	7	0	0	13	3	163

Табл. 1: Матриця невідповідностей для моделі керування жестами без використання аугментації

З таблиці 1 видно, що точність класифікації цієї моделі склала 74%

Class	0	1	2	3	4	5
0	115	1	0	14	0	1
1	3	161	7	10	4	0
2	1	4	125	15	2	0
3	3	4	14	545	27	6
4	0	0	2	7	143	0
5	0	0	0	10	0	176

Табл. 2: Матриця невідповідностей для моделі керування жестами з використанням аугментації

Точність моделі на таблиці 2 склала 90%

Ці моделі тренувалися на однакових вхідних даних. Єдина відміність полягає в тому, що для другої моделі дані було аугментовано. Ми могли досягнути подібного результату, якби мали більше вхідних даних. Аугментація дозволяє уникнути процесу збору даних. Проте, навіть якби ми зібрали більше даних, то могли б їх аугментувати для покращення точності моделі.

Перевірити роботу моделі можна за допомогою функції **gesture_recognizer**. У цій функції ми зчитуємо зображення з камери та передаємо його у функцію **process** бібліотеки mediapipe. З цієї функції ми отримуємо лендмарки та нормалізуємо їх згідно максимальних та мінімальних значень. Нормалізовані лендмарки передаємо моделі, яка класифікує їх. З метою демонстрації роботи інтерфейсу, програма виводить на екран жест та час, витрачений на роботу моделі (див. рис. 3).

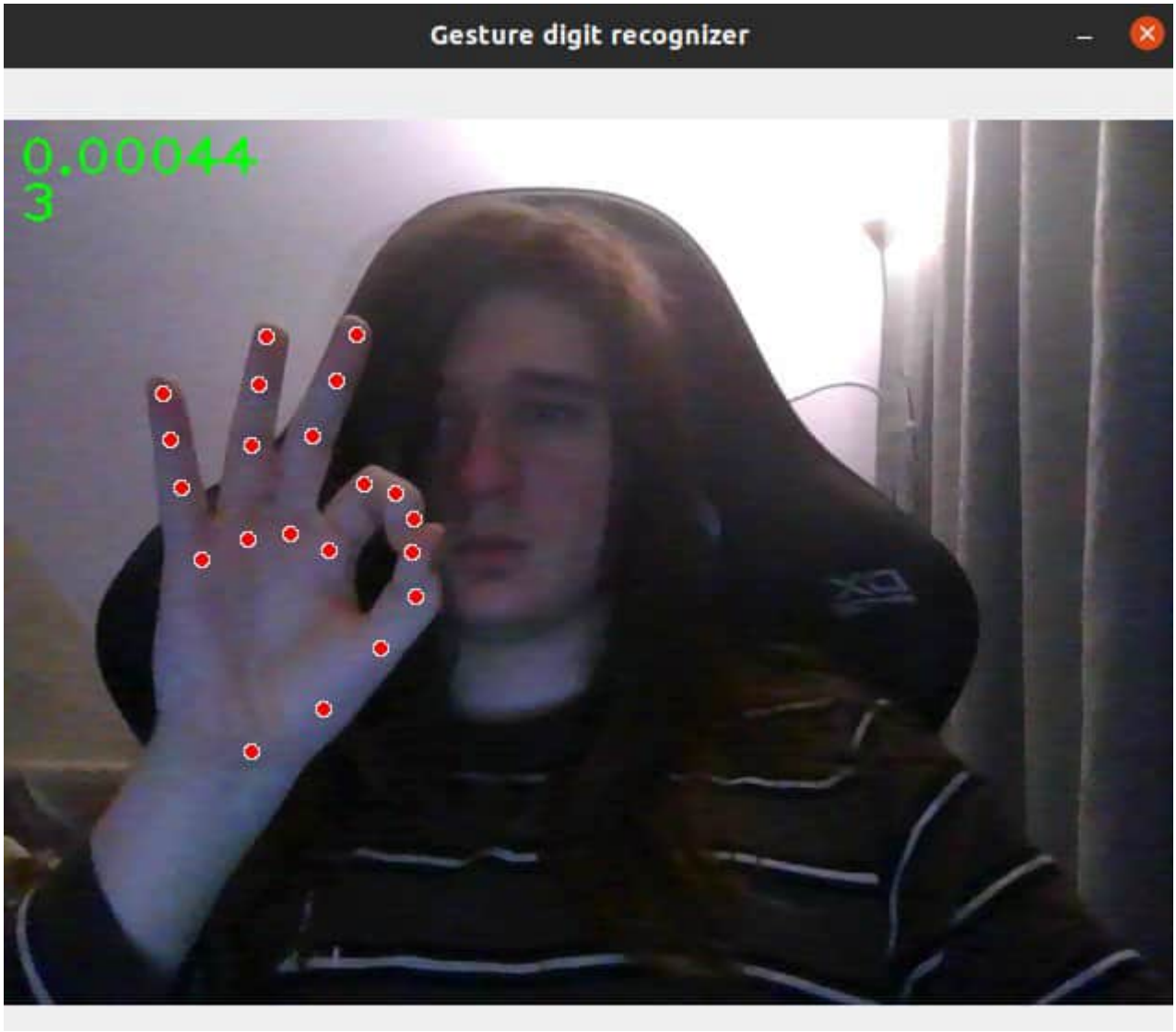

Рис. 3: Приклад роботи програми (класифікує 3 підняти пальці)

Варто зазначити, що моделі, треновані на даних без z координати, мали гіршу точність ніж моделі, треновані на даних із z координатою.

З результатів тренування моделей видно, що на точність моделі впливає кількість та якість тренувальних даних. Для покращення точності моделі на етапі тренування можна використовувати аугментацію даних. У розпізнаванні жестів z координата, освтілення та якість зображення відіграють важливу роль.

Модель керування жестами було натреновано на датасеті з жестами для мови глухонімих[4]. А модель керування рухами було натреновано на двох датасетах[3, 4], оскільки при тренуванні лише на першому датасеті модель не розпізнавала жест закритої долоні правильно. Також під час тренування другої моделі модуль було доповнено кодом для підбору гіперпараметрів для кращого тренування моделі.

2.3 Реалізація першої програми

За основу взято код програми для безконтактних інтерфейсів. В програму було інтегровано модель керування жестами. Модель класифікує 6 жестів: 0, 1, 2, 3, 4, 5. Модель розпізнає кількість піднятих пальців і класифікує їх, як відповідний жест. Ця модель використовувалась лише під час роботи з меню введення чисел (див. рис. 4). Для навігації меню програми використовувалась модель натренована на інших жестах (див. рис. 5).

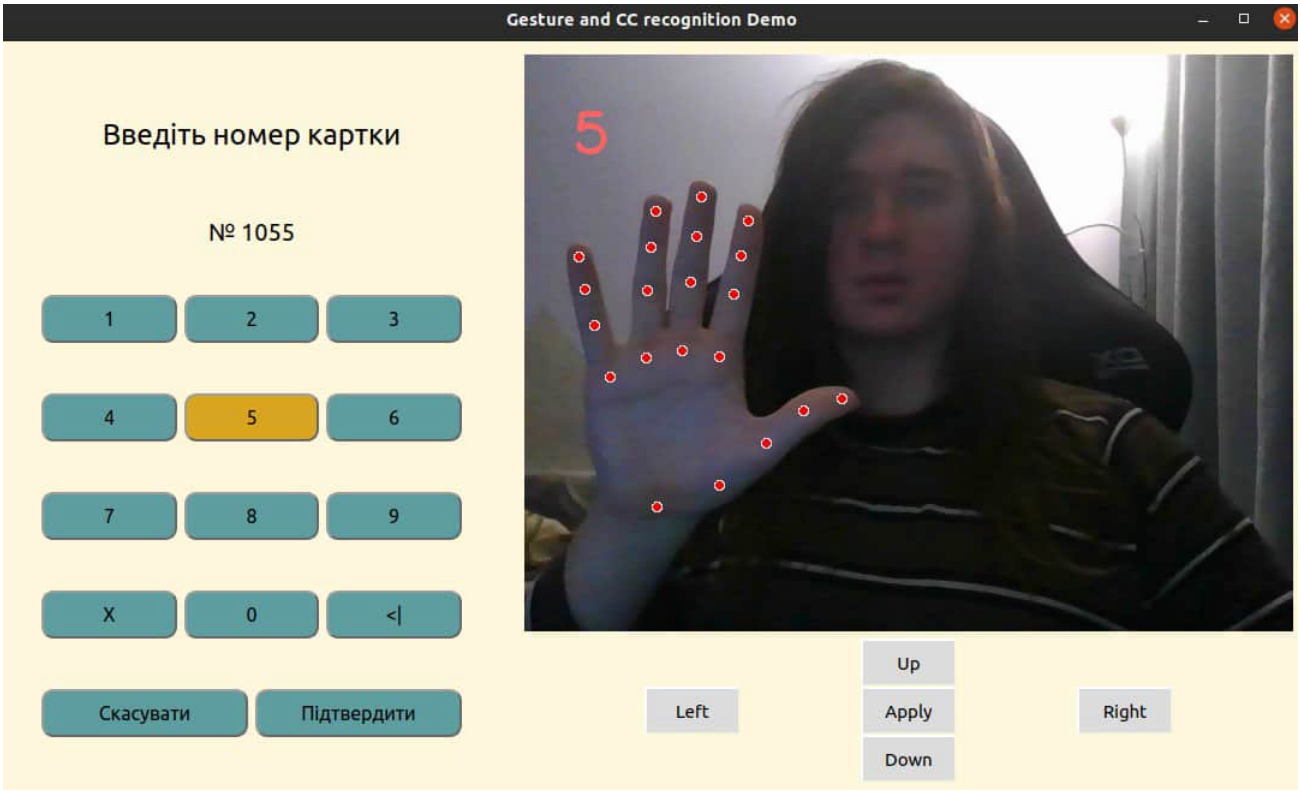

Рис. 4: Приклад роботи моделі у меню вводу чисел

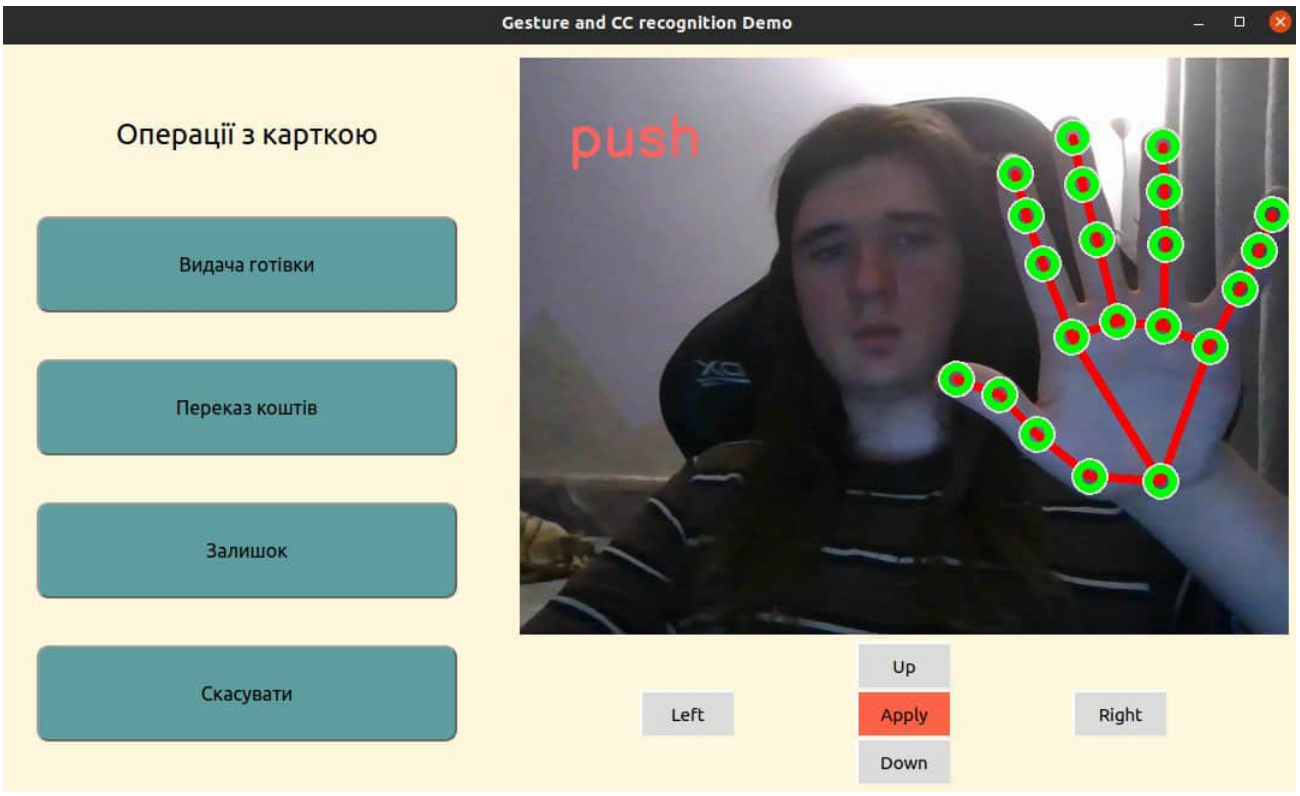

Рис. 5: Приклад роботи моделі навігації UI

На рис. 4 та рис. 5 використовується ідентичний жест, проте дві моделі класифікують його по різному.

Class	0	1	2	3	4	5
0	131	0	0	0	0	1
1	0	184	0	0	1	0
2	0	0	146	1	0	0
3	0	0	2	594	1	2
4	0	0	0	0	152	0
5	1	0	0	0	0	185

Табл. 3: Матриця невідповідностей моделі, використаної у меню вводу чисел

Точність моделі, використаної у меню вводу чисел, складає 99.4% (табл. 3).

Під час роботи з меню вводу чисел використовується система підрахунку кадрів, коли на екрані жест не змінюється протягом 5-ти кадрів, він реєструється як команда. Без цієї системи програма кожного кадру виконувала команду для відповідного жесту. Також зображення отримане з openCV конвертується з формату BGR у RGB перед тим, як воно передається mediapipe. Це потрібно для кращої роботи medipipe, оскільки в бібліотеки виникали труднощі

з розпізнаванням двох долонь одночасно.

Опис жестів та відповідних команд, які розпізнає програма:

Жест	**Команда GUI**
рука показує 0	0
піднятий 1 палець	1
підняті 2 пальці	2
підняті 3 пальці	3
підняті 4 пальці	4
підняті 5 пальців	5
Кількість піднятих пальців на обох руках дорівнює 6	6
Кількість піднятих пальців на обох руках дорівнює 7	7
Кількість піднятих пальців на обох руках дорівнює 8	8
Кількість піднятих пальців на обох руках дорівнює 9	9
На обох руках піднято 5 пальців	Прийняти
обидві руки показують 0	Видалити

Варто зазначити, що модель для розпізнавання чисел можна використовувати для навігації, проте тоді навігація інтерфейсом не була б інтуїтивною. Хоча модель для розпізнавання чисел й має тільки 6 класів, її можна використовувати не лише для вводу цифр, а й для вводу різноманітних команд. Наприклад, у програмі використовується дві команди: видалити (рис. 6) та прийняти, які задаються комбінаціями (0, 0) та (5, 5) відповідно.

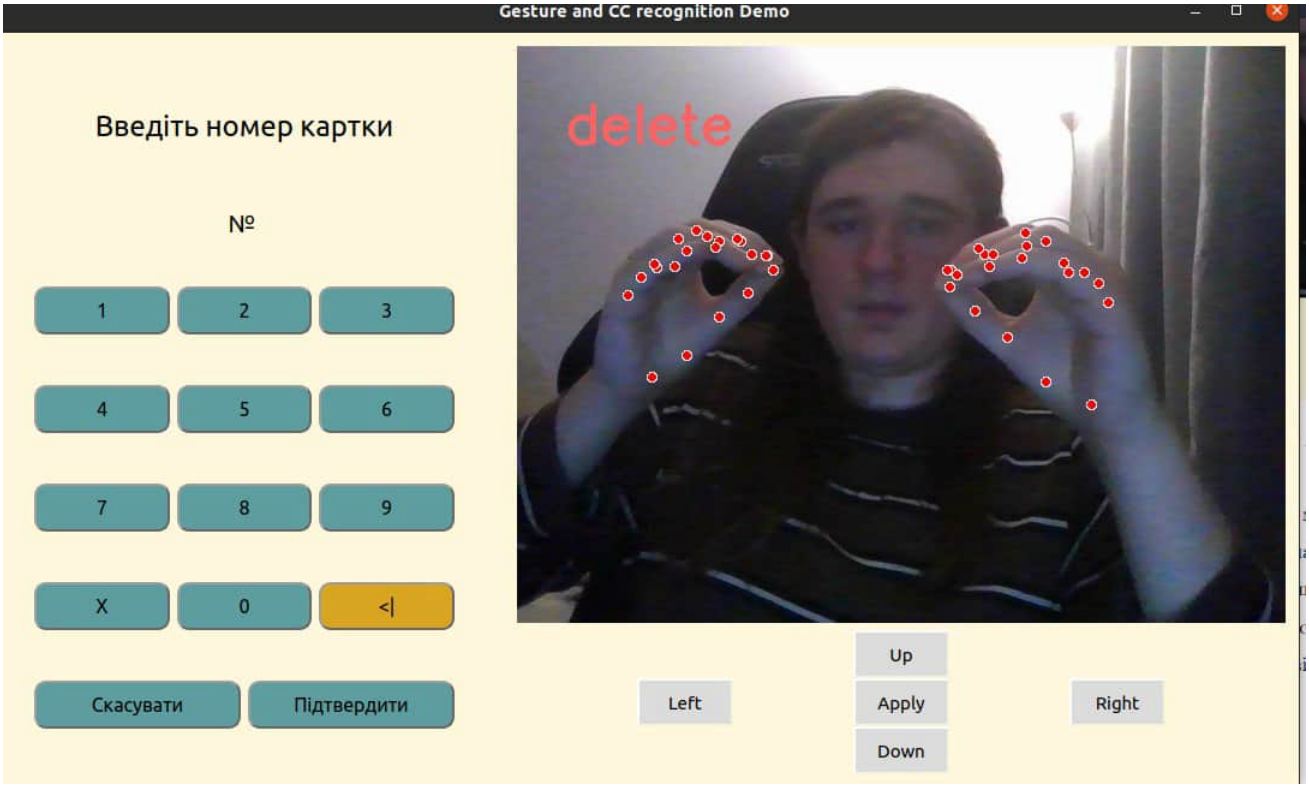

Рис. 6: Команда видалити

Оскільки користувачу потрібна лише одна долоня, щоб задати цифру від 0 до 5, то комбінації (0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (0, 5) можна використовувати, щоб задавати команди. Також для цього можна застосовувати і комбінацію (5, 5), адже $5 + 5 = 10$, а 10 – це число, а не цифра. Тож для цієї моделі можна запрограмувати 7 команд.

2.4 Реалізація другої програми

Матриці невідповідностей для цієї моделі немає, оскільки вона класифікує лише два класи, тому не була б інформативна. Точність цієї моделі під час тренування склала 99.6%.

Діаграма роботи програми є на рис. 7. Модель використовується для розпізнання жестів відкритої та закритої долоні. Якщо долоня закрита більше 2-ох, але менше 6-ти кадрів поспіль, то коли користувач відкриває її, відбувається натискання лівої кнопки миші функцією **click** бібліотеки PyAutoGui. Рішення не реєструвати клік, якщо долоня була закрита менше

3 кадрів було прийнято з метою запобігання небажаних викликів функції у випадку, коли модель помилково розпізнала відкриту долоню як закриту. Якщо долоня закрита більше 5 кадрів, то цей факт реєструється як натиснута ліва кнопка миші функцією **mouseDown**. Цей варіант використовується, якщо користувач, наприклад, хоче перетягнути якийсь елемент. Коли користувач відкриває долоню, викликається функція **mouseUp**. 3 кадри для реєстрації кліку було встановлено на основі результатів проведеного мною тестування. За бажанням це значення можна змінити.

На ідею такого безконтакного інтерфейсу мене надихнули смартфони, де значна частина операцій відбувається за допомогою простого дотику, або перетягування. Якщо для реєстрації кліку використовується модель керування рухами, то для того щоб знати, де натиснути ЛКМ, ми опрацьовуємо дані, отримані з mediapipe. Ми зчитуємо дані про зап'ястя – 0 лендмарку, за нею отримуємо інформацію про позицію долоні на зображенні. Маючи координати зап'ястя на зображені, ми переводимо їх в координати на екрані, використовуючи прості формули:

$$wrist_x = sensitivity_x * \frac{screenWidth}{imageWidth} * x - borderPixels_x$$

$$wrist_y = sensitivity_y * \frac{screenHeight}{imageHeight} * y - borderPixels_y$$

Тут вираз $\frac{screenWidth}{imageWidth} * x$ відповідає за трансляцію координат зі зображення на екран. Параметри sensitivity та borderPixels введені для покращення досвіду користувача при взаємодії з елементами, що розташовані по краях екрану. У моєму випадку для sensetivity ці параметри дорівнюють 1.7 та 1.4 для x та y відповідно, й 700 та 400 для borderPixels. Отримавши координати зап'ястя, ми порівнюємо їх з попередніми координатами з минулого кадру. Якщо відстань між цими точками не перевищує 40, то ми не оновлюємо позицію курсора миші. Ця перевірка була імплементована, оскільки лендмарка зап'ястя, що повертається з mediaріре, кожного разу має дещо інші координати, навіть якщо долоня не рухалась. Отримані координати передаються у функцію PyAutoGUI **moveTo**.

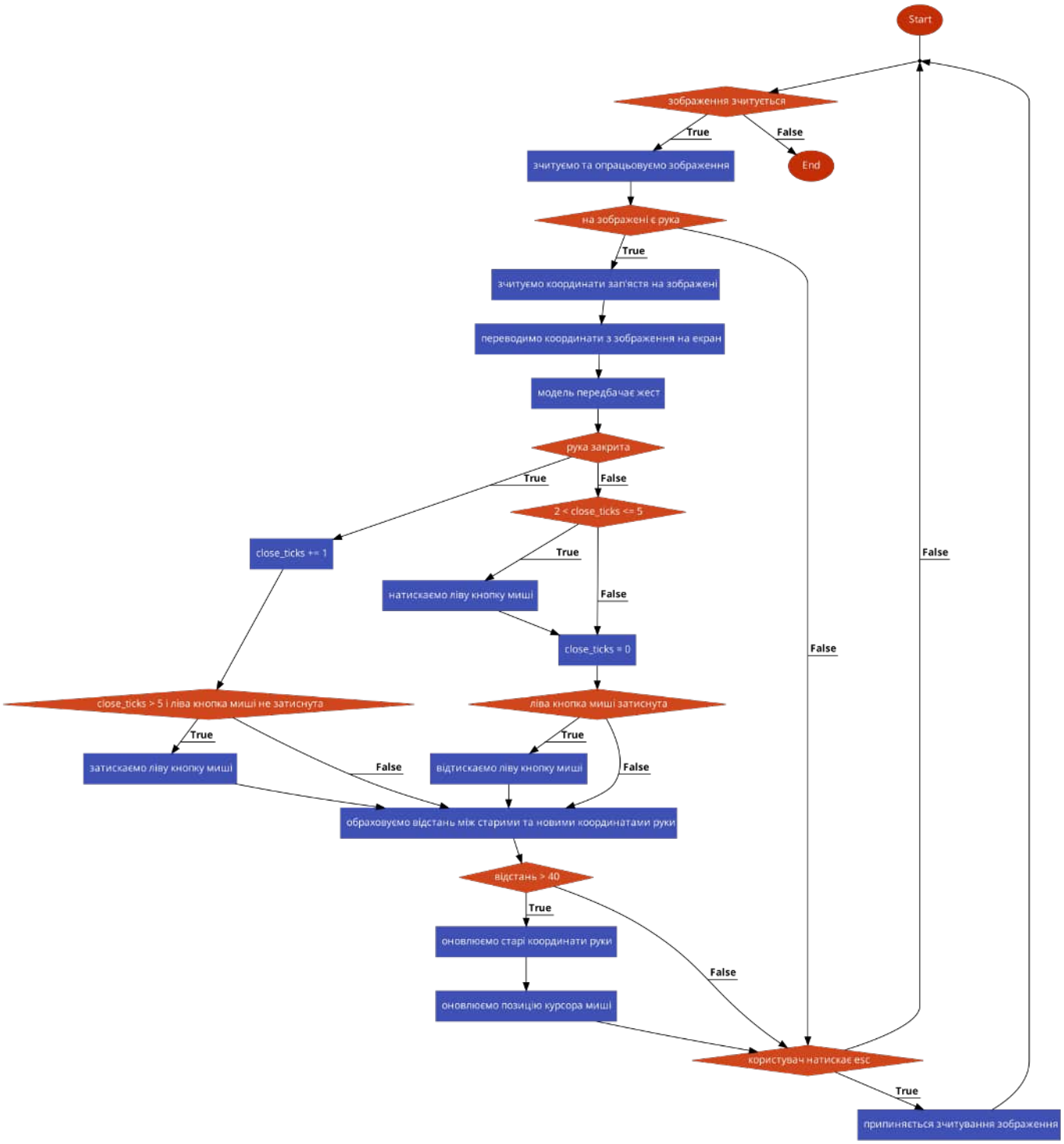

Рис. 7: Діаграма роботи програми

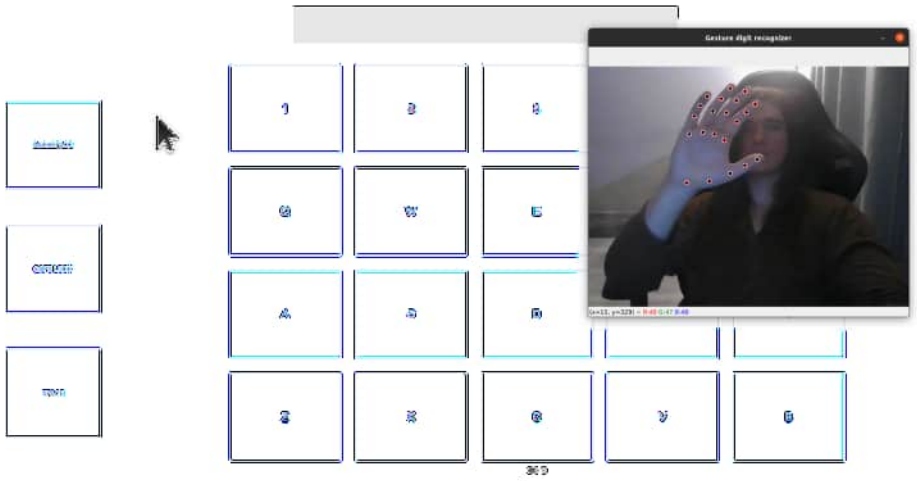

Рис. 8: Приклад роботи програми

На рис. 8 роздільна здатність екрану 1920x1080, розмір зображення 640x480. Координати (203; 243) на зображенні було переведено у координати (335.3; 365.4) на екрані.

Також було створено UI, в якому було імплементовано наступні компоненти: головне меню, клавіатура, цифрова клавіатура.

Рис. 9: Цифрова клавіатура

На рис. 9 можна побачити, що усі компоненти з якими можна взаємодіяти було збільшено для полегшення роботи з інтерфейсом.

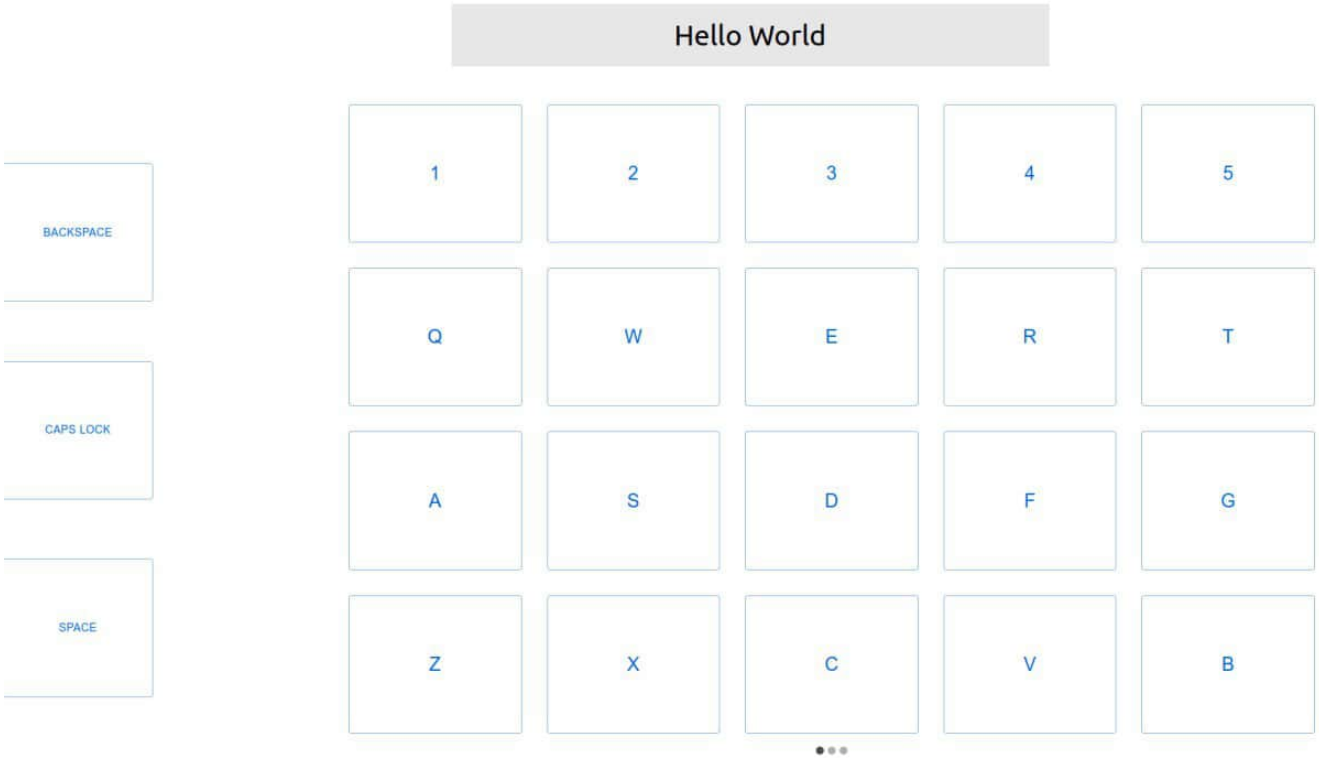

Рис. 10: Клавіатура

Оскільки на простій клавіатурі клавіш більше, ніж на цифровій, то її складніше розмістити на екрані. Є два варінти, щоби це зробити: перший – відобразити всю клавіатуру за рахунок зменшення розміру клавіш, другий – розбити клавіатуру на декілька частин та збільшити клавіші. Хоч обидва варінти робочі, у своїй імплементації я вибрав другий, оскільки з ним легше працювати.

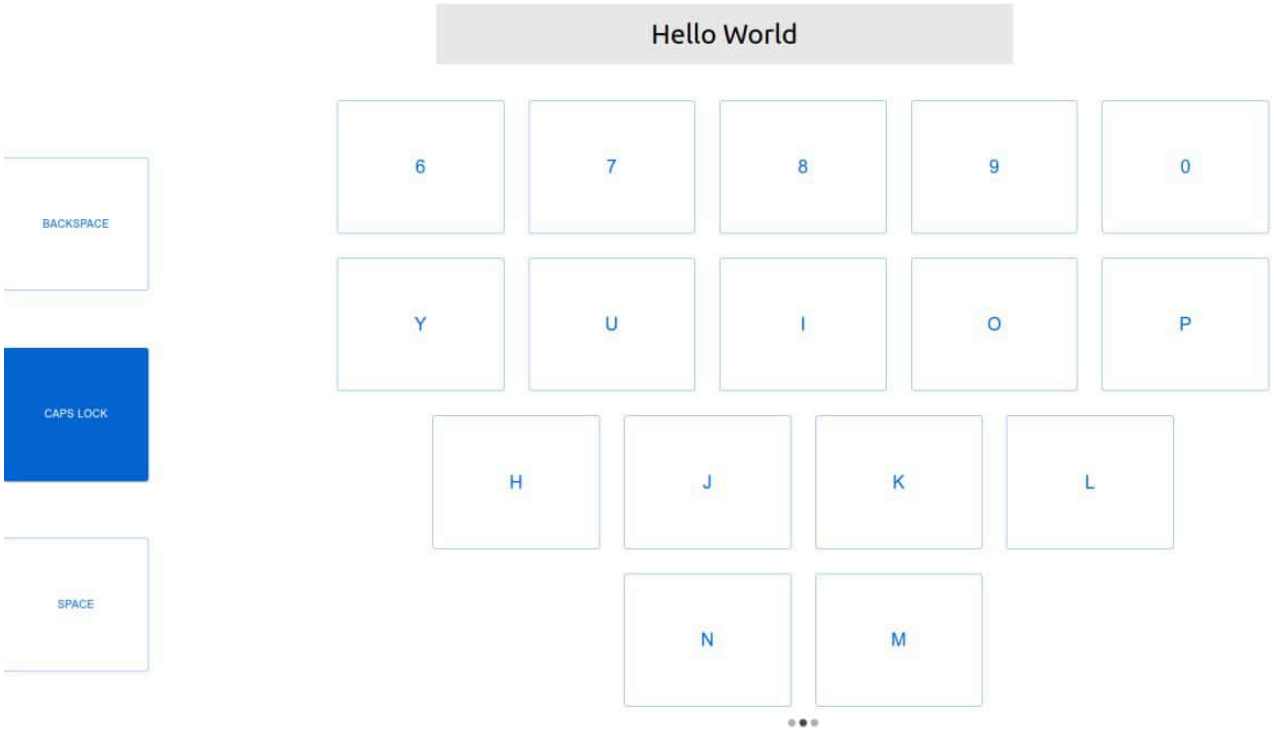

Рис. 11: Клавіатура – II частина

Клавіатуру було розбито на три частини за схемою, поданою на рис. 10 - 12. Тут перші дві частини – це вертикально розділена клавіатура, а третя – спецсимволи.

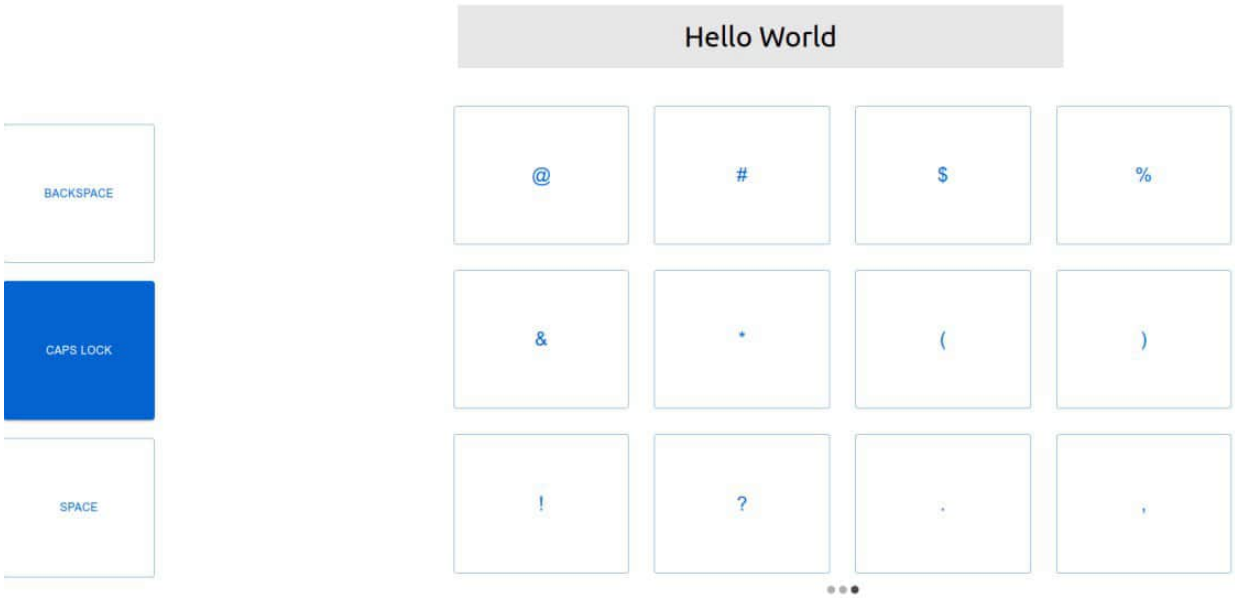

Рис. 12: Клавіатура – спецсимволи

Користувач переключається між частинами клавіатури, перетягуючи її вправо або вліво. Щоб досягнути цього ефекту, я використав React Material UI Carousel[11]

2.5 Порівняння методів

Порівнюючи запропоновані способи керування безконтактним інтерфейсом користувача – керування жестами та керування рухами – можна сказати, що перший, напевне, є кращим для простіших інтерфейсів. Оскільки програмі потрбіно тільки розпізнати жест, то робота з інтерфейсом відбувається швидше. Проте для масштабніших програм цей метод навряд чи підходить. Хоча його і можна інтегрувати у ці програми кількома шляхами:

1. підлаштувати UI під уже існуючу модель
2. розпізнавати комбінації жестів, як окремі команди
3. збільшити кількість жестів, які розпізнає модель

Зазначимо, що жоден із цих варіантів не є оптимальним. Для прикладу, розглянемо меню набору тексту. Керування жестами добре підходить для набору числових значень, але не рядків символів. Для першого варіанта, візьмемо модель, що розпізнає 6 жестів. Нам потрібна клавіатура для набору тексту, вона може бути імплементована різними методами, проте незалежно від реалізації, щоби ввести один символ потрібно 2 жести. Для другого випадку знову розглянемо модель з 6 класами. Припустимо, що програма виконує 42 команди (6 без комбінацій + 6*6 комбінацій (у випадку, що програма розрізняє ліву та праву долоні)). Цей підхід був би занадто складним для опанування. У третьому випадку все просто, програма розпізнає усі необхідні жести для введення тексту, проте це відбувається не інтуїтивно. До того ж сумнівно, що багато людей захочуть опановувати мову глухонімих.

Проте при керуванні рухами цих проблем не виникає, оскільки програма намагається імітувати сенсорні екрани. Цей метод дозволяє працювати зі звичними нам UI. Також моделі достатньо класифікувати 2 жести для повноцінної роботи програми. Звісно, такий метод не можна вважати найзручнішим способом керування, тож традиційні методи взаємодії з девайсами він не замінить, але він достатньо інтуїтивний для виконання звичних нам завдань.

Вибір моделі безконтактного керування графічним інтерфейсом залежить від самого інтерфейсу, з яким нам потрібно працювати. Якщо в програми простий інтерфейс, то краще використовувати модель керування жестами, оскільки вона швидша й точніша за модель керування рухами. Якщо ж інтерфейс програми є складнішим, то краще використовувати

керування рухами, адже така модель буде простішою та інтуїтивнішою за керування жестами.

	Метод керування жестами	**Метод керування рухами**
Швидкодія користування	+	+-
Складність імплементації	-	+
Зручність користування	+	+-
Інтеграція в різні інтерфейси	-	+

3 Висновки

У ході виконання роботи отримано такі результати:

1. Створено 2 методи безконтактного керування ПК.
2. Створено графічний інтерфейс користувача для тестування методів безконтактного керування.
3. Проведено тестування цих 2 методів за допомогою інтерфейсу.
4. В результаті тестувань було вдосконалено методи керування та інтерфейс користувача.

Бекзонтактні інтерфейси – це перспективна сфера, яка потребує подальшого розвитку.

СПИСОК ВИКОРИСТАНИХ РЕСУРСІВ

1. Іванов С. Про розпізнавання окремих ознак частин тіла людини звикористанням обмежених обчислювальних ресурсів / С. Іванов, А. Музичук // Вісник ЛНУ. Серія прикл. матем. та інформ. – 2021.– С. 103-114. - Available from: http://publications.lnu.edu.ua/bulletins/index.php/ami/article/view/11337/11788
2. Pandas documentation - Available from: https://pandas.pydata.org/docs/
3. hand-sign-images Dataset - Available from: https://www.kaggle.com/datasets/ash2703/handsignimages
4. Sign Language Digits Dataset - Available from: https://www.kaggle.com/datasets/javaidahmadwani/sign-language-digits-dataset
5. NumPy documentation - Available from: https://numpy.org/doc/stable/
6. Scikit-learn - Available from: https://scikit-learn.org/stable/
7. OpenCV - Available from: https://opencv.org/
8. Mediapipe hands - Available from: https://google.github.io/mediapipe/solutions/hands
9. PyAutoGUI - Available from: https://pyautogui.readthedocs.io/en/latest/
10. Ivanov S.A. Touchless GUI with CV and ML: demo. - Available from: https://www.dropbox.com/s/s2z2hswuy97pgat/gestureRecognition.mp4?dl=0
11. React Material UI Carousel - Available from: https://learus.github.io/react-material-ui-carousel/
12. Material UI - Available from: https://mui.com/
13. Microsoft kinect - https://en.wikipedia.org/wiki/Kinect
14. Google Home - https://home.google.com/welcome/
15. Leap Motion Controller - https://www.ultraleap.com/product/leap-motion-controller/