**Київський національний університет імені Тараса Шевченка**

Факультет інформаційних технологій

Кафедра програмних систем і технологій

**УДК 004.942**

*На правах рукопису*

# ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

**Тема:** “ Програмне забезпечення ідентифікації якості виконання вправ щодо східних єдиноборств ”

*(назва згідно з наказом ректора)*

**Спеціальність –** 121 “Інженерія програмного забезпечення”

## ПОЯСНЮВАЛЬНА ЗАПИСКА

**БР.ІПЗ -**

*(позначення)*

**Студент**

**ІПЗ-41_____/Андрій ЖОВНІРСЬКИЙ/**
(шифр групи) (підпис) (дата)(розшифровка підпису)

**Науковий керівник**

**д.т.н., проф._____/Віктор ШЕВЧЕНКО/**
(посада) (підпис) (дата) (розшифровка підпису)

**Консультант з питань нормоконтролю**

**фахівець_____/Тамара ЧАПОВСЬКА/**
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту
з питань нормоконтролю
**Завідувач кафедри**

**д.т.н.,проф._____/Олексій БИЧКОВ/**
(посада) (підпис) (дата) (розшифровка підпису)

Київ - 2021

**Київський національний університет імені Тараса Шевченка**

Факультет інформаційних технологій
Кафедра програмних систем і технологій
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 121 "Інженерія програмного забезпечення"

ЗАТВЕРДЖЕНО
Зав. кафедри _програмних систем і технологій_

_____ (_Олексій БИЧКОВ_)

(підпис) (прізвище та ініціали)

# ЗАВДАННЯ

## НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Жовнірському Андрію Ігоровичу

(прізвище, ім'я, по-батькові)

**1. Тема бакалаврської роботи " Програмне забезпечення ідентифікації якості виконання вправ щодо східних єдиноборств "**

керівник проекту (роботи) Шевченко Віктор Леонідович, д.т.н., професор

затверджені наказом вищого навчального закладу від **"** _ " _____ 202_ р. №_____

**2. Строк подання студентом роботи** _____ 202_ р.

**3. Вихідні дані до проекту (роботи)** Теоретичні концепції та формальні моделі побудови та функціонування інформаційних та програмних технологій певного класу

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

1. Аналіз існуючих методів пошуку руки на зображенні.
2. Розробити алгоритм знаходження руки на зображенні.
3. Розробити алгоритм ідентифікації якості виконання вправ.
4. Виконати програмну реалізацію.

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

1. Типи датчиків розпізнавання рухів (рис. 1.1 , ст. 13)
2. Алгоритм SIFT для ідентифікації жестів (рис. 2.1(а), ст. 20)
3. Приклад дескриптора функції SIFT (рис. 2.1(б), ст. 20)
4. Відображення шуму при засвітленні (рис. 3.1, ст. 23)
5. Найпоширеніші кольори у вигляді RGB кортежів (рис. 3.6, ст. 29)

6. Консультанти розділів проекту (роботи)

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| Методи аналізу даних отриманих з зображень | Шевченко Віктор Леонідович | | |

**7. Дата видачі завдання** _____

**Керівник**_____(Віктор ШЕВЧЕНКО)

**Завдання прийняв до виконання** _____ (Андрій ЖОВНІРСЬКИЙ)

# КАЛЕНДАРНИЙ ПЛАН

| **№ п/п** | **Назви етапів бакалаврської роботи** | **Строк виконання етапів роботи** | **Примітка** |
|---|---|---|---|
| 1 | Підбір і вивчення літератури | | виконано |
| 2 | Аналіз існуючих методів та алгоритмів | | виконано |
| 3 | Розробка алгоритмів | | виконано |
| 4 | Програмна реалізація розроблених алгоритмів | | виконано |
| 5 | Затвердження пояснювальної записки роботи завідувачем кафедри | | виконано |

**Студент – бакалавр**_____(Андрій ЖОВНІРСЬКИЙ)

**Керівник роботи**_____(Віктор ШЕВЧЕНКО)

# АНОТАЦІЯ

**Випускна кваліфікаційна бакалаврська робота:** с. 68, рис. 18, табл. 1, додат. 6, джерела 15.

**Тема:** Програмне забезпечення ідентифікації якості виконання вправ щодо східних єдиноборств.

**Об'єкт дослідження:** ідентифікації якості виконання вправ щодо східних єдиноборств.

**Мета роботи:** розробка ПЗ для ідентифікації якості виконання рухів на основі зображень у реальному часі, допомога у навчанні правильного виконання вправ, а також допомога у перевірці виконання вправ на іспитах чи змаганнях за допомоги ПЗ.

**Предмет дослідження:** технології розпізнавання об'єктів та рухів, досліджуються методи аналізу даних із зображень.

**Результати дослідження:**

Досліджено існуючі концепції розпізнавання жестів, зокрема засновані на зображеннях. Запропоновано власний підхід щодо ідентифікації руки на зображенні, а також власний алгоритм ідентифікації якості виконання вправи, щодо східних єдиноборств.

**Висновок**

В результаті досліджень було розроблено алгоритм ідентифікації якості виконання вправ, що в свою чергу заснований на власному методі пошуку дефектів. Розроблено та реалізовано алгоритм знаходження руки. Реалізовано ПЗ, що виконує розпізнавання руки та правильність виконання вправи рукою щодо східних єдиноборств у реальному часі.

АЛГОРИТМ, КЛАСТЕР, СХІДНІ ЄДИНОБОРСТВА, ІДЕНТИФІКАЦІЯ ЯКОСТІ, ВПРАВА, ЗОБРАЖЕННЯ

# SUMMARY

**Final qualifying bachelor's thesis:** pages 68, figures 18, tables 1, appendices 6, sources 15.

**Topic:** Software for evaluation of performance accuracy in martial arts techniques.

**Object of research:** evaluation of performance accuracy in martial arts techniques.

**Purpose:** software development to identify the quality of performance of movements based on real-time images, assistance in learning the correct execution of exercises, as well as assistance in checking the performance of exercises in exams or competitions with the help of software.

**Subject of research:** technologies of recognition of objects and movements, methods of analysis of data from images are investigated.

**Research results:**

Existing concepts of gesture recognition, in particular based on images, are investigated. The own approach concerning identification of a hand on the image, and also own algorithm of identification of quality of performance of an exercise, concerning east martial arts is offered.

**Conclusion:**

As a result of research the algorithm of identification of quality of performance of exercises which in turn is based on own method of search of defects was developed. An algorithm for finding a hand has been developed and implemented. Implemented software that performs hand recognition and evaluation quality of hand exercises concerning martial arts, in real time.

ALGORITHM, CLUSTER, EASTERN MARTIAL ARTS, EVALUATION QUALITY, EXERCISE, IMAGE

# АНОТАЦИЯ

**Выпускная квалификационная бакалаврская работа:** с. 68, рис. 18, табл. 1, доп. 6, источ. 15.

**Тема:** Програмное обеспечение идентификации качества выполнения упражнений касаемо восточных единоборств.

**Объект исследования:** Идентификация качества выполнения упражнений касаемо восточных единоборств.

**Цель работы:** разработка ПО для идентификации качества выполнения упражнений на основе изображений в реальном времени, помощь в обучении правильному выполнению упражнений, а также помощь в проверке выполнения упражений на экзаменах или соревнованиях с помощью ПО.

**Предмет исследования:** технологии распознавания обьектов и движений, исследуються методы анализа данных из изображений.

**Результаты исследования:**

Исследовано существующие концепции распознавания жестов, в частности основанные на изображениях. Предложено собственный подход касаемо идентификации руки на изображении, а также собственный алгоритм идентификации качества выполнения упражнений касаемо восточных единоборств.

**Вывод:**

В результате исследований было разработано алгоритм идентификации качества выполнения упражнений, который в свою очередь основан на собственном методе поиска дефектов. Разработано и реализовано алгоритм поиска руки. Реализовано ПО, которое выполняет распознавание руки и правильность выполнения упражнения рукой касаемо восточных единоборств в реальном времени.

АЛГОРИТМ, КЛАСТЕР, ВОСТОЧНЫЕ ЕДИНОБОРСТВА, ИДЕНТИФИКАЦИЯ КАЧЕСТВА, УПРАЖНЕНИЕ, ИЗОБРАЖЕНИЕ

# ЗМІСТ

Стр.

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ - програмне забезпечення

ІЧ - Інфрачервоний

HSV - Hue, Saturation, Value

RGB - Red, Green, Blue

MIT - Массачусетський технологічний інститут

# ВСТУП

## Актуальність роботи

Наразі більшість видів спорту йде назустріч технологіям у різних аспектах, як, наприклад, технологія фіксації взяття воріт у футболі, чи фіксація «уколів» у фехтуванні. Тож виникає питання: чому б не розробити щось подібне щодо східних єдиноборств?

Іспити, є важливими для тих, хто хоче постійно прогресувати та підвищувати рівень майстерності у специфічному контексті. Специфіка отримання ступеня майстерності у бойових мистецтвах зазвичай базується на системі кольорів поясів. Щоб довести свій рівень, претендент має вірно виконати так звані «форми», а вони мають безліч деталей, і деякі екзаменатор може не помітити, що дозволяє говорити про незаслужене визнання майстерності. Задля уникнення таких випадків може слугувати дана робота. Нічого подібного, на момент розробки, раніше не було реалізовано. Дане ПЗ є інноваційним та важливим кроком у світі бойових мистецтв.

Даний програмний продукт, також, має допомогти підвищити загальний рівень вмінь спортсменів та посприяти досягненню вищих результатів, адже людина не здатна ідеально побачити всі помилки у виконанні вправ, на відміну від машини.

## Порівняння роботи з відомими розв’язаннями проблеми

На момент виконання роботи, не існує жодного варіанту ідентифікації якості виконання вправ щодо східних єдиноборств. Робота є унікальною та оригінальною.

Хоча існують додатки, що виконують схожу функцію, такі як програми

щодо розпізнавання жестів, обличь, номерних знаків та ін. Але вони не є рішенням даної проблеми, адже рухи в бойових мистецтвах специфічні і мають досить багато деталей.

**Мета і задачі дослідження**

Метою бакалаврської роботи є розробка програмного забезпечення, щодо ідентифікації якості виконання вправ щодо східних єдиноборств. Допомога у навчанні правильного виконання вправ, а також допомога у перевірці виконання вправ на іспитах чи змаганнях за допомоги програмного забезпечення. ПЗ має отримувати зображення виконання вправи у реальному часі, визначати чи правильно вправа виконана.

Завдання дослідження:

1. Розглянути можливі варіанти реалізації пошуку руки на зображенні.
2. Вибрати алгоритми, які будуть використані при розпізнаванні вправ.
3. Обрати засоби розробки.
4. Розробити алгоритм знаходження рук на зображенні.
5. Розробити алгоритм розпізнавання вправ.
6. Виконати програмну реалізацію.

**Об’єктом дослідження є** ідентифікація якості виконання вправ щодо східних єдиноборств.

**Предметом дослідження** технології розпізнавання об’єктів та рухів, досліджуються методи аналізу даних із зображень.

**Методи дослідження**

Для отримання зображення використовується метод отримання за допомогою камери. Для ідентифікації якості виконання вправ використовуються

формули знаходження кутів, площ та відстаней, результати яких аналізуються за розробленим алгоритмом.

**Наукова новизна отриманих результатів**

Досліджено можливості отримання зображень в режимі реального часу та розроблено алгоритм ідентифікації якості виконання вправ щодо східних єдиноборств. Запропоновано використання методу косинуса для ідентифікації проміжків у долоні або «кулаці» та використання підрахунку проміжків, як параметр ідентифікації якості виконання вправи. Виконана програмна реалізація алгоритму.

**Практичне значення одержаних результатів**

Одержане ПЗ дозволяє отримувати зображення в реальному часі, за допомоги камери ПК. Реалізований алгоритм ідентифікації якості виконання вправ щодо східних єдиноборств дозволяє виконувати аналіз якості чотирьох вправ: «Tsuki» (кулак), «Shuto» («ребро» долоні), «Shotei» («п’ята» долоні) та «Nukite» (палець). Кількість підтримуваних вправ може збільшуватись шляхом незначного доповнення алгоритму.

Тобто дане ПЗ виконує свою основну функцію, ідентифікація якості вправ коректна і згідна очікувань.

**Особистий внесок студента**

Основним результатом є:

1. запропонований автором підхід використання підрахунку проміжків, як параметр ідентифікації якості виконання вправи;
2. приведена реалізація алгоритму що виконує підрахунок проміжків аналізуючи зображення в реальному часі.

**Апробація результатів випускної кваліфікаційної бакалаврськоїроботи**

Результати бакалаврського дослідження були представлені на

Всеукраїнському конкурсі студентських наукових робіт з інформатики та кібернетики ВНТУ 2020-2021 навчального року. Було особливо відмічено високу оригінальність ідей, самостійність роботи та практичне впровадження результатів.

**Публікації**

За результатами наукових досліджень, проведених у бакалаврській роботі, було опубліковано тези в збірнику узагальнених матеріалів конференції MSTIoE 2020-7 (7-ма Східно-Європейська конференція «Математичні та програмні технології Internet of Everything»), яка проходила на базі кафедри програмних систем і технологій Київського національного університету імені Тараса Шевченка 22-23.12.2020 року.

**Структура та обсяг роботи**

Робота викладена на 68 сторінках друкованого тексту, який складається із вступу, 3 розділів, висновків, списку використаних джерел (15 найменувань). Робота містить 1 таблицю, 18 рисунків та 6 додатків, обсягом 24 стор.

# РОЗДІЛ 1

# МЕТОДИ ЗЧИТУВАННЯ ДАНИХ ДЛЯ РОЗПІЗНАВАННЯ ВПРАВ

Перед процесом розпізнавання вправ робочі дані повинні бути зібрані датчиками. В цьому розділі датчики аналізуються на основі різних технологій захоплення даних. Як показано на Рис. 1, існують дві основні категорії збору даних: засновані на зображеннях і не засновані на зображеннях.

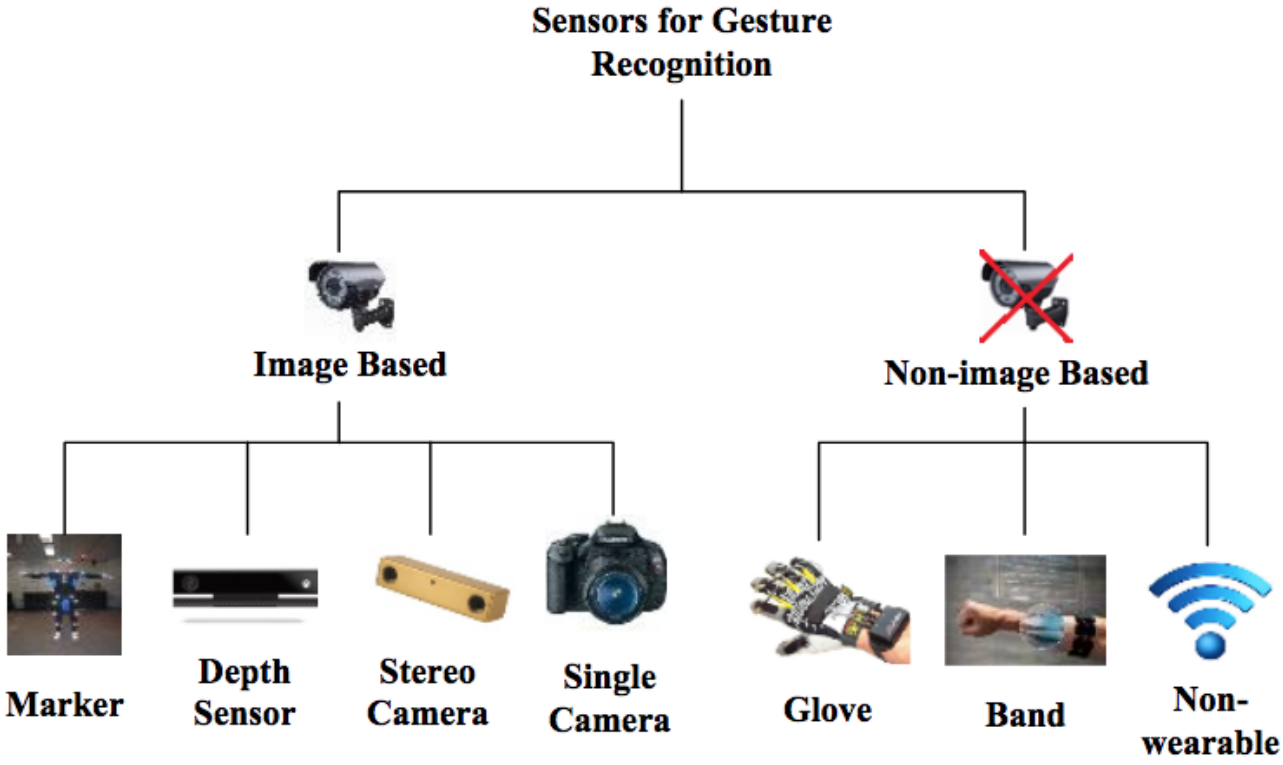



Рис. 1.1 Різноманітні типи датчиків розпізнання рухів.

## 1.1 Методи на основі отримання зображень

Технології часто надихаються природою. Ми використовуємо наші очі, щоб розпізнавати жести. Тому для роботів розумно використовувати камери, щоб «бачити»

жести. Підходи, засновані на роботі з зображеннями, далі поділяються на чотири категорії.

### 1.1.1 Маркер

У маркерному підході датчиком є оптична камера. У більшості рішень на основі маркерів користувачам необхідно носити видимі маркери. Сьогодні ми отримуємо набагато більш швидку графічну обробку в порівнянні з двадцятьма роками раніше. В результаті на ринку є безліч датчиків розпізнавання жестів.

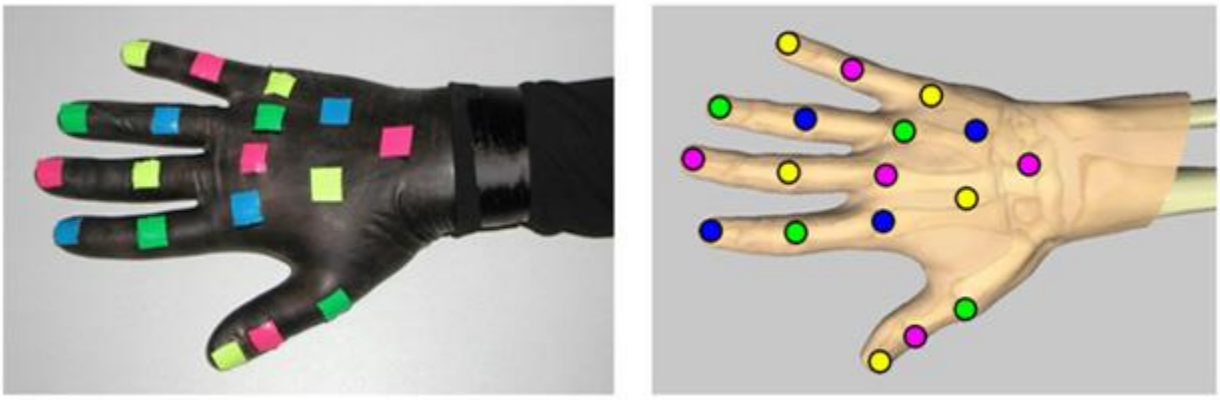


Рис. 1.2 Відслідковування руки за кольоровими маркерами

### 1.1.2 Камера

На початку 90-х років дослідники почали аналізувати жести за допомогою одиночної камери. Недоліком однокамерного підходу є обмеження кута огляду, що впливає на надійність системи. Однак в недавніх дослідженнях застосовувався однокамерний підхід до високошвидкісного розпізнаванню жестів. Система використовує датчик швидкості і спеціально розроблений процесор візуальних обчислень для досягнення високошвидкісного розпізнавання жестів.

### 1.1.3 Стереокамера

Щоб домогтися надійного розпізнавання жестів, дослідники запропонували стереопідхід до створення 3D-зору. Тут ми визначаємо стереокамерні підходи як додатки, які використовують дві оптичні камери для створення інформації про глибину

3D. Багато стереопідходів до камер відповідали аналогічному процесу. Хоча системи стереокамер поліпшили стійкість у зовнішньому середовищі, вони як і раніше страждають від таких проблем, як складність обчислень і труднощі з калібруванням.

### 1.1.4 Сенсор глибини

Останнім часом технології глибокого зондування швидко розвиваються. Ми визначаємо датчик глибини як моно-датчик глибини. Моно-датчики глибини мають ряд переваг в порівнянні з традиційними стереокамерами. Наприклад, можна запобігти проблемам настройки калібрування і умов освітлення. Крім того, вихідною інформацією датчика глибини є інформацією про глибину 3D. У порівнянні з інформацією про колір інформація про глибину 3D спрощує проблему ідентифікації жестів. Існує два типи загальних НЕ стерео-датчиків глибини: камера з «часом прольоту» (ToF) і Microsoft Kinect (або аналогічні ІЧ-датчики).

Основним принципом камер ToF є визначення часу проходження світла. У різних публікаціях були введені приклади розпізнавання жестів на основі ToF-камер. Перевагою камер ToF є більш висока частота кадрів. Обмеження камери ToF полягає в тому, що дозвіл камери сильно залежить від її світлочутливості і рефлексії.

Microsoft Kinect надає дешеве і просте рішення для розпізнавання жестів. Kinect - інфрачервоний датчик глибини. Аналогічними датчиками є ASUS Xtion Pro і Apple PrimeSense. Kinect має ІЧ-випромінювач, ІЧ-датчик і датчик кольору. Він широко використовується в сфері розваг, освіти і досліджень з великим співтовариством розробників. Доступно безліч інструментів і проектів з відкритим вихідним кодом. Деякі дослідники реалізували системи розпізнавання жестів на основі Kinect на коротких відстанях. Через обмежену дозволу, в даний час Kinect може використовуватися для розпізнавання жестів тіла і розпізнавання жестів рук на невеликій відстані. Для розпізнавання жестів рук і кистей рук на відстані більше 2 метрів краще використовувати інші підходи.

## 1.2 Методи, засновані не на зображеннях

У розпізнаванні жестів протягом довгого часу домінували датчики на основі зображень. Недавні розробки в MEMS і сенсорних технологіях значно поліпшили технології розпізнавання жестів, засновані не на зображеннях.

### 1.2.1 Рукавички

Жестові інтерфейси на основі рукавичок також використовуються для розпізнавання жестів. Зазвичай методи на основі рукавичок вимагають проводового підключення акселерометрів і гіроскопів. Однак громіздка рукавичка з проводами може викликати проблеми в HRC. Підходи на основі рукавичок також мають складності в процедурах калібрування і настройки.

### 1.2.2 Браслет

Інша безконтактна технологія використовує сенсори на браслетах. Сенсори встановлені на браслеті або аналогічних носяться пристроях. Сенсори на браслетах дозволяють використовувати бездротові технології і датчики електроміограми, що дозволяє уникнути підключення кабелів. Сенсори повинні контактувати з зап'ястям; руки і пальці користувача можуть бути вільні. Прикладом браслетного датчика є пристрій Myo.

### 1.2.3 Безконтактні пристрої

У третьому типі технологій, не пов'язаних з зображеннями, використовуються датчики, не призначені для носіння. Безконтактні датчики можуть виявляти жести без контакту з людським тілом. Google представив Project Soli, систему контролю радіолокації і розпізнавання жестів на радіочастотному спектрі (радар). Пристрій здатний розпізнавати різні жести рук на невеликій відстані. Протягом багатьох років MIT є провідним новатором в області розпізнавання жестів. Технологія електричного поля була вперше розроблена в MIT. Нещодавно Adib з MIT представив систему

WiTrack і RF-Capture, яка відстежує рух користувача з радіочастотних сигналів, відбитим від людського тіла. Система здатна захоплювати людські жести навіть з іншої кімнати через стіну з точністю до 20 см. Таким чином, технології, не придатні для носіння, є перспективними і швидко зростаючими сенсорними технологіями для розпізнавання жестів.

## 1.3 Порівняння сенсорних технологій

Таблиця 1.1 містить порівняння різних сенсорних технологій. Показано переваги та недоліки різних підходів. Зрозуміло, що жоден датчик не підходить для всіх додатків. На основі вищевказаних методів представлені два види сенсорних технологій:

Сенсори для використання всередині приміщень: датчики глибини є найбільш перспективними технологіями на основі зображень. Датчики глибини мають переваги простоти калібрування, установки і швидкості обробки даних. Існує велика спільнота розробників додатків, яке надає готові рішення.

Безконтактні сенсори є найбільш перспективною технологією серед підходів, заснованих не так на зображенні. Вони не вимагають прямого контакту з користувачами. Ця категорія сенсорів також є швидкозростаючою галуззю на ринку технологій.

Таблиця 1.1

Переваги та недоліки різних сенсорних технологій

| | Переваги | Недоліки |
|---|---|---|
| Маркери | Низьке обчислювальне навантаження | Маркери на тілі користувача |

Продовження табл. 1.1

| | | |
|---|---|---|
| Камера | Простота установки | Низький рівень надійності |
| Стереокамера | Надійність | Складність розрахунків, складнощі з калібровкою |
| ToF камера | Висока частота кадрів | Дозвіл залежить від потужності світла і відображення |
| Microsoft Kinect | Підтримка програмного забезпечення для розпізнавання жестів тіла | Неможливо використовувати для розпізнавання жестів руки на відстані більше 2 метрів |
| Рукавичка | Швидкість відгуку, точність трекінгу | Громіздкий пристрій з проводами |
| Браслет (сенсори на базі браслета, носима електроніка) | Швидкість відгуку, область дії | Браслет повинен контактувати з тілом людини |
| Безконтактні пристрої | Не потребує контакту з тілом | Низький дозвіл, технологія недостатньо зріла |

# РОЗДІЛ 2
# МЕТОДИ АНАЛІЗУ ДАНИХ ОТРИМАНИХ З ЗОБРАЖЕНЬ

## 2.1 Візуальні характеристики

Людські руки і тіло мають унікальні візуальними особливостями. У розпізнаванні жестів на основі зображень жести складаються з фрагментів людських рук і / або тіла. Тому використання таких візуальних ознак в ідентифікації жестів цілком обґрунтовано.

### 2.1.1 Колір

Колір - це проста візуальна функція для ідентифікації жестів з фонової інформації. Однак на системи розпізнавання жестів на основі кольору сильно впливають освітлення і тіні в складному середовищі HRC. Ще одна поширена проблема у виявленні кольору шкіри полягає в тому, що колір шкіри людини сильно різниться серед людських рас. Через перерахованих вище проблем, в сучасних підходах, колір шкіри розглядається тільки як один з багатьох параметрів при ідентифікації жестів.

### 2.1.2 Локальні признаки

У розпізнаванні жестів на основі зображення умови освітлення сильно впливають на якість ідентифікації жестів. Тому багато дослідників використовують метод локальних ознак, який не чутливий до умов освітлення. Локальний підхід до об'єктів - це деталізований підхід на основі текстур. Він розкладає зображення на більш дрібні області, які не відповідають частинам тіла. Як показано на Рис. 3, однією з найбільш важливих локальних функцій є перетворення ознак інваріантних об'єктів (SIFT). Метод SIFT є обертальним, трансляційним, масштабується і частково освітленим інваріантом. Існує кілька подібних методів локальних ознак, наприклад, SURF і ORBare, запропоновані в більш пізні роки. Як правило, підходи до локальних особливостей також розглядаються тільки як один з безлічі параметрів при ідентифікації жестів.

Кілька методів ідентифікації, таких як методи форми та контуру, методи руху і методи навчання, засновані на локальних ознаках.

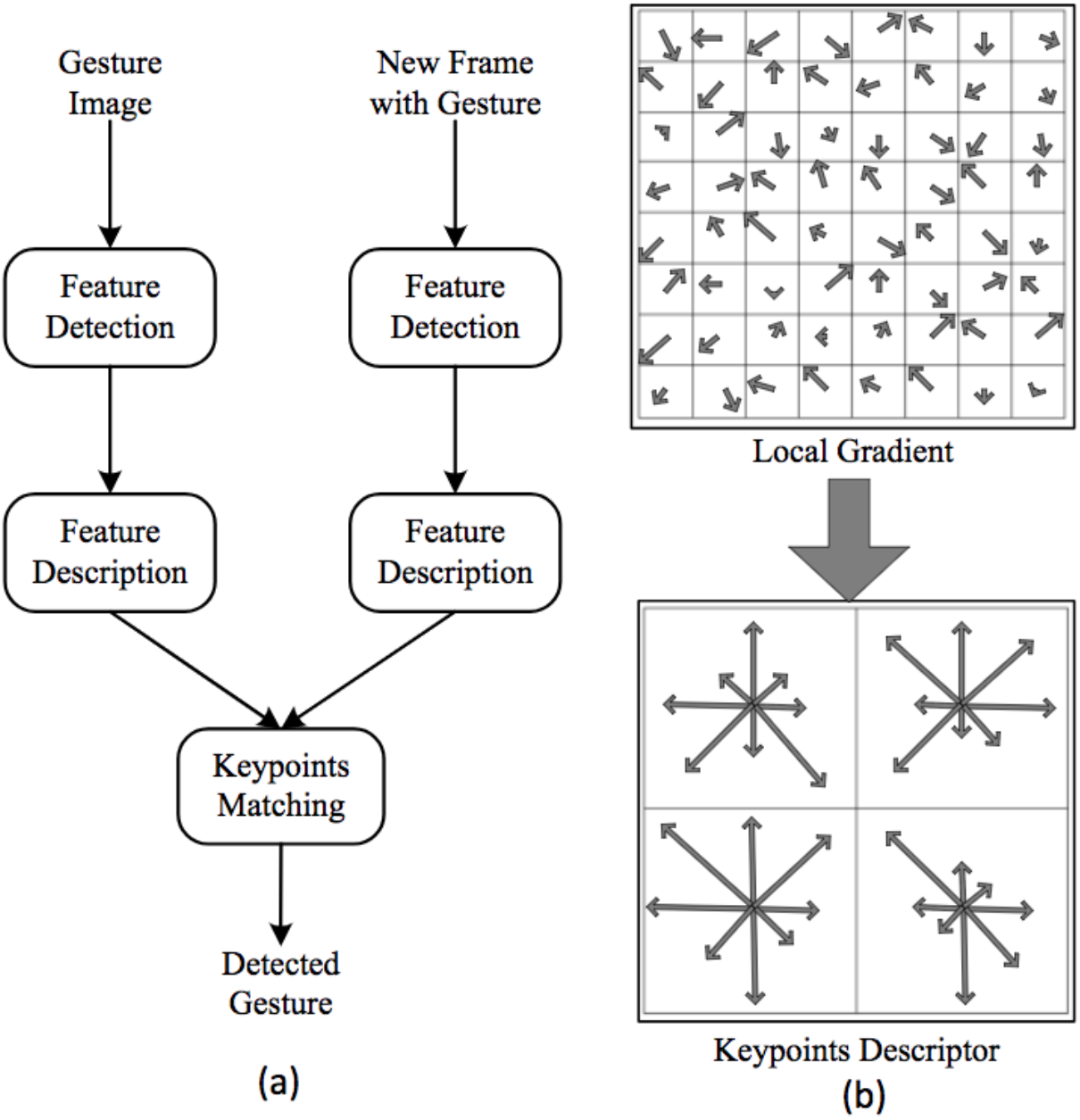



Рис. 2.1 SIFT: (a) алгоритм SIFT для ідентифікації жестів; (b) Приклад дескриптора функції SIFT.

### 2.1.3 Форма і контур

Інший інтуїтивний і простий спосіб ідентифікації жестів - використання унікальної форму і контуру людського тіла в середовищі HRC. Істотний внесок в визначення форми і відповідності був внесений Belongie (співавторство). Вони ввели метод дескриптора контексту форми. Дескриптор контексту форми використовується для виявлення схожих фігур на різних зображеннях. Розробка датчика глибини дає можливість точно вимірювати форму поверхні. 3D-моделі, створені на основі технологій, дозволяють дуже детально представляти форму людського тіла.

### 2.1.4 Рух

У конкретному середовищі HRC людина є єдиним об'єктом, що рухається в масиві необроблених даних. Тому рух є корисною функцією для виявлення людських жестів. Оптичний потік є ключовою технологією ідентифікації жестів на основі руху. Оптичний потік не потребує віднімані фону, що є перевагою в порівнянні з підходами на основі форми та контуру. Кілька додатків розпізнавання жестів реалізовані на основі методу оптичного потоку. Дала і Турау представили знаменитий метод гістограм орієнтованих градієнтів (HOG). Дескриптори HOG ділять кадри зображення на блоки. Для кожного блоку обчислюється гістограма. Серед підходів, заснованих не так на зображенні, розпізнавання жестів на основі руху є популярним методом. Граничні значення і фільтрація зазвичай застосовуються до необроблених даних з датчика для ідентифікації людських жестів.

# РОЗДІЛ 3

# РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 3.1 Аналіз методу розпізнання на основі кольору шкіри

Колір шкіри є важливою інформацією для сегментації руки на зображенні і так само для її відстеження на послідовних відеокадрах. Сегментація на основі кольору шкіри була використана в різних методах для виявлення руки на відеокадрах.

Для даного підходу вирішальну роль часто грає вибір колірного простору. Для досягнення стійкості до зміни освітлення, зазвичай надається перевага використання тих просторів, в яких ефективно розрізняються компоненти хроматичності (chromaticity) і яскравості (luminance) кольорів. З цієї причини, методи виявлення руки за кольором шкіри найчастіше використовують простори HSV, YCrCb, і YUV, щоб було можливо приблизно описувати «хроматичність» шкіри, а не значення її видимого кольору.

У цих методах виключається компонент яскравості, щоб уникнути ефекту затінення, впливу зміни освітлення. Тоді залишившийся двовимірний колірний вектор є постійним і характерним тільки для області шкіри. Гістограма, побудована для області, де присутня шкіра, також містить пікове значення, що збігається з кольором шкіри.

Таким чином, можна виділяти тільки область руки шляхом встановлення деяких порогів у створеній гістограммі і використання методу аналізу сполучених компонентів. Алгоритми на основі кольору шкіри часто стикаються з обмеженнями в тих випадках, коли змінюється розподіл кольору об'єкта при зміні умов освітлення.

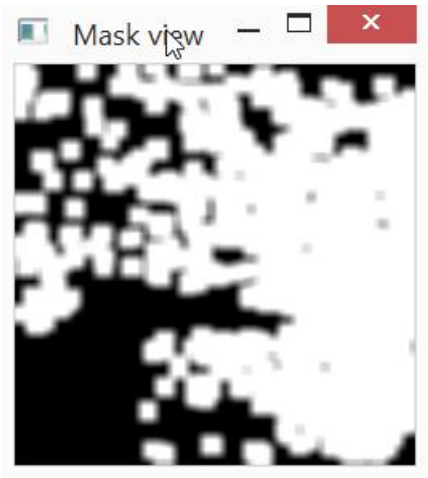



Рис. 3.1 Відображення шуму при засвітленні

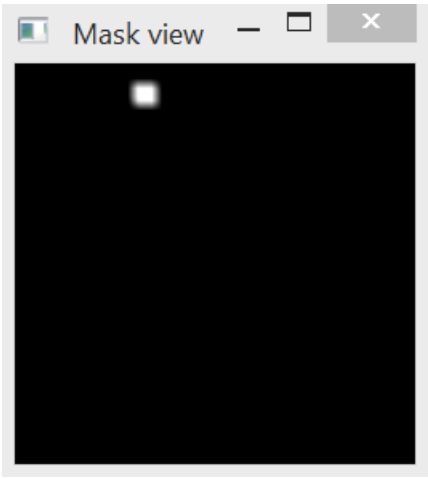



Рис. 3.2 Відображення шуму при максимальному затемненні

Незважаючи на високу швидкість обробки і простоту в реалізації, алгоритми і методи на основі кольору шкіри мають великі недоліки:

- вони не можуть розрізняти об'єкти з схожим кольором (наприклад, обличчя, руки та інші частини тіла);
- їх ефективність сильно залежить від освітлення і різко знижується при його зміні.

Для усунення цих недоліків слід дотримуватися певних умов (наприклад, постійне освітлення, на екрані допускається присутність тільки одного об'єкта, користувач повинен носити сорочку з довгими рукавами, і т.д.), або здійснення додаткової обробки, яка може бути дуже складною.

## 3.2 Використані інструменти

Мовою програмуванняя розроблюваного програмного забезпечення було обрано Python. Дана мова є однією з найбільш популярних вищих мов програмування, зокрема завдяки своєму простому синтаксису, відсутністю необхідності контролювати пам’ять, а також найбільшою перевагою є безліч доступних бібліотек для вирішення різноманітних задач.

Для реалізації поставленого завдання потрібно відтворити потік зображення у реальному часі. Для цього існує бібліотека OpenCV, розширення якої доступне для багатьох мов програмування, зокрема і Python. За допомоги цієї бібліотеки можна виконувати різні функції у роботі із зображеннями. В даній роботі за допомоги OpenCV реалізовано відображення відео в реальному часі з веб-камери користувача, реалізовано управління та конвертація кольорів отриманих з зображень, оформлено вивід результатів.

Для виконання операцій з великими масивами даних існує бібліотека Numeric Python. Дана бібліотека підтримує роботу з великими, багатовимірними масивами та матрицями, а також велику кількість різноманітних математичних функцій для операцій з цими масивами.

Також використана бібліотека Math, яка слугує для підтримки тривіальних математичних операцій, таких як: пошук сінуса, косинуса, тангенса та інших операцій з кутами, розрахунок коренів різних степенів та ін.

## 3.3 Розробка додатку

### 3.3.1 Вхідні дані. Відображення відео. Сегментування кольорів. Маска

Програма просить користувача ввести з переліку номер руху, що слід перевіряти (див. рис. 4). Поточна версія програми підтримує перевірку чотирьох «технік» карате.

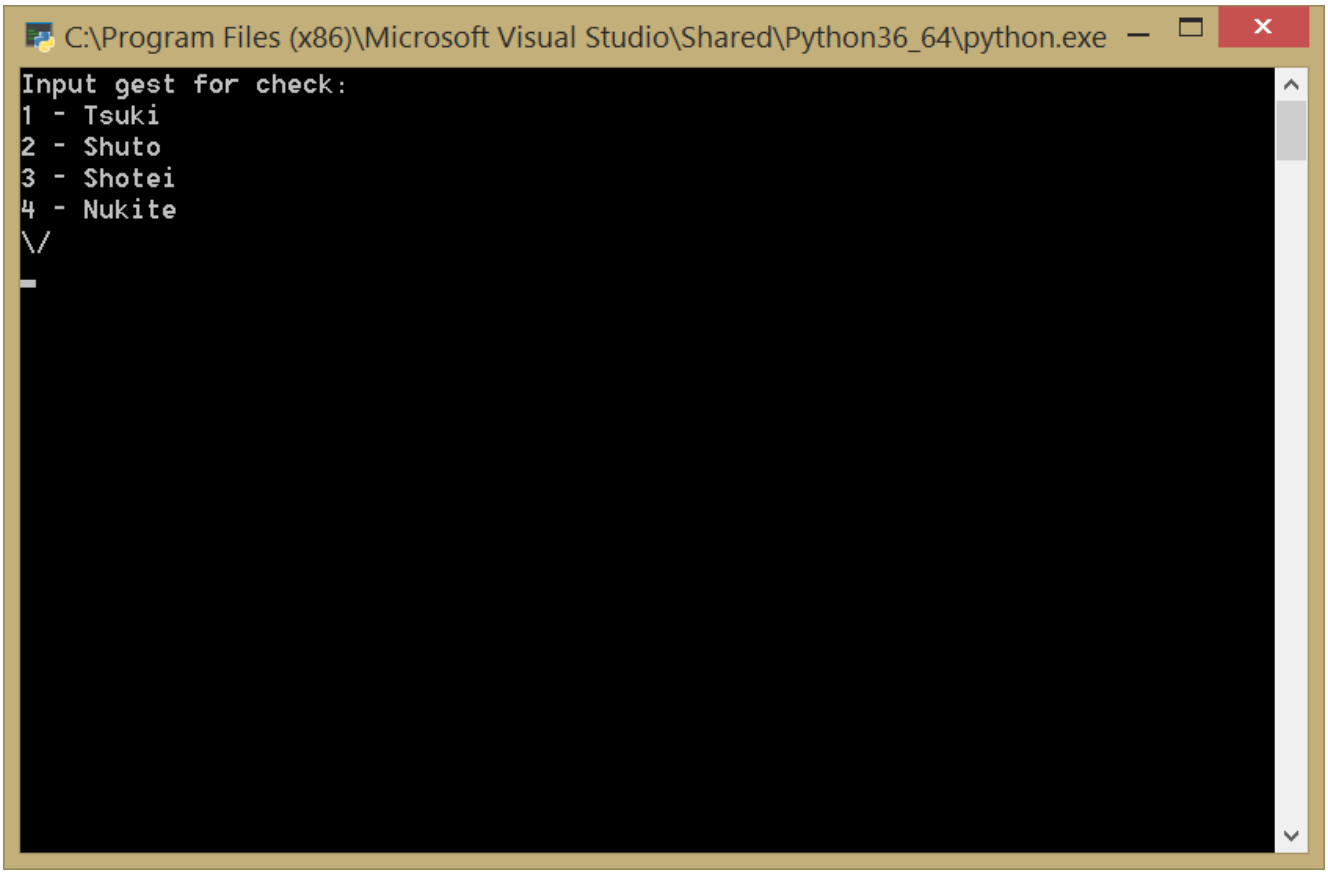



Рис. 3.3 Відображення консолі.

Даний функціонал реалізовано за допомоги консольних можливостей розробки програм мовою Python. Запустивши додаток, користувачу потрібно ввести номер вправи, яку слід перевірити. Так, як на даний момент додаток підтримує оцінку невеликої кількості вправ, реалізація даного функціоналу у консолі є найпростішою та найдоречнішою. Підтримується перевірка таких вправ: ударів руками «Tsuki» (кулак), «Shuto» («ребро» долоні), «Shotei» («п’ята» долоні), «Nukite» (палець). Нижче представлена реалізація вводу даних користувачем:

```
def input_gesture_type():
    print("Input gest for check:\n1 - Tsuki\n2 - Shuto\n3 - Shotei\n4 - Nukite\n\/")
    gesture_num = (int(input()))
    gestures = ['Tsuki', 'Shuto', 'Shotei', 'Nukite']
    gesture_name = gestures[gesture_num - 1]
    return gesture_name
gesture_type = input_gesture_type()
```

Як можна визначити з коду, отримання даних від користувача відбувається за наступною послідовністю: виклик методу для запиту даних, відображення переліку вправ за номерами, запит введення числа від користувача, ініціалізація масиву імен вправ, ідентифікація обраного жесту.

За цим програма виконує вмикання основної камери ПК користувача. Дану функцію виконує метод opencv.VideoCapture(0). Дана функція приймає на вхід ідентифікатор веб-камери, зазвичай основна веб-камера ініціалізована як 0.

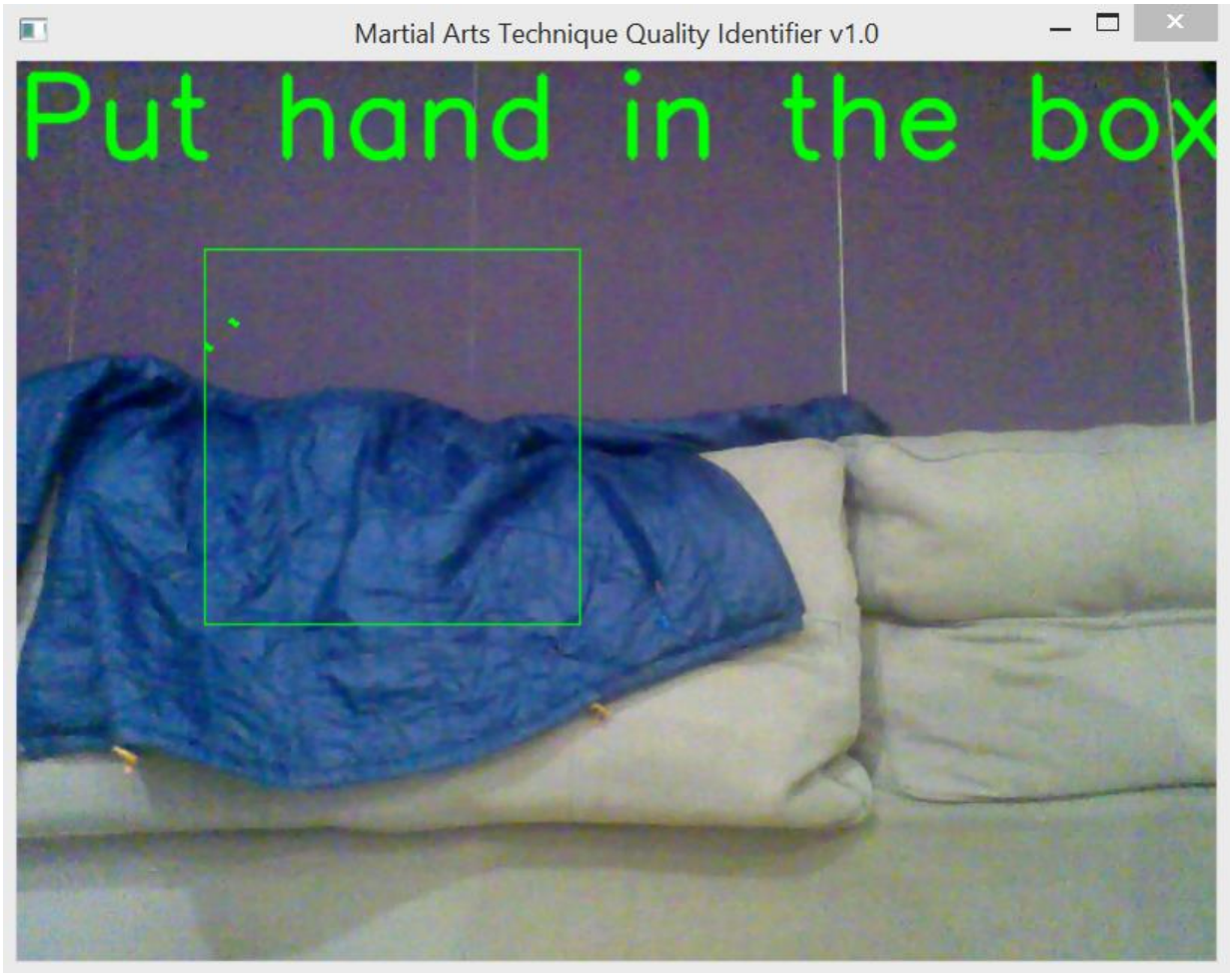



Рис. 3.4 Відображення відео.

Після цього, на екран виводиться вікно, що транслює зображення з відеокамери ПК, а також зону ідентифікації, яка є зеленим квадратом (див. рис. 5), і в даному квадраті відбувається перетворення всіх кольорів BGR у кольори HSV для перетворення у білий та чорний (див. форм. 3.1, 3.2, 3.3, 3.4, 3.5).

При виявленні на зображенні областей, що мають колір шкіри, поряд зі звичайним RGB поданням кольору, тобто интенсивностями червоної, зеленої і синьої складової кольору, використовується подання, засноване на кольоровості - HSV уявлення (Hue - колір або відтінок, Saturation - насиченість, Value - значення).

$$H = \frac{\arctan(y/x)}{2\pi} \quad (3.1)$$

$$S = \sqrt{x^2 + y^2} \quad (3.2)$$

$$V = \frac{R+G+B}{3} \quad (3.3)$$

де

$$x = R - 0.5 * (G + B) \quad (3.4)$$

$$y = \frac{\sqrt{3}}{2} * (G - B) \quad (3.5)$$

Реалізація:

```
retval, window = webcam.read()

window = opencv.flip(window, 1)

core = np.ones((3, 3), np.uint8)

rectangle_for_mask = window[100:300, 100:300]

opencv.rectangle(window, (100, 100), (300, 300), (0, 255, 0), 0)
```

Кожне зображення складається з набору пікселів. Піксель - це будівельний блок зображення. Якщо уявити зображення у вигляді сітки, то кожен квадрат в сітці містить один піксель, де точці з координатою (0, 0) відповідає верхній лівий кут зображення. Наприклад, уявімо, що у нас є зображення з роздільною здатністю 400x300 пікселів. Це означає, що наша сітка складається з 400 рядків і 300 стовпців. У сукупності в нашому зображенні є 400 * 300 = 120000 пікселів.

У більшості зображень пікселі представлені двома способами: у відтінках сірого і в колірному просторі RGB. У зображеннях у відтінках сірого кожен піксель має значення між 0 і 255, де 0 відповідає чорному, а 255 відповідає білому. А значення між 0 і 255

приймають різні відтінки сірого, де значення ближче до 0 темніші, а значення ближче до 255 світліші (Рис. 6).

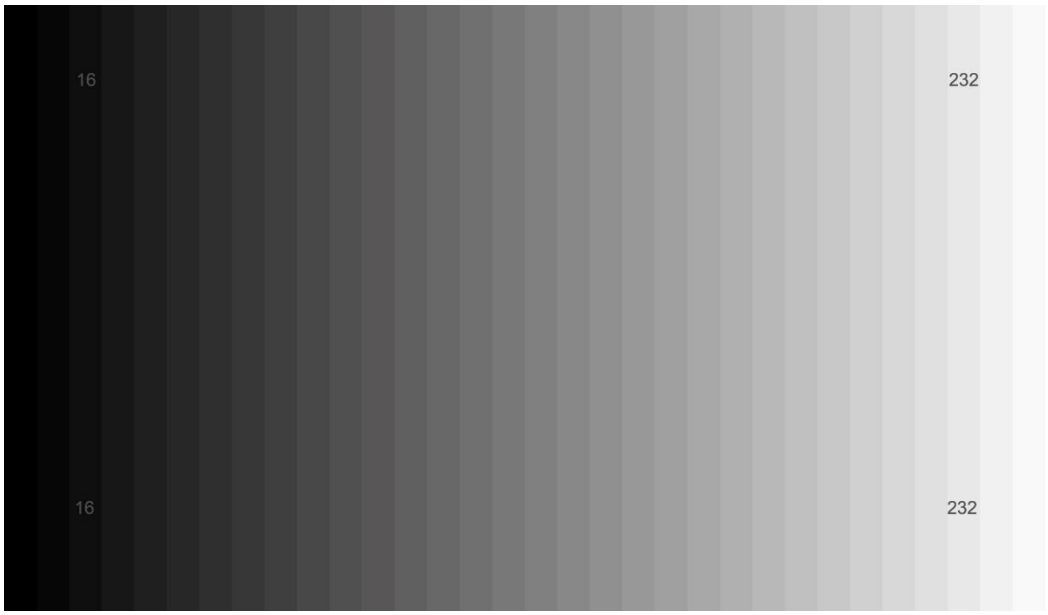



Рис. 3.5 Пікселі відтінків сірого.

Кольорові пікселі представлені в колірному просторі RGB (red, green, blue - червоний, зелений, синій), де одне значення для червоної компоненти, одне для зеленої і одне для синьої. Кожна з трьох компонент представлена цілим числом в діапазоні від 0 до 255 включно, яке вказує як «багато» кольору міститься. Виходячи з того, що кожна компонента представлена в діапазоні [0,255], то для того, щоб представити насиченість кожного кольору, нам буде достатньо 8-бітного цілого без знакового числа. Потім ми об'єднуємо значення всіх трьох компонент в кортеж виду (червоний, зелений, синій). Наприклад, щоб отримати білий колір, кожна з компонент має дорівнювати 255: (255, 255, 255). Тоді, щоб отримати чорний колір, кожна з компонент має дорівнювати 0: (0, 0, 0).

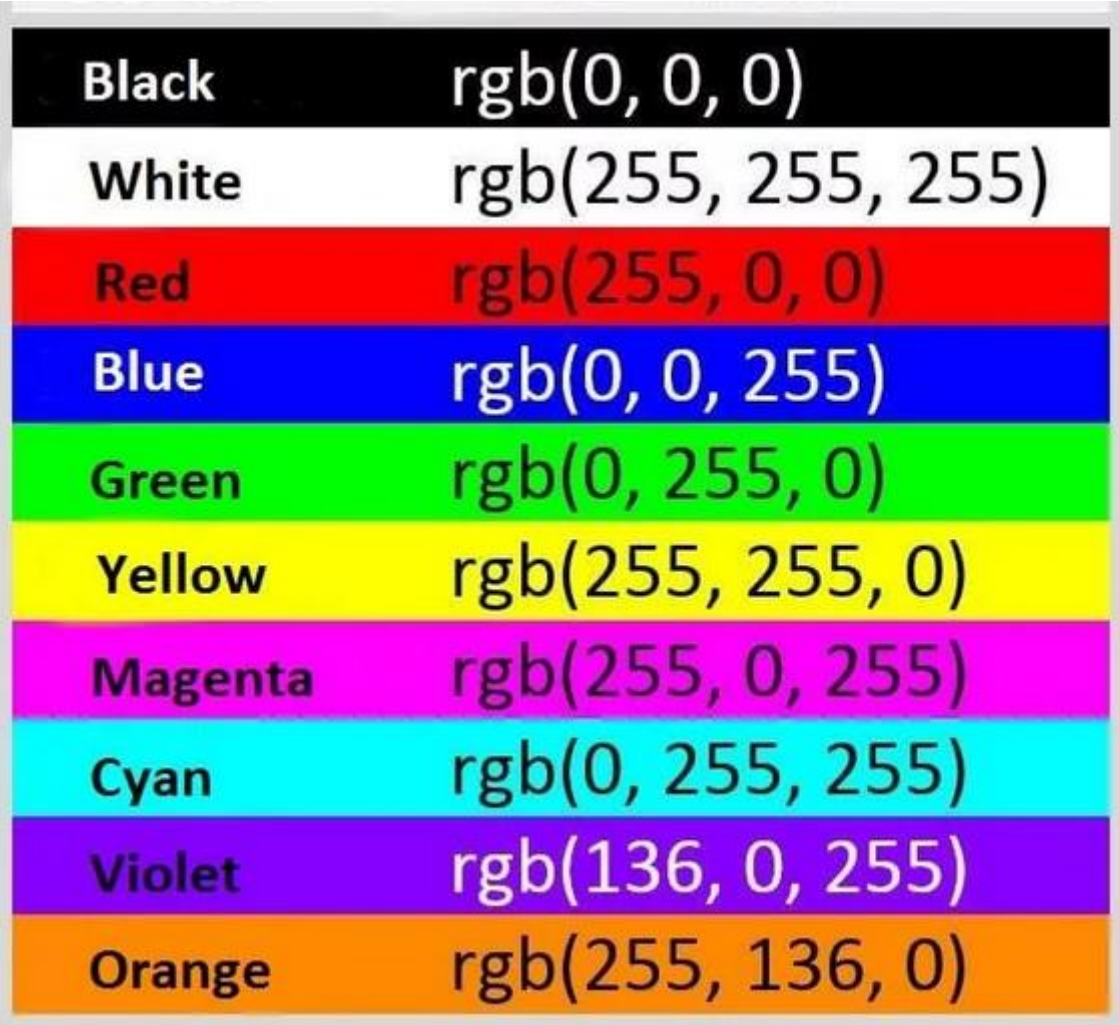



Рис. 3.6 Найпоширеніші кольори у вигляді RGB кортежів.

Після визначаються сегменти кольорів шкіри у HSV. І для маски було визначено, що елемент, який має колір шкіри, буде вважатися 1 або білим, а якщо має інший колір - вважатиметься 0 або чорним. Тож маска відображає руку білим кольором (див. рис. 8). Також, для зменшення шумів в масці, було трохи розширено і розмите зображення.

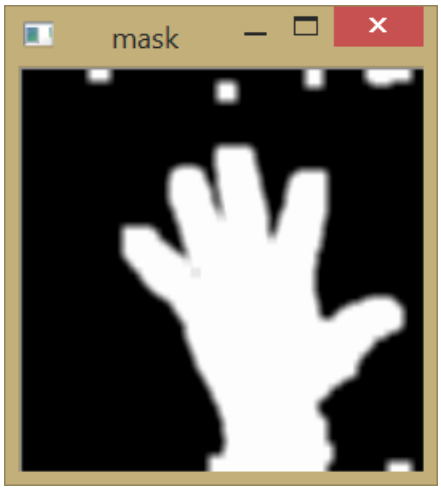



Рис. 3.7 Відображення маски.

Для того щоб виконати перетворення формату кольору з RGB у HSV використано метод cv2.cvtColor() яка приймає два аргументи: зображення та ідентифікатор що визначає з якого у який формат потрібно конвертувати. Реалізація у коді виглядає так:

converted_hsv_color = opencv.cvtColor(rectangle_for_mask,
opencv.COLOR_BGR2HSV)

Наступним кроком є визначення кольорового діапазону. Для цього використано метод inRange(зображення, колір 1, колір 2), де:

- зображення — зображення, на яке накладається фільтр;
- колір 1 — початковий колір діапазона;
- колір 2 — кінцевий колір діапазона.

Реалізація у коді виглядає так:

down_color = np.array([0, 20, 70], dtype=np.uint8)

up_color = np.array([20, 255, 255], dtype=np.uint8)

mask = opencv.inRange(converted_hsv_color, down_color, up_color)

Для того щоб зменшити кількість шумів, як вже зазначалося, було трохи розмите зображення. Для цього використано метод cv2.GaussianBlur(). Це найбільш часто

використовуваний метод розмиття. Розмиття за Гаусом - це, по суті, середнє розмиття, але розмиття за Гаусом засноване на середньозваженому значенні: чим ближче відстань до точки, тим більше вага, і чим далі точка, тим менша вага. З точки зору непрофесіонала, гауссова фільтрація - це процес зваженого усереднення всього зображення, значення кожного пікселя отримується шляхом виваженого усереднення самого себе і інших значень пікселів в околиці.

Рівняння розподілу Гаусса в N вимірах має вигляд (див. форм. 3.6):

$$G(r) = \frac{1}{(2\pi\delta^2)^{N/2}} e^{-r^2/(2\delta^2)} \quad (3.6)$$

Шумозаглушення за допомогою прямокутного фільтра має істотний недолік: пікселі на відстані r від оброблюваного здійснюють на результат той же ефект, що і сусідні.

Ефективніше шумозаглушення можна, таким чином, здійснити, якщо вплив пікселів один на одного буде зменшуватися з відстанню (див. форм. 3.7) (окремий випадок - для двох вимірів):

$$G(u, v) = \frac{1}{2\pi\delta^2} e^{-(u^2+v^2)/(2\delta^2)} \quad (3.7)$$

де u - відстань від центру по горизонтальній осі, v - відстань від центру по вертикальній осі r - це радіус розмиття, δ - стандартне відхилення розподілу Гаусса.

У разі двох вимірів - ця формула задає поверхню, що має вигляд концентричних кіл з розподілом Гаусса від центральної точки. Пікселі, де розподіл відмінний від нуля, використовуються для побудови матриці згортки, яка застосовується до вихідного зображення. Значення кожного пікселя стає середньо зваженим для околиці. Початкове значення пікселя приймає найбільшу вагу (має найвищу Гаусове значення), і сусідні пікселі приймають менші ваги, в залежності від відстані до них. У теорії, розподіл в кожній точці зображення буде ненульовим, що вимагало б обчислення вагових коефіцієнтів для кожного пікселя зображення.

Для застосування даного фільтру використовується згортка по функції (див. форм. 3.8, 3.9):

$$I'(i,j) = \sum_{l=-n}^{n} \sum_{k=-m}^{m} I(i-l)(j-k) * \frac{1}{\sqrt{2\pi}\delta} e^{-\frac{d^2}{2\delta^2}} \quad (3.8)$$

$$d = \sqrt{l^2 + k^2} \quad (3.9)$$

Реалізація в коді:

mask = opencv.GaussianBlur(mask, (5, 5), 100)


Рис. 3.8 Відображення маски без розмиття.

На рис. 9 зображено відображення маски під час виконання вправи «Shotei», але функція розмиття не активована. Як бачимо посеред долоні є проміжок, що потім стане причиною визначення виконання вправи неправильним. Тож, дана функція є однією з найважливіших у реалізації

### 3.3.2 Контур руки

Після попередніх дій, знаходиться контур рук на зображенні. Під контуром мається на увазі лінії обмеження будь-якої області, що відображується у зоні ідентифікації. Максимально велика площа області очевидно буде рукою. Область руки виділяється опуклими лініями. Ця область є контролем руки. Після цього контур руки було

апроксимовано до максимальної площі, щоб зменшити невелику кількість шуму, присутнього там. Реалізація в коді:

```
contours, hierarchy = opencv.findContours(mask, opencv.RETR_TREE,
opencv.CHAIN_APPROX_SIMPLE)
counter = max(contours, key=lambda x: opencv.contourArea(x))
epsilon = 0.0005 * opencv.arcLength(counter, True)
approximation = opencv.approxPolyDP(counter, epsilon, True)
```

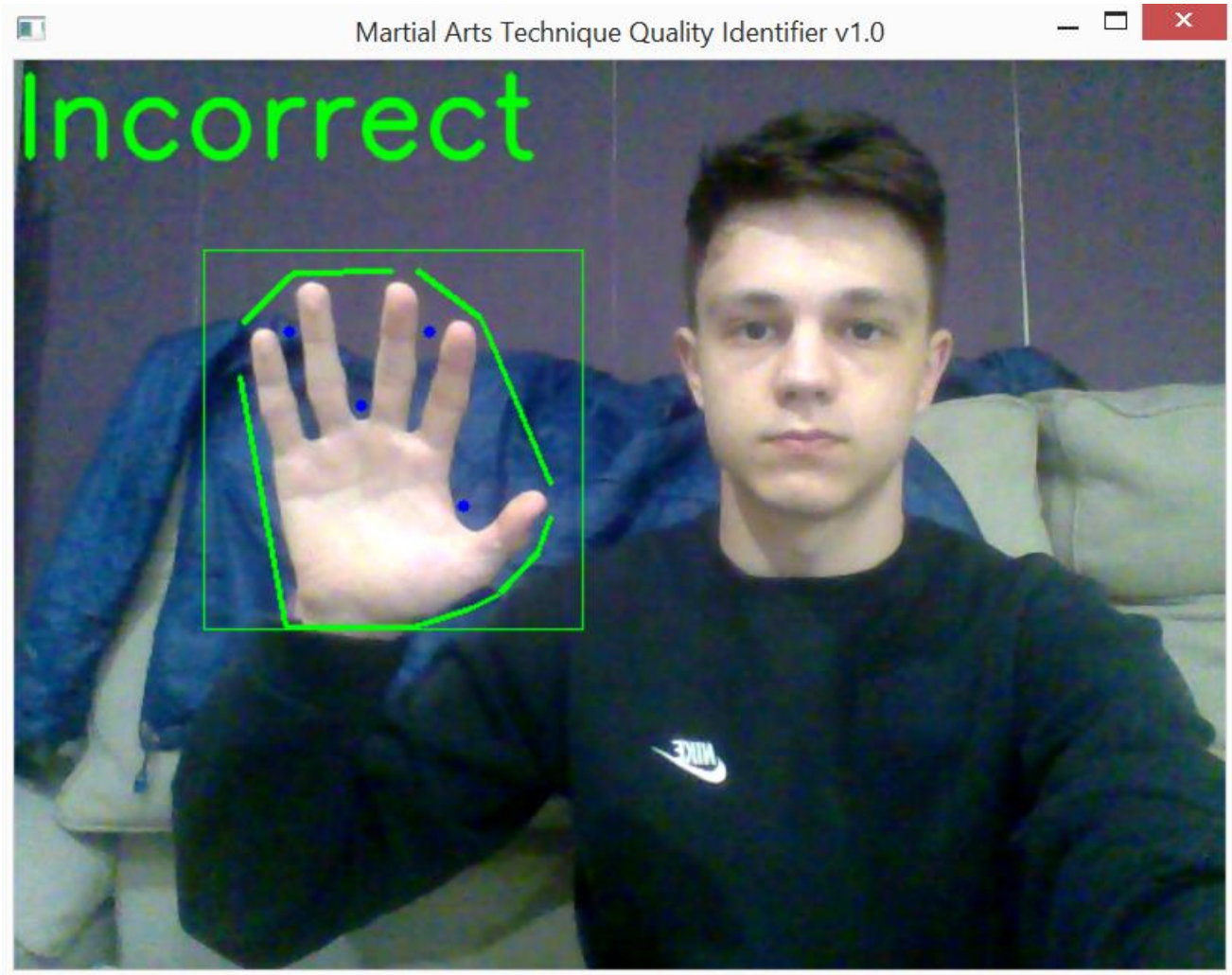



Рис. 3.9 Відображення контуру руки

### 3.3.3 Дефекти

Після цього виконується пошук дефектів в області контролю для ідентифікації пальців. Дефектами виявились площі, які не охоплені рукою у зоні контролю. Такі, як зони між пальцями.

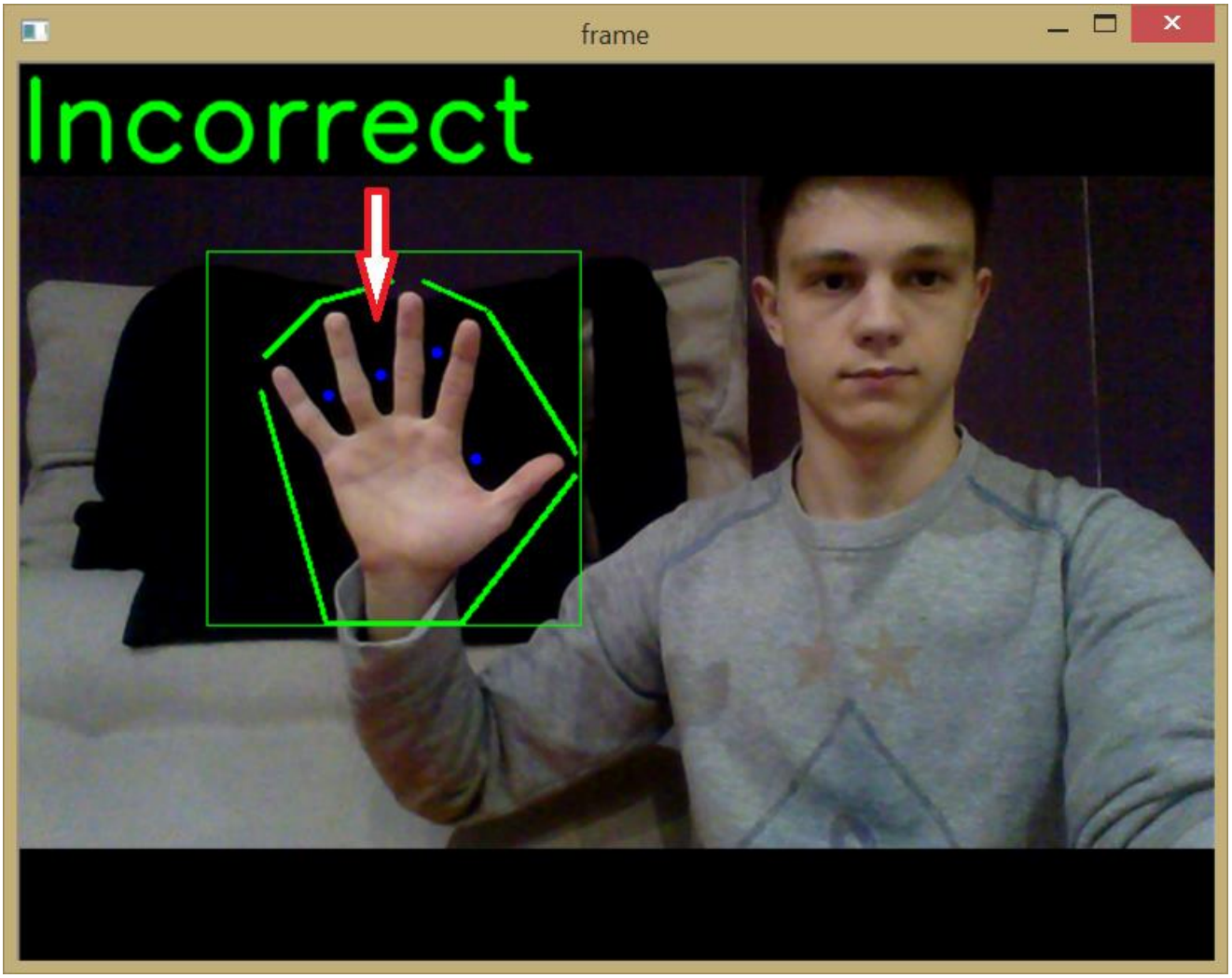



Рис. 3.10 Відображення дефектів.

Але для виділення пальців потрібно відсортувати ці дефекти. Це виконано за допомогою кутів (див. форм. 3.10). Всі кути між пальцями менше 90 градусів. За допомогою правила косинуса знаходиться кут цих дефектів, і якщо кут менше 90 градусів, збільшується лічильник дефектів на 1 і відображається синє коло у місці знаходження дефекта.

$$\alpha = \text{acos}\left(\frac{b^2+c^2-a^2}{2bc}\right) * 57 \quad (3.10)$$

де a, b, c – це сторони трикутника.

Також іноді зайві дефекти можуть з’явитися під крайніми пальцями. Щоб уникнути цього, було визначено мінімально можливу відстань від контуру до точки (див. форм. 3.11, 3.12, 3.13). Якщо користувач наближає пальці то точка зникає. Це є мінімальною дистанцією.

$$s = \frac{a+b+c}{2} \quad (3.11)$$

$$ar = \sqrt{(s * (s-a) * (s-b) * (s-c))} \quad (3.12)$$

$$d = \frac{2*ar}{a} \quad (3.13)$$

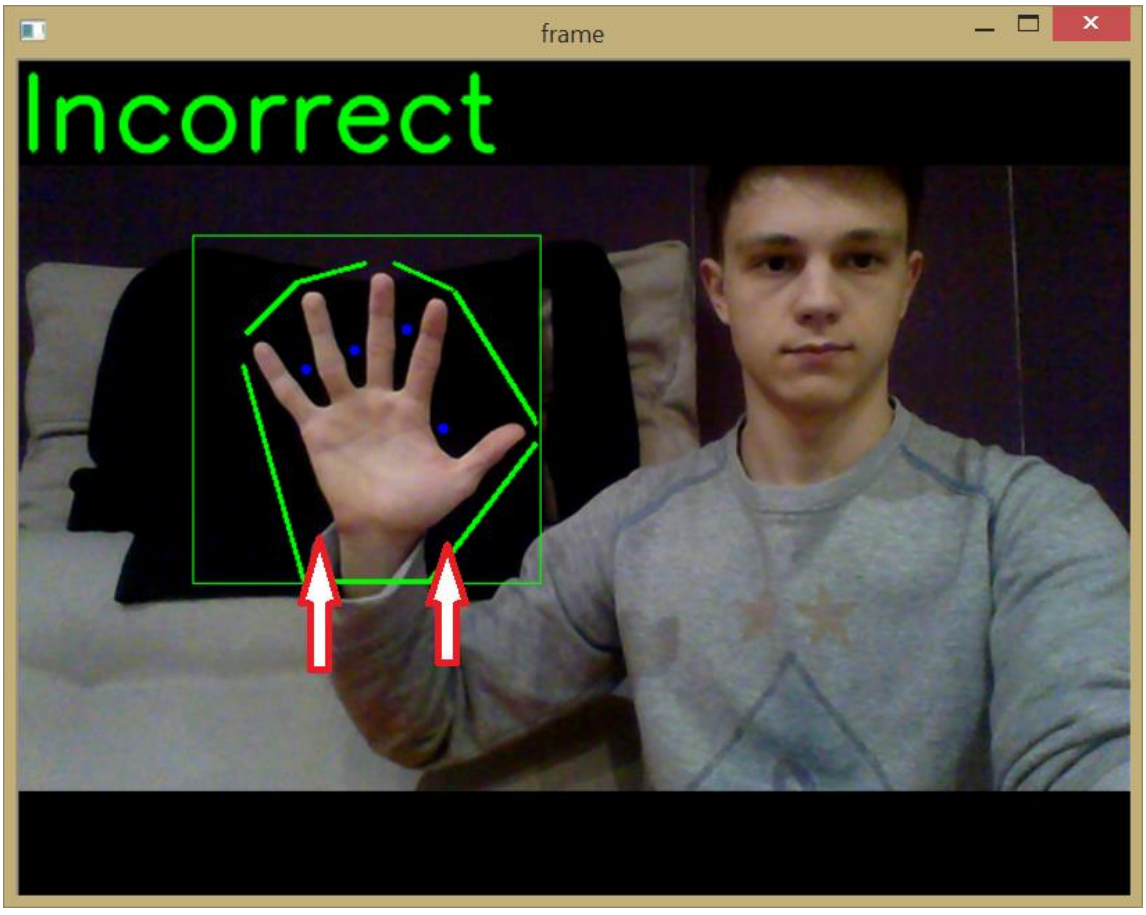



Рис. 3.11 Відображення зайвих дефектів.

Реалізація в коді:

```
source_parameter, end_parameter, far_parameter, distance = defects[i, 0]
source = tuple(approximation[source_parameter][0])
end = tuple(approximation[end_parameter][0])
far = tuple(approximation[far_parameter][0])
pt = (100, 180)

side_one = math.sqrt((end[0] - source[0]) ** 2 + (end[1] - source[1]) ** 2)
side_two = math.sqrt((far[0] - source[0]) ** 2 + (far[1] - source[1]) ** 2)
side_three = math.sqrt((end[0] - far[0]) ** 2 + (end[1] - far[1]) ** 2)
source_parameter = (side_one + side_two + side_three) / 2
triangle_area = math.sqrt(source_parameter * (source_parameter - side_one) *
(source_parameter - side_two) * (source_parameter - side_three))

distance = (2 * triangle_area) / side_one

corner = math.acos((side_two ** 2 + side_three ** 2 - side_one ** 2) / (2 * side_two *
side_three)) * 57

if corner <= 90 and distance > 20:
        defect_number += 1
        opencv.circle(rectangle_for_mask, far, 3, [255, 0, 0], -1)
```

### 3.3.4 Ідентифікація правильності виконання

Отримавши інформацію про охоплювану рукою площу та кількість дефектів, можна визначити якість виконання вправи. Для ударів «Tsuki» (кулак), «Shuto» («ребро» долоні), «Shotei» («п’ята» долоні) кількість дефектів повинна дорівнювати 0, так як основа цих ударів це тримати кисть згуртовано, що дає їй силу, а відокремлення хоча б одного пальця призведе, в кращому випадку, до втрати сили, а в гіршому – до травми.

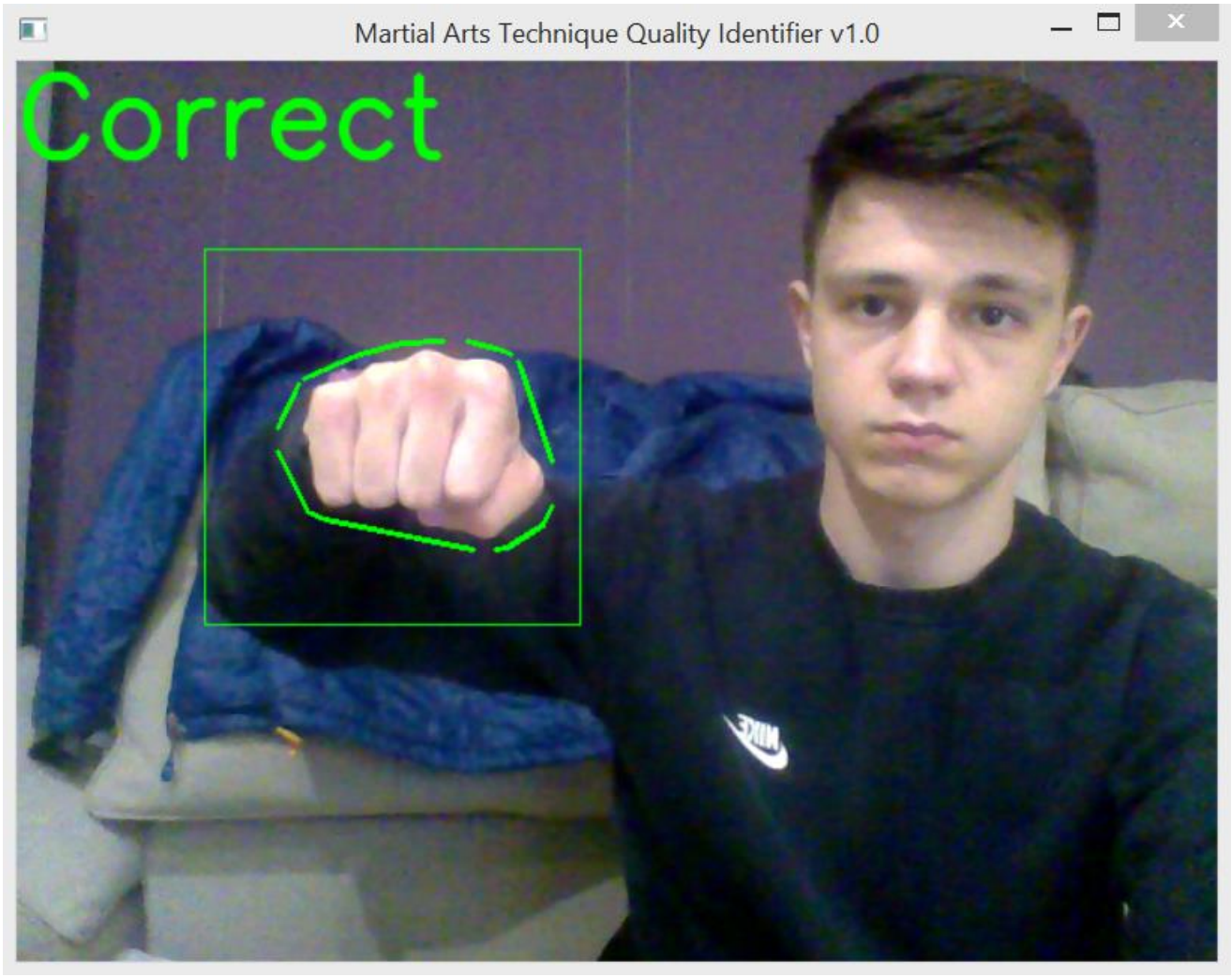



Рис. 3.12 Варіант правильного виконання «Tsuki»

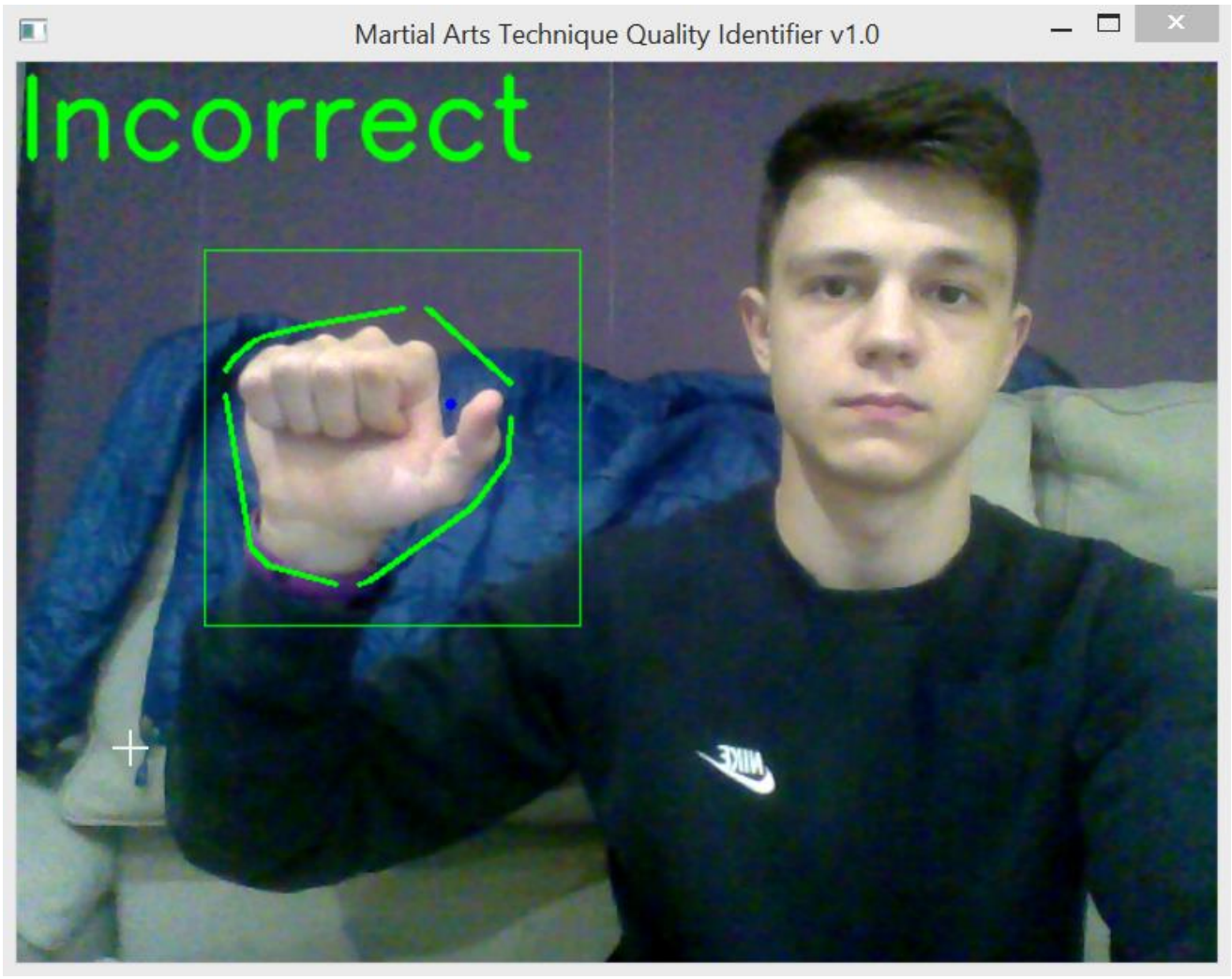



Рис. 3.13 Варіант неправильного виконання «Tsuki»

Для удару «Nukite» (палець) кількість дефектів повинна дорівнювати 1, так як даний удар виконується в очі, пальці повинні бути згуртовані з розмежуванням між «безіменним» та «середнім» пальцями. Але, водночас, «великий» палець обов’язково повинен бути притиснутий уздовж «вказівного» згідно його техніки виконання в карате.

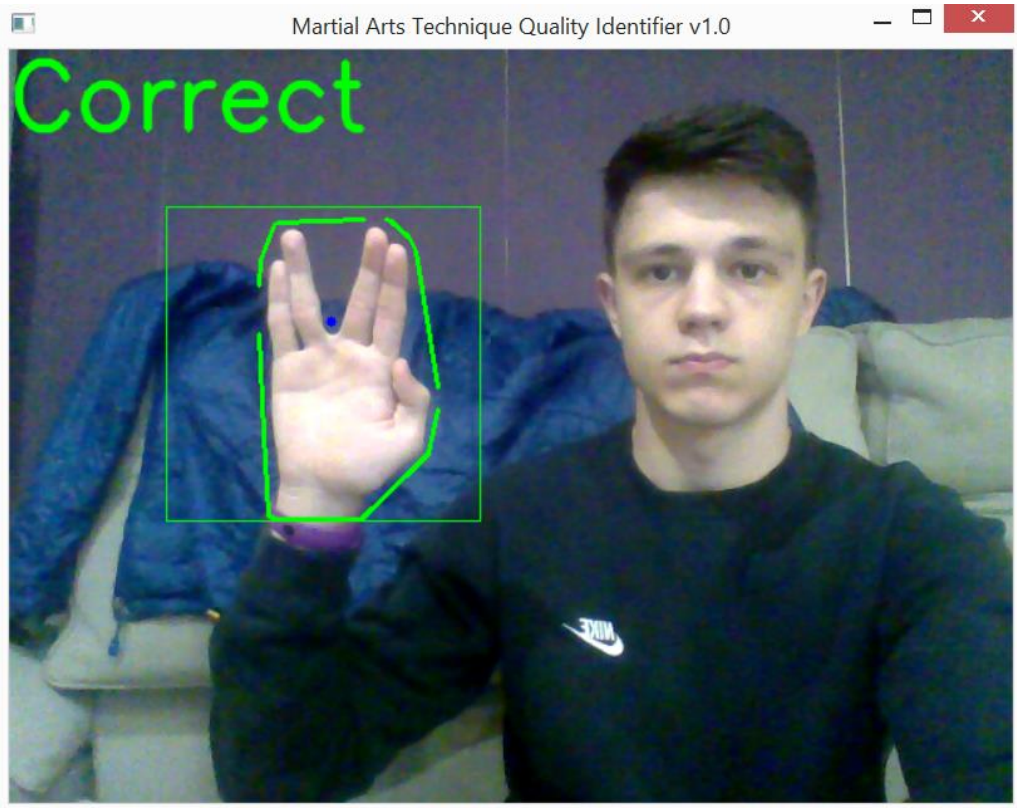



Рис. 3.14 Варіант правильного виконання «Nukite»

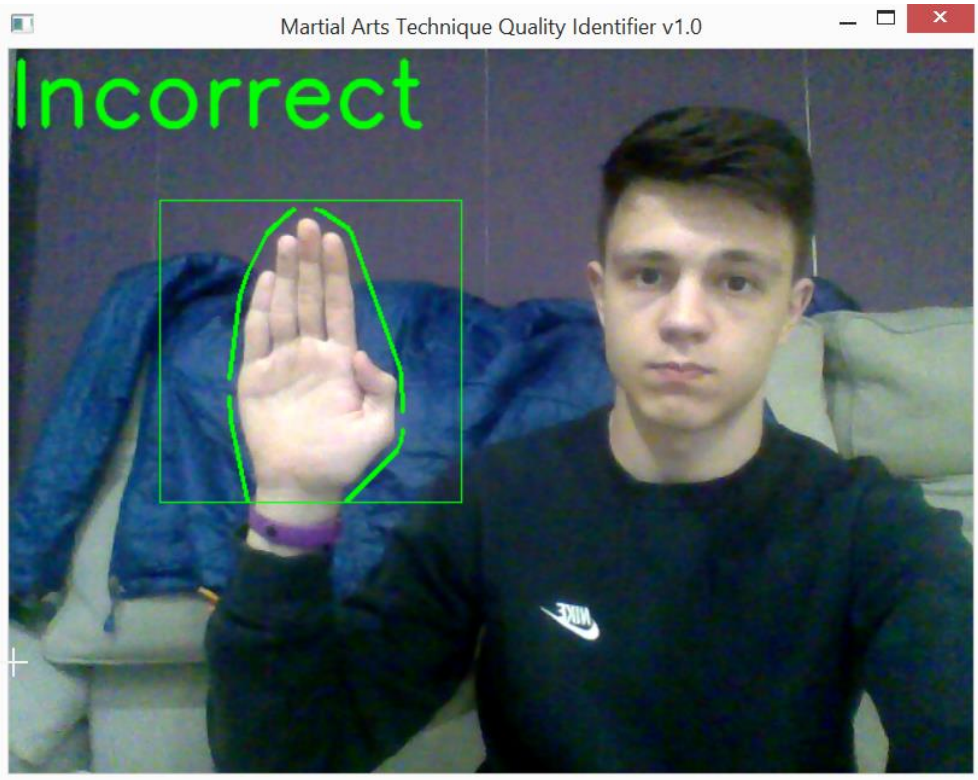



Рис. 3.15 Варіант неправильного виконання «Nukite»

Реалізація в коді:

```
if defect_number == 0 and (gesture_type == 'Tsuki' or gesture_type == 'Shotei' or gesture_type == 'Shuto'):
        if area_contour < 2000:
opencv.putText(window, 'Put hand in the box', (0, 50), font, 2, (0, 255, 0), 3, opencv.LINE_AA)
        else:
opencv.putText(window, 'Correct', (0, 50), font, 2, (0, 255, 0), 3, opencv.LINE_AA)
elif defect_number > 0 and (gesture_type == 'Tsuki' or gesture_type == 'Shotei' or gesture_type == 'Shuto'):
        if area_contour < 2000:
opencv.putText(window, 'Put hand in the box', (0, 50), font, 2, (0, 255, 0), 3, opencv.LINE_AA)
        else:
opencv.putText(window, 'Incorrect', (0, 50), font, 2, (0, 255, 0), 3, opencv.LINE_AA)
elif defect_number == 1 and gesture_type == 'Nukite':
        opencv.putText(window, 'Correct', (0, 50), font, 2, (0, 255, 0), 3, opencv.LINE_AA)
elif defect_number != 1 and gesture_type == 'Nukite':
        opencv.putText(window, 'Incorrect', (0, 50), font, 2, (0, 255, 0), 3, opencv.LINE_AA)
```

# ВИСНОВКИ

В даній роботі було досліджено можливості отримання зображень в режимі реального часу. Розроблено алгоритм ідентифікації якості виконання вправ щодо східних єдиноборств. Запропоновано використання методу косинуса для ідентифікації проміжків у долоні або «кулаці» та використання підрахунку проміжків, як параметр ідентифікації якості виконання вправи. Виконана програмна реалізація алгоритму.

Пропонована реалізація алгоритму застосовується для ідентифікації якості виконання чотирьох вправ руками щодо східних єдиноборств. Програмний код побудовано так, що програма може легко розширюватись та підтримувати оцінювання набагато більшої кількості вправ.

Розроблене ПЗ, в перспективі є суттєвим кроком до автоматизації оцінювання спортсменів. Даний крок дозволить справедливо оцінювати спортсмена і уникнути будь-яких сумнівів щодо отриманої степені майстерності та правильності виконання вправ.

Результати бакалаврського дослідження були представлені на Всеукраїнському конкурсі студентських наукових робіт з інформатики та кібернетики ВНТУ 2020-2021 навчального року. Було особливо відмічено високу оригінальність ідей, самостійність роботи та практичне впровадження результатів.

Було опубліковано тези в збірнику узагальнених матеріалів конференції MSTIoE 2020-7 (7-ма Східно-Європейська конференція «Математичні та програмні технології Internet of Everything»), яка проходила на базі кафедри програмних систем і технологій Київського національного університету імені Тараса Шевченка 22-23.12.2020 року.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. OpenCV - WikiZero. – [ Електронний ресурс]. – Режим доступу: https://www.wikizero.com/uk/OpenCV
2. Python - WikiZero. – [ Електронний ресурс]. – Режим доступу: https://www.wikizero.com/uk/Python
3. OpenCV в Python - Хабр. – [ Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/519454/
4. NumPy в Python - Хабр. – [ Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/352678/
5. Распознавание жестов для взаимодействия с ИИ: от теории к последним достижениям - Integral. – [ Електронний ресурс]. – Режим доступу: http://integral-russia.ru/2020/07/30/raspoznavanie-zhestov-dlya-vzaimodejstviya-s-ii-ot-teorii-k-poslednim-dostizheniyam/
6. Применение возможностей Intel Perceptual Computing SDK для расширения границ взаимодействия людей с ограниченными возможностями с внешним миром - Интуит. – [ Електронний ресурс]. – Режим доступу: https://intuit.ru/studies/courses/10620/1104/lecture/24035
7. Виявлення ознак (комп'ютерний зір) - Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%92%D0%B8%D1%8F%D0%B2%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F_%D0%BE%D0%B7%D0%BD%D0%B0%D0%BA_(%D0%BA%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B8%D0%B9_%D0%B7%D1%96%D1%80)
8. Scott Shell, An Introduction To Numpy and Scipy by Scott Shell. 2019.
9. Обработка изображений Python-OpenCV-07-Gaussian Blur - Русские Блоги. – [Електронний ресурс]. – Режим доступу: https://russianblogs.com/article/8557564668/

10. OpenCV на python: цветовой фильтр - ROBOTCLASS. – [Електронний ресурс]. – Режим доступу: https://robotclass.ru/tutorials/opencv-color-range-filter/

11. Python - Python. – [Електронний ресурс]. – Режим доступу: https://www.python.org/

12. OpenCV - OpenCV. – [Електронний ресурс]. – Режим доступу: https://opencv.org/

13. Нейросеть от Google AI распознает жесты в реальном времени - Neurohive. – [Електронний ресурс]. – Режим доступу: https://neurohive.io/ru/novosti/nejroset-ot-google-ai-raspoznaet-zhesty-v-realnom-vremeni/

14. Распознавание жестов лайк/дизлайк с помощью компьютерного зрения - DOU. – [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/27803/

15. Training a Neural Network to Detect Gestures with OpenCV in Python – towards data science. – [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/training-a-neural-network-to-detect-gestures-with-opencv-in-python-e09b0a12bdf1

Додаток А

## ВИХІДНИЙ КОД ПРОГРАМИ

```
import cv2 as opencv
import numpy as np
import math


class quality_check:
    def __init__(self):
        print("Input gest for check:\n1 - Tsuki\n2 - Shuto\n3 - Shotei\n4 - Nukite\n\/")
        self.gesture_num = (int(input()))
        self.gestures = ['Tsuki', 'Shuto', 'Shotei', 'Nukite']
        self.gesture_name = self.gestures[self.gesture_num - 1]

    @staticmethod
    def evaluation_accuracy(webcam, gesture_name):
        while 1:
            try:
                retval, window = webcam.read()
                window = opencv.flip(window, 1)
                core = np.ones((3, 3), np.uint8)

                rectangle_for_mask = window[100:300, 100:300]
                opencv.rectangle(window, (100, 100), (300, 300), (0, 255, 0), 0)
                converted_hsv_color = opencv.cvtColor(rectangle_for_mask,
opencv.COLOR_BGR2HSV)
```

```
down_color = np.array([0, 20, 70], dtype=np.uint8)
up_color = np.array([20, 255, 255], dtype=np.uint8)

mask = opencv.inRange(converted_hsv_color, down_color, up_color)

mask = opencv.dilate(mask, core, iterations=4)

mask = opencv.GaussianBlur(mask, (5, 5), 100)

contours, hierarchy = opencv.findContours(mask, opencv.RETR_TREE,
opencv.CHAIN_APPROX_SIMPLE)

counter = max(contours, key=lambda x: opencv.contourArea(x))

epsilon = 0.0005 * opencv.arcLength(counter, True)
approximation = opencv.approxPolyDP(counter, epsilon, True)

hull = opencv.convexHull(counter)

area_contour = opencv.contourArea(counter)

hull = opencv.convexHull(approximation, returnPoints=False)
defects = opencv.convexityDefects(approximation, hull)

defect_number = 0

for i in range(defects.shape[0]):
```

```
            source_parameter, end_parameter, far_parameter, distance = defects[i, 0]
            source = tuple(approximation[source_parameter][0])
            end = tuple(approximation[end_parameter][0])
            far = tuple(approximation[far_parameter][0])
            pt = (100, 180)

            side_one = math.sqrt((end[0] - source[0]) ** 2 + (end[1] - source[1]) **
2)
            side_two = math.sqrt((far[0] - source[0]) ** 2 + (far[1] - source[1]) ** 2)
            side_three = math.sqrt((end[0] - far[0]) ** 2 + (end[1] - far[1]) ** 2)
            source_parameter = (side_one + side_two + side_three) / 2
            triangle_area = math.sqrt(
                source_parameter * (source_parameter - side_one) *
(source_parameter - side_two) * (
                        source_parameter - side_three))

            distance = (2 * triangle_area) / side_one

            corner = math.acos(
                (side_two ** 2 + side_three ** 2 - side_one ** 2) / (2 * side_two *
side_three)) * 57

            if corner <= 90 and distance > 20:
                defect_number += 1
                opencv.circle(rectangle_for_mask, far, 3, [255, 0, 0], -1)

            opencv.line(rectangle_for_mask, source, end, [0, 255, 0], 2)
```

```
            font = opencv.FONT_HERSHEY_SIMPLEX
            if defect_number == 0 and (
                gesture_name == 'Tsuki' or gesture_name == 'Shotei' or gesture_name
== 'Shuto'):
                if area_contour < 2000:
                    opencv.putText(window, 'Put hand in the box', (0, 50), font, 2, (0, 255,
0), 3, opencv.LINE_AA)
                else:
                    opencv.putText(window, 'Correct', (0, 50), font, 2, (0, 255, 0), 3,
opencv.LINE_AA)
            elif defect_number > 0 and (
                gesture_name == 'Tsuki' or gesture_name == 'Shotei' or gesture_name
== 'Shuto'):
                if area_contour < 2000:
                    opencv.putText(window, 'Put hand in the box', (0, 50), font, 2, (0, 255,
0), 3, opencv.LINE_AA)
                else:
                    opencv.putText(window, 'Incorrect', (0, 50), font, 2, (0, 255, 0), 3,
opencv.LINE_AA)
            elif defect_number == 1 and gesture_name == 'Nukite':
                opencv.putText(window, 'Correct', (0, 50), font, 2, (0, 255, 0), 3,
opencv.LINE_AA)
            elif defect_number != 1 and gesture_name == 'Nukite':
                opencv.putText(window, 'Incorrect', (0, 50), font, 2, (0, 255, 0), 3,
opencv.LINE_AA)

            # opencv.imshow('Mask view', mask)
            opencv.imshow('Martial Arts Technique Quality Identifier v1.0', window)
```

```
            except:
                print("Put hand in the box")
            key = opencv.waitKey(5) & 0xFF
            if key == 27:
                break

    def execute(self):
        webcam = opencv.VideoCapture(0)
        self.evaluation_accuracy(webcam, self.gesture_name)
        opencv.destroyAllWindows()
        webcam.release()


if __name__ == '__main__':
    start_check = quality_check()
    start_check.execute()
```

Додаток Б

# ДІАГРАМА СТАНІВ

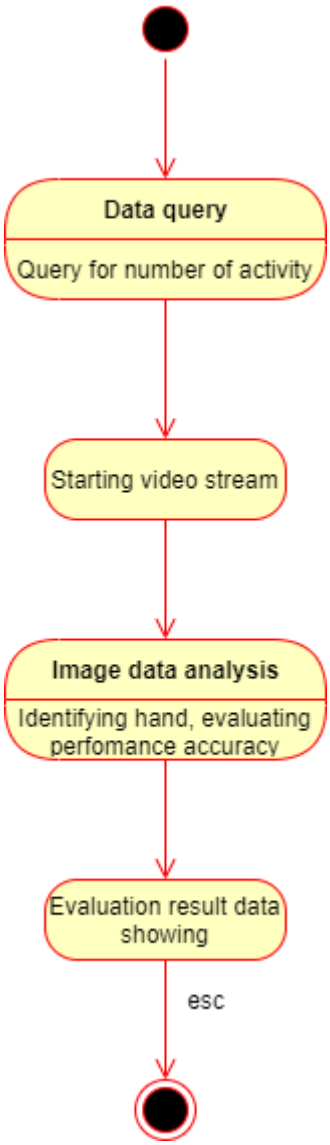



Рис. Б.1 Діаграма станів

Додаток В

# ДІАГРАМА ПРЕЦЕДЕНТІВ

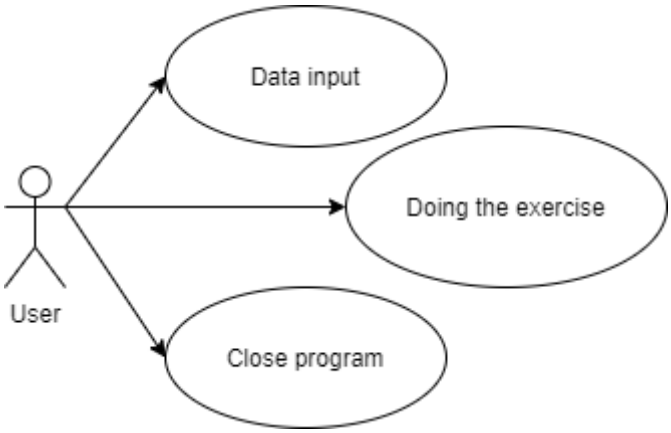



Рис. В.1 Діаграма прецедентів

Додаток Д

# ДІАГРАМА ПОСЛІДОВНОСТЕЙ

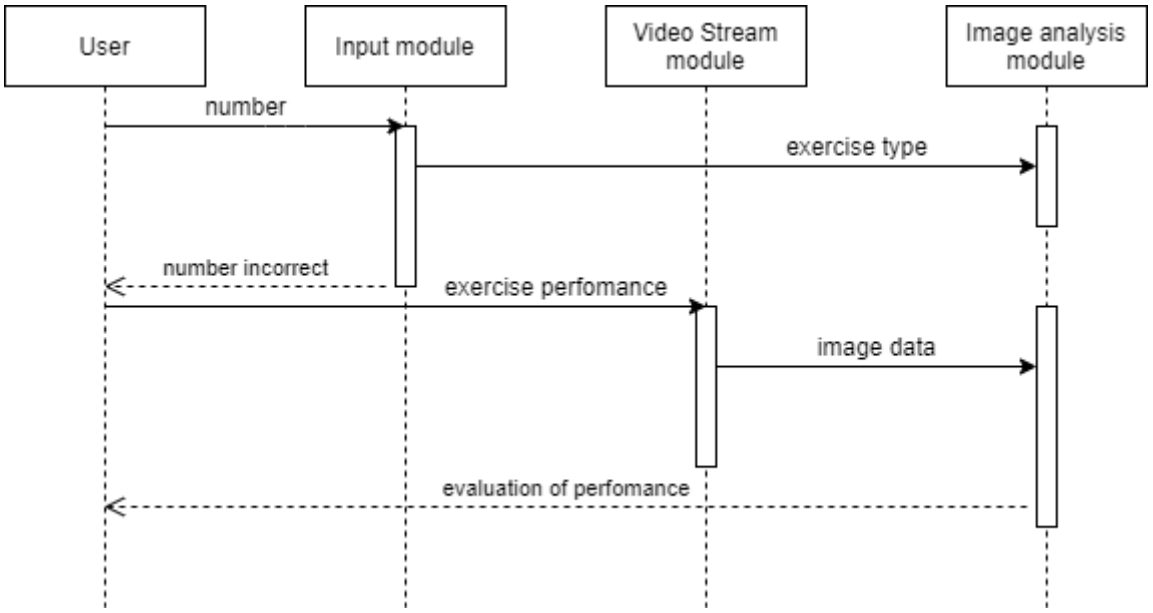



Рис. Д.1 Діаграма послідовностей

Додаток Е

# СКРІНШОТИ ТЕСТУВАННЯ ВИКОНАННЯ ПРОГРАМИ

Е.1

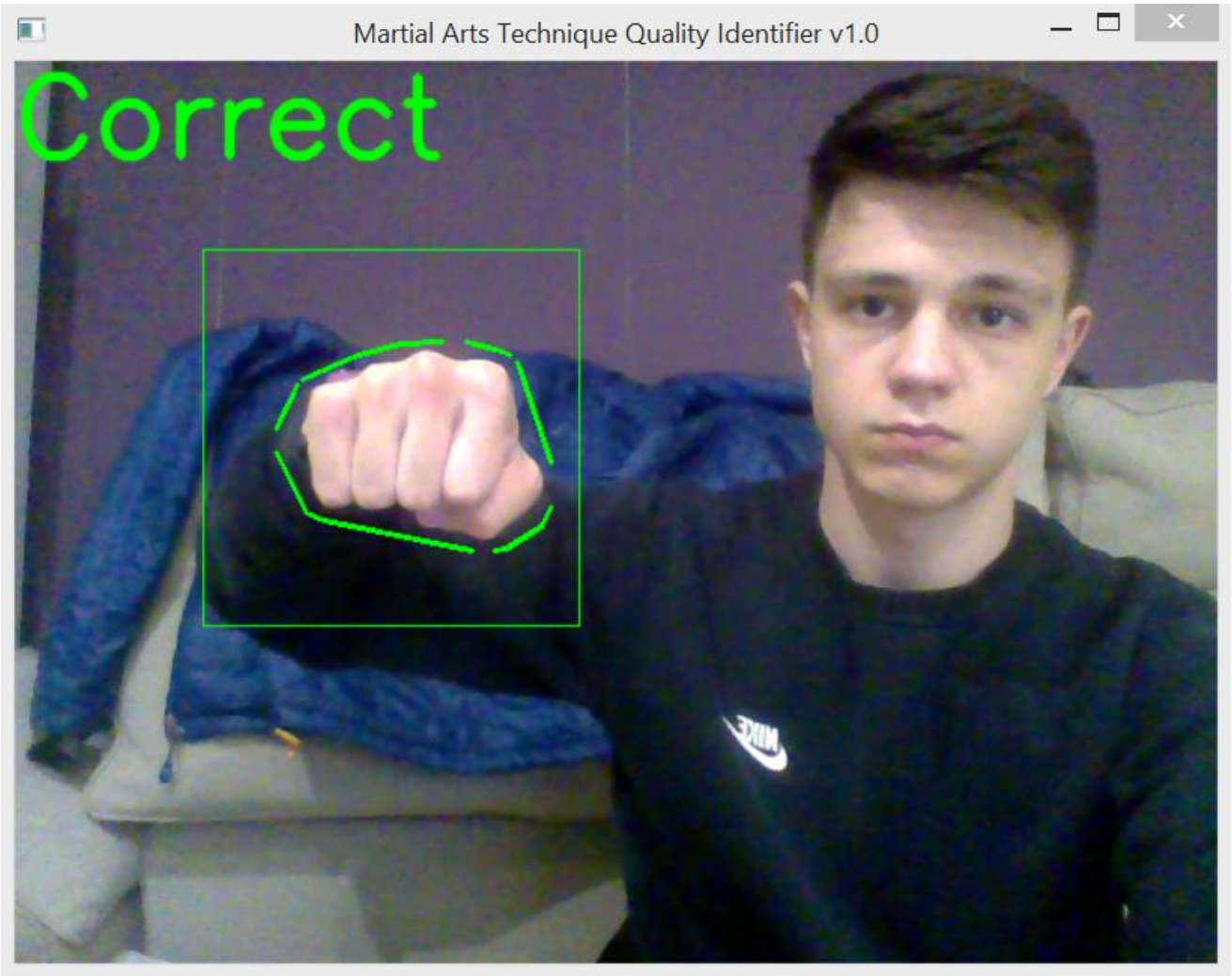



Рис. Е.1.1 Виконання вправи «Tsuki»

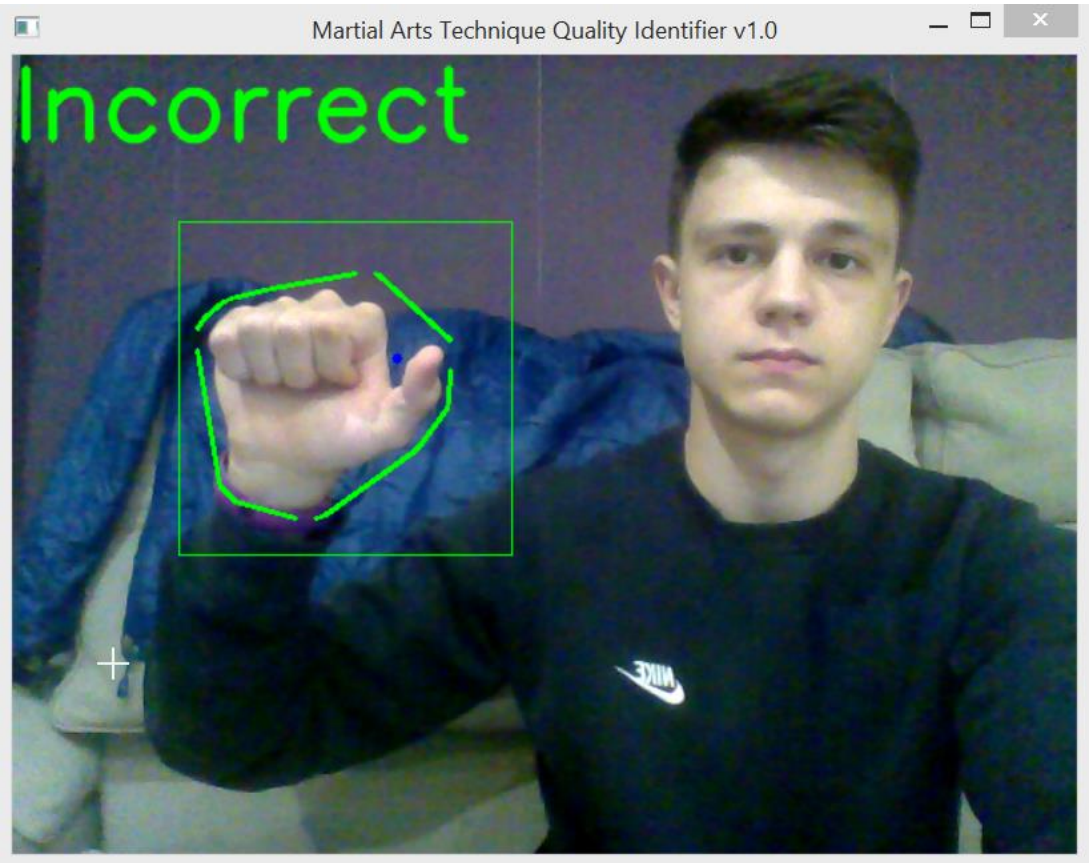



Рис. Е.1.2 Виконання вправи «Tsuki»

Е.2 – Вправа «Shuto»:

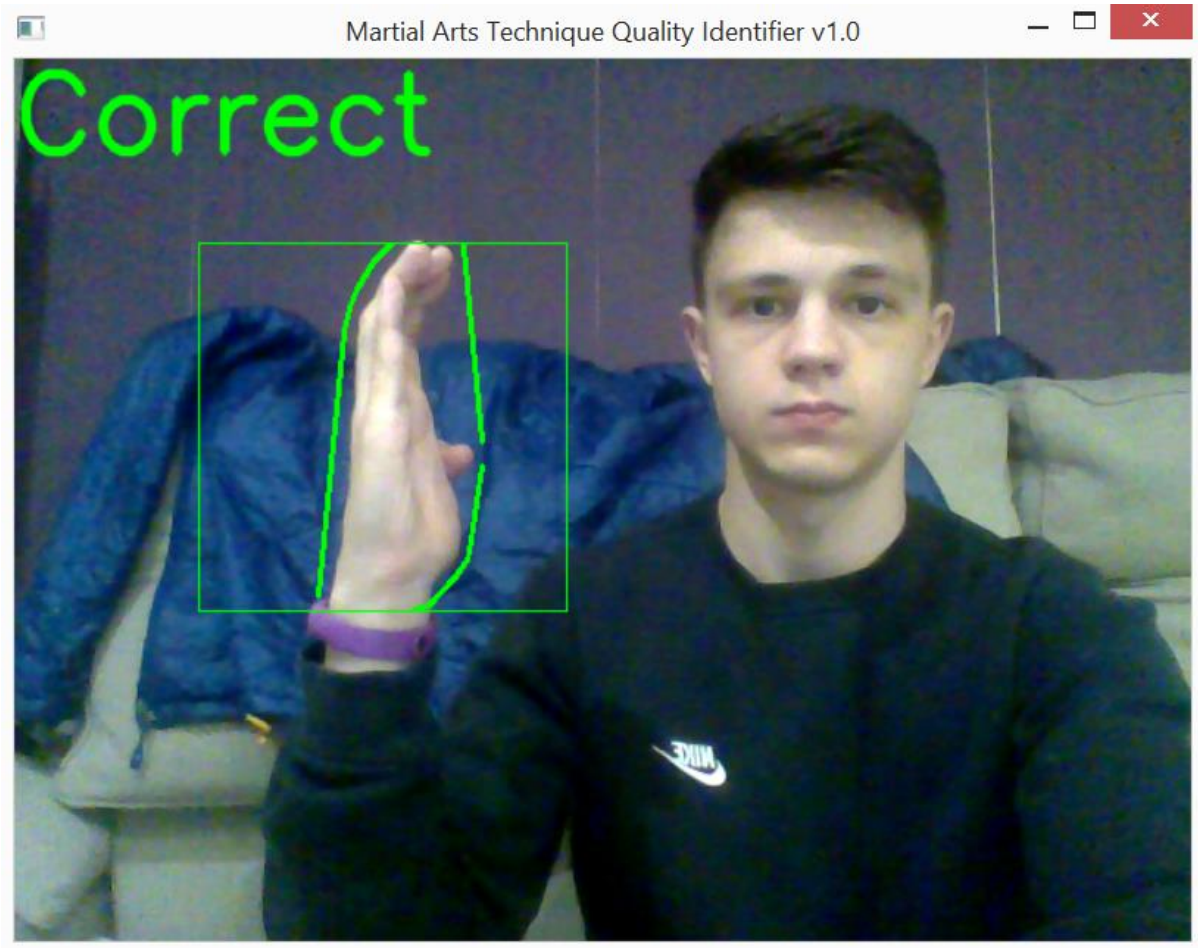



Рис. Е.2.1 Виконання вправи «Shuto»

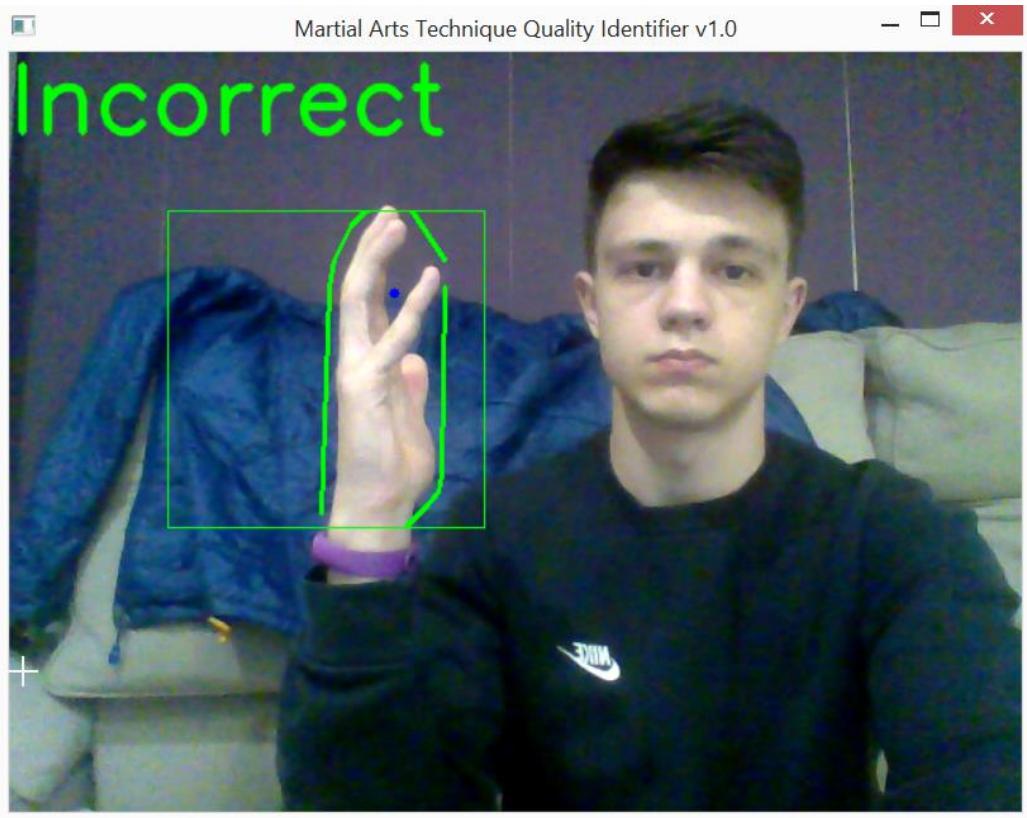



Рис. Е.2.2 Виконання вправи «Shuto»

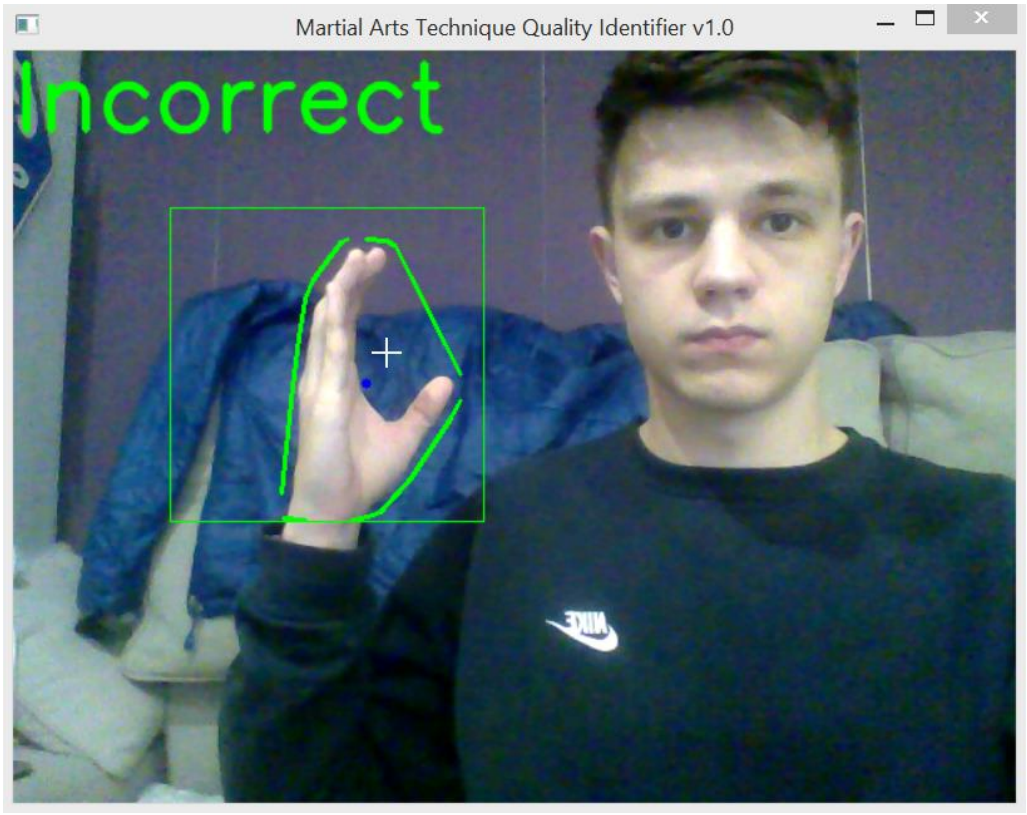



Рис. Е.2.3 Виконання вправи «Shuto»

Е.3 – Вправа «Shotei»:

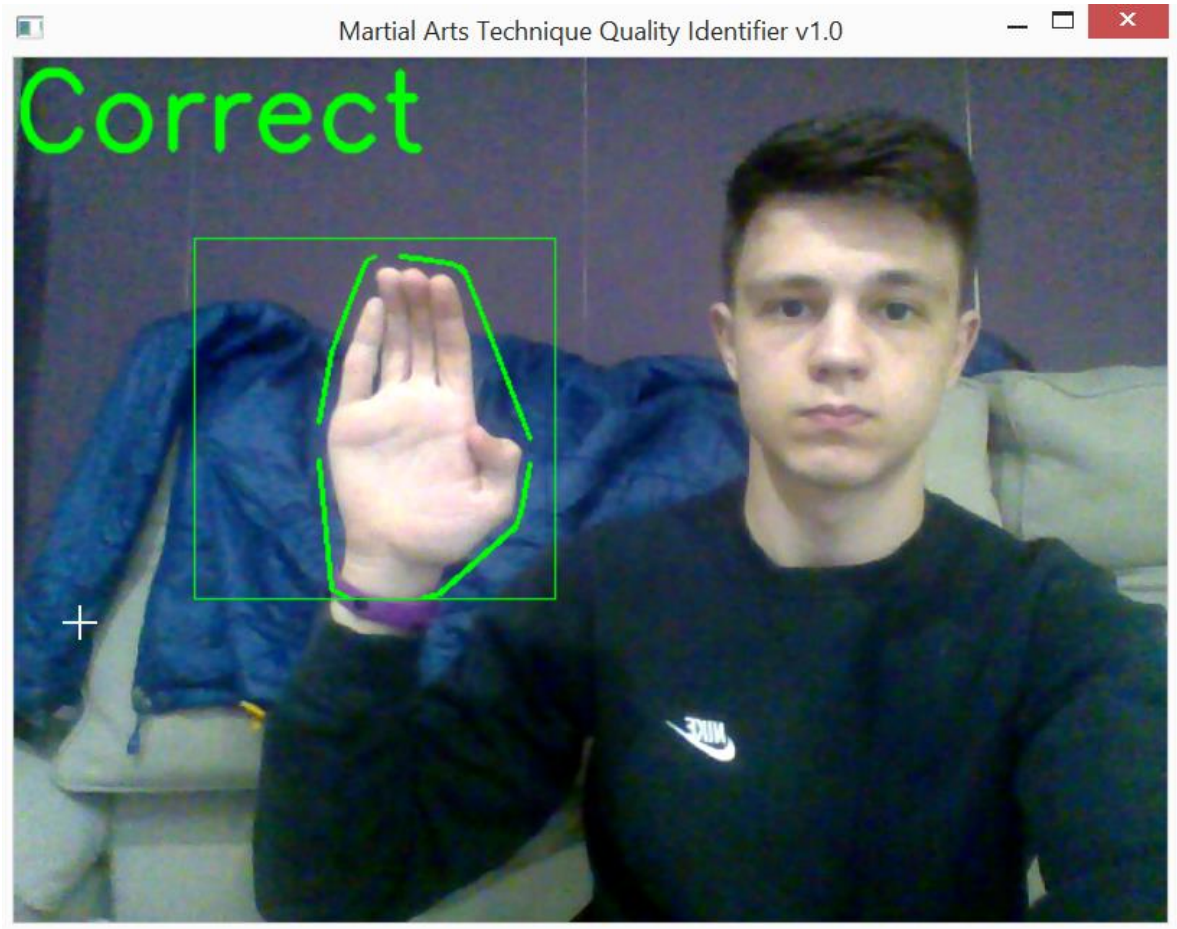



Рис. Е.3.1 Виконання вправи «Shotei»

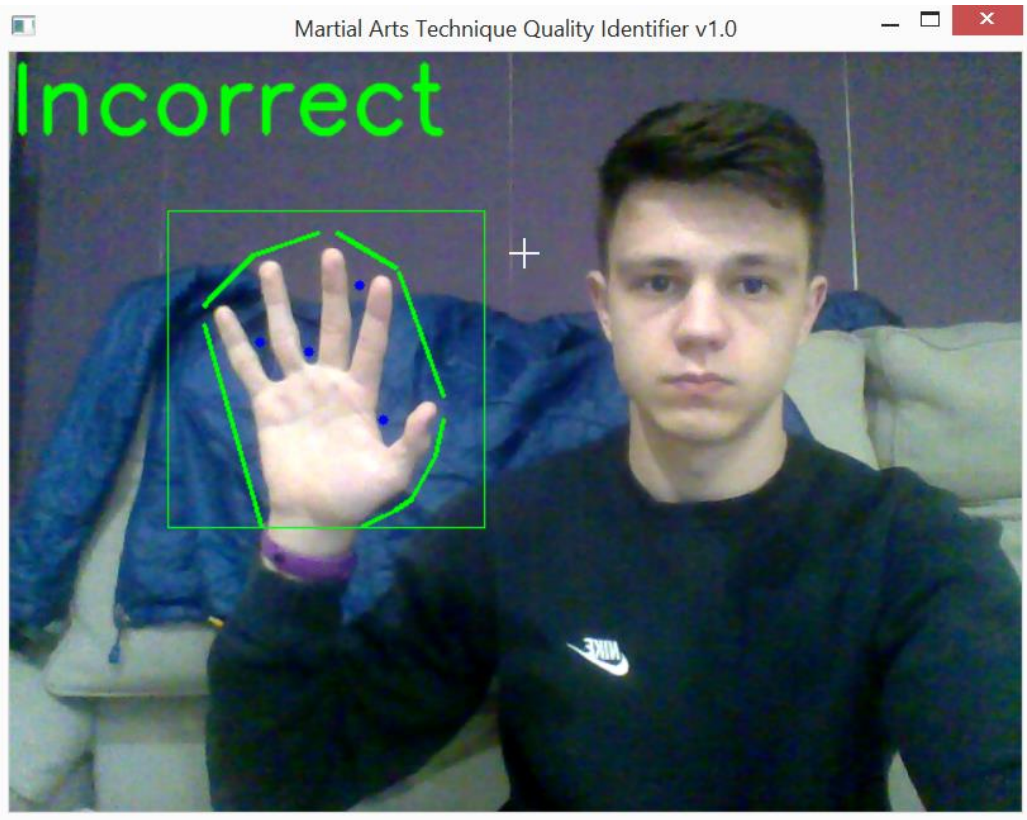



Рис. Е.3.2 Виконання вправи «Shotei»

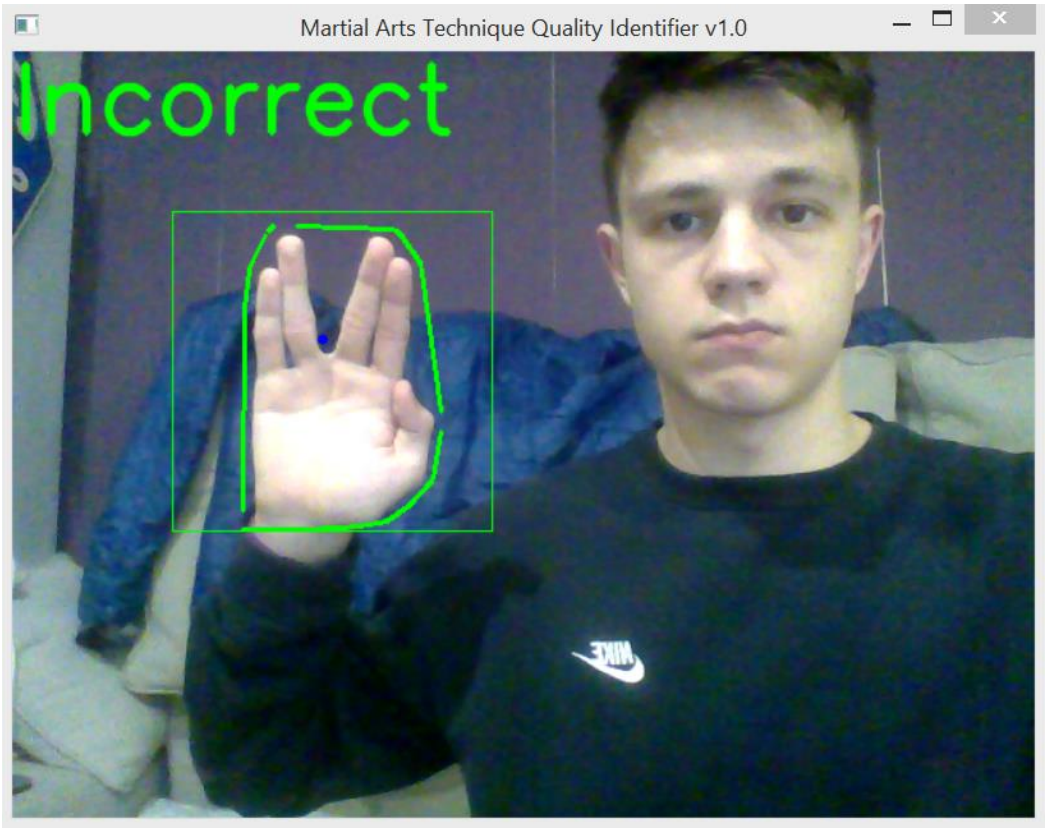



Рис. Е.3.3 Виконання вправи «Shotei»

Е.4 – Вправа «Nukite»:

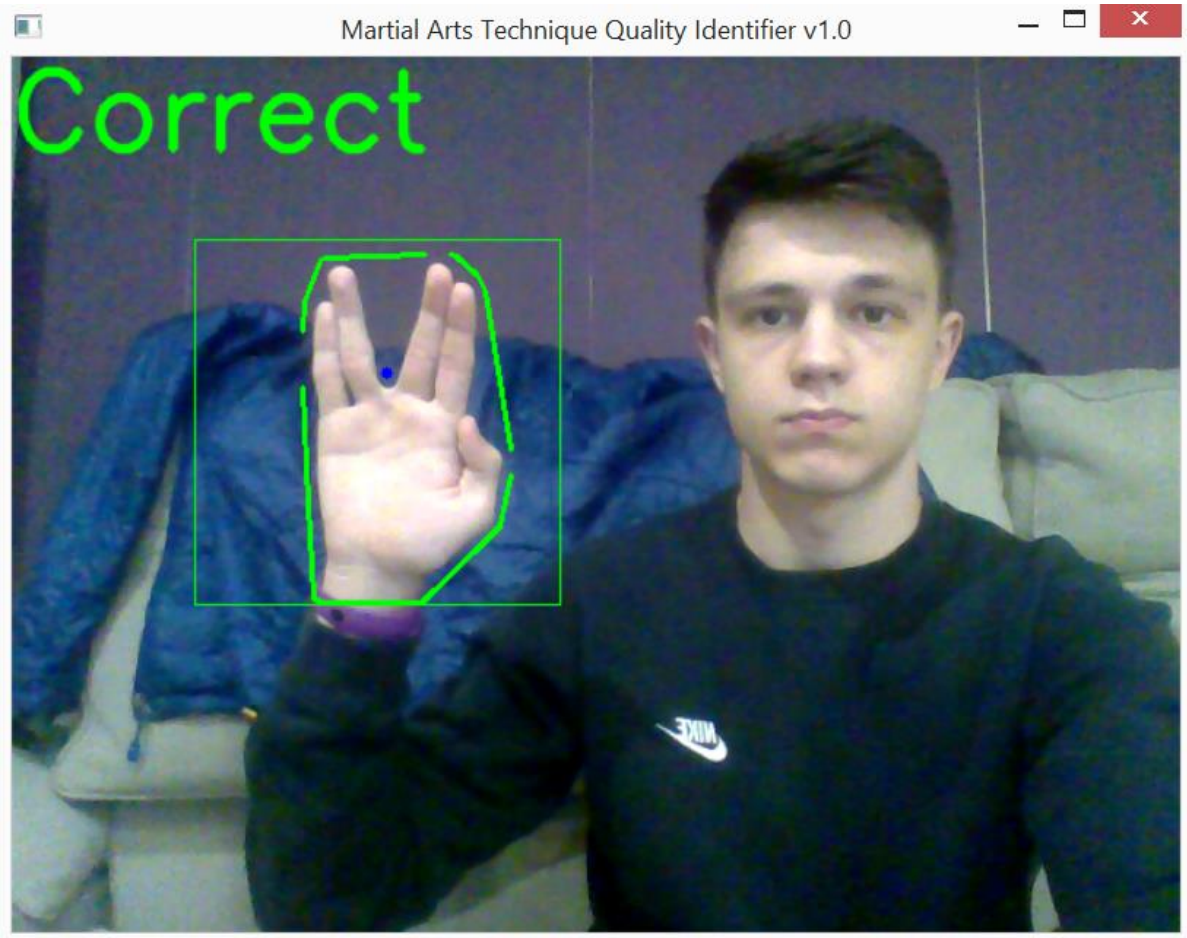



Рис. Е.4.1 Виконання вправи «Nukite»

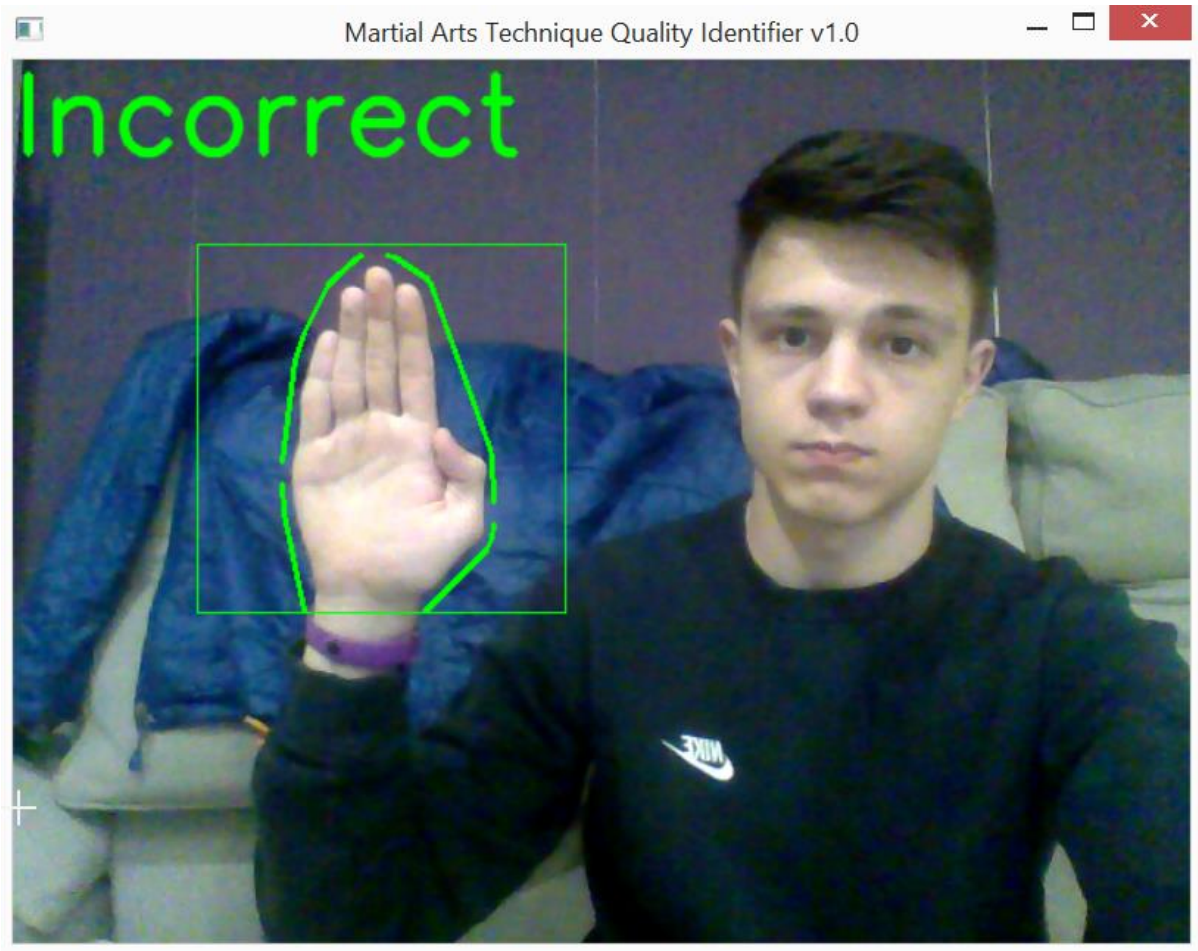



Рис. Е.4.2 Виконання вправи «Nukite»

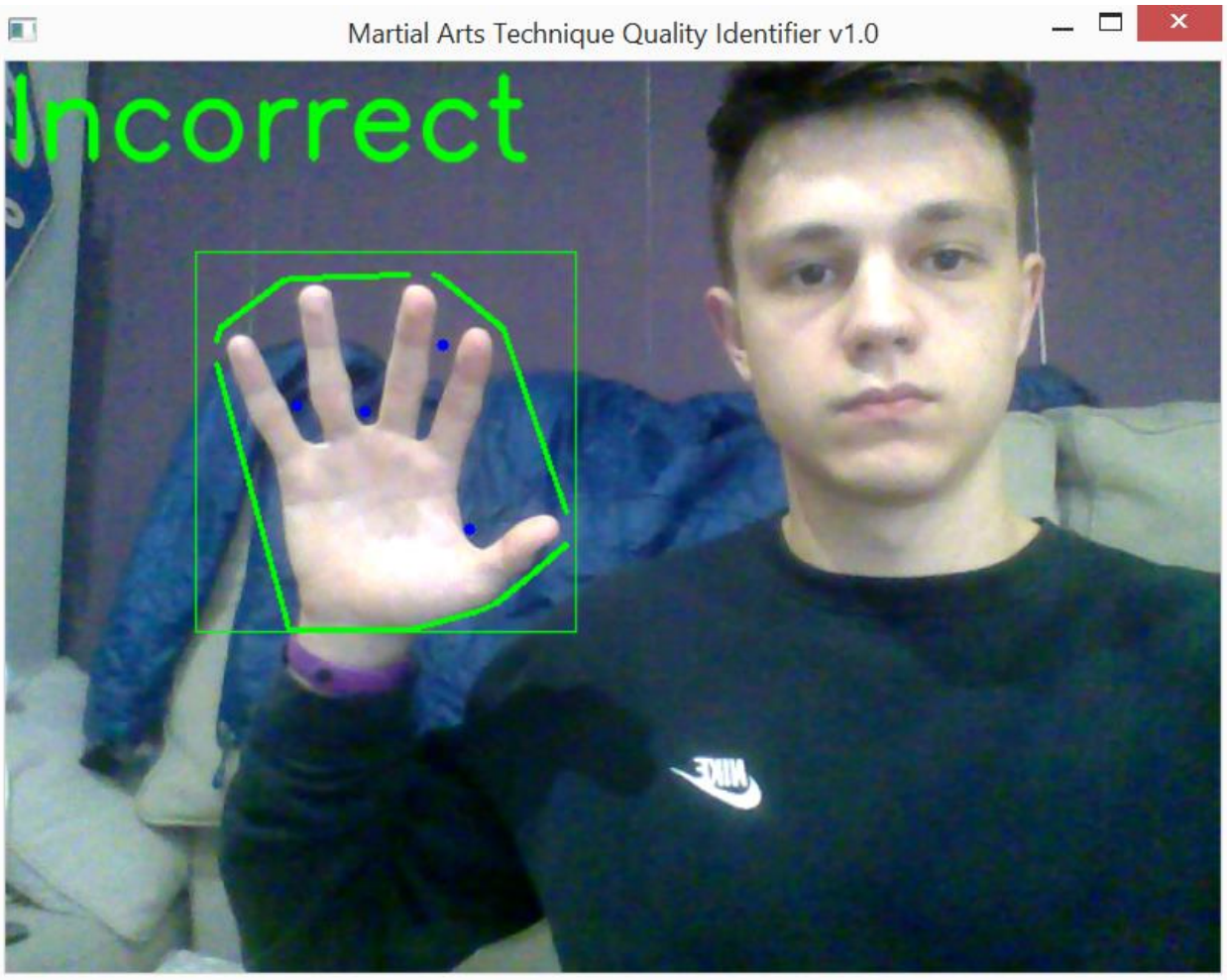



Рис. Е.4.3 Виконання вправи «Nukite»

Додаток Ж

**Taras Shevchenko National University of Kyiv**

# Software for evaluation of performance accuracy in martial arts techniques

## Software Architecture Document (SAD)

**Content Owner: Andrii Zhovnirskyi**

**DOCUMENT NUMBER: 1.0**

**RELEASE: 1.0**

**RELEASE DATE: 01.06.2021**

# TABLE OF CONTENTS

# 1. Introduction

## 1.1. Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

## 1.2. Scope

This Software Architecture Document provides an architectural overview of the Software for evaluation of performance accuracy in martial arts techniques. The software performs hand recognition and evaluation quality of hand exercises concerning martial arts, in real time.

System has 3 modules: input module, video stream module and image analysis module.

## 1.3. Definitions

Martial arts techniques – is set of special exercises which sportsmen must know and do perfectly to prove his skill and get his next mastery.

## 1.4. References

Applicable references are:

- Martial arts exercises
- Gesture recognition
- Python
- OpenCV
- Numeric Python

# 2. Architecture Representation

This document presents the architecture as a series of views; use case view, sequence view, statechart view. Software system implements console interface for input data and simple video threading from video camera to get image data of performed exercise.

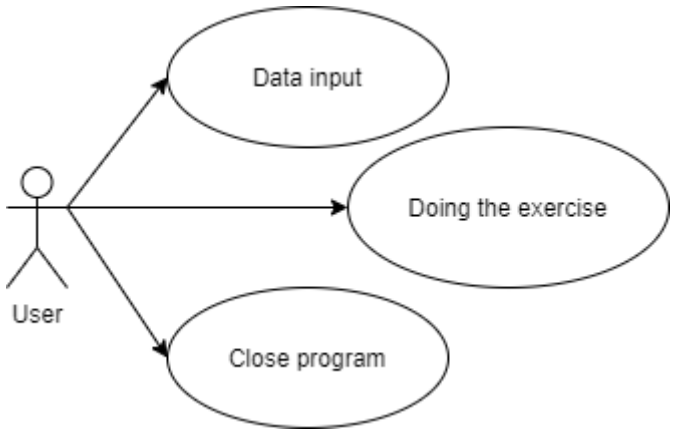
# 3. Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

- Video stream must be real-time.
- Evaluation point must be deployed during all the performance time.

**4. Use-case View**

A description of the use-case view of the software architecture. The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.



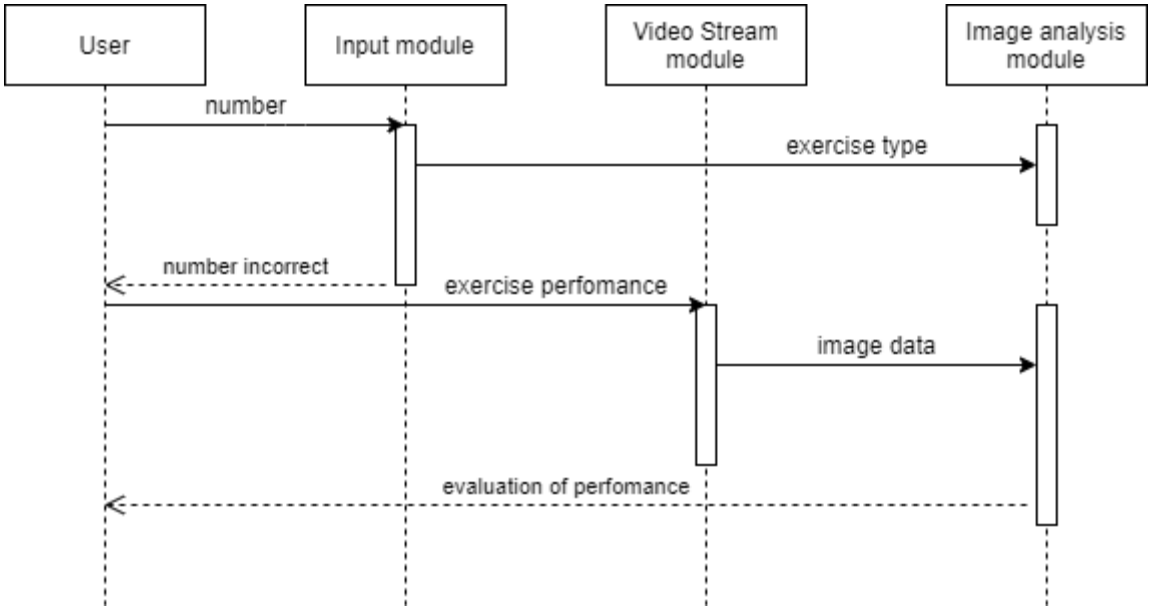

Programming system has 3 types of use-cases:

- Data input – user inputs number of the exercise that needs to evaluate from the list. If number is correct – video stream starts.
- Doing the exercise – when video stream started user needs to perform the exercise by making it fit in the highlighted square.
- Close program – if user needs to finish his work with the program and close it, he/she needs to push ‘‘esc’’ button.

## 5. Sequence View

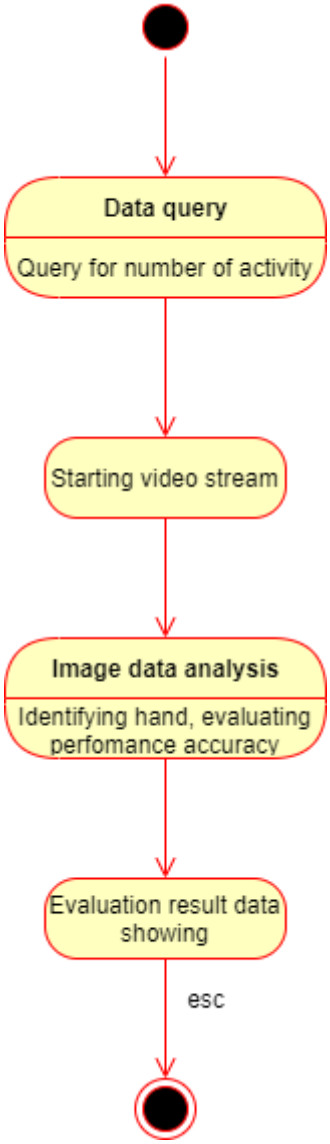
UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

The sequence view of the program is comprised of the 4 main parts: directly user, input module, video stream module, image analysis module. And it also has connections.




## 6. Statechart View

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.




When program starts it quers a data from the user. That state name is Data query. When data input passed the program goes to Starting video stream state. The next state is Image data analysis that provides evaluation analysis of performed exercise. And the last state is showing result of performance. Then, when ‘‘esc’’ button pressed the program is closed.

## 7. Size and Performance

The chosen software architecture supports the key sizing and timing requirements:

- The system shall support only 1 user at any given time of video stream.
- The video stream shall be displayed in real-time with no more than a 1 second latency.
- The system shall show evaluation result with no more than a 0.5 second latency.