**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**ІМЕНІ ТАРАСА ШЕВЧЕНКА**
Факультет комп'ютерних наук та кібернетики
Кафедра теоретичної кібернетики

«До захисту допущено»
Завідувач кафедри
Ю. В. Крак _____
(підпис)
«___» _____20___ р.

# Кваліфікаційна робота
## на здобуття ступеня бакалавра

за спеціальністю 122 Комп’ютерні науки
на тему:

## ДОСЛІДЖЕННЯ ПІДХОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ РОЗПІЗНАВАННЯ ЖЕСТІВ

Виконав студент 4-го курсу
Крищенко Владислав Романович _____
(підпис)

Науковий керівник:
професор, доктор фіз.-мат. наук
Крак Юрій Васильович _____
(підпис)

Засвідчую, що в цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ-2021

# РЕФЕРАТ

**Обсяг роботи:** 59 сторінок, 40 ілюстрацій, 1 таблиця, 69 джерел посилань.

**Об'єкт роботи:** методи розпізнавання жестів. Предметом роботи є реалізація системи на Python з використанням OpenCV.

**Мета роботи:** Дослідити існуючі технології та методи розпізнавання жестів та на їх базі створити свою систему розпізнавання жестів.

**Методи розроблення:** проаналізувати існуючі методи, вивчити наявні системи з розпізнавання, розробка системи.

**Інструменти розроблення:** мова програмування - Python, PyCharm – інтегроване середовище розробки (IDE) для мови Python, OpenCV – Open Computer Vision Library, бібліотека, яка надає все необхідне, для розпізнавання за допомогою комп'ютерного зору. Також під час написання кваліфікаційної роботи було використано сервіс Miro для створення діаграм

**Результати роботи:** Було досліджено наявні системи розпізнавання, способи їх реалізації та сфери використання, проаналізовано їх переваги та недоліки, і в результаті було розроблено власну систему розпізнавання

**Висновки та пропозиції щодо розвитку об'єкта розроблення й доцільності продовження розробок:** при подальшому розвитку дана система має великий потенціал у багатьох сферах діяльності людини, як у розвальній індустрії так і в таких галузях, як медицина та промисловість

**Ключові слова:** OpenCV; hand gesture recognition; HCI; hand tracking; classification; detection and segmentation; dactyl recognition; convolutional neural net-works; розпізнавання жестів; відстеження рук; класифікація; виявлення та сегментація; розпізнавання дактильної абетки; згорткові нейронні мережі.

# ЗМІСТ

# СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

HCI - Human-computer interaction

CAD - Computer-aided design

CAM - Computer-aided manufacturing

ANN - Artificial neural network

HMM - Hidden Markov model

OpenCV - Open Source Computer Vision Library

XML - eXtensible Markup Language

UI - User Interface

VXL - Vision-something-Libraries

TLD - Top-level domain

HOG - Histogram of oriented gradients

CNN - Convolutional neural network

# ВСТУП

**Актуальність роботи та підстави для її виконання**. Розпізнавання жестів рук за останні роки привертає велику увагу з боку науковців та промисловості завдяки очевидній перевазі над традиційними методами взаємодії людини та комп'ютера з точки зору зручності та комфорту. Ця сфера досліджувалась з різних точок зору, серед яких, підходи, засновані на комп'ютерному зорі, які забезпечують найбільш природні та інтуїтивні інтерфейси. У цій роботі представлений огляд методів та технологій розпізнавання жестів рук ,зокрема на основі комп'ютерного зору, з акцентом на динамічних жестах рук.

Отже, **метою** роботи є дослідження існуючих підходів для розпізнавання жестів рук людини, та на основі досліджених методів реалізувати свою систему розпізнавання

**Об'єкти та методи розроблення**. Об'єктом розроблення є системи для розпізнавання жестів. **Предметом роботи** є реалізація системи на Python з використанням OpenCV

# РОЗДІЛ 1. ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ОБРАЗІВ ТА ЖЕСТІВ

## 1.1 Історія та розвиток поняття людино-комп’ютерної взаємодії

**Людино-комп'ютерна взаємодія** *(англ. human-computer interaction,* ***HCI)*** – це засіб зв’язку між людиною, яка є користувачем комп'ютерною системи та самою цією системою, в яку входять різні пристрої вводу та виведення інформації за допомогою додаткового програмного забезпечення [1].

Такі терміни як: інтерфейс людина-система (HSI) та інтерфейс людина-машина (HMI) використовуються як синоніми.

Людино-комп'ютерна взаємодія розвивалася в різних сферах досліджень таких як: комп’ютерні науки, поведінкові науки, дизайн, медіа-дослідження та ряду інших галузей досліджень

Початком ергономічної фази людино-комп'ютерної взаємодії можна вважати з оприлюднення дисертації Айвена Сазерленда (Sutherland, 1963), яка дала початок розвитку комп'ютерної графіки, як окремої науки. Комп’ютерній графіці потрібні були ергономічні проекти за для ефективного управління складними моделями CAD / CAM систем. Man-machine symbiosis (Licklider, 1960), Augmentation of human intellect (Engelbart, 1963) і Dynabook (Kay, Goldberg, 1977) ці роботи продовжили дослідження в цих областях. І в результаті цих наукових досліджень з’явилися інструменти, без яких важко уявити зараз роботу з комп'ютером: «миша», поелементно-адресуєме (bitmap) відображення, «вікно», «робочий стіл», point-and-click-редактори [2].

В наш час все більше з’являється нових технології та пристроїв, які постійно ускладняються та стають доступнішими для взаємодії людини та комп'ютера. До таких пристроїв входять: графічні пристрої, сенсорні пристрої та пристрої голосового введення. Ці пристрої налаштовуються таким чином, щоб взаємодія людини з комп’ютером була максимально комфортною та ефективною. Дизайн інтерфейсу машини може включати технічні прийоми експертної системи, щоб запропонувати користувачеві потужні обчислення на основі отриманих знань.

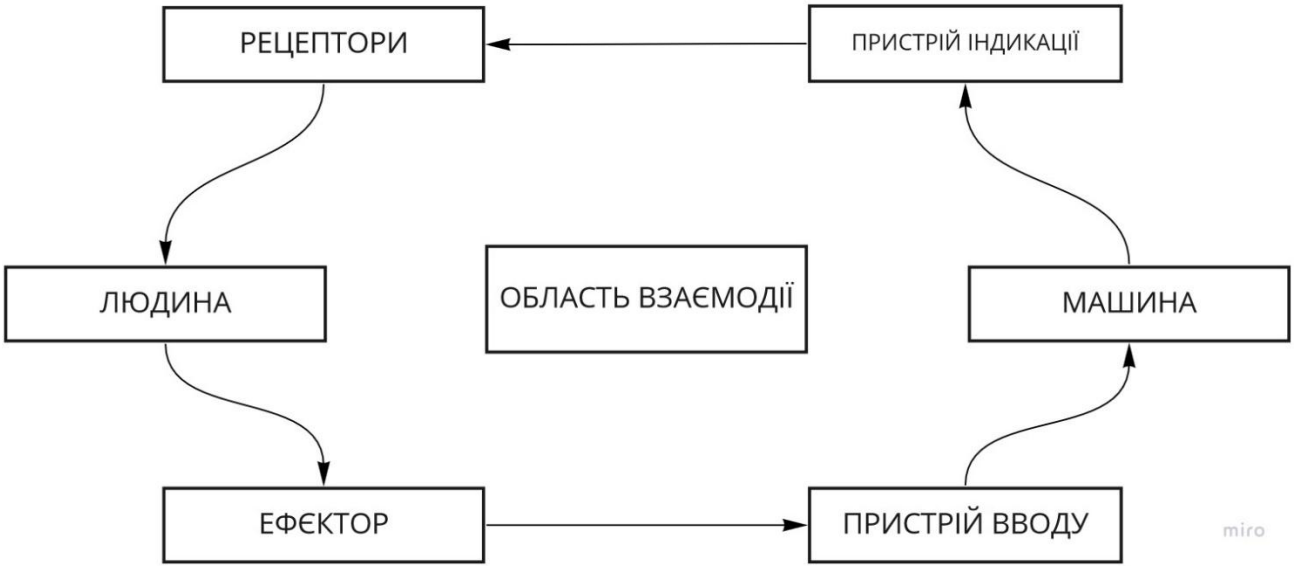



Рис. 1 Структура Людино-комп’ютерної взаємодії

Покращення взаємодії людини з комп’ютером за допомогою підвищення ергономічності в роботі з машиною є основною задачею людино-комп’ютерної взаємодії. Також HCI займається:

- методологією та розвитком проектуванням інтерфейсів (тобто, залежно від потреб та завдань, які потребують користувачі, розробки якісного UI в заданих рамках, оптимізації під необхідні властивості, такі як: здатність до навчання і ефективність використання);

- методами реалізації інтерфейсів (наприклад, програмні інструментарії, бібліотеки і раціональні алгоритми);
- методами для оцінки та порівняння таких інтерфейсів;
- розробкою нових інтерфейсів і методів взаємодії;
- теорією взаємодії

### 1.2 Людино-машинна взаємодія заснована на жестах людини

В останні декілька років все більше отримує уваги людино-комп’ютерна система, яка заснована на жестах людини.

Люди використовують жести для вираження емоцій та підкреслення інформації замість розмови, або під час неї.

Мова жестів багата на способи вираження людьми найрізноманітніших емоцій і значень, наприклад, образи, ворожості, дружелюбності або схвалення по відношенню до інших. Більшість людей використовують при розмові жести і мову тіла на додаток до слів. Багато жестів використовуються людьми підсвідомо [3].

HCI може використати жести для передачі інформації від людини до машини, машина ж в свою чергу може використати цю інформацію для виконання певних завдань.


Рис. 2 Сфери використання людино-комп’ютерних систем

Розпізнавання жестів можна застосовувати в таких областях:

1. Управління комп'ютером і побутовими приладами
    - В певному комп’ютерному додатку постійно зіставляються жести рук людини (які зчитуються за допомогою певного пристрою) та комп’ютерної команди. Тобто людина показує жест, система його розпізнає та відправляє команду комп’ютеру
    - Позиція долоні зіставляється з курсором миші і команди натискання кнопок миші зіставляються з конфігураціями руки
2. Створення природних людино-машинних інтерфейсів для глухонімих
    - Систему розпізнавання жестів можна застосувати для введення тексту в комп'ютер за допомогою жестів руки, що для людей з вадами слуху значно полегшить роботу та спілкування за допомогою комп’ютера.
3. Маніпуляція тривимірними моделями об'єктів

- Маючи 3D координати руки і кінчиків пальців, можна створити систему, яка дозволить керувати моделями в усіх напрямках 3D простору. Що повинно значно полегшити роботу для багатьох професій в першу чергу для комп'ютерних дизайнерів.

4. Додатки віртуальної реальності
    - Якщо ускладнити систему розпізнавання жестів додавши до неї додаткові пристрої, наприклад шолом віртуальної реальності, можна буде створити дуже реалістичні додатки, де користувач зможе взаємодіяти з об'єктами у тривимірному просторі.

## 1.3 Комп'ютерний зір. Методи обробки зображення

### 1.3.1 Загальні методи розпізнавання жестів

В наш час було отримано багато результатів досліджень щодо розпізнавання жестів на основі традиційних методів, які можна класифікувати за трьома категоріями, а саме: методи, засновані на узгодженні шаблонів, методи, засновані на штучних нейронних мережах і методи, засновані на імовірнісних статистичних моделях [4].

Методи узгодження на основі шаблонів засновані на виборі відповідних ознак, які можуть бути або регіональними ознаками або рисовими особливостями жесту. Вектори об'єктів збираються із зображення жесту для розпізнавання та порівняння його з відповідними векторами об'єктів у збереженому шаблоні, а відстань між зображеннями жесту та шаблону обчислюється для визначення класу зображення жесту, який слід розпізнавати за мінімальним дистанційним методом [5]. Цей метод є найбільш простим і зрозумілим методом узгодження при розпізнаванні жестів. Однак обмеження

цього методу полягає в тому, що він є більш чутливим до такої інформації, як розмір та обертання вхідного зображення. Швидкість розпізнавання цього


методу поступово зменшується із збільшенням типів жестів.

Рис. 3 Процес розпізнавання жестів за методом узгодження шаблонів

Штучні методи, засновані на нейронних мережах, були більш цікавими методами для дослідників до появи глибоких нейронних мереж. Штучні нейронні мережі (ANN) працюють розподіленим чином і мають класифікаційні властивості навчання. Теоретично штучні нейронні мережі можуть апроксимувати довільно складні нелінійні відображення. Традиційна структура штучної нейронної мережі обмежена неглибокою нейронною мережею, яка, як правило, містить лише просту структуру вхідного шару, прихованого шару та вихідного шару. Отже, можливість вивчати особливості є більш обмеженою, що також робить кінцевий ефект розпізнавання середнім.

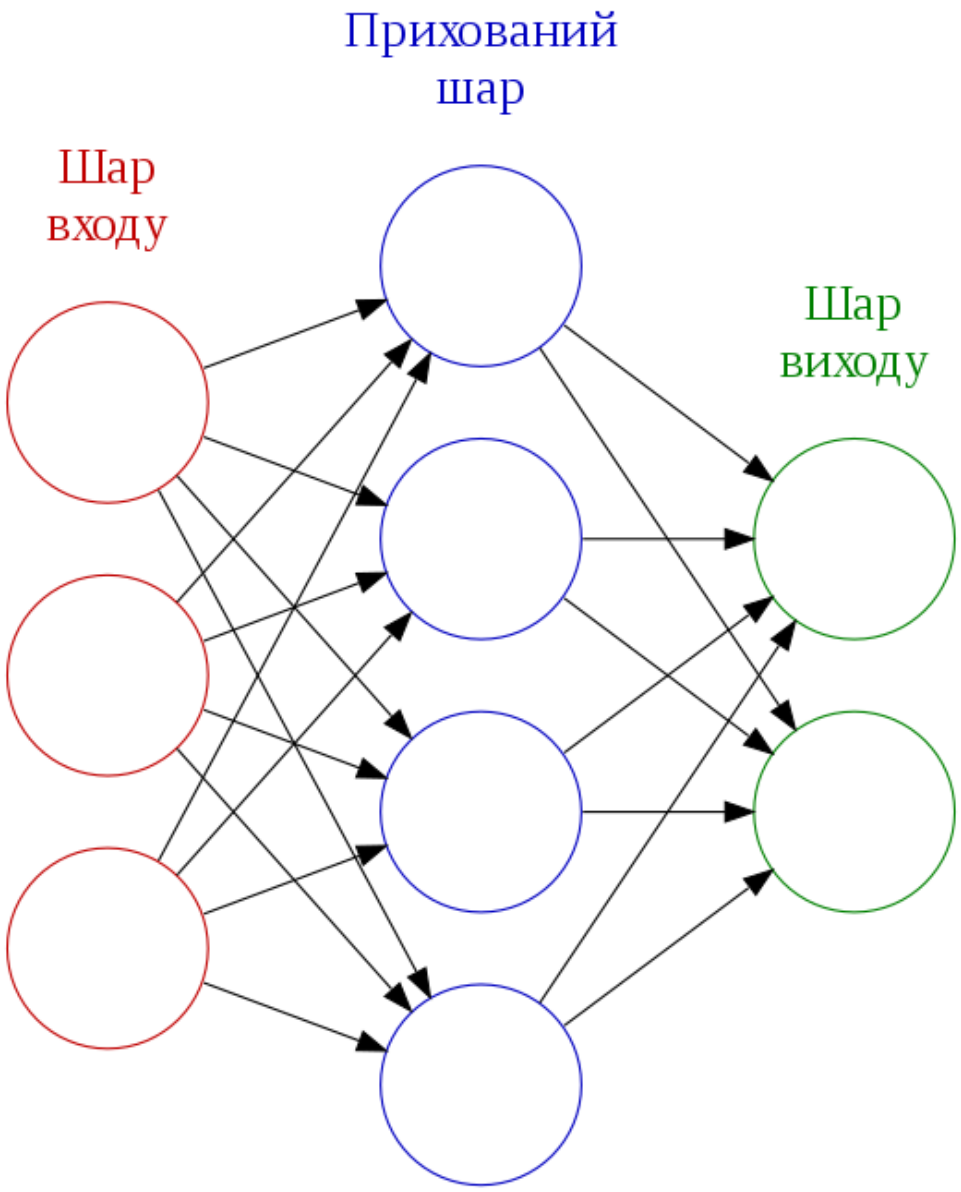
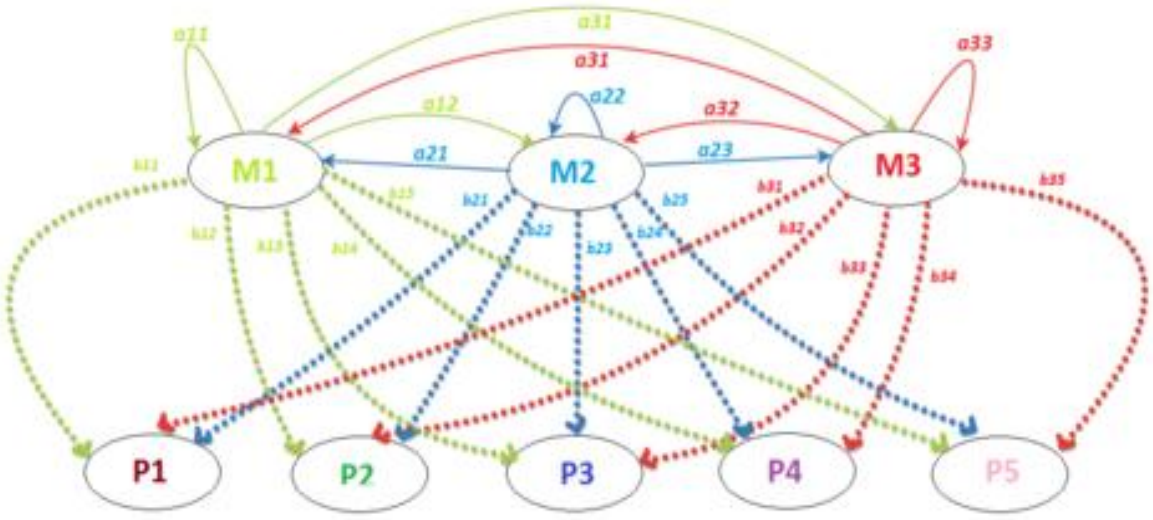



Рис. 4. Приклад штучної нейронної мережі

Методи, засновані на імовірнісних статистичних моделях, як правило, набувають деяких особливостей жесту та імовірнісним чином класифікують поточний жест за ознаками. Типовим представником цього підходу є прихована Марковська модель(HMM). Цей метод включає подвійний стохастичний процес передачі стану та виходу спостереження, який вимагає більшої кількості параметрів і призводить до більш низької швидкості розпізнавання. А проміжна швидкість стану передачі невідома, тому можна виконати лише нечітке




обчислення, що також впливає на кінцевий ефект розпізнавання класифікації.

Рис. 5. Приклад прихованої марковської моделі

### 1.3.2 Базовий процес розпізнавання жестів

Базовий процес розпізнавання жестів можна описати наступним чином:

- По-перше, потік інформації про зображення фіксується через камеру, після чого проводиться попередня обробка, наприклад, фільтрація галасливих даних (пошук та сегментація жестів);

- Потім він вводиться в систему розпізнавання, і вилучення особливостей виконується за допомогою глибокого навчання для функцій жестів;
- І вже в кінці, алгоритм класифікації завершується навчанням та штучною граматикою, визначеною системою для розпізнавання


результатів [6].

Рис. 6 Базовий процес розпізнавання жестів

Процес вилучення та аналізу особливостей значно відрізняється для статичних та динамічних жестів. Статичні жести потрібно оцінювати лише різними методами на зображеннях, тоді як динамічні жести - це процес змін,

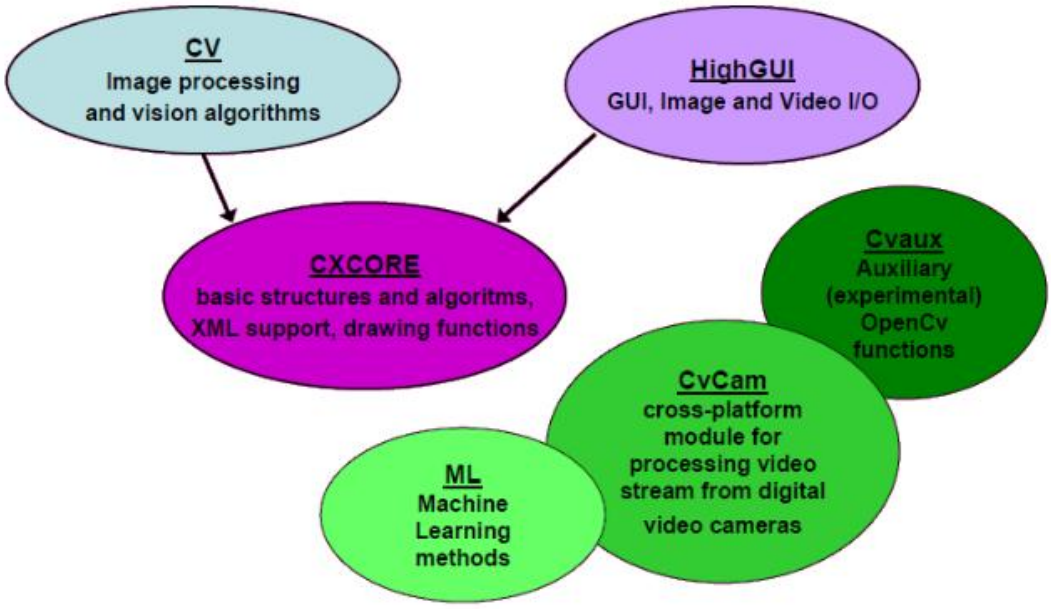
певної сукупності дій протягом певного періоду, тому системі потрібно визначити початкові та кінцеві позначки часу динамічних жестів. Динамічні жести, на певному проміжку часу, якимось чином можуть стати статичними жестами.

Візуальне розпізнавання жестів зосереджується на обробці зображень, а вилучення жесту безпосередньо впливає на кінцевий результат розпізнавання [7]. Традиційне розпізнавання жестів для комп'ютерного зору містить різні алгоритми, і як покращити послідовність та надійність розпізнавання жестів було в центрі уваги досліджень багатьох досліджень.

Базовий процес розпізнавання жестів показаний на рисунку 6.

### 1.3.3 Бібліотека OpenCV

Це бібліотека, яка до 1-ї версії розроблялася в Центрі розробки




програмного забезпечення Intel.

Рис. 7 Структура бібліотеки OpenCV

OpenCV написана на мові високого рівня (C / C ++) і містить алгоритми для: інтерпретації зображень, калібрування камери за зразком, усунення оптичних спотворень, визначення подібності, аналіз переміщення об'єкта, визначення форми об'єкту та відслідковування об'єкта, 3D-реконструкції, сегментації об'єкта, розпізнавання жестів і т.д.

Ця бібліотека дуже популярна за рахунок своєї відкритості та можливості безкоштовно використовувати її як в навчальних, так і комерційних цілях.

Фактично, OpenCV - це набір типів даних, функцій і класів для обробки зображень алгоритмами комп'ютерного зору.

Основні модулі бібліотеки:

- cxcore – ядро (Містить базові структури даних і алгоритми):
  - базові операції над багатовимірними числовими масивами
  - матрична алгебра, математичні ф-ції, генератори випадкових чисел
  - Запис / відновлення структур даних в/з XML
  - базові функції 2D графіки
- CV - модуль обробки зображень і комп'ютерного зору
  - базові операції над зображеннями (фільтрація, геометричні перетворення, перетворення колірних просторів і т.д.)
  - аналіз зображень (вибір відмінних ознак, морфологія, пошук контурів, гістограми)
  - аналіз руху, спостереження за об'єктами
  - виявлення об'єктів, зокрема осіб
  - калібрування камер, елементи відновлення просторової структури
- Highgui - модуль для введення/виведення зображень і відео, створення призначеного для користувача інтерфейсу
  - захоплення відео з камер та з відео файлів, читання/запис статичних зображень.

- функції для організації простого UI (всі демо додатки використовують HighGUI)

- Cvaux - експериментальні і застарілі функції
  - просторовий зір: стерео калібрації, само калібрації
  - пошук стерео-відповідності, кліки в графах
  - знаходження і опис рис обличчя
- CvCam - захоплення відео
  - дозволяє здійснювати захоплення відео з цифрових відео-камер (підтримка припинена і в останніх версіях цей модуль відсутній)

### 1.3.4 Бібліотека VXL

VXL (Vision-something-Libraries) - це колекція бібліотек на C ++, призначена для дослідження та впровадження комп'ютерного зору. Він був створений на основі TargetJr та IUE з ціллю створити послідовну систему, яка буде одночасно легкою та швидкою. VXL написаний на ANSI / ISO C ++ і призначений для більшості платформ. Основними бібліотеками VXL є:

- vnl (numerics): числові контейнери та алгоритми. напр. матриці, вектори, розкладання, оптимізатори.
- vil (візуалізація): завантаження, збереження та обробка зображень у багатьох поширених форматах файлів, включаючи дуже великі зображення.
- vgl (геометрія): геометрія для точок, кривих та інших елементарних об'єктів у 1, 2 або 3 вимірах.
- vsl (потокове введення / виведення), vbl (основні шаблони), vul (утиліти): Різна функціональність, незалежна від платформи.

Окрім основних бібліотек, існують бібліотеки, що охоплюють числові алгоритми, обробку зображень, системи координат, геометрію камери, стерео, маніпуляції з відео, відновлення структури з руху, моделювання ймовірностей, дизайн графічного інтерфейсу, класифікацію, надійну оцінку, відстеження особливостей, топологію , маніпуляції зі структурою, 3d-зображення та багато іншого.

Кожна основна бібліотека є легкою і може використовуватися без посилання на інші основні бібліотеки. Подібним чином, непрофільні бібліотеки не залежать одна від одної, тому ви можете компілювати та зв'язувати лише ті бібліотеки, які вам дійсно потрібні.

Портативність та ефективність колекції vxl - це риси, які значною мірою зумовлені її походженням. Пакет був побудований шляхом вилучення основних функціональних можливостей двох великих систем: "Image Understanding Environment" (IUE) та "Target junior" (TargetJr). Хоча ці середовища містили безліч корисного коду та сприяли проведенню багатьох досліджень, вони широко відомі своєю «масою» та труднощами при навчанні ними користуватися. Проблеми, які найбільше поширені при використанні VXL:

- "Занадто важкий": Щоб використовувати навіть одну невелику процедуру, велика частина середовища повинна бути включена в програму користувача, збільшуючи розмір програми, час компіляції та час запуску. Іноді збільшення стає таким, що налагоджувачі та інші інструменти не працюють.
- "Занадто повільний": У деяких випадках дизайн попередніх систем обмежував ефективність коду, який можна було написати при їх використанні. Частково це сталося через недоліки використовуваних компіляторів C ++, а також через використання замінених з тих пір парадигм програмування.

- "Занадто незрозумілий": Оскільки IUE - це велика система, призначена для використання як середовище програмування, вона накладає стилістичні химерності на програмістів, які її використовують. Найбільш очевидним з них було використання LaTeX для генерації коду, який мав багато переваг, але в кінцевому підсумку був відхилений користувачами.

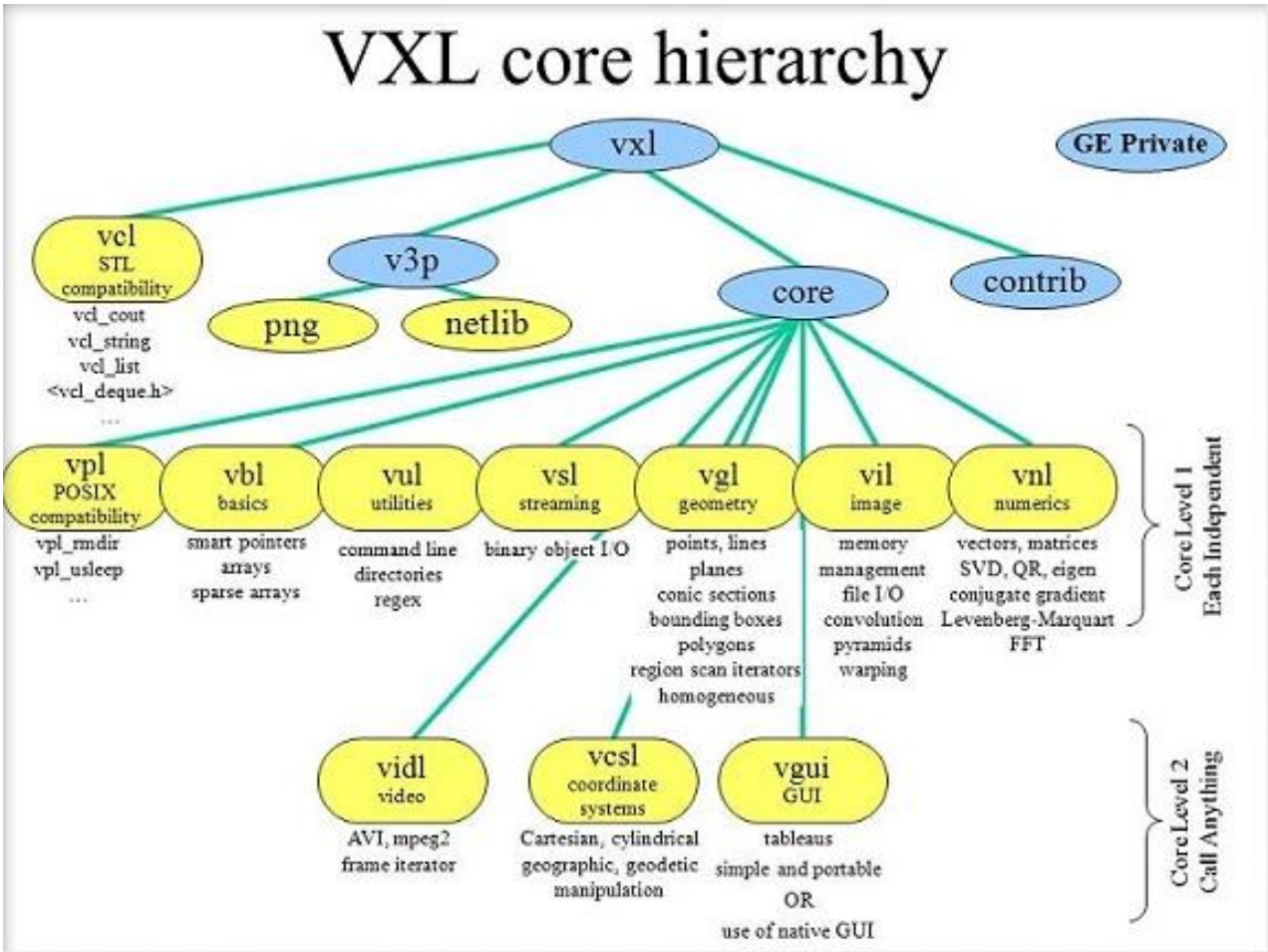


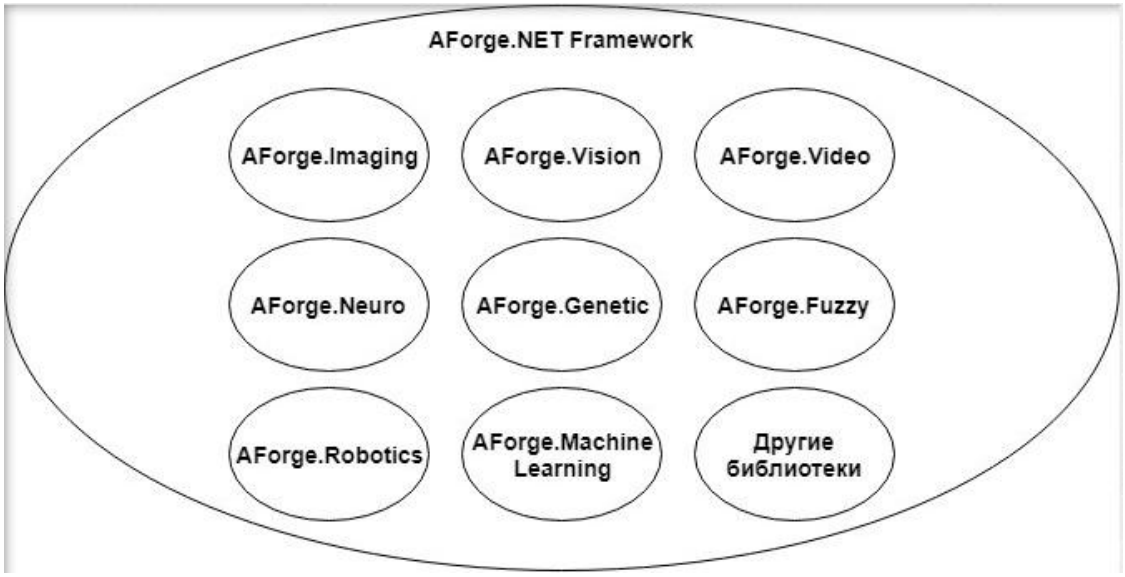

Рис. 8 Ієрархічна будова ядра VXL

### 1.3.5 Бібліотека AForge.NET

AForge.NET є бібліотекою з відкритим вихідним кодом, створеної на мові C #, яка призначена для розробників і дослідників в області комп'ютерного зору.

Крім того, в бібліотеці є функціонал для розробників в області штучного інтелекту. Спектр можливостей бібліотеки досить широкий: обробка зображень, нейронні мережі, генетичні алгоритми, нечітка логіка, машинне навчання, робототехніка і багато іншого [8].

- Бібліотека включає кілька основних компонентів:
  - AForge.Imaging - бібліотека підпрограм для обробки зображень і фільтрів.
  - AForge.Vision - бібліотека комп'ютерного зору.
  - AForge.Video - набір бібліотек для роботи з відео.
  - AForge.Neuro - бібліотека для виконання різноманітних дій і операцій з нейронними мережами.
  - AForge.Genetic - бібліотека підпрограм для використання генетичних алгоритмів з метою вирішення різних завдань.
  - AForge.Fuzzy - бібліотека для роботи з нечіткою логікою.
  - AForge.Robotics - бібліотека, яка забезпечує підтримку деяких методів, що застосовуються в сфері робототехніки.
  - AForge.MachineLearning - бібліотека для роботи з елементами



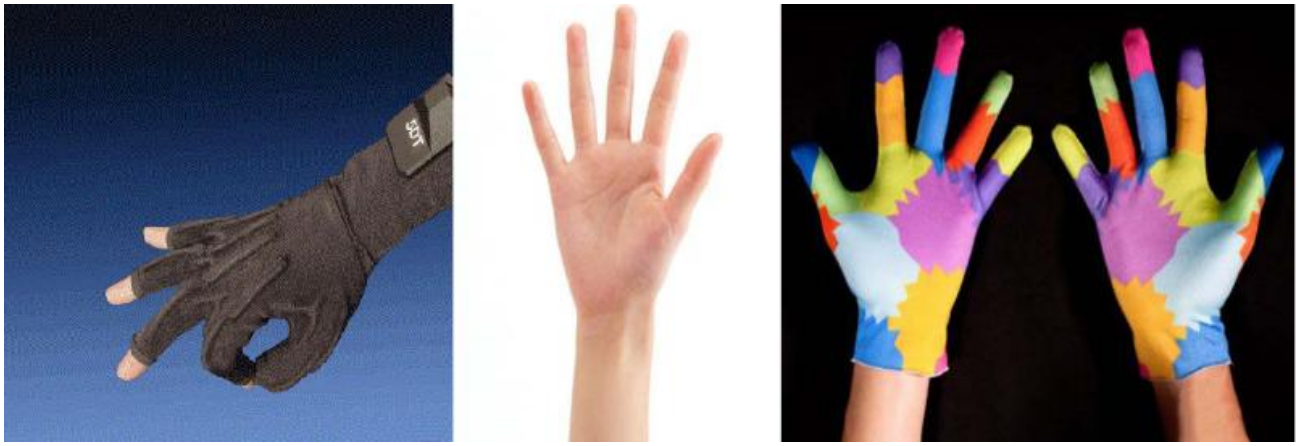
машинного навчання [9]

Рис. 9 Структура фреймворку AForge.NET

AForge.NET постійно поліпшується і прогресує. З цією бібліотекою існує велика кількість прикладів, які демонструють її роботу, і html-документація, яка підтримується в актуальному стані і допомагає розробникам у використанні даного фреймворка. Існує, так само як і у бібліотеки OpenCV, активна спільнота, в якій можна отримати необхідну інформацію, ставлячи питання розробникам, або поділитися власними результатами. Але, на жаль, кількість учасників цієї спільноти поступається своєю кількістю аналогічним з OpenCV. Ще одним обмеженням на шляху розробника є той факт, що вся документація по бібліотеці написана тільки на англійській мові. Зважаючи на це можливі труднощі у вивченні і освоєнні даного фреймворку. На рис. 9 приведена загальна структура бібліотеки AForge.NET.

# РОЗДІЛ 2 ІСНУЧІ ТЕХНОЛОГІЇ ТА МЕТОДИ РОЗПІЗНАВАННЯ ЖЕСТІВ РУК

Для забезпечення розпізнавання жестів рук запропоновано численні підходи, які можна класифікувати на різні категорії. Загальна систематика базується на тому, чи потрібні додаткові пристрої для збору необроблених даних. Таким чином, найчастіше їх класифікують на розпізнавання жестів рук на основі провідних рукавичок [10], розпізнавання жестів рук на основі комп'ютерного зору [11] та розпізнавання жестів рук на основі кольорових



рукавичок [12] (див. Рис. 10).

Рис. 10 Розпізнавання жестів рук на базі провідних (інформаційних) рукавичок, комп'ютерного зору та кольорових рукавичок (зліва направо).

## 2.1 Розпізнавання жестів на основі провідних рукавичок

Підходи, що базуються на провідних рукавичках, вимагають, щоб користувач носив громіздкий рукавичкоподібний пристрій, який оснащений датчиками, які можуть відчувати рухи рук і пальців і передавати інформацію на

комп’ютер. Перевагами цих підходів є висока точність і швидка реакція. Однак ці методи не надто природні та гнучкі, і рукавички для передачі даних можуть

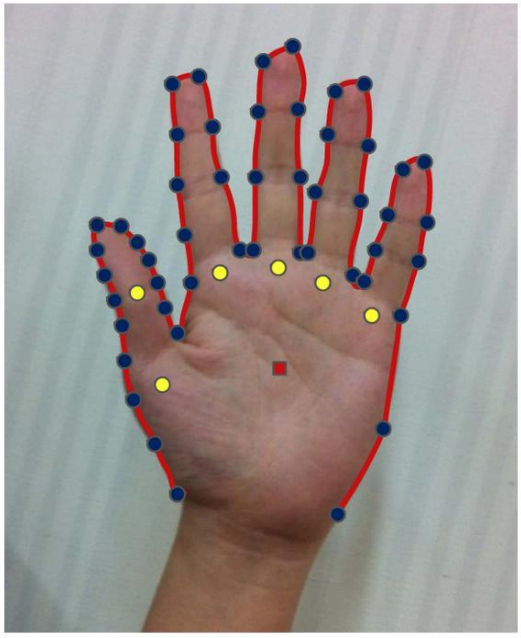
бути досить дорогими.

Рис. 11 Приклад провідної (інформаційної) рукавички

## 2.2 Розпізнавання жестів на основі комп’ютерного зору

Підходи, засновані на комп’ютерному зорі, не вимагають від користувача що-небудь носити. Натомість відеокамери використовуються для зйомки зображень рук, які потім обробляються та аналізуються за допомогою техніки комп’ютерного зору. Цей тип розпізнавання жестів рукою є простим, природнім та зручним для користувачів, і в даний час це найбільш популярні підходи до розпізнавання жестів. Однак є ще кілька проблем, які потрібно вирішити, наприклад, зміна освітлення, захаращення фону, часткова або повна оклюзія тощо. Розпізнавання жестів рук на основі комп’ютерного зору можна додатково

класифікувати на підходи, засновані на зовнішньому вигляді, та підходи на основі 3D-моделі [13]. Підходи на основі зовнішнього вигляду витягують низькорівневі функції з 2D-зображень і порівнюють їх із заздалегідь визначеними шаблонами жестів. Вони відносно прості в обчисленні, але втрата інформації про глибину робить її сприйнятливою до фонових порушень. На відміну від цього, підходи, засновані на 3D-моделях, можуть використовувати інформацію про глибину і набагато дорожчі в обчислювальному плані, але



можуть ефективніше визначати жести рук.

Рис. 12 Використання алгоритмів комп’ютерного зору

## 2.3 Розпізнавання жестів на основі кольорових рукавичок

Підходи, засновані на кольорових рукавичках, представляють компроміс між підходами, заснованими на провідних рукавичках та методами, які використовують комп’ютерний зір. Власне, вони подібні до останнього методу, за винятком того, що за допомогою кольорових рукавичок фаза попередньої обробки зображень (наприклад, сегментації, локалізації та виявлення рук) може

бути значно спрощена. Недоліки подібні до підходів, заснованих на методах в яких використовуються провідні рукавички: вони є неприродними і не

підходять для додатків із великою кількістю користувачів.

Рис. 13 Приклад використання кольорової рукавички

### 2.4 Технології та ключові алгоритми розпізнавання жестів

Розпізнавання жестів рук включає три важливі фази:

1. Виявлення та сегментацію
2. Відстеження
3. Класифікацію

Частина виявлення та сегментації полягає у виявленні рук та сегментуванні відповідних областей зображення з фону. Відстеження використовує просторову та часову інформацію послідовних кадрів зображень для оцінки траєкторій руху рук. Класифікація використовує траєкторії рук, як вхідні дані для ідентифікації конкретних типів жестів. Структура типової системи розпізнавання жестів рукою показана на рис. 14.


Рис. 14 Структура типової системи розпізнавання жестів рук

### 2.4.1 Виявлення та сегментація

Ця частина належить до етапу попередньої обробки, який визначає області зображень, що відповідають завданню, і усуває фон. Це дуже важливо, і складно з тієї причини, що воно генерує вихідні дані для наступних етапів відстеження та класифікації, а у реальних ситуаціях фон може бути дуже галасливим. Для динамічного розпізнавання жестів рук головним чинниками є рух рук замість їх статичної пози, і обчислювальна складність повинна бути якомога меншою. Отже, часто надають перевагу особливостям низького рівня та простим методам, таким як: колір шкіри, форма та відкидання фону.

#### 2.4.1.1 Колір шкіри

Визначення кольору шкіри - це найпопулярніший метод сегментації через його простоту та зручність. Вибір кольорового простору є ключовим фактором для виявлення кольору шкіри. Оскільки кольоровий простір RGB чутливий до

умов освітлення і має відносно високі обчислювальні витрати, є кілька інших ортогональних кольорових просторів, які можуть краще відокремити кольоровість від компонентів яскравості, такі як HSV [14], YCbCr [15], YUV [16] та YIQ [17-18]. У цих кольорових просторах інформація про кольоровість часто представлена одним або двома компонентами, які використовуються для сегментації. У кольоровому просторі HSV інформація про колір представлена компонентом H (відтінок) та S (насиченість) і незмінною до зміни V (значення) протягом більшої частини часу. Як результат, H та S часто вибирають для подання інформації про колір. У колірному просторі YCbCr компонент яскравості зображення представлений Y, тоді як компонент кольоровості - Cb та Cr.

Для обраного кольорового простору загальними методами сегментації є:

1) використання постійного порогу або адаптивного порогу, вибраного емпірично, для ідентифікації ділянок шкіри, які зазвичай називають шкірними фільтрами;
2) обчислення подібності між заздалегідь визначеною моделлю кольору шкіри та вхідним зображенням (тобто ймовірність того, що піксель або блок пікселів є частиною руки)

Перевагами сегментації кольору шкіри є:

1) вона обчислювально недорога і може бути реалізована в режимі реального часу;
2) вона незмінна проти обертання, часткового стискання та зміни пози.

Однак цей метод не здатний вирішити завдання задовільно, оскільки:

1) він чутливий до зміни освітленості;
2) він не може обробляти складний фон з кольорами, схожими на шкіру, або іншими предметами шкіри, такими як людське обличчя та рука людини;

3) багато з цих підходів обмежені попередньо визначеними моделями і недостатньо надійні.

#### 2.4.1.2 Контур

Геометричні особливості рук також можна використовувати для виявлення різними способами. Контур рук можна отримати, застосовуючи оператори виявлення країв. Порівняно з виявленням кольору шкіри, контурна характеристика має перевагу в тому, що вона не залежить від освітленості та варіації кольору шкіри. Однак його ефективності можуть заважати оклюзії та зміна точки зору. Крім того, виявлення країв часто може призвести до великої кількості помилкових точок, які належать фоновим об'єктам. Отже, інтеграція форми, текстури та кольорових особливостей може забезпечити кращі показники [19].

Існує кілька методів, зосереджених на формі рук, таких як виявлення кінчиків пальців [20]. Однією із підказок для виявлення кінчиків пальців є, викривлення [21]. Зіставлення шаблонів - ще одна техніка, яка часто застосовується при виявленні кінчиків пальців. Шаблонами можуть бути зображення кінчиків пальців [22] або самі пальців [23]. Однак узгодження шаблонів має деякі недоліки:

1) Складні в обчисленні;
2) Вони не можуть впоратися зі зміною масштабу або обертанням руки;
3) Зовнішній вигляд руки може різко змінюватися, що ускладнює вибір правильного шаблону.

### 2.4.2 Відстеження

Проблема відстеження рук головним чином пов'язана з тим, що рука є складним об'єктом і не може розглядатися як жорсткий об'єкт, а швидкість жестів може бути досить швидкою. Типовими алгоритмами відстеження рук є: оптичний потік, Camshift, фільтрація Калмана, фільтрація частинок, алгоритм

конденсації тощо. Всі ці алгоритми базуються на певних припущеннях, а інтеграція різних алгоритмів часто може принести кращі результати.

#### 2.4.2.1 Оптичний потік

Методи оптичного потоку - це піксельні підходи до оцінки руху об’єкта, де передбачається пряма залежність між рухом об’єкта та змінами інтенсивності в межах послідовності зображень [24]. Серед існуючих методів оцінки оптичного потоку методи на основі градієнта характеризуються припущенням про сталість яскравості та просторово-часову гладкість [25].

Основною перевагою використання оптичного потоку при відстеженні є те, що воно не вимагає жодних попередніх знань про зовнішній вигляд об’єкта. Однак ця перевага може бути порушена через високу обчислювальну складність. Для додатків розпізнавання жестів рук у режимі реального часу часто використовується метод Лукаса – Канаде, заснований на моделі руху блоку, який не містить ітераційної процедури [26].

Методи, засновані на оптичному потоці, добре працюють, коли об'єкти рухаються не дуже швидко, і є лише кілька рухомих об'єктів. Для додатків, що вимагають високої точності, таких як навігація робота, самого оптичного потоку може бути недостатньо через фактор дрейфу, який викликаний накопиченням помилок. У цій ситуації використання інформації про колір при оптичному відстеженні потоку може привести до кращих результатів [27].

#### 2.4.2.2 Camshift

Camshift, скорочене від “Безперервно адаптивний середній зсув”, - це алгоритм, заснований на середньому алгоритмі зсуву. Спочатку він використовувався для відстеження обличчя [28] і досяг гарних результатів. Він надійний проти шуму та відволікаючих факторів. Однак лише цей алгоритм не може отримати хорошої продуктивності відстеження рук. Причина полягає в тому, що Camshift визначає ціль за допомогою прямокутника, тоді як рука є

об’єктом з великою кількістю увігнуто-опуклих ділянок, які не можуть бути добре представлені прямокутником. Для відстеження рук необхідно коригування вихідного алгоритму, наприклад, включення фільтрації Калмана [29].

#### 2.4.2.3 Фільтрування частинок

Фільтрування частинок може використовуватися для відстеження об’єктів на щільно захараченому тлі. Він виконує випадковий пошук, щоб отримати оцінку заднього розподілу, що описує конфігурацію об’єкта. Порівняно з фільтрацією Калмана, цей підхід має очевидну перевагу, оскільки він не обмежений унімодальним характером гауссових щільностей, які не можуть одночасно представляти альтернативні гіпотези. Однак у фільтру є проблема: явище виродження [30]. Після кількох ітерацій всі зразки, крім одного, матимуть незначну вагу, і велику кількість обчислювальних зусиль буде витрачена даремно.

Для вирішення проблеми виродження, використовується фільтр Кальмана для створення складних розподілів пропозицій, які легко інтегрують поточне спостереження [31]. Тим часом алгоритм конденсації, який був використаний для навчання відстеження кривих на захараченому тлі, може досягти кращої ефективності, ніж фільтри Кальмана, а також роботи в реальному часі [32]. Фільтрація частинок та алгоритм середнього зсуву також можуть бути поєднані для збереження переваги середнього зсуву при вирішенні проблеми виродження фільтрації частинок[33].

#### 2.4.2.4 TLD

Зовсім новий алгоритм відстеження під назвою TLD (Tracking-Learning-Detection) [34] викликав величезний інтерес в науковців. Він використовує трекер та детектор для самостійного відстеження об’єкта, тоді як «учень» оцінює помилки детектора та оновлює його, щоб уникнути подібних помилок у

майбутньому. Завдяки поєднанню цих трьох модулів на різних тестових зразках були отримані конкурентні результати відстеження. Однак, застосовуючи його до ручного відстеження, він часто не може досягти такого ж хорошого результату, як на інших об'єктах, таких як наприклад автомобілі. Причина знову полягає в специфічних особливостях людських рук та мінливих зовнішніх проявів, які дуже важко вчасно оцінити.

### 2.4.3 Класифікація

Для розпізнавання жестів рук завершальним етапом є класифікація. Вона використовує інформацію, вилучену та оброблену попередніми кроками, і видає результати розпізнавання.

Існують різні підходи до розпізнавання жестів, які можна приблизно розділити на дві категорії:

- методи, засновані на математичних моделях, таких як Модель прихованого Маркова (HMM) і Кінцева машина (FSM),
- методи, засновані на м’яких обчисленнях, таких як мережевий нейтралітет.

Серед них HMM є найбільш часто використовуваним підходом до розпізнавання жестів.

HMM - це марковська модель, в якій модельована система вважається марковським процесом з неспостережуваними (прихованими) станами [35]. Набір прихованих параметрів визначається із набору пов’язаних спостережуваних параметрів. У ранні часи HMM в основному використовувався для розпізнавання мовлення та досягав хороших результатів [36]. Поступово його почали застосовувати до розпізнавання жестів, особливо розпізнавання мови жестів [37]. Популярність HMM у розпізнаванні жестів руки багато в чому пов’язана з тим, що кожен жест, який є просторово-часовим, можна зручно розглядати як послідовність поз, а HMM довела здатність ефективно

моделювати просторово-часову інформацію. Кожен жест моделюється різним HMM, а заданий невідомий жест перевіряється кожним HMM, і жест, що відповідає HMM з найкращим збігом, повертається як остаточний результат розпізнавання.

### 2.5 Застосування

Оскільки жести рук давно слугують важливим середовищем спілкування в людському суспільстві, вони також можуть відігравати не менш важливу роль в комп'ютеризованому світі.

#### 2.5.1 Альтернатива сенсорним пристроям

Планшети та смартфони перенесли наше повсякденне життя в еру управління за допомогою дотику, позбувшись традиційних пристроїв введення, таких як миша та клавіатура. Тим часом методи розпізнавання жестів рук, що базуються на комп’ютерному зорі, штовхають нас на крок далі, забезпечуючи безконтактне рішення (сенсорні датчики більше не потрібні). Багато великих IT-компаній, включаючи Microsoft, Samsung, Sony та Qualcomm, інтегрували методи розпізнавання жестів руками у свої продукти, такі як смарт-телевізори, відеоігрові консолі та мобільні процесори. Найближчим часом дуже ймовірно, що люди зможуть за допомогою пальців та руху рук зручно керувати різними пристроями [38, 39]. Наприклад, у проекті Kinect Magic Cursor клацання лівою

кнопкою миші можна змоделювати, піднявши ліву руку над плечем.

Рис. 15 Сенсор Kinect

### 2.5.2 Віртуальне 3D середовище

У віртуальному 3D-середовищі датчик глибини Kinect може бути використаний для створення 3D-анімації [40]. Користувачі можуть завантажувати 3D-моделі, такі як іграшки, в систему, які будуть виступати в ролі акторів. Потім система може використовувати 3D-моделі як маріонетки для анімаційного шоу (рис. 16). У недорогому інтерфейсному пристрої для взаємодії з об’єктами у віртуальному середовищі за допомогою жестів рук [41] основна архітектура складається з центрального обчислювального модуля, який може визначати положення пальців у конкретний момент, а потім переробляти пози рук. Завдяки ряду ключових патентів [42 - 44], GestureTek виріс як світовий лідер у галузі управління 3D-жестами руками. Наприклад, його система WallFX може створювати динамічні настінні дисплеї, відстежуючи та реагуючи на витончені рухи людського тіла, дозволяючи взаємодію в режимі реального часу між користувачем та дисплеєм та перетворюючи будь-яку вертикальну поверхню в інтерактивну систему відображення. Його GestureTrack3D Hand Tracker здатний слідувати двома руками, утворюючи режим мультитач, і може

відстежувати до десяти рук одночасно.

Рис. 16 Приклад 3D-анімації перед датчиком Kinect.

### 2.5.3 Хірургічна система

В хірургічному середовищі системи розпізнавання жестів можуть допомогти лікарям маніпулювати цифровими зображеннями під час медичних процедур, використовуючи жести замість сенсорних екранів або комп'ютерних клавіатур (рис. 17). Наприклад, система управління жестами рук дозволяє лікарям залишатися на місці під час операції без необхідності керувати традиційними комп’ютерними інтерфейсами [45, 46].

Рис. 17 Приклад розпізнавання жестів руки в хірургічній системі.

### 2.5.4 Мова жестів

Мова жестів використовує мову тіла для передачі значення замість звуків. Різні методи розпізнавання жестів на основі комп’ютерного зору були вбудовані в перекладачі мови жестів [47]. Зазвичай пристрій захоплення використовується для пошуку та відстеження рук та запису фігур та траєкторій рук, які представлені векторними елементами. Після узгодження з відповідними знаками вектори ознак порівнюються з бібліотекою граматики, щоб визначити,

чи мають ознаки сенс у контексті граматики. Значущий граматичний контекст вимагає розумних попередніх та наступних ознак. Для статичних жестів руками перекладачі можуть досягти точності понад 99% [48, 49]. Однак для динамічних жестів руками все ще важко досягти порівнянної продуктивності, і включення демографічної інформації користувача та контекстних баз даних може дещо підвищити продуктивність [50].

### 2.5.5 Управління роботами

Венецький та Тіман [51] запропонували систему розпізнавання жестів для керування робота, яка дозволяє користувачам керувати роботом за допомогою жестів рук. Система складається з робочого блоку, відео або інфрачервоної камери, прикріпленої до робочого блоку для зйомки зображень рук, блоку розпізнавання жестів та бази даних жестів. Також можна використовувати систему тренування роботів при вивченні нових жестів в Інтернеті чи інтерактивно [52].

## 2.6 Аналіз роботи системи розпізнавання жестів

### 2.6.1 Технології збору даних

Щоб зібрати інформацію застосовують технології, які поділяються на контактні та безконтактні.

Таблиця 1. Технології збору даних

| Контактні | Безконтактні |
|---|---|
| До контактних технологій використовуються раніше описані рукавички (провідні та кольорові рукавички) за допомого яких фіксуються рухи руки користувача. І на основі цих даних формується модель руки, з використання | До безконтактних технологій відносяться методи та бібліотеки комп’ютерного зору, які за допомогою камер зчитують на зображенні рух рук користувача, і обробляють його за допомогою різних алгоритмів. |

| | |
|---|---|
| спеціальних програм. | |

### 2.6.2 Обробка зображень

На цьому етапі алгоритми виділяють та відокремлюють пікселі шкіри, які зображують руку людини, та використовуються при визначенні жесту. Щоб алгоритм працював правильно йому треба відділити пікселі шкіри долоні від передпліччя. За допомогою середніх значень пікселів, які ми відстежуємо можна обчислити центроїд руки.

Обчислення положення пікселя проводиться за формулою:

$$L = \{\vec{x} | S\big(r(\vec{x}), g(\vec{x}), b(\vec{x})\big) = 1\}$$

Центроїд руки:

$$\overrightarrow{\mathrm{c_р}} = \frac{1}{|L|} \sum_{\vec{x} \in L} \vec{x}$$

Центроїд долоні:

$$\overrightarrow{\mathrm{c_д}} = \frac{1}{|L_д|} \sum_{\vec{x} \in L_д} \vec{x}$$

Де |L| – це кількість елементів L.

Край долоні можна знайти за допомогою сканування двох ліній, які паралельні, одна з яких поєднує два центроїда. Цей вектор визначається за формулою

$$\overrightarrow{c_{dif}} = \big(x_{dif}, y_{dif}\big) = \vec{c}_р - \vec{c}_д$$

Кут повороту руки:

$$\theta_р = \tan^{-1}\left(\frac{y_{dif}}{x_{dif}}\right)$$

А краї долоні обчислюються окремо для кожної точки $\vec{p}_1(s_1)$ вздовж лінії:

$$\vec{p}_1(s_1) = c + s \left( \frac{\cos\left(\theta_{\text{р}} + \frac{\pi}{2}\right)}{\sin\left(\theta_{\text{д}} + \frac{\pi}{2}\right)} \right)$$

Де (-50 ≤ $S_1$ ≤ 50)

Для кожної функції $S_1$ обчислюється кількість пікселів долоні $n(S_1)$ вздовж лінії

$$\vec{p}_2(s_1, s_2) = \vec{p}_1 + s_2 \left( \frac{cos(\theta_{\text{р}})}{sin(\theta_{\text{д}})} \right)$$

Де (-50 ≤ $S_2$ ≤ 50)

Дві точки, які визначають краї долоні $\vec{b}_{left}$ або $(x_{left}, y_{left})$ та $\vec{b}_{right}$ або $(x_{right}, y_{right})$, дорівнюють $\vec{p}_1(s_1)$, коли значення $n(S_1)$ нижче певного порогу. За допомогою порогу можливо виділити значення пікселів, які знаходяться в області долоні.

Радіус області долоні визначається як:

$$r_{\text{д}} = \max\left(|\vec{b}_{left} - \vec{c}_{\text{д}}|, |\vec{b}_{right} - \vec{c}_{\text{д}}|\right)$$

# РОЗДІЛ 3. ОПИС МЕТОДУ РОЗПІЗНАВАННЯ ЖЕСТІВ УКРАЇНСЬКОЇ ДАКТИЛЬНОЇ АБЕТКИ ЗА ДОПОМОГОЮ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

## 3.1 Існуючі підходи для розпізнавання жестів на базі згорткових нейронних мереж

Виявлення жестів може розглядатися як типове завдання виявлення об'єкта, який має набір загальноприйнятих та нових підходів як у класичному комп'ютерному зорі, так і в глибинному навчанні, зокрема з згортковими нейронними мережами. Нейронні мережі демонструють високу якість виявлення об'єктів, коли об'єкт має різноманітні спотворення, різний масштаб та різні умови освітлення, шум, розмитість і т.д.

Однією з ключових ідей при переході від статичного виявлення об’єкта до динамічного виявлення об’єкта є використання кількох поступових кадрів із відеовходу замість одного вхідного зображення для того, щоб використовувати додаткові часові дані серед просторових даних.

Жести дактильної абетки знаходяться з використанням різних підходів, заснованих на класичному комп'ютерному зорі з певними особливостями, такими як орієнтація гістограм [53], гістограми орієнтованих градієнтів (HOG) [54] або наборів функцій [55]. Хоча найсучасніші архітектури розпізнавання жестів базуються на CNN [56, 57, 58], подібно до інших завдань комп'ютерного зору.

Зазвичай для досягнення більш високої продуктивності з точки зору точності набору даних жестів архітектура CNN ускладнюється [59, 60].

### 3.2 Розпізнавання жестів дактильної абетки

Розпізнавання жестів, як частина крос-платформної технології для моделювання та розпізнавання української дактильної мови, має здійснюватися за допомогою крос-платформних інструментів. Підхід розпізнавання жестів залежить від типу вхідної інформації, з якою вони працюють. У разі використання алгоритмів на базі тривимірних моделей або скелетних алгоритмів підхід може використовувати об'ємну або скелетну модель, або їх комбінацію. Хоча ці підходи, як правило, важкі в обчисленні і вимагають додаткового обладнання від користувача. Інший тип підходів, моделі, засновані на зовнішньому вигляді, отримують параметри безпосередньо із зображення або послідовності зображень (у випадку, якщо відео використовується як вхідне зображення). Для навчання моделі розпізнавання, як правило використовуються певні техніки знаходження зразків або підходи машинного навчання. Через відсутність необхідності в додатковому обладнанні, крім простої веб-камери, цей тип підходів дуже часто обирають для крос-платформної технології. Деякі підходи, наприклад, Ong et al. [61] пропонує видобуток послідовних шаблонів (Sequential Pattern Mining) для виявлення ознак на основі деревоподібних структур.

Згорткові нейронні мережі (CNN) - це клас глибоких нейронних мереж, які є регуляризованими версіями багатошарових перцептронів, що найчастіше застосовуються для аналізу зображень та відео. CNN особливо якісно показують себе в аналізі зображень завдяки здатності враховувати посилання на місцевість даних на зображенні. Отже, CNN показує найсучасніші результати в задачах класифікації та розпізнавання зображень [62, 63]. Ще однією перевагою згорткових нейронних мереж є відсутність необхідності в рукотворних функціях, на відміну від традиційних алгоритмів узгодження шаблонів. Процес навчання бере вхідні дані, знаходить усі функції, необхідні для розпізнавання, і зберігає їх як вагові коефіцієнти моделі. CNN надійно виконують завдання

класифікації або розпізнавання об'єкта на зображенні, незалежно від масштабу вхідного зображення, умов освітлення, оклюзій, шуму тощо. Хоча для навчання такої моделі необхідний достатній набір даних. Зазвичай архітектура CNN складається з набору згорткових, об'єднаних і ReLU шарів. Структура Tensorflow забезпечує міжплатформену та ефективну реалізацію згорткових нейронних мереж.

### 3.3 Розпізнавання жестів за допомогою MobileNet

Архітектура MobileNetV3 (малюнок 18) - це нова мобільна архітектура, розроблена на основі моделі MobileNet. MobileNetV3 розширює свого попередника двома основними ідеями. Залишкові блоки з'єднують початок і кінець згорткового блоку за допомогою з'єднання пропуску. Додавши ці два стани, мережа має можливість отримати доступ до попередніх активацій, які не були змінені в згортковому блоці. Цей підхід виявився важливим для побудови мереж великої глибини. Перший крок розширює мережу, використовуючи згортку 1x1, оскільки наступна згортка по глибині 3x3 вже значно зменшує кількість параметрів. Потім ще одна згортка 1x1 стискає мережу, щоб відповідати початковій кількості каналів. Коефіцієнт 6 протиставлений 4 у наведеному прикладі. Кількість вхідних каналів с і, як часто блок повторюється n. Нарешті, s повідомляє, чи використовувався при першому повторенні блоку 2 етап для процесу зменшення дискретизації. Це звичайна збірка згорткових блоків.

Удосконалити цю мережу можна двома способами:

1. Видалення шару
2. Нерівномірність swish

В останньому блоці, рівень розширення 1x1, взятий з Інвертованого залишкового блоку (Inverted Residual Unit) MobileNetV2, який переміщується

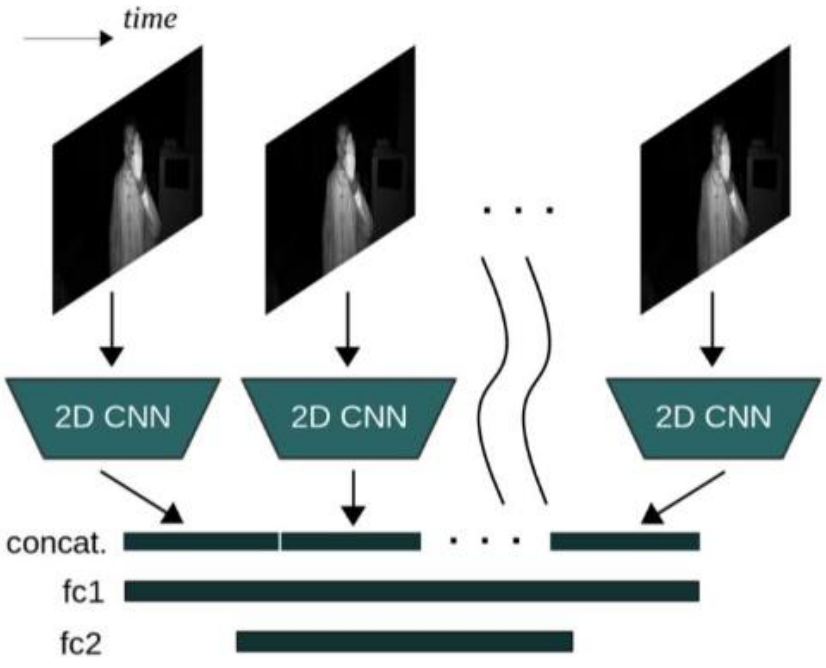
повз шар об'єднання. Це означає, що рівень 1x1 працює на картах розмірів 1x1

| Input | Operator | exp size | #out | SE | NL | s |
|---|---|---|---|---|---|---|
| $224^2 \times 3$ | conv2d, 3x3 | - | 16 | - | HS | 2 |
| $112^2 \times 16$ | bneck, 3x3 | 16 | 16 | ✓ | RE | 2 |
| $56^2 \times 16$ | bneck, 3x3 | 72 | 24 | - | RE | 2 |
| $28^2 \times 24$ | bneck, 3x3 | 88 | 24 | - | RE | 1 |
| $28^2 \times 24$ | bneck, 5x5 | 96 | 40 | ✓ | HS | 2 |
| $14^2 \times 40$ | bneck, 5x5 | 240 | 40 | ✓ | HS | 1 |
| $14^2 \times 40$ | bneck, 5x5 | 240 | 40 | ✓ | HS | 1 |
| $14^2 \times 40$ | bneck, 5x5 | 120 | 48 | ✓ | HS | 1 |
| $14^2 \times 48$ | bneck, 5x5 | 144 | 48 | ✓ | HS | 1 |
| $14^2 \times 48$ | bneck, 5x5 | 288 | 96 | ✓ | HS | 2 |
| $7^2 \times 96$ | bneck, 5x5 | 576 | 96 | ✓ | HS | 1 |
| $7^2 \times 96$ | bneck, 5x5 | 576 | 96 | ✓ | HS | 1 |
| $7^2 \times 96$ | conv2d, 1x1 | - | 576 | ✓ | HS | 1 |
| $7^2 \times 576$ | pool, 7x7 | - | - | - | - | 1 |
| $1^2 \times 576$ | conv2d 1x1, NBN | - | 1024 | - | HS | 1 |
| $1^2 \times 1024$ | conv2d 1x1, NBN | - | k | - | - | 1 |

замість 7x7, що робить його ефективним з точки зору обчислень та затримки.

Рис. 18 Архітектура MobileNetV3

Шар розширення вимагає багато обчислень. Але тепер, коли він переміщений за шаром об’єднання, нам не потрібно робити стиснення, зроблене проекційним шаром з попереднього блоку. Таким чином, ми можемо видалити цей проекційний шар та шар фільтруючого кільця з попереднього вузького



блоку.

Рис. 19 Конвеєр 3D-згорток в MobileNetv3.

### 3.4 Колекція української дактильної абетки для розпізнавання за допомогою MobileNet

Оскільки навчання згорткової нейромережі навряд чи залежить від великого та різноманітного набору даних, для досягнення достатньо високого рівня метрики було зібрано набір букв української дактильної абетки з різними характеристиками. Кожен жест складається з 1500 зразків зображень, і 50 різних рук людей, які демонстрували жести, з таким розподілом: 70% чоловічих та 30% жіночих рук. Були використані різні умови освітлення (з розподілом 20% зображень у поганих умовах освітлення, 30% у посередніх умовах освітлення та 50% у хороших умовах освітлення). Близько 10% зображень були спотворені шумом та розмиттям. Загалом у якості навчального набору даних було зібрано ~ 50 000 оригінальних зображень. Після застосування додаткових методів збільшення набору даних (таких як обертання, обрізання, дзеркальне відображення тощо), кінцевий набір даних отримав близько 150 000 зображень. Для тестування цілей було обрано частку 10% набору даних, що зробило остаточний навчальний набір даних 135 000 зображень, а остаточний набір даних - 15 000 зображень.

Для навчального процесу згорткової нейромережі на основі архітектури MobileNet, щоб виконати завдання розпізнавання жестів жестів української дактильної абетки слід було зібрати відповідний набір даних, оскільки доступні набори даних української мови жестів у вільному доступі - відсутні. Було розроблено спеціальне програмне забезпечення для запису коротких відеопослідовностей жестів українського дактильного алфавіту, показаних різними людьми.

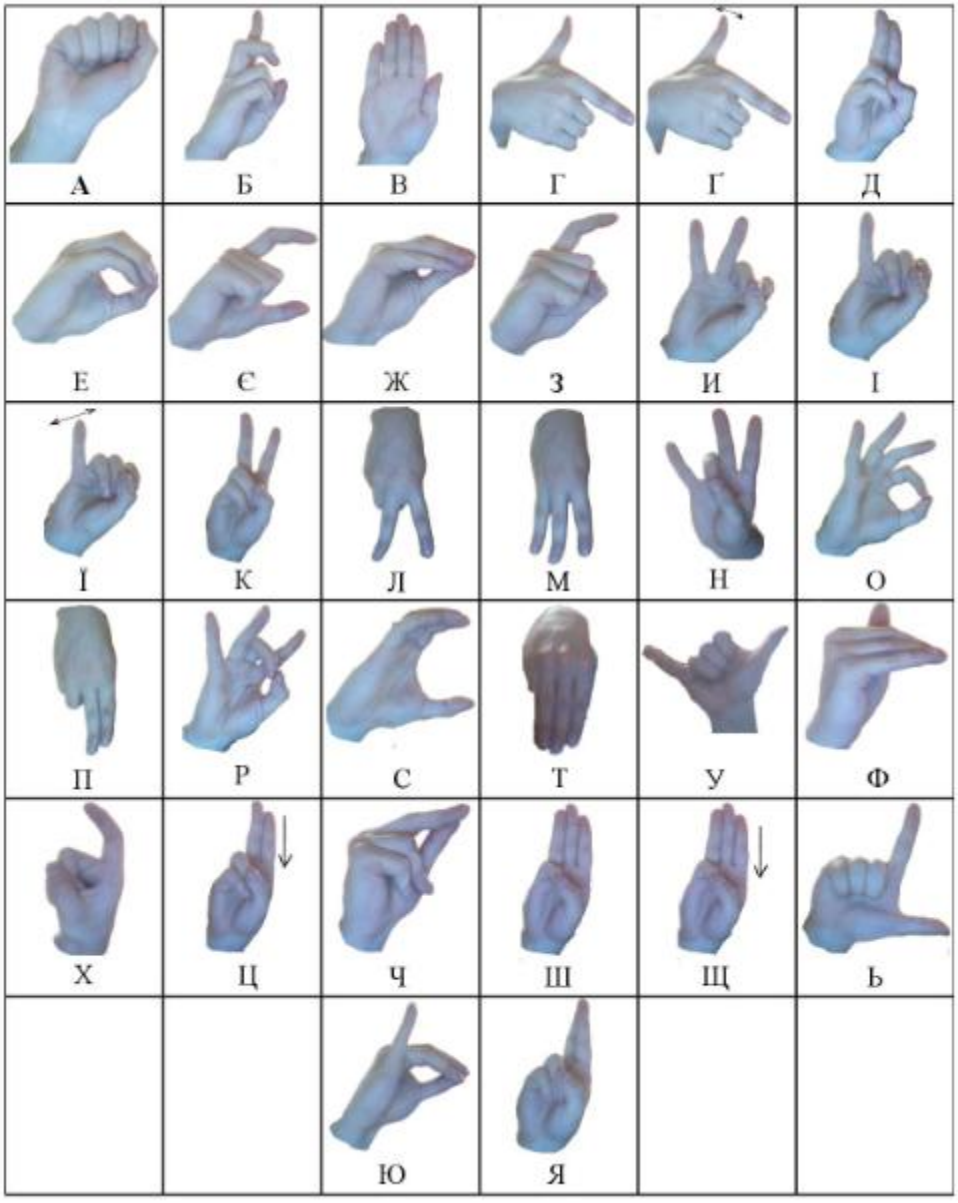



Рис 20. Набір жестів української дактильної абетки

### 3.5 MobileNetV3 із 3D-згортками

На малюнку 19 показано конвеєр із підходом просторово-часового моделювання, що використовується для 2D моделей CNN. Характеристики кожного 8 кадру витягуються за допомогою одного і того ж 2D CNN та об'єднуються, зберігаючи їх незмінними. Потім два рівні повністю зв’язаних (fc) шарів застосовуються для того, щоб отримати умовні оцінки класу. Причина полягає в тому, що шари fc можуть робити висновки про часові відносини, навіть не знаючи, що це послідовність взагалі. Розмір витягнутих особливостей

2D CNN складає 64 одиниці для кожного кадру. З першим шаром fc розмірність функції зменшена з 64 × 8 = 512 до 256. З другим шаром fc розмірність

| Layer / Stride | Repeat | Output size |
|---|---|---|
| Input clip | | $c\times8\times112\times112$ |
| Conv1($3\times3\times3$)/s(1,2,2) | 1 | $32\times8\times56\times56$ |
| Block/s(1,1,1) | 1 | $16\times8\times56\times56$ |
| Block/s(1,2,2) | 2 | $24\times8\times28\times28$ |
| Block/s(2,2,2) | 3 | $32\times4\times14\times14$ |
| Block/s(2,2,2) | 4 | $64\times2\times7\times7$ |
| Block/s(1,1,1) | 3 | $96\times2\times7\times7$ |
| Block/s(2,2,2) | 3 | $160\times1\times1\times1$ |
| Block/s(1,1,1) | 1 | $320\times1\times1\times1$ |
| Conv($1\times1\times1$)/s(1,1,1) | 1 | $1280\times1\times1\times1$ |
| Linear($1280\times NumCls$) | 1 | *NumCls* |

зменшена до кількості класів.

Рис 21. Архітектура MobileNetv3 з 3D-згортками

З іншого боку, 3D CNN в основному містить просторово-часове моделювання і не потребує додаткового механізму. Було збільшено SqueezeNet та MobileNetV3 таким чином, що вони приймають 8 кадрів як вхідні дані. Детальна інформація про 3D-MobileNetV3 наведена на малюнку 21.

### 3.6 Проведення експерименту з розпізнавання жестів

На кожному тренуванні застосовувались стандартні методи боротьби із переобладнанням нейронної мережі.

Під час навчального процесу згорткової нейронної мережі на основі архітектури MobileNet було створено кілька модифікацій архітектури, щоб знайти найкращий компроміс кількості шарів та точності. У певний момент точність натренованої моделі перестала зростати, тому отриману архітектуру

було визначено оптимальною з точки зору архітектури та точності, що показано на малюнку 22.

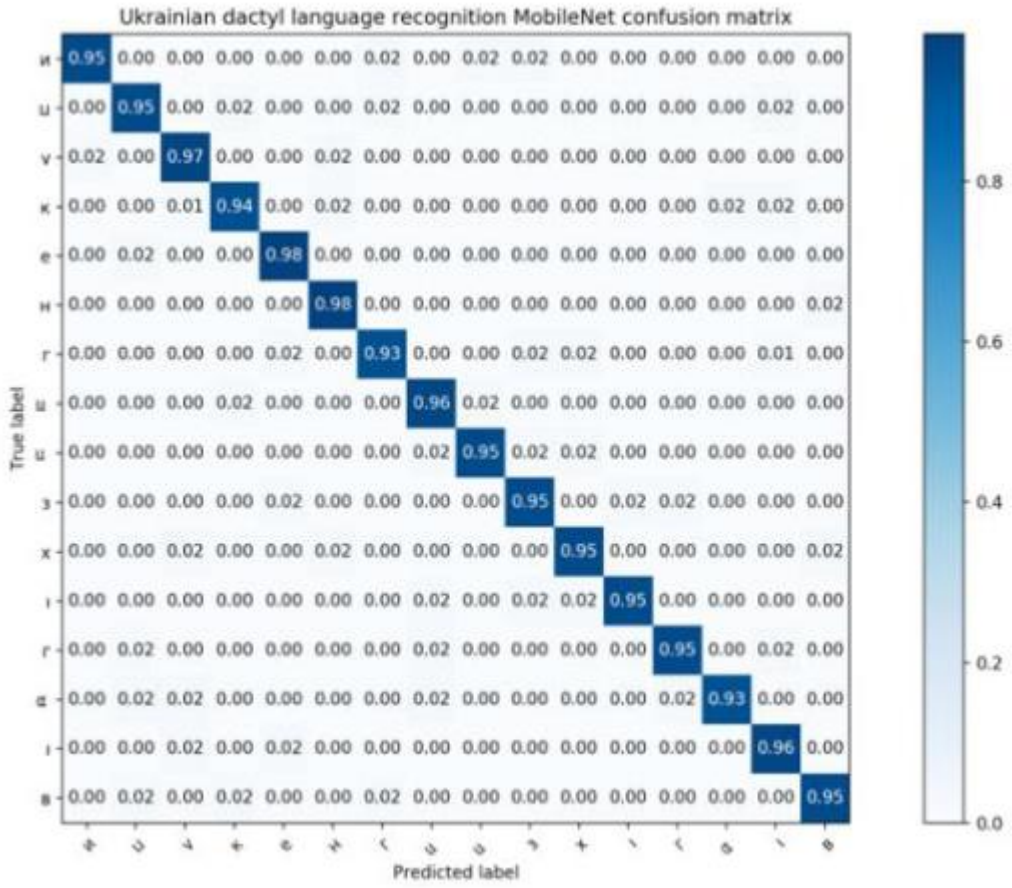



Рис 22. Матриця оптимальної архітектури моделі

# РОЗДІЛ 4. РЕАЛІЗАЦІЯ З ВИКОРИСТАННЯМ OPENCV ТА PYTHON

На основі описаних у цій роботі методі. Спробуємо розробити свою систему розпізнавання жестів з використанням мови програмування Python та бібліотеку комп'ютерного зору OpenCV.

Ми будемо використовувати такі версії програм Python 3.8 та OpenCV 4 – це оновленні версії, які підтримують роботу з розпізнаванням жестів рук.

Для початку на треба підключити потрібні нам бібліотеки комп'ютерного

```
# Imports
import numpy as np
import cv2
import math
```

зору та математики.

Рис 23. Підключення бібліотек

Щоб імпортувати бібліотеки cv2 та numpy для початку їх треба встановити. Це робиться за допомогою двох команд в терміналі: pip install numpy та pip install opencv-python.

Наступним етапом йде підключення камери, де в дужках можна вказати ір-адрес зовнішньої камери через яку, буде зчитуватися зображення, або як в нашому випадку можна вказати «0» і тоді буде зчитуватися зображення з

```
# Open Camera
capture = cv2.VideoCapture(0)
```

інтегрованої веб-камери.

Рис 24. Підключення камери

Після чого запускається цикл while. Його робота заключається в тому що спочатку він дивиться чи відкрита точка захоплення зображення, що означає поки процес захоплення зображення не розпочався процес зупиниться.

```
while capture.isOpened():

    # Capture frames from the camera
    ret, frame = capture.read()
```

Рис 25. Створення циклу while

Далі ми створюємо прямокутне підвікно де надаємо відому нам довжину (x, y) та метрику, після цього ми обрізаємо все що знаходиться не у виділеній зоні і зберігаємо їх в такі кадровані зображення.

```
# Get hand data from the rectangle sub window
cv2.rectangle(frame, (100, 100), (300, 300), (0, 255, 0), 0)
crop_image = frame[100:300, 100:300]
```

Рис 26. Створення підвікна

Після цього ми використовуємо один із методів фільтрації зображення, в нашому випадку це розмиття по Гауссу. Даний ефект використовується для зменшення зашумленості зображення та зниження деталізації.

```
# Apply Gaussian blur
blur = cv2.GaussianBlur(crop_image, (3, 3), 0)
```

Рис 27. Застосування розмиття Гауса

Далі ми конвертуємо отримане зображення з формату BGR у формат HSV,  який найбільше підходить для комп’ютерного зору.

```
# Change color-space from BGR -> HSV
hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)
```

Рис 28. Конвертація зображення

Створюємо маску, кольору шкіри людини, для цього треба ввести значення кольору вручну. Створюється бінарне зображення, де білий колір буде

```
# Create a binary image with where white will be skin colors and rest is black
mask2 = cv2.inRange(hsv, np.array([2, 0, 0]), np.array([20, 255, 255]))
```

кольором шкіри, а решта - чорним

Рис 29. Створення маски шкіри

Використовуємо ядро морфологічного перетворення, для того щоб переконатися що ми не втрачаємо потрібної нам інформації під час фільтрації зображення.

```
# Kernel for morphological transformation
kernel = np.ones((5, 5))
```

Рис 30. Використання ядра морфологічних перетворень

Застосовуйте морфологічні перетворення, щоб відфільтрувати фоновий

```
# Apply morphological transformations to filter out the background noise
dilation = cv2.dilate(mask2, kernel, iterations=1)
erosion = cv2.erode(dilation, kernel, iterations=1)
```

шум

Рис 31. Фільтрація фонового шуму

Знову використовуємо розмиття Гауса до вже розмитого зображення, щоб можна було обрахувати поріг. Поріг в свою чергу потрібен нам для того щоб створити сегментацію нашої руки

```
# Apply Gaussian Blur and Threshold
filtered = cv2.GaussianBlur(erosion, (3, 3), 0)
ret, thresh = cv2.threshold(filtered, 127, 255, 0)

# Show threshold image
cv2.imshow("Thresholded", thresh)
```

Рис 32. Розмиття Гауса для знаходження порогу

Після цього ми намагаємося знайти контури нашої руки. Контур – це крива, яка з’єднує всі неперервні точки, та має однакову інтенсивність кольору. Тому він намагається автоматичного його знайти за тим же самим кольором.

```
# Find contours
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

Рис 33. Знаходження контуру

Спробуємо знайти контур використавши функцію max до нього

```
try:
    # Find contour with maximum area
    contour = max(contours, key=lambda x: cv2.contourArea(x))
```

Рис 34. Знаходження контуру з функцією max

Створюємо обмежуючий прямокутник навколо контуру, щоб знайти

```
# Create bounding rectangle around the contour
x, y, w, h = cv2.boundingRect(contour)
cv2.rectangle(crop_image, (x, y), (x + w, y + h), (0, 0, 255), 0)

# Find convex hull
hull = cv2.convexHull(contour)
```

випуклий контур

Рис 35. Створення обмежуючого прямокутника

Додаємо частину, яка буде малювати контури на вихідному зображенні

```
# Draw contour
drawing = np.zeros(crop_image.shape, np.uint8)
cv2.drawContours(drawing, [contour], -1, (0, 255, 0), 0)
cv2.drawContours(drawing, [hull], -1, (0, 0, 255), 0)
```

Рис 36. Графічне зображення контурів

Знайдемо дефекти опуклості та використаємо правило косинуса, щоб знайти кут дальньої точки від початкової та кінцевої точки, тобто опуклі точки (кінчики пальців) для всіх дефектів.

```
# Find convexity defects
hull = cv2.convexHull(contour, returnPoints=False)
defects = cv2.convexityDefects(contour, hull)

# Use cosine rule to find angle of the far point from the
# tips) for all defects
count_defects = 0
```

Рис 37. Знаходження дефектів опуклості

Тепер за рахунок функції знаходження дефектів опуклості спробуємо знайти кінчики пальців та додамо вивід інформації про те скільки пальців бачить комп’ютер

```
for i in range(defects.shape[0]):
    s, e, f, d = defects[i, 0]
    start = tuple(contour[s][0])
    end = tuple(contour[e][0])
    far = tuple(contour[f][0])

    a = math.sqrt((end[0] - start[0]) ** 2 + (end[1] - start[1]) ** 2)
    b = math.sqrt((far[0] - start[0]) ** 2 + (far[1] - start[1]) ** 2)
    c = math.sqrt((end[0] - far[0]) ** 2 + (end[1] - far[1]) ** 2)
    angle = (math.acos((b ** 2 + c ** 2 - a ** 2) / (2 * b * c)) * 180) / 3.14

    # if angle > 90 draw a circle at the far point
    if angle <= 90:
        count_defects += 1
        cv2.circle(crop_image, far, 1, [0, 0, 255], -1)

    cv2.line(crop_image, start, end, [0, 255, 0], 2)
```

Рис 38. Знаходження кінчиків пальців

```
# Print number of fingers
if count_defects == 0:
    cv2.putText(frame, "ONE", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2,(0,0,255),2)
elif count_defects == 1:
    cv2.putText(frame, "TWO", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2,(0,0,255), 2)
elif count_defects == 2:
    cv2.putText(frame, "THREE", (5, 50), cv2.FONT_HERSHEY_SIMPLEX, 2,(0,0,255), 2)
elif count_defects == 3:
    cv2.putText(frame, "FOUR", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2,(0,0,255), 2)
elif count_defects == 4:
    cv2.putText(frame, "FIVE", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2,(0,0,255), 2)
else:
    pass
```

Рис 39. Вивід інформації про видиму кількість пальців

Ось що ми отримаємо в результаті:

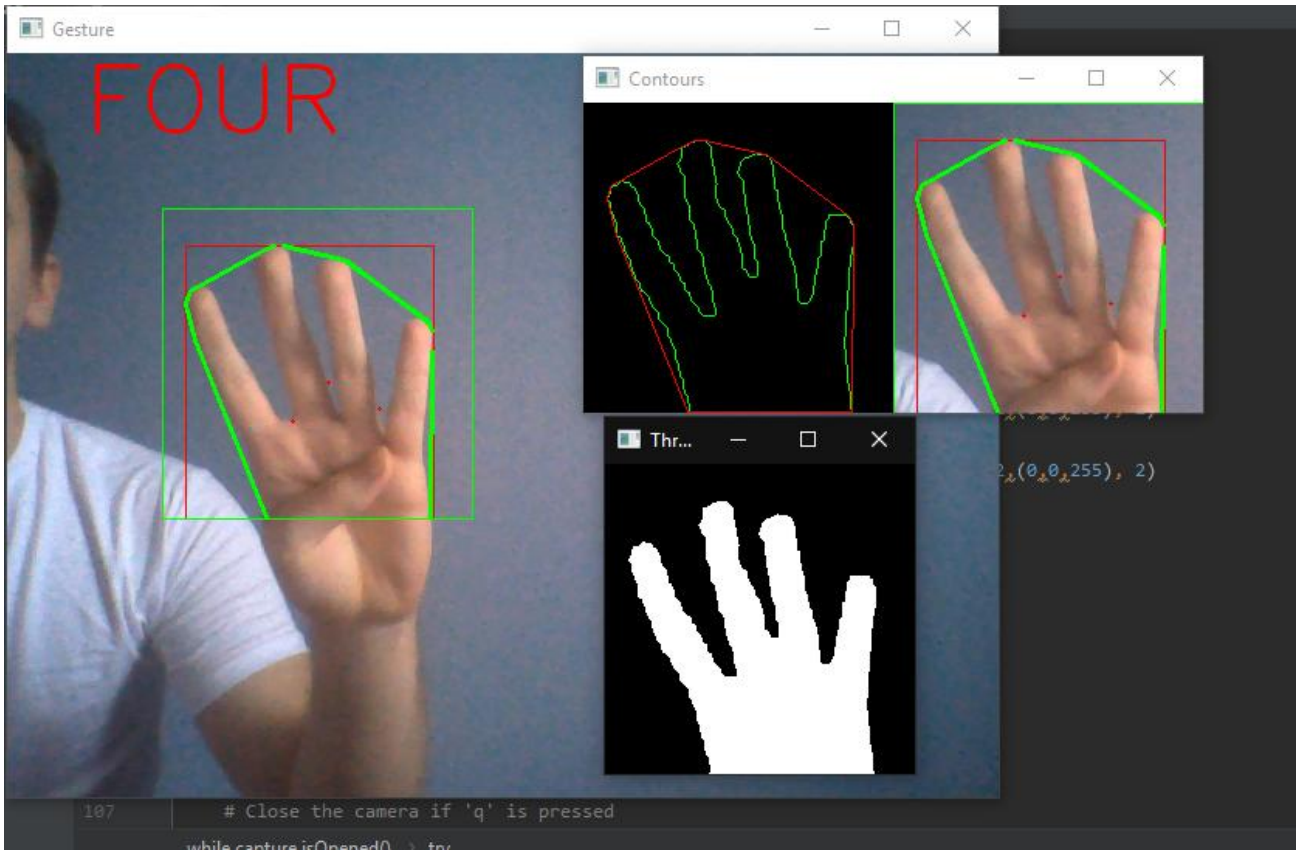



Рис 40. Отриманий результат

## ВИСНОВКИ

Динамічне розпізнавання жестів рук на основі комп’ютерного зору відіграє важливу роль у розвитку HCI. У цій роботі розглянуто деякі ключові алгоритми та застосування розпізнавання жестів рукою в реальному світі. Що стосується додатків, то розпізнавання жестів рук має значний вплив на наше повсякденне життя в багатьох аспектах. З бурхливим розвитком складних методів з’явився ряд успішних застосувань [64, 65]. Однак більшість цих програм дуже сильно залежать від різних факторів, таких як простий фон, нормальне освітлення та статична камера. Для розпізнавання жестів за загальних обставин слід вирішити низку питань, таких як складний фон,

часткова оклюзія, порушення фонового режиму, повторна поява об’єкта, зміна освітленості та робота в режимі реального часу. Більше того, рука є найбільш гнучкою частиною людського тіла і має високий коефіцієнт корисної дії. Це створює додаткові виклики, оскільки має різні види зовнішності і може швидко змінюватися. Сама швидкість руху рук може бути також дуже швидкою.

Щоб впоратися з цими проблемами, багато алгоритмів можна поєднати разом для роботи між собою, і в різних випадках досягаються гарні результати. Для складного фону злиття моделі кольору шкіри та моделі фігури може ефективніше виконувати завдання сегментації. Слід зауважити, що подібно до поєднання декількох алгоритмів, використання фонової інформації та інформації про об’єкт у попередніх кадрах може підвищити точність відстеження, але неминуче призведе до збільшення складності обчислень.

Тож на основі цих знань було розроблено приклад програми того що на даний момент нам можуть запропонувати відкриті бібліотеки в загальному доступі такій, як наприклад OpenCV, яку було використано у цій роботі. На сьогоднішній день можна сказати що результат доволі непоганий, але ця сфера ще потребує подальшого розвитку і має великий потенціал.

## ВИКОРИСТАНІ ДЖЕРЕЛА

1. Human-computer interface [Електронний ресурс] – Режим доступу до ресурсу: https://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/human-computer-interface

2. Hewett, Baecker, Card, Carey, Gasen, Mantei, Perlman, Strong and Verplank. CHAPTER 2.1 Definition of HCI from a Curricula for Human-Computer Interaction by ACM Special Interest Group on Computer-Human Interaction Curriculum Development Group

3. Жести рук [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%96%D0%B5%D1%81%D1%82

4. Ku L. Y., Liu L. W., Ye S. X., et al. (2020) Research on access control based on hybrid face and gesture recognition. Science and Technology Innovation Herald, 17:118-121+123.

5. Liu X. H., Deng B. S., Pei Y., et al. (2020) Research on wearable gesture interaction system and recognition algorithm. Small Microcomputer Systems, 41:2241-2248.

6. Duan C. X., Bai Y. (2020) Design of robot demonstration and teaching system combined with deep learning[J]. Computer measurement and control, 28(11):164-169.

7. Cheng R., Shi J. F. (2021) Research on hand gesture recognition algorithm based on convolutional neural network. Electronic design engineering, 29:179-184.

8. AForge.NET :: Framework [Електронний ресурс] – Режим доступу до ресурсу: http://www.aforgenet.com/framework/.

9. Використання бібліотеки AForge.NET і її додатку Accord.NET Framework при розпізнаванні обличь в реальному часі | Стаття у журналі «Молодий вчений» [Електронний ресурс] – Режим доступу до ресурсу: https://moluch.ru/archive/154/43602/

10. J. Weissmann and R. Salomon, “Gesture Recognition for Virtual Reality Applications Using Data Gloves and Neural Networks,” In Proceedings of the International Joint Conference on Neural Networks, vol.3, pp. 2043-2046, 1999.

11. Y. Wu and T. S. Huang, “Vision-Based Gesture Recognition: A Review,” In Proceedings of the International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction, pp. 103-115, 1999.

12. R. Y. Wang and J. Popović, “Real-Time Hand-Tracking with a Color Glove”, ACM Transactions on Graphics, vol. 28(3), Jul 2009.

13. V. I. Pavlovic, R. Sharma and T. S. Huang, “Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19(7), pp. 677-695, 1997

14. A. R. J. Francois and G. G. Medioni, “Adaptive Color Background Modeling for Real-Time Segmentation of Video Streams,” In Proceedings of the International Conference on Imaging Science, Systems, and Technology, vol. 1(121), pp. 227-232, 1999

15. D. Chai and K. N. Ngan. “Locating Facial Region of a Head-and-Shoulders Color Image,” In Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition, pp. 124-129, 1998.

16. A. A. Argyros and M. I. A. Lourakis. “Real-Time Tracking of Multiple Skin-Colored Objects with a Possibly Moving Camera,” In Proceedings of the 8th European Conference on Computer Vision, Lecture Notes in Computer Science, vol. 3023, pp. 368-379, 2004.

17. H. S. Yoon, J. Soh, Y. J. Bae and H. S. Yang, “Hand Gesture Recognition Using Combined Features of Location, Angle and Velocity,” Pattern Recognition, vol. 34(7), pp. 1491-1501, 2001.

18. H. K. Lee and J. H. Kim, “An HMM-Based Threshold Model Approach for Gesture Recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21(10), pp. 961-973, 1999.

19. J. Q. Wang and Y. Yagi, “Integrating Color and Shape-Texture Features for Adaptive Real-Time Object Tracking”, IEEE Transactions on Image Processing, vol. 17(2), pp. 235-240, 2008.

20. K. Oka, Y. Sato and H. Koike, “Real-Time Fingertip Tracking and Gesture Recognition”, IEEE Computer Graphics and Applications, vol. 22(6), pp. 64-71, 2002.

21. A. A. Argyros and M. I. A. Lourakis. “Vision-Based Interpretation of Hand Gestures for Remote Control of a Computer Mouse”, In Proceedings of the International Conference on Computer Vision in Human-Computer Interaction, pp. 40-51, 2006.

22. J. L. Crowley, F. Berard and J. Coutaz. “Finger Tracking as an Input Device for Augmented Reality,” In Proceedings of the International Workshop on Gesture and Face Recognition, pp. 195-200, 1995.

23. J. M. Rehg and T. Kanade. “Model-Based Tracking of Self-Occluding Articulated Objects”, In Proceedings of the 5thInternational Conference on Computer Vision, pp. 612-617, 1995.

24. S. Mitra and T. Acharya, “Gesture Recognition: A Survey,” IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, vol. 37(3), pp. 311-324, 2007.

25. D. Fleet and Y. Weiss, “Optical Flow Estimation,” Handbook of Mathematical Models in Computer Vision, pp. 237-257, 2006.

26. H, Zhou, “An OPS Hand Tracking Algorithm Based on Optical Flow,” In Proceedings of the WRI World Congress on Software Engineering, vol. 1, pp. 190-194, May, 2009.

27. K. R. T. Aires, A. M. Santana and A. A. D. Medeiros, “Optical Flow Using Color Information: Preliminary Results”, In Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 1607-1611, 2008

28. G. R. Bradski, “Computer Vision Face Tracking For Use in a Perceptual User Interface,” In Proceedings of the IEEE Workshop on Applications of Computer Vision, pp. 214–219, 1998.

29. J. C. Peng, L. Z. Gu and J. B. Su, “The Hand Tracking for Humanoid Robot using Camshift Algorithm and Kalman Filter”, Journal of Shanghai Jiaotong University, 2006.

30. M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A Tutorial on Particle Filters for On-Line Non-Linear/Non-Gaussian Bayesian Tracking,” IEEE Transactions on Signal Processing, vol. 50(2), pp. 174-189, 2002.

31. Y. Rui and Y. Q. Chen, “Better Proposal Distributions: Object Tracking Using Unscented Particle Filter”, In Proceedings of the 2001 IEEE Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. II-786-793, 2001.

32. M. Isard and A. Blake, “Condensation–Conditional Density Propagation for Visual Tracking”, International Journal of Computer Vision, vol. 29(1), pp. 5–28, 1998.

33. C. F. Shan, T. N. Tan and Y. C. Wei, “Real-Time Hand Tracking using a Mean Shift Embedded Particle Filter”, Pattern Recognition, vol. 40(7), pp. 1958-1970, 2007.

34. Z. Kalal, K. Mikolajczyk and J. Matas, “Tracking-Learning-Detection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34(7), pp. 1409-1422, 2012.

35. Hidden Markov Model. [Електронний ресурс] – Режим доступу до ресурсу: http://en.wikipedia.org/ wiki/Hidden_Markov_model

36. L. R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,” In Proceedings of the IEEE, vol. 77(2), pp. 257-286, 1989.

37. T. Starner and A. Pentland, “Real-time American Sign Language Recognition from Video using Hidden Markov Models,” In Proceedings of the International Symposium on Computer Vision, pp. 265-270, Nov 1995.

38. D. Minnen, “Fast Fingertip Detection for Initializing a Vision-Based Hand Tracker,” U.S. Patent, Application NO. 13/430,509, Oct. 25, 2012.

39. P. T. Keaton, S. Dominguez and A. H. Sayed, “Vision-Based Pointer Tracking and Object Classification Method and Apparatus,” U.S. Patent 7148913, Dec. 12, 2006.

40. R. T. Held, A. Gupta, B. Curless and M. Agrawala, “3D Puppetry: A Kinect-Based Interface for 3D Animation,” In Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology, pp. 423-434, 2012.

41. S. S. Rautaray and A. Agrawal, “Real Time Hand Gesture Recognition System For Dynamic Applications,” International Journal of UbiComp, vol. 3(1), pp. 21-31, Jan 2012.

42. E. Hildreth and F. MacDougall, “Video Image Based Control System,” U.S. Patent 5534917, Jul. 9, 1996.

43. E. Hildreth, “Optical Flow Based Tilt Sensor,” U.S. Patent 7379566, May. 27, 2008.

44. A. Shamaie, “Tracking Bimanual Movements,” U.S. Patent 7379563, May. 27, 2008.

45. B. Itkowitz, S. P. DiMaio and T. Zhao, “Method and Apparatus for Hand Gesture Control in a Minimally Invasive Surgical System,” U.S. Patent Application NO. 12/886,993, Sep. 21, 2010.

46. N. Frielinghaus, C. Hamilton and W. Steinle, “Gesture Support for Controlling and/or Operating a Medical Device,” U.S. Patent Publication NO. 2524279 A1, Nov. 21, 2012.

47. H. Birk and T. B. Moeslund, “Recognizing Gestures From the Hand Alphabet Using Principal Component Analysis,” Master Thesis, Laboratory of Image Analysis, Aalborg University, Denmark, 1996

48. S. Meena, “A Study on Hand Gesture Recognition Technique,” Master Thesis, Department of Electronics and Communication Engineering, National Institute of Technology, Rourkela, Orissa 769 008, India,2011

49. R. Lockton, “Hand Gesture Recognition using Computer Vision,” [Електронний ресурс] – Режим доступу до ресурсу: http://research.microsoft.com/en-us/um/people/ awf/bmvc02/project.pdf

50. J. Tardif, “Machine Based Sign Language Interpreter,” U.S. Patent, Application NO. 12/794,455, Jun. 4, 2010.

51. L. Venetsky and J. W. Tieman, “Robotic Gesture Recognition System,” U.S. Patent 7606411, Oct. 20, 2009.

52. C. Lee and Y. S. Xu, “Online, Interactive Learning of Gestures for Human/Robot Interfaces,” In Proceedings of the IEEE International Conference on Robotics and Automation, vol. 4, pp. 2982-2987, 1996.

53. Freeman, W., Roth, M. (1995). Orientation histograms for hand gesture recognition. In International workshop on automatic faceand gesture recognition, vol.12, 296-301.

54. Prasuhn, L., Oyamada, Y., Mochizuki, Y., Ishikawa, H. (2014). A HOG-based had gesture recognition system on a mobile device. In 2014 IEEE International Conference on Image Processing (ICIP), 3973-3977, IEEE.

55. Dardas, N., Georganas, D. (2011). Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. IEEE Transactions on instrumentation and measurement, 60: 3592-3607.

56. Kopuklu, O., Kose, N., Rigoll, G. (2018). Motion fused frames: Data level fusion strategy for hand gesturerecognitionarXivpreprint arXiv:1804.07187

57. Molchanov,P., Gupta, S., Kim, K., Pulli. K. (2015). Multi-sensor system for driver's hand-gesture recognition. In Automatic Face and Gesture Recognition (FG), 11th IEEE Inter-national Conference and Workshops on, vol.1, 1–8. IEEE

58. Molchanov, P., Gupta, S., Kim, K.,Kautz, J. (2015). Hand gesture recognition with 3d convolutional neural networks. In The IEEE Conference on Computer Vision and Pattern Recognition(CVPR) Workshops

59. Hu, J., Shen, L., Sun,G. (2018) Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, 7132–7141.

60. He, K., Zhang, X., Ren, S., Sun., J. (2016) Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 770–778.

61. Ong, E. (2012). Sign language recognition using sequential pattern trees. In: Computer Vision and Pattern Recognition(CVPR), IEEE Conference on IEEE pp. 2200-2207.

62. American Sign language: Real-time American Sign Language Recognition with Convolutional Neural Networks(2015).Brandon Garcia Stanford University Stanford, CA.

63. Bobic, V. (2016). Hand gesture recognition using neural network based techniques, School of Electrical Engineering, University of Belgrade

64. J. P. Wachs, M. Kölsch, H. Stern and Y. Edan, "Vision-Based Hand-Gesture Applications," Communications of the ACM, vol. 54(2), pp. 60-71, 2011.

65. S. S. Rautaray and A. Agrawal, “Vision Based Hand Gesture Recognition for Human Computer Interaction: A Survey,” Artificial Intelligence Review, Published Online: 06 November, 2012.

66. Krak, Y.G., Krak, Y.V., Barchukova, B.A. (2011). Human hand motion parametrization for dactylemes modeling, Journal of Automation and Information Sciences, 43 (12), 1-11.

67. Krak,I., Kondratiuk, S.(2017). Cross-platform software for the development of sign communication system: Dactyl language modelling,Proceedings of the 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT, 1, 167-170.DOI: 10.1109/STC-CSIT.2017.8098760

68. Introduction: What is VXL? [Електронний ресурс] – Режим доступу до ресурсу: https://vxl.github.io/

69. Miro [Електронний ресурс] – Режим доступу до ресурсу:

https://miro.com/