Київський національний університет імені Тараса Шевченка

**Лабораторна робота №2**

**З навчального курсу «Розпізнавання жестів з використанням нейромереж»**

**проф. Крак Ю.В.**

Виконав:

студент 4 курсу

факультету кібернетики

спеціальність «Комп’ютерні науки»

групи ТТП-42

Ходаков Максим

**Київ 2025**

### Вступ

У цьому звіті розглядається розробка системи розпізнавання жестів, що базується на використанні бібліотек **OpenCV** та **MediaPipe**. Основною задачею системи є виявлення руки за допомогою вебкамери, розрахунок показника «відкритості руки» (ratio) та класифікація жесту для визначення літери «X».

Система застосовується для аналізу рухів руки в режимі реального часу. Вона дозволяє збирати дані, логувати параметри роботи (FPS, коефіцієнт відкритості, схожість) і формувати підсумковий звіт, що може бути використаний для подальшого дослідження або оптимізації алгоритму.

### Теоретичні аспекти розпізнавання жестів

### Основні принципи роботи

Розпізнавання жестів базується на аналізі положення та руху руки, яке визначається за допомогою ключових точок (ландмарків).

* **MediaPipe Hands** забезпечує автоматичне визначення 21 лендмарку руки, що дозволяє побудувати скелет руки та аналізувати її положення.
* **OpenCV** використовується для обробки зображень, відображення результатів та організації інтерфейсу користувача (вікна відео, trackbar для налаштувань).

### Обчислення коефіцієнта відкритості

Ідея полягає в тому, щоб виміряти відстані між зап’ястям (landmark 0) та кінчиками пальців (landmarks 4, 8, 12, 16, 20).

* Середня відстань між цими точками нормалізується відносно розмірів руки (визначених як максимальна ширина або висота bounding box, що охоплює всі лендмарки).
* Отриманий коефіцієнт (ratio) використовується для визначення того, наскільки рука "відкрита". Якщо значення менше за порогове значення, система класифікує жест як літеру «X».

### Класифікація жесту

Для класифікації використовується порівняння обчисленого коефіцієнта відкритості з заданим порогом (openness\_threshold).

* Якщо ratio нижче порогу, система вважає, що жест відповідає літері «X».
* Розраховується також показник «схожості», який відображає відсоткову близькість ratio до порогового значення. Цей показник допомагає оцінити, наскільки точно розпізнано жест.

### Детальний опис коду

Нижче наведено розбір основних блоків коду, який реалізує описану логіку.

### 1. Імпорт бібліотек та налаштування логування

import cv2

import mediapipe as mp

import numpy as np

import time

import csv

import logging

import pandas as pd

Цей розділ забезпечує доступ до основних бібліотек, що використовуються для обробки зображень (OpenCV), розпізнавання жестів (MediaPipe), математичних операцій (NumPy), роботи з часом, збереження даних (CSV, Pandas) та логування подій.

Логування налаштовується таким чином, що всі повідомлення записуються у файл gesture.log з позначенням часу та рівня повідомлення. Це дозволяє вести аналіз роботи програми після завершення її виконання.

### 2. Задавання початкових параметрів

openness\_threshold = 0.6

Це значення визначає початковий поріг, за яким система розпізнає, що жест відповідає літері «X». Воно може бути змінене за допомогою інтерфейсу користувача (trackbar).

### 3. Функція оновлення порогу (update\_threshold)

def update\_threshold(x):

global openness\_threshold

openness\_threshold = x / 1000.0

Ця функція пов’язана з trackbar, який дозволяє користувачу у реальному часі регулювати порогове значення відкритості руки. Значення з trackbar нормалізується до діапазону [0, 1].

### 4. Обчислення коефіцієнта відкритості руки (compute\_openness)

def compute\_openness(landmarks):

wrist = np.array([landmarks[0].x, landmarks[0].y])

fingertips\_indices = [4, 8, 12, 16, 20]

distances = []

xs = []

ys = []

for lm in landmarks:

xs.append(lm.x)

ys.append(lm.y)

for idx in fingertips\_indices:

tip = np.array([landmarks[idx].x, landmarks[idx].y])

distances.append(np.linalg.norm(tip - wrist))

avg\_distance = np.mean(distances)

min\_x, max\_x = min(xs), max(xs)

min\_y, max\_y = min(ys), max(ys)

hand\_size = max(max\_x - min\_x, max\_y - min\_y)

if hand\_size == 0:

return 0

return avg\_distance / hand\_size

Ця функція отримує список лендмарків руки. Спершу визначається координата зап’ястя (landmark 0). Потім для кінчиків пальців обчислюються відстані до зап’ястя. Для кожного лендмарку збираються координати, за допомогою яких формується bounding box, що дозволяє визначити розміри руки. Повертається нормалізоване відношення – коефіцієнт відкритості.

### 5. Функція класифікації жесту (classify\_gesture\_by\_openness)

def classify\_gesture\_by\_openness(landmarks, threshold):

ratio = compute\_openness(landmarks)

similarity = max(0, min(100, (1 - abs(ratio - threshold) / threshold) \* 100))

if ratio < threshold:

return "X", True, ratio, similarity

else:

return "Not X", False, ratio, similarity

Ця функція обчислює коефіцієнт відкритості руки та розраховує відсоток схожості між обчисленим значенням та порогом. Якщо коефіцієнт менше порогу, вважається, що жест відповідає літері «X». Повертаються класифікаційний ярлик, логічне значення (True/False), обчислений коефіцієнт та відсоток схожості.

### 6. Ініціалізація MediaPipe Hands та підготовка до роботи з камерою

mp\_hands = mp.solutions.hands

hands = mp\_hands.Hands(static\_image\_mode=False,

max\_num\_hands=1,

min\_detection\_confidence=0.7,

min\_tracking\_confidence=0.5)

mp\_draw = mp.solutions.drawing\_utils

cap = cv2.VideoCapture(0)

if not cap.isOpened():

print("Camera open error!")

exit()

frame\_width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

frame\_height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

* **MediaPipe Hands** ініціалізується для розпізнавання однієї руки з певними пороговими значеннями для детекції та трекінгу.
* За допомогою OpenCV відкривається вебкамера. Якщо камера недоступна, програма завершується.
* Отримуються розміри кадру, які потрібні для запису відео.

### 7. Налаштування вікна відео та trackbar

cv2.namedWindow("Video")

cv2.createTrackbar("Openness Thresh", "Video", int(openness\_threshold \* 1000), 1000, update\_threshold)

Створюється основне вікно для відображення відео та trackbar, що дозволяє регулювати поріг відкритості в режимі реального часу.

### 8. Налаштування параметрів запису відео та логування даних

record\_video = False

video\_writer = None

fourcc = cv2.VideoWriter\_fourcc(\*'XVID')

output\_video\_file = "gesture\_output.avi"

data\_log = []

landmarks\_log = []

start\_time = time.time()

prev\_frame\_time = time.time()

* Змінна **record\_video** контролює, чи записується відео.
* **video\_writer** – об’єкт для запису відео з використанням кодека XVID.
* **data\_log** та **landmarks\_log** – списки для зберігання даних про жести та координати лендмарків, що дозволяє побудувати підсумковий звіт.
* Фіксується початковий час роботи та час попереднього кадру для розрахунку FPS.

### 9. Основний цикл обробки кадрів

Основний цикл складається з наступних кроків:

1. **Зчитування кадру**
   * Кожен кадр з вебкамери зчитується за допомогою cap.read().
   * Якщо кадр не зчитано, цикл продовжується.
2. **Перетворення кадру**
   * Кадр перевертається горизонтально для зручності відображення.
   * Обчислюється поточний час і FPS як обернене значення часу між кадрами.
3. **Обробка зображення для розпізнавання руки**
   * Кадр конвертується з формату BGR (OpenCV) у RGB (MediaPipe).
   * MediaPipe аналізує зображення і повертає набір лендмарків, якщо рука виявлена.
4. **Обчислення параметрів розпізнавання**
   * Якщо рука знайдена, для першого набору лендмарків:
     + Малюються лендмарки на зображенні.
     + Зберігаються координати лендмарків у список для логування.
     + Викликається функція класифікації, що обчислює коефіцієнт відкритості, відсоток схожості та повертає ярлик жесту («X» або «Not X»).
5. **Відображення інформації на кадрі**
   * На кадрі за допомогою cv2.putText виводяться значення FPS, коефіцієнта відкритості та відсоток схожості.
   * Залежно від значення схожості (поріг 70%), виводиться повідомлення про визначення літери «X».
6. **Запис відео та обробка клавіш**
   * Якщо режим запису увімкнено, кадри записуються у відеофайл.
   * За допомогою cv2.waitKey(1) обробляються натискання клавіш:
     + **q** – вихід з програми.
     + **r** – старт/зупинка запису відео.
     + **s** – збереження даних у CSV файли для подальшого аналізу.
7. **Логування та збереження даних**
   * Параметри кожного кадру (час, FPS, значення ratio, схожість та класифікація) додаються до логів.
   * Дані записуються у списки, які використовуються для формування підсумкового звіту.

### 10. Завершення роботи програми

Після завершення циклу:

* Звільняються всі ресурси (камера, відеозаписувач).
* Всі відкриті вікна закриваються за допомогою cv2.destroyAllWindows().
* За допомогою бібліотеки **pandas** формується звіт, що містить середні значення FPS, коефіцієнту відкритості, відсоток схожості та статистику по розпізнаних жестах.
* Підсумковий звіт записується у текстовий файл gesture\_report.txt та виводиться у консоль.

### Висновки

1. **Реальний час та ефективність.**  
   Система розпізнавання жестів дозволяє аналізувати відеопотік у реальному часі із високим рівнем точності. Обчислення FPS та логування даних забезпечують можливість оптимізації алгоритму.
2. **Гнучкість налаштувань.**  
   Використання trackbar для регулювання порогу відкритості дозволяє адаптувати систему під різні умови освітлення, відстань до камери та індивідуальні особливості руки.
3. **Модульність коду.**  
   Код побудовано з розбиттям на функції, що полегшує подальше модифікування та розширення системи (наприклад, для розпізнавання інших жестів).
4. **Логування та звітність.**  
   Завдяки ретельному логуванню роботи програми можна проводити аналіз ефективності алгоритму, виявляти можливі недоліки та розробляти стратегії покращення розпізнавання.

GitHub посилання: <https://github.com/maksymkhodakov/CVLab2>

### Перелік використаних джерел

1. **OpenCV Documentation**  
   Документація з використання OpenCV для роботи із зображеннями та відео.  
   URL: https://docs.opencv.org/
2. **MediaPipe Hands**  
   Офіційна документація та приклади використання MediaPipe для розпізнавання рук.  
   URL: https://google.github.io/mediapipe/solutions/hands.html
3. **NumPy Documentation**  
   Документація для роботи з бібліотекою NumPy для числових операцій.  
   URL: https://numpy.org/doc/
4. **Python Logging Module**  
   Документація по модулю logging для налаштування логування у Python.  
   URL: <https://docs.python.org/3/library/logging.html>
5. **Pandas Documentation**  
   Офіційна документація для роботи з бібліотекою pandas для аналізу даних.  
   URL: https://pandas.pydata.org/pandas-docs/stable/