Київський національний університет імені Тараса Шевченка

**Лабораторна робота №2**

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

**«** **Розпізнавання жесту літери “Б”»**

Виконав:

студент 4 курсу

факультету кібернетики

спеціальність «Комп’ютерні науки»

групи ТТП-42

Чебан Богдан Володимирович

**Київ 2025**

**Вступ**

Я розпочинав цю лабораторну роботу з метою глибоко дослідити можливості сучасних технологій комп’ютерного зору для автоматичного розпізнавання жестів. За останні роки спостерігається значний прогрес у цій галузі завдяки впровадженню передових методів машинного навчання та обробки зображень, що дозволяє створювати системи, здатні розуміти та інтерпретувати людські жести в режимі реального часу.

Головною задачею мого дослідження було створення системи, яка здатна розпізнавати українську літеру «Б» за допомогою аналізу 3D-координат руки, отриманих за допомогою бібліотек MediaPipe та OpenCV. Мене цікавило, як комп’ютер може "бачити" і аналізувати тонкі деталі руху руки, визначаючи не тільки загальну конфігурацію, але й порівнюючи її з еталонними зображеннями дактильної абетки.

Одним із важливих аспектів дослідження було не тільки отримання правильної класифікації жесту, але й збір детальних даних про кожен кадр. Це включало аналіз частоти кадрів (FPS), розрахунок коефіцієнта відкритості руки, обчислення відсотка схожості до еталонного жесту, а також збереження координат окремих точок (landmarks) руки для подальшого аналізу. Мене цікавило, які умови (освітленість, шум, кут зйомки) впливають на якість розпізнавання та як можна оптимізувати алгоритм, щоб система була стійкою до змін зовнішніх умов.

Також я прагнув розробити систему, яка забезпечує не лише візуальний відгук у режимі реального часу, але й створює розширену звітність для подальшої оцінки ефективності. Для цього я впровадив механізми логування, збереження даних у CSV-файли та побудови підсумкового звіту із використанням Pandas. Цей підхід дозволяє мені аналізувати результати експериментів, виявляти слабкі місця в алгоритмі та розробляти стратегії для подальшого покращення системи.

**Завдання**

Розпізнавання жесту: Визначити, чи відповідає поточна конфігурація руки жесту літери «Б». Для цього я використовую набір 21 тривимірних координат (landmarks), які надає MediaPipe Hands.

Аналіз ефективності розпізнавання: Розрахувати такі показники, як стабільність (тобто послідовність отримуваних результатів протягом заданого часу) та схожість поточного жесту до еталонного вигляду літери «Б». Результати аналізу виводяться в режимі реального часу.

Звітність: Зібрати всі дані за кожен кадр – включаючи FPS, класифікацію, коефіцієнт відкритості (ratio) та схожість жесту – і зберегти їх у CSV-файлах. Окрім цього, я зберігаю координати кожного landmark для більш детального просторового аналізу.

Візуалізація: Побудувати графік стабільності розпізнавання з використанням Matplotlib і забезпечити можливість регулювання порогового значення класифікації за допомогою trackbar.

Підсумкова звітність: За завершення роботи сформувати підсумковий звіт із загальними показниками сесії, який містить інформацію про середній FPS, середній коефіцієнт відкритості, середню схожість та розподіл розпізнаних жестів.

**Основний зміст звіту**

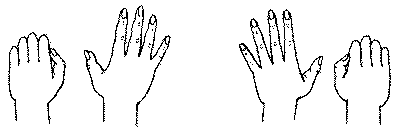
***1. Загальний опис характеристик жесту, що позначає літеру «Б»***

Жест, який позначає літеру «Б» у українській дактильній абетці, має унікальну конфігурацію руки, що відрізняється від звичайного розкриття. Я детально проаналізував цей жест і визначив кілька ключових аспектів, які допомагають його ідентифікувати:

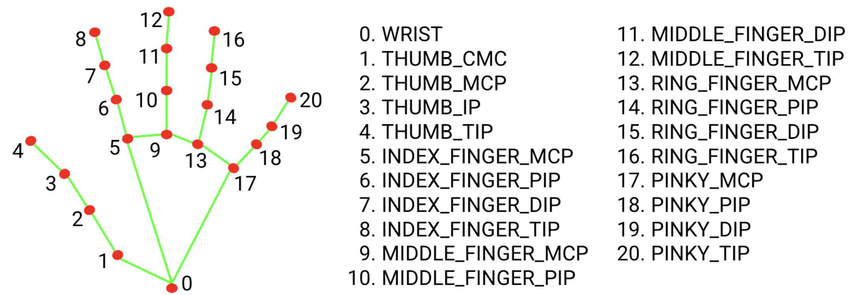
1. **Конфігурація пальців:**  
   У жесті «Б» пальці зазвичай не розгорнуті повністю, а знаходяться у стані часткового зігнуття. Це означає, що порівняно з відкритою рукою, кінчики пальців розташовані ближче до зап’ястя. Така конфігурація створює виразну лінію, що може нагадувати контур літери «Б».



1. **Стисненість кисті:**  
   У цьому жесті кисть руки, як правило, є дещо стисканою. Це забезпечує додатковий візуальний контур, який відрізняється від розкритої руки. Стиснена кисть допомагає зменшити загальний розмір bounding box руки, що є ключовим параметром для розрахунку коефіцієнта відкритості.  
   *Фотографія 2:* Варто розмістити порівняльне зображення: одна фотографія – відкрита кисть, інша – стиснена кисть із жестом «Б».



1. **Алгоритм класифікації:**  
   Я розробив алгоритм, який обчислює так званий коефіцієнт відкритості руки. Для цього я визначаю середню відстань від зап’ястя (landmark 0) до кінчиків пальців (наприклад, landmarks 4, 8, 12, 16, 20) і ділю її на загальний розмір руки, визначений як максимальна ширина або висота bounding box. Якщо отримане відношення менше за встановлений поріг, я класифікую жест як «Б».



1. **Обчислення схожості:**  
   Крім базової класифікації, я розраховую відсоток схожості поточного жесту до еталонного. Це дозволяє оцінити, наскільки точно конфігурація руки відповідає очікуваній формі літери «Б». Відхилення коефіцієнта відкритості від порогового значення використовується для обчислення відсоткового показника схожості, що дає змогу визначити точність класифікації.

***2. Характеристики зображення***

Я враховував кілька ключових параметрів зображення, оскільки вони безпосередньо впливають на якість розпізнавання жестів. Нижче наведено детальний опис кожного з параметрів:

**Освітленість:**

Якість зображення значною мірою залежить від рівня освітленості. Достатнє освітлення дозволяє камері чітко фіксувати контури руки та деталі її конфігурації. При гарному освітленні алгоритм може ефективно виділяти ключові точки (landmarks) руки, що критично для розрахунку коефіцієнта відкритості. Однак при недостатній освітленості контрасти знижуються, що призводить до нечіткого зображення і, як наслідок, зниження точності розпізнавання.



**Наявність шумів:**

Технічні особливості камери, низька якість сенсора або зовнішні умови (наприклад, електромагнітні перешкоди) можуть спричиняти появу шумів у зображенні. Ці шуми можуть викривляти контури руки і призводити до появи «хибних» точок при розпізнаванні. Це ускладнює процес виділення ключових елементів, таких як зап’ястя або кінчики пальців, що є необхідними для розрахунку коефіцієнта відкритості.



**Природне/штучне освітлення:**  
Тип освітлення також має значний вплив на якість зображення. Природне освітлення, зазвичай, забезпечує більш м’яке і рівномірне освітлення, що сприяє кращій передачі кольорів і контрастів. Штучне освітлення може створювати різкі тіні або нерівномірні освітлені зони, що іноді призводить до змін у колірній гаммі зображення. Для оптимального розпізнавання важливо використовувати стабільне, рівномірне джерело освітлення, яке не створює різких змін у яскравості протягом сесії зйомки.



**при природному освітленні при штучному освітленні**

***3. Умови забезпечення видимості елементів жесту***

Фокусування камери: Рука повинна знаходитися в центрі кадру та бути чітко зафокусованою. Від цього залежить, наскільки точно алгоритм зможе виділити окремі landmarks.

Стабільність зображення: Важливо, щоб камера не рухалась надто активно. Стабільне положення дозволяє уникнути розмиття та втрати деталей.

Контрастність: Елементи руки повинні чітко відрізнятися від фону, що забезпечує кращу видимість контурів пальців та кисті.

Розміщення руки: Рука має бути розташована так, щоб усі важливі частини (зокрема, зап’ястя та кінчики пальців) були видимими. Перекриття частин руки або їх часткова видимість може призвести до неправильного розпізнавання.

***4. Умови, за яких втрачається видимість елементів жесту***

Низька освітленість або перенасичене освітлення: При занадто слабкому або, навпаки, дуже яскравому освітленні камера може не правильно передавати контури руки.

Високий рівень шуму: Якщо зображення містить значні шуми або розмиття, деталі можуть бути спотворені.

Неправильний кут зйомки: Якщо рука розташована під кутом, коли частина її елементів (наприклад, пальці) не потрапляє у кадр, це ускладнює аналіз.

Перекриття елементів: Коли рука частково перекривається іншими об’єктами або однією частиною самої себе, це може призвести до втрати або неправильного визначення landmarks.

***5. Набір параметрів (координат тривимірного скелету руки)***

За допомогою MediaPipe Hands я можу отримати 21 landmark, кожен з яких містить три координати: x, y, z. Ці дані складають повноцінний тривимірний опис положення руки, що дає можливість виконувати низку важливих завдань:

Аналіз положення:  
  
Координати дозволяють визначити точне розташування ключових точок руки, таких як зап’ястя, кінчики пальців, центр кисті та інші критичні елементи. Завдяки цьому можна отримати детальний опис конфігурації руки в реальному просторі.

Обчислення відстаней:

Використовуючи координати окремих landmarks, можна розрахувати відстані між ними. Ці відстані є основою для обчислення різних показників, наприклад, коефіцієнта відкритості руки, який визначається як відношення середньої відстані від зап’ястя до кінчиків пальців до загального розміру руки (bounding box). Це дозволяє кількісно оцінити конфігурацію руки та правильно класифікувати жест.

Створення 3D-моделі:

Дані координат дозволяють побудувати віртуальну 3D-модель руки, що забезпечує можливість виконання додаткового просторового аналізу та візуалізації. За допомогою такої моделі можна перевіряти геометричні властивості руки, відтворювати її положення в 3D-просторі, а також інтегрувати результати в системи віртуальної чи доповненої реальності.

Порівняння з еталоном:

Отримані координати можна порівнювати з анатомічними описами, стандартними еталонами або даними з наукової літератури. Таке порівняння дозволяє перевірити точність розпізнавання жестів та адаптувати алгоритм до реальних умов, забезпечуючи кращу відповідність між віртуальними параметрами та реальним виглядом руки.

***6. Співвідношення віртуальних параметрів із реальними описами у видатковому матеріалі***

Координати landmarks:   
Вони відображають реальне положення окремих частин руки. Наприклад, положення зап’ястя або кінчиків пальців можна порівняти з медичними та анатомічними описами.

Коефіцієнт відкритості:   
Цей показник, розрахований як відношення середньої відстані від зап’ястя до кінчиків пальців до розміру руки, дозволяє кількісно оцінити, наскільки рука стискається або розкривається. Це співвідношення можна порівняти з фотографічними даними або еталонними зображеннями жестів.  
Схожість жесту:   
Обчислена як функція відхилення від встановленого порогу, вона дозволяє виміряти, наскільки поточна конфігурація руки наближена до очікуваної форми літери «Б». Цей показник допомагає перевести суб’єктивне сприйняття форми в кількісні дані.

***7. Алгоритм для покращення виділення запропонованих ознак***

Попередня обробка зображення:

На початковому етапі можна застосувати методи зменшення шуму (наприклад, гаусовий фільтр) та корекції освітлення (адаптивна порогова обробка), що допомагають покращити якість зображення.

Адаптивне налаштування порогів:

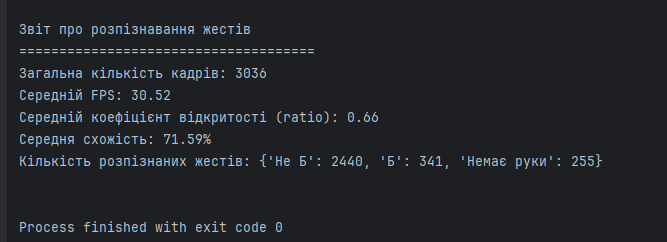
Використання trackbar дозволяє в режимі реального часу коригувати порогове значення коефіцієнта відкритості. Це дозволяє адаптувати алгоритм до змін умов освітлення або положення камери.

Використання машинного навчання:

Для більш точного розпізнавання можна навчити класифікатор (наприклад, нейронну мережу, SVM чи інший метод) на основі зібраних даних координат (landmarks) і числових характеристик. Це дозволить значно покращити точність, особливо у складних умовах.

Комбінований підхід: Поєднання попередньої обробки з використанням глибоких нейронних мереж для аналізу 3D-координат допоможе зробити систему більш стійкою до змін умов зйомки, забезпечуючи високу точність розпізнавання навіть за низької освітленості або при високому рівні шуму.

**Робота програми і код**



Код:

import cv2  
import mediapipe as mp  
import numpy as np  
import time  
import csv  
import logging  
import matplotlib.pyplot as plt  
import pandas as pd # для підсумкової звітності  
  
# Налаштування логування  
logging.basicConfig(filename="gesture.log", level=logging.INFO,  
 format="%(asctime)s - %(levelname)s - %(message)s")  
  
# Налаштування matplotlib для live plotting  
plt.ion()  
fig, ax = plt.subplots()  
stability\_times = []  
stability\_values = []  
(line\_plot,) = ax.plot(stability\_times, stability\_values, 'b-', label="Стабільність")  
ax.set\_xlabel("Час (с)")  
ax.set\_ylabel("Стабільність (%)")  
ax.set\_title("Реальний графік стабільності розпізнавання")  
ax.legend()  
  
# Глобальна змінна для порогового значення відкритості (наприклад, 0.6)  
openness\_threshold = 0.6  
  
  
def update\_threshold(x):  
 global openness\_threshold  
 # x змінюється від 0 до 1000; масштаб до [0, 1]  
 openness\_threshold = x / 1000.0  
  
  
# Функція для обчислення коефіцієнта відкритості руки  
def compute\_openness(landmarks):  
 *"""  
 Обчислює відношення середньої відстані від зап’ястя (landmark 0)  
 до кінчиків пальців (landmarks 4, 8, 12, 16, 20) до розміру руки.  
 Розмір руки визначається як максимальна ширина або висота bounding box.  
 """* 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)  
 # Захищаємося від ділення на нуль  
 if hand\_size == 0:  
 return 0  
 return avg\_distance / hand\_size  
  
  
# Функція класифікації на основі відкритості руки  
def classify\_gesture\_by\_openness(landmarks, threshold):  
 ratio = compute\_openness(landmarks)  
 # Обчислюємо "схожість" як відсоток, де 100% – коли ratio == threshold.  
 # Чим більше відхилення, тим менше схожість.  
 similarity = max(0, min(100, (1 - abs(ratio - threshold) / threshold) \* 100))  
 # Якщо коефіцієнт відкритості менший за поріг – вважаємо, що це "Б"  
 if ratio < threshold:  
 return "Б", True, ratio, similarity  
 else:  
 return "Не Б", False, ratio, similarity  
  
  
# Ініціалізація MediaPipe Hands  
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  
  
# Ініціалізація вебкамери  
cap = cv2.VideoCapture(0)  
if not cap.isOpened():  
 print("Помилка відкриття камери!")  
 exit()  
  
frame\_width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))  
frame\_height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))  
  
# Створення вікна для відео та trackbar для налаштування порогу відкритості  
cv2.namedWindow("Відео")  
cv2.createTrackbar("Openness Thresh", "Відео", int(openness\_threshold \* 1000), 1000, update\_threshold)  
  
# Налаштування для запису відео (за потреби)  
record\_video = False  
video\_writer = None  
fourcc = cv2.VideoWriter\_fourcc(\*'XVID')  
output\_video\_file = "gesture\_output.avi"  
  
# Лог для збереження даних (опційно)  
data\_log = [] # Запис: час, FPS, gesture, коефіцієнт відкритості, схожість  
  
# Для збереження координат кожного landmark  
landmarks\_log = [] # Запис: час, lm\_0\_x, lm\_0\_y, lm\_0\_z, ..., lm\_20\_x, lm\_20\_y, lm\_20\_z  
  
# Для аналізу стабільності за останню секунду  
recent\_frames = [] # Список кортежів (timestamp, gesture\_label)  
  
start\_time = time.time()  
prev\_frame\_time = time.time()  
last\_console\_update\_time = time.time()  
  
print("Натисніть 'q' для виходу, 'r' для старт/стоп запису відео, 's' для збереження даних у CSV.")  
  
while True:  
 ret, frame = cap.read()  
 if not ret:  
 continue # Якщо кадр не прочитано, переходимо до наступного  
 frame = cv2.flip(frame, 1)  
 current\_time = time.time()  
 fps = 1 / (current\_time - prev\_frame\_time)  
 prev\_frame\_time = current\_time  
  
 img\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)  
 results = hands.process(img\_rgb)  
  
 # За замовчуванням – "Немає руки"  
 gesture\_label = "Немає руки"  
 recognized = False  
 ratio = 0  
 similarity = 0  
  
 # Якщо рука виявлена, обробляємо лише першу руку  
 if results.multi\_hand\_landmarks:  
 for hand\_landmarks in results.multi\_hand\_landmarks:  
 mp\_draw.draw\_landmarks(frame, hand\_landmarks, mp\_hands.HAND\_CONNECTIONS)  
 # Запис координат кожного landmark  
 lm\_data = {"timestamp": current\_time - start\_time}  
 for idx, lm in enumerate(hand\_landmarks.landmark):  
 lm\_data[f"lm\_{idx}\_x"] = lm.x  
 lm\_data[f"lm\_{idx}\_y"] = lm.y  
 lm\_data[f"lm\_{idx}\_z"] = lm.z  
 landmarks\_log.append(lm\_data)  
  
 gesture\_label, recognized, ratio, similarity = classify\_gesture\_by\_openness(hand\_landmarks.landmark,  
 openness\_threshold)  
 cv2.putText(frame, f"Жест: {gesture\_label}", (10, 40),  
 cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)  
 break # Обробка лише першої руки  
  
 cv2.putText(frame, f"FPS: {int(fps)}", (10, 80),  
 cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, (255, 0, 0), 2)  
 cv2.putText(frame, f"Ratio: {ratio:.2f}", (10, 120),  
 cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, (0, 255, 255), 2)  
 cv2.putText(frame, f"Similarity: {similarity:.2f}%", (10, 160),  
 cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, (255, 255, 0), 2)  
  
 # Запис відео, якщо увімкнено  
 if record\_video:  
 if video\_writer is None:  
 video\_writer = cv2.VideoWriter(output\_video\_file, fourcc, fps, (frame\_width, frame\_height))  
 video\_writer.write(frame)  
  
 cv2.imshow("Відео", frame)  
 key = cv2.waitKey(1)  
 if key & 0xFF == ord('q'):  
 break  
 elif key & 0xFF == ord('r'):  
 record\_video = not record\_video  
 if not record\_video and video\_writer is not None:  
 video\_writer.release()  
 video\_writer = None  
 print("Запис відео:", "Увімкнено" if record\_video else "Вимкнено")  
 elif key & 0xFF == ord('s'):  
 # Запис даних з gesture та landmarks у CSV  
 with open('gesture\_data\_log.csv', 'w', newline='', encoding='utf-8') as csvfile:  
 fieldnames = ['timestamp', 'fps', 'gesture\_label', 'ratio', 'similarity']  
 writer = csv.DictWriter(csvfile, fieldnames=fieldnames)  
 writer.writeheader()  
 for row in data\_log:  
 writer.writerow(row)  
 with open('landmarks\_data\_log.csv', 'w', newline='', encoding='utf-8') as csvfile:  
 # Визначаємо усі ключі з першого запису (якщо він існує)  
 if landmarks\_log:  
 fieldnames = list(landmarks\_log[0].keys())  
 writer = csv.DictWriter(csvfile, fieldnames=fieldnames)  
 writer.writeheader()  
 for row in landmarks\_log:  
 writer.writerow(row)  
 print("Дані збережено у gesture\_data\_log.csv та landmarks\_data\_log.csv")  
 logging.info("Дані збережено у gesture\_data\_log.csv та landmarks\_data\_log.csv")  
  
 # Запис даних для аналізу  
 data\_log.append({  
 'timestamp': current\_time - start\_time,  
 'fps': fps,  
 'gesture\_label': gesture\_label,  
 'ratio': ratio,  
 'similarity': similarity  
 })  
  
 # Додаємо поточну класифікацію до recent\_frames  
 recent\_frames.append((current\_time, gesture\_label))  
 recent\_frames = [(t, g) for (t, g) in recent\_frames if current\_time - t <= 1]  
  
 # Оновлення консолі кожну секунду: обчислення стабільності  
 if current\_time - last\_console\_update\_time >= 1:  
 if recent\_frames:  
 total = len(recent\_frames)  
 count\_same = sum(1 for t, g in recent\_frames if g == gesture\_label)  
 stability = (count\_same / total) \* 100  
 else:  
 stability = 0  
 console\_msg = (f"Поточний жест: {gesture\_label} | Стабільність: {stability:.2f}% | "  
 f"Openness ratio: {ratio:.2f} | Схожість до 'Б': {similarity:.2f}% | Поріг: {openness\_threshold:.3f}")  
 print(console\_msg)  
 logging.info(console\_msg)  
  
 # Оновлення графіку стабільності  
 stability\_times.append(current\_time - start\_time)  
 stability\_values.append(stability)  
 line\_plot.set\_data(stability\_times, stability\_values)  
 ax.relim()  
 ax.autoscale\_view()  
 plt.draw()  
 plt.pause(0.001)  
  
 last\_console\_update\_time = current\_time  
  
cap.release()  
if video\_writer is not None:  
 video\_writer.release()  
cv2.destroyAllWindows()  
  
plt.ioff()  
plt.show()  
  
print("Програма завершена.")  
logging.info("Програма завершена.")  
  
# Додаткова звітність результатів за допомогою pandas  
df = pd.DataFrame(data\_log)  
if not df.empty:  
 avg\_fps = df['fps'].mean()  
 avg\_ratio = df['ratio'].mean()  
 avg\_similarity = df['similarity'].mean()  
 gesture\_counts = df['gesture\_label'].value\_counts().to\_dict()  
 total\_frames = len(df)  
  
 report\_text = (  
 f"Звіт про розпізнавання жестів\n"  
 f"=====================================\n"  
 f"Загальна кількість кадрів: {total\_frames}\n"  
 f"Середній FPS: {avg\_fps:.2f}\n"  
 f"Середній коефіцієнт відкритості (ratio): {avg\_ratio:.2f}\n"  
 f"Середня схожість: {avg\_similarity:.2f}%\n"  
 f"Кількість розпізнаних жестів: {gesture\_counts}\n"  
 )  
  
 with open('gesture\_report.txt', 'w', encoding='utf-8') as f:  
 f.write(report\_text)  
  
 print("\n" + report\_text)  
 logging.info("Підсумковий звіт збережено у gesture\_report.txt")  
else:  
 print("Немає даних для звітності.")

**Висновок**

Під час реалізації цього проекту я створив систему, яка здатна розпізнавати жест літери «Б» за допомогою MediaPipe та OpenCV. Завдяки використанню розрахунку коефіцієнта відкритості, я зміг якісно класифікувати конфігурацію руки, а також обчислити відсоткову схожість до еталонного жесту. Розширена звітність, яка включає логування подій, збереження даних у CSV-файлах, координат кожного landmark, а також формування підсумкового звіту з використанням Pandas, дозволяє мені детально аналізувати ефективність системи. Візуалізація у режимі реального часу через графік стабільності допомагає відслідковувати послідовність розпізнавання та визначати оптимальні параметри. В майбутньому інтеграція методів машинного навчання дозволить ще більше підвищити точність розпізнавання навіть у складних умовах.

**Список використаної літератури та ресурсів**

1. Документація MediaPipe:  
   <https://google.github.io/mediapipe/>  
   Я використовую MediaPipe для отримання координат тривимірного скелету руки, що є основою для розпізнавання жестів.
2. Документація OpenCV:  
   <https://docs.opencv.org/>  
   OpenCV забезпечує обробку зображень та роботу з відеопотоком, що є критично важливим для реального часу.
3. Книга "Programming Computer Vision with Python"  
   Автор: Jan Erik Solem  
   Ця книга надала мені базові знання щодо обробки зображень та застосування методів комп'ютерного зору.
4. Книга "Learning OpenCV 4"  
   Автори: Adrian Kaehler, Gary Bradski  
   У цій книзі розглядаються сучасні методи роботи з OpenCV, що допомогло мені розробити ефективну систему розпізнавання.
5. Офіційний сайт NumPy:  
   <https://numpy.org/>  
   NumPy використовується для обчислення математичних операцій над координатами та іншими даними.
6. Офіційний сайт Matplotlib:  
   <https://matplotlib.org/>  
   Matplotlib дозволяє мені візуалізувати результати в режимі реального часу та створювати графіки для подальшого аналізу.
7. Офіційний сайт Pandas:  
   <https://pandas.pydata.org/>  
   За допомогою Pandas я збираю, аналізую та формую підсумковий звіт про роботу системи розпізнавання.