ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
КАФЕДРА ЕОМ

Кваліфікаційна робота

Другий рівень (магістр)

Дослідження системи комп'ютерного зору для реалізації в реабілітаційній ортопедії

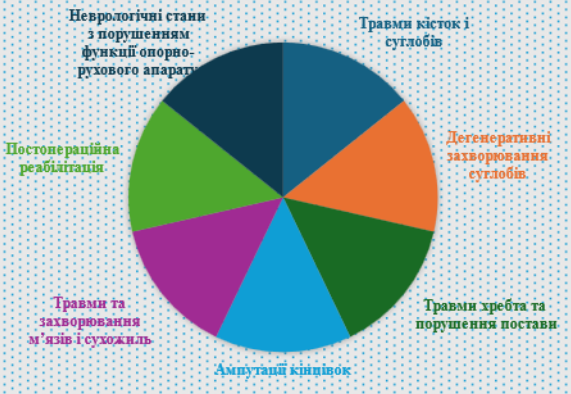
Автор
Коробко В.Ю.
ст. гр. КСМм-23-1

Керівник
Барковська О.Ю.
доц. каф. ЕОМ

ОГЛЯД ПРОБЛЕМНОЇ ОБЛАСТІ.

Стани, з якими працюють ортопеди-реабілітологи

	Переваги	Недоліки
Оптичні методи	Висока точність у маркерних системах, безконтактний збір даних у безмаркерних	Залежність від освітлення, обмеження простору
Інерційні методи (IMU)	Компактні, не залежать від освітлення, ефективні у великих просторах	Похибка накопичується з часом, особливо при тривалому відстеженні
Магнітні методи	Не залежать від видимості, підходять для закритих середовищ	Може виникати перешкоди від навколишнього середовища
Ультразвукові методи	Висока точність у малих просторах, відсутність потреби в камерах	Залежність від акустичних умов, обмеження у великих приміщеннях
Механічні методи	Точність відстеження рухів суглобів, підходять для медичних і промислових застосувань	Незручні для тривалого використання, обмежують природність рухів



ОГЛЯД ПРОБЛЕМНОЇ ОБЛАСТІ.

Застосування методів КЗ в напрямах медицини

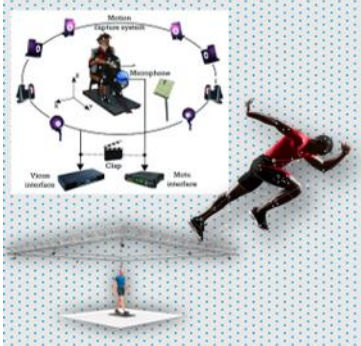
Напрям медицини	Задача	Отримуваний результат	Методи або моделі нейронних мереж	Максимальна точність
Дерматологія	Класифікація шкірних уражень	Діагностика раку шкіри, меланоми	ResNet, MobileNet, Inception	89-94%
Радіологія	Виявлення пухлин на рентгенограмах, КТ, МРТ	Рання діагностика раку, моніторинг прогресу лікування	CNN, ResNet, U-Net	До 95% точності
Офтальмологія	Аналіз зображень очного дна	Виявлення діабетичної ретинопатії	CNN, VGG, EfficientNet	93%
Кардіологія	Аналіз ехокардіограм, ЕКГ	Діагностика захворювань серця	LSTM, CNN	До 92%
Хірургічна навігація	Виявлення та сегментація органів	Підтримка хірургічних втручань	R-CNN, Mask R-CNN, Faster R-CNN	До 90%
Патологія	Аналіз гістопатологічних зразків	Виявлення аномалій тканин	DenseNet, EfficientNet, Transformer	88-96%
Реабілітація	**Моніторинг рухів пацієнтів**	**Відновлення функцій після травм**	**PoseNet, OpenPose**	**85-90%**
Онкологія	Виявлення метастазів	Рання діагностика та моніторинг лікування	VGG, AlexNet , ResNet	92-95%
Гастроентерологія	Аналіз зображень ендоскопії	Виявлення поліпів, виразок	YOLO, Faster R-CNN	90-95%
Стоматологія	Аналіз панорамних рентгенограм	Виявлення карієсу, аномалій зубів	U-Net, Faster R-CNN	До 88%

3

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ В ДАНІЙ ОБЛАСТІ

Маркерні методи:

- Vicon Motion Systems
- OptiTrack
- Qualisys



Безмаркерні методи:

- OpenPose
- MediaPipe
- Microsoft Kinect

Використання акселерометрів:

- Xsens MVN
- Noraxon
- Delsys Trigno

4

МЕТА ТА ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

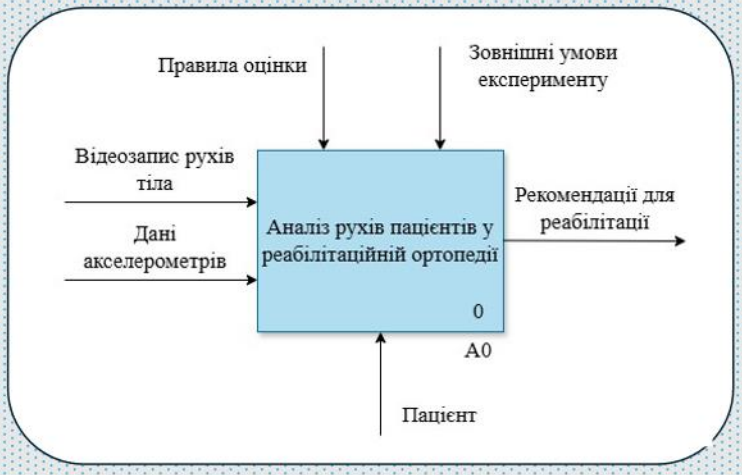
Метою кваліфікаційної роботи є дослідження ефективності застосування системи комп'ютерного зору у поєднанні із електронними датчиками у реабілітаційній ортопедії.

Для досягнення поставленої мети мають бути вирішені наступні задачі:

- провести аналіз сучасних технологій комп'ютерного зору та їхнього використання у реабілітаційній ортопедії;
- розробити прототип системи, яка зможе відслідковувати та аналізувати рухи пацієнта;
- оцінити ефективність застосування методів комп'ютерного зору при різних навколишніх умовах для визначення стану суглобів при ходьбі;
- оцінити ефективність застосування електронних датчиків для визначення стану суглобів при ходьбі;
- розробити метод комбінованого аналізу даних про стан суглобів пацієнтів.

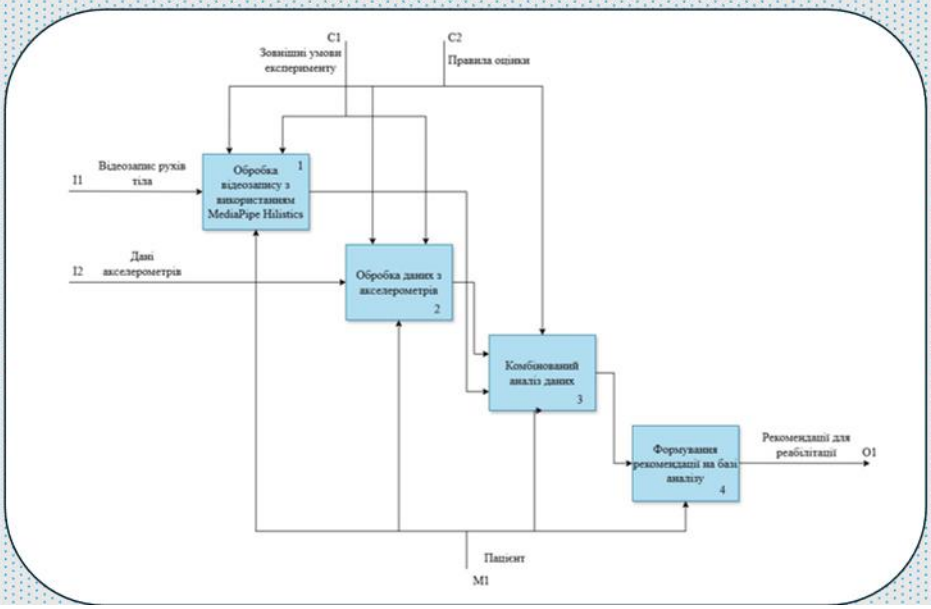
5

ЗАПРОПОНОВАНА УЗАГАЛЬНЕНА МОДЕЛЬ ВІДДАЛЕНОГО МОНІТОРИНГУ КОНТРОЛЮ ПОСТАВИ



6

ДЕКОМПОЗИЦІЯ ЗАПРОПОНОВАНОЇ СИСТЕМИ ВІДДАЛЕНОГО МОНІТОРИНГУ КОНТРОЛЮ ПОСТАВИ



7

ДОСЛІДЖУВАНІ ЗАЛЕЖНОСТІ ТА ВПЛИВИ

Вплив якості освітлення на точність детектування порушень постави:

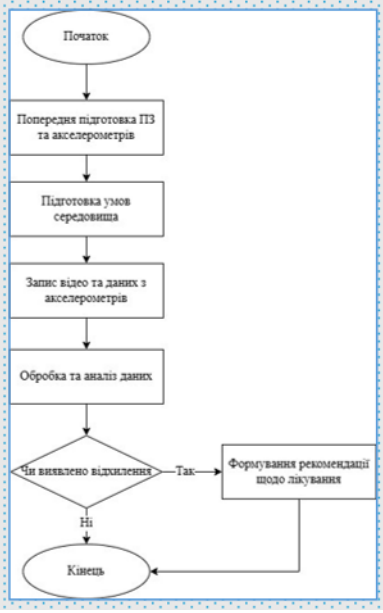
- гарним освітлення є від 300 люксів або 300 люменів на м^2;
- погане освітлення є до 100 люксів або 100 люменів на м^2.

Вплив складності фону приміщення:

- однотонний фон є ідеальною умовою для роботи з методами комп'ютерного зору;
- зашумлений фон або різнокольоровий фон з високим контрастом може створювати перешкоди при визначенні ключових точок тіла.

Вплив швидкості ходи, як змінної умови, яка може по-різному впливати на результат:

- повільна, до 50 кроків на хвилину;
- швидка, від 100 кроків на хвилину.



8

ДЕМОНСТРАЦІЯ ДОСЛІДЖУВАНИХ СТАНІВ

Кіфоз

Сколіоз

9

ОТРИМАНІ РЕЗУЛЬТАТИ ОБРОБКИ ДАНИХ З АКСЕЛЕРОМЕТРІВ

Time	Device name	Chip Time()	Acceleration X(g)	Acceleration Y(g)	Acceleration Z(g)	Angular velocity X(°/s)	Angular velocity Y(°/s)	Angular velocity Z(°/s)	Angle X(°)	Angle Y(°)	Angle Z(°)	Magnetic field X(ut)	Magnetic field Y(ut)	Magnetic field Z(ut)	Temperature(℃)	Quaternions 0()	Quaternions 1()	Quaternions 2()	Quaternions 3()
01:04:51.449	d4:b7:46:13:0c:06	2085-97-25 251:145:253:1615	-0.529	-0.285	0.742	-1.953	-33.081	-16.602	-20.187	37.837	75.256	20	-37.7	10.367	25.59	-0.69049	0.33234	-0.1738	-0.61841
01:04:51.503	e5:16:bf:b4:d4:5a	2085-97-244 01:254:07:570	0.221	0.742	0.46	19.836	32.471	29.663	55.206	-17.463	-146.398	-14.7	-18.4	66.287	26.38	0.311	0.00226	-0.47195	-0.82489
01:04:51.658	d4:b7:46:13:0c:06	2085-113-81 00:252:164:10429	-0.509	-0.359	0.812	-35.95	-29.846	-15.259	-24.406	30.322	75.657	20	-37.7	10.367	25.59	-0.71106	0.31827	-0.07465	-0.62241
01:04:51.684	e5:16:bf:b4:d4:5a	2085-113-81 00:138:50:1932	0.254	0.924	0.368	-9.644	-18.372	13.367	61.183	-17.688	-135.209	-14.7	-18.4	66.287	26.38	0.39484	0.05896	-0.51767	-0.75665
01:04:51.804	e5:16:bf:b4:d4:5a	2085-97-130 02:11:07:683	0.417	0.864	0.382	17.578	10.742	18.86	61.425	-19.616	-135.61	-17.1	-15.6	71.627	26.38	0.39484	0.05896	-0.51767	-0.75665

Формула обчислення кута нахилу тулуба:

$$K_{кіфоз} = acrtan\left(\frac{Angle\ Z}{Angle\ Y}\right)$$

Формула обчислення коефіцієнту асиметрії тулуба:

$$K_{сколіоз} = \frac{|L-R|}{\frac{L+R}{2}} \times 100\%$$

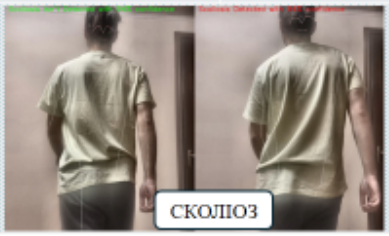
10

ОТРИМАНІ РЕЗУЛЬТАТИ БЕЗМАРКЕРНОЇ ОБРОБКИ

	Похибка
Контрольний експеримент	6%
Експеримент з відхиленням	6%

Графік залежності похибки визначення кіфозу від зовнішніх умов

	Похибка
Контрольний експеримент	6%
Експеримент з відхиленням	6%



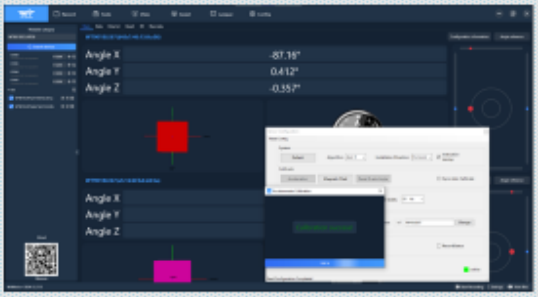
Графік залежності похибки визначення сколіозу від зовнішніх умов

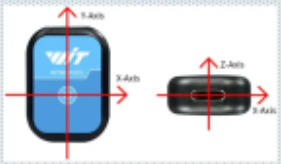
11

ОТРИМАНІ РЕЗУЛЬТАТИ ОБРОБКИ ДАНИХ З АКСЕЛЕРОМЕТРІВ

	Кут нахилу	
	до 50 кроків на хвилину	до 100 кроків на хвилину
Контрольний експеримент (нормальне положення корпусу)	20°	30°
Експеримент з відхиленням (при корпусі із нахилом)	41°	54°

	Коефіцієнт асиметрії	
	до 50 кроків на хвилину	до 100 кроків на хвилину
Контрольний експеримент (нормальне положення корпусу)	1.5%	3%
Експеримент з відхиленням (при корпусі із нахилом)	8.2%	13.5%





Калібрування акселерометрів WitMotion WT9011DCL-BT50

12

ОТРИМАНІ РЕЗУЛЬТАТИ КОМБІНОВАНОЇ ОБРОБКИ

	Похибка
Контрольний експеримент	5%
Експеримент з відхиленням	5%

	Похибка
Контрольний експеримент	5%
Експеримент з відхиленням	5%

13

АНАЛІЗ РЕЗУЛЬТАТІВ

Метод	Переваги	Недоліки	Рекомендоване застосування
Безмаркерний метод	Не потребує спеціального обладнання; Швидкість аналізу; Мінімальні витрати на впровадження.	Висока залежність від освітлення та фону.	Для початкового аналізу в умовах гарного освітлення.
Використання акселерометрів	Висока точність вимірювань; Незалежність від освітлення та фону.	Не працює в реальному часі; Складність у встановленні та калібруванні.	Для моніторингу рухів у складних умовах (погане освітлення, зашумлений фон).
Комбінований метод	Висока точність вимірювань; Висока стійкість до зовнішніх факторів. Комплексний аналіз рухів.	Не працює в реальному часі; Складність синхронізації даних.	Для точного аналізу складних випадків та створення індивідуальних програм.

14

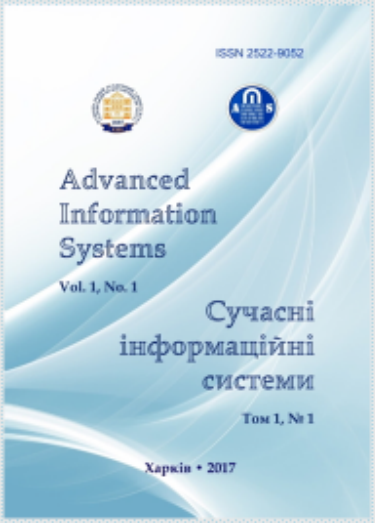
ВИСНОВКИ

- У межах кваліфікаційної роботи було проведено дослідження систем комп'ютерного зору для їхнього застосування в реабілітаційній ортопедії. Основна увага зосереджувалася на аналізі методів захоплення рухів антропометричних об'єктів, оцінці ефективності безмаркерних методів у порівнянні з використанням акселерометрів, а також їхньому комбінуванні для підвищення точності результатів.
- Аналіз безмаркерних методів та використання акселерометрів показав, що безмаркерні методи забезпечують швидкість обробки даних і підходять для широкого спектра застосувань, але їх точність залежить від зовнішніх факторів, та використання акселерометрів демонструє високу точність незалежно від умов, але вимагають правильного калібрування та більший час на обробку даних.
- Було виявлено, що покращення освітлення, використання простого однотонного фону та впровадження додаткових алгоритмів обробки (фільтрація шумів, покращення контрасту) знижують похибку безмаркерного методу до 6%. Інтеграція даних акселерометрів із відеозйомкою дозволила досягти комбінованої похибки на рівні 5.05%, що свідчить про переваги сумісного використання методів.

15

АПРОБАЦІЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

- Коробко В.Ю., Барковська О.Ю Дослідження системи комп'ютерного зору для реалізації в реабілітаційній ортопедії. // Проблеми інформатизації : XII міжнародна науково-технічна конференція. - 21-22 листопада 2024. –с.72. doi: https://doi.org/10.32620/PI.24.t2
- Olesia Barkovska, Dmytro Oliinyk, Oleksandr Ruskikh, Volodymyr Korobko, Peter Sedlacek (2024). Study of methods for detecting optical markers in the system of human gait and posture analysis. Advanced Information Systems, 9(4).



16

ДОДАТОК Б

Код програми для виявлення кіфозу

```
import cv2
import mediapipe as mp
import math
import time
import os

# Initialize MediaPipe Pose
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5)
mp_draw = mp.solutions.drawing_utils

def calculate_angle(a, b, c):
    """Calculate angle between three points (a, b, c)."""
    a_x, a_y, a_z = (a["x"], a["y"], a["z"]) if isinstance(a,
dict) else (a.x, a.y, a.z)
    b_x, b_y, b_z = (b["x"], b["y"], b["z"]) if isinstance(b,
dict) else (b.x, b.y, b.z)
    c_x, c_y, c_z = (c["x"], c["y"], c["z"]) if isinstance(c,
dict) else (c.x, c.y, c.z)

    radians = math.atan2(c_y - b_y, c_x - b_x) - math.atan2(a_y
- b_y, a_x - b_x)
    angle = abs(math.degrees(radians))
    return angle if angle <= 180 else 360 - angle

def calculate_midpoint(point1, point2):
    """Calculate the midpoint between two landmarks."""
    return {
        "x": (point1.x + point2.x) / 2,
        "y": (point1.y + point2.y) / 2,
        "z": (point1.z + point2.z) / 2,
        "visibility": (point1.visibility + point2.visibility) /
2,
    }

def check_side_posture(landmarks):
    """Improved detection for kyphosis with additional landmarks
and normalization."""
    nose = landmarks[mp_pose.PoseLandmark.NOSE.value]
    left_shoulder =
landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value]
    right_shoulder =
landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value]
    left_hip = landmarks[mp_pose.PoseLandmark.LEFT_HIP.value]
    right_hip = landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value]
```

```
    left_knee = landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value]
    right_knee =
landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value]

    # Midpoints for shoulders and hips
    mid_shoulder = calculate_midpoint(left_shoulder,
right_shoulder)
    mid_hip = calculate_midpoint(left_hip, right_hip)
    mid_knee = calculate_midpoint(left_knee, right_knee)

    # Calculate kyphosis angle using spine deviation
    kyphosis_angle = calculate_angle(mid_hip, mid_shoulder,
nose)

    # Calculate normalized vertical alignment from hip to knee
    vertical_alignment = calculate_angle(mid_knee, mid_hip,
mid_shoulder)

    # Detect kyphosis based on multiple criteria
    kyphosis_detected = not (kyphosis_angle >= 151 and
vertical_alignment >= 165)  # Adjust thresholds as needed

    return kyphosis_detected, kyphosis_angle, vertical_alignment

# Open the video file
video_path = "sample_new.mp4"  # Replace with your video path
cap = cv2.VideoCapture(video_path)

if not cap.isOpened():
    print("Error: Unable to open video file.")
    exit()

# Video writer setup
output_path = os.path.join(os.getcwd(),
"output_video_clahe.avi")  # Save in the same folder as the
script
fps = int(cap.get(cv2.CAP_PROP_FPS))
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter(output_path, fourcc, fps, (frame_width,
frame_height))

# Initialize counters
kyphosis_frames = 0
detection_frames = 0
body_present = False

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("End of video reached.")
        break
```

```
    # Start timer for frame processing
    start_time = time.time()

    # Apply CLAHE
    lab = cv2.cvtColor(frame, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
    cl = clahe.apply(l)
    lab = cv2.merge((cl, a, b))
    frame = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)

    # Apply Gaussian Blur
    frame = cv2.GaussianBlur(frame, (5, 5), 0)

    # Convert the frame to RGB
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    result = pose.process(rgb_frame)

    if result.pose_landmarks:
        # Body is detected
        body_present = True
        detection_frames += 1

        # Draw landmarks
        mp_draw.draw_landmarks(frame, result.pose_landmarks,
mp_pose.POSE_CONNECTIONS)

        # Get landmarks
        landmarks = result.pose_landmarks.landmark

        # Check for side-view posture defects
        kyphosis_detected, kyphosis_angle, vertical_alignment =
check_side_posture(landmarks)

        # Increment kyphosis frame count if kyphosis is detected
        if kyphosis_detected:
            kyphosis_frames += 1

        # Calculate percentage of kyphosis frames
        kyphosis_percentage = (kyphosis_frames /
detection_frames) * 100

        # Visualize the results
        cv2.putText(frame, f"Kyphosis Angle:
{kyphosis_angle:.1f}", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
(0, 255, 0), 2)
        cv2.putText(frame, f"Vertical Alignment:
{vertical_alignment:.1f}", (50, 80), cv2.FONT_HERSHEY_SIMPLEX,
0.8, (0, 255, 0), 2)
        cv2.putText(frame, f"Kyphosis Frames:
{kyphosis_frames}", (50, 110), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
(0, 0, 255), 2)
```

```
        cv2.putText(frame, f"Detection Frames:
{detection_frames}", (50, 140), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
(0, 255, 255), 2)
        cv2.putText(frame, f"Kyphosis %:
{kyphosis_percentage:.2f}%", (50, 170),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 0), 2)

        if kyphosis_detected:
            cv2.putText(frame, "Kyphosis Detected", (50, 200),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    else:
        # Reset counters if body is not detected
        body_present = False
        kyphosis_frames = 0
        detection_frames = 0

    # End timer for frame processing
    end_time = time.time()
    processing_time = (end_time - start_time) * 1000  # Convert
to milliseconds

    # Display the processing time on the frame
    cv2.putText(frame, f"Processing Time: {processing_time:.2f}
ms", (50, 230), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 0), 2)

    # Write the frame to the output video
    out.write(frame)

    # Display the frame
    cv2.imshow("Side-View Posture Analysis", frame)

    # Exit on pressing 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
out.release()
cv2.destroyAllWindows()

print(f"Output video saved at {output_path}")
```

ДОДАТОК В

Код програми для виявлення сколіозу

```
import cv2
import mediapipe as mp
import math
import time
import os

# Initialize MediaPipe Pose
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5)
mp_draw = mp.solutions.drawing_utils

def calculate_midpoint(point1, point2):
    """Calculate the midpoint between two landmarks."""
    return {
        "x": (point1.x + point2.x) / 2,
        "y": (point1.y + point2.y) / 2,
        "z": (point1.z + point2.z) / 2,
        "visibility": (point1.visibility + point2.visibility) /
2,
    }

def detect_scoliosis(landmarks):
    """Detect scoliosis based on lateral deviation of
midpoints."""
    left_shoulder =
landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value]
    right_shoulder =
landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value]
    left_hip = landmarks[mp_pose.PoseLandmark.LEFT_HIP.value]
    right_hip = landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value]

    # Calculate midpoints for shoulders and hips
    mid_shoulder = calculate_midpoint(left_shoulder,
right_shoulder)
    mid_hip = calculate_midpoint(left_hip, right_hip)

    # Calculate lateral deviation
    lateral_deviation = abs(mid_shoulder["x"] - mid_hip["x"])

    # Threshold for scoliosis detection (adjust based on data)
    scoliosis_detected = lateral_deviation > 0.05  # Example
threshold for deviation

    return scoliosis_detected, lateral_deviation
```

```
# Open the video file
video_path = "sample.mp4"  # Replace with your video path
cap = cv2.VideoCapture(video_path)

if not cap.isOpened():
    print("Error: Unable to open video file.")
    exit()

# Video writer setup
output_path = os.path.join(os.getcwd(),
"output_video_scoliosis.avi")  # Save in the same folder as the
script
fps = int(cap.get(cv2.CAP_PROP_FPS))
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter(output_path, fourcc, fps, (frame_width,
frame_height))

# Initialize counters
scoliosis_frames = 0
detection_frames = 0
body_present = False

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("End of video reached.")
        break

    # Measure processing start time
    start_time = time.time()

    # Convert the frame to RGB
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    result = pose.process(rgb_frame)

    if result.pose_landmarks:
        # Body is detected
        body_present = True
        detection_frames += 1

        # Draw landmarks
        mp_draw.draw_landmarks(frame, result.pose_landmarks,
mp_pose.POSE_CONNECTIONS)

        # Get landmarks
        landmarks = result.pose_landmarks.landmark

        # Check for scoliosis
        scoliosis_detected, lateral_deviation =
detect_scoliosis(landmarks)
```

```
        # Increment scoliosis frame count if scoliosis is
detected
        if scoliosis_detected:
            scoliosis_frames += 1

        # Calculate percentage of scoliosis frames
        scoliosis_percentage = (scoliosis_frames /
detection_frames) * 100

        # Visualize the results
        cv2.putText(frame, f"Lateral Deviation:
{lateral_deviation:.4f}", (50, 50), cv2.FONT_HERSHEY_SIMPLEX,
0.8, (0, 255, 0), 2)
        cv2.putText(frame, f"Scoliosis Frames:
{scoliosis_frames}", (50, 80), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
(0, 0, 255), 2)
        cv2.putText(frame, f"Detection Frames:
{detection_frames}", (50, 110), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
(0, 255, 255), 2)
        cv2.putText(frame, f"Scoliosis %:
{scoliosis_percentage:.2f}%", (50, 140),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 0), 2)

        if scoliosis_detected:
            cv2.putText(frame, "Scoliosis Detected", (50, 170),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    else:
        # Reset counters if body is not detected
        body_present = False
        scoliosis_frames = 0
        detection_frames = 0

    # Measure processing end time and calculate duration
    end_time = time.time()
    processing_time = (end_time - start_time) * 1000  # Convert
to milliseconds

    # Display the processing time on the frame
    cv2.putText(frame, f"Processing Time: {processing_time:.2f}
ms", (50, 200), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 0), 2)

    # Write the frame to the output video
    out.write(frame)

    # Display the frame
    cv2.imshow("Scoliosis Detection", frame)

    # Exit on pressing 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
cap.release()
out.release()
cv2.destroyAllWindows()

print(f"Output video saved at {output_path}")
```