In [ ]:
```python
#Import libraries

from mediapipe.tasks import python
from mediapipe.tasks.python import vision
import cv2
import mediapipe as mp

from mediapipe.python.solutions.pose import PoseLandmark
import numpy as np
from google.colab.patches import cv2_imshow
from google.colab import files

from cvzone.PoseModule import PoseDetector
import math
```

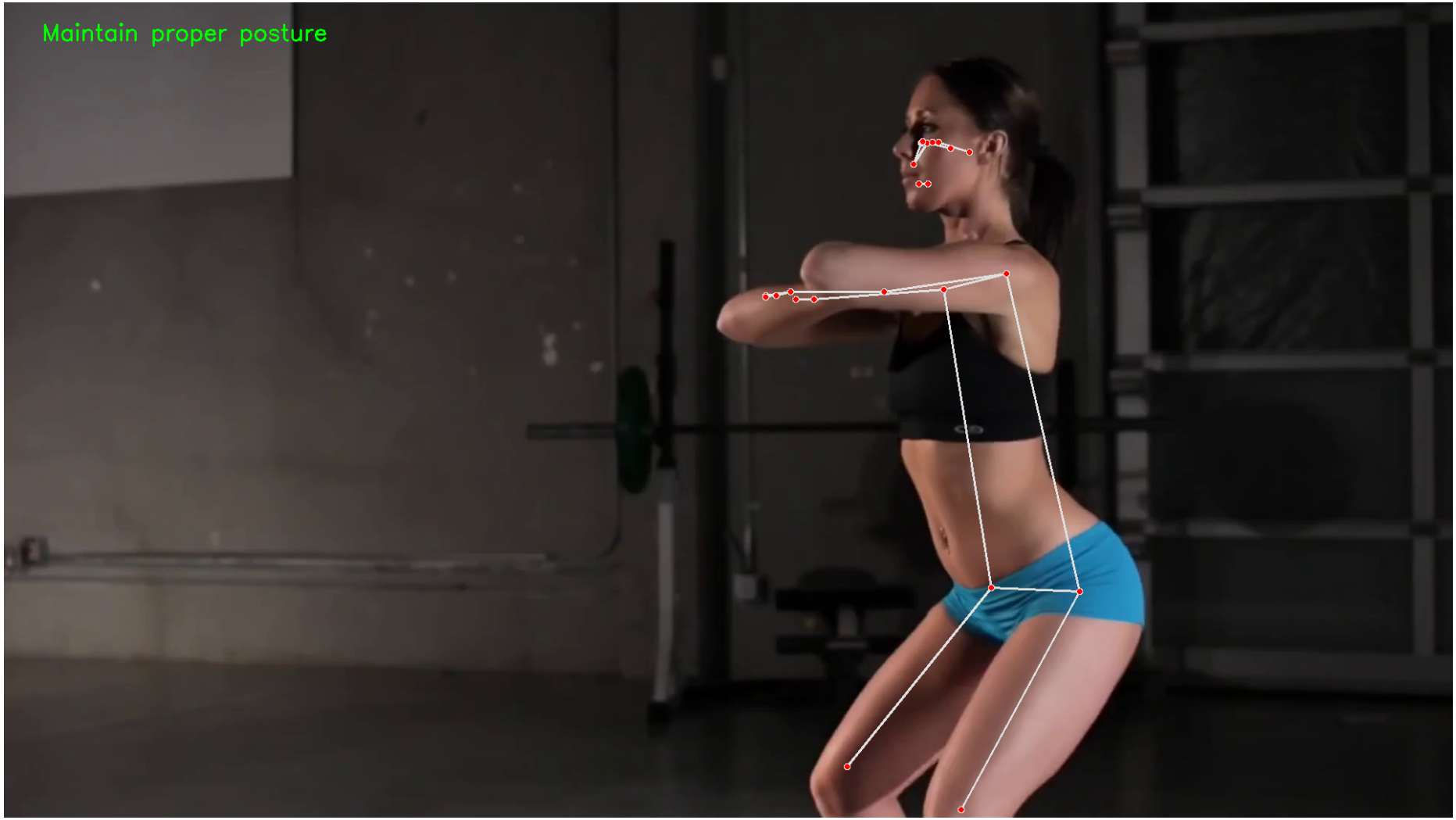
```
Requirement already satisfied: cvzone in /usr/local/lib/python3.10/dist-packages (1.6.1)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (from cvzone) (4.8.0.76)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from cvzone) (1.25.2)
```

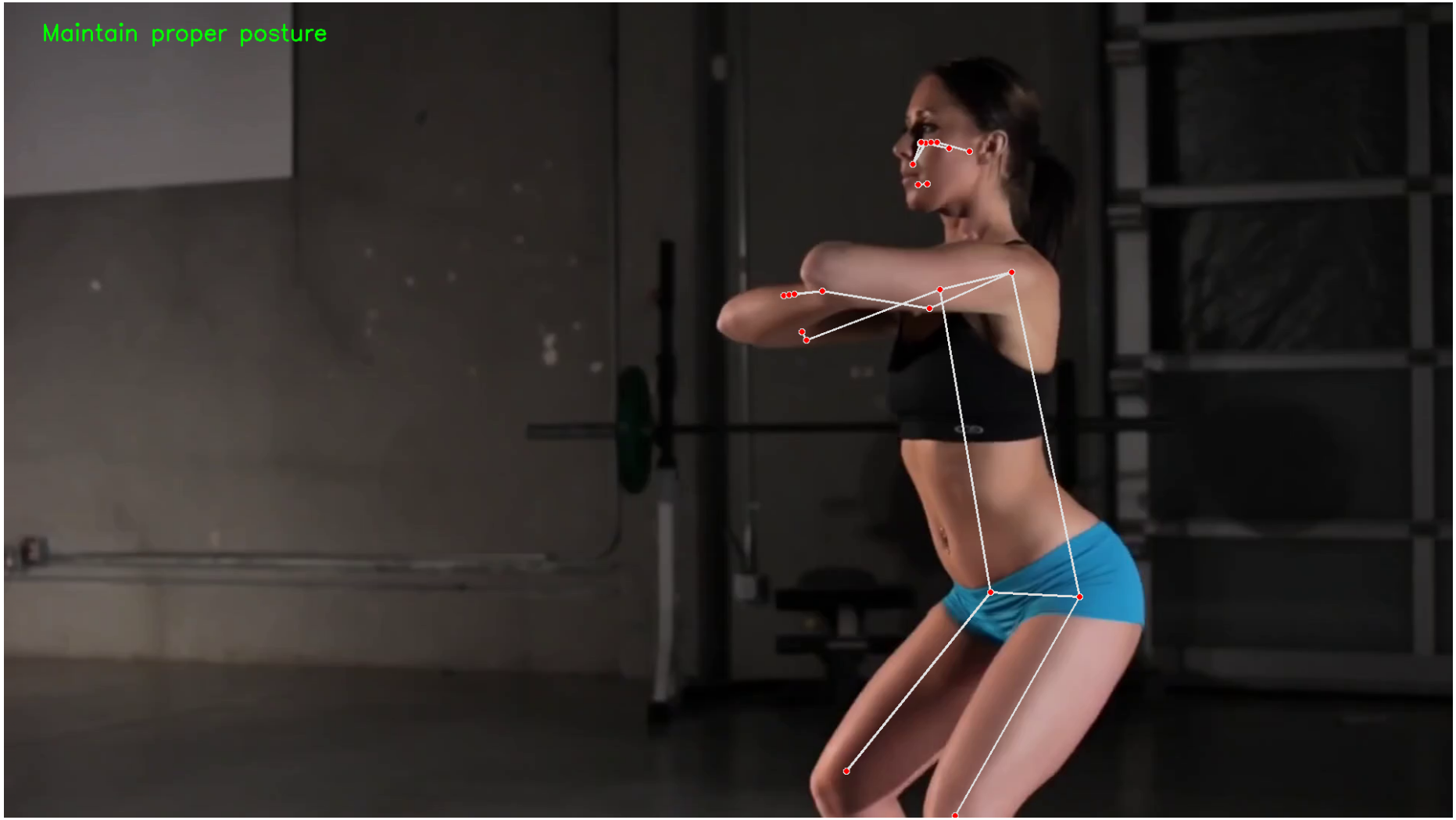
In [ ]:
```python
## Data
cap = cv2.VideoCapture('/content/drive/MyDrive/Dataset/document_6224009744844066547.mp4')
```

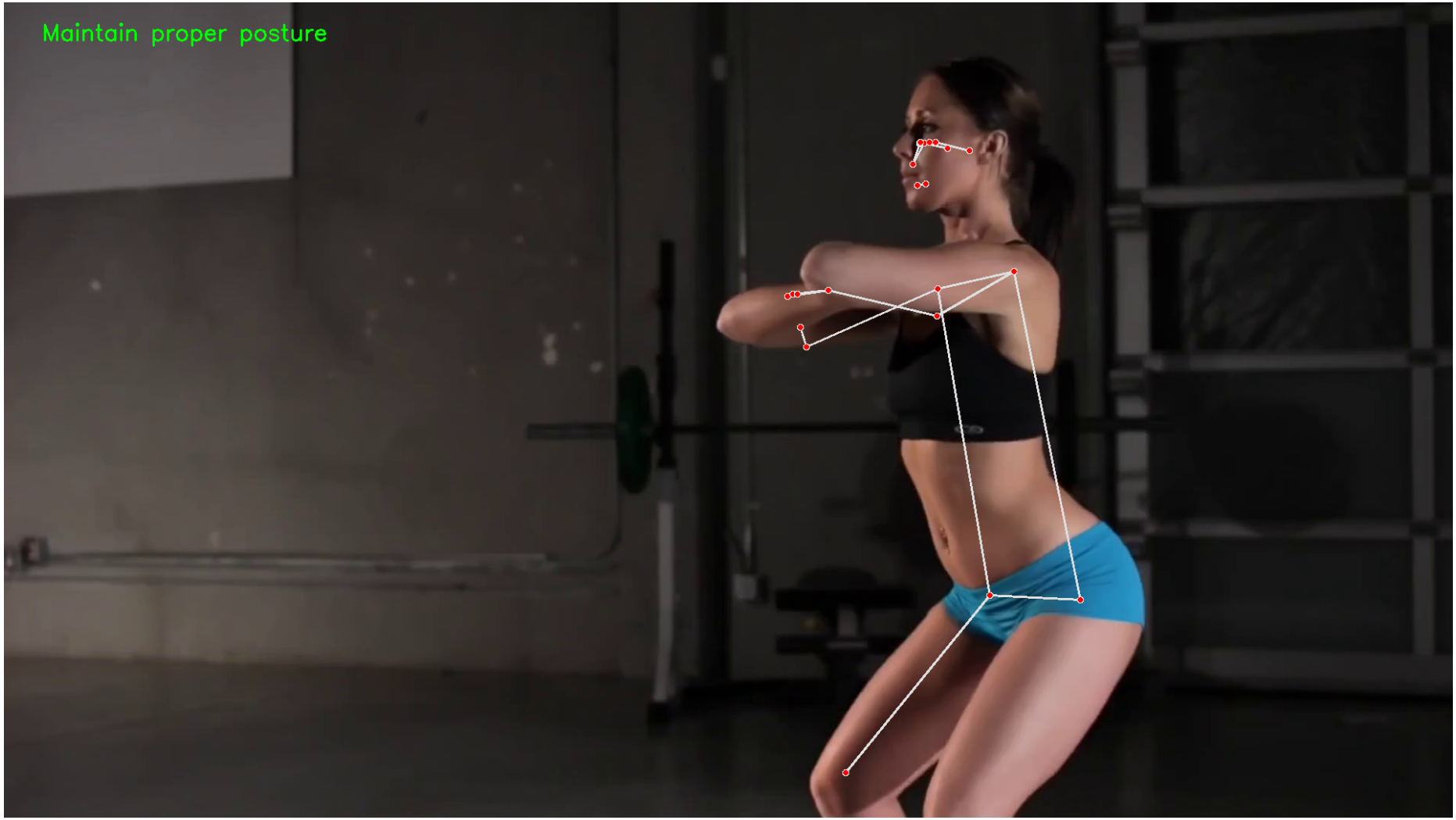
In [ ]:
```python
# Create a VideoWriter object to save the processed video
output_video_path = 'output_video.mp4'
fps = int(cap.get(cv2.CAP_PROP_FPS))
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter(output_video_path, fourcc, fps, (width, height))

# Create a pose landmarker instance with the video mode:
pose_landmarker = mp.solutions.pose.Pose(static_image_mode=False, model_complexity=0, min_detection_confidence=0.5, min_tracking_

def analyze_squat(pose_landmarks):
    """Analyze squat movement based on pose landmarks."""
    # Extract landmark points for relevant body parts
    left_hip = pose_landmarks.landmark[PoseLandmark.LEFT_HIP]
    left_knee = pose_landmarks.landmark[PoseLandmark.LEFT_KNEE]
    left_ankle = pose_landmarks.landmark[PoseLandmark.LEFT_ANKLE]
    right_hip = pose_landmarks.landmark[PoseLandmark.RIGHT_HIP]
    right_knee = pose_landmarks.landmark[PoseLandmark.RIGHT_KNEE]
    right_ankle = pose_landmarks.landmark[PoseLandmark.RIGHT_ANKLE]
```

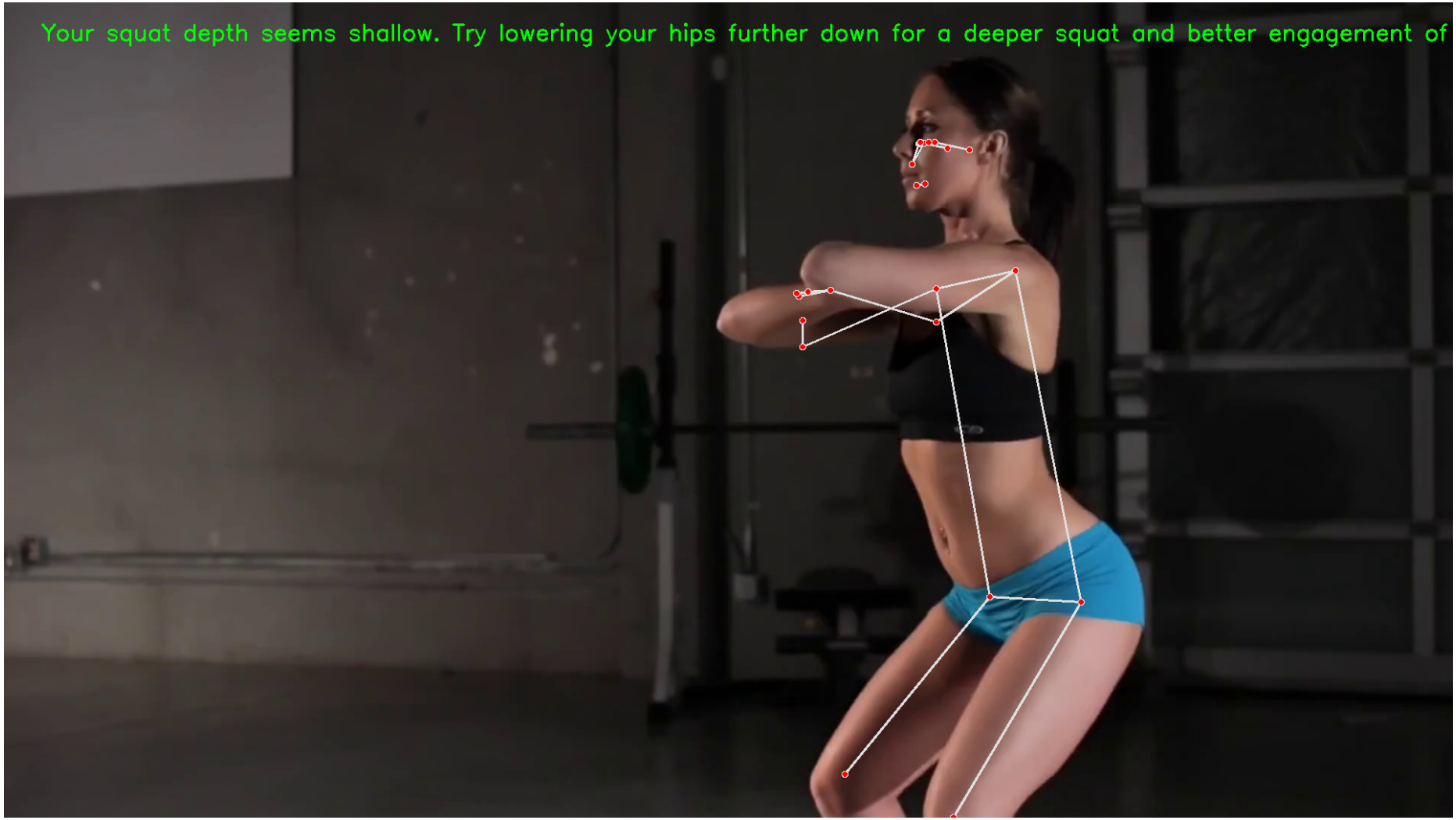
```
    # Calculate the angle between hips, knees, and ankles
    left_angle = calculate_angle(left_hip, left_knee, left_ankle)
    right_angle = calculate_angle(right_hip, right_knee, right_ankle)

    # Provide feedback based on the angle
    if left_angle < 25 and right_angle < 25:
            feedback = "Your body posture resembles standing rather than squatting. Focus on bending your knees and lowering your
    elif 25 <= left_angle < 90 and 25 <= right_angle < 90:
        feedback = "Good squat form"
    elif 90 <= left_angle <= 115 and 90 <= right_angle <= 115:
        feedback = "You're squatting with a moderate range of motion. Aim for a deeper squat for better muscle engagement."
    elif 115 <= left_angle <= 150 and 115 <= right_angle <= 150:
        feedback = "Your squat depth seems shallow. Try lowering your hips further down for a deeper squat and better engagement
    else:
        feedback = "Maintain proper posture"

    return feedback

def calculate_angle(a, b, c):
    """Calculate the angle between three points."""
    ab = (b.x - a.x, b.y - a.y)
    bc = (c.x - b.x, c.y - b.y)
    dot_product = ab[0] * bc[0] + ab[1] * bc[1]
    magnitude_ab = (ab[0] ** 2 + ab[1] ** 2) ** 0.5
    magnitude_bc = (bc[0] ** 2 + bc[1] ** 2) ** 0.5
    cos_angle = dot_product / (magnitude_ab * magnitude_bc)
    angle_rad = np.arccos(cos_angle)
    angle_deg = np.degrees(angle_rad)
    return angle_deg

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Convert the frame to RGB for processing by MediaPipe
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Perform pose landmarking on the frame
    pose_landmark_results = pose_landmarker.process(rgb_frame)

    # Check if pose landmarks are detected
    if pose_landmark_results.pose_landmarks:
```

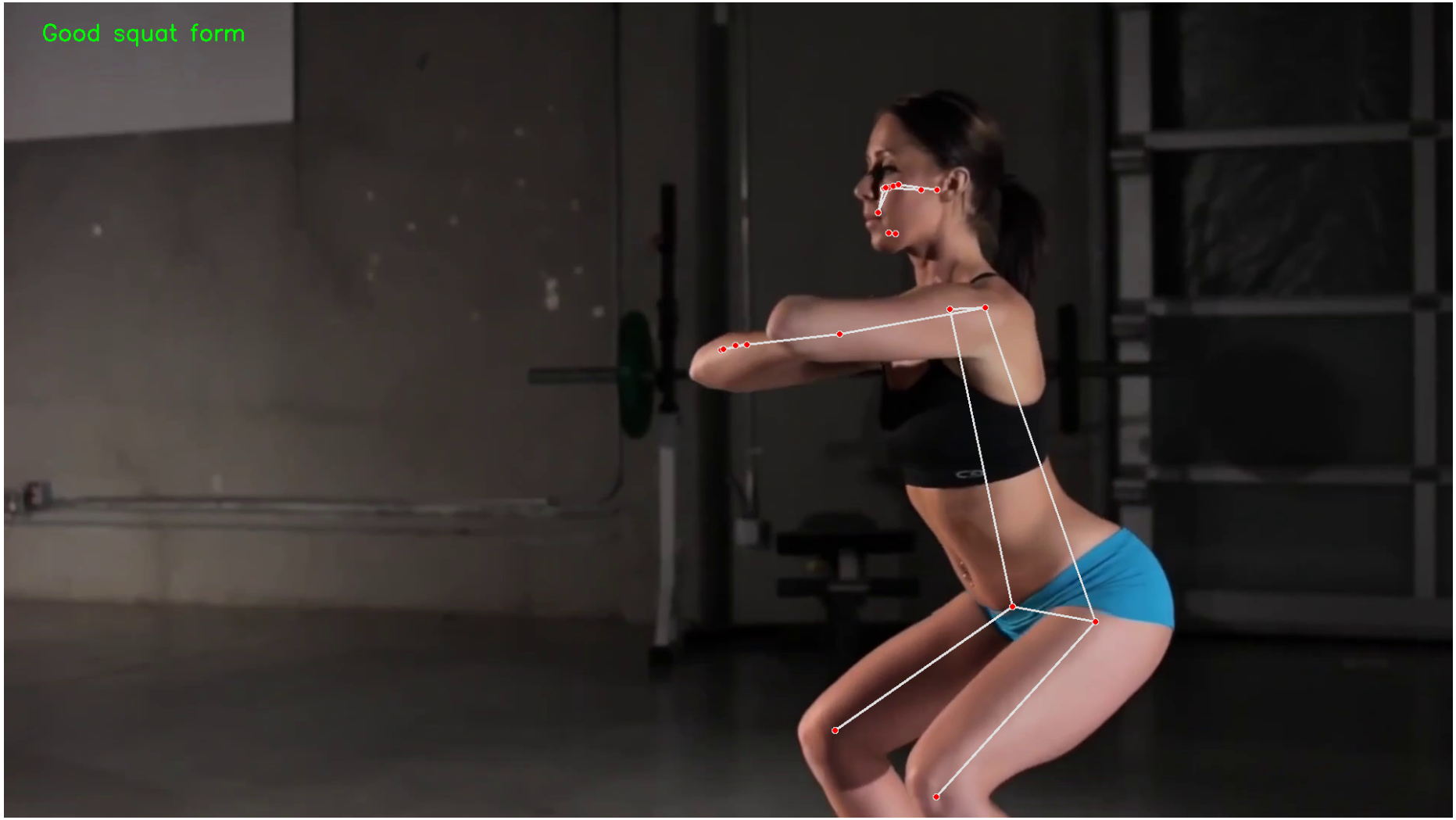
```
        # Analyze the movement and provide feedback
        feedback = analyze_squat(pose_landmark_results.pose_landmarks)
        # Display feedback on the frame
        cv2.putText(frame, feedback, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

    # Draw pose landmarks on the frame (optional)
    mp.solutions.drawing_utils.draw_landmarks(frame, pose_landmark_results.pose_landmarks, mp.solutions.pose.POSE_CONNECTIONS)

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

    # Display the frame
    cv2_imshow(frame)

    # Break the loop if 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the VideoCapture, VideoWriter, and destroy all OpenCV windows
cap.release()
out.release()
cv2.destroyAllWindows()

# Download the output video
files.download(output_video_path)
```

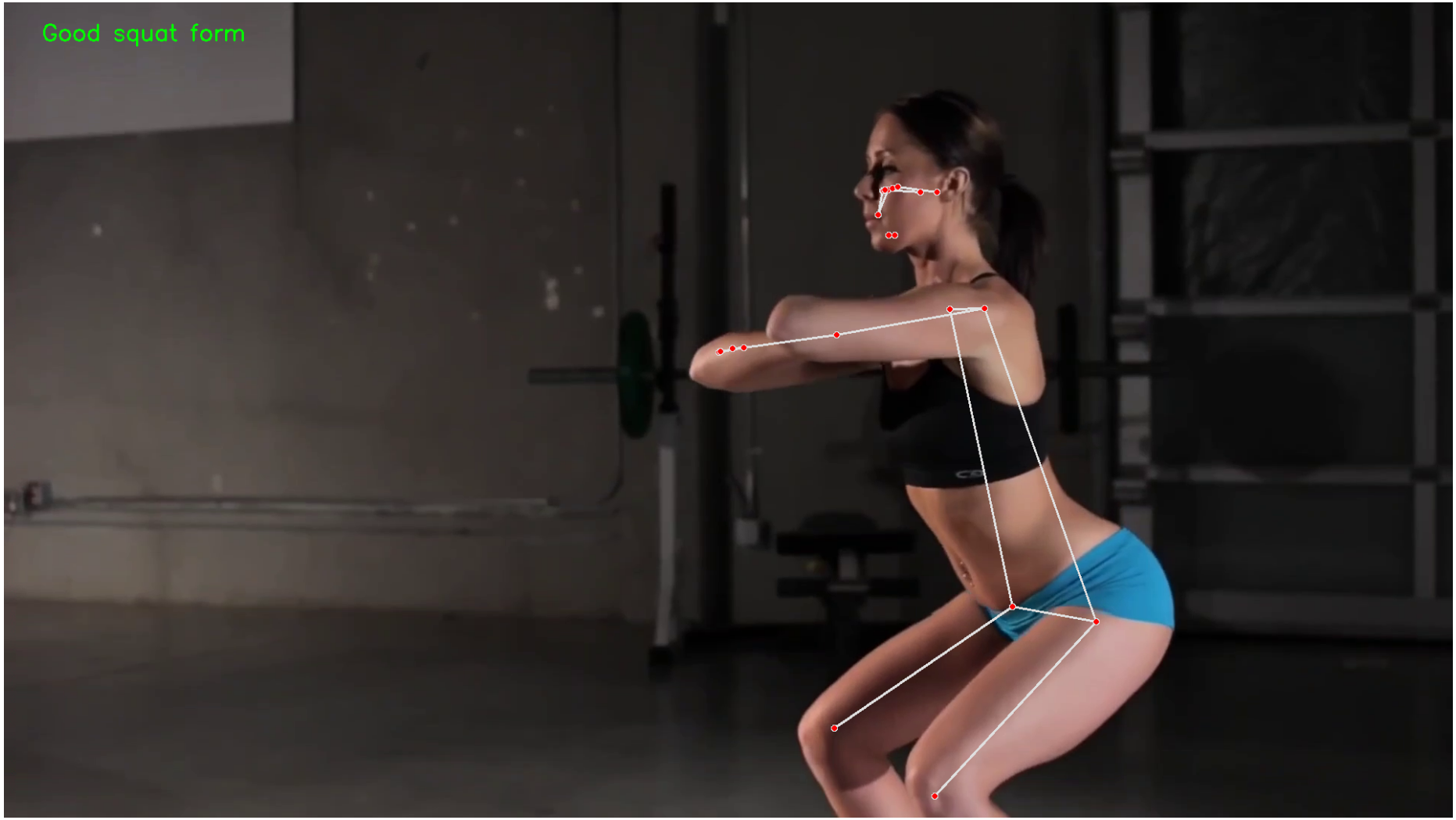

Maintain proper posture

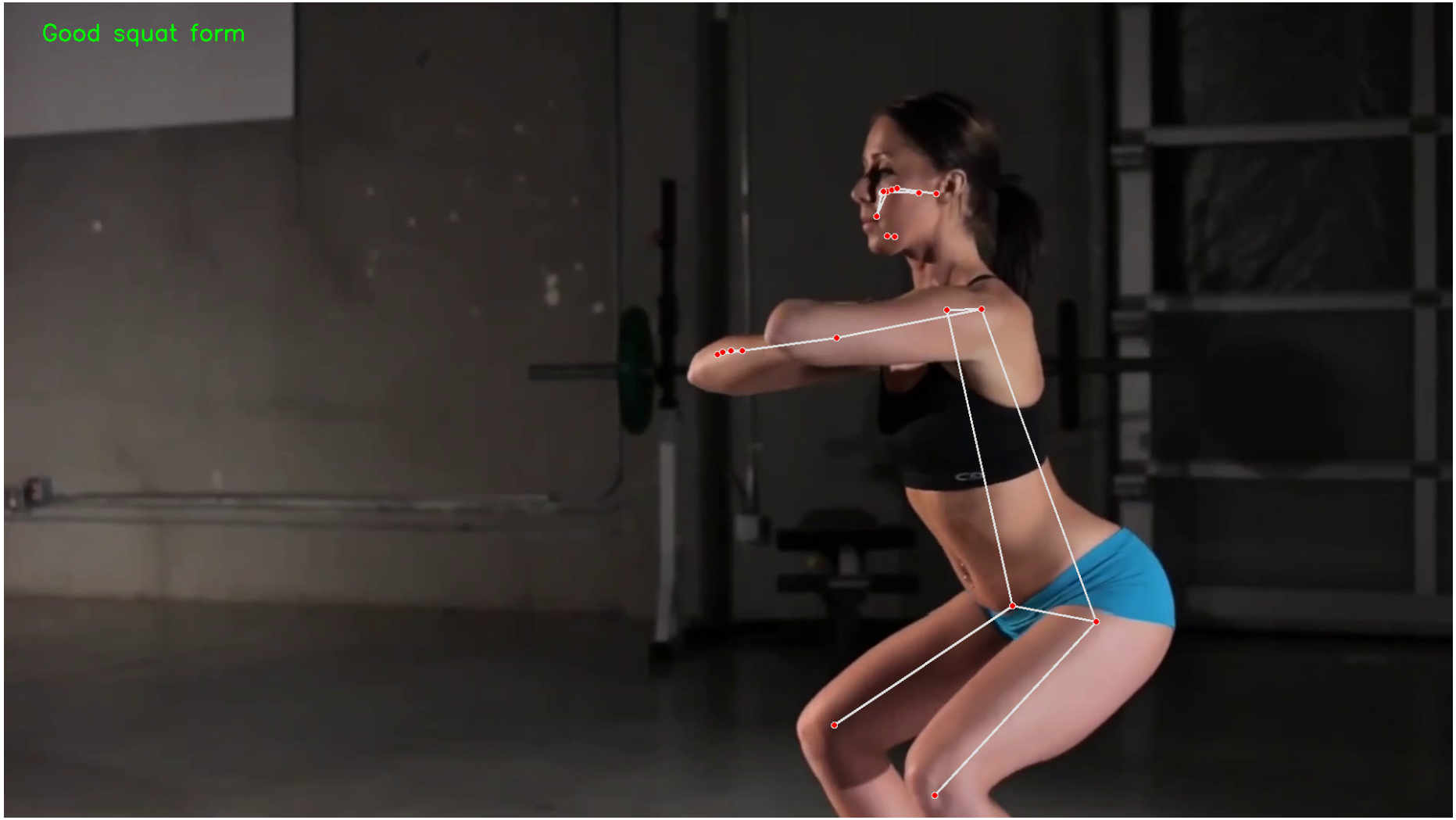

Maintain proper posture

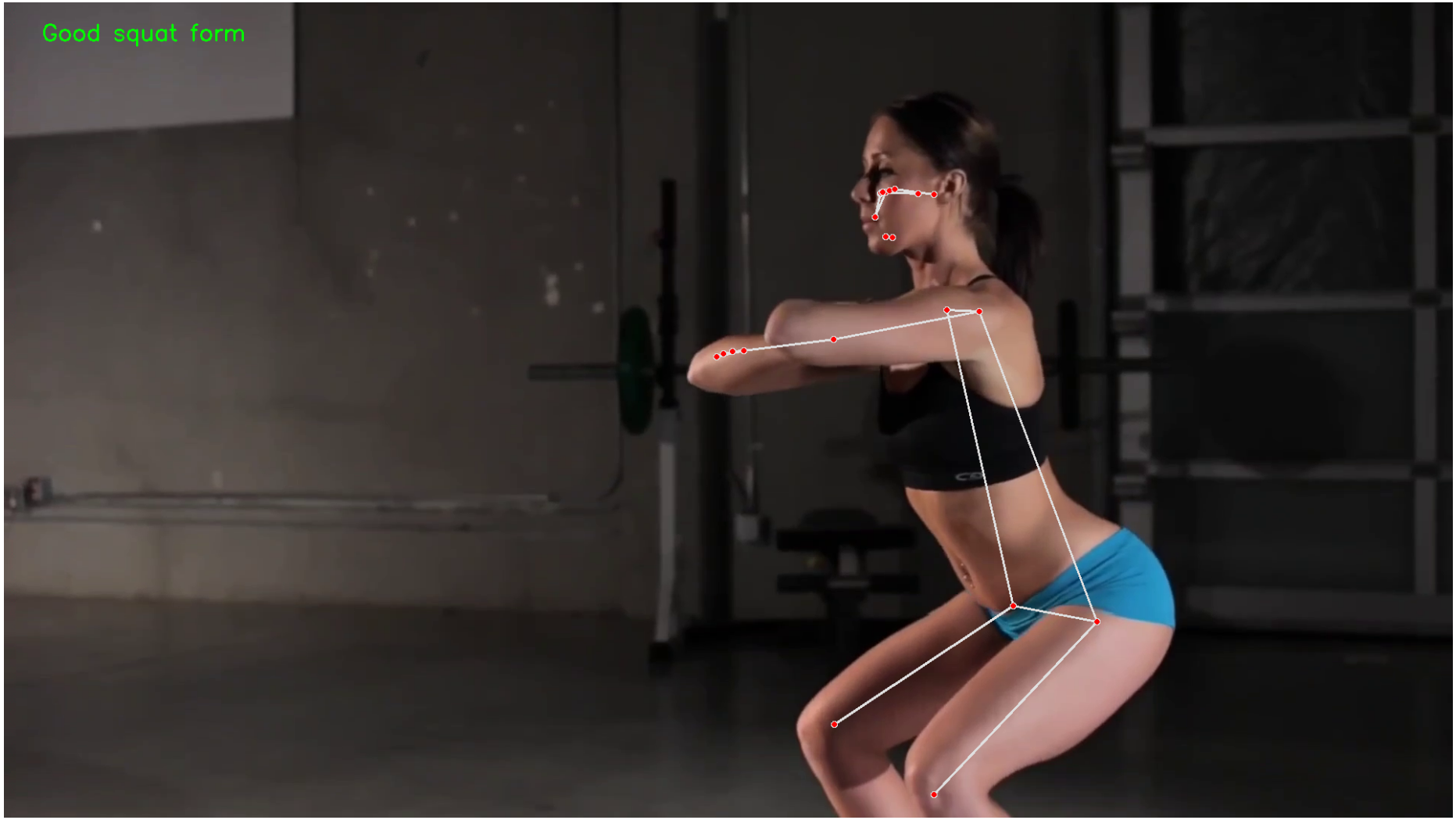

Maintain proper posture

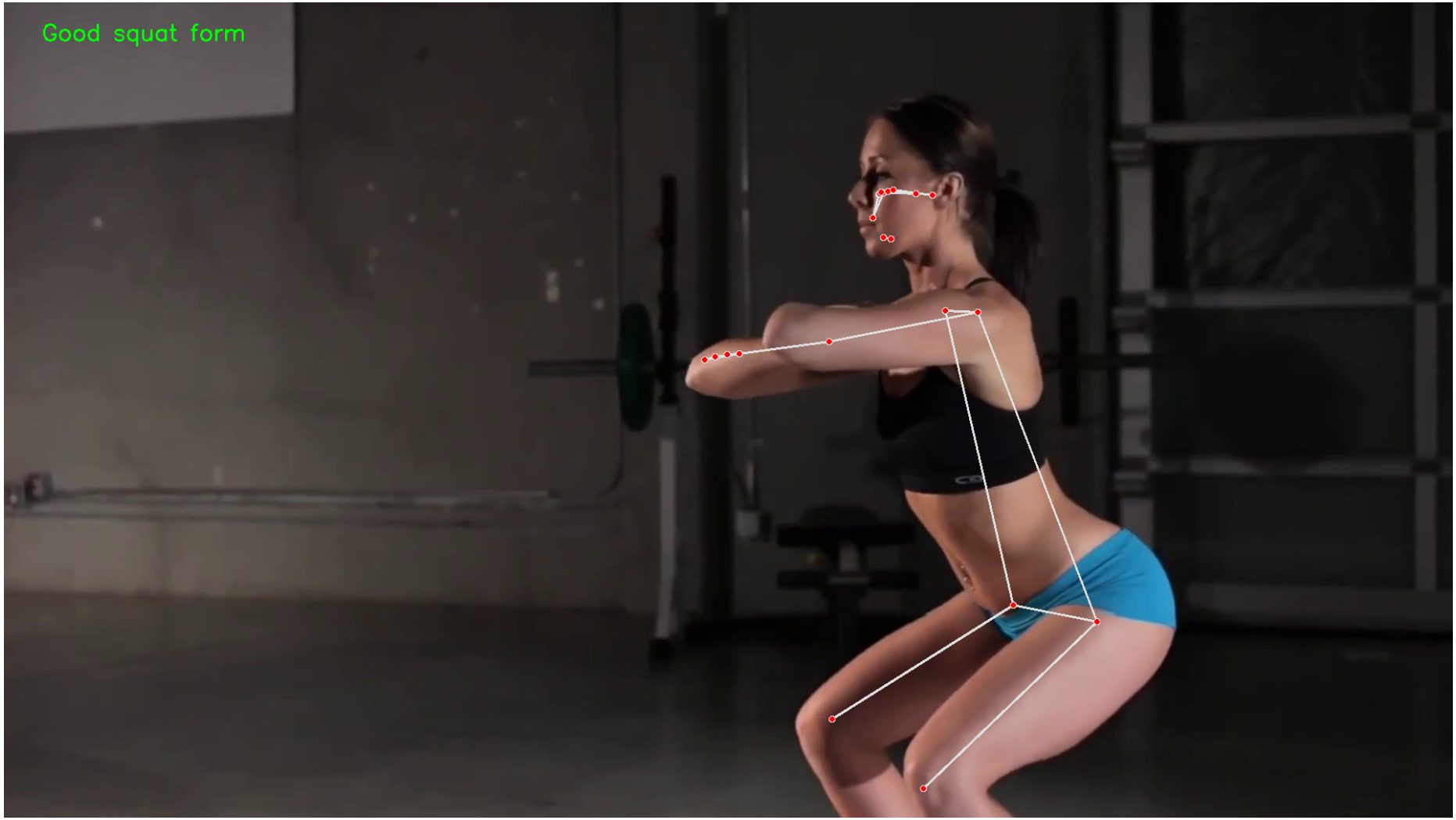

Your squat depth seems shallow. Try lowering your hips further down for a deeper squat and better engagement of

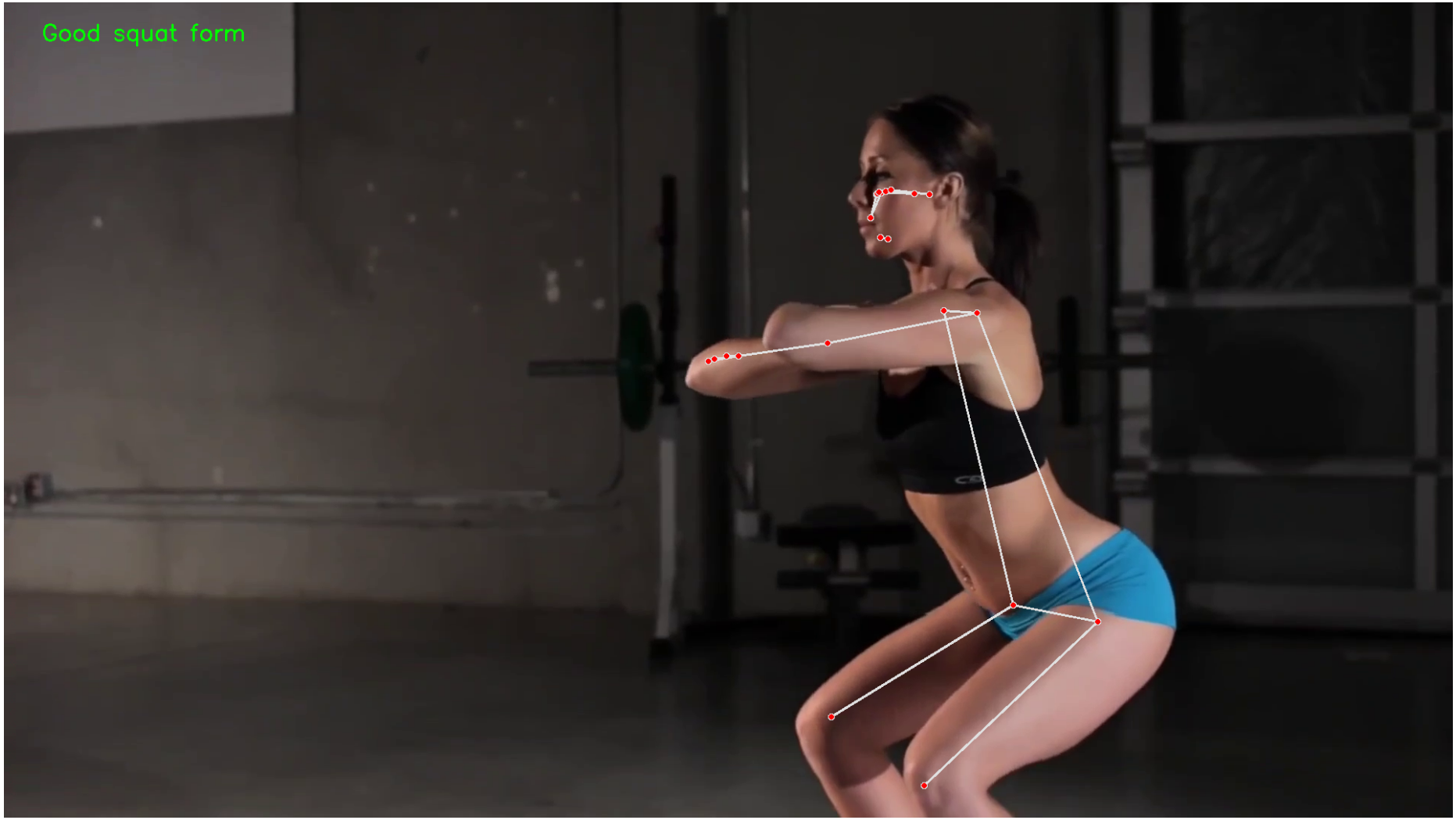

Good squat form

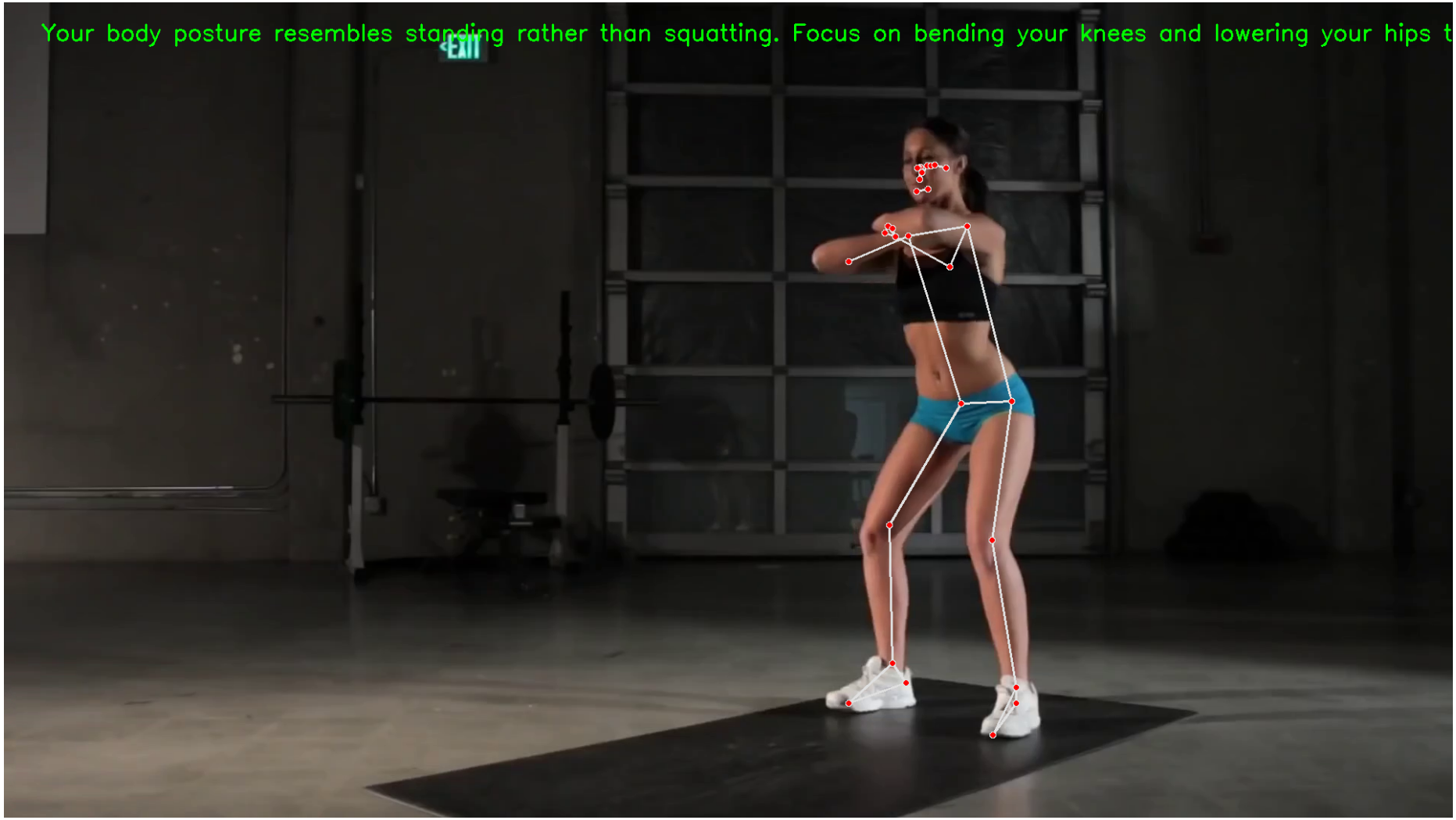

Good squat form

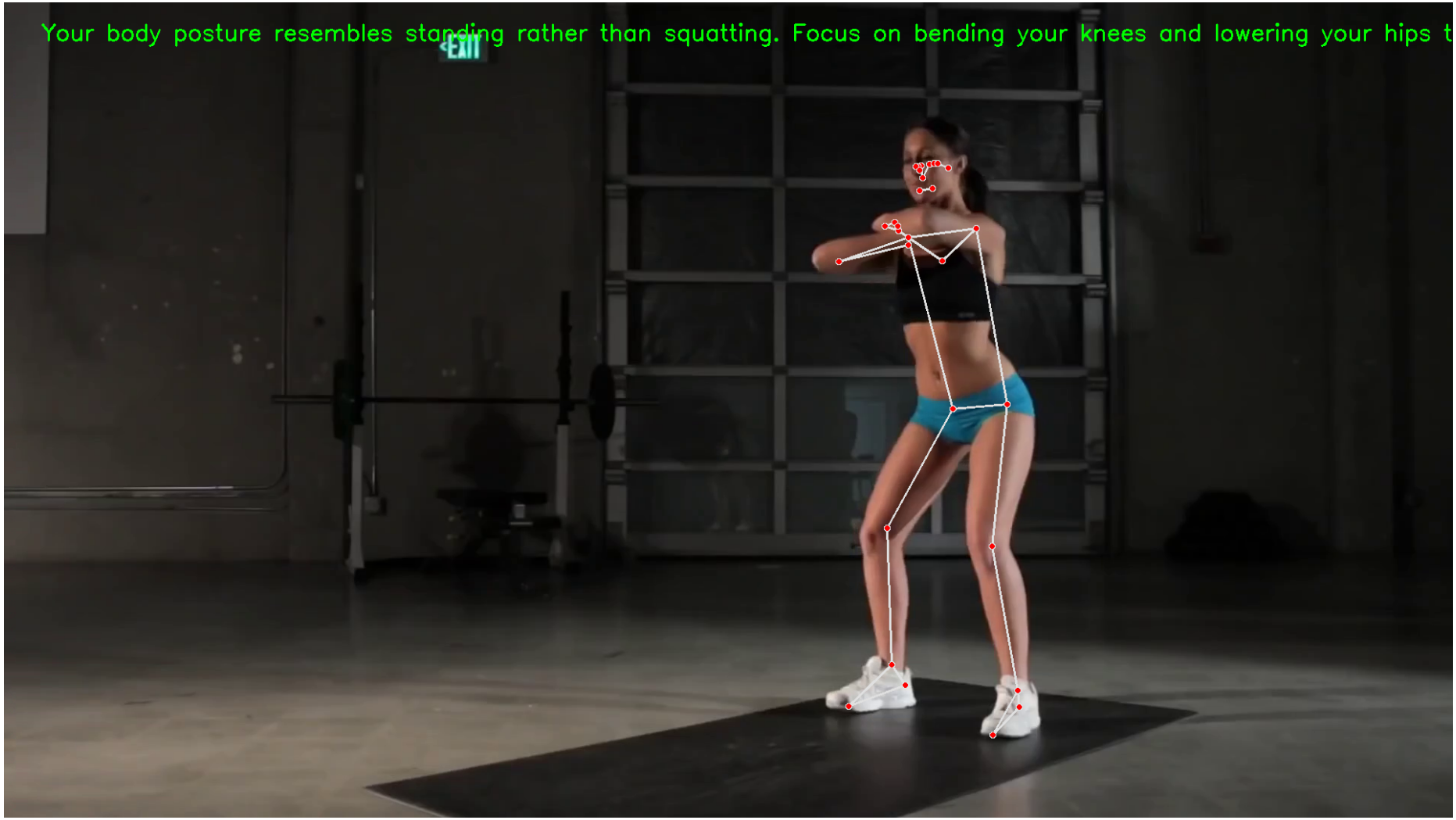

Good squat form

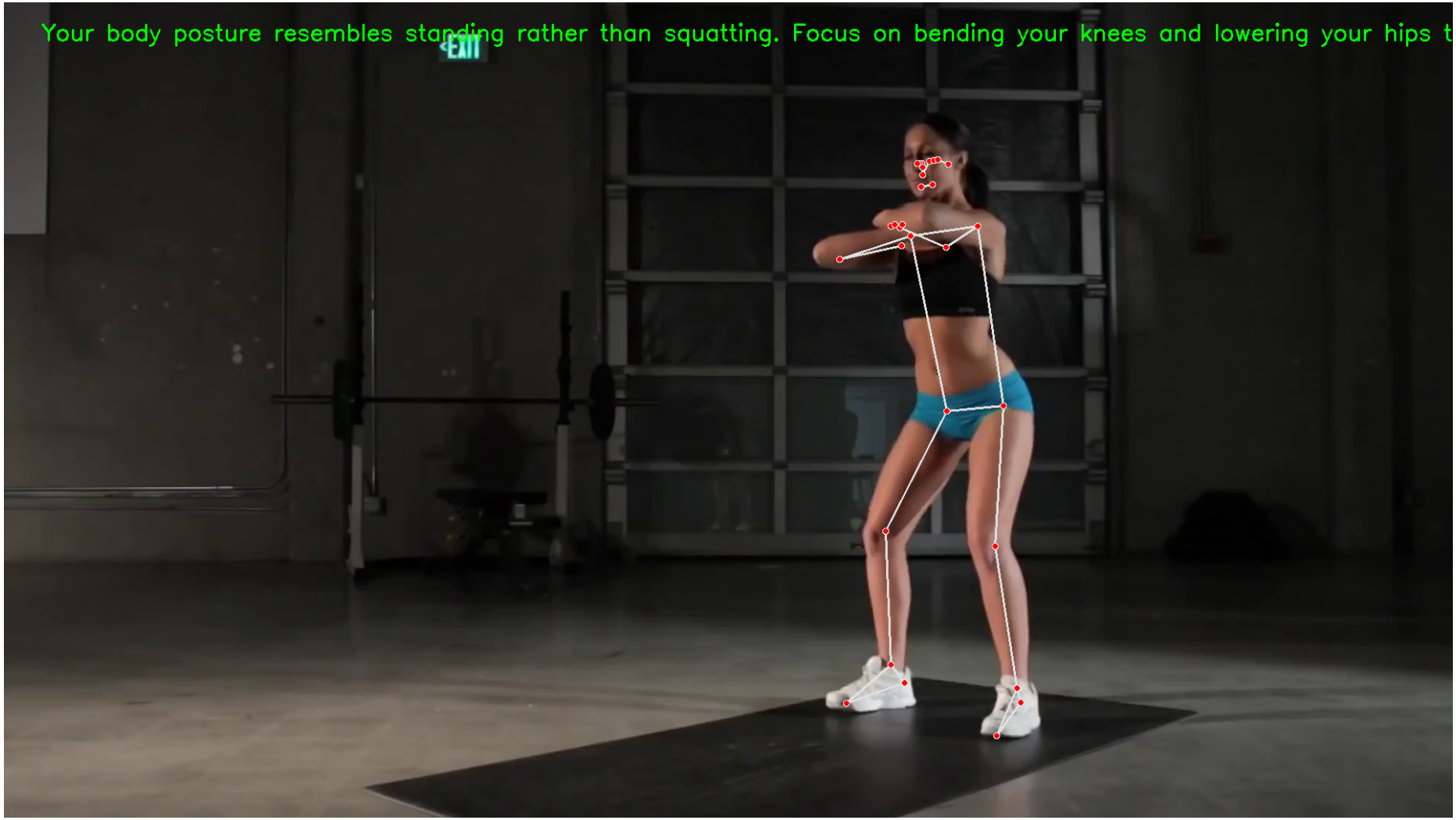

Good squat form

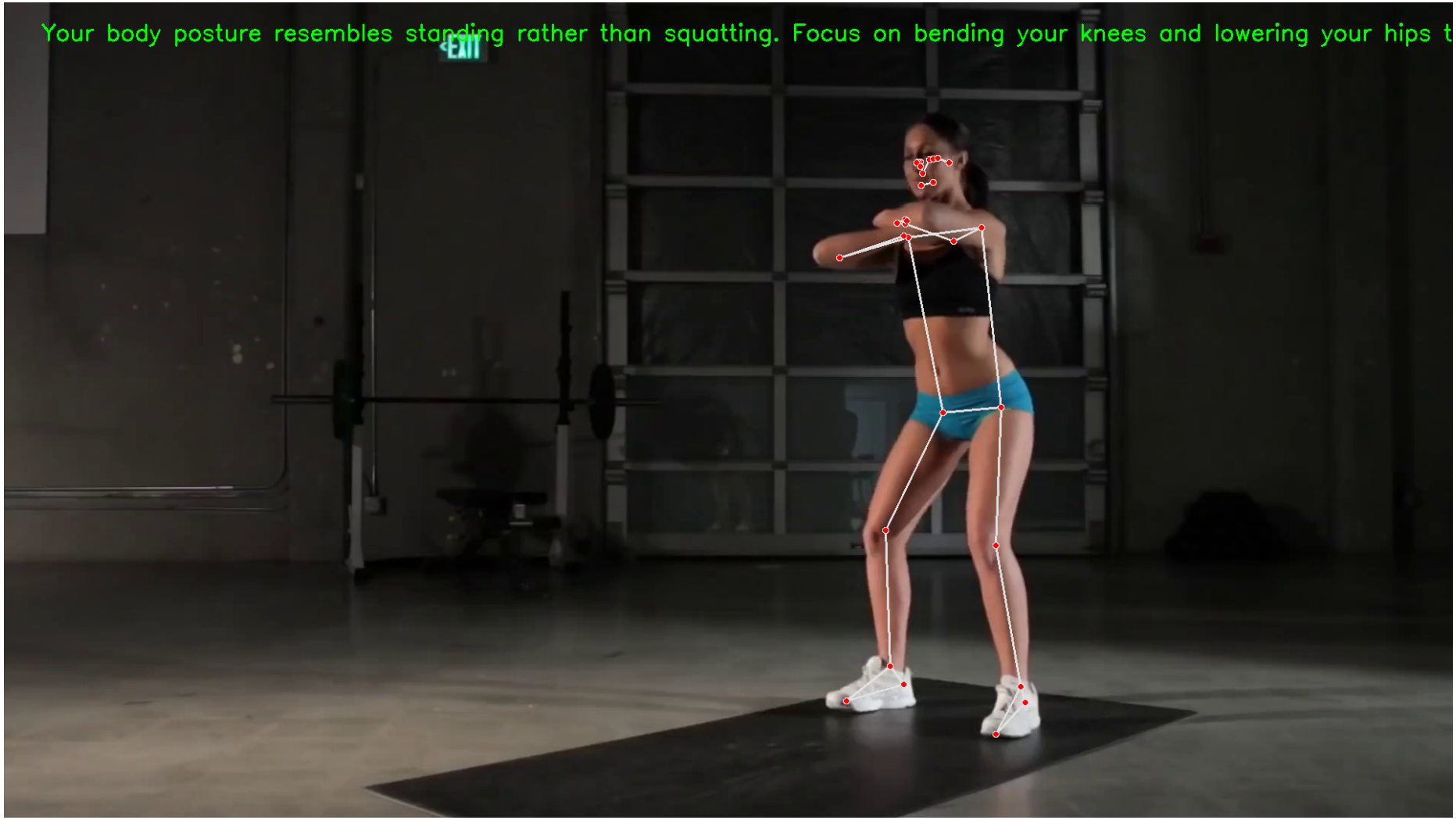

Good squat form


Good squat form


Your body posture resembles standing rather than squatting. Focus on bending your knees and lowering your hips t
EXIT


Your body posture resembles standing rather than squatting. Focus on bending your knees and lowering your hips t
EXIT


Your body posture resembles standing rather than squatting. Focus on bending your knees and lowering your hips t
EXIT


Your body posture resembles standing rather than squatting. Focus on bending your knees and lowering your hips t
EXIT