**HUMAN ACTIVITY RECOGNITION SYSTEM**

Hello Tech Watchers I am developing a system that will detect the human activities like sitting, standing, walking in the realtime videos.We will use Human Pose estimation for this.

# **HUMAN POSE ESTIMATION**

# Human pose estimation algorithms analyze the video frames to determine the position and orientation of a person's body in each frame. Once you have the pose information, you can infer whether the person is standing or sitting based on the positions of key body parts such as the head, shoulders, hips, knees, and feet.

**STEPS TO ACCOMPLISH THIS TASK**

**Data Collection**: Collect a dataset of videos containing people standing and sitting in various environments and camera angles.

**Preprocessing**: Preprocess the videos to standardize the resolution, frame rate, and format if necessary.

**Human Pose Estimation**: Apply a human pose estimation algorithm to each frame of the video to extract the poses of the individuals. There are various pose estimation models available, such as OpenPose, PoseNet, and DeepLabCut.

**Feature Extraction**: Extract features from the pose data that can help distinguish between standing and sitting postures. For example, the angles between different body parts, the height of the hips relative to the ground, or the distance between the feet can be useful features.

**Classification**: Train a machine learning model, such as a support vector machine (SVM), random forest, or deep neural network, using the extracted features to classify each frame as either standing or sitting.

**Evaluation**: Evaluate the performance of the trained model on a separate validation dataset to assess its accuracy and generalization ability.

**Integration**: Integrate the trained model into your video processing pipeline to automatically detect whether a person is standing or sitting in real-time or on pre-recorded videos.

**REFRENCES**

[**https://dayta.nwu.ac.za/articles/dataset/Human\_pose\_dataset\_sit\_stand\_pose\_classes\_/23290937?file=41049788**](https://dayta.nwu.ac.za/articles/dataset/Human_pose_dataset_sit_stand_pose_classes_/23290937?file=41049788)

[**https://github.com/hafizas101/Real-time-human-pose-estimation-and-classification**](https://github.com/hafizas101/Real-time-human-pose-estimation-and-classification)

**https://www.youtube.com/watch?v=rh8Vzp0REP0**

<https://op>

**STEP 1: PREPROCESSING THE DATA**

I am using video editing software or programming libraries (e.g., OpenCV in Python) to standardize the resolution, frame rate, and format of the collected videos. Resize the videos to a consistent resolution, such as 720p or 1080p, to ensure uniformity across the dataset. Adjust the frame rate of the videos to a common value, such as 30 frames per second (FPS), to maintain consistency in temporal information. Convert the videos to a standardized format, such as MP4 or AVI, for compatibility and ease of use in further analysis.

import cv2

**CODE FOR VIDEO PREPROCESSING**

def preprocess\_video(input\_path, output\_path, target\_resolution=(1280, 720), target\_fps=30, target\_format='mp4'):

# Open the input video file

cap = cv2.VideoCapture(input\_path)

# Get the original video properties

original\_fps = cap.get(cv2.CAP\_PROP\_FPS)

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

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

# Create a VideoWriter object to write the processed video

fourcc = cv2.VideoWriter\_fourcc(\*'mp4v') # Specify the codec for MP4 format

out = cv2.VideoWriter(output\_path, fourcc, target\_fps, target\_resolution)

# Loop through the video frames and process each frame

while True:

ret, frame = cap.read()

if not ret:

break # Break the loop if no more frames are available

# Resize the frame to the target resolution

frame = cv2.resize(frame, target\_resolution)

# Write the resized frame to the output video

out.write(frame)

# Release the video capture and writer objects

cap.release()

out.release()

# Print a message to indicate completion

print("Video preprocessing complete.")

# Example usage:

input\_video\_path = 'input\_video.mp4'

output\_video\_path = 'output\_video.mp4'

preprocess\_video(input\_video\_path, output\_video\_path)

**Preprocessing has been done.**

**STEP 2 : ANNOTATION (manually or automated)**

After obtaining the preprocessed video **output.mp4**, you can proceed with applying pose estimation to detect the poses of individuals in each frame. Here's a suggested workflow to continue the process:

**Apply Pose Estimation:**

Use a pre-trained pose estimation model to analyze each frame of the preprocessed video and detect the poses of individuals. The pose estimation model should provide keypoint coordinates representing the joints of each person detected in the frame.

**Posture Classification:**

Based on the detected poses, classify each frame as "standing" or "sitting" using predefined criteria or thresholds. You can use the positions of key body joints (e.g., hips, knees) to determine whether a person is standing or sitting.

**Automated Annotation:**

Automatically annotate each frame of the video with the determined posture classification (e.g., "standing" or "sitting"). This step involves processing the output of the pose estimation model and applying the classification criteria to assign labels to each frame.

**Manual Verification (Optional):**

Despite the automated annotation, consider manual verification to ensure the accuracy of annotations, especially in challenging cases or where the pose estimation model may struggle to accurately detect poses.

**Save Annotations:**

Save the annotated video frames along with their corresponding posture labels to a structured format such as CSV, JSON, or XML. Ensure that each annotation entry includes the frame number and the assigned posture label.

**Documentation and Quality Control:**

Document the annotation process, including details of the pose estimation model used, classification criteria, and any manual verification steps performed.

Implement quality control measures to ensure consistency and accuracy in the annotated data, such as cross-validation or inter-annotator agreement analysis.

**OPENPOSE ALGORITHM**

## OpenPose is one of the most widely used pose estimation models, known for its accuracy and real-time performance. It detects multiple key body joints and provides accurate pose estimations for various human poses, including complex interactions between body parts.

## We will apply the pose estimation model to each frame of the preprocessed video.

# To apply pose estimation to each frame of the preprocessed video and detect the poses of individuals, you'll need to use a pre-trained pose estimation model. Here's a step-by-step guide on how to do this using the OpenPose model with Python and OpenCV.

# # pip install opencv-python(open terminal and type it)

# #Now download the pretrained model of openpose from it repository.

enposes.com