# **Detailed Documentation for Video Processing and Keypoint Visualization using Yolov8 with the 3DPW Dataset**

## **Overview**

This script performs video processing for human pose estimation using YOLO from the Ultralytics library. It downloads and unzips video data, extracts keypoints, and visualizes these keypoints over video frames and on a blank canvas.

Link to Notebook: <https://github.com/Aadharsh1/ML-Deep-Learning/blob/main/Pose_Estimation_Experiments/3DPW_Dataset_Experiments/Yolov8_3DPW.ipynb>

## **Dependencies**

The script utilizes several libraries:

* os for directory operations.
* cv2 (OpenCV) for image and video processing.
* numpy for numerical operations.
* pickle for loading serialized Python object files.
* YOLO from Ultralytics for pose estimation.

## **Downloading and Preparing Data**

The script downloads two datasets:

1. **Sequence Files**: Containing pose sequence information.
2. **Image Files**: Containing corresponding images for sequences.

!wget -O sequenceFiles.zip [URL]

!wget -O imageFiles.zip [URL]

!unzip sequenceFiles.zip

!unzip imageFiles.zip

## **Function: create\_video\_from\_images**

Converts a sequence of images into a video file.

**Parameters:**

* sequence\_name: Name of the sequence to process.
* input\_dir: Directory containing image files.
* output\_dir: Destination directory for the output video.
* fps: Frames per second of the output video.

**Process:**

1. Sorts all image files.
2. Reads the first image to determine video dimensions.
3. Initializes a VideoWriter object.
4. Iterates over all images, reading and adding them to the video stream.

## **YOLO Model Initialization**

Initializes a YOLO model for pose estimation with a pre-trained model.

model = YOLO('yolov8n-pose.pt')

## **Keypoint Detection Loop**

Processes each video, detects keypoints using the YOLO model, and stores the keypoints for later analysis.

for sequence\_name in sequence\_names:

# Video capturing and processing logic here

## **Function: get\_2d\_joint\_locations**

Loads joint location data from a sequence file and returns a dictionary containing joint coordinates for each video.

## **Function: compute\_average\_distances**

Calculates the average distance between predicted and ground truth keypoints across all videos.

## **Function: plot\_keypoints\_on\_video**

Plots detected keypoints on the original video frames.

**Parameters:**

* video\_path: Path to the input video.
* truth\_keypoints: Actual keypoints from the ground truth.
* predicted\_keypoints: Predicted keypoints from the YOLO model.
* average\_distances: Average distances for keypoints to indicate the accuracy of predictions.

**Visualization Logic:**

1. Loads the video and initializes a video writer.
2. Iterates over each frame and each keypoint.
3. Draws keypoints and circles indicating prediction accuracy.

## **Function: plot\_keypoints\_on\_blank\_canvas**

Draws keypoints and skeletal connections on a blank canvas, creating a "stickman" representation.

**Additional Details:**

* Defines connections between joints to draw limbs.
* Uses line drawing for limbs and circle drawing for joints.

## **Execution**

After defining paths and loading keypoints, the script calls the visualization functions to generate videos showing the keypoints overplayed on the original video and on a blank canvas.

plot\_keypoints\_on\_video(video\_path, truth\_keypoints, predicted\_keypoints, average\_distances)

plot\_keypoints\_on\_blank\_canvas(video\_path, truth\_keypoints, predicted\_keypoints, average\_distances)

**Output:** Generates videos that are saved to specified directories, showcasing the pose estimation results in two different formats.