# **VideoPose3d for Image Processing and Pose Estimation Pipeline**

This documentation explains a Python script designed for video processing and pose estimation using a combination of tools including FFmpeg, VideoPose3D, and Detectron2.

## **Initial Setup**

### **FFmpeg Installation**

!pip install ffmpeg

### **Cloning VideoPose3D**

Next, it clones the VideoPose3D repository, which is a deep learning tool for 3D pose estimation from video data.

!git clone https://github.com/facebookresearch/VideoPose3D

### **Setting Up the Environment**

Directories for storing checkpoints and other outputs are created within the cloned repository.

cd VideoPose3D

mkdir checkpoint

cd checkpoint

### **Downloading Pretrained Model**

A pretrained model specific to human pose estimation is downloaded to the checkpoint directory.

!wget https://dl.fbaipublicfiles.com/video-pose-3d/pretrained\_h36m\_detectron\_coco.bin

## **FFmpeg Usage for Video and Image Preparation**

FFmpeg is used to convert images to a video with one frame for further processing.

# Resize image to even dimensions

!ffmpeg -i /content/man1.jpg -vf "scale=trunc(iw/2)\*2:trunc(ih/2)\*2" /content/resized\_man1.jpg

# Convert resized image to a video

!ffmpeg -loop 1 -i /content/resized\_man1.jpg -c:v libx264 -t 1 -pix\_fmt yuv420p /content/inputs/man1\_video.mp4

## **Detectron2 Installation**

Detectron2, Facebook AI's next generation software system for object detection and segmentation, is installed.

!python -m pip install 'git+https://github.com/facebookresearch/detectron2.git'

### **REPLACE /VideoPose3d/inference/infer\_video\_d2.py script with the code present in 1\_video\_pose\_3d(videos).ipynb.**

## **Running Detectron2 Inference**

The script moves to running inference with the detectron2 model, using 2D keypoints extracted from video frames.

cd /content/VideoPose3D/inference

!python infer\_video\_d2.py --cfg COCO-Keypoints/keypoint\_rcnn\_R\_101\_FPN\_3x.yaml --output-dir /content/output/ --image-ext mp4 /content/inputs

## **Data Preparation for Pose Estimation**

Before running the pose estimation model, it's necessary to prepare the input data. The script uses the prepare\_data\_2d\_custom.py script to parse 2D detections from detectron2 and format them for the pose estimation model.

cd /content/VideoPose3D/data

!python prepare\_data\_2d\_custom.py -i /content/output -o myvideos

**Description:**

* The script parses 2D pose detection results stored in the /content/output directory.
* It processes these detections and saves them in a format suitable for VideoPose3D, outputting to the VideoPose3d/data directory.

## **Running the Pose Estimation Model**

After preparing the data using Detectron2, you can run the pose estimation model. This involves specifying numerous parameters including the dataset type, keypoint directory, architecture configuration, and the checkpoint for the pretrained model.

cd /content/VideoPose3D

!python run.py -d custom -k myvideos -arc 3,3,3,3,3 -c checkpoint --evaluate pretrained\_h36m\_detectron\_coco.bin --render --viz-subject man1\_video.mp4 --viz-action custom --viz-camera 0 --viz-video /content/inputs/man1\_video.mp4 --viz-output /content/results/man1\_3d.mp4 --viz-size 6

**Parameters Explained:**

* -d custom: Specifies a custom dataset.
* -k myvideos: Uses keypoints from the myvideos directory.
* -arc 3,3,3,3,3: Defines the architecture of the model.
* --evaluate pretrained\_h36m\_detectron\_coco.bin: Evaluates using a pretrained model.
* --render: Enables rendering of the output.
* --viz-subject man1\_video.mp4: Specifies the subject video for visualization.
* --viz-action custom: Indicates a custom action label for visualization.
* --viz-camera 0: Uses the first camera perspective available.
* --viz-video: Specifies the path to the input video.
* --viz-output: Designates the path for the output video.
* --viz-size 6: Sets the size of the visualization.

**Execution Details:**

* Loads the custom dataset and keypoints.
* Prepares the model with the specified architecture and the pretrained weights.
* Processes the video to estimate poses, rendering the results to the specified output file.

## **Converting output back to an image**

The final step involves extracting a frame from the rendered video for visualization purposes. This is done using FFmpeg to capture the first frame.

!ffmpeg -i man1\_3d.mp4 -vf "select=eq(n\,0)" -q:v 3 /content/results/man1\_3d\_frame.jpg

**Description:**

* This command extracts the first frame from the man1\_3d.mp4 video.
* It uses the FFmpeg filter select=eq(n\,0) to specify the first frame.
* The output is saved as a JPEG image