# **VideoPose3D Pose Estimation Pipeline Documentation**

This documentation outlines the process of using the VideoPose3D framework to perform pose estimation on video files. The pipeline is designed to extract human poses from videos using a combination of Detectron2 for keypoint detection and VideoPose3D for pose reconstruction.

Link to Notebook: <https://github.com/Aadharsh1/ML-Deep-Learning/blob/main/Pose_Estimation_Experiments/VideoPose3D_Experiments/1_video_pose_3d(videos).ipynb>

## **Setup and Installation**

**Clone VideoPose3D Repository**:  
  
!git clone https://github.com/facebookresearch/VideoPose3D

cd VideoPose3D

mkdir checkpoint

**Download Pretrained Model**: Navigate to the checkpoint directory and download the pretrained model.  
  
cd checkpoint

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

cd /content

**Create Required Directories**: Set up directories for inputs, outputs, results, and coordinate storage.  
  
!mkdir inputs

!mkdir output

!mkdir results

!mkdir coordinates

**Install Detectron2**: Install Detectron2, which is used for keypoint detection.  
  
!python -m pip install 'git+https://github.com/facebookresearch/detectron2.git'

## **Pose Estimation Process**

### **Configuration and Data Preparation**

1. **Replace the Infer Script**: Modify the infer\_video\_d2.py script in the /VideoPose3D/inference directory with the provided code in the linked notebook. This script handles the processing of video files to extract keypoints using Detectron2.
2. **Prepare Input Videos**: Place all videos for pose estimation in the /content/inputs folder.

### **Running Pose Estimation**

**Run Keypoint Detection**: Execute the modified infer\_video\_d2.py script to detect keypoints and save the results in the /content/output directory.  
   
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

**Prepare 2D Data**: Convert the output from Detectron2 into a format usable by VideoPose3D.  
  
cd /content/VideoPose3D/data

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

### **Visualization and Output**

**Run VideoPose3D**: Execute the VideoPose3D model to estimate and render poses on the videos. Specify the subject, action, and output preferences.  
  
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 walk1.mp4 --viz-action custom --viz-camera 0 --viz-video /content/inputs/walk1.mp4 --viz-output /content/results/walk1\_result.mp4 --viz-size 6 --viz-export /content/coordinates/walk1\_coordinates.npy

1. **Review Results**: Check the output videos and coordinate data:
   * Videos with pose overlays are saved in /content/results.
   * Coordinate data of keypoints is saved in /content/coordinates.

### **Coordinate Analysis**

import numpy as np

coordinates = np.load('/content/coordinates/walk1\_coordinates.npy')

print(coordinates)

print(coordinates.shape)