# **Video Processing and Pose Estimation Pipeline Documentation for Pen Action Dataset with Detectron2**

This documentation describes a Python-based pipeline for extracting video frames fromthe pen action dataset, performing pose estimation, and visualizing the results using detectron2. The pipeline integrates various tools and libraries, including Google Colab, OpenCV, and custom scripts for pose estimation and visualization.

## **Installation and Setup**

1. **Google Colab Setup**:
   * Mount Google Drive to access datasets stored in Google Drive:  
     from google.colab import drive
   * drive.mount('/content/drive')
2. **Extract Dataset**:
   * Extract a .tar.gz file containing the dataset using tarfile library:  
       
     import tarfile
   * tar\_path = '/content/drive/My Drive/Penn\_Action.tar.gz'
   * extract\_to = '/content'
   * with tarfile.open(tar\_path, 'r:gz') as tar:
   * tar.extractall(path=extract\_to)
   * print(f'Extracted files to {extract\_to}')
3. **Video Frame Extraction**:
   * Convert frames stored as images into a video file using cv2.VideoWriter:  
       
     import cv2
   * import os
   * def frames\_to\_video(frames\_dir, output\_dir, video\_name, fps=30):
   * frame\_files = sorted([os.path.join(frames\_dir, f) for f in os.listdir(frames\_dir) if f.endswith('.jpg')])
   * frame = cv2.imread(frame\_files[0])
   * height, width, layers = frame.shape
   * size = (width, height)
   * video\_path = os.path.join(output\_dir, video\_name + '.mp4')
   * out = cv2.VideoWriter(video\_path, cv2.VideoWriter\_fourcc(\*'mp4v'), fps, size)
   * for filename in frame\_files:
   * img = cv2.imread(filename)
   * out.write(img)
   * out.release()
   * print(f'Video saved at {video\_path}')
4. **Load and Parse Annotation Data**:
   * Extract keypoints and bounding box data from .mat files using scipy.io.loadmat:  
       
     from scipy.io import loadmat
   * import numpy as np
   * def extract\_labels(mat\_file):
   * mat = loadmat(mat\_file)
   * keypoints = np.array([np.column\_stack((mat['x'][frame], mat['y'][frame])) for frame in range(mat['x'].shape[0])])
   * bbox = mat['bbox'] if 'bbox' in mat else None
   * labels = {'keypoints': keypoints, 'bbox': bbox}
   * return labels
5. **Pose Estimation Setup**:
   * Clone and set up a pose estimation repository, e.g., VideoPose3D, and download pretrained models:  
       
     !git clone https://github.com/facebookresearch/VideoPose3D.git
   * !cd VideoPose3D/checkpoint
   * !wget https://dl.fbaipublicfiles.com/video-pose-3d/pretrained\_h36m\_detectron\_coco.bin
6. **Inference and Data Preparation**:
   * Run pose estimation on the videos using detectron2 and prepare 2D pose data:
   * Take note to replace infer\_d2.py script with the code in the notebook. The code is also present in the linked notebook.
   * The steps for inference is the same as the other notebooks where detectron2 is being used  
       
     !python infer\_video\_d2.py --cfg COCO-Keypoints/keypoint\_rcnn\_R\_101\_FPN\_3x.yaml --output-dir /content/output/ --image-ext mp4 /content/inputs
   * !python prepare\_data\_2d\_custom.py -i /content/output -o myvideos
7. **Visualization of Keypoints**:
   * Plot keypoints and bounding boxes on video frames and a blank canvas to visualize accuracy and errors:  
       
     import numpy as np
   * def adjust\_keypoints\_multi\_frame(det\_keypoints):
   * indices = [0, 6, 5, 8, 7, 10, 9, 12 ,11, 14, 13, 16, 15]
   * adjusted\_keypoints = det\_keypoints[:, indices, :]
   * return adjusted\_keypoints
   * def plot\_keypoints\_on\_video(video\_path, truth\_keypoints, predicted\_keypoints, bboxes):
   * # Open video, read frames, draw keypoints and bounding boxes, and save output video.
   * pass
   * def plot\_keypoints\_on\_blank\_canvas(video\_path, truth\_keypoints, predicted\_keypoints, bboxes):
   * # Create a blank canvas, draw skeleton lines and keypoints, and save output video.
   * pass

## **Dependencies**

* Python 3.8+
* OpenCV
* NumPy
* SciPy
* Matplotlib
* Detectron2
* PyTorch
* VideoPose3D

Ensure all dependencies are installed and properly configured before running the pipeline.