Human Pose Estimation

Problem statement
To obtain the angle between two bones at joints in the human body in real-time from monocular/stereo videos/images.

Method
A comparison of the average precision and run-times of various methods along with links to Github repositories and the related papers can be found here.

The methods for estimating human pose can follow either a top-down or bottom-up approach. A top-down approach involves running a person detector first on images to find bounding boxes and then applying a pose-estimation model. A bottom-up approach involves finding limbs in an image and then grouping them using association scores. Due to the nature of the two approaches, it is clear that only a bottom-up approach would be useful for estimating human pose in real-time since the run-time in a top-down approach grows linearly with the number of people in the image although it's more accurate. Thus, there is a trade-off between speed and accuracy.

The current state-of-the-art model in human pose estimation is a model that uses soft-gated skip connections with an accuracy of 94.1%. However, there is no performance analysis mentioned in their paper to know how it would perform in real-time scenarios. The fastest method, however, is the Xnect model that can process thirty frames per second and doesn't compromise much on accuracy. It also seems to deal with occlusions that other methods don't seem to address.

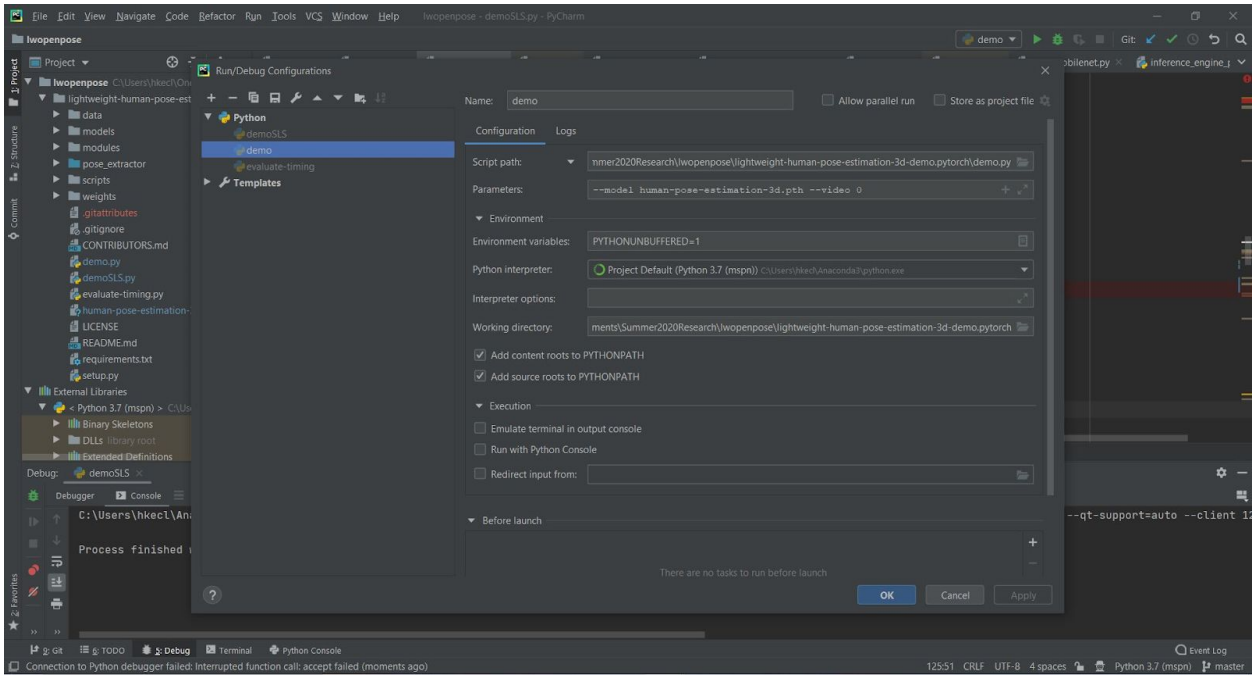
We decided to test existing projects that had a decent amount of documentation and support, rather than trying to implement one ourselves from scratch. The two projects and their setups are described below.

Lightweight OpenPose:

Please find below the steps to get the basic light-weight open pose model demo working on your laptop:

1. Fork and clone the repository: https://github.com/hvenugopalan/lwopenpose
2. Open the project in PyCharm and follow all the steps outlined in the pre-requisite section of the README through the terminal. (Ensure you have Python 3.7.x installed prior to this.) Also, download the model weights(human-pose-estimation-3d.pth) from their google drive.

3. Add a new run/debug configuration as shown in the screenshot attached.
4. On running the project, you should be able to see a window pop up with the angles at the joints showed on the left.



OpenPose 3D:

1. Depending on your operating system, follow the installation instructions described here to set up OpenPose locally in your machine. Ensure that all the dependencies and requirements are satisfied prior to this.
2. I used two different cameras - my laptop webcam and my mobile phone - to capture frames from different angles for 3D pose estimation. You can calibrate the two cameras and find their intrinsic and extrinsic parameters using the instructions described here.
3. I used DroidCam to use my mobile phone as a secondary webcam to my laptop. There are many other options you could try as well.
4. While calibrating the two cameras extrinsically, you have to pass two frames taken at approximately the same instance of time from each of the two cameras to the calibration tool. Since I couldn't get the exact timestamp of the frames captured using OpenCV, I wrote a multi-threaded program that captures and numbers the frames in the exact order that the calibration tool requires. Please find the code for the same here.
5. The command to generate the extrinsic parameters on Windows would look something like this:

   .\\build\\x64\\Release\\Calibration.exe --mode 2 --grid_square_size_mm 20.0 --grid_number_inner_corners 7x7 --omit_distortion --calibration_image_dir ./extrinsics/p_0811 --cam0 1 --cam1 0

6. Save the intrinsic parameters to .\models\cameraParameters\flir. The parameters for the two cameras should be named as frame_intrinsics.xml and frame_intrinsics_0.xml. (frame_intrinsics_<cam_id - 1>.xml).
7. Once the two cameras are calibrated, use the following command to run OpenPose and generate keypoints:

   .\build\x64\Release\OpenPoseDemo.exe --image_dir .\examples\media --3d --3d_views 2 --3d_min_views 2 --write_json .\keypoints1 --write_images .\openposeim --number_people_max 1

8. Once the keypoints are generated, you could use this program to process keypoint files and generate angles.

Observation

We found that OpenPose 3D gave more accurate results than the corresponding 2D model. There were far fewer variations in the angles generated for the same pose across multiple frames in the 3D case as well.

Resources:

- Human Pose Estimation 101: https://github.com/cbsudux/Human-Pose-Estimation-101
- 2D Human Pose Estimation: https://nanonets.com/blog/human-pose-estimation-2d-guide/?utm_source=github&utm_medium=social&utm_campaign=pose&utm_content=cbsudux
- 3D Human Pose Estimation: https://nanonets.com/blog/human-pose-estimation-3d-guide/