Using OpenPose for Education Research Pose Analyses

Overview of OpenPose

OpenPose is a real-time multi-person system to jointly detect a human body, hand, facial, and foot keypoints on single images. Videos are a series of single images called frames, so OpenPose detects individuals' keypoints in each frame separately. It was developed by the Perceptual Computing Lab of Carnegie Mellon University, and this tool has widely gained prominence in research and industry for its accuracy and capability to analyze visual data. OpenPose works by using a model trained on the COCO dataset to predict body poses, which can then be processed further for specific applications. In our case, we will be using OpenPose to process our videos to extract keypoints and save them as an output, then use a Python script in order to process the output and assign the keypoints to individuals in the video.

Overview of the Tracking Process

We devised a post-processing approach to further extend the usefulness of OpenPose output data, especially for education research, by tracking the data of the individuals between frames in order to be able to assign unique IDs to the individuals detected by the software. We were motivated to explore the effectiveness of OpenPose in analyzing classroom videos. Our approach calculates the pixel distances between individual's body keypoint data between frames in order to assign the data to the individuals across time. However, we noticed some difficulties with this approach, as the tracking process relies on consistently detected individuals and keypoint data. The inconsistencies of person detection manifested challenges, which were primarily caused by low camera angles causing occlusion, difficulty in tracking lower body movements due to seated students, and close proximities between individuals. For individuals who were detected, however, our approach was largely accurate and was checked by manual verification. Further details of the tracking approach was published in a conference paper. The post-processing scripts can be found on the GitHub page here.

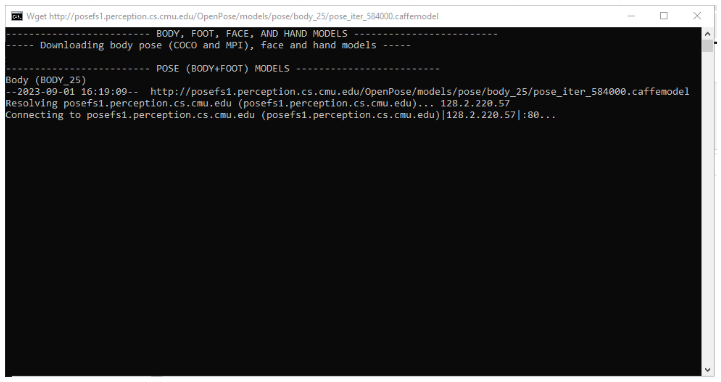
Part I: OpenPose Video Processing

Notes:

- *It is recommended that you run OpenPose on a Windows computer with a discrete GPU (e.g., an Nvidia GPU using CUDA), as some of the more detailed pose configuration options (e.g., 135-keypoint models) can be computationally demanding.*
- *These instructions on this document are a way to help you get started in order to obtain output in the one manner. More information can be found on the OpenPose Github repository.*

Instructions:

1. Downloading and extracting the software
    a. Visit the release page and download the latest v1.7.0 OpenPose zip file here, called *openpose-1.7.0-binaries-win64-gpu-python3.7-flir-3d_recommended.zip*
    b. Extract the downloaded zip file to a directory where you can easily access it, such as in a folder in the Documents directory, it should unzip a folder called "openpose-1.7.0-binaries-win64-gpu-python3.7-flir-3d_recommended", and navigate inside the folder to find another folder called "openpose"
2. Downloading the models
    a. Within the "openpose" folder, navigate inside the "models" folder (i.e., the path to this should be "openpose-1.7.0-binaries-win64-gpu-python3.7-flir-3d_recommended\openpose\models"), and double-click on the *getBaseModels* batch file, which will open a window which looks like the image below, and wait for it to complete the downloads.

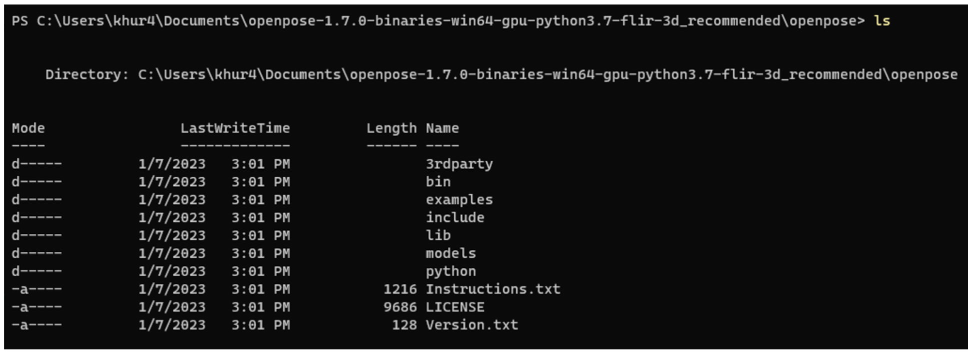


```
Wget http://posefs1.perception.cs.cmu.edu/OpenPose/models/pose/body_25/pose_iter_584000.caffemodel
------------------------- BODY, FOOT, FACE, AND HAND MODELS -------------------------
----- Downloading body pose (COCO and MPI), face and hand models -----

------------------------- POSE (BODY+FOOT) MODELS -------------------------
Body (BODY_25)
--2023-09-01 16:19:09--  http://posefs1.perception.cs.cmu.edu/OpenPose/models/pose/body_25/pose_iter_584000.caffemodel
Resolving posefs1.perception.cs.cmu.edu (posefs1.perception.cs.cmu.edu)... 128.2.220.57
Connecting to posefs1.perception.cs.cmu.edu (posefs1.perception.cs.cmu.edu)|128.2.220.57|:80...
```

    b. Optionally, clicking the *getCOCO_and_MPII_optional* batch file will download additional models
3. Running OpenPose on a demo video and saving OpenPose output as JSON format
    a. We will use the command-line interface (e.g., Powershell) to run OpenPose. OpenPose needs to be run from the folder where it can locate the "models" folder from Step 2.
    b. Using Powershell, change the directory location to when it can directly access the "models" folder. If you are not familiar with using command-line navigation, this may be a helpful resource. Using "pwd" (i.e., command to display current path) and "ls" (i.e., command to list all items in the current directory), along with "cd XYZ" (i.e., command to change directory to XYZ–for example), navigate to the

location where “ls” displays what is displayed below.



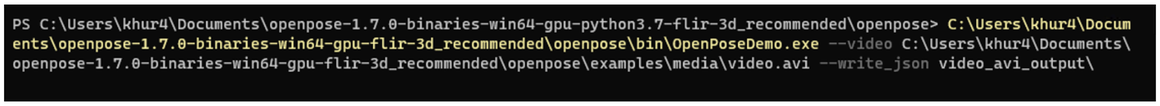
```
PS C:\Users\khur4\Documents\openpose-1.7.0-binaries-win64-gpu-python3.7-flir-3d_recommended\openpose> ls


    Directory: C:\Users\khur4\Documents\openpose-1.7.0-binaries-win64-gpu-python3.7-flir-3d_recommended\openpose


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          1/7/2023   3:01 PM                3rdparty
d-----          1/7/2023   3:01 PM                bin
d-----          1/7/2023   3:01 PM                examples
d-----          1/7/2023   3:01 PM                include
d-----          1/7/2023   3:01 PM                lib
d-----          1/7/2023   3:01 PM                models
d-----          1/7/2023   3:01 PM                python
-a----          1/7/2023   3:01 PM           1216 Instructions.txt
-a----          1/7/2023   3:01 PM           9686 LICENSE
-a----          1/7/2023   3:01 PM            128 Version.txt
```

c. From this location, we will run the *OpenPoseDemo.exe* file inside the “bin” folder to process the *video.avi* demo file found inside the “examples/media” folder. The syntax for this in the command looks like this on my machine:

   C:\Users\khur4\Documents\openpose-1.7.0-binaries-win64-gpu-flir-3d_recommended\openpose\bin\OpenPoseDemo.exe --video C:\Users\khur4\Documents\openpose-1.7.0-binaries-win64-gpu-flir-3d_recommended\openpose\examples\media\video.avi --write_json video_output/



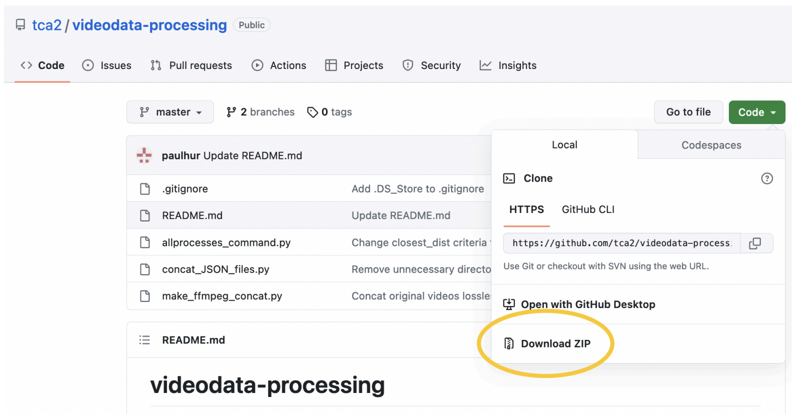
```
PS C:\Users\khur4\Documents\openpose-1.7.0-binaries-win64-gpu-python3.7-flir-3d_recommended\openpose> C:\Users\khur4\Docum
ents\openpose-1.7.0-binaries-win64-gpu-flir-3d_recommended\openpose\bin\OpenPoseDemo.exe --video C:\Users\khur4\Documents\
openpose-1.7.0-binaries-win64-gpu-flir-3d_recommended\openpose\examples\media\video.avi --write_json video_avi_output\
```

d. Note that the “--video” part of the command precedes the path to the “video.avi” file, and that “--write_json” part of the command precedes the path to the directory to which the JSON files will be saved.

Part II: OpenPose JSON Output Tracking

Notes:

- *Before starting, either download the ZIP file (as circled below), or clone the repository to your local machine.*



paulhur Update README.md

.gitignore	Add .DS_Store to .gitignore
README.md	Update README.md
allprocesses_command.py	Change closest_dist criteria
concat_JSON_files.py	Remove unnecessary directo
make_ffmpeg_concat.py	Concat original videos lossle

https://github.com/tca2/videodata-process

Use Git or checkout with SVN using the web URL.

README.md

videodata-processing

Instructions:

1. Concatenating OpenPose JSON output
    a. Once OpenPose has completed processing the video and saved the JSON files, you may notice that there are thousands of files inside the output folder. This is because OpenPose saves data from each frame of video in individual files, and they need to be concatenated (i.e., merged together into one file) before tracking. From the ZIP file, after extraction, run *concat_JSON_files.py* file from Powershell (i.e., “python concat_JSON_files.py”, you may have to locate the extracted files first via navigating the command-line as we did before). It will ask the user for the directory which holds JSON output, and then save the concatenated CSV file (e.g., *FILENAME.csv*) to the directory above the user-defined directory.
2. Tracking the concatenated CSV file
    a. The *allprocesses_command.py* tracking script within the extracted ZIP has several useful arguments for handling the tracking process. It uses Python argparse as the command-line interface. Note that currently, the "--only_track" argument must be used, as OpenPose processing code is not yet integrated.
    b. A sample command to use the *allprocesses_command.py* tracking script on the now concatenated *video_concat.csv* would be:

    ```
    python allprocesses_command.py –only_track --file video_concat.csv
    ```

    c. Optionally, you can also use the “--region” flag to assign 4 values (X_LEFT Y_TOP X_RIGHT Y_BOTTOM) of the region that you want to limit the tracking process. Thus, if you wanted to track the default demo video to a particular region, you may do something similar to:

    ```
    python allprocesses_command.py –only_track --file video_concat.csv --region 250 20 300 200
    ```