**OpenPose Documentation HCR-Lab**

OpenPose: represented the first real-time multi-person system to jointly detect human body, hand, facial, and foot keypoints (in total 135 keypoints) on single images. (<https://github.com/CMU-Perceptual-Computing-Lab/openpose>). Here at HCR Lab we are using OpenPose to analyse video data of human walking.

1. Setup and Installation
2. Processing video data with OpenPose

1.Setup and Installation

* Operating system: Ubuntu 20.04
* Install CMake GUI: $ sudo apt-get install cmake-qt-gui
* Nvidia GPU version prerequisites: CUDA 11.1.1 (cuDNN 8.1.0) for Ubuntu 20
* $ sudo bash ./scripts/ubuntu/install\_cuda.sh
* Download cuDNN Library for Linux (x86\_64) and extract zip in ‘/usr/local/cuda-{version}/’ directory.
* Install Caffe, OpenCV, and Caffe prerequisites:
* $ sudo apt-get install libopencv-dev
* $ sudo bash ./scripts/ubuntu/install\_deps.sh
* # Python 3 (default and recommended)
* $ sudo apt-get install python3-dev
* $ sudo pip3 install numpy opencv-python
* $ sudo pip install numpy opencv-python

### Clone OpenPose:

$ git clone https://github.com/CMU-Perceptual-Computing-Lab/openpose

$ cd openpose/

$ git submodule update --init --recursive --remote

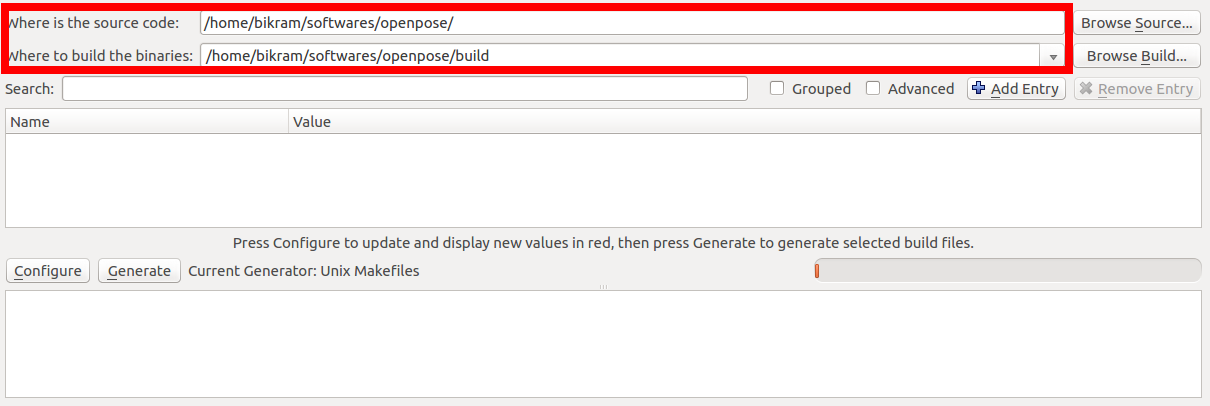
### CMake Configuration:

$ cd {OpenPose\_folder}

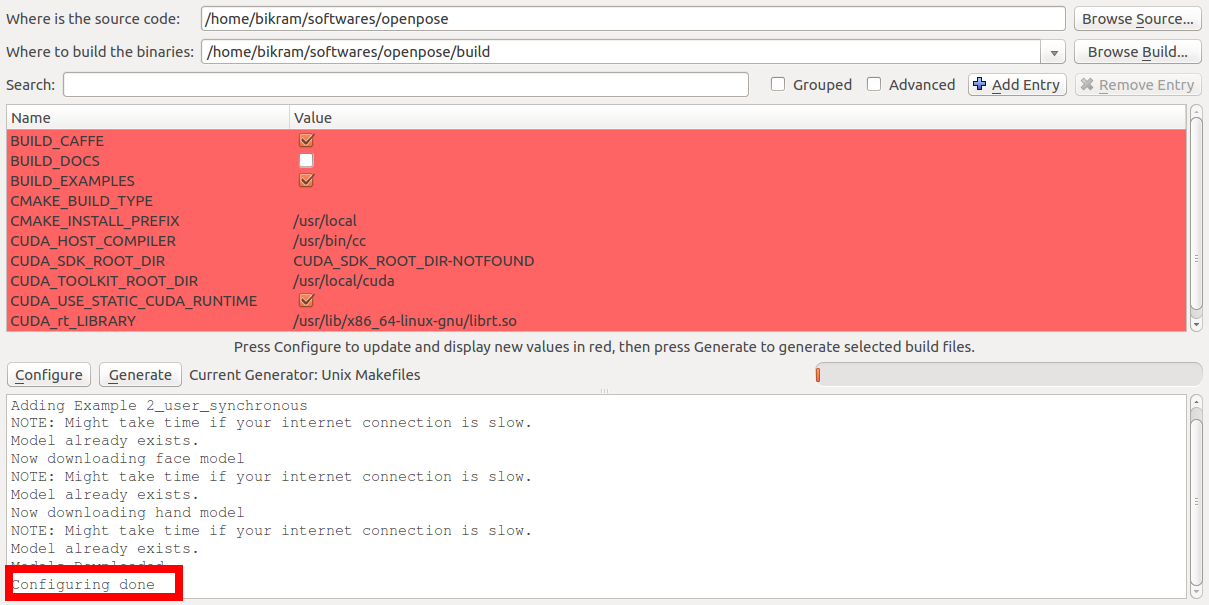
$ mkdir build/

$ cd build/

$ cmake-gui ..



* Select Configure button, generator -> Unix Makefiles and x64 platform and use default native compiler.
* Enable the BUILD\_PYTHON flag



* Press the Generate button and proceed tocompilation and now close CMake.

Compilation:

* $ cd build/
* $ make -j`nproc`

2. Processing video data OpenPose:

* Processing video file and saving output Skeleton video
* $ ./build/examples/openpose/openpose.bin --video /path-to-video-file/file\_name.avi --write\_video /output-path/result\_file\_name.avi

## JSON, Images

* $ ./build/examples/openpose/openpose.bin --video /video\_file\_path/video.avi --write\_json output\_jsons\_dir/
* $ ./build/examples/openpose/openpose.bin --video /video-file-path/video.avi --write\_images output\_images\_path/

## Only Skeleton without Background Image:

* $ ./build/examples/openpose/openpose.bin --video examples/media/video.avi --disable\_blending

### Maximum Accuracy Configuration

* $ ./build/examples/openpose/openpose.bin --net\_resolution "1312x736" --scale\_number 4 --scale\_gap 0.25
* Advance flags:
* <https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/advanced/demo_advanced.md>

**What parameters can be extracted from this model?**

There are different approaches available to extract joint features from body skeleton model

1. Using more descriptive joints
2. By computing additional features

* Relative angles and distances based on start skeleton
* Joint Movement Volume Features
* Histogram of Joint Position differences
* Covariance of 3D Joints
* Histogram of Oriented Displacements

1. As per our reviewed paper (<https://doi.org/10.1101/2020.07.24.218776>)

* <https://github.com/janstenum/GaitAnalysis-PoseEstimation> provides matlab source code to extract joint angles and gaitParameters.

#It is applicable in real time but it requires a High Performance Computing system to work smoothly.

#It is a pre-trained model by Carnegie Mellon University researchers, they used CMU Panoptic Studio dataset to train the model. At the algorithmic level parameter tuning is not possible.

* It produces a .mat file which provides:

(Use command >> load('/path\_of\_file/file\_name.mat'))

* **‘data\_openpose’** contains data matrices ‘raw\_data’ (raw input from JSON files), ‘corrected\_data’ (data with left-right ID errors corrected), ‘gapFill\_data’ (gap filled data) and ‘filt\_data’ (filtered data). Data matrices are organized as: time frames X keypoints (see list below) X coordinates (1, horizontal; 2, vertical). The struct ‘scaling’ contains the scaling factor used to dimensionalize pixel coordinates of OpenPose keypoints.
* **‘events\_openpose’** contains the data frames of heel-strike and toe-off gait events.
* ‘**videoInfo**’ contains MATLAB VideoReader variables used to display still frames from original video recording and labelled video recording.
* **‘frameInfo’** contains logical vectors and matrices that indicate which frames have been corrected, switched and gap filled.
* ‘**gait parameters**’ contains step times, stance times, swing times, double support times and step lengths for all steps and the average gait speed.
* **‘jointAngles’** contains sagittal plane hip, knee and ankle angles

**Limitations:**

* Side camera view gives more accuracy instead of front camera recording.
* Camera distance from walking person effects output data.
* At some points output value drops suddenly in matlab analysis, at that point we need to manually correct those values
* It require high performance computing system