Dance Game Engine

Arthur Murray Dance Studio ~~~~~~~~~~~~~~~~~ Our Blog ~~~~~~~~~~~~~~~ Our Website

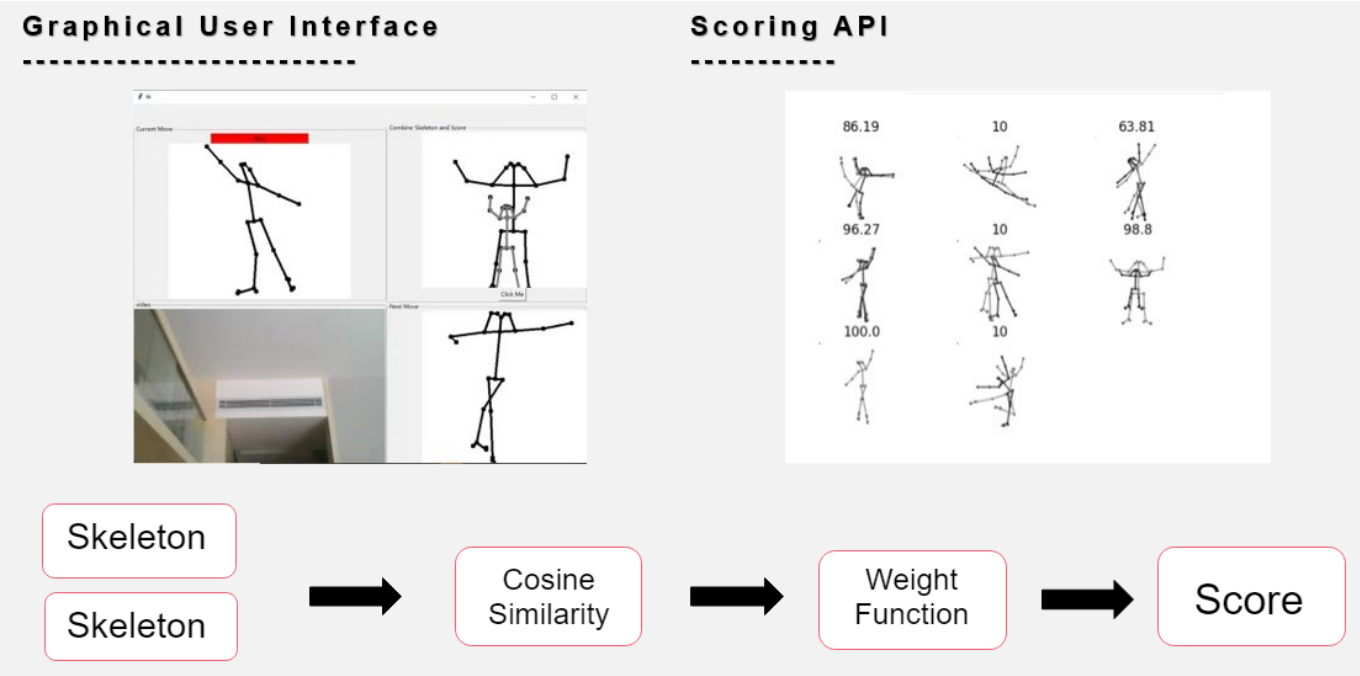
Abstract

Aimed at helping inexperienced dancers to learn basic dancing moves and gain interests in dancing, we worked with Arthur Murray Dance Studio to develop a video game engine. We provide a graphical frontend that prompts users to follow the example moves shown on screen, and it gives feedback by querying the set of APIs developed by our team, which extract skeletal information from images and evaluate the similarity between the example and that of the users. We performed pose estimation with the OpenPose system developed by CMU. The similarity score between two skeletons is calculated using a heuristically-determined one-variable function with the skeletons' cosine similarity as input. Both the graphical frontend and the set of APIs are designed to be modular and extendible, providing large potential for enhancement and extension of the game itself, and to other fields where it may fit.

Key Features

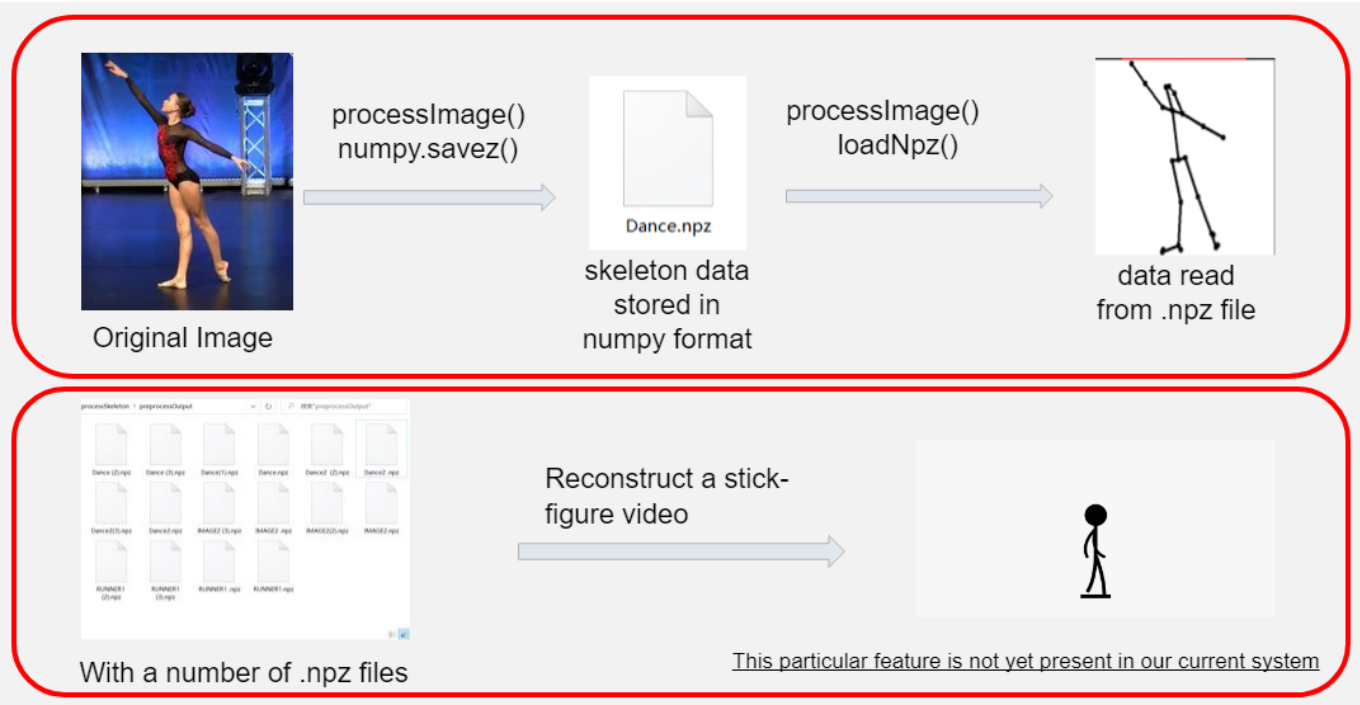
Graphical User Interface

We provide a graphical frontend that prompts users to imitate the example moves shown on the screen. The set of APIs are called under the hood of a gaming GUI, to evaluate the users' postures against the examples, giving a similarity score as output. Both the GUI and API layers serve as interfaces to users, which could be dancers or developers, depending on the context for which they like to use the system. The similarity score between two skeletons is calculated using a heuristically determined one-variable function with the skeletons' cosine similarity as input. We show the similarity of 10 sets of images. We also show the same evaluation routine embedded within the GUI.



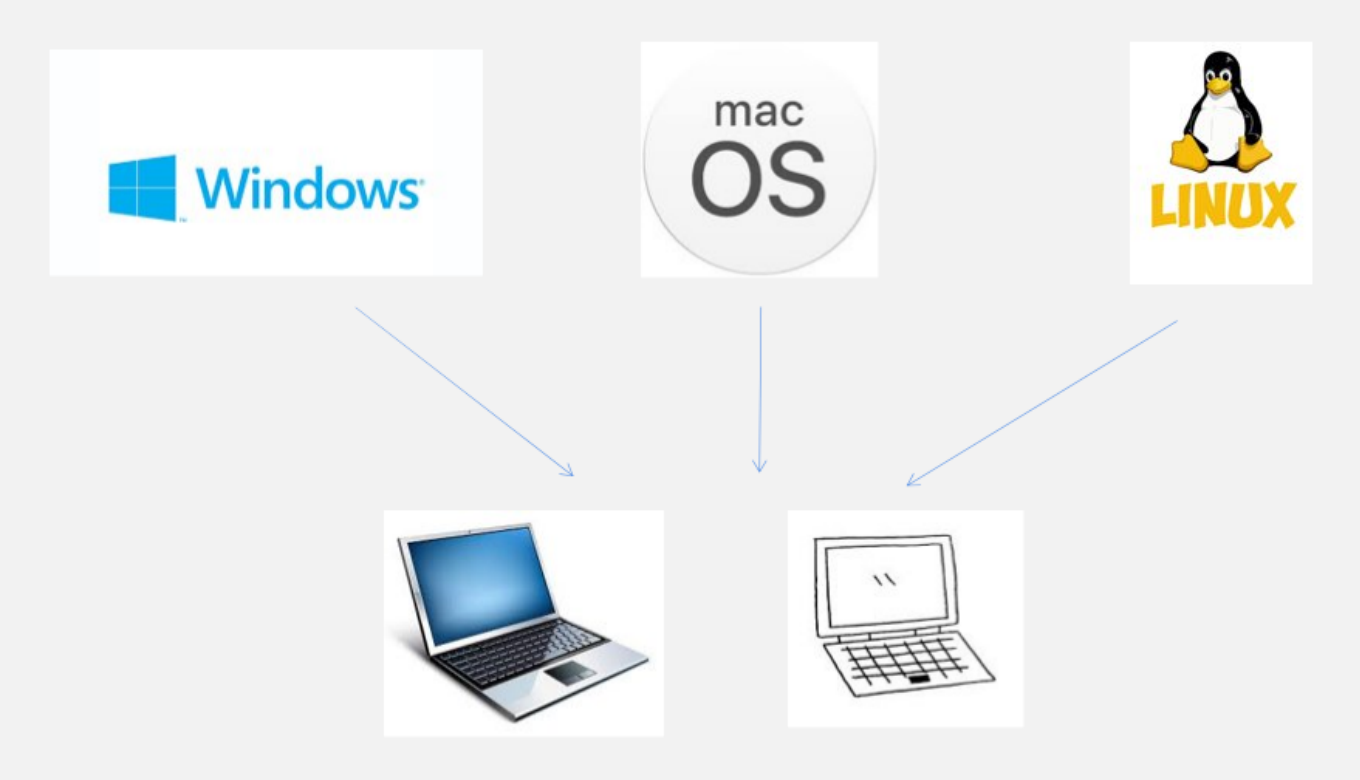
Preprocessing Image

We support a routine to preprocess images, save the skeletons locally, and set the basis to reconstruct example dance videos into cleaner, more polished, and more flexible stick-figure based forms.



Multi Platform

The system is deployable across MacOS, Windows, and Linux. It only requires a PC equipped with webcams, as found in ordinary households; no other additional hardware is required.



Installation Guide

Python Package Dependencies

Install packages for python:

Numpy, matplotlib, opencv

```
pip install numpy

pip install matplotlib

pip install opencv-python
```

For MacOS Users

Create parent directory

Create directory at desired location, name it "danceGame"

```
mkdir danceGame
```

Clone team 13 repository under danceGame directory

```
cd danceGame

git clone https://github.com/UCLComputerScience/COMP0016_2020_21_Team13
```

Install openpose

Official installation instructions:

Clone openpose repository under danceGame directory

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

Install prerequisites

cd into the openpose folder:

```
cd openpose
```

Mac OS Prerequisites

1. If you don't have `brew`, install it by running `bash scripts/osx/install_brew.sh` on your terminal.
2. Install **CMake GUI**: Run the command `brew cask install cmake`.
3. Install **Caffe, OpenCV, and Caffe prerequisites**: Run `bash scripts/osx/install_deps.sh`.

Note: As for now, cmake has to be installed using brew install --cask cmake

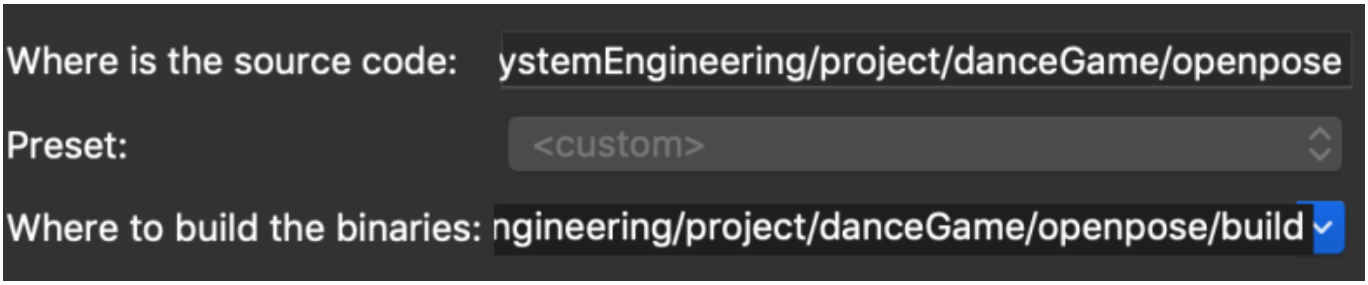
Build with Cmake

```
cd {openpose_folder}
```

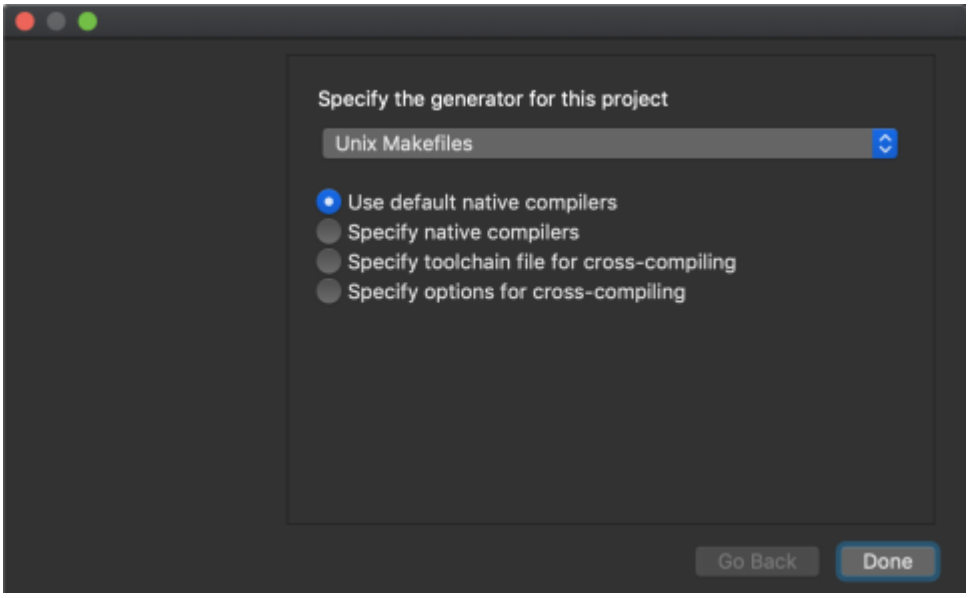
(Note: this may be unnecessary if you are still under the same folder as from section 3.2)

```
mkdir build/

cd build/

cmake-gui ..
```

Verify the location of installation.



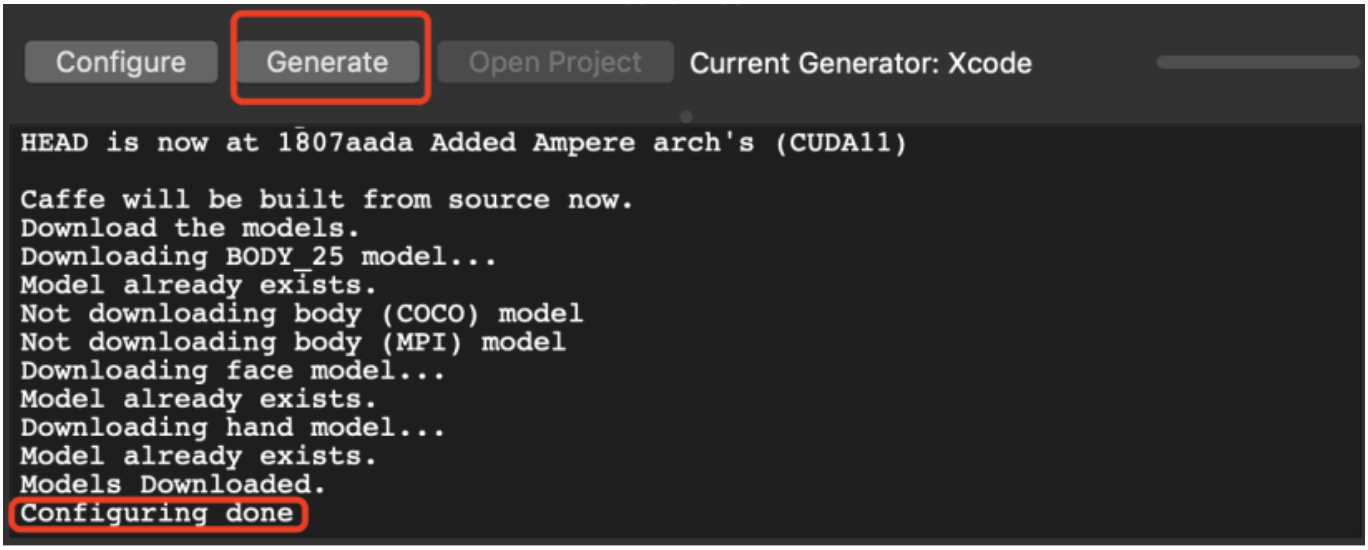
Press configure, use Xcode as generator.



Enable BUILD_PYTHON and configure again.

Set GPU_MODE and configure again.

If Configuring done appears, then press Generate.



Generating done will appear below, and you can close CMake.

Compilation

```
cd build/
```

(Note: this may be unnecessary if you are still under the same folder as from section 3.3)

```
make -j`sysctl -n hw.logicalcpu`
```

Note: I faced the problem Could NOT find vecLib (missing: vecLib_INCLUDE_DIR), and I had to manually install Caffe.

If the default compilation fails with Caffe errors, install Caffe separately and set `BUILD_CAFFE` to false in the CMake config. Steps:

- Re-create the build folder: `rm -rf build; mkdir build; cd build`.
- `brew uninstall caffe` to remove the version of Caffe previously installed via cmake.
- `brew install caffe` to install Caffe separately.
- Run `cmake-gui` and make the following adjustments to the cmake config:
  i. `BUILD_CAFFE` set to false.
  ii. `Caffe_INCLUDE_DIRS` set to `/usr/local/include/caffe`.
  iii. `Caffe_LIBS` set to `/usr/local/lib/libcaffe.dylib`.
  iv. Run `Configure` and `Generate` from CMake GUI.

Running openpose

Running the openpose executable:

```
cd openpose/
./build/examples/openpose/openpose.bin --image_dir examples/media/
```

Running the python tutorials

```
cd openpose/build/examples/tutorial_api_python

python3 01_body_from_image.py
```

Note: I notice that cmake has problem finding python that are installed by brew, and running python3 01_body_from_image.py will result in an error:

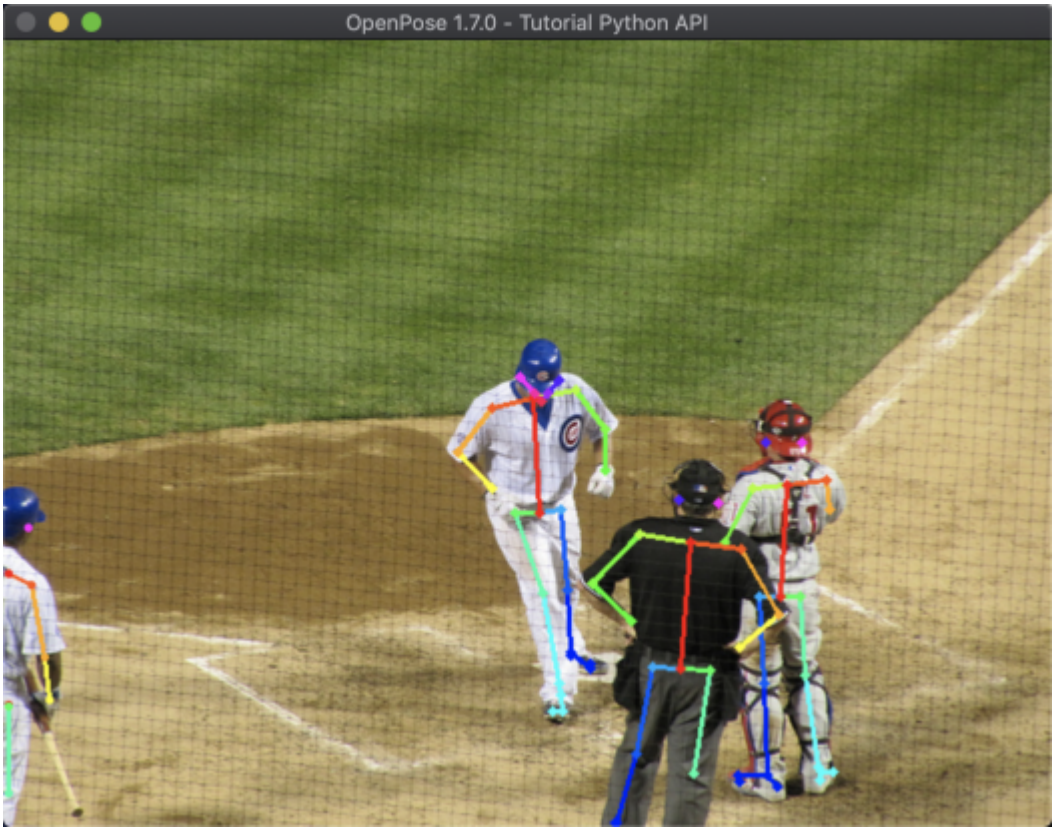
```
Error: OpenPose library could not be found. Did you enable `BUILD_PYTHON` in CMa
ke and have this Python script in the right folder?
dlopen(../../python/openpose/pyopenpose.so, 2): Symbol not found: __PyThreadStat
e_Current
  Referenced from: ../../python/openpose/pyopenpose.so
  Expected in: flat namespace
 in ../../python/openpose/pyopenpose.so
```

Adding the following lines in CmakeLists.txt(directly under the openpose folder) will solve the problem:

```
set(Python_ADDITIONAL_VERSIONS 3.9)

find_package(PythonLibs 3 REQUIRED)
```

```
### VERSION INFO
set(OpenPose_VERSION_MAJOR 1)
set(OpenPose_VERSION_MINOR 7)
set(OpenPose_VERSION_PATCH 0)
set(OpenPose_VERSION ${OpenPose_VERSION_MAJOR}.${OpenPose_VERSION_MINOR}.$
{OpenPose_VERSION_PATCH})

set(Python_ADDITIONAL_VERSIONS 3.9)
find_package(PythonLibs 3 REQUIRED)

### OS-DEPENDENT FLAGS
set(CMAKE_MACOSX_RPATH 1)
```

The expected output:



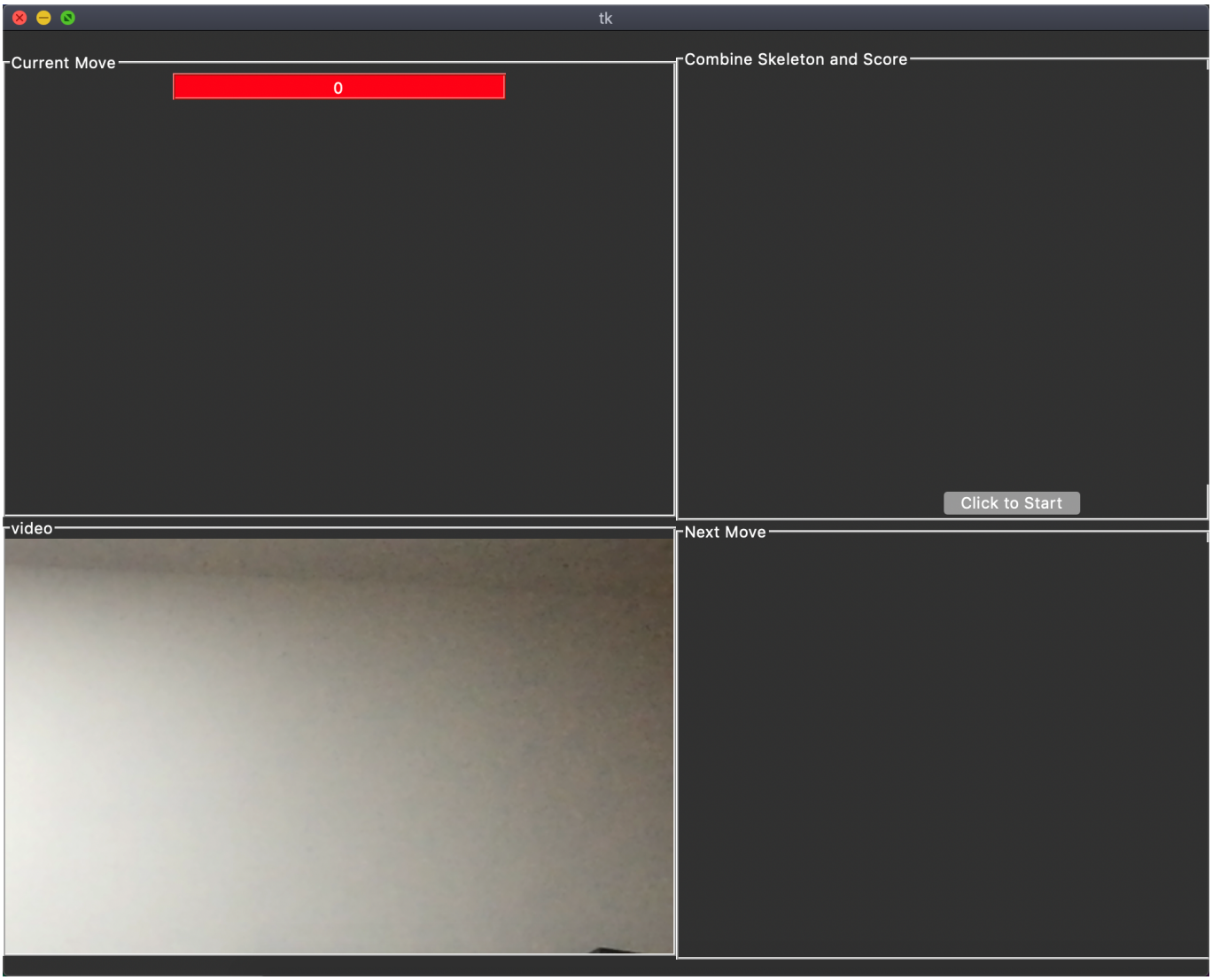
Running the game

```
cd danceGame/COMP0016_2020_21_Team13/
```

(parent folder created in section 1)

```
python3 gameInterface/mainGUI.py
```

Note: make sure any program you run is from the COMP0016_2020_21_Team13 as the working directory. The expected output:



For Windows 10 Users

Clone team 13 repositories

```
git clone [https://github.com/UCLComputerScience/COMP0016_2020_21_Team13]
(https://github.com/UCLComputerScience/COMP0016_2020_21_Team13)
```

(parent directory path should not be too long as windows maximum path length is 256)

Install OpenPose

Official installation instructions: Tools:

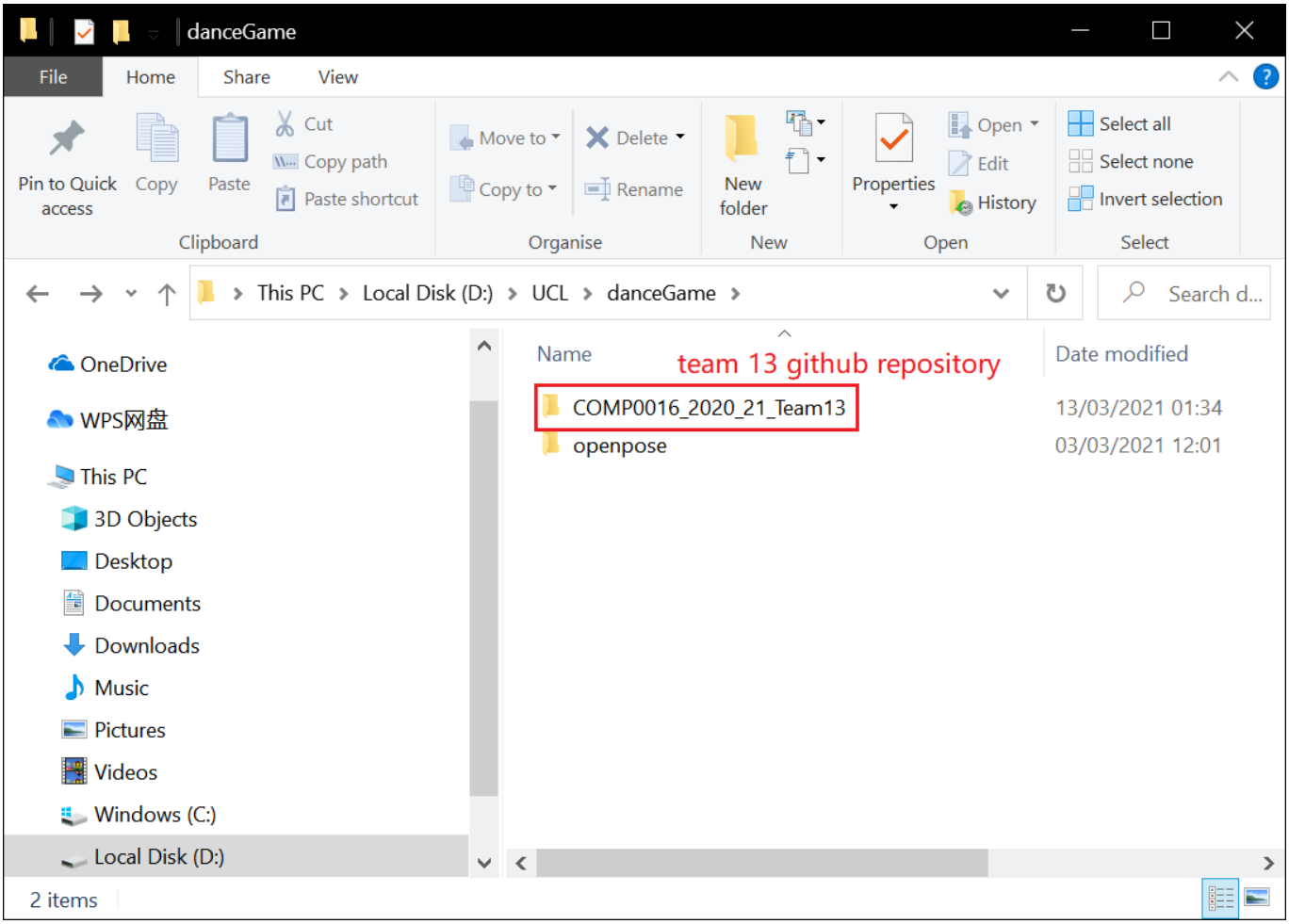
Python version 3.7

Visual Studio 2019 Community

CMake 3.20.0

Clone OpenPose repository below team 13 github repository

```
git clone [https://github.com/CMU-Perceptual-Computing-Lab/openpose]
(https://github.com/CMU-Perceptual-Computing-Lab/openpose)
```



Prerequisites

Openpose Official Prerequisites

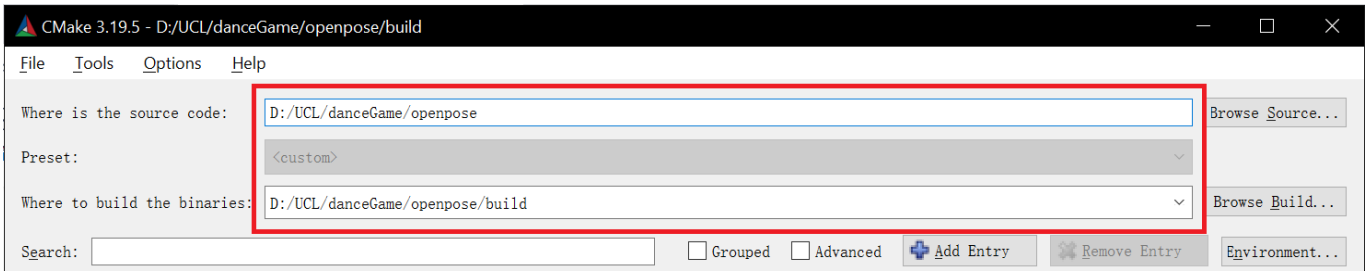
Install CMake GUI 3.20.0

- CMake automatically downloads all the Windows DLLs. Alternatively, you might prefer to download them manually:
  - Dependencies:
    - Note: Leave the zip files in `3rdparty/windows/` so that CMake does not try to download them again.
    - Caffe (if you are not sure which one you need, donwload the default one):
      - CUDA Caffe (Default): Unzip as `3rdparty/windows/caffe/`.
      - CPU Caffe: Unzip as `3rdparty/windows/caffe_cpu/`.
      - OpenCL Caffe: Unzip as `3rdparty/windows/caffe_opencl/`.
    - Caffe dependencies: Unzip as `3rdparty/windows/caffe3rdparty/`.
    - OpenCV 4.2.0: Unzip as `3rdparty/windows/opencv/`.

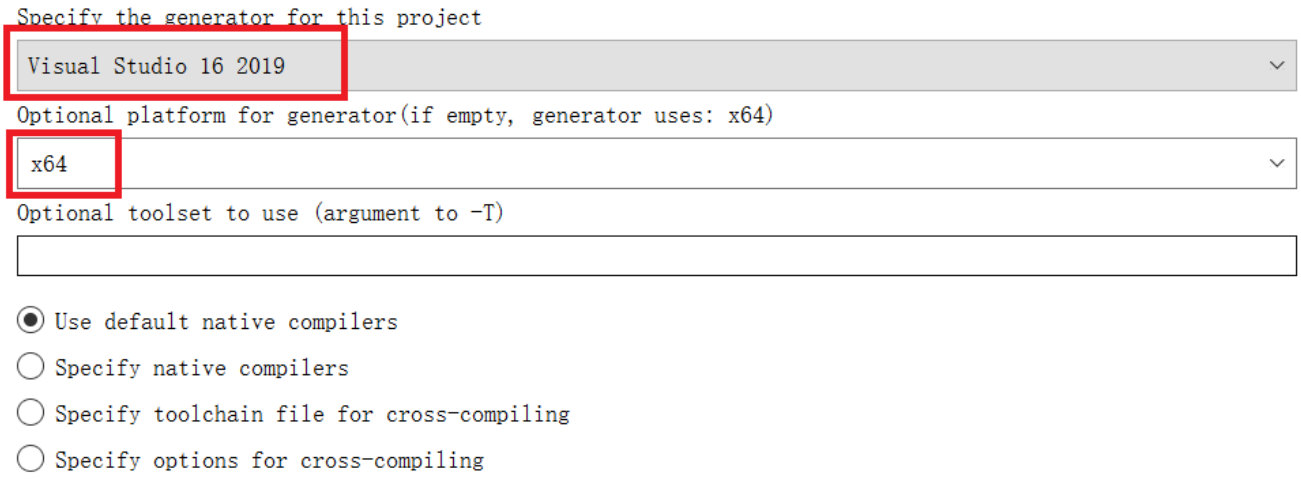
Clone pybind11 in the openpose/3rdparty directory mentioned above.

```
cd openpose/3rdparty
git clone https://github.com/pybind/pybind11
```

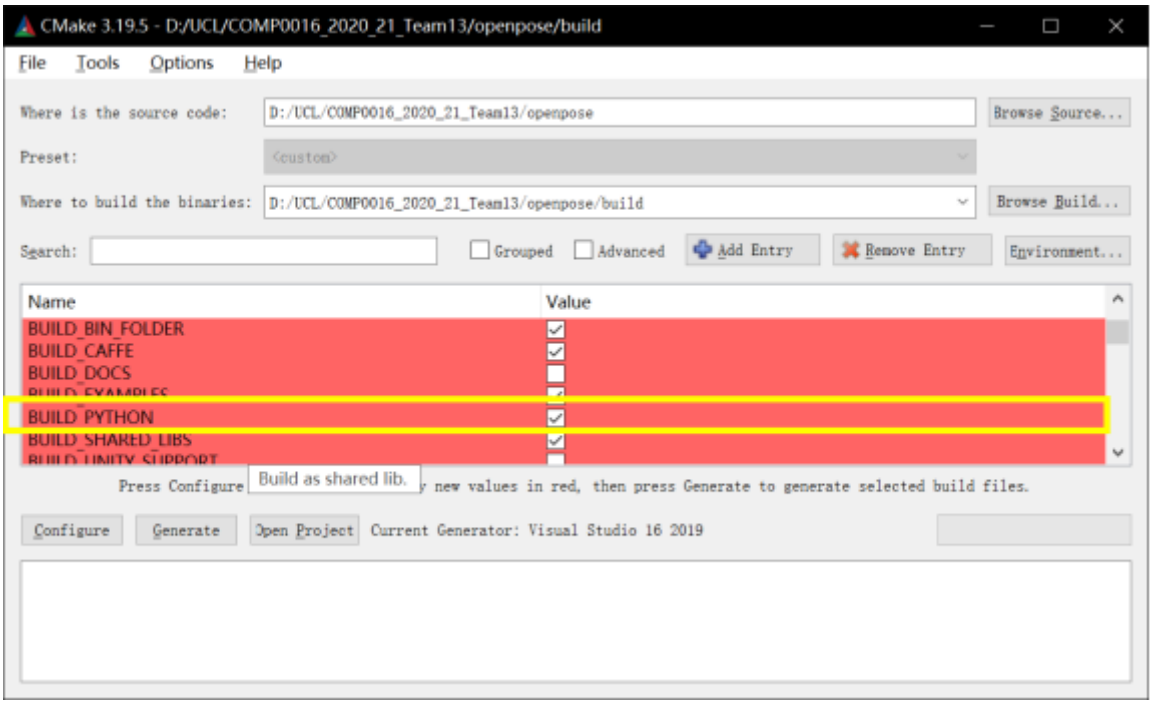
Cmake Configuration

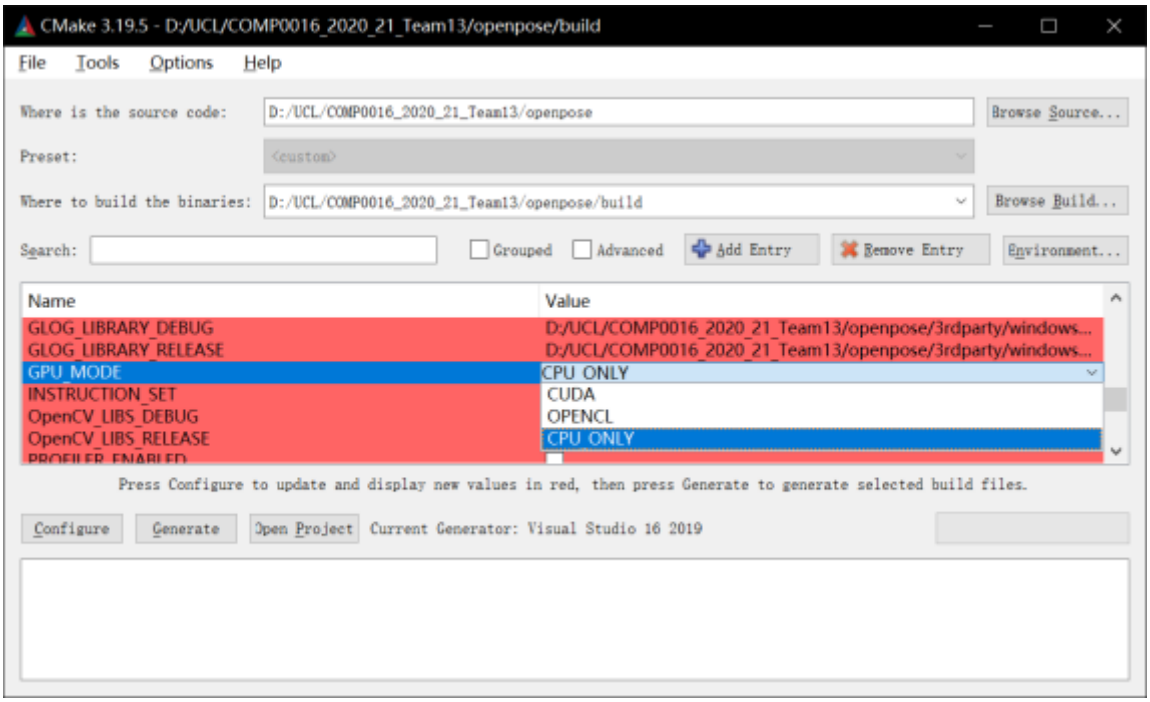


Press configure, use Visual Studio 16 2019 as generator.



Click on build python and cpu only for cpu mode:



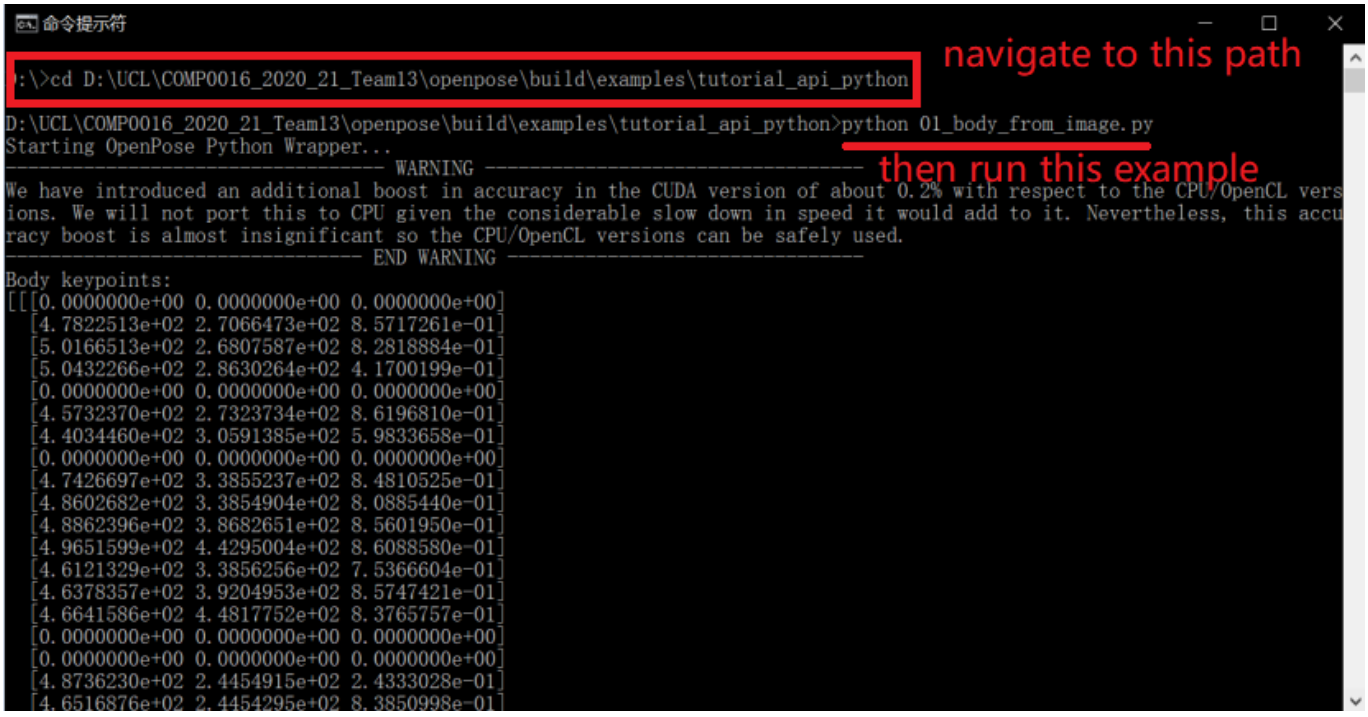


Press "Configure" and then "Generate"

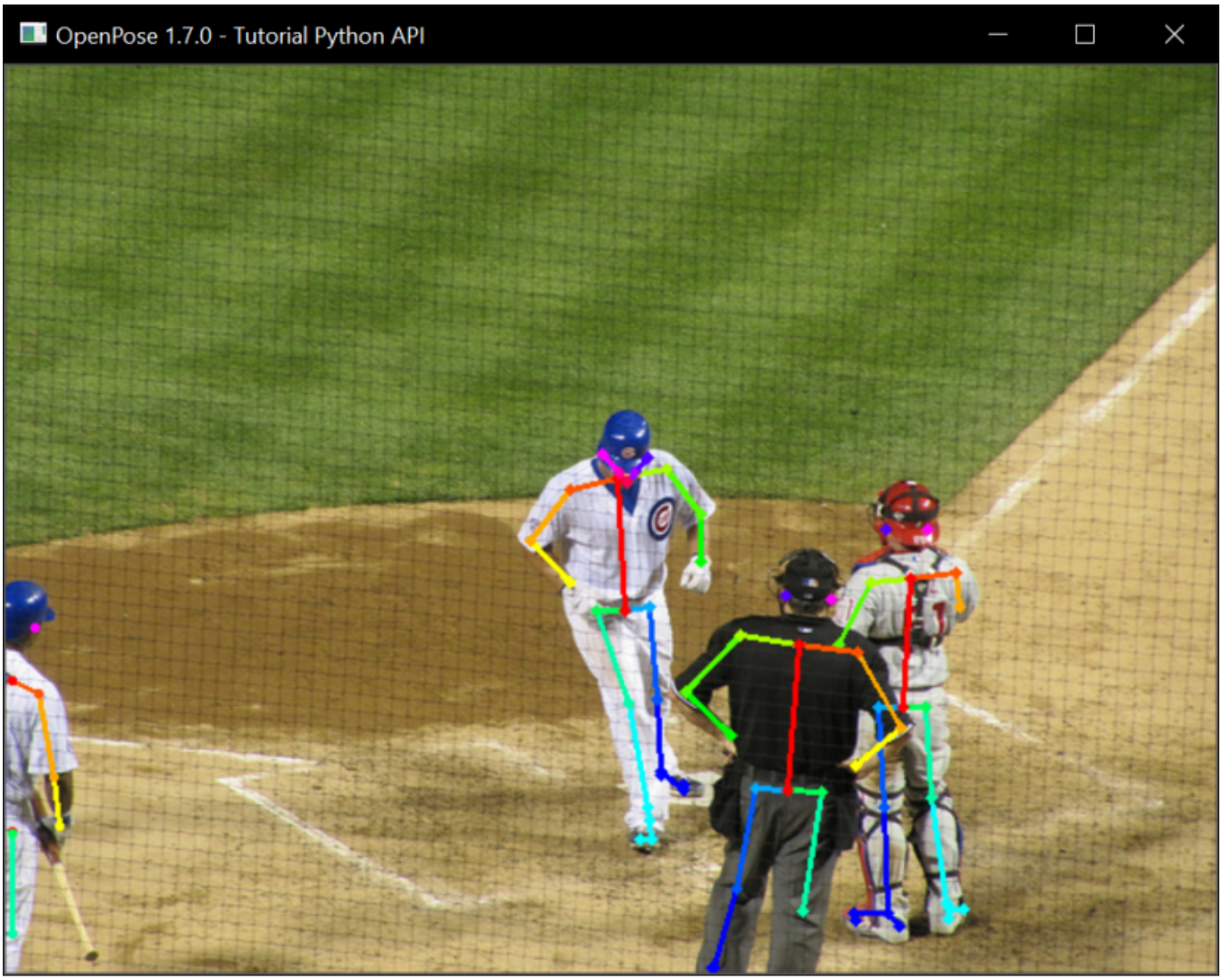
Run Openpose

Check OpenPose was properly installed by running any demo example: 01_body_from_image.py.

Open terminal, navigate to the path tutorial_api_python and then run 01_body_from_image.py. (or run it through visual studio code)



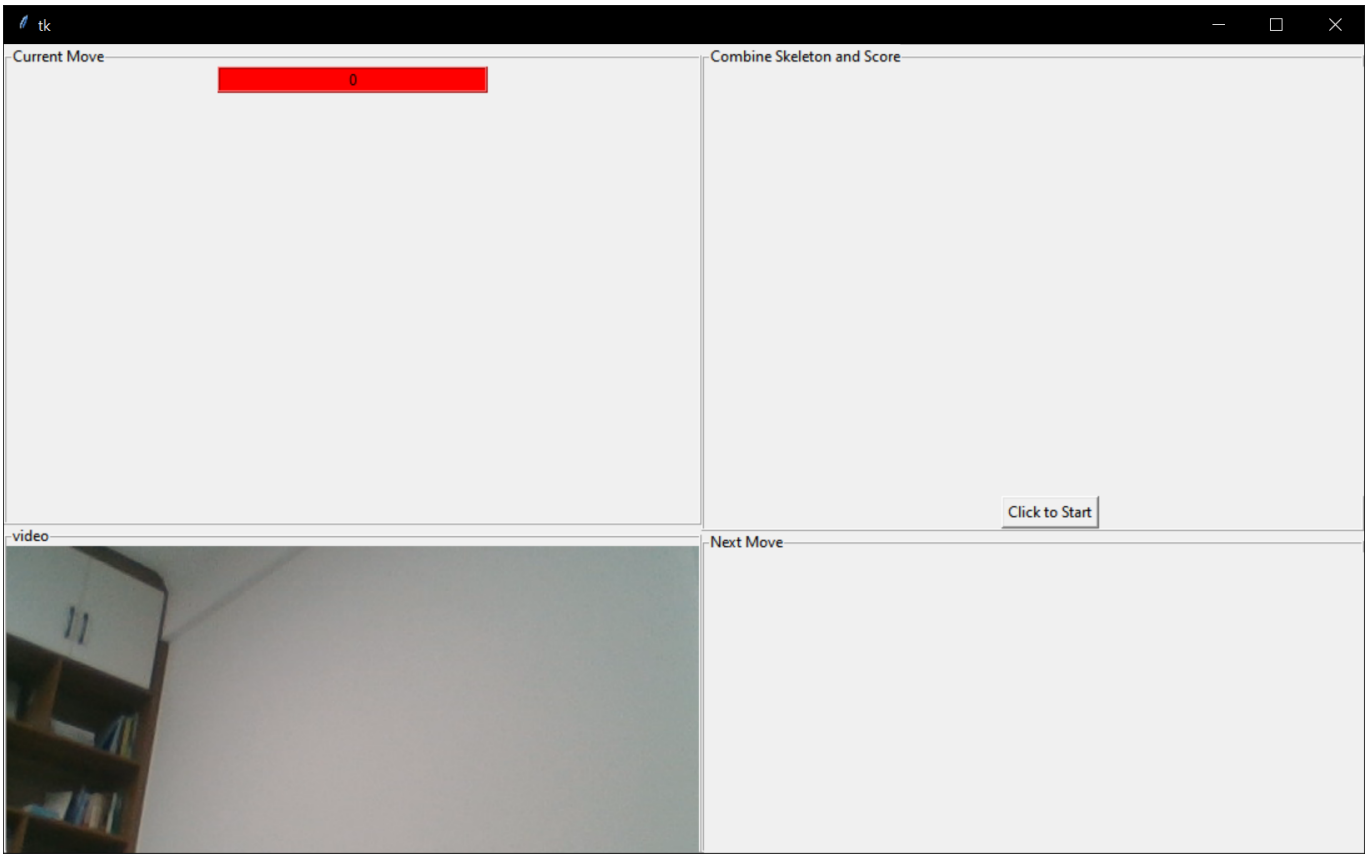
If openpose is properly installed, the following picture will pop out.



Run the Game

```
cd danceGame/COMP0016_2020_21_Team13/

python3 gameInterface/mainGUI.py
```

Expected Output:



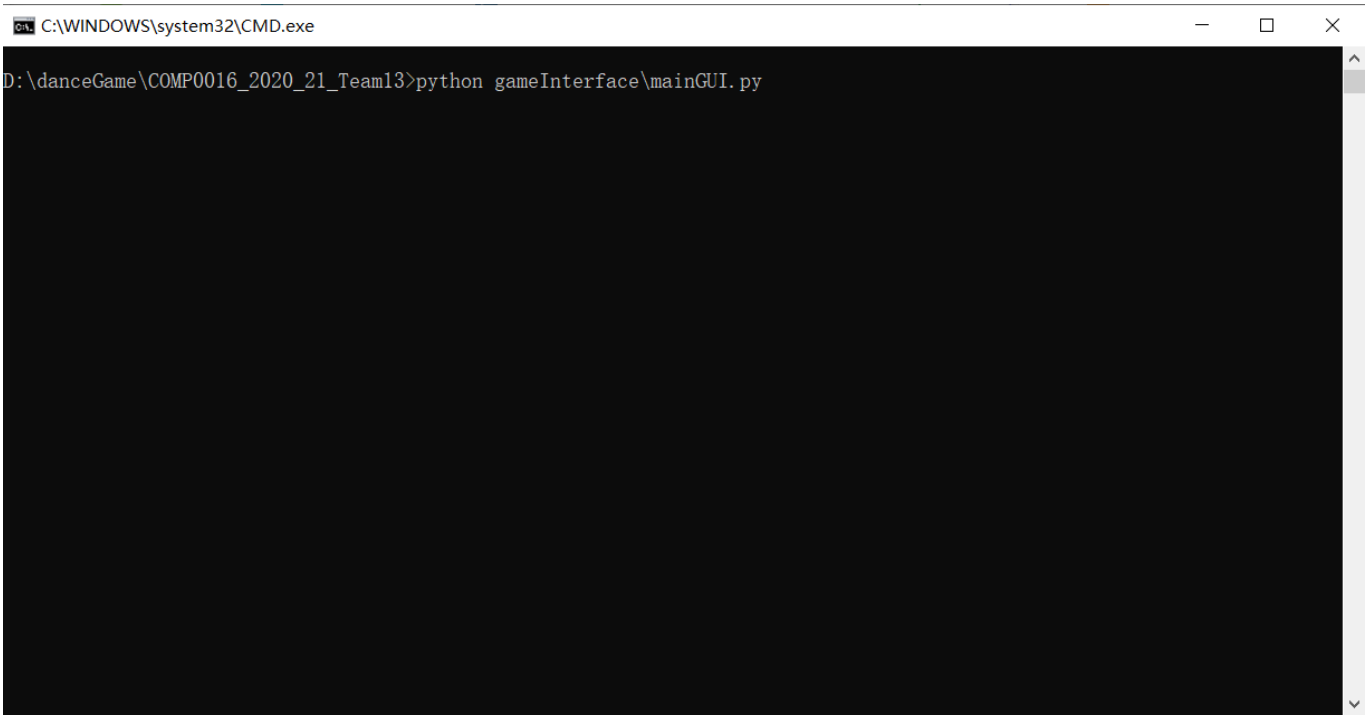
User Manual For Both MacOS and Win10 Users

User Maunal

Important Notice:

To correctly run any files that has imported team13api.py,

ALL FILES MUST RUN AT THE DIRECTORY OF COMP0016_2020_21_Team13 FOLDER:

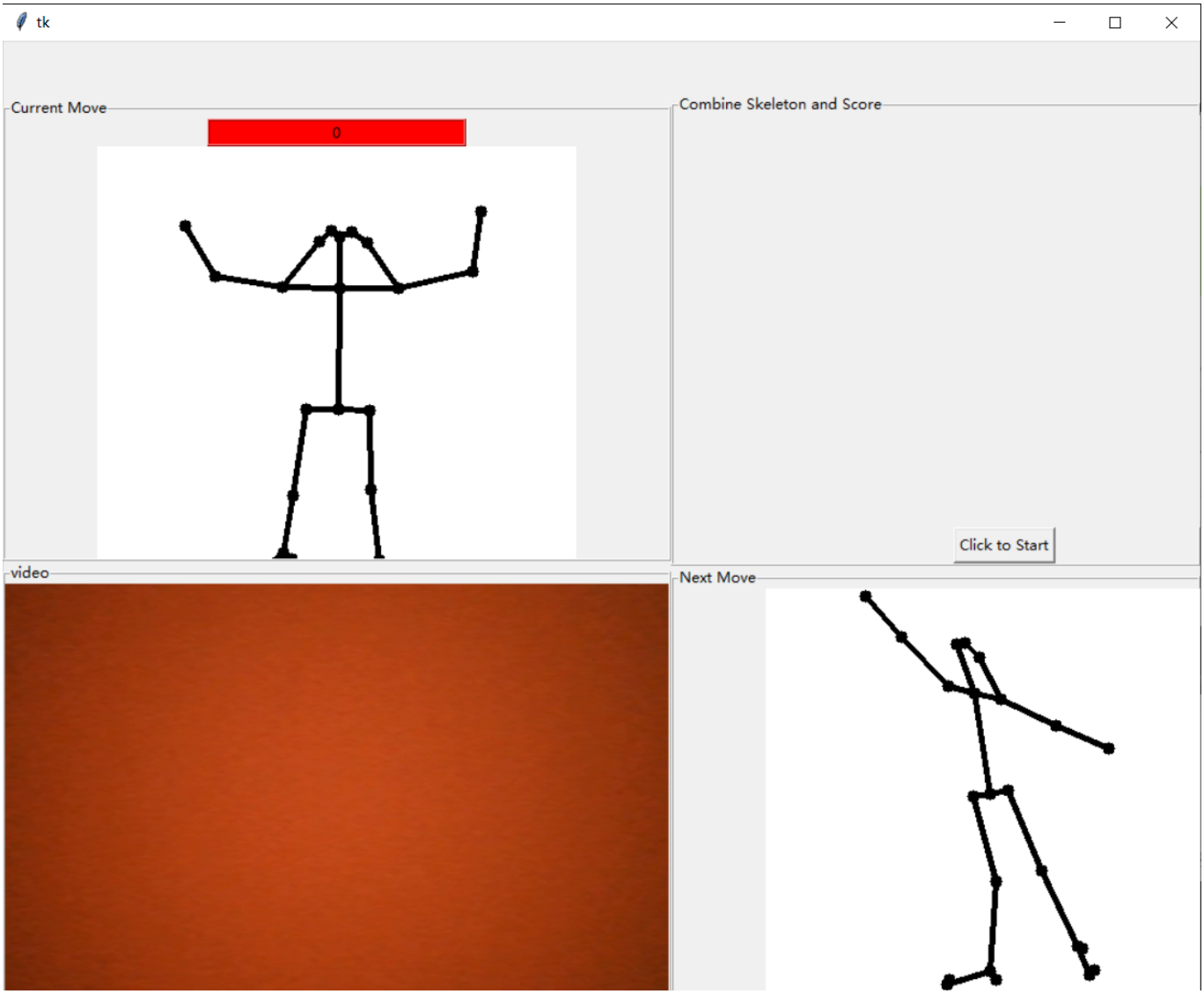


Unless if you changed the location of openpose folder, REMEMBER to change the directory in the team13api.py AND that in opInfo.py, accordng to your platform

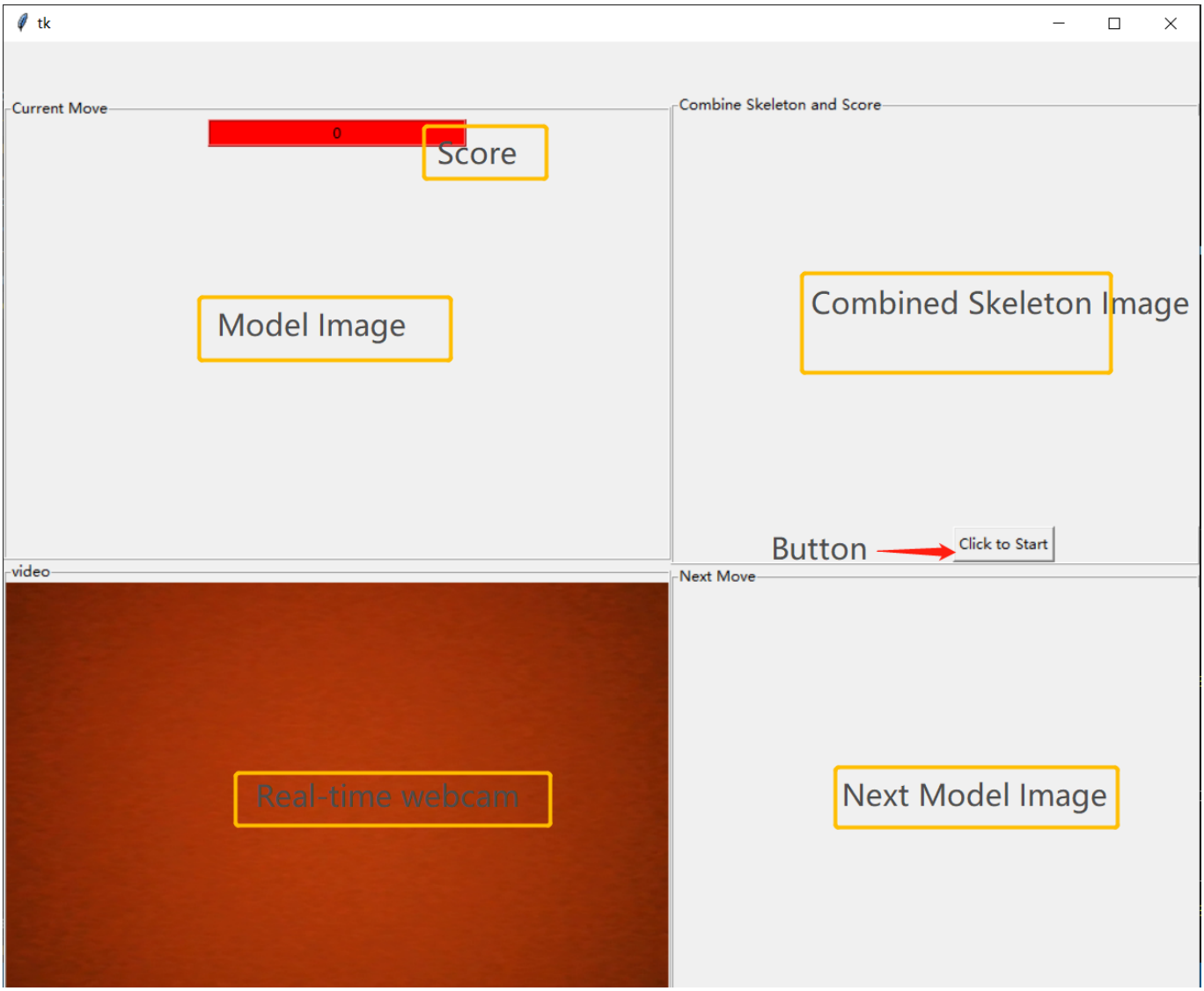
```
try:
    # Windows Import
    if platform == "win32":
        # Change these variables to point to the correct folder (Release/x64 etc.)
        sys.path.append(dir_path + r'/../../openpose/build/python/openpose/Release');
        os.environ['PATH']  = os.environ['PATH'] + ';' + dir_path + r'/../../openpose/build/x64/Release;' +  dir_path + '/../../openpose/build/bin;'
        import pyopenpose as op
    else:
        # Change these variables to point to the correct folder (Release/x64 etc.)
        sys.path.append('../openpose/build/python');
        # If you run `make install` (default path is `/usr/local/python` for Ubuntu), you can also access the OpenPose/python module from there. This will install OpenPos
        # sys.path.append('/usr/local/python')
        from openpose import pyopenpose as op
except ImportError as e:
    print('Error: OpenPose library could not be found. Did you enable `BUILD_PYTHON` in CMake and have this Python script in the right folder?')
    raise e

params = dict()
params["model_folder"] = "../openpose/models/"
```

gameInterface folder

Go to danceGame\COMP0016_2020_21_Team13\gameInterface, and run mainGUI.py under directory COMP0016_2020_21_Team13 see example above. Expected output:



Start the game by running it under COMP0016_2020_21_Team13 as the working directory



Testing folder

1. IntegrationTest\scoring. Code folder holds `imageScoringTest.py` which process a list of images and output all the images with their skeleton
2. UnitTest\skeletonRecognition. Code folder holds `skeletonRecognition.py` which processes a list of pairs of images stored in data folder and output all the skeleton in a window when the program finished.
3. UnitTest\webcamTest. Code folder holds `webcamTesting.py` which turns on the webcam and show real-time video shoot by the webcam on user's computer.

PreprocessSkeleton folder

1. Input folder is where images need to be preprocessed are placed, remember to delete preprocessOutput folder before running `preprocessSkeleton.py`
2. `preprocessSkeleton.py`, a program that preprocess images stored at input folder, save output preprocessed files as .npz flies, create a folder called preprocessOutput and all the .npz files are stored in the preprocessOutput folder.

Legal Issue

Legal Statement

The software is an early proof of concept for development purposes and should not be used as-is in a live environment without further redevelopment and/or testing. No warranty is given and no real data or personally identifiable data should be stored. Usage and its liabilities are your own.

Our software follows the AGPLv3 license

Data Privacy Consideration

The data we used in our system are all pictures. The pictures shown when you run the python files in openpose/examples/tutorial_api_python and stored in openpose/examples/media are downloaded when we cloned openpose github repository. Other pictures in COMP0016_2020_21_Team13/testing/integrationTesting/scoring/data are screenshoted from videos in Youtube. All of these data are used only to see if the skeleton finding and skeleton matching algorithms are working correctly. Our game runs a local copy and will not store any user data or images but only process them on runtime.

The pictures shown on our website are screenshoted from the following websites:

https://www.bilibili.com/video/BV1PK4y1Y7jt?t=56&p=2

https://www.youtube.com/watch?v=kbM5aK-M82g&list=RDkbM5aK-M82g&index=1

https://www.youtube.com/watch?v=MOwaUlXZxkI

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

https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/00_index.md

The other pictures of graphs on our website are drawn by ourselves.

If you have any problems with the pictures please contact us.

Software Licences

This project incorporates material from the project(s) listed below.

1. CMU-Perceptual-Computing-Lab/openpose (https://github.com/CMU-Perceptual-Computing-Lab/openpose)

COPYRIGHT: The Software is owned by Licensor and is protected by United States copyright laws and applicable international treaties and/or conventions.

licence type: SOFTWARE LICENSE AGREEMENT.

2. Tkinter

licence type: MIT.

3. Matplotlib

licence type: PSF.

4. Python 3.7.1

licence type: PSF

5. PIL

licence type: HPND.

6. Numpy

licence type: BSD 3-Clause "New" or "Revised" License.

7. Opencv

license type: Apache License.

Credits

System developed by Zhichen Xu, Jianping Huang, Qianhui Zhang. Clients and organisations Arthur Murray Dance Studio, Adrian Persad, Shaun Persad. Supervisors and Teaching Assistants Dr Yun Fu, An Zhao. University College London