Imperial College London

Real-time Hand Pose Estimator

Qi Ye Tae-Kyun Kim

Imperial College London

Contents

1 Method description

This is the discriminative part of the method for hand pose estimation proposed in [1] with small modifications. Please refer to the paper for detailed method description.

The modifications are

1. As the cascaded refinement only achieves about 1mm error drop, for the estimators for the partial poses, only 1 stage is used.
2. In [1], four layers are used while in this implementation, three layers are constructed as the dip and tip joints are highly correlated.
3. [1] assumes the hand is detected; in this project, an extra hand detector based on unet[2] is trained.

2 Testing

2.1 Offline testing

The run-time of the hand detection on GTX1080Ti, Intel i7-3820 CPU, is about **20fps** and that of the pose estimator is about **25fps**. The mean error tested on the unseen subject is **13.76mm**. Please run the file *code/src/test_pose_estimator_offline.py* for verification.

The size of all the models is **239MB,** which is stored under **data/hier/model**.

To run the code,

1. Download the *code.zip* and *data.zip*. Unzip the folders and put them under the same directory.
2. Open the command line
3. Go to the *code* directory
4. Type *python -m src.test_pose_estimator_offline* to execute the code.

Note: for all the scripts, please change it to the directory *save_dir* accordingly if error reports that no such file is found.

2.2 Online testing

The pipeline with a single process is in *code/src/pose_demo.py.*

The real-time pipeline with two processes is implemented in *code/src/pose_demo_multiprocess.py*.

The run-time of the whole pipeline is about **15fps**. The implementation is not fully parallelized: only the hand detector and the pose estimator are in parallel; the estimators for all joints on the fingers can also be implemented in parallel.

The bottleneck for the run-time efficiency of the current model is the hand detector. To speed up, train the hand detector with downsampled input images, say downsample the original image 480*640 to 120*160, which should have a very small influence on the detection rate.

If a detector whose input size is 120*160 is used and all the estimator for the finger joints are run in parallel, the time needed for running the whole pipeline should be the time of running the *palm_stage_0* model or *pixel_hand_unet_fullimg* model, which is expected to be at least 60fps offline.

3 Training

The training is conducted on images of 9 subjects of Bighand dataset[3] with scale, translation, in-plane rotation augmentations. Please see the code for the setting of all parameters.

The code for training the model is under the *code/src/Hier_Estimator* folder, which should be run in the order as follows:

1. *pixel_hand_unet_fullimg.py*
2. *palm_stage_0.py* (depending on the trade-off between efficiency and accuracy, a further refinement model for the palm joints can be trained)
3. *pip_stage_0.py*
4. *dtip_stage_0.py*

To run the code,

1. Go to the *code* directory
2. Type *python -m src.Hier_Estimator.pixel_hand_unet_fullimg.py* to execute the code. Change the directory in the code where the images are stored.

4 Dependency

4.1 OS

Windows10

CUDA 8 + cuDNN 5

Visual studio 2017

4.2 Training and Testing on offline images

Python 3.5

matplotlib

scipy

pil

skimage

numpy

scikit-learn

h5py

mpl_toolkits

Recommendation: install anaconda to get all the above dependencies with python3.5 and 64bit

tensorflow

keras

opencv for python

4.3 Real-time pipeline

Intel RealSense Camera SR300

Camera driver

Python wrapper of the realsense

To run *pose_demo.py* and *pose_demo_multiprocess.py*, install the camera driver and build the realsense package for python3.5.

For Intel realsense installation, please follow the link to build from source for python3.5 https://github.com/IntelRealSense/librealsense/blob/development/doc/installation_windows.md

https://github.com/IntelRealSense/librealsense/tree/development

Check the requirements of the depth camera, like Visual studio 2017 and windows 10. Please see the appendix for some extra guidance.

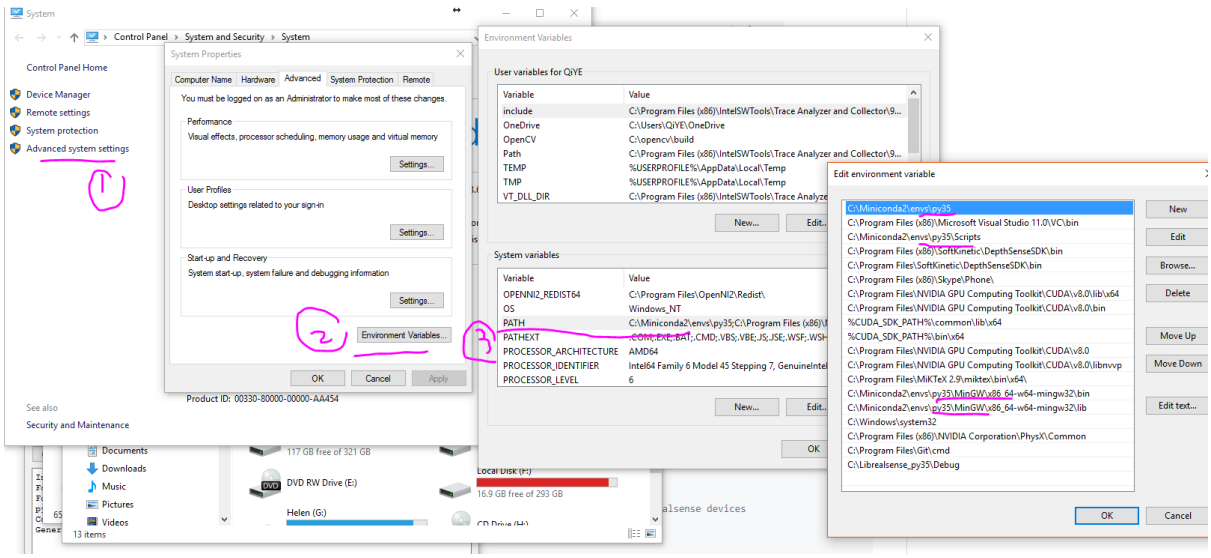
5 Appendix: build realsense python wrapper

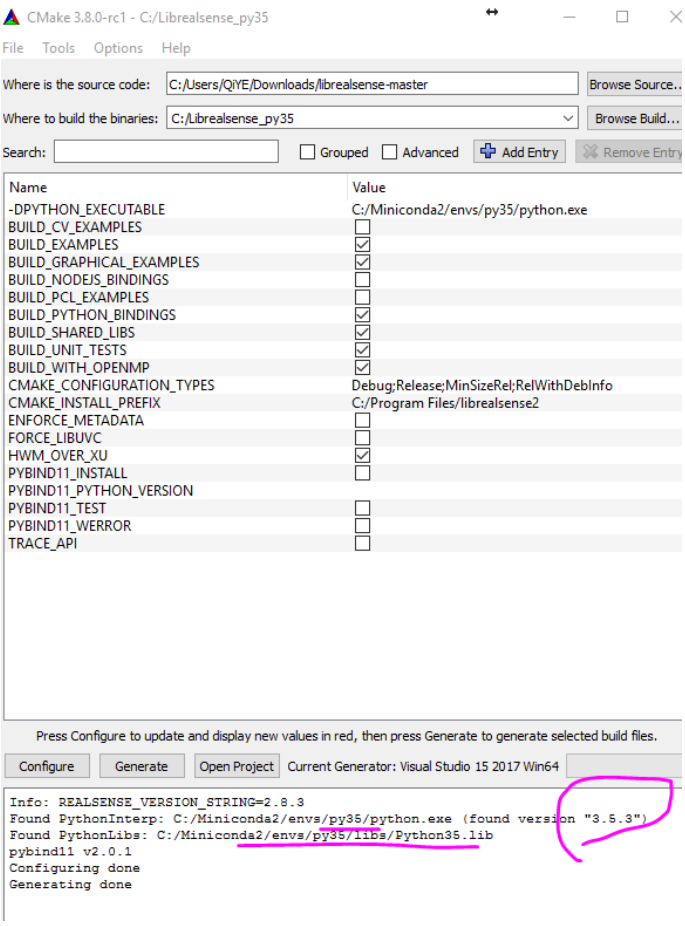
Maker sure the python version and architecture matches, otherwise errors will be reported, for examples:

> The error ImportError: DLL load failed: The specified module could not be found might indicate versions mismatch or architecture (x86 vs x64) mismatch.
>
> The DLL error. Python2.7.dll is not matched with current python version.

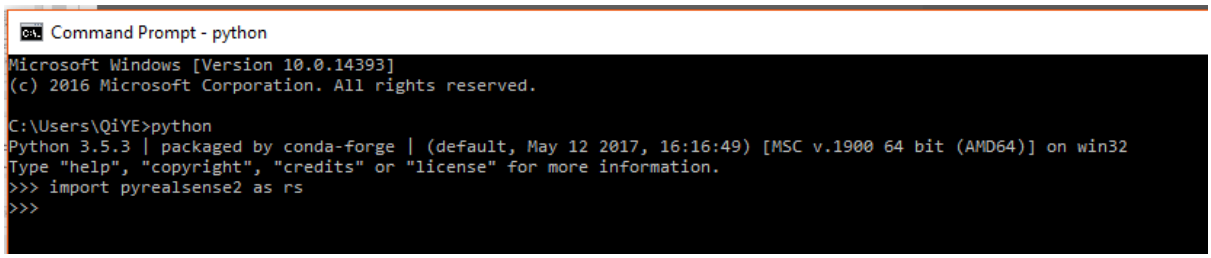
Follow the guidance below together with
https://github.com/IntelRealSense/librealsense/blob/development/doc/installation_windows.md

1. Install CMake-gui
2. Make sure the python 3.5 is used for compiling. If not, change the system path to the python3.5 path. (only setting -DPYTHON_EXECURABLE below doesn't work on my computer)





3. Run CMake-gui, fill the source code downloaded from the link and destination binaries path you want to store the binaries.
4. Press configure
5. Make sure tick the BUILD_PYTHON_BINDING and add -DPYTHON_EXECURABLE
6. Press Generate
7. Make sure the messages in the bottom panel show the right python version
8. Press OpenProject
9. Build the project and add the complied path to the system path. For example, my destination binaries path is Librealsense_py35, add the path *C:\Librealsense_py35\Debug* to system path
10. Copy *pyrealsense2.cp35-win_amd64.pyd* to *the python package directory. For example, I copy C:\Librealsense_py35\Debug\pyrealsense2.cp35-win_amd64.pyd* to the path *C:\Miniconda2\envs\py35\Lib\site-packages*
11. Open command line and run



No error. Pyrealsense2 imported successfully.

6 Reference

[1] Q. Ye, S. Yuan, T-K. Kim. Spatial Attention Deep Net with Partial PSO for Hierarchical Hybrid Hand Pose Estimation. ECCV2016.

[2] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham, 2015.

[3] S. Yuan, Q. Ye, B. Stenger, S. Jain, T-K. Kim. BigHand2.2M Benchmark: Hand Pose Dataset and State of the Art Analysis. In CVPR 2017