

1 of 5

8.1

Monday, 5 October 2020 8:28 PM

Since I cant understand why would the points be different and I cant seem to find a solution for this, I decided to open an issue on the openpose and realsense github
<https://github.com/CMU-Perceptual-Computing-Lab/openpose/issues/1713>
<https://github.com/IntelRealSense/librealsense/issues/7497>

=> Therefore, I am going to try assistive-rehab module.

Group 2 meeting 9pm-9.25pm

- Get everyone sync up with what we do.
- Talk about what we having troubles on
- Set a date for a physical meeting so we can start on integration, and discuss on an aesthetic version of our final project

2 of 5

8.2

Tuesday, 6 October 2020 9:19 AM

Requirements

- Supported Operating Systems: Linux, Windows, macOS ✓
- C++11 compiler ✓
- CMake 3.5 ✓
- [ycm](#)
- [icub-contrib-common](#)
- [yarp](#) (3.1.100 or higher)
- [iCub](#)
- [OpenCV](#) (3.4.0 or higher)
- [yarpOpenPose](#)
- [lpopt](#)
- [yarp.js](#)

From here, seems to have 2 main foreign modules:

- 1) [Yarp](#)
- 2) [icub](#)

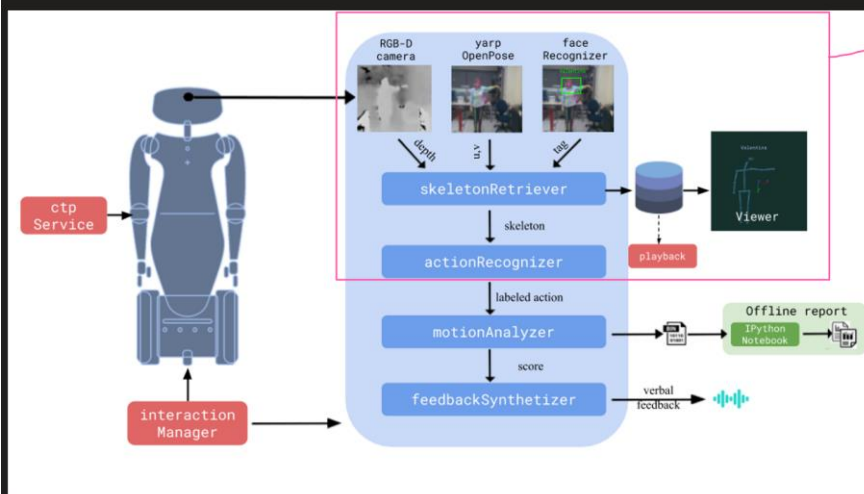
What is [yarp](#)?

A middleware between hardware robotics and software libraries.

Basically for robotics.

What is [icub](#)?

Software for humanoid robotics



This is what i want, without the face recognition

Trying to install this resulted in me getting alot of errors during installation.

The worst part is when compiling [yarp](#), it wasnt able to get my realsense directory. Hence, when i compile, the realsense portion is not enabled.

While searching through intel's community forum:

<https://community.intel.com/t5/forums/searchpage/tab/message?q=3d%20human&advanced=false>

I realised that alot of users used this programme called "OpenVINO™ Toolkit".

Looking through the documentation leads to:

http://docs.openvinotoolkit.org/latest/omz_demos_README.html

->

https://docs.openvinotoolkit.org/latest/omz_demos_python_demos_human_pose_estimation_3d_demo_README.html

It seems that it can get a 3D pose estimation using python.

STOP AT step 1 of Model Optimizer Configuration Steps

ERROR: After October 2020 you may experience errors when installing or updating packages. This is because pip will change the way that it resolves dependency conflicts.

We recommend you use --use-feature=2020-resolver to test your packages with the new resolver before it becomes the default.

```
google-auth 1.22.1 requires setuptools>=40.3.0, but you'll have setuptools 28.8.0 which is incompatible.
tensorflow 2.3.0 requires requests<3,>=2.21.0, but you'll have requests 2.18.4 which is incompatible.
tensorflow 2.3.0 requires setuptools>=41.0.0, but you'll have setuptools 28.8.0 which is incompatible.
tensorflow 2.3.1 requires numpy<1.19.0,>=1.16.0, but you'll have numpy 1.19.1 which is incompatible.
mxnet 1.5.0 requires numpy<1.17.0,>=1.8.2, but you'll have numpy 1.19.1 which is incompatible.
Successfully installed absl-py-0.10.0 astunparse-1.6.3 cachetools-4.1.1 certifi-2020.6.20 chardet-3.0.4 decorator-4.4.2
defusedxml-0.6.0 gast-0.3.3 google-auth-1.22.1 google-auth-oauthlib-0.4.1 google-pasta-0.2.0 graphviz-0.8.4 grpcio-1.32.0
h5py-2.10.0 idna-2.6 importlib-metadata-2.0.0 keras-preprocessing-1.1.2 markdown-3.2.2 mxnet-1.5.0 networkx-2.5 oauthlib-3.1.0
onnx-1.7.0 opt-einsum-3.3.0 protobuf-3.13.0 pyasn1-0.4.8 pyasn1-modules-0.2.8 requests-2.18.4 requests-oauthlib-1.3.0
rsa-4.6 six-1.15.0 tensorflow-2.3.0 tensorflow-plugin-wit-1.7.0 tensorflow-2.3.1 tensorflow-estimator-2.3.0 termcolor-1.1.0
test-generator-0.1.1 typing-extensions-3.7.4.3 urllib3-1.22 werkzeug-1.0.1 wheel-0.35.1 wrapt-1.12.1 zipp-3.3.0
```

WARNING: You are using pip version 20.2.2; however, version 20.2.3 is available.
You should consider upgrading via the 'd:\studies\python\python.exe -m pip install --upgrade pip' command.

Warning: please expect that Model Optimizer conversion might be slow.
You can boost conversion speed by installing protobuf-*.egg located in the
"model-optimizer\install_prerequisites" folder or building protobuf library from sources.
For more information please refer to Model Optimizer FAQ, question #80.

Meeting - 2.45pm

- 1) Aspect ratio of code might cause a distortion of the camera
- 2) Tiny wire for master & slave
- 3) Help each other. If one got problem, help him/her.
- 4) ANOTHER MEETING ON FRIDAY

Do a start - intermediate - end stance.

Take screen grab, superimpose, show.

Do a workflow and then we seperate see get each snippet.

Let her do code checking.

The checkpoint now is to do superimpose check on images. See correct on images first (static) before moving to real time.

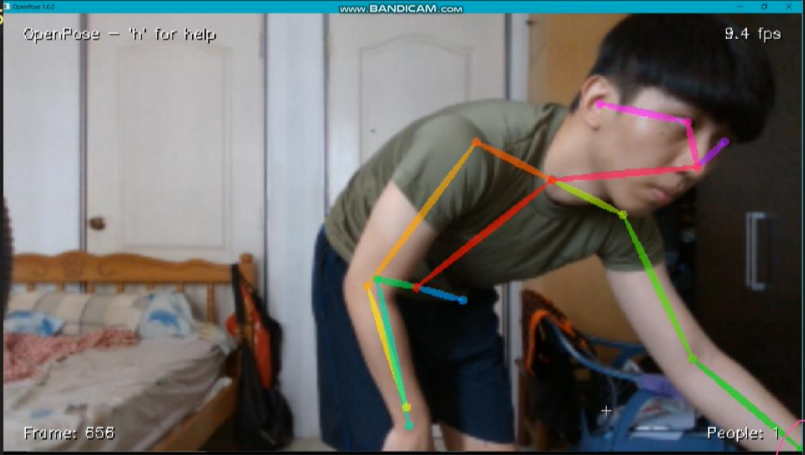
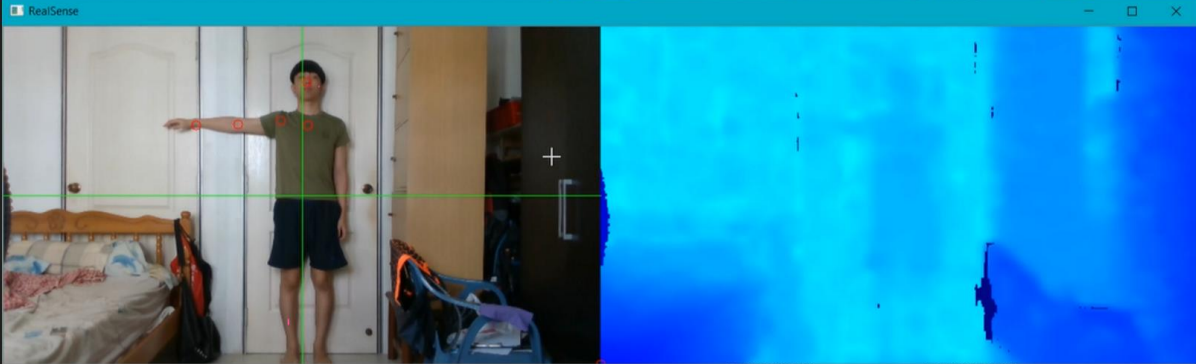
9) Aspect ratio test [6/10/20]

Depth Field of View (FOV) (Horizontal × Vertical × Diagonal)	86° × 57° (±3)
Depth Stream Output Resolution	Up to 1280 x 720
Depth Stream Output Frame Rate	Up to 90 fps
Minimum Depth Distance (Min-Z)	0.1 m
Sensor Shutter Type	Global Shutter
Maximum Range	Approx. 10 meters; Varies depending on calibration, scene, and lighting condition
RGB Sensor Resolution and Frame Rate	1920 x 1080 at 30 fps

Both 16:9

1280x720 scaled down by 2 -> 640x360
 1920x1080 scaled down by 3 -> 640x360
 Setting Keypoint_scale 0 and camRes 640x360
 "7": [633.415, 357.553, 0.120146]

Seems good

Week 8

```

graph LR
    camera --> Realsense_SDK[Realsense SDK]
    Realsense_SDK --> Colour_frame[Colour_frame]
    Realsense_SDK --> Depth_frame[Depth_frame]
    Realsense_SDK --> Viewer[Viewer with guidance skeleton and user skeleton]
    Guidance_Json[Guidance Json files] --> Realsense_SDK
    Colour_frame --> Openpose[Openpose]
    Depth_frame --> Openpose
    Openpose --> One_Frame["1 frame"]
    Openpose --> Json_files["Json files of my coordinates per frame input"]
    Json_files --> Viewer
  
```

Task distribution:
 Wp -> Do the math to find out the angles and the allowance for a pose to be counted as correct.
 Nab -> help to figure out how to import json files using python and extract the necessary numbers

1) Since i got the correct 2D points, i want to draw it out in realsense viewer.

2) Next, i want to run openpose inside of realsense so that each time the pipeline in realsense receives a frame from camera, it will send the colour_frame into openpose.
 So that openpose can get me another skeleton that moves with the user.

- To do this:
 - 1) I need to get openpose running by itself (no bat files to run with the parameters) ✓
 - 2) able to take in a frame (picture into openpose) ✓
 - 3) save to json file in a folder and has no display ✓
 - Realsense part---
 - 4) save each colour_frame that it streams from camera into an image in a folder ✓
 - 5) get openpose running inside the realsense pipeline ✓ *Achieved as of 7/10/2020*
 - 6) supply the openpose with the images inside the pipeline ✓
 - 7) import the json files into python to get the user skeleton to move with the frame shown in the viewer (in blue) ✓
 - 8) superimpose the end skeleton (in red) ✓
 - 9) superimpose the start skeleton ✓
 - 10) integrate the math into this to trigger colours to green when user skeleton within the allowance of the guidance skeleton ✓ *Achieved as of 9/10/20*

Note: Add in the start pose and make it transition to the end pose

But I would need more conditions to make the starting pose not so sensitive to false negatives in the background.

**Colour_image
Need go through the mumpy part first before saving*

Might have to save like 5 frames at a time, then keep replacing those 5 frames. So that it doesnt flood the storage.

**For 6) might have to get it to do this first. Make sure the json output is stable.*

- For future to increase fps perhaps can use: <https://www.pyimagesearch.com/2015/12/21/increasing-webcam-fps-with-python-and-opencv/>

3) [9/10/2020] Can extract data using `nab`'s work

4) BUT!!!!!!

The fps is very very low because i need to run `openpose` every time a new frame comes in. So every time need to execute `openpose` takes up `alot` of time.

5) 812 -> dot(12, 18) -> 12 is 2-1, 18 is 8-1

123 -> dot(21, 23) -> 21 is 1-2, 23 is 3-2

Functional

- 1) Since guidance skeleton is already fixed in place, maybe can just import the angles?
 - Is there really a need to calculate? Since is already predetermined.
 - Can just have a `py` file for future angle calculations to make developers' work more automatic

[REJECTED on 10/10/20]

2) More angles condition

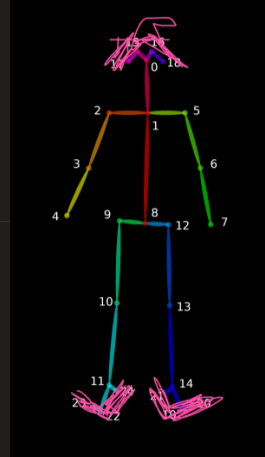
3) A starting question to choose what which exercise

OR

Using back the GUI and having different pose run different script which essentially just have different values of start and end.

Aesthetic

- 1) Remove drawing of points other than 0 from the head and toes ✓



For now just want ankles (11,14)

[DONE on 10/10/20]

- 2) Lines for user guidance skeleton. Maybe make the lines thinner and translucent is possible.