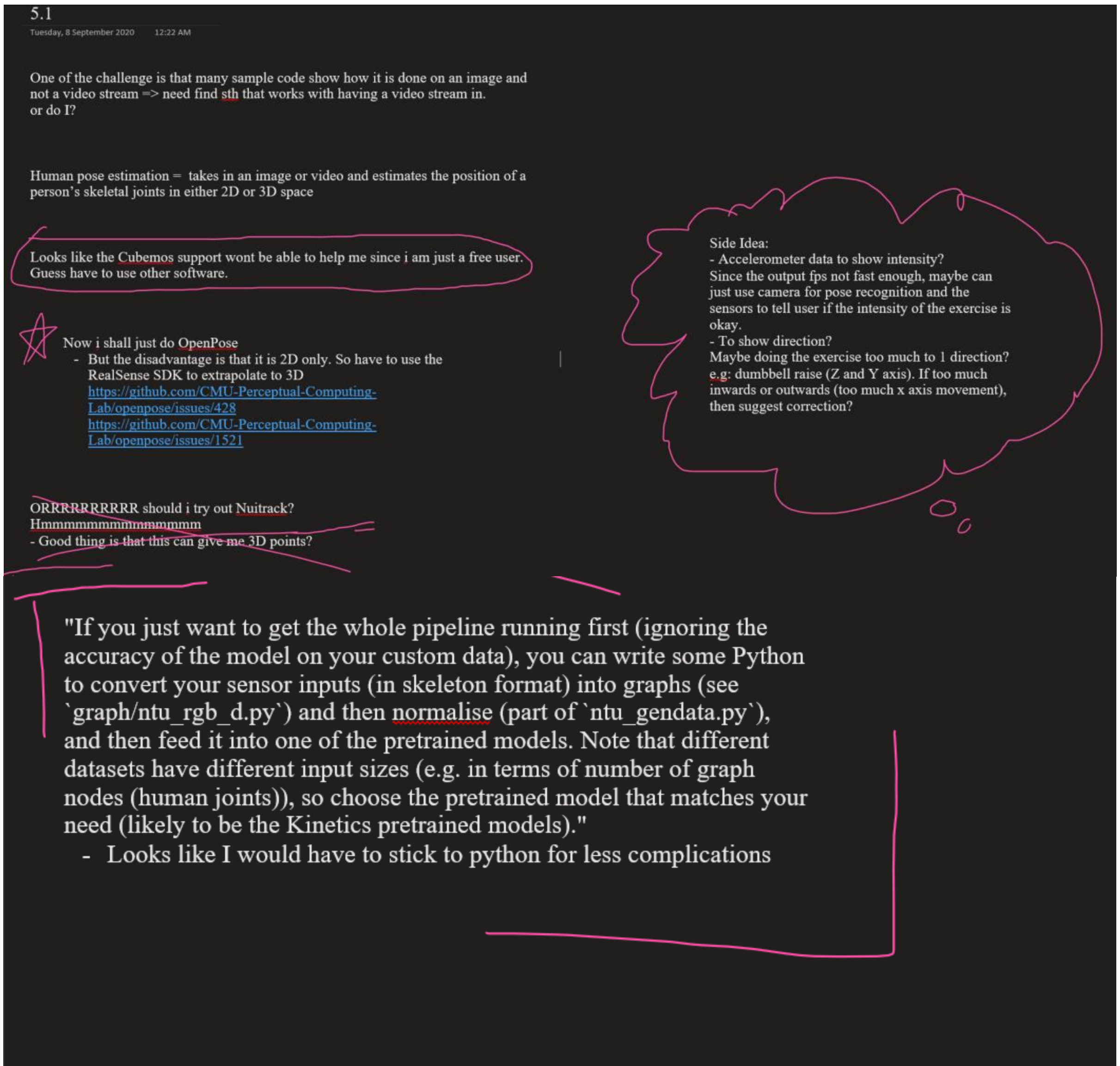


5.1

Tuesday, 8 September 2020 12:22 AM

One of the challenge is that many sample code show how it is done on an image and not a video stream => need find sth that works with having a video stream in.
or do I?

Human pose estimation = takes in an image or video and estimates the position of a person's skeletal joints in either 2D or 3D space

Looks like the Cubemos support wont be able to help me since i am just a free user. Guess have to use other software.

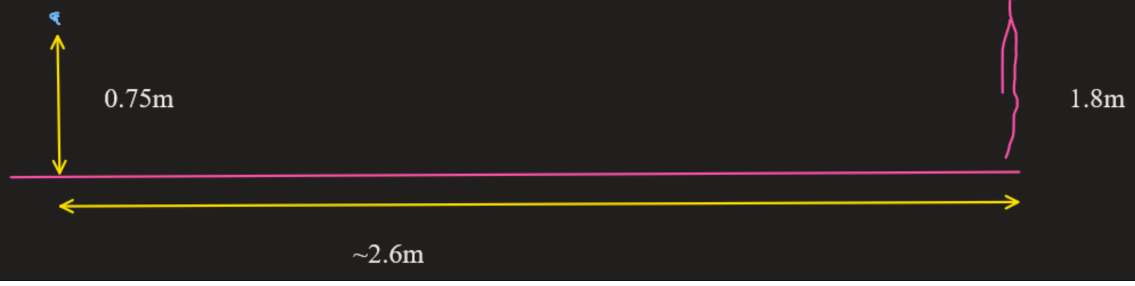
Now i shall just do OpenPose
- But the disadvantage is that it is 2D only. So have to use the RealSense SDK to extrapolate to 3D
  https://github.com/CMU-Perceptual-Computing-Lab/openpose/issues/428
  https://github.com/CMU-Perceptual-Computing-Lab/openpose/issues/1521

~~ORRRRRRRRRR should i try out Nuitrack?~~
~~Hmmmmmmmmmmmmmm~~
~~- Good thing is that this can give me 3D points?~~

Side Idea:
- Accelerometer data to show intensity?
Since the output fps not fast enough, maybe can just use camera for pose recognition and the sensors to tell user if the intensity of the exercise is okay.
- To show direction?
Maybe doing the exercise too much to 1 direction? e.g: dumbbell raise (Z and Y axis). If too much inwards or outwards (too much x axis movement), then suggest correction?

"If you just want to get the whole pipeline running first (ignoring the accuracy of the model on your custom data), you can write some Python to convert your sensor inputs (in skeleton format) into graphs (see `graph/ntu_rgb_d.py`) and then normalise (part of `ntu_gendata.py`), and then feed it into one of the pretrained models. Note that different datasets have different input sizes (e.g. in terms of number of graph nodes (human joints)), so choose the pretrained model that matches your need (likely to be the Kinetics pretrained models)."
- Looks like I would have to stick to python for less complications

Camera on tripod, on a box, 2 tapes and chair



5.2 - I & R (implement and Results)

Thursday, 10 September 2020 10:41 PM

Important Links:
https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/include/openpose/flags.hpp
https://cmu-perceptual-computing-lab.github.io/openpose/html/flags_8hpp.html
- for flags
https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/output.md#camera-matrix-output-format
- output format
https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/media/keypoints_pose_25.png
- keypoint image

Installing Openpose
- Does the openposedemo work?
- Can I run the python examples?

Now to start reading and seeing how in the world do I code this out.

How do I present the camera side?
Pose2pose??
https://github.com/HaminyG/openpose-video-keypoints-in-json-file/blob/master/openpose_video_json
- getting keypoint from saved video

Run the py file

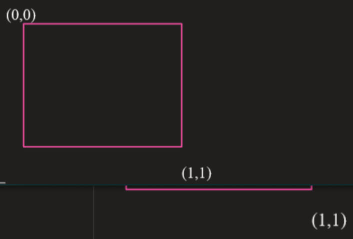
Get output but don't display it

1) How to get ouput json files for number of ppl and keypoints:
   - go to the directory with the code
   - run cmd
   - Lets say want run the webcam one in tutorial_api_python, then type " openpose_python.py --write_json output/ --display 0 --render_pose 0 "

   - adding '--part_candidates' changes to 0-18 type output.
   - adding '--keypoint_scale 3' normalised to 0 and 1.

   - IMPORTANT: made a bash file that help me run the py script with the flags to produce an output folder with JSON files

   Result 1 (testing with no video recorded to check frames):



```
{"version":1.3,"people":[{"person_id":[-1],
    "pose_keypoints_2d":[
```

```
{"version":1.3,"people":[{"person_id":[-1],
    "pose_keypoints_2d":[
        574.522,84.6471,0.855848,
        576.45,163.072,0.853093,
        523.562,165,0.773728,
        427.568,180.693,0.77568,
        331.552,196.395,0.757932,
        633.25,161.174,0.770652,
        741.049,147.378,0.77427,
        838.986,133.726,0.870979,
        582.37,351.145,0.612031,
        541.268,351.195,0.610205,
        570.516,507.805,0.829209,
        576.526,648.894,0.755967,
        619.514,349.187,0.640119,
        631.323,500.103,0.760103,
        633.273,646.979,0.771377,
        558.905,71.0314,0.858117,
        588.164,69.111,0.890623,
        541.185,86.6408,0.940226,
        603.859,82.7611,0.755392,
        648.922,678.38,0.700729,
        660.718,668.599,0.697843,
        619.657,652.94,0.691815,
        562.746,678.302,0.609835,
        552.99,668.549,0.632986,
        588.18,654.832,0.75431],
```

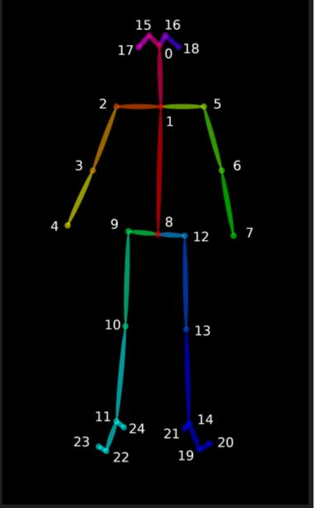
An array pose_keypoints_2d containing the body part locations and detection confidence formatted as x1,y1,c1,x2,y2,c2,....

Why so many sets?

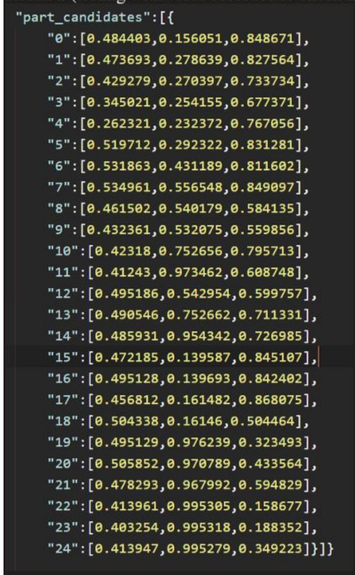
```
"part_candidates":[{
```

```
"part_candidates":[{
    "0":[574.522,84.6471,0.855848,1040.82,441.317,0.233816],
    "1":[576.45,163.072,0.853093,925.197,462.868,0.122519,901.689,470.689,0.119836,889.97,476.525,0.122627],
    "2":[523.562,165,0.773728,936.956,472.67,0.318198,839.036,488.312,0.10243],
    "3":[427.568,180.693,0.77568,948.748,576.487,0.0617706],
    "4":[331.552,196.395,0.757932],
    "5":[633.25,161.174,0.770652,933.021,462.843,0.0800747,837.06,482.423,0.127706],
    "6":[741.049,147.378,0.77427],
    "7":[838.986,133.726,0.870979],
    "8":[582.37,351.145,0.612031],
    "9":[541.268,351.195,0.610205],
    "10":[570.516,507.805,0.829209],
    "11":[576.526,648.894,0.755967],
    "12":[619.514,349.187,0.640119],
    "13":[631.323,500.103,0.760103],
    "14":[633.273,646.979,0.771377],
    "15":[558.905,71.0314,0.858117,1034.91,423.67,0.270953],
    "16":[588.164,69.111,0.890623,1050.6,423.679,0.0805154],
    "17":[541.185,86.6408,0.940226,999.665,415.867,0.271903],
    "18":[603.859,82.7611,0.755392],
    "19":[648.922,678.38,0.700729],
    "20":[660.718,668.599,0.697843],
    "21":[619.657,652.94,0.691815],
    "22":[562.746,678.302,0.609835],
    "23":[552.99,668.549,0.632986],
    "24":[588.18,654.832,0.75431]}]}
```

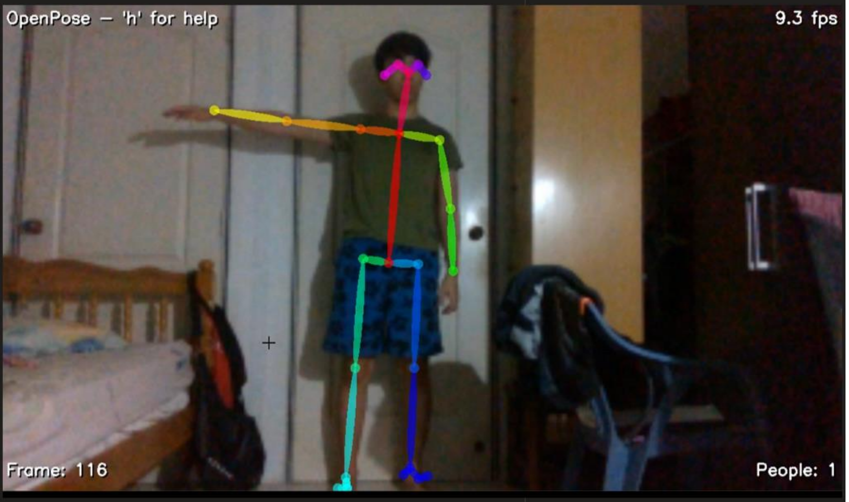
```
{0, "Nose"},
//  {1, "Neck"},
//  {2, "RShoulder"},
//  {3, "RElbow"},
//  {4, "RWrist"},
//  {5, "LShoulder"},
//  {6, "LElbow"},
//  {7, "LWrist"},
//  {8, "MidHip"},
//  {9, "RHip"},
//  {10, "RKnee"},
//  {11, "RAnkle"},
//  {12, "LHip"},
//  {13, "LKnee"},
//  {14, "LAnkle"},
//  {15, "REye"},
//  {16, "LEye"},
//  {17, "REar"},
//  {18, "LEar"},
//  {19, "LBigToe"},
//  {20, "LSmallToe"},
//  {21, "LHeel"},
//  {22, "RBigToe"},
//  {23, "RSmallToe"},
//  {24, "RHeel"},
```



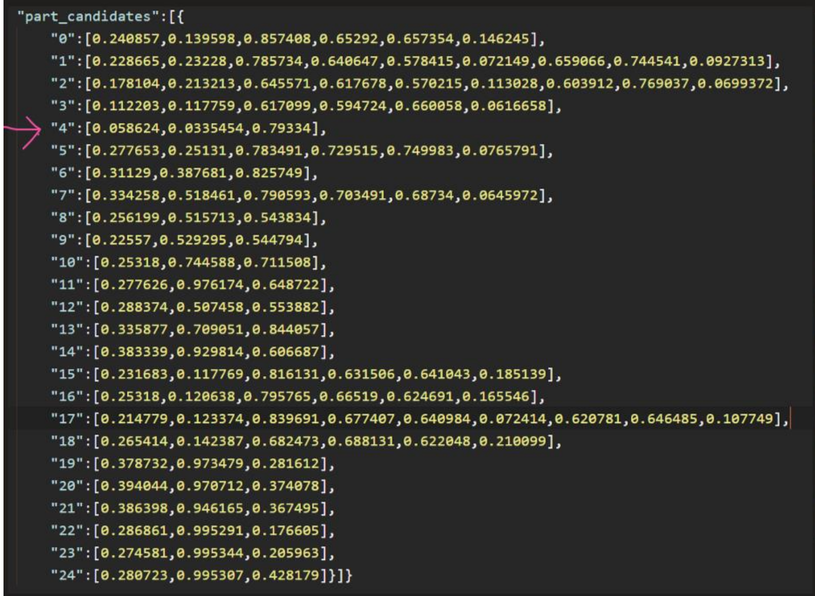
Result 2 (testing with video recorded to check frames):



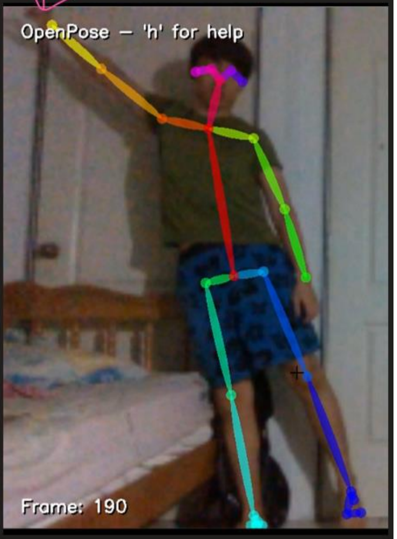
```
"part_candidates":[{
    "0":[0.484403,0.156051,0.848671],
    "1":[0.473693,0.278639,0.827564],
    "2":[0.429279,0.270397,0.733734],
    "3":[0.345021,0.254155,0.677371],
    "4":[0.262321,0.232372,0.767056],
    "5":[0.519712,0.292322,0.831281],
    "6":[0.531863,0.431189,0.811602],
    "7":[0.534961,0.556548,0.849097],
    "8":[0.461502,0.540179,0.584135],
    "9":[0.432361,0.532075,0.559856],
    "10":[0.42318,0.752656,0.795713],
    "11":[0.41243,0.973462,0.608748],
    "12":[0.495186,0.542954,0.599757],
    "13":[0.490546,0.752662,0.711331],
    "14":[0.485931,0.954342,0.726985],
    "15":[0.472185,0.139587,0.845107],
    "16":[0.495128,0.139693,0.842402],
    "17":[0.456812,0.161482,0.868075],
    "18":[0.504338,0.16146,0.504464],
    "19":[0.495129,0.976239,0.323493],
    "20":[0.505852,0.970789,0.433564],
    "21":[0.478293,0.967992,0.594829],
    "22":[0.413961,0.995305,0.158677],
    "23":[0.403254,0.995318,0.188352],
    "24":[0.413947,0.995279,0.349223]}]}
```

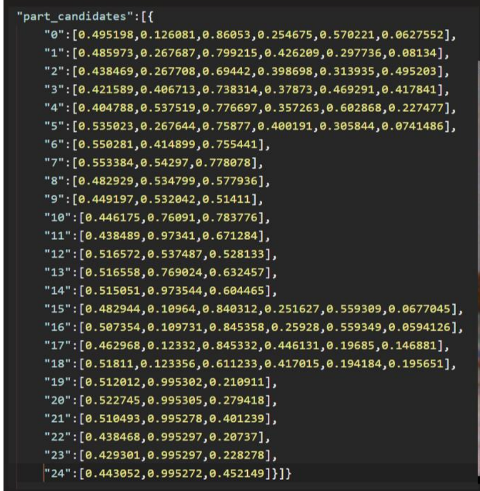


Looks like the corner of the frame is (0,0)

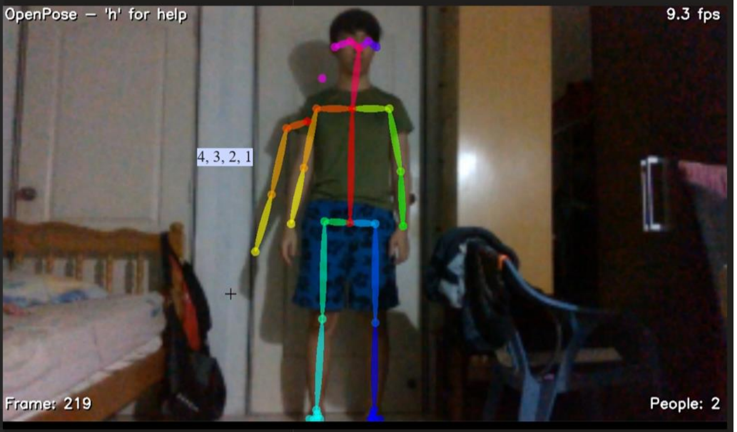


```
"part_candidates":[{
    "0":[0.240857,0.139598,0.857408,0.65292,0.657354,0.146245],
    "1":[0.228665,0.23228,0.785734,0.640647,0.578415,0.072149,0.659066,0.744541,0.0927313],
    "2":[0.178104,0.213213,0.645571,0.617678,0.570215,0.113028,0.603912,0.769037,0.0699372],
    "3":[0.112203,0.117759,0.617099,0.594724,0.660058,0.0616658],
    "4":[0.058624,0.0335454,0.79334],
    "5":[0.277653,0.25131,0.783491,0.729515,0.749983,0.0765791],
    "6":[0.31129,0.387681,0.825749],
    "7":[0.334258,0.518461,0.790593,0.703491,0.68734,0.0645972],
    "8":[0.256199,0.515713,0.543834],
    "9":[0.22557,0.529295,0.544794],
    "10":[0.25318,0.744588,0.711508],
    "11":[0.277626,0.976174,0.648722],
    "12":[0.288374,0.507458,0.553882],
    "13":[0.335877,0.709051,0.844057],
    "14":[0.383339,0.929814,0.606687],
    "15":[0.231683,0.117769,0.816131,0.631506,0.641043,0.185139],
    "16":[0.25318,0.120638,0.795765,0.66519,0.624691,0.165546],
    "17":[0.214779,0.123374,0.839691,0.677407,0.640984,0.072414,0.620781,0.646485,0.107749],
    "18":[0.265414,0.142387,0.682473,0.688131,0.622048,0.210099],
    "19":[0.378732,0.973479,0.281612],
    "20":[0.394044,0.970712,0.374078],
    "21":[0.386398,0.946165,0.367495],
    "22":[0.286861,0.995291,0.176605],
    "23":[0.274581,0.995344,0.205963],
    "24":[0.280723,0.995307,0.428179]}]}
```



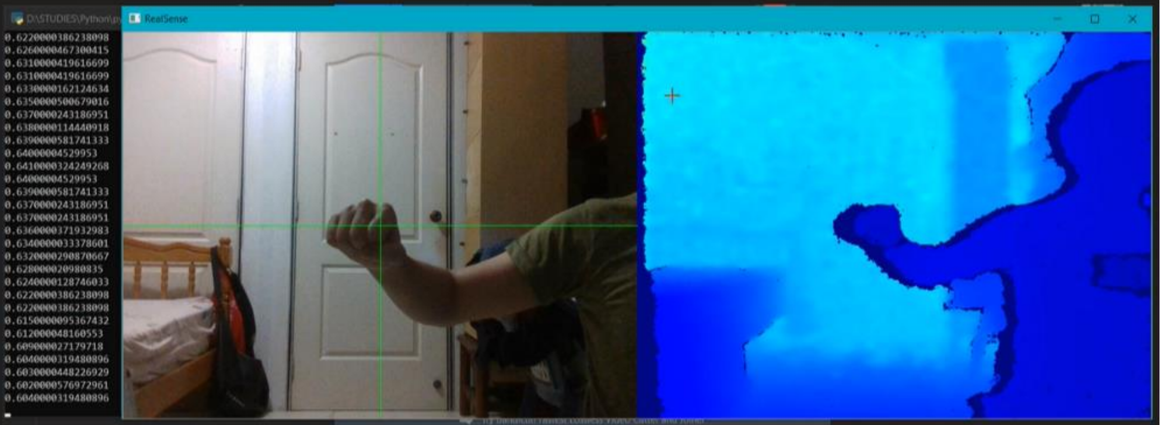


```
"part_candidates":[{
    "0":[0.495198,0.126081,0.86053,0.254675,0.570221,0.0627552],
    "1":[0.485973,0.267687,0.799215,0.426209,0.297736,0.08134],
    "2":[0.438469,0.267708,0.69442,0.398698,0.313935,0.495203],
    "3":[0.421589,0.406713,0.738314,0.37873,0.469291,0.417841],
    "4":[0.404788,0.537519,0.776697,0.357263,0.602868,0.227477],
    "5":[0.535023,0.267644,0.75877,0.400191,0.305844,0.0741486],
    "6":[0.550281,0.414899,0.755441],
    "7":[0.553384,0.54297,0.778078],
    "8":[0.482929,0.534799,0.577936],
    "9":[0.449197,0.532042,0.51411],
    "10":[0.446175,0.76091,0.783776],
    "11":[0.438489,0.97341,0.671284],
    "12":[0.516572,0.537487,0.528133],
    "13":[0.516558,0.769024,0.632457],
    "14":[0.515051,0.973544,0.604465],
    "15":[0.482944,0.10964,0.840312,0.251627,0.559309,0.0677045],
    "16":[0.507354,0.109731,0.845358,0.25928,0.559349,0.0594126],
    "17":[0.462968,0.12332,0.845332,0.446131,0.19685,0.146881],
    "18":[0.51811,0.123356,0.611233,0.417015,0.194184,0.195651],
    "19":[0.512012,0.995302,0.210911],
    "20":[0.522745,0.995305,0.279418],
    "21":[0.510493,0.995278,0.401239],
    "22":[0.438468,0.995297,0.20737],
    "23":[0.429301,0.995297,0.228278],
    "24":[0.443052,0.995272,0.452149]}]}
```



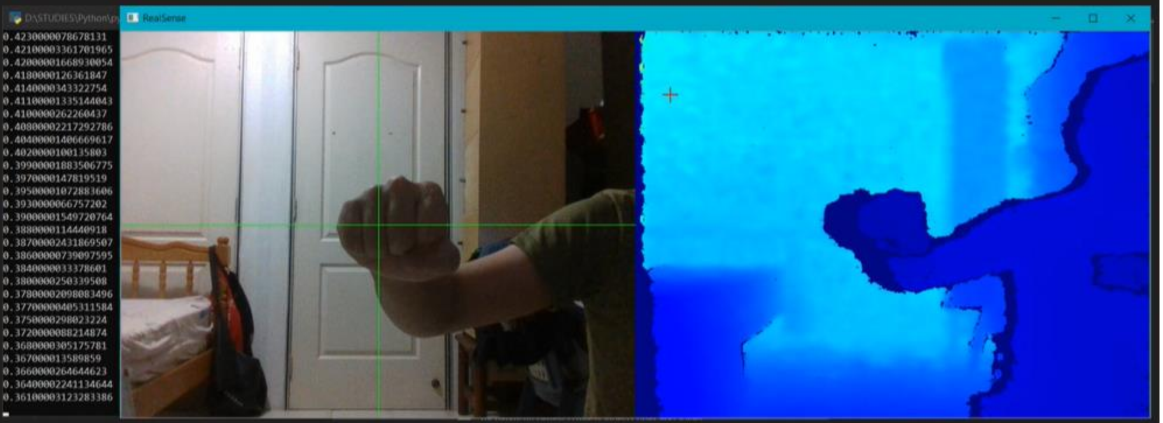
- Added "number_people_max = 1" as a parameter and it stopped detecting my shadow, though the points still have >1 set (>3 elements)

1) Getting depth from Intel Realsense
https://github.com/IntelRealSense/librealsense/issues/1904

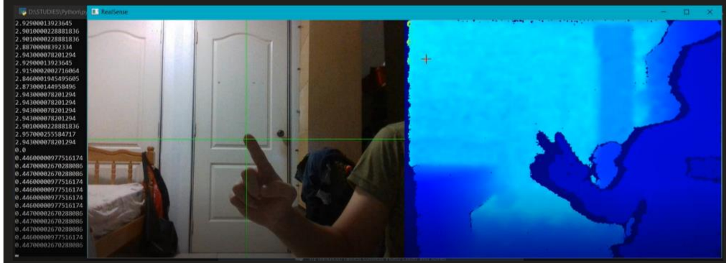


```
0.6220000386238098
0.6260000467300415
0.6310000419616699
0.6310000419616699
0.633000162124634
0.6350000500679016
0.6370000243186951
0.6380000114440918
0.6390000581741333
0.64000004529953
0.6410000324249268
0.64000004529953
0.6390000581741333
0.6370000243186951
0.6370000243186951
0.6360000371932983
0.6340000033378601
0.6320000290870667
0.628000020980835
0.6240000128746033
0.6220000386238098
0.6220000386238098
0.6150000095367432
0.612000048160553
0.609000027179718
0.6040000319480896
0.6030000448226929
0.6020000576972961
0.6040000319480896
```

Resolution: (640, 480)
Finding camera depth distance in the middle (cross in the middle of thickness 1)

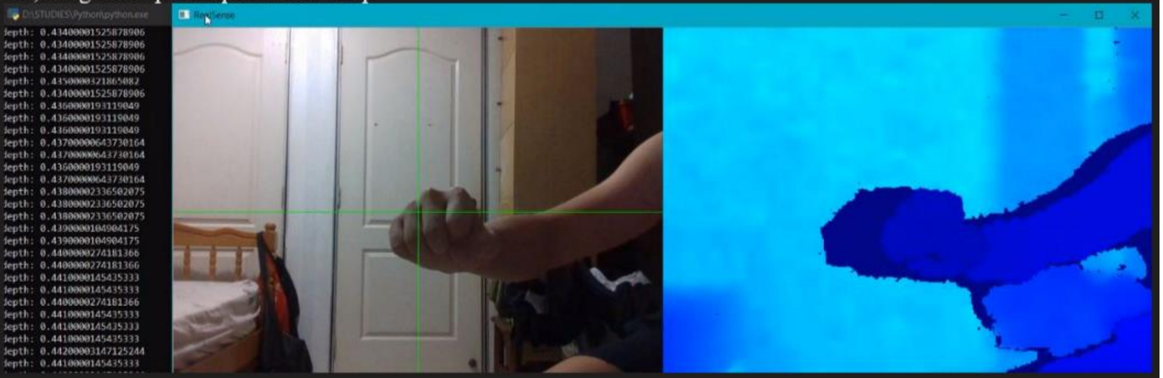


```
0.4230000078678131
0.4210000336170196
0.4200000166893005
0.418000012636184
0.414000034332275
0.4110000133514404
0.410000026226043
0.4080000221729278
0.4040000140666961
0.402000010013580
0.3990000188350677
0.397000014781951
0.3950000107288360
0.393000006675720
0.3900000154972076
0.388000011444091
0.3870000243186950
0.3860000073909759
0.384000003337860
0.380000025033950
0.3780000209808349
0.3770000040531158
0.375000029802322
0.372000008821487
0.368000030517578
0.36700001358985
0.366000026464462
0.3640000224113464
0.3610000312328338
```



```
2.92900013923645
2.9010000228881836
2.9010000228881836
2.88700008392334
2.943000078201294
2.92900013923645
2.9150002002716064
2.8460001945495605
2.873000144958496
2.943000078201294
2.943000078201294
2.943000078201294
2.943000078201294
2.9010000228881836
2.957000255584717
2.943000078201294
0.0
0.44600000977516174
0.44700002670288086
0.44700002670288086
0.44600000977516174
0.44600000977516174
0.44600000977516174
0.44600000977516174
0.44700002670288086
0.44700002670288086
0.44700002670288086
0.44600000977516174
0.44700002670288086
```

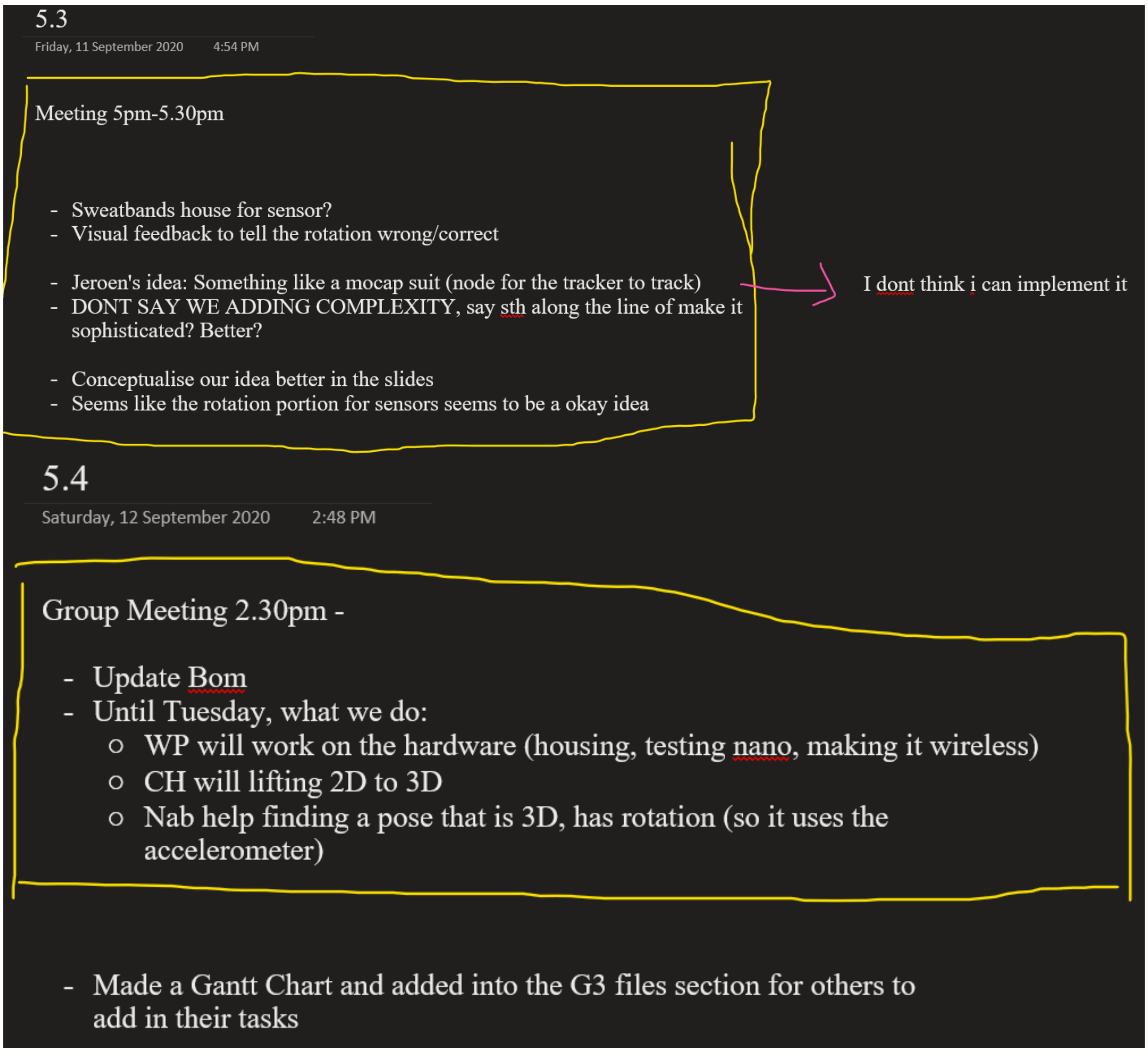
2.1) Aligned depth map to colour map



```
depth: 0.4340000152587890
depth: 0.4340000152587890
depth: 0.4340000152587890
depth: 0.4340000152587890
depth: 0.435000032186508
depth: 0.4340000152587890
depth: 0.436000019311904
depth: 0.436000019311904
depth: 0.436000019311904
depth: 0.4370000064373016
depth: 0.4370000064373016
depth: 0.436000019311904
depth: 0.4370000064373016
depth: 0.4380000233650207
depth: 0.4380000233650207
depth: 0.4380000233650207
depth: 0.439000010490417
depth: 0.439000010490417
depth: 0.440000027418136
depth: 0.440000027418136
depth: 0.441000014543533
depth: 0.441000014543533
depth: 0.440000027418136
depth: 0.441000014543533
depth: 0.441000014543533
depth: 0.441000014543533
depth: 0.4420000314712524
depth: 0.441000014543533
```

2) [13/9/20] rs.rs2_deproject_pixel_to_point

Seems like need the actual pixel position, not in the format of (0,0) to (1,1). But in (0,0) to (640, 480)



5.3

Friday, 11 September 2020 4:54 PM

Meeting 5pm-5.30pm

- Sweatbands house for sensor?
- Visual feedback to tell the rotation wrong/correct

- Jeroen's idea: Something like a mocap suit (node for the tracker to track) → I dont think i can implement it
- DONT SAY WE ADDING COMPLEXITY, say sth along the line of make it sophisticated? Better?

- Conceptualise our idea better in the slides
- Seems like the rotation portion for sensors seems to be a okay idea

5.4

Saturday, 12 September 2020 2:48 PM

Group Meeting 2.30pm -

- Update Bom
- Until Tuesday, what we do:
  - WP will work on the hardware (housing, testing nano, making it wireless)
  - CH will lifting 2D to 3D
  - Nab help finding a pose that is 3D, has rotation (so it uses the accelerometer)

- Made a Gantt Chart and added into the G3 files section for others to add in their tasks