CSCM77 Computer Vision and Deep Learning Lab Class

The lab classes consist of tasks that are designed to be completed during the lab sessions. If you do not complete the tasks during the lab class then you should do them at home and have them ready to be marked off in the next lab class.

You should aim to get all tasks completed and marked off by staff in the lab class. You can only get the lab sheet signed off in the lab classes (not by email, etc.). Lab demonstrators are available to answer questions and discuss the topics as well as sign off tasks.

All lab tasks must be uploaded to Blackboard before the deadline stated on each lab sheet in order to be signed off. You will be required to download your submission from Blackboard in order to be signed off for the lab.

Marks will be awarded if the tasks are marked off by staff during the session they are handed out; or if they are marked off by staff during the following session.

The challenge tasks are optional and are not worth extra marks. However, it is a good idea to attempt these as completing them will help strengthen your programming and problem solving skills. You may need to do a little research to find out how to complete the challenge tasks.

CSCM77 Lab Class 1

Release date: 31/01/2020
Due date: 14/02/2020

This lab is about getting familiar with the KinectV2 depth sensor. There are two Jupyter Notebooks provided for this lab which are provided on Blackboard: `CVDL_Server.ipynb` and `CVDL_Server.ipynb`.

`CVDL_Server.ipynb` will be running on the demonstrator's server machine. This script maintains an open TCP/IP socket which listens for incoming requests. Once a request is made, the server responses by transmitting the color map (image), depth map and infrared map from the connected KinectV2. You should not run this script, but rather read the functionality and understand what is happening with the KinectV2 and getting of frames when requested.

`CVDL_Student.ipynb` is the notebook you will be running. This script attempts to create a socket connection with the server and requests `n_frames` number of frames from the KinectV2. The frames are returned and stored in the numpy ndarrays `img`, `dep`, and `infra` respectively.

☐ Task 1.1

This task is about familiarizing yourself with the Python package PyKinect2. PyKinect2 is an open source Python wrapper for the KinectV2 API, available at https://github.com/Kinect.

1. Download and study the two Jupyter notebooks.
2. Run `CVDL_Student.ipynb` and collect a given number of frames from the KinectV2.
3. Plot the captured images, depth maps and infrared maps. You can create a matplotlib subplot to arrange these nicely.
4. Plot a 3D scatter plot of the pointcloud utilising depths captured by the depth sensor, and coloring the pointcloud with the RGB data from the color sensor.

☐ Challenge Task

1. Explore the pykinect2 module on GitHub. This API contains the ability to grab numerous data streams from the KinectV2. There are several methods and attributes available within the PyKinectRuntime class designed to capture this information.
2. The KinectV2 is able to track the state of the hand gestures of an individual in frame. How many gestures are available, what do they look like? What is a "Lasso" gesture?
3. How many bodies can the KinectV2 track simultaneously? How many joints on the human skeleton does it track? How does this correlate to the `bodies` and `bodies.joints` attribute and `body_joints_to_color_space` method?
4. The KinectV1 and V2 use a Random Forest to identify body parts. What is a Random Forest? Have a read of the paper, "Real-Time Human Pose Recognition in Parts from Single Depth Images" (Shotton *et al.*, 2011), a seminal paper on the use of the depth sensor for body-part tracking.