Sprint Retrospective, Planning and Daily Stand Ups

# Week 1

Due to current conditions my project plan has changed slightly. I will first work on implementing an object detection and recognition algorithm and then I will attempt to simulate a robot with a stereo camera. I am doing the object detection first so that I can do that in conjunction with researching robot simulation software that will be adequate.

On this first week I intend to set up a git hub and link my project supervisor to it.

Following that I will start playing around with object detection and recognition algorithms in Jupiter notebook.

Because of the lack of hardware I have decided to try to use Webots (a Robot Simulator) to simulate a robot with a stereo camera on the top. I will need to retrain any object recognition algorithms once the hardware has been received.

## Monday

I spent most of the day researching YOLO and how to use the libraries. I set up a git hub and linked my supervisor to it. But due to internet problems I could not accomplish as much as I intended.

Tomorrow I hope to have a preliminary YOLO implementation.

## Tuesday

<https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/>

<https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/>

<https://www.pyimagesearch.com/2017/11/06/deep-learning-opencvs-blobfromimage-works/>

<https://pjreddie.com/darknet/yolo/>

<https://arxiv.org/pdf/1804.02767.pdf>

Today I managed to implement YOLO using Darknets pretrained model. It uses the webcam currently to detect objects in the live feed.

Tomorrow I hope to have made a simulation environment. Maybe begin to move the object detection code over.

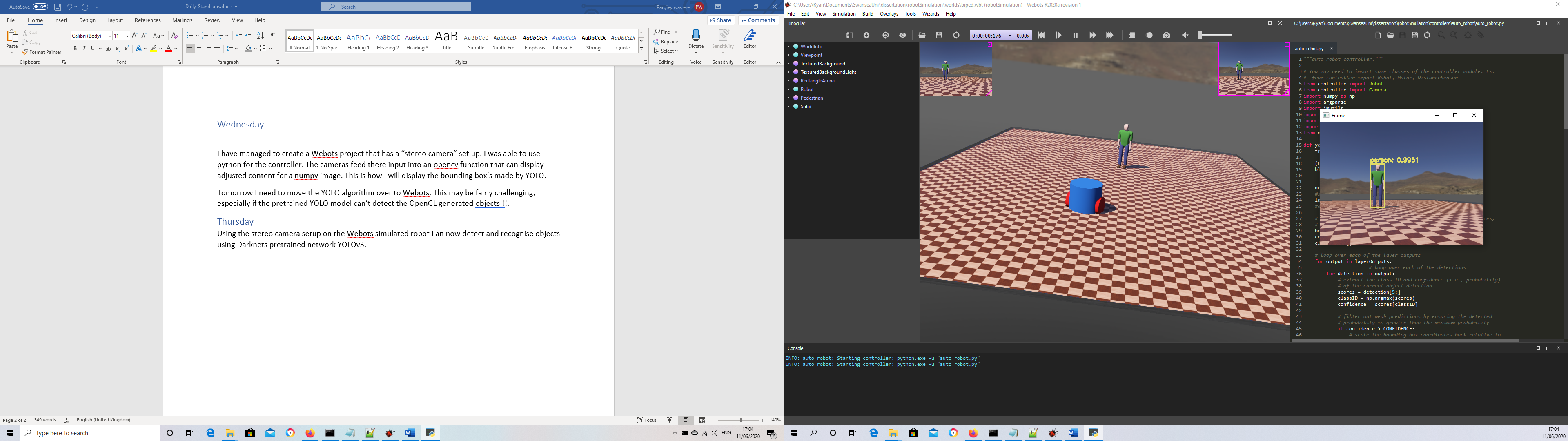
## Wednesday

I have managed to create a Webots project that has a “stereo camera” set up. I was able to use python for the controller. The cameras feed there input into an opencv function that can display adjusted content for a numpy image. This is how I will display the bounding box’s made by YOLO.

Tomorrow I need to move the YOLO algorithm over to Webots. This may be fairly challenging, especially if the pretrained YOLO model can’t detect the OpenGL generated objects !!.

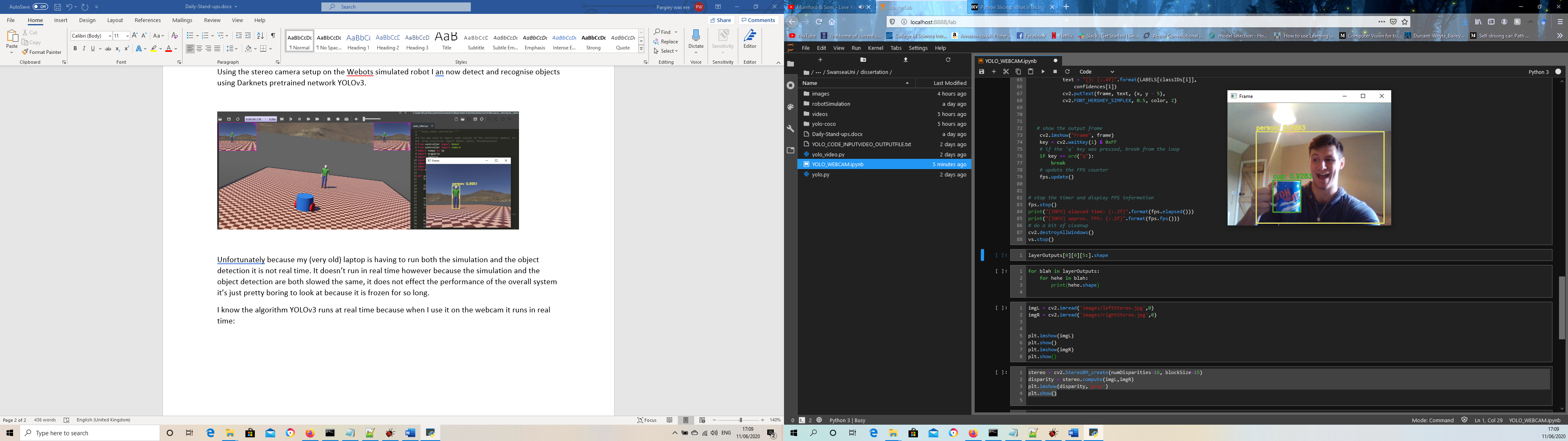
## Thursday

Using the stereo camera setup on the Webots simulated robot I an now detect and recognise objects using Darknets pretrained network YOLOv3.

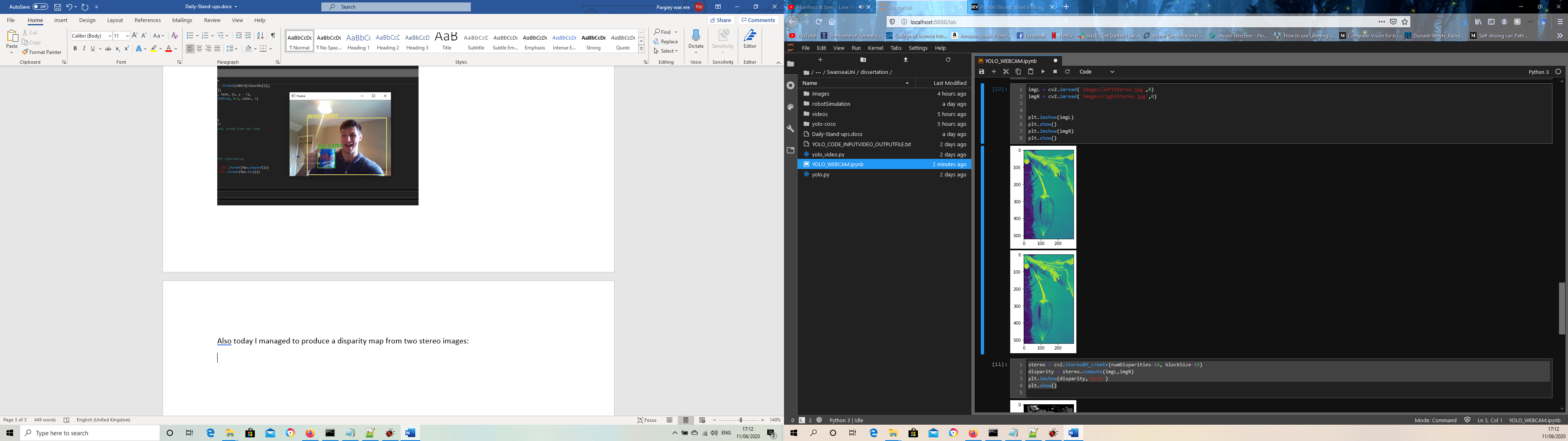
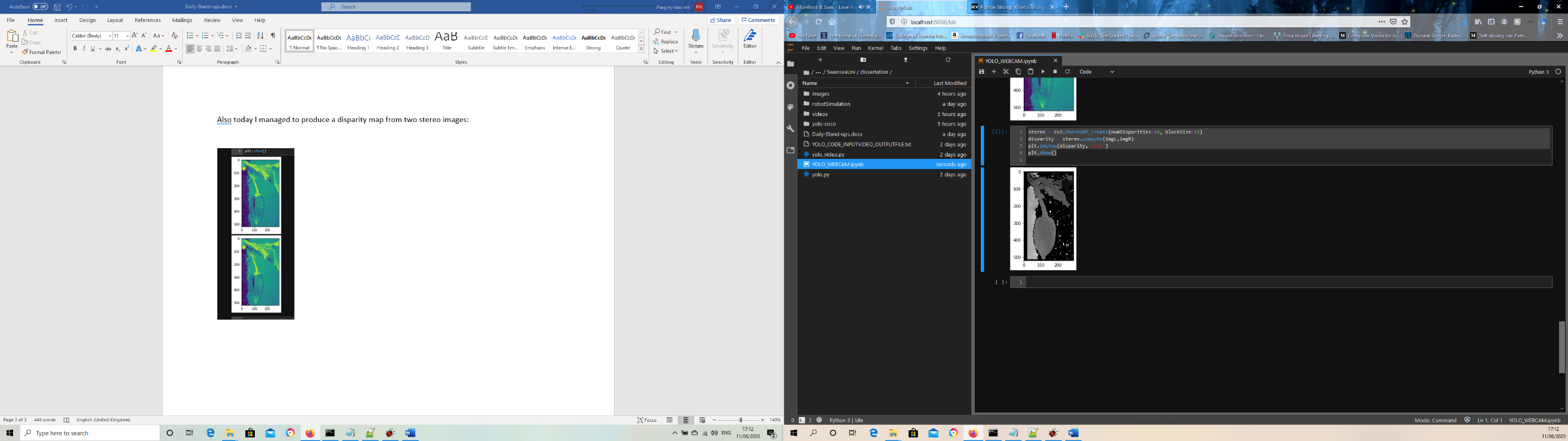


Unfortunately because my (very old) laptop is having to run both the simulation and the object detection it is not real time. It doesn’t run in real time however because the simulation and the object detection are both slowed the same, it does not effect the performance of the overall system it’s just pretty boring to look at because it is frozen for so long.

I know the algorithm YOLOv3 runs at real time because when I use it on the webcam it runs in real time:



Also today I managed to produce a disparity map from two stereo images:



This code was then able to be transferred to my simulation:



There appears to be a fair bit of noise on the depth map, this might be because of the OpenGL rendering but if it becomes a problem later on I can try to add some post processing to reduce the noise.

Tomorrow I will spend a small amount of time cleaning the code up on the simulation and the notebook. The main task for tomorrow will be to play around with the object detection. Now that I have got it working it might be a good idea to train some models myself or even try different algorithms.

Tomorrow will be a bit of a play day whilst I experiment.

## Friday

<https://www.learnopencv.com/training-yolov3-deep-learning-based-custom-object-detector/>

Today I played around with my object detector. I tidied the YOLO code slightly (needs some more work) and I implemented a faster R-CNN and (partially) implemented a Single Shot Detector.

Tomorrow I will finish off the tidying and experimenting, and start looking into object tracking a bit more seriously.

# Week 2

After spending the weekend looking into object tracking, the job for this week will be to begin implementing an object tracking algorithm.

## Monday

I have managed to partially implement a simple centroid tracking algorithm, I think I have over complicated it a little bit but it currently works for single objects (it knows that a single object is the same across frames) however it will take a bit more work to get it to recognise multiple objects.

Tomorrow I will aim to add multiple objects to my centroid tracker. Once this is done the framework for trying more complex trackers will be in place if needed.

## Tuesday

I can now track multiple objects, I still have an error where if two objects overlap the tracker will start tracking one object as the same object twice.

Tomorrow I will try to fix the problem with the tracker. And then clean up all the code.

## Wednesday

Problem solved!! Centroid tracking is complete, it’s limitation is that the id’s can swap if the objects overlap, but this is a limitation of the method rather than the implementation. The code has been broken up into methods, so it is slightly more readable.

Tomorrow I will look into implementing some more involved object tracking and start researching the use of the Kalman filter so that I can achieve SLAM.

## Thursday

I spent the day researching how to implement SLAM with the information I currently have.

<http://www.jdl.link/doc/2011/201911015261440619_detect-slam%20making%20object%20detection%20and%20slam%20mutually%20beneficial.pdf>

<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8482266>

<https://dspace.mit.edu/bitstream/handle/1721.1/36832/16-412JSpring2004/NR/rdonlyres/Aeronautics-and-Astronautics/16-412JSpring2004/A3C5517F-C092-4554-AA43-232DC74609B3/0/1Aslam_blas_report.pdf>

<http://www.luigifreda.com/wp-content/uploads/2016/10/vslam-main-slides-1.pdf>

## Friday

I again spent most of the day researching SLAM, I have found that I will use a feature detector to detect landmarks and use the object detector (YOLO) to categorize moving objects so they are ignored as landmarks for SLAM. This could change at a later date.

I have already used opencv to create a SIFT feature extractor. Tomorrow I will continue work with SLAM and hopefully begin using the Kalman filter on the results from the visual odometry.

# Week 3

This week I want to have most of SLAM implemented. I have spent most of last week trying to figure out how I will implement it, unfortunately I still need a bit of work to figure it all out completely.

## Monday

Today I looked up SLAM more and have created a new indoor environment on my robot simulator. I also turned the robot with a stereo camera into a PROTO so that it can be used in multiple worlds.

Tomorrow I need to translate my image coordinates into coordinates in cartesian space so that I can position the landmarks in a coordinate system that makes sense.

## Tuesday

I believe I have converted image coordinates into a world coordinate system by using this algorithm:

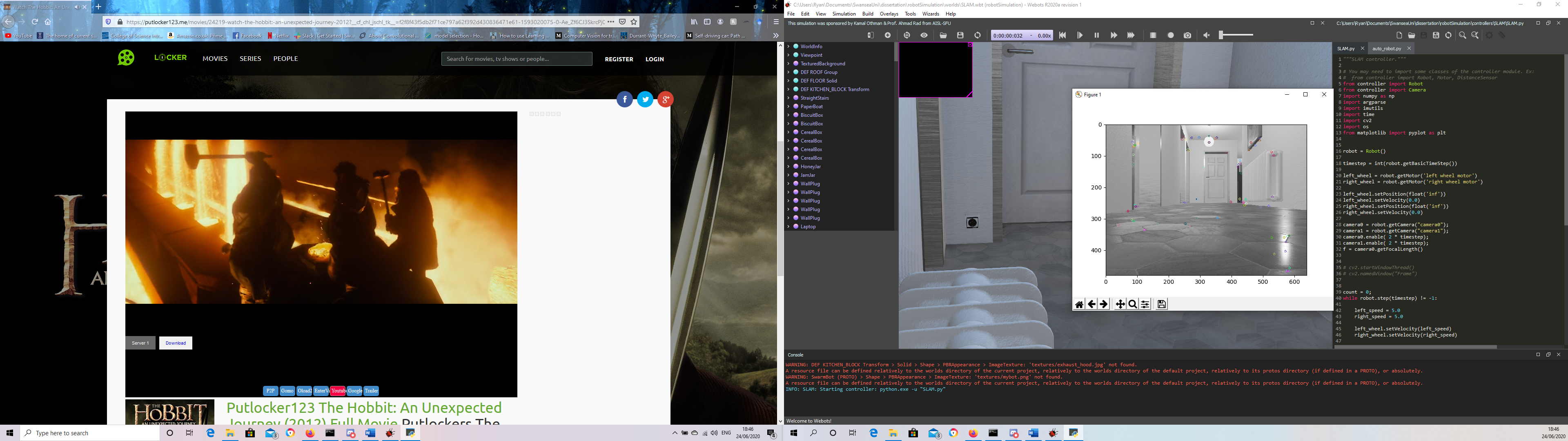
Where x1 and y1 is the x and y points in one of the two images (left), f is the focal length of both cameras and Z is:

Hopefully this has done what I think it has and given me the points in the real world, but I have had difficulties visualising the results to check.

Tomorrow I will try to use this conversion of pixel coordinates to find feature locations and match them between frames using Euclidean distance. After this I can begin work or the extended Kalman filter and the meat behind SLAM.

## Wednesday

Today I added sift feature detection to my simulation, this may be a waste of time as I will probably end up using object’s from the object detection. I also translated these points into real world coordinates.



Tomorrow I will do feature mathching and establish them as landmarks if they persist across multiple frames. If I have time after doing this the next step will be to begin work on the kalman filter.