1. **Object detection**

**Computer vision report**

YOLO was used for object detection. Out of 34 selected test frames, without any pre-processing, it detected a total of 318 objects of which 4 objects were erroneous, giving an accuracy of 98.74% on the test set. On some frames the algorithm failed to detect objects, however this mostly occurred on objects that were far away, occluded or distorted due to lighting or noise. Increasing the confidence threshold of YOLO from 0.5 to 0.55 reduced the total number of detected objects by 10. To improve the performance of the implementation practically, a GPU enabled YOLO implementation should be used, as this would reduce the average object detection speed of 2s by a large amount.

1. **Comparison of disparity calculation techniques**

Monocular, pyramid stereo matching and OpenCV’s Hirschmuller approach to disparity calculation are examined in the following section. The H. Hirschmuller algorithm [1] was the initial method used to obtain disparity. The result was a noisy image that worked well for close objects; however, it often had missing, or erroneous regions. The next approach was to utilize Pyramid Stereo Matching [2], which is a technique that leverages machine learning. The network was trained on the KITTI 2015 dataset [3, 4]. Finally, a monocular depth estimation network [5] was tested which required only a single image, rather than a stereo pair. This produced smooth results but wasn’t as accurate as the PSM-net stereo approach.

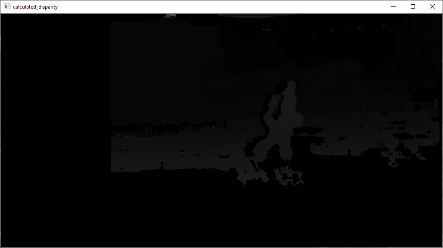
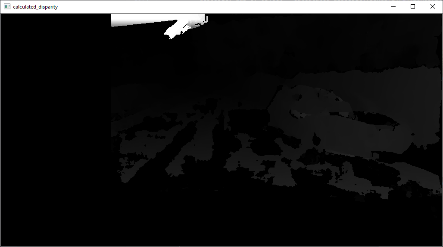


Figure 1: Disparity images calculated by the H Hirschmuller algorithm.

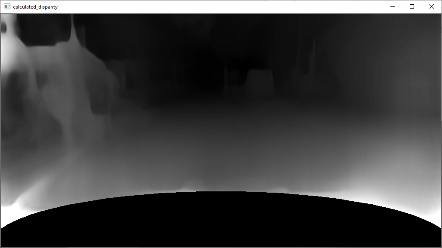
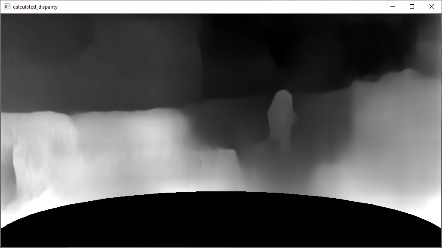
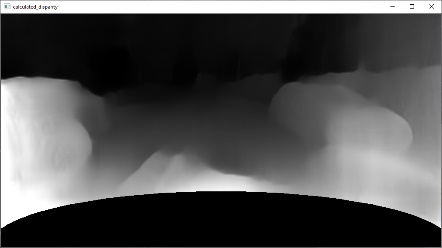


Figure 3: Disparity images calculated by the monocular depth network.

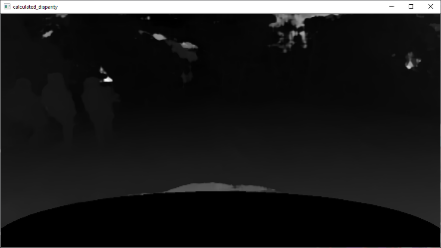
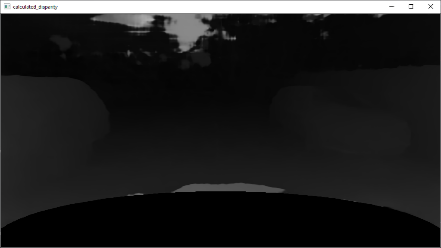
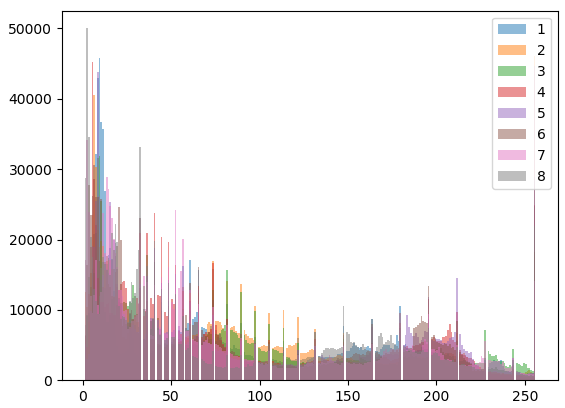
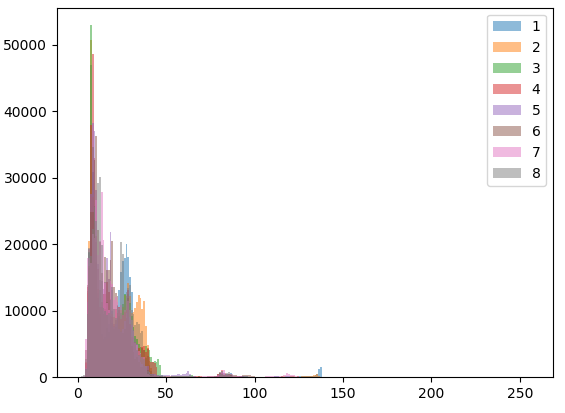
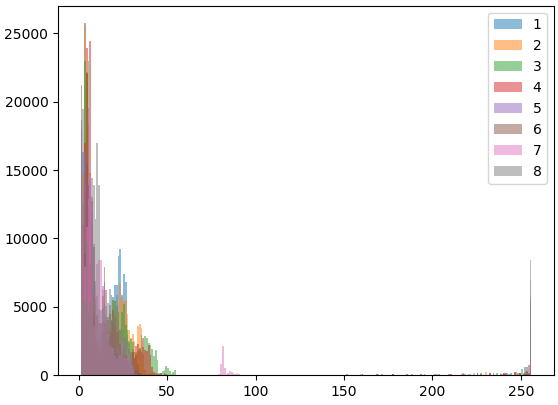


Figure 2: Disparity images calculated by the pyramid stereo matching network.

The main factors used to determine the effectiveness of the disparity solution were the ability to represent geometry on the input, the amount of noise, as-well as continuity in challenging frames.

Figure 4: Histograms of the above three approaches for the first 8 frames of the dataset



As shown in figure 4, PSMNet produces the best results regarding temporal consistency and amount of error in each frame.

1. **Estimating depth from disparity**

Taking the mean of the corresponding disparity map from the bounding boxes produced by YOLO (the object detection algorithm) means that you will also include regions that are in the background of the image, thus lowering the disparity of the object unintentionally. Initially using the bounding boxes given from YOLO, a semantic segmentation approach was explored as an alternative to detect the areas of objects and what the corresponding object was. This enabled the ability to obtain the disparity from regions which could be segmented from the background, giving a more accurate disparity value. Results of the segmentation overlaid with the bounding boxes of each object are given in figure 5. The segmentation approach works well for close objects, however as objects are smaller the further away it becomes more complex to segment them properly. In most frames the segmentation map can pickup objects in the immediate vicinity of the vehicle, sometimes encountering noise in darker or blurry regions.

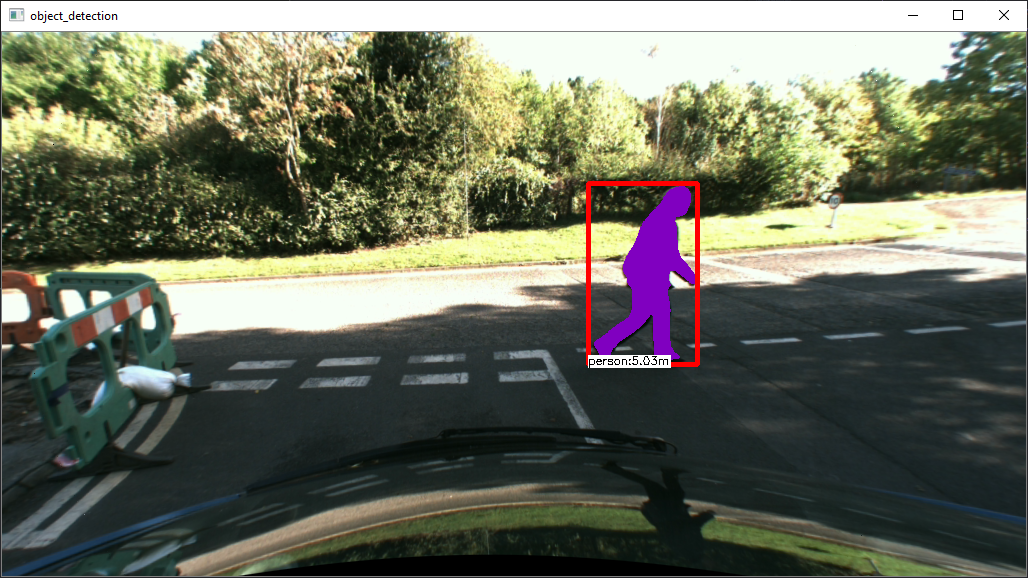
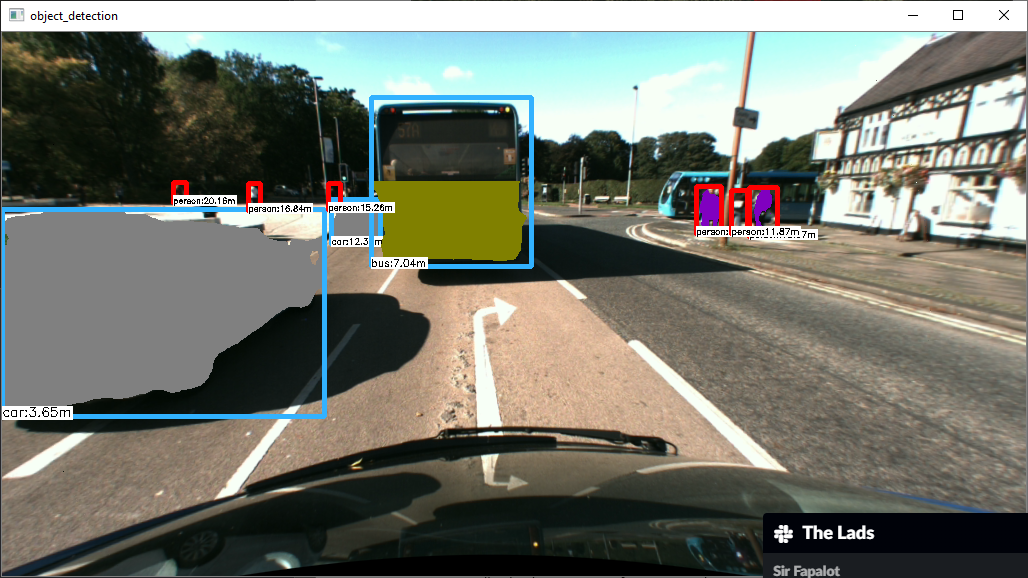


Figure 4: Semantic segmentation results overlaid inside the bounding boxes detected by YOLO.

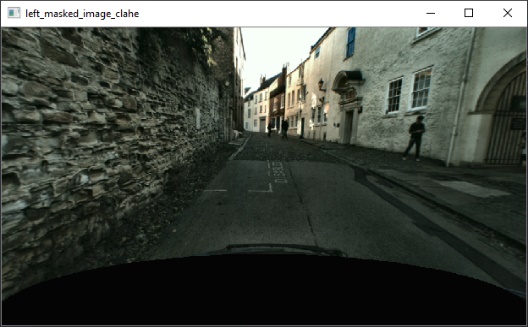
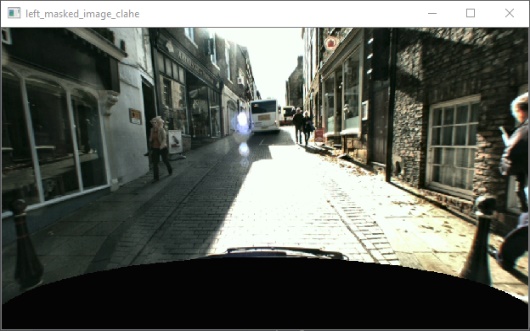
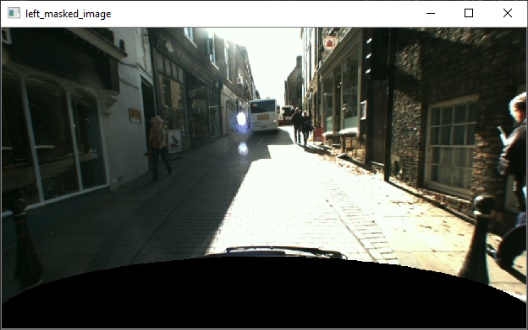
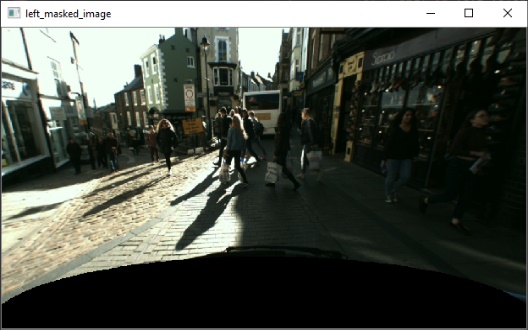
In addition to segmenting the bounding boxes, for buses, the segmentation is cropped to half its height as windows in buses can lead to discrepancies in disparity values.

1. **Heuristics & Optimizations implemented**



Figure 5: The top half of busses is cut off to not use the disparity calculated in windowed regions.

CLAHE (contrast limited adaptive histogram equalization) was also utilized to improve the contrast in frames where lighting was over-exposed. This led to easier recognition of objects qualitatively, and for the same test frames, YOLO was able to identify 9 more objects when using a confidence threshold of 0.55. Results of CLAHE (before and after) are given below:



When detecting objects in the scene, reflections on the bonnet of the car could cause erroneous objects to be detected. Therefore, an ellipse is sliced out of the left and right images when performing calculations on them.





Figure 6: The bottom of both images contains the bonnet, so an ellipse is used to mask it to prevent objects being detected inside the region.

Finally, the disparity map often contained some irregularities due to lighting effects and other possible causes. To remediate this issue, I limited the disparity maps values to a minimum and maximum threshold of the region’s standard deviation, in the hope that outliers would be removed. This thresholding approach was not generalized however, as some outliers cover large regions and therefore are classed as normal in the segmented region. Results of the disparity before and after attempting to remove outliers are given below:

1. **Conclusion**

The per frame runtime is 2.865 seconds. The solution works well for distances up to 16m, however once objects get further away their distances are often under-predicted. For this application, accuracy of objects at larger distances isn’t as important as those that are nearby – however a possible improvement is to train using varied data and explore how scaling could lead to increased accuracy. It also handles varied lighting conditions; however, the solution may not perform well in drastically different environments such as snow due to reflections. A further extension to the current implementation would be to use Generative Adversarial Networks for reducing the effect of motion blur.

1. **References**