The design and implementation of the object detection and distance ranging algorithm combines snippets of code from 'yolo.py' [1], 'stereo_disparity.py' and 'stereo_to_3d.py' [2]. After configuring variables such as threshold values for object identification and camera calibration for stereo depth estimation, the algorithm runs through each stereo pair. Since dense stereo ranging is integrated to project image pixels into three dimensional coordinates, a disparity map for each stereo pair must be generated. Therefore, both stereo images are converted into grayscale. Before computing the disparity image, the grayscale images are reduced from 8-bit grayscale to values ranging between zero and sixty-four. By lowering the resolution of the grayscale image, the disparity map can be computed far quicker without a noticeable drop in accuracy. Once mapped, the algorithm reduces noise in the disparity image using OpenCV's 'filterSpeckles' [5] function by setting a maximum difference between neighbouring disparity pixels of 123.

Next, the program implements 'yolo.py' methods to classify objects in the current image. Given that people and vehicles are the two categories of interest, only items from zero to eight in 'coco.names' require consideration. To avoid objects from being misidentified within each scene, a relatively high confidence threshold of 0.56 is set to identify and contain cars in blue rectangles and people in orange rectangles. Moreover, any object containing pixels below five hundred and over the centre of the image are disregarded as these are just reflections of objects from the front windscreen or the

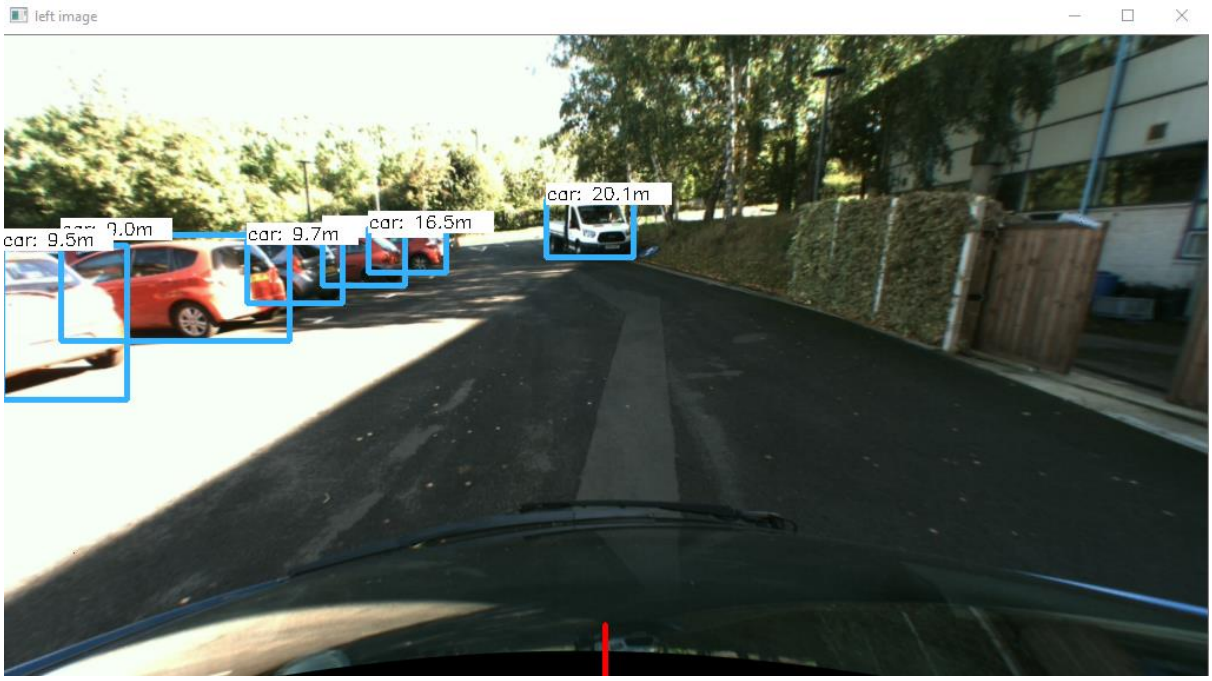

Figure 1. All rectangle containing any pixel over the red line (at the bottom of the scene) are removed from the output image.

car itself. See figure 1 for more information. Ideally, the red zone would cover the whole of the windscreen, however since each object is identified by a rectangle, some of these rectangles may cross parts of the onboard car's windscreen. By confining the red zone to a short line, the onboard car was still removed from the output without affecting neighbouring objects.

As dense stereo ranging is used to gather the distance between the onboard vehicle and the objects within the scene, each object must be mapped into three dimensional coordinates. The program tackles using the disparity image and the fixed camera parameters from the stereo calibration to convert each object's region of interest into three-dimensional point clouds. Pythagoras' theorem is

Figure 2a. Example of a distance calculation between the car and an object (person: 5.7m).

then applied to each individual coordinate in the point cloud to estimate the distance between the camera and every point found. As the region of interest is a rectangle in the two-dimensional image, not every point in the point cloud will correspond to a point on the object. Therefore, the minimum distance from the object to the camera is set not as the minimum value in the point cloud, but as the value of the lower quartile. If the object covers over 50% of the region of interest, the probability of returning a distance offset to the object is significantly reduced. Additionally, this lowers the chance of returning anomalous results. Figure 2 shows how the distance from the car to the person is set to 5.7 metres despite not being the shortest distance found. Alternatively, if no points are to be found in the point cloud, the dimensions of the region of interest are increased and the point cloud is recalculated.

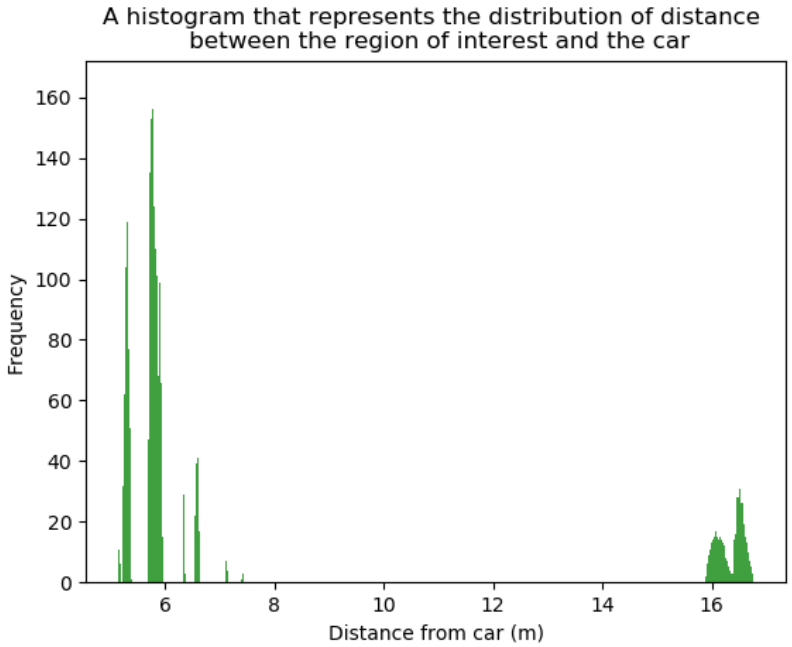

Figure 2b. A histogram representing the distribution of distance between the car and the 3d point cloud generated from the region of interest in Figure 2a.

Currently, all the distances constructed from the point cloud are in respect to the camera and not the car. Therefore, a further 1.5 metres is taken off the selected distance (the value of the lower quartile) to represent the minimum distance between the object and car. This effectively models the car as a sphere of radius 1.5, centered at the location of the camera. Figure 3 projects this sphere

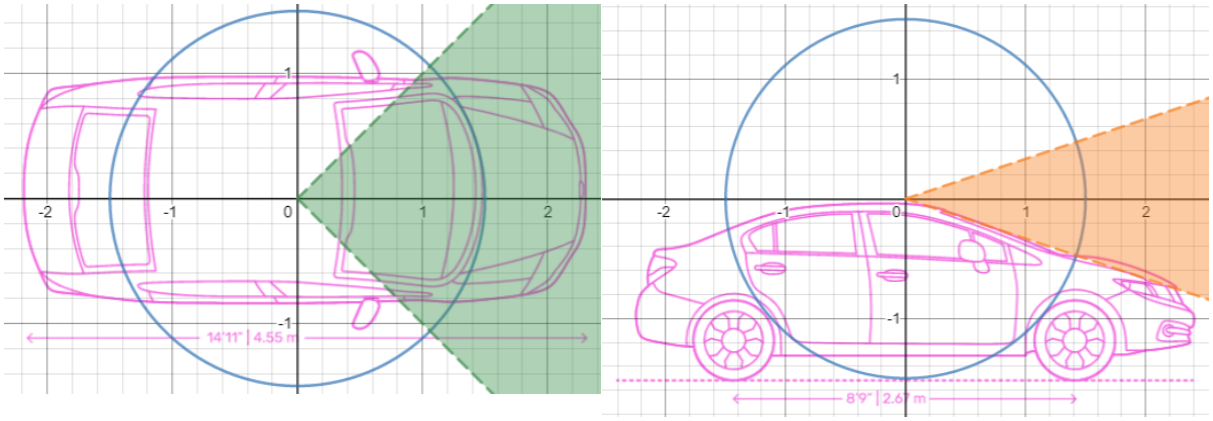

Figure 3. A two-dimensional projection of a 1.5m radius sphere onto a Honda Civic about the centre of the roof of the car. An approximated field of view is also shown within the shaded areas. [3, 4]

onto a regular shaped car in two-dimensions, assuming the camera is placed centrally above the windscreen of the car. Evidently, the major sections of the sphere that do not contain the car are to the sides and above. However, since the camera does not display the wing mirrors in any of the scenes, the angle of view can be limited to within the green area. Thus, the path of the distance between the object and car cannot cover the white empty section of the sphere to the left and right of the car, or from behind. Likewise, the field of view in the y-axis can be estimated to the orange area, and therefore once more reduces the possibility of having the distance cover an empty section of the sphere.

References:

1. Breckon, T. 2019. “Python examples cv.” Available from: https://github.com/tobybreckon/python-examples-cv
2. Breckon, T. 2019. “Stereo disparity.” Available from: https://github.com/tobybreckon/stereo-disparity
3. Desmos Inc. 2019. “Calculator.” Available from: https://www.desmos.com/calculator
4. DIMENSIONS.GUIDE, 2019. “Compact cars.” Available from: https://www.dimensions.guide/element/honda-civic
5. Opencv dev team. 2019. “filterSpeckles.” Available from: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#filterspeckles