**Report**

The goal of this project was to develop a system that accurately detected pedestrians and vehicles from an image and estimated the distance to these objects. I made use of object detection through YOLO, and implemented two approaches to stereo vision: dense and sparse. I consider the dense approach my primary solution, with sparse being an alternative approach to reach the same conclusion.

My Solution – Design Choices

Some improvements to the initial integration of dense stereo vision and YOLO:

**Histogram Equalisation**: Illumination in the provided dataset was an issue, so an attempt to reduce this before calculating the disparity map was made. I used CLAHE to do so – an advanced histogram equalisation method.

As visible, the disparity map on the right (CLAHE) is m

* **Stereo\_SGBM parameters**: Adjusting the input parameters for the SBGM object was critical in producing an accurate, but somewhat noiseless, disparity map. Many parameters allowed for the removal of later functions calls – e.g. filterSpeckles was integrated into the object itself through several of its founding arguments.

PHOTOS

* **Calculating distance**: For each object a Gaussian kernel was generated (based on the size of the object detected). The convolution of this kernel with a matrix of disparity information (centre: centre of object’s box) is then performed, and the resulting disparity is used to calculate distance.
* Altering the size of the kernel was imperative to success as without doing so, further objects would have poor distances or the time to calculate would significantly increase.

PHOTOS

* **YOLO edge detecting**: In an effort to detect more objects (primarily pedestrians) a simple edge enhancing technique was developed. A kernel was generated (for edge detection) and the image was filtered with this. Previously undetected objects were detected, however, some previously detected objects weren’t. To solve this, YOLO is performed on two images and the objects that appear twice are filtered out.
* A downside is increased time for program to run. Given more time and resources this would’ve been something to work on.

PHOTOS

Some changes that proved less effective:

* **Weighted least square (WLS) filter**: The use of the WLS filter along with the use of a left-right-consistency based confidence was attempted to improve the quality of the resulting disparity map (through ‘filling in the holes’). The hope here was a smoother, less noisy disparity map would subsequently improve distance calculations. However, I found that the removal of noise also removed much useful information from the map, with this causing a decrease in distance quality.

PHOTOS

**Qualitative/Quantitative evidence of performance (150):**

**Comparison between Sparse and Dense Implementations (150)**:

**Disparity map – with and without CLAHE**