Computer Vision Summative – kmgj58

King Him Cheung

My initial approach in tackling this problem was to implement the basic RANSAC algorithm to get a baseline of results.

Figure 1. Basic RANSAC – With attempt to optimise parameters

My results showed planes that at times fit nicely in the image, however there are many cases where lighting issues, grass and objects impacted negatively on these planes.

Figure 2. Basic RANSAC – Inaccurate plane detection and grass error

Table 1. Time for algorithms to output result (seconds) with parameters: 5000 features points close to plane; 688 Trials

Image	**Basic RANSAC (max 688 trials)**	**Basic RANSAC with Pre-processing**	**RANSAC with reduced Disparity Image**	**RANSAC with reduced Disparity and pre-processing**
1506942473.484027	12.89	13.46	8.61	9.17
1506942493.476187	10.94	11.79	8.67	8.26
1506943937.380111	12.39	12.01	7.64	8.39
1506943921.379931	10.60	12.39	8.35	8.67
1506943904.380412	11.28	12.72	8.54	9.71
1506943719.478566	9.54	11.23	7.52	8.91
1506943393.478088	11.67	11.80	8.54	8.37
1506943087.478034	15.13	13.36	8.13	8.73
1506942884.476479	11.28	13.97	8.16	8.87
1506942679.476572	Reached Max Trials	15.26	8.52	8.88
AVERAGE RUN TIME	11.13	12.80	8.27	8.80

From table. 1 we see that the average time to run our initial program on a small sample of 10 random images is approximately 11 seconds. This gives me a comparable result to later implementations.

Looking at the errors that came up from my initial program and the run time I looked at different approaches to fix separately these issues including:

1. Image Pre-processing to improve lighting issues in images
2. Reduction in size of feature points space to improve run time and improve accuracy of selected feature points

Image Pre-processing

I considered lighting-invariant colour spaces in my approach to improve the disparity image, initially I converted the left coloured image to the HSV colour space and reduced saturation and value to reduce the brightness of areas in the image and to enhance the areas covered by shadows. Results lead me to threshold these two areas of the colour space.

Lighting-Invariant colour space featured in [1] creates a grayscale image that appears to have reduced lighting issues however the algorithm to create this grayscale image requires a coloured image and because our rectified image is already in grayscale unfortunately no further advancements in this area could be made.

Pre-processing images has also led me to use histogram equalisation to improve the contrast of the image.

Figure 3. Lighting Invariant Colour Space [1] (left – light invariant image, right – original coloured image)

Figure 4. HSV Colour Space Thresholding (left – image thresholded, right – original coloured image)

Figure 5. Histogram Equalisation – (left – histogram equalised, right – grayscale image)

Feature Points Detection

After experimentation with different feature point detection algorithms I decided to use canny and orb detection. Beforehand, I used the entire disparity

map as feature points and searched through that to find points, canny and orb cuts down this search space and focuses only on important points. In addition, I removed the top section of the image and removed the colour green by removing the green range of hues in the HSV space this led to a significant speed up and improved performance. I.e. reduction in detecting grass.

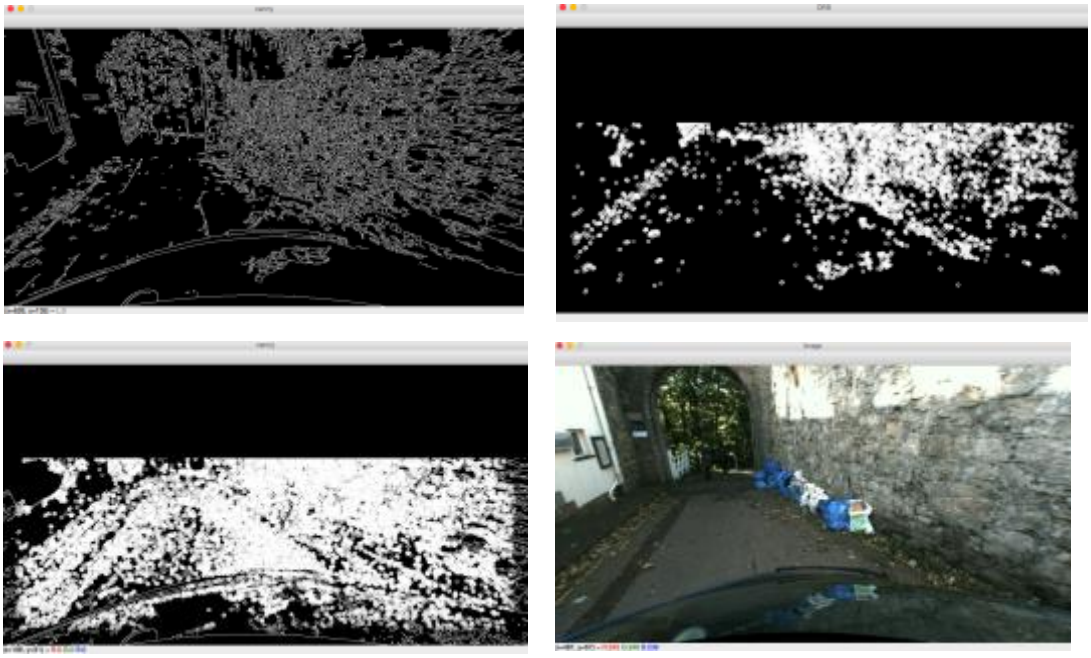

Figure 6. Feature Detection – Canny and Orb (top left: canny, top-right: orb, bottom-left: canny and orb with HSV Colour Space Change, bottom-right: Original)

Figure 7. Reduced Disparity image – reduced feature point search space

From table. 1 we can see a reduction in time using this reduced disparity image, moreover it is possible to reduce the number of feature points needed to verify the plane to get just as good results which will reduce even more the time taken to run.

Furthermore, to increase accuracies of our plane, I've decided to limit the plane angle so that it must meet the criteria that it must not be too vertical, this reduced the likelihood that walls being chosen as planes.

Finally, I calculated the normal to the plane and have added this to my end resulting image.

Figure 8. Normal Line on Plane

To tackle object detection, I looked at points that we're found within the drawing of the plane and selected points that we're in the Y direction above or below the plane by a large margin which suggests a standing object, however the results of this is not always accurate and have shown false positives, this detection feature looks at areas close to objects rather than directly landing on the object. To improve the points detected I removed points found on the front of the car by using a mask.

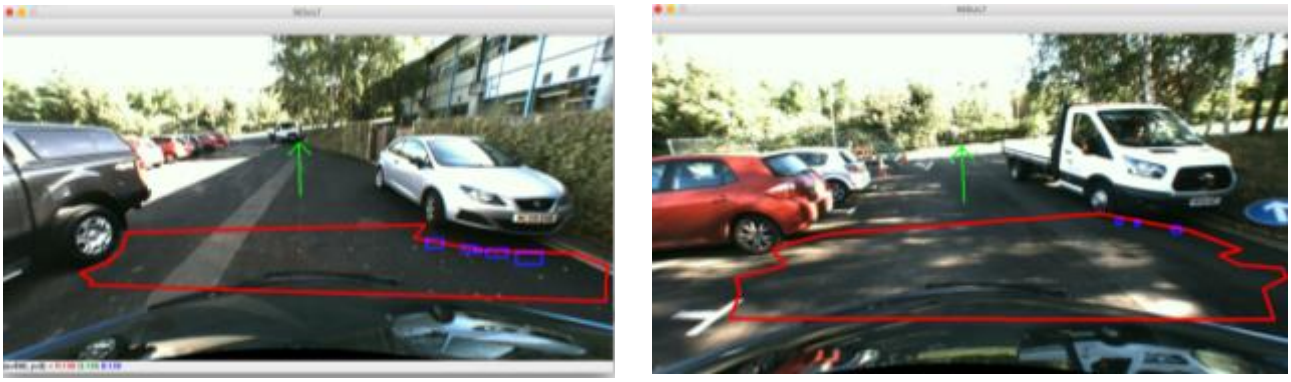

Figure 9. Object Detection

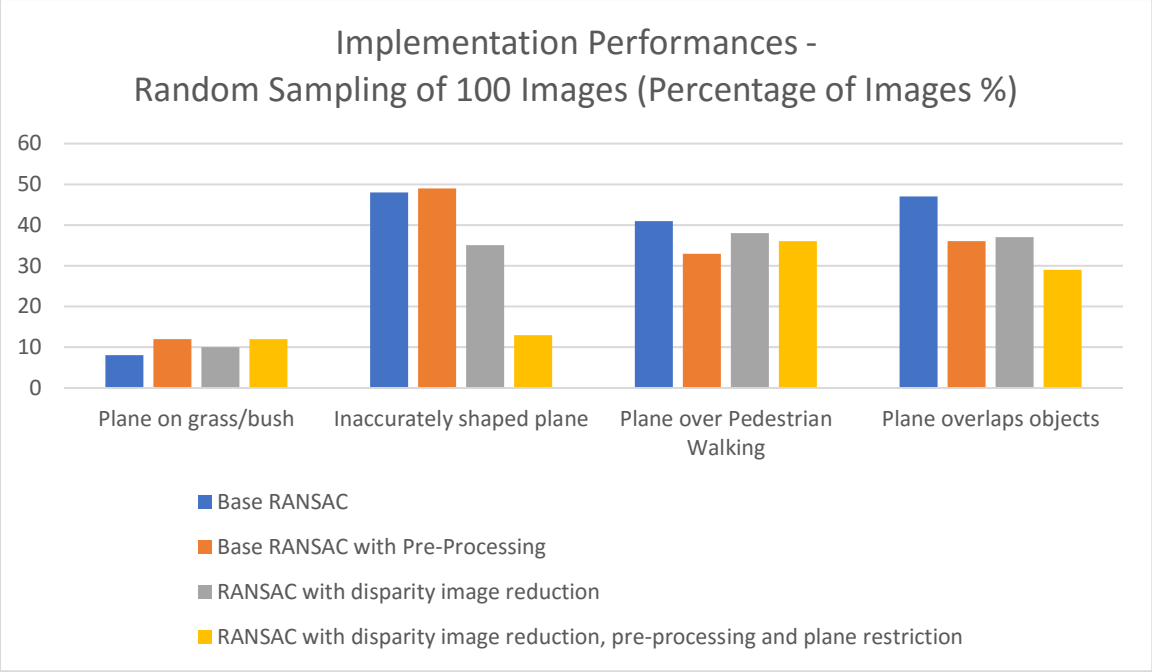

Figure 10. Sample Data of implementation Performance

As expected, after adding the plane angle restriction, inaccurately shaped planes were reduced significantly.

Overall, my implementation has found planes that do manoeuvre around objects quite well and it is much more robust to lighting problems and grass/bushes compared to the base RANSAC algorithm as can be seen by figure 10.

[1] W. Maddern, A. Stewart, C. McManus, B. Upcroft, W. Churchill, and P. Newman, "Illumination Invariant Imaging: Applications in Robust Vision-based Localisation, Mapping and Classification for Autonomous Vehicles," in Proceedings of the Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 2014.