COMP90086 Final Project

Group 55

Einon McGrory-Perich - 992697
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
emcgroryperi@student.unimelb.edu.au

Xing Yang Goh - 1001969
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
xingyangg@student.unimelb.edu.au

***Abstract*—This paper will demonstrate the effectiveness of solving the correspondence problem in disparity image generation from stereo images. A naive block-matching algorithm will first be implemented and then compared to a minimum cost path dynamic programming algorithm to produce a smooth disparity image. The algorithms will be evaluated against real-life stereo camera images.**

I. Introduction

Depth images are important tools for many applications such as obstacle avoidance, virtual reality, robotics and scene reconstruction. Stereo matching uses matched correspondences in two images taken at similar time instances in similar positions to construct a depth image. They do this by producing a disparity map from these geometric differences, by measuring the difference in matched pixels. This method can often be cheaper than other technologies such as Lidar imaging. This report will evaluate the performances of various stereo-matching algorithms.

II. Literature review

To construct a depth image from stereo images, a disparity map is needed. To do this, every pixel in one pixel needs to be matched to the most similar features, and the difference in their pixel values can be used to indicate how far a translation has occurred. To reduce the complexity of this problem, epipolar lines can be used to identify the line on which a corresponding pixel lies. For purely horizontal translations, such as what occurs with most stereo cameras, the epipolar line is flat in both images indicating any matching pixels lie on the same point in the vertical axis. Dhond [1] explores various stereo-matching techniques to improve the global consistency of local matches. Area-based matching using pixel intensity patterns and a similarity metric, however, this approach is sensitive to distortions from viewing perspectives which introduces contrast and illumination shifts. Additionally, occlusions in the image will result in erroneous matches.

To match pixels on the corresponding epipolar line, a metric is required that calculates how similar or dissimilar two pixels are. Hisham et al. [2] found that the sum of squared differences (SSD) and normalised cross-correlation (NCC) metrics provide good correspondences. SSD is less computationally intensive, requiring only square operations and pixel subtraction. Conversely, NCC is more computationally intensive involving multiplication, division and square root operations. Despite this, NCC is advantageous in area matching operations as it is more robust to changes in illuminations between scenes.

Optimisation methods, such as those proposed by Cox et al. [3] can then be applied to improve the quality of the disparity map. These methods typically consider the disparity of neighbouring pixels and attempt to prevent sudden changes by using dynamic programming methods to calculate a minimum cost path using cohesivity constraints and maximum likelihood functions, resulting in a smoother disparity image. Ma [4] further refines the noisy disparity images using a weighted median filtering algorithm for artifact removals.

III. Methodology

A. Dataset

To test the disparity image algorithms, stereo images from a large-scale data set of autonomous driving situations are used [5]. This data set provides a ground truth comparison of disparity with LIDAR imaging, which provides sub-pixel accurate depth measures. Note that for the applications of our disparity algorithms, the images are converted from RGB to grey-scale representation for computational efficiency and to reduce the computations on 3 channels to a single one. To illustrate our methodology process, we will illustrate examples with the stereo image pair in figure 1.

B. Simple Block Matching

The Naive approach to disparity matching iterates through each pixel of the left image with a defined window size and uses the stereo camera assumption that the right image experiences purely horizontal translation. By iterating through each column pixel of the same row in the right image, a corresponding pixel can be identified with the highest similarity metric. Using the pixel difference, we produce a naive disparity map of the image.

1) Block Size: The first parameter that can be chosen is the window size of each block. Larger blocks will include more of the surrounding pixels, before trying to find a similar block in the other image. Changing the value of this window can have noticeable impacts. Smaller window sizes use fewer neighbouring pixels, which often leads to incorrect matches

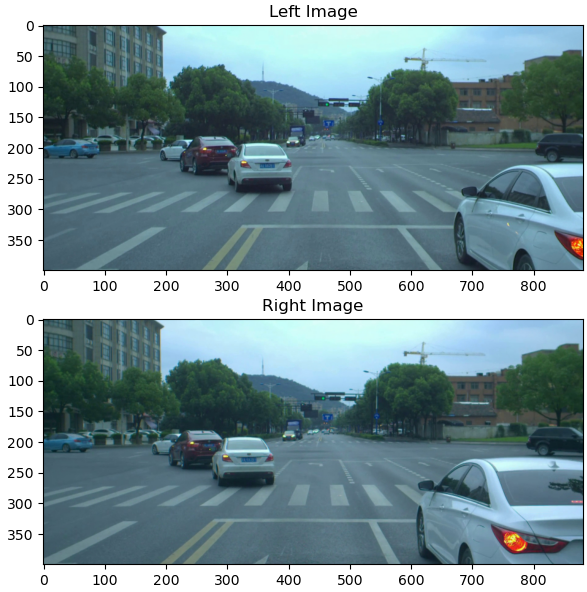

Fig. 1. Stereo Image used for Methodology

and a noisier image. Larger window sizes result in a smoother disparity image that lacks finer details. Figure 2 highlights the importance of appropriate window sizes, with the 5x5 window introducing noise over flat surfaces such as the road which is most likely due to their repeating textures. It does, however, capture the shapes of objects due to the sudden changes in depth. These erroneous matches are caused by insufficient surrounding information to distinguish textures during the block-matching process. Conversely, the 21x21 window size contains fewer incorrect matches with a smoother disparity profile along flat regions, however, lacks the detail that the smaller window provides as shown by the fewer overall contours.

2) Dissimilarity Metric: When comparing two blocks against each other, a dissimilarity metric needs to be calculated that determines how different the two windows are.

The first metric evaluated is the sum of square differences (SSD), which sums the square differences of every pixel in the two windows, and can be calculated with equation 1. The smallest SSD will then most likely correlate to the matched positions.

$$\sum_{i,j \in W} (I_l(i,j) - I_r(i+x, j+y))^2 \tag{1}$$

where W is the window dimensions, and I_l and I_r are the windows in the left and right images, displaced by x and y pixels due to the difference in scene.

The second metric is normalised cross-correlation (NCC), which takes the element-wise product of each pixel in the window, before dividing it by the product of each window's norm. This results in a normalised value that signifies how similar they are, as a value between 0 and 1. The equation is given in 2.

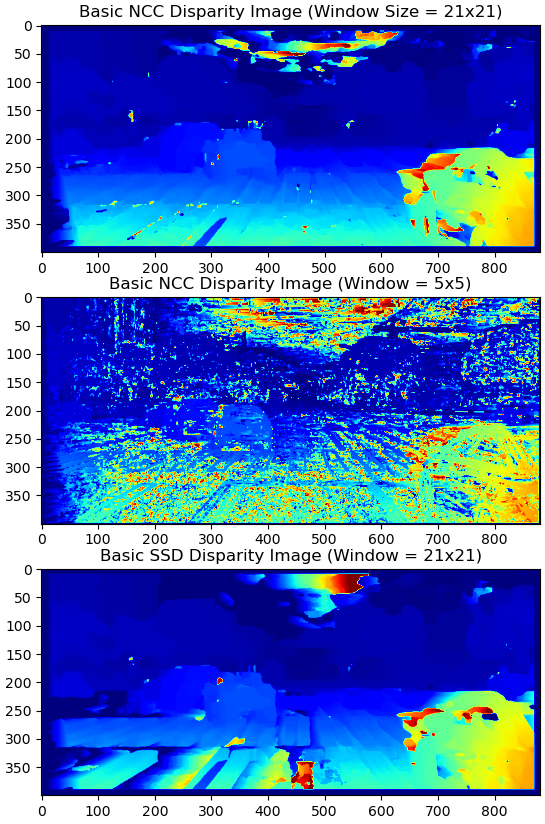

Fig. 2. Effect of Varying Window Size and Dissimilarity Metric

$$\frac{\sum_{i,j \in W} (I_l(i,j) I_r(i+x, j+y))}{\sum_{i,j \in W} (I_l(i,j))^2 \sum_{i,j \in W} (I_r(i+x, j+y))^2} \tag{2}$$

Comparing the normalised correlation and SSD measures in figure 3, we can see that the peak in the normalised correlation corresponds to the trough of the SSD. The performance from these measures can be seen in figure 2, where the NCC image produces much smoother disparity surfaces compared to the SSD metric. This difference could be caused by slight changes in illumination and contrast between the stereo images, which NCC is more robust to compared to SSD.

This, however, is still insufficient due to the spots of high disparity when there is no sudden change. Therefore, a smoothing term needs to be included.

C. Performance Improvements

1) Boundary size: Given that block matching iterates through the entire column of the right image for each pixel, we can set a boundary window for this operation to reduce the computational intensity of the operation. To determine an appropriate boundary size, we make the assumption that the corresponding pixel on the right image must exist to the left of the left image's pixel, given that the images are captured in the same instance. Additionally, we can limit the left boundary to 150 pixels, an empirical value determined by observing the difference in the pixel difference of close

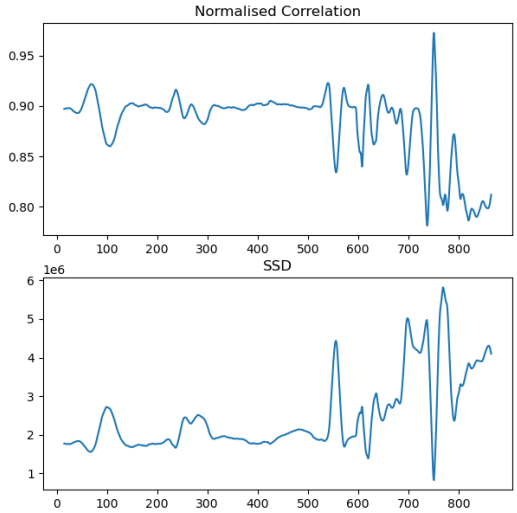

Fig. 3. SSD and NCC Dissimilarity on the Tail Light of Car

objects. This leads to a reduction in extreme pixel difference outliers, constrains the cost minimisation for the shortest path and improves computational efficiency.

2) Tensor operations: To improve the computation time of the matching algorithm, tensors were used which significantly improve the dissimilarity calculation, as the process simplifies to efficient matrix operations. A tensor operation was only used when matching a pixel on the left image, to the pixels in its corresponding region in the right image. This simple step reduced the processing time by more than a factor of 20 in some cases for the basic matching algorithm.

3) Multiprocessing: Since each pixel calculation can be performed independently, we can distribute the computational load to the CPU cores of the system by pooling the row iterations for parallel operations and joining the outputs to form the final disparity map. This further reduces the processing time by a factor of roughly 5.

D. Optimisation by Scanline Consistency

To smooth the image, a path through the disparity space image can be constructed which restricts sudden changes in disparity levels.

1) Disparity Space Image: A disparity space image (DSI) can be constructed for any line containing matching points. Every pixel in one epipolar line is matched to every pixel on the corresponding line in the other image, and the dissimilarity metrics are stored. They can then be presented in an image, shown in figure 4, where the pixels on the left image are shown on the horizontal axis, and the matched pixels are shown on the vertical axis. Regions with high dissimilarity are shown as white lines, with places of low dissimilarity in black. For NCC, the inverse was taken as a value to minimise. The disparity of any point on this graph can be calculated as its distance from the diagonal.

The right DSI in figure 4 has removed the boundaries not of interest, as negative disparities cannot occur due to the image shifting to the right, and a safe estimate for the maximum disparity had already been chosen as 150 pixels.

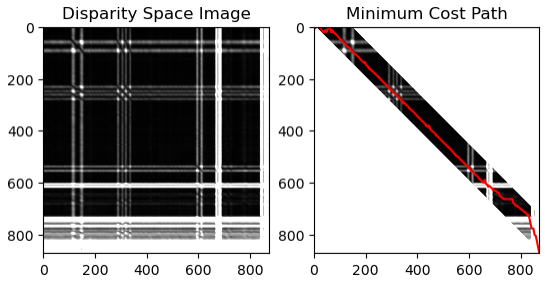

Fig. 4. Disparity Space Image and Minimum Cost Path

2) Minimum Cost Path and Cost Function: To identify the minimum cost path, a cost function needs to be used that considers adjacent pixels. One such function is to select the minimum of its direct neighbours, as it moves towards its target. The number of pixels it considers is a parameter that can be chosen. It was found empirically that the 5 nearest pixels produced a reliable low-cost path. If the reach was made higher, the path would move toward unsuitable local maxima, introducing noise in the final image. Too low a reach, and it would be unable to identify alternative paths with a lower cost.

A smaller window size of 7x7 was found to increase performance as the DSI had greater detail. Repeating textures did not have the same effect as seen in figure 3 as local minimums of adjacent pixels were preferred resulting in a more consistent image. Furthermore, since textures will have relatively similar correlation scores, they will not have a substantial influence on the costs for the shortest path.

Figure 4 contains a distinct black diagonal line throughout most of the left image, that stops at the intersection of the two white lines. This would be the most appropriate path to follow as it has a low total dissimilarity and is the most direct. When a minimum cost path algorithm was implemented, the final path followed this line before veering away to pass through the thick white bands where the cost is low.

3) Adaptive median filter: The disparity image in figure 5 produced by the minimum cost algorithm produces an accurate depth representation of the scene, with fine details for the trees and cars. However, there are horizontal streaking artifacts throughout the image. These artifacts are a direct result of the adjacent pixel reach constraint during the shortest path calculation, which prohibits sudden changes in disparities. Consequently, these horizontal streaks are produced when the shortest path has to take iterative steps for convergence towards a lowest-cost point.

To reduce the effects of this horizontal streaking, a median filter can be implemented, which takes the median pixel value in a specified window. This effectively permits large changes in disparity, such as at the edges of the car, since the median value in an appropriately sized kernel will correspond to the disparity level that the path converged to. However, note that a median filter will reduce the detail in the disparity map

and as a result, lead to smoother features. This median filter kernel size is influenced by the adjacent pixel reach value and has been empirically chosen to be 20x20 to produce the best results. On the sample images, the filter decreased the RMSE score by a noticeable amount, sometimes as large as 4 units.

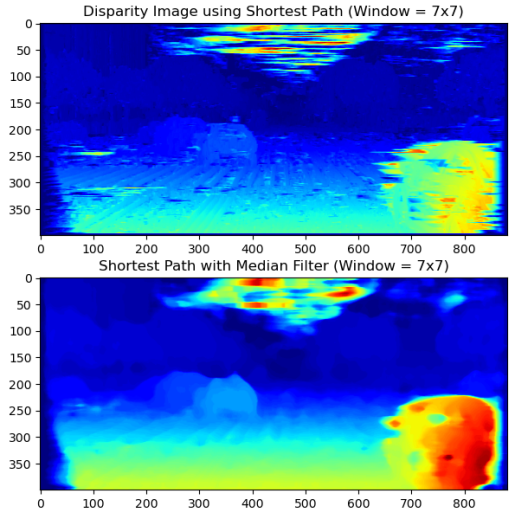

Fig. 5. Disparity Image using an Optimising Algorithm

IV. Results

Table IV shows the models performance to the ground truth lidar scan, using the metrics of pixel accuracy and Root Mean Square Error (RMSE).

TABLE I
Final Results

Metric	Image 1	Image 2	Image 3
RMSE	12.25	8.84	8.62
<4	0.90	0.89	0.92
<2	0.84	0.82	0.84
<1	0.68	0.66	0.67
<0.5	0.45	0.41	0.43
<0.25	0.24	0.20	0.22
Time (s)	64.5	52.7	58.8

The image used during the methodology section, shown in figure 5, had the worst RMSE but performed quite well in the pixel accuracy metrics. When the disparity image is observed, it produces an appropriate image where the depth is uniform, and the objects clearly visible and at an appropriate depth.

Image 2, shown in figure 6, had the best RMSE and similar subpixel accuracy. The disparity image also clearly shows nearby objects with an appropriate amount of detail.

Figure 7 shows image 3 which had the best RMSE accuracy, and produced a suitable disparity image.

All images processed in approximately 50-65 seconds, which was significantly faster than the inefficient methods due to the performance improvements. This sort of time would not be sufficient for most applications, such as autonomous driving. This could be made faster by using tensor operations on all processing, as well as by using a graphical processing unit (GPU) which has more computing power.

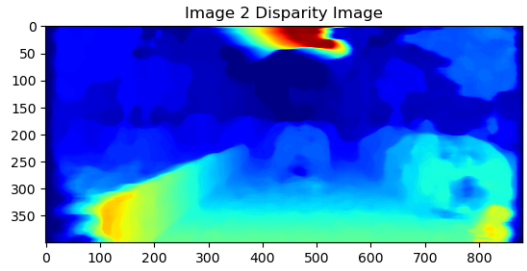

Fig. 6. Image 2

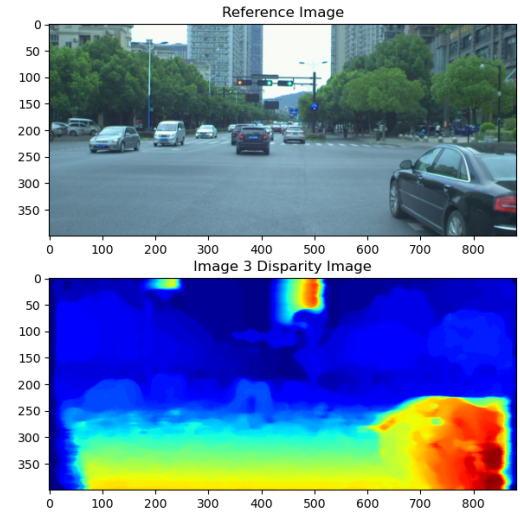

Fig. 7. Image 3

Similarly, in all of the images there is noticeable error in the sky, where the pixel disparities are large. This is most likely due to the lack of any distinctive features, and so the algorithm is unable to determine a suitable path or matching pixel. Changes to the cost function, such as by preferring a constant disparity could resolve this. This would not be a concern in the context of autonomous cars, as the sky can be disregarded.

V. Conclusion

The model produced demonstrated it was capable of creating reasonable disparity images with sufficient detail from the provided stereo inputs. The limitations and drawbacks of the simple block matching and dissimilarity metrics were considered, and an optimisation method was implemented that improved this. The final disparity images produced from the samples demonstrate good performance on the metrics, and clearly defined the depth of the scene.

REFERENCES

[1] U. R. Dhond and J. K. Aggarwal, "Structure from stereo-a review," *IEEE transactions on systems, man, and cybernetics*, vol. 19, no. 6, pp. 1489–1510, 1989.

[2] M. Hisham, S. N. Yaakob, R. Raof, A. A. Nazren, and N. Wafi, "Template matching using sum of squared difference and normalized cross correlation," in *2015 IEEE student conference on research and development (SCOReD)*. IEEE, 2015, pp. 100–104.

[3] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs, "A maximum likelihood stereo algorithm," *Computer vision and image understanding*, vol. 63, no. 3, pp. 542–567, 1996.

[4] Z. Ma, K. He, Y. Wei, J. Sun, and E. Wu, "Constant time weighted median filtering for stereo matching and beyond," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 49–56.

[5] G. Yang, X. Song, C. Huang, Z. Deng, J. Shi, and B. Zhou, "Drivingstereo: A large-scale dataset for stereo matching in autonomous driving scenarios," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.