Stereo Disparity with Local and Global Methods

Ze Pang
School of Computing and Information Systems
University of Melbourne
Melbourne, Australia
ZEP@student.unimelb.edu.au

Sanqiang Jiang
School of Computing and Information Systems
University of Melbourne
Melbourne, Australia
sanqiangj@student.unimelb.edu.au

***Abstract*—This paper aims to solve the problem of stereo matching by developing various correspondence methods to calculate the disparity map between two images and then estimate the performance with the ground truth image obtained using LiDAR. Throughout the paper both local and global correspondence methods have been discussed. For the local method, we use different variants of SSD and NCC methods and also SAD to find the disparity. After that, a new intrascanline search algorithm as global methods have been applied based on local methods to improve the performance.**

***Index Terms*—computational stereo, stereo matching, disparity**

I. Introduction

The problem of stereo matching is to find correspondences between a pair of images taken from the same object at different locations using the same camera. One of the most common ways to find the correspondence is to calculate the disparity between pixels in two images. The intuition is the object will move differently at a different distance from the camera. The shorter length can lead to a larger disparity in the image. Since the real-world object behaves differently as in theory, many problems occur in the study of computational stereo problems like occlusion. Also, the computational cost needed to be considered in some applications like object detection in self-driving cars. Therefore although stereo matching has been studied for decades since the 1970s, this is still an active research field in recent years.

There are two different general approaches to find the stereo disparities between correspondence pixels in one pair of images. The first general class of approach is the local methods. One of the most popular and efficient methods is block matching. The first part of this paper uses this method with various different metrics. Block matching finds the disparity between two images by maximizing the resulting matching scores from two patches of pixels in two images. The another general approach is the global method. The second part of this paper uses dynamic programming by finding the shortest path in the disparity space image (DSI). This method considers the whole scan-line at each time and gives better results than local methods.

II. Preprocessing

In the problem of stereo matching, it is desirable to have a non-verged stereo, i.e. the image planes are parallel. To check whether the given set of images satisfies this assumption, we plot epipolar lines on one set of images and observe they are parallel and epipoles are met at infinity. Therefore there is no need to apply the rectification on these images.

One advantage of having a non-verged stereo is that the correspondence pixels on two images lie on the same scan line. This can help us better to find the disparity map with less computational costs since we do not need to search the entire image to get the corresponding pixels.

Another preprocessing step is to convert all the images into grey-scale. In this problem of stereo matching, finding correspondence pixels do not need channel information and will be very computationally expensive if channel information has been used to find the disparity.

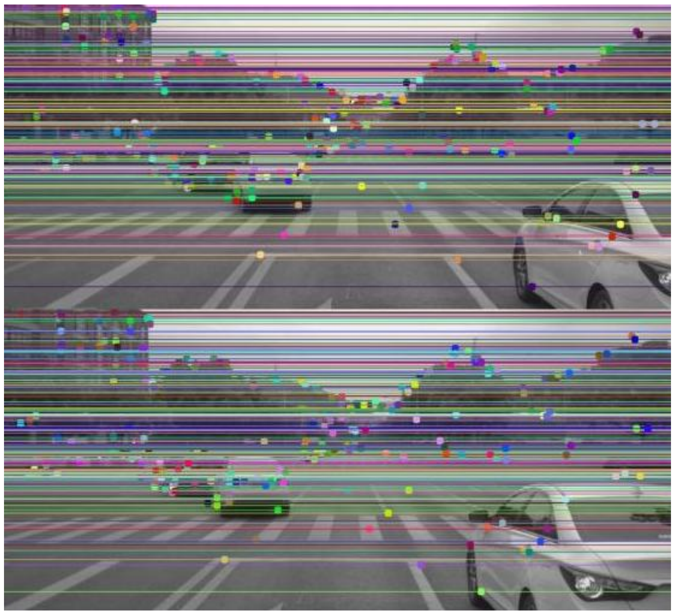
Fig. 1. Parallel epipolar lines indicate no need for rectification

III. Local Correspondence Methods

A. Metric and Formula

This section will introduce different metrics for block matching as a local correspondence method. Three basic metrics used to calculate the correspondence value are the Sum of Absolute Distance (SAD), Sum of Squared Distance

(SSD) and Normalised Cross-Correlation (NCC). Their basic formulas are given below (Chang et al., 2022 [1] Oliveier et al. 1993 [4]):

$$C_{SAD} = \sum_{x,y} |I_1(x,y) - I_2(x+d,y)| \tag{1}$$

$$C_{SSD} = \sum_{x,y} (I_1(x,y) - I_2(x+d,y))^2 \tag{2}$$

$$C_{NCC} = \frac{\sum_{x,y}(I_1(x,y) \times I_2(x+d,y))}{\sqrt{\sum_{x,y}(I_1(x,y))^2} \times \sqrt{\sum_{x,y}(I_2(x+d,y))^2}} \tag{3}$$

$$C_{ZNCC} = \frac{\sum_{x,y} \Delta I_1(x,y) \cdot \Delta I_2(x-d,y)}{\sigma_1(x,y) \times \sigma_2(x+d,y)} \tag{4}$$

$$\text{where } \Delta I(x,y) = I(x,y) - \overline{I(x,y)} \tag{5}$$

$$\sigma(x,y) = \sqrt{\sum_{x,y} \Delta I(x,y)^2}, \tag{6}$$

Note that some literature will subtract the mean from each block during the estimation for the formula of NCC. In this paper, we will adapt the above formula for NCC and refer to the mean normalised version of NCC as ZNCC.

The basic metrics for SAD, SSD, NCC and ZNCC have been defined using above formulas. The value calculated for NCC and ZNCC are have range of [0,1] by Cauchy–Schwarz inequality. For ZSAD and ZSSD methods, we follow the same process to normalize mean and standard deviation as defined in (5) and (6) before taking into the formula. However the results from ZSAD and ZSSD are no longer have a range of [0,1]. There are other normalized version of SAD and ZSSD available, we accept the above formulas due to their simplicity and easy to fit in our block matching methods.

The block matching algorithm states that for each pixel on the left image, a pixel on the right image lays on the scan-line is the correspondence pixel if it has the largest correspondence value calculated using metrics defined above. Therefore, the C_{SAD} and C_{SSD} and their variants calculate the matching error and needs to be minimised during matching, and C_{NCC} and its variant need to be maximised to find the optimal disparity.

B. Implementation

During the implementation of the block matching algorithm, we observed it is significantly computational costly to loop over all the pixels on the left image and also need to compare the correspondence value between blocks of values. Therefore we decide to replace the pixel operation with the image operation and use convolution to accelerate the process. Taking NCC as an example, the new objective function has been defined below (f is a convolutional kernel):

$$\arg\max_{d \leq D} \frac{f * (I_1 \times I_2(d))}{\sqrt{f * (I_1^2)} \times \sqrt{f * (I_2(d)^2)}} \propto \frac{f * (I_1 \times I_2(d))}{\sqrt{f * (I_2(d)^2)}} \tag{7}$$

The new procedure is defined as follows: for each disparity d smaller than the defined maximum search range D, a new right image has been formed by shifting to right d pixels. Taking NCC as an example. First, we need to multiply the left image with the transformed right image to obtain another new image. Then an average kernel or kernel of 1s has been applied to the second new image forming the numerator of the objective function. The denominator has a similar process, except the new image is formed by multiplying the right transformed image with itself and needs to take a square root after the convolution. After obtaining a total of D convoluted values for each pixel on the final image, the optimum disparity for each pixel is the maximum of these values. This method can significantly save computation time due to the use of the Numpy and OpenCV library, and it only cost around a second to find a disparity map for two images.

C. Design Choice

There are various design choices during the implementation. In the original algorithm, we need to compare each block by taking the summation of each element. This is inflexible and relatively difficult to extend to tune the parameters like the block size and the weight of each component in the block. In this section, we will compare the performance of different metrics with different design choices.

The first design choice is the block size. The larger block size would include more intensity variation for reliable matching, but the effects of projective distortion need to be considered during the block size increase. Large windows are necessary to prevent incorrect matches in areas with minimal texture. Window-based stereo techniques perform poorly close to object boundaries. Since this window-fixed method assumes that there are similar disparities at all pixels inside the window if a window spans a depth boundary, some points in the window will be aligned with the foreground disparity, while the other points are matched at the background disparity [2]. The blocks used in this paper are all square and the size is odd. From Fig 2, the performance of SAD and SSD drop to minimum when the size is 35. However, the performance of NCC reaches its minimum at 95. After reaching the minimum, the performances start to increase, suggesting the optimal size for SAD and SSD is 35 and for NCC is 95. The NCC has the best performance when the block size is large enough. The reason could be the NCC has additional normalising terms thus more stable compare with another two and can get more information from larger block.

Figure3 shows the RMS error with different choices of the kernel for different methods. The overall performance of averaging filtering is better than Gaussian and Median filtering. So it's fail to reject that all pixels in the matching window are not equally important. The input images may influence the importance of pixels in the windows, and the images used in this project have simple backgrounds, clear objects, also each object is relatively large, so the complex operations can instead lead to the loss of information.

Figure4 shows the change of RMS error after applying different kernels on the resulting disparity map. In order to make results becomes smoother and have better performance, averaging, Gaussian and median filtering are tested. Averaging and Gaussian filtering can lead to continuous disparities and possible to compare 0.5 and 0.25 pixels disparities with the ground truth image. This operation improves the performance of all three methods. NCC in general has the lowest RMS error after performing smoothing, with RMS around 20.

D. *Evaluation*

The Table1 shows the RMS error and run time of one pair of images for different methods implemented in this project. In general the normalized method has better result than the unnormalized methods with the cost of extra time computing the normalizing terms. For all unnormalized methods, the RMSe are similar with each other, however, the ZSSD has the best performance with a lowest mean RMS error of 14.87 among the normalized methods. The NCC and ZNCC need have higher computational cost than the SSD and ZSSD, which also have higher cost than SAD and its variant.

The Table2 shows the fractions of pixels with errors less than 4,2,1,0.5 and 0.25 for six different algorithms implemented in this project. The SSD has the overall best performance with the most significant fractions in different precision of pixels. We observe that in general, we have a trade-off between low RMS errors and low fraction value. If we smooth the disparity map, the RMS error will decrease and fraction will also decrease, indicating we have general low disparity value but not accurate enough.

Because the signal (intensity variation) to noise ratio is low, a poor disparity estimate is produced when the window is too small and does not cover enough intensity variation. The position of maximum correlation or minimum SSD may not represent correct matching due to different projective distortions in the left and right images however, if the window is too large and covers a region where the depth of scene points (i.e. disparity) varies.

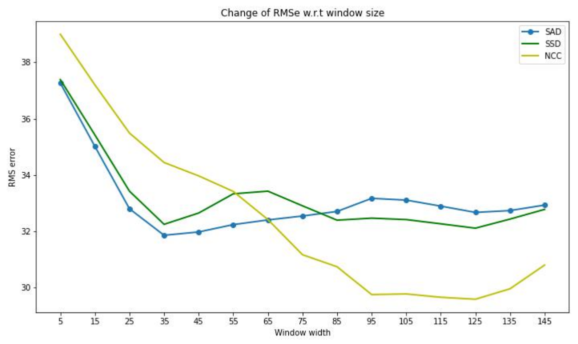

Fig. 2. The plot of performance with respect to the block size

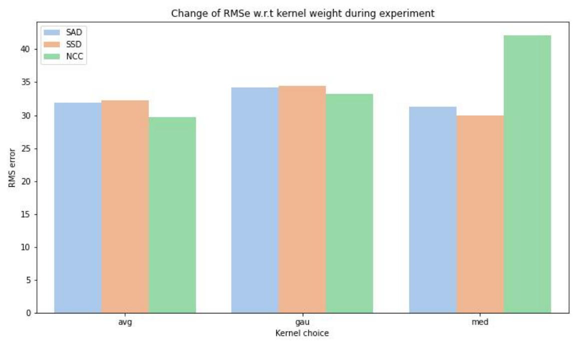

Fig. 3. The plot of performance with respect to the kernel choice for calculating correspondence values

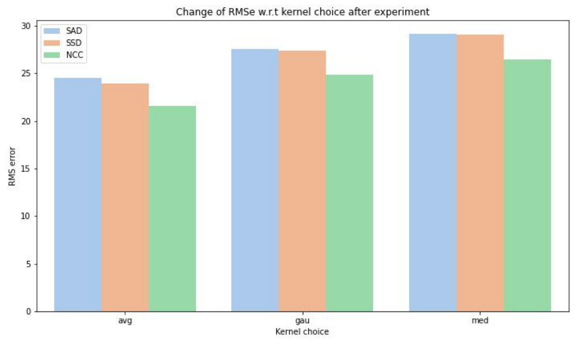

Fig. 4. The plot of performance with respect to the kernel choice on disparity map

	SAD	SSD	NCC
RMSe	18.36 ±2.57	18.07±1.95	18.77±2.40
time	0.93±0.10	0.94±0.13	1.16 ±0.13
	ZSAD	ZSSD	ZNCC
RMSe	17.53±0.90	14.87 ±1.74	17.79 ±2.22
time	1.08±0.14	1.10±0.18	1.21±0.22

TABLE I
PERFORMANCE FOR DIFFERENT METRICS

	SAD	ZSAD	SSD
Frac4	$21e-2 \pm 6.6e-2$	$17e-2 \pm 3.5e-2$	$22e-2 \pm 3.5e-2$
Frac2	$10e-2 \pm 3.2e-2$	$8.3e-2 \pm 1.9e-2$	$11e-2 \pm 2.5e-2$
Frac1	$5.0e-2 \pm 1.5e-2$	$4.0e-2 \pm 0.99e-2$	$5.5e-2 \pm 1.3e-2$
Frac0.5	$2.4e-2 \pm 0.77e-2$	$2.0e-2 \pm 0.5e-2$	$2.8e-2 \pm 0.6e-2$
Frac0.25	$12.5e-3 \pm 3.9e-3$	$9.9e-3 \pm 0.3e-2$	$14.0e-3 \pm 3.0e-3$
	ZSSD	NCC	ZNCC
Frac4	$20e-2 \pm 4.5e-2$	$14e-2 \pm 6.0e-2$	$15e-2 \pm 5.3e-2$
Frac2	$9.9e-2 \pm 3.0e-2$	$7.1e-2 \pm 3.4e-2$	$7.5e-2 \pm 2.6e-2$
Frac1	$4.9e-2 \pm 1.6e-2$	$3.6e-2 \pm 1.7e-2$	$3.7e-2 \pm 1.3e-2$
Frac0.5	$2.4e-2 \pm 0.8e-2$	$1.8e-2 \pm 0.8e-2$	$1.8e-2 \pm 0.7e-2$
Frac0.25	$12.1e-3 \pm 4.3e-2$	$9.0e-3 \pm 4.3e-3$	$9.4e-3 \pm 3.2e-3$

TABLE II
FRACTION FOR DIFFERENT METHODS

IV. GLOBAL CORRESPONDENCE METHODS

A. *Intrascanline Search*

One of key issues related with the local correspondences methods is each time it only compares two pixels based on certain window size of their surrounding blocks, which ignore the global information and can lead to discontinuities in the disparity map in practice. The global correspondence methods used in this section is intrascanline search which can be used based on the previous local methods(Brown et.al 2003 [5]). The intuition is to add a strong constraint to enforce the learned disparity map to be continuous and avoid very unexpected large disparities. Let N be the width of image, D be the search range on the scan-line, the disparity space image (DSI) with size $N * D$ can be obtained after the convolution steps in the previous block matching. This DSI can be thought as sets of correspondence values of each pixel on the scan-line calculated using previous metric. In previous step, the disparity of pixels on this scan-line is the position d of DSI which maximize the correspondence values. In this intrascaline search method, an additional step has been done to improve the performance.

Since we want the disparity on the same scan-line to be continuous and also relatively small, the problem can be thought as finding an optimal path from one side of DSI to another side. Although using dynamic programming is best and most computational efficient way to find this optimum path for each scan-line in practice, this methods will add additional $O(N * D)$ time cost to the previous local methods. In this paper, after finding the optimal path, the disparity is obtained by the index d for each pixel on this path.

This paper we will consider two variants of intrascaline search using dynamic programming. The first one is the strong method (DP-ST), the disparity value for each pixel on the scan-line is the index d of correspondence values on the shortest path. Another method is the soft method (DP-SO), this method is not as strong as the previous one. In this method, the correspondence value will multiply a decimal value if it's on the shortest path and then perform the same procedure as the local method to select the optimal disparity. This method will give some extra tolerance and allow the disparities value not on the shortest path for the pixels. Note that the local method is equivalent to the DP-SO with multiplying value of 1, i.e. don't need to consider the shortest path and DP-ST is equivalent to DP-SO with multiplying value of $-\infty$ if the correspondence value is positive.

B. *Evaluation*

In this section we will compare 3 methods of DP-ST and DP-SO and ZSSD which has the lowest RMSe in previous section. The window size set for the matching block all sets to 35×35. We will no longer perform smoothing technique to the result after finding the optimal disparity map, which can help us better understanding the nature of difference behind. The multiplying value is set to 0.8 for DP-SO. As in the previous experiment, we apply the log transformation on the correspondence value to avoid any overflow or underflow. The results is shown in the Table 2.

Without the smoothing on the disparity map, the ZSSD value is around 33, which is much larger than the results given in the previous experiment. This shows the local method gives very noisy result and need to smooth in order to get lower RMSe. Compare with both RMSe and fraction of disparities less than threshold values, we found the DP-ST generally performs better than DP-SO, which also performs better than the ZSSD method. Remember that both DP-ST and DP-SO are computed based on the result of ZSSD, this experiment indicate that the global method of using dynamic programming to find the optimal disparity by finding the shortest path in the DSI can improve the local method on various metrics. Also we found add a relatively weak constraint and higher tolerance to disparities not on the shortest path will not gives a better performance.

Also in practice, smooth the result after finding the optimal disparity map will not have a large improvement on two global methods as on the local method. The reason is disparity values on each scan-line is very optimum and any smoothing will worsen the performance. However this global method only considers the optimum disparity only on one axis, the smoothing technique or other methods perform based on the second axis of DSI is necessarily to give a further improvement(Brown et.al 2003 [5]).

	ZSSD_NP-ST	ZSSD_NP-SO	ZSSD
RES error	25.56 ± 4.01	26.66 ± 4.11	33.31 ± 3.11
Frac4	$9.0e-2 \pm 3.6e-2$	$8.9e-2 \pm 3.6e-2$	$6.8e-2 \pm 3.3e-2$
Frac2	$3.9e-2 \pm 1.7e-2$	$3.9e-2 \pm 1.6e-2$	$2.9e-2 \pm 1.5e-2$
Frac1	$1.3e-2 \pm 0.6e-2$	$1.3e-2 \pm 0.6e-2$	$0.9e-2 \pm 0.5e-2$
Frac0.5	$1.3e-2 \pm 0.6e-2$	$1.3e-2 \pm 0.6e-2$	$0.9e-2 \pm 0.5e-2$
Frac0.25	$1.3e-2 \pm 0.6e-2$	$1.3e-2 \pm 0.6e-2$	$0.9e-2 \pm 0.5e-2$
Time(s)	206.16 ± 5.19	202.88 ± 2.63	1.13 ± 0.12

TABLE III
PERFORMANCE OF ZSSD WITH LOCAL AND GLOBAL METHODS

V. Suggestion for future improvements

In our experiments, the best RMS error is 14.87 using the smoothed results from ZSSD and 25.56 from the global method DP-ST without any smoothing. As explained in the previous section, we only consider one axis in the global method and can combine some other interscanline constraints to further imporve our result. In addition, some adaptive window size searching methods [3] could be used.

Apart from using the methods of block matching for the local method and dynamic programming for the global method, there are other correspondence methods can be used: gradient-based optimization and feature matching for the local method and intrinsic curves and also famous graph cuts for the global methods can be also taken into consideration. Moreover, deep learning methods such as AcfNet, PSMNet and PDS could also be used to get a better performance.

VI. Conclusions

In this project, we have processed stereo-matching algorithms on the pair of images from a moving vehicle for a disparity map. Both local and global methods like Intrascanline search was applied to improve the performance. Our best RMS error is given by ZSSD as the local metric and a variant of intrascanline search method with a strong constraint as the global method.

References

[1] Chang, Q., Zha, A., Wang, W., Liu, X., Onishi, M., Lei, L., Er, M. J., amp; Maruyama, T. (2022). Efficient stereo matching on embedded gpus with zero-means cross correlation. Journal of Systems Architecture, 123, 102366. https://doi.org/10.1016/j.sysarc.2021.102366

[2] Scharstein, D., Szeliski, R., Zabih, R. (n.d.). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001). https://doi.org/10.1109/smbv.2001.988771

[3] Kanade, T., amp; Okutomi, M. (1994). A stereo matching algorithm with an adaptive window: Theory and experiment. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(9), 920–932. https://doi.org/10.1109/34.310690

[4] Olivier Faugeras, Thierry Viéville, Eric Theron, Jean Vuillemin, Bernard Hotz, et al.. Real-time correlation-based stereo : algorithm, implementations and applications. [Research Report] RR-2013, INRIA. 1993. ⟨inria-00074658⟩

[5] Brown, M. Z., Burschka, D., amp; Hager, G. D. (2003). Advances in computational stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(8), 993–1008. https://doi.org/10.1109/tpami.2003.1217603