Mathematical Biosciences and Engineering

MBE, 18 (4): 3215–3226.
DOI: 10.3934/mbe.2021160
Received: 29 January 2021
Accepted: 04 April 2021
Published: 07 April 2021

http://www.aimspress.com/journal/MBE

Research article

Binocular stereo matching algorithm based on MST cost aggregation

Jian Zhang[1], Yan Zhang[2], Cong Wang[1], Huilong Yu[1] and Cui Qin[1,*]

[1] School of information and Communication Engineering, Nanjing Institute of Technology, Nanjing 211167, China
[2] Swissgrid Ltd, short-term Network Modelling Bleichemattstrasse 31, Aarau, 5001, Switzerland

* **Correspondence:** Email: qincui@njit.edu.cn.

Abstract: For common binocular stereo matching algorithms in computer vision, it is not easy to obtain high precision and high matching speed at the same time. In this paper, an improved binocular stereo matching algorithm based on Minimum Spanning Tree (MST) cost aggregation is proposed. Firstly, the performance of the parallel algorithm can be improved by reducing the height of the tree. Then, an improved Root to Leaf (L2R) cost aggregation algorithm is proposed. By combining stereo matching technology with parallel computing technology, the above method can realize synchronous parallel computing at the algorithm level. Experimental results show that the improved algorithm has high accuracy and high matching speed for binocular stereo vision.

Keywords: binocular vision, stereo matching, cost aggregation, minimum spanning tree

1. Introduction

Image sensors are widely used in automatic and sensing devices to realize object detection and recognition, thanks to the fast development of imaging and computer technologies. Stereo matching is of great importance for computer vision appositions, such as autonomous driving [1,2], target detection and recognition [3,4]. It also plays important role in the fields of robot navigation [5], and space detection [6,7]. Various stereo matching algorithms have been developed so far. While most algorithms are cable of meeting the matching quality, it tends to have the problem of high cost of computer time. Real time applications demand enhanced matching quality and reduced processing time. Therefore, applicable methods are needed to meet and balance these two seemly contradictory requirements for real time processing.

Stereo matching algorithms can be classified into global and local algorithms, normally. The

local algorithms tend to have lower computational complexity and lower accuracy comparing with global algorithms. Recently developed algorithms target to find balance between quality and computation time. Among those, the non-global stereo matching method based on Minimum Spanning Tree (MST) is proved to be efficient and advantageous [8]. The method however still suffers the problem of high computation time when deals with real time applications. Parallel processors and graphics processor (GPU) are great help to increase the computational speed. Especially parallel processor, having the advantage of low power requirements, are of many research concerns. There are some promising results have been obtained in the development of parallel-sequential matching algorithms.

In this paper, an advanced method is proposed to build the MST based stereo matching algorithm by Yang on parallel processors, which leads to reduced computation time at the same maintain the original algorithm's advantage of high accuracy.

2. MST algorithm and cost aggregation

The weighted connected graph $G=(V,E)$ is supposed to have n vertices, $|V=n|$. Then, the spanning tree of G is a minimal connected subgraph $G'=(V',E')$ of G. where, V' satisfies $V'=V$ and E' satisfies $E'\subseteq E \wedge |E'|=n-1$.

If G' is the MST of G, the E' satisfies $E_0 \subseteq E$ for any E_0. When $|E_0|=n-1$, the equation (1) will always hold.

$$\sum_{e'\in E'} W(\mathrm{e}') \le \sum_{e_0\in E_0} W(\mathrm{e}_0) \tag{1}$$

where, $W(e)$ represents the weight of an edge e.

The MST of a connected graph can be calculated by Prim algorithm or Kruskal algorithm [9]. The difference between these two algorithms is that the Prim algorithm starts from an origin, and continuously expands its range to select the edge with the smallest weight; The Kruskal algorithm looks at the whole, and constantly selects the smallest weight edge from the whole.

In most local stereo matching algorithms, the matching cost of a pixel p is calculated within a certain domain window of p. In other words, the matching cost is determined by the pixels which are located inside the domain widow. The pixels which are outside of the domain widow have no influxes on the matching cost.

In order to make the calculation of matching cost global, that is, all pixels in the image can affect the matching cost of the pixel, image I can be regarded as an undirected weighted connected graph $G=(V,E)$. The vertex set V is all pixels of I and edge set E is the relationship between adjacent pixels of I. Therefore, G is a 4-connected grid graph. The weight of the edge (u,v) of G is defined as the gradient of the pixel gray value as shown in equation (2).

$$W(u,v)=W(v,u)=|I(u)-I(v)| \tag{2}$$

According to the structure of MST, for the two points p and q in the image, if q is far away from p and the color differences between q and p is large, the number of path hops between q and p in MST is large and the influence on the calculation of matching cost of p is weak.

Therefore, the distance $D(p,q)$ between the two points p and q in the MST can represent not

only the spatial difference of pixels, but also the intensity difference of pixel values between them. The definition of similarity $S(p,q)$ between p and q is shown in equation (3).

$$S(p,q) = S(q,p) = \exp(-\frac{D(p,q)}{\sigma}) \tag{3}$$

where, σ is the adjustment parameter of similarity. Therefore, under MST, bilateral filtering function used in cost aggregation stage can be extended to equation (4).

$$C_d^A(p) = \sum_{q \in c(p)} S(p,q) C_d(q) = \sum_{q \in c(p)} \exp(-\frac{D(p,q)}{\sigma}) C_d(q) \tag{4}$$

where, $C_d^A(p)$ represents the aggregation cost of pixels p at parallax d, $C_d(q)$ represents the matching cost of pixels q at parallax d, $c(p)$ represents the node connected with p.

The similarity of two pixels in an image depends on the distance between the two nodes in the MST. Therefore, the MST can be organized into a tree structure. The leaf to root (L2R) cost aggregation process is used to calculate the cost of each pixel affected by the nodes in its subtree. The root to leaf (R2L) cost aggregation process is used to calculate the cost of each pixel affected by nodes other than those in its subtree in MST. After L2R and R2L processes, the cost of each pixel affected by all other pixels can be obtained. The final matching costs were aggregated over each influencing pixel and then over each pixel included.

3. An improved BFS algorithm

Breadth first search (BFS) is a search algorithm in graph structure [10]. The algorithm starts from a source node, traverses the child nodes of the node in turn, and then traverses the child nodes of these child nodes, and so on, until traversing the complete graph. Therefore, BFS algorithm needs to use an open closed table to record the traversed nodes. The open records the nodes to be traversed, and the closed records the traversed nodes. Similar to the BFS algorithm, Level Synchronous Parallel BFS (LSP-BFS) algorithm maintains three node sets: visited node set V, current level node set C, and next level node set N. The algorithm takes out the elements in C and places the adjacent nodes in N in parallel. This process is iterated until N is empty set. Before the next iteration process, let $C = N$, $N = \emptyset$.

Since the process of cost aggregation algorithm based on MST is carried out on the MST of the reference image, it is necessary to reduce algorithm to tree structure BFS algorithm. The BFS algorithm is mainly used for the general graphs which may have loops. The tree structure is an acyclic graph. The nodes in the tree structure have clear partial sequence relations. When a node is accessed, the parent node of the node must have been accessed, and the child node of the node must not have been accessed. Therefore, it doesn't need to check whether the adjacent points have already been accessed in the tree structure BFS algorithm. It can save the closed container.

3.1. LSP-BFS algorithm for tree structure

BFS algorithm on tree structure can omit closed container. Similarly, LSP-BFS algorithm on tree structure can omit the visited node set V. In the extended traversal range, it only needs all the child

nodes to join the next level node set N. It is not necessary to determine whether the child node has been traversed. The pseudo code of the improved LSP-BFS algorithm is shown in algorithm 1.

Algorithm 1. TREE-LSP-BFS (T, s).

```
1   C := emptySet
2   level := 0
3   s.lev := level
4   N := [1]
5   while N != emptySet do
6       C := N
7       N := emptySet
8   foreach c in C do /* parallel execution */
9           access c
10      foreach n in CHILD(c) do /* parallel execution */
11              N := N union [1]
12              n.lev := level + 1
13      end
14   end
15      level++
16  end
```

(a) (b) (c) (d)

Figure 1. Test examples.

It can be seen that LSP-BFS algorithm has two characteristics:

(1) Each layer requires additional synchronization. Synchronization is a time-consuming operation. If the MST height is too high, more synchronization operations are required.

(2) The number of parallel operations in a BFS layer depends on the number of nodes in that layer. The BFS layer with more nodes is more parallel, which can take better advantage to the performance of parallel devices. There are four images in Figure 1. The four images in are converted into MST. The vertex corresponding to the upper left pixel as the root node. A tree is constructed with the root node. The results of the number of nodes in the tree hierarchy are shown in Table 1. The node hierarchy distribution is shown in Figure 2. The statistical method is to sum the number of nodes per 50 layers. In the distribution map, the abscissa is the serial number of the hierarchy (numbering every 50 levels), and the ordinate is the number of nodes in the hierarchy (summing the nodes of every 50 layers). It can be seen from Figure 2 that the number of nodes in higher level and lower level is less, and the number of nodes in middle level is more. Therefore, in general, the node distribution of tree structure is sparse in the higher level and lower level, and the middle level is more denser.

Table 1. Tree level statistics of the test examples.

Test examples No	(a)	(b)	(c)	(d)
Nodes number	110592	168750	168750	166222
Number of MST layers	2619	2915	3457	2370

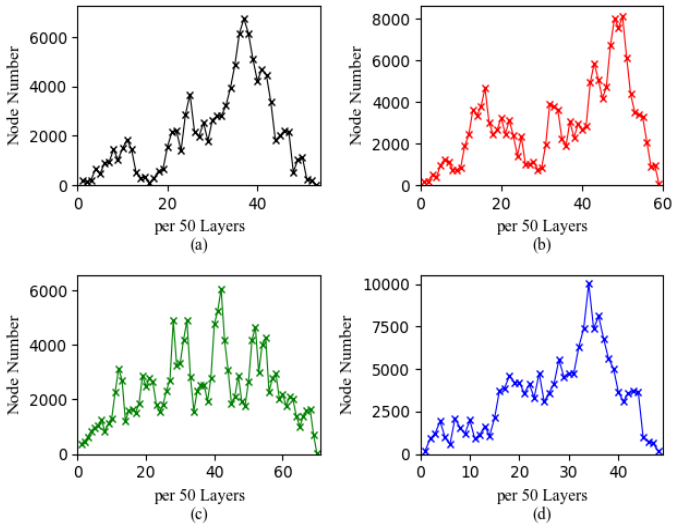

Figure 2. Node distributions of each layer in the test examples.

According to these two characteristics, in order to give full play to the performance of parallel devices, the parallel BFS algorithm needs to reduce the height of the tree, and increases the number of nodes in each layer of the tree. As shown in Figure 3, tree (a) and tree (b) originate from the same acyclic graph, but the root node of tree (a) is v_1, the root node of tree (b) is v_2. The height of tree (a) and (b) are 5 and 4, respectively. The node density in the second layer of tree (a) is significantly lower than that in the tree (b). Therefore, for the same spanning tree, when the root nodes are different, the lower the height of the tree, the higher the nodes density in the middle level.

When the number of nodes in the same layer is large enough, LSP-BFS can play the largest role. Therefore, in order to make LSP-BFS algorithm can be applied to solve the aggregation cost of MST, it is necessary to reduce the height of tree. If the middle node v_r of the longest path of acyclic graph is taken as the root node of MST, the MST with the lowest height can be obtained. Therefore, it is necessary to find the longest path of acyclic graph. A simple method is to use BFS twice to find the longest path. The basic process of the algorithm is as follows: first, let any point v_1 of the graph be starting point, use BFS to find the point v_2 which is farthest from v_1. Then from the point v_2 start, use BFS to find the point v_3 which is farthest from v_2. The path of v_2 to v_3 is the longest path of the graph.

This algorithm can find the longest path, but it is need to traverse all of the longest paths to find v_r. This process takes extra time. In order to solve this problem, this paper proposes a pruning method. The node v_r can be found out in twice level traversal time by using the pruning method.

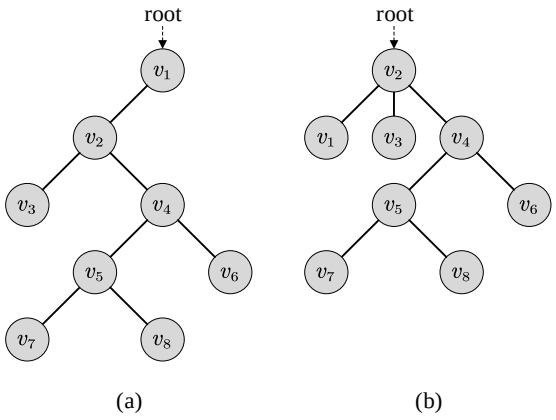

Figure 3. The height of the tree.

In an acyclic graph, nodes with only one adjacent node are called branch nodes. The node v_r can be found by deleting these branch nodes. For a given acyclic graph $G = (V, E)$ and a source node s, the process of the algorithm is as follows:

(1) Starting from the source node s, all the branch nodes are searched by BFS, then all the branch nodes are put into the branch node set N of the next layer;

(2) Let the branch node set $V = N$, $N = \emptyset$. For each node v in V, if the adjacent node of node v is not in the next level branch node set N, then v is added to N and deleted from G.

(3) Step (2) is performed continuously until the number of nodes in G is 1 or 2. If the number of remaining nodes is 1, the node is υ_r. If the number of nodes is 2, any one of them can be regarded as υ_r.

Algorithm 2. CUT-BRANCH (G, s).

```
1   V := emptySet
2   N := emptySet
3   foreach v in BFS(G, s) do
4       if number of v.Nbr() == 1 then
5           add v in N
6       end
7   end
8   while G.size() > 2 do
9       V := N
10      N := emptySet
11      foreach v in V do
12          if v.Nbr() is not in N then
13              add v.Nbr() in N
14          end
15          remove v from V
16      end
17  end
```

The pseudo code of pruning method is shown in algorithm 2. In this method, the time complexity of finding v_r is $O(2l) = O(3l) = O(l)$. where l is the longest path length of G. Because G is traversed twice, the time complexity of pruning method is $O(2l) = O(l)$. The time complexity of the algorithm for finding the longest path through BFS twice is also $O(2l) = O(l)$. Therefore, the performance of the pruning algorithm is equivalent to that of BFS twice method. It does not need to find v_r on the longest path, so the pruning method can reduce one traversal time, and the algorithm design is more simpler.

For the test samples in Figure 1, after reducing the height of the tree by using pruning method, the statistics of nodes for each layer are shown in Figure 4 and table 2. In the Figure 4, the black dotted line represents the node distribution of each layer when the tree height is not reduced; the red solid line represents the node distribution of each layer when the tree height is reduced. It can be seen from Figure 4 that as the tree height decreases, the density of nodes for the middle layer also increases.

It can be seen from table 2, the tree height of test examples (a) and (b) is reduced by about 45%, and the tree height of test examples (c) and (d) is reduced by about 25%.

Table 2. Tree level statistics of test examples after reducing tree height.

Test examples No	(a)	(b)	(c)	(d)
Nodes number	110592	168750	168750	166222
Number of layers for the tree height is not reduced	2619	2915	3457	2370
Number of layers for the tree height is reduced	1562	1503	2596	1756
Level reduction percentage	40.389%	48.439%	24.906%	25.907%
Root node coordinates	(61, 165)	(142, 270)	(1, 218)	(69, 181)

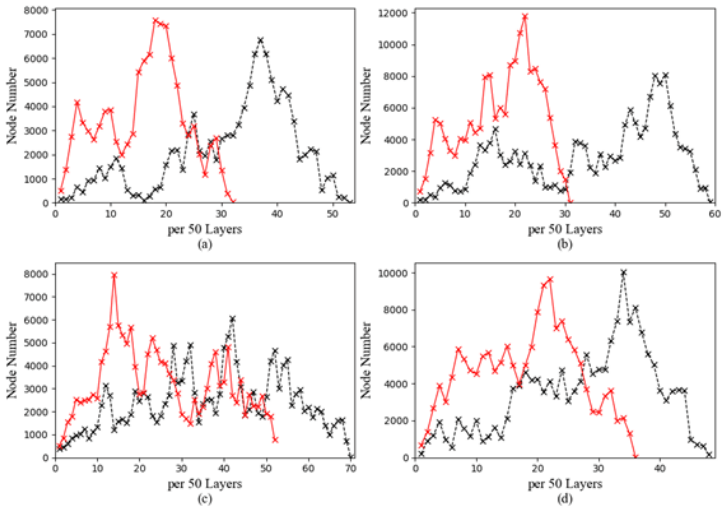

Figure 4. Node distribution of each layer in the test examples.

3.2. Improved L2R cost aggregation algorithm

The acyclic graph G and a source node s are obtained for a certain image. The improved L2R cost aggregation algorithm can maintain a queue Q and a linked list L, where L is the storing linked list. The process of the algorithm is as follows:

(1) Put s in the team;

(2) Use the variable l to save the length of Q and insert a new node N at the end of L;

(3) Set an element from Q as e, insert the e into the tail of N, and queue all the child nodes of e;

(4) Perform step (3) l times;

(5) Continue to perform steps (2) to (4) until Q is empty;

(6) Traverse L from the tail to the head. When each node of L is traversed, the aggregate cost of all nodes in the node is calculated. This step can be calculated in parallel.

Obviously, the nodes in the linked list L store the nodes of each layer of G. After traversing G, the length of L is the height of the tree with s as the root node. Variable l stores the number of nodes in each layer. The pseudo code of the algorithm is shown in algorithm 3.

Algorithm 3. REFINE-L2R (G, s).

```
1   Q := [1]
2   L := emptySet
3   while Q.size() > 0 do
4       l := Q.size()
5       L.add_new_node()
6       for i in [1 : l] do
7           e := Q.pop()
8           L.newNode.add(e)
9           foreach n in set of children of e do
10              Q.push(n)
11          end
12      end
13  end
14  for p in [end node of L : start node of L] do
15      foreach n in p do /*parallel execution*/
16          compute aggregation cost of n
17      end
18  end
```

4. Experimental results and analysis

All the algorithms are implemented and tested on Windows 10 operating system computer by using Visual C + + and NVIDIA CUDA. The hardware conditions of the computer are Intel (R) core (TM) i5-6300HQ CPU @ 2.30 GHz, NVIDIA GeForce GTX 960m graphics card and 8.00 GB memory. Middlebury stereo vision test set is widely used test set in academia. it includes test sets for binocular vision, multi vision and other technologies. Cones, Teddy, Tsukuba and Venus binocular vision test sets are used in this paper, as shown in Figure 5. The first row images are the reference images of the test set (the image of the left imaging system); the second row images are the target images of the test set (the image of the right imaging system); the third row images are the real disparity images. The parameters of the test set are shown in Table 3.

The process of parallel cost aggregation based on MST is as follows: (1) median filtering; (2)

calculating the weight of edges; (3) reordering by the weight; (4) generating MST by the Kruskal algorithm; (5) pruning and constructing tree; (6) L2R process; (7) R2L process.

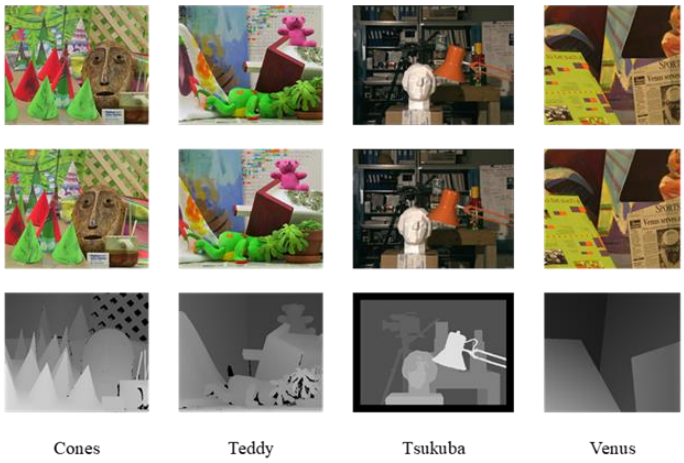

Figure 5. Middlebury test set.

Table 3. Test set parameters.

Test set	Cones	Teddy	Tsukuba	Venus
Size	450 × 375	450 × 375	384 × 288	434 × 383
Disparity Maximum disparity	60	60	16	32

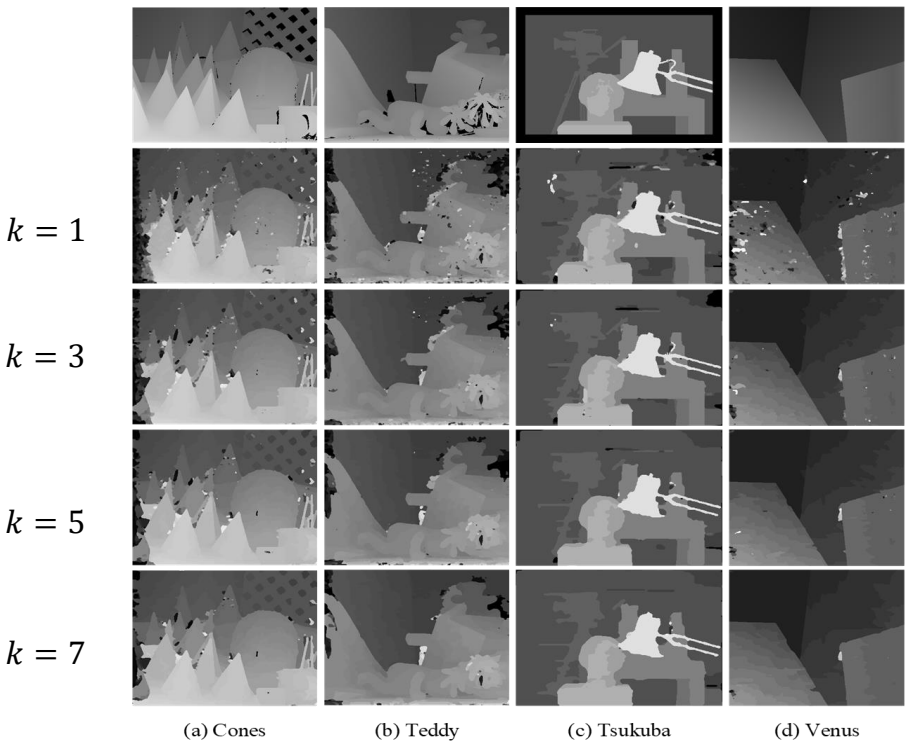

Figure 6. Image median filtering test.

Table 4. Image median filtering test results.

Test set	$k=1$	$k=3$	$k=5$	$k=7$
Cones	25.053%	16.704%	19.771%	21.316%
Teddy	26.453%	15.477%	21.427%	23.015%
Tsukuba	10.220%	3.220%	6.446%	7.951%
Venus	15.769%	6.112%	7.951%	12.356%

(a) Cones (b) Teddy (c) Tsukuba (d) Venus

Figure 7. Edge weight test.

Table 5. Test results for calculating edge weight.

Test set	Cones	Teddy	Tsukuba	Venus
$W_1(p_1, p_2)$	16.704%	23.015%	7.951%	12.356%
$W_2(p_1, p_2)$	26.219%	18.129%	11.523%	18.713%

Image median filtering is a nonlinear filtering method. The calculation formula for each pixel of image median filtering is shown in equation (5).

$$I(p(i_0, j_0)) = \frac{1}{k^2}\sum_{i\in(-(k-1)/2,(k-1)/2)}\sum_{j\in(-(k-1)/2,(k-1)/2)} I(p(i_0+i, j_0+j)) \quad (5)$$

where, k is the size of filtering window, the value of k is odd. When the window is too large, the image will become blurred. The parameter selection in this paper starts from the minimum window (k = 1) and gradually grows.

The test results using image median filter are shown in Figure 6 and table 4. In the Figure 6, the images in the first row are real disparity images, and the images in the rows 2,3,4 and 5 are the result of setting $k = 1$, $k = 3$, $k = 5$, and $k = 7$, respectively. It can be seen from Figure 6 and table 4 that

the image median filtering has a great influence on the results. When $k = 3$, the effect of image median filtering is better. Equation (6) and equation (7) are selected to test the edge weight. The test results are shown in Figure 7 and table 5. In the Figure 7, the images in the rows 1,2 and 3 are the real disparity images, the result images of a function $W_1(p_1, p_2)$ is selected and the result images of a function $W_2(p_1, p_2)$ is selected, respectively. It can be seen from Figure 6 and Figure 7 that the effect is the better when the function $W_2(p_1, p_2)$ is selected.

$$W_1(p_1, p_2) = \max\{\Delta_R, \Delta_G, \Delta_B\} \tag{6}$$

$$W_2(p_1, p_2) = |Gray(p_1) - Gray(p_2)| \tag{7}$$

When calculating MST by using Kruskal algorithm, it is need to sort all the edges in descending order of weight. The time complexity of common sorting algorithms is $O(nlog(n))$. However, the weight of the edge will not be greater than 255 in this paper, so histogram sorting can be used. The time complexity of this method is $O(1)$.

The time test results of the algorithm are shown in table 6. The numbers in brackets in the table 6 represent the running speed ratio. The running speed ratio of MST algorithm using GPU is compared with the corresponding algorithm using CPU. The running speed ratio of the algorithm proposed in this paper is compared with the corresponding MST algorithm using GPU. It can be seen from table 6 that the MST algorithm can be accelerated by about 17 times by using GPU, and the optimization algorithm proposed in this paper can increase the speed by about 20% on this basis. Therefore, the optimization algorithm proposed in this paper can speed up about 20 times compared with the algorithm using CPU, and it can meet the requirements of real-time processing.

Table 6. Algorithm running time.

Test set	Cones	Teddy	Tsukuba	Venus
CPU	0.863s	0.797s	0.173s	0.455s
GPU (using MST)	0.0494s (17.5)	0.0324s (24.6)	0.0131s (13.2)	0.0257s (17.7)
GPU (Using the optimization algorithm in this paper)	0.0358s (1.380)	0.0271s (1.196)	0.0123s (1.065)	0.0220s (1.168)

5. Conclusion

An advanced method is developed in this paper to improve the computation efficiency for stereo matching. The proposed method is to reduce the height of the tree used for MST algorithm in parallel processing, hence dramatically speed up the computation. An aggregation of matching cost is also developed simultaneously in order to achieve the hierarchical synchronous parallel computing. The cost is based on Leaf-to-Root aggregation.

The simulation results show that proposed parallel processing stereo matching algorithm using reduced high of spanning tree and L2R aggregation matching cost enhance the computation speed. It is a matter of 20 times speed-up as proved. This method therefore provides a promising approach to real-time stereo processing, which can of great importance of Binocular stereo matching.

Acknowledgments

This work was supported by the Natural Science Foundation of Jiangsu Province (BK20191012); Scientific Research Foundation of Nanjing Institute of Technology (JCYJ201822, CKJB201803 and CXY201932).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. G. Yang, X. Song, C. Huang, Driving stereo: A large-scale dataset for stereo matching in autonomous driving scenarios, *IEEE CVF Conference on Computer Vision and Pattern Recognition*, 2020, 899–908.
2. A. Z. Joseph, C. L. Priyankac, P. Sankaran, Stereo vision-based speed estimation for autonomous driving, *2019 International Conference on Information Technology, Bhubaneswar, India*, **6** (2019), 201–205.
3. M. Cheng, Y. Zhang, Y. Su, J. M. Alvarez, H. Kong, Curb detection for road and sidewalk detection, *IEEE Trans. Veh. Technol.*, **67** (2018), 10330–10342.
4. M. Faria, A. Ferreira, H. Pérez-Leon, I. Maza, A. Viguria, Autonomous 3D exploration of large structures using an UAV equipped with a 2D LIDAR, *Sensors*, **22** (2019), 4849–4852.
5. R. Wang, M. Z. Luo, N. K. Wang, L. J. Lu, Accuracy study of a binocular-stereo-vision-based navigation robot for minimally invasive interventional procedures, *World J. Clin. Cases*, **8** (2020), 69–78.
6. C. Yan, H. He, Y. Qiao, Measuring the wave height based on binocular cameras, *Sensors*, **19** (2019), 1338–1342.
7. M. G. Mozerov, V. Joost, One-view occlusion detection for stereo matching with a fully connected CRF model, *IEEE T. Image Process.*, **6** (2019), 1–2.
8. C. Zhang, C. He, Z. Chen, W. Liu, M. Li, J. Wu, Edge-preserving stereo matching using minimum spanning tree, *IEEE Access*, **7** (2019), 177909–177921.
9. S. Dutta, D. Patra, H. Shankar, P. A. Verma, Development of GIS tool for the solution of minimum spanning tree problem using prim's algorithm, *ISPRS Technical Commission 8th Mid-Term Symposium*, **8** (2014), 1105–1114.
10. H. Guo, L. Huang, Y. Lu, J. Ma, C. Qian, Z. Wang, Accelerating BFS via data structure-aware prefetching on GPU, *IEEE Access*, **6** (2018), 60234–60248.

©2021 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)