Further Optimized Parallel Algorithm of Watershed Segmentation Based on Boundary Components Graph[1]

Haifang Zhou[1], Xuejun Yang[1], Yu Tang[2], Nong Xiao[1]

[1] Institute of Computer, National University of Defense Technology, Changsha, China
[2] Institute of Electronic Technology, National University of Defense Technology
{haifang_zhou, yutang18}@sina.com

Abstract. *Watershed segmentation/transform* is a classical method for image segmentation in gray scale mathematical morphology. Nevertheless watershed algorithm has strong recursive nature, so straightforward parallel one has a very low efficiency. Firstly, the advantages and disadvantages of some existing parallel algorithms are analyzed. Then, a *Further Optimized Parallel Watershed Algorithm* (FOPWA) is presented based on boundary components graph. As the experiments show, FOPWA optimizes both running time and relative speedup, and has more flexibility.

1 Introduction

Watershed segmentation/transform is a classical and effective method for image segmentation in gray scale mathematical morphology. This method, with a wide perspective, has been applied successfully into some fields like remote sensing images processing of satellite and radar, biomedical applications and computer vision. However, watershed transform is a relatively time consuming task for its low efficiency, and in above fields, such as in remote sensing applications large size images, e.g. 1024×1024, 3000×3000 or larger, are not uncommon and must be processed in real time usually. Therefore, to study watershed algorithm easy to be paralleled is meaningful in real applications.

2 Related Work

Meijster and Roerdink had proposed a three-stage parallel watershed algorithm (M-R algorithm for short) in [1] based on components graph, which was designed for a ring-architecture with shared memory. But there are some potential logic errors/limits in M-R algorithm, as shown in [2]. Therefore, [2] pointed out an improved parallel

[1] This work is partially supported by the National 863 High Technology Plan of China under the grant No. 2002AA1Z201, 2002AA104510 and 2002AA714021, and the Grid Project sponsored by China ministry of education under the grant No. CG2003-GA00103.

watershed algorithm (IPWA for short) for distributed memory system, which got better performance. With the further study, we find that the adaptability of these two algorithms is limited: 1) The parallel efficiency of two algorithms is very low when they meet images with content of large size objects. 2) They are only designed for the segmentation of the images containing many plateaus with large area. 3) Because of simplified computation of plateaus, algorithms probably end up with images that contain thick watersheds, which need post-processing.

Moga and Cramariuc etc. had given some parallel methods of watershed transform based on definition by topographical distance [3]. We have learned from [4] that the proposed method based on Ordered Queue (OQ for short) is derived from optimal sequential watershed algorithm, but its scalability is quite limited. While an alternative solution, namely image integration by sequential scanning, introduced by [5], provides an equitable work load on multiprocessors, and hence a better relative speedup, but the absolute running time of this algorithm is very long. And then literature [6] proposed a method named rain-falling or hill-climbing simulation, which reduced re-scanning overhead through computing lower-complete image, but introduced undesirable overhead caused by the lower distance computation and preserved data dependent character of the algorithm. In addition, these algorithms do not construct watershed lines, but only labeled regions [3], needing post-processing.

3 An Optimized Parallel Method: FOPWA

Considering positive and negative contributions of above algorithms we proposed a *Further Optimized Parallel Watershed Algorithm* (FOPWA) based on definition of watershed transform by topographical distance. Topographical distance based watershed transform is started by detecting minima of the input data (called *seed-pixels*); ordered region growing is then performed according to *lower distance*. Lower distance is formally defined as following:

Definition 1. Let $D \subseteq Z^2$, f be a digital gray value image in domain D, and G be underlying grid of f. $f(p)$ denotes the gray value of pixel p. *Lower distance* d is defined as: $d(p) = 0$ if p is a minimum; otherwise, $d(p)$ equals the length of the shortest path $\{p = p_0, p_1, ..., p_s = q\}$ from p to a pixel q such that $\forall i \in \{1,2,...,s\}$, $(p_{i-1}, p_i) \in G$, $f(q) < f(p)$, and, if $s > 1, \forall i \in \{1,2,...,s-1\}, f(p_i) = f(p_{i-1})$.

Each non-seed pixel is put into different catchments (regions) in an increasing order of gray levels. This recursive label propagation is called *flooding*. We define a 2D ordering relation to satisfy parallel requirement for flooding.

Definition 2. *2D ordering relation* can be formulated by two conditions: Condition 1. If $f(q) = \min_{r \in N_G(p)} \{f(r) \mid f(r) < f(p)\}$, then q is the preceding-pixel of p (also called p is flooded by q), and $L(p) = L(q)$. $L(.)$ denotes the output label image. Condition 2. If $f(p) = f(q)$ and $d(p) = d(q) + 1$, then $L(p) = L(q)$ .Where $d(p)$ stands for lower distance of p with initial value ∞ ($d(p) = \infty$), which denotes a maximum value, and $N_G(p)$ for the set of neighboring pixels of p with respect to surrounding pattern.

We also assigned a unique label to each of boundary pixels with preceding-pixels, which are looked as seed-pixel like minima. Consequently, based on this 2D ordering relation, each processor can correctly and exhaustively delimit the extent of regions crossing the local sub-domains, regardless of what is happening in other processors; hence parallel computing could be realized. We name these additional seed-pixels as *pseudo seed-pixels*. Then, we define a *Boundary Components Graph* (BCG) to record the ordering relation between pseudo seed-pixel and its preceding for flooding and merging in later steps. Note that BCG is only related to boundary pixels of each sub-domain, not all pixels in the input image, so the size of components graph is reduced.

Definition 3. Considering the input image f as a direct valued graph $G = (V, E, f)$, in which V is the set of pixels in the graph, and E is the set of edges of the graph defining the connectivity. BCG $G^* = (V^*, E^*, f^*)$ ($f^* = f$) can be defined as following: 1) If $v \in V \wedge L(v) = \mathrm{H}$ or $L(v) \neq \mathrm{H} \wedge (\exists p, p \in V \wedge p \in N_G(v) \wedge L(p) = \mathrm{H})$, then $v \in V^*$. 2) $\forall u, v \in V^*$, if $L(u) \neq \mathrm{H} \wedge L(v) = \mathrm{H}$, and u and v satisfies Condition 1 or Condition 2 described above, then $(u, v) \in E^*$. 3) $\forall u, v \in V^*$, if $L(u) \neq \mathrm{H} \wedge L(v) = \mathrm{H} \wedge f^*(u) = f^*(v) \wedge d(u) = d(v) = \infty$, then $(u, v) \in E^*$.

For the implementation, the global domain D of size $X \times Y$ is split among N processors in sub-domains D_i; FOPWA has four stages: 1) Detecting real and pseudo seed-pixels and building local BCGs. 2) Local flooding. We use Ordered Queue (OQ) to realize the local watershed transform, for OQ is derived from optimal serial watershed algorithm that can obtain relatively shorter running time of local tasks. 3) Global merging. We use similar process to merging components graphs as IPWA [2]. 4) Broadcasting the merging result to each processor to updating the local results.

4 Experiments and Conclusion

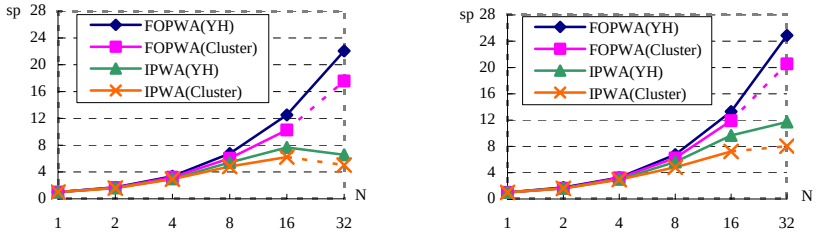

Fig. 1. The comparison of FOPWA and IPWA in speedups. The left is the speedup curves of two algorithms for the test image *Lena* with size of 512×512. The right is for the test image *Airport* with size of 1024×1024

Firstly, we realized FOPWA and IPWA [2] on two parallel platforms, and tested for various images with different size (256/512/1024/2048). Moreover, performance of

FOPWA is also compared with existing algorithms in running time and speedup. One of the two parallel platforms is a Cluster system with 16 nodes. Another parallel platform is YinHe supercomputer (YH for short), which includes 32 processors.

Fig.1 compares FOPWA with IPWA in speedup on two different platforms (dashed curves represent speedup trend for Cluster system when number of processors is over 16). From this figure, we can draw some conclusions: 1) FOPWA outperforms IPWA, and has better scalability. 2) FOPWA is less data dependent. 3) The result got from YH is better than that from Cluster for YH has better network.

FOPWA outperforms other existing parallel algorithms by combining the advantages of components-graph based method and distance based method but not introducing additional overhead. Table 1 compares performance of FOPWA with some existing parallel algorithms. As the experiments show, FOPWA optimizes both running time and relative speedup, and has more flexibility and adaptability, compared against old methods.

Table 1. The comparison of FOPWA and other parallel algorithms in serial time and speedups on YH. Test image is *Airport* with size of 1024×1024, *N* is the number of nodes in system.

Algorithms	Serial time (s)	Speedup(*N*=16)	Speedup(*N*=32)
Rigid OQ-based	11.402	5.548	7.358
Sequential scanning	52.882	14.964	26.296
Rain-falling	5.958	2.877	4.532
Connected component	11.883	11.777	21.527
IPWA	1.445	9.658	11.709
FOPWA	1.312	13.299	24.876

References

1. Meijster, A., Roerdink, J. B. T. M.: A proposal for the implementation of a parallel watershed algorithm. In Hlavac, V., Sara, R. (eds.): Computer Analysis of Images and Patterns. New York. (1995) 790-795.
2. Zhou, H. F., Jiang, Y. H., X.J. Yang, X. J.: An Improved Parallel Watershed Algorithm for Distributed Memory System. In Zhou, W. L. (ed.): Proceedings of the 5th International Conference on Algorithms and Architecture for Parallel Processing. IEEE Computer Society. Los Alamitos (2002) 310-314.
3. Zhou, H. F., Jiang Y. H., Yang, X. J.: Researches on serial and parallel strategies of watershed transform. Journal of national university of defense technology. Vol. 24(6). (2002)71-76. (in Chinese, cited by EI)
4. Moga, A. N., Viero, T., Dobrin, B. P.: Implementation of a distributed watershed algorithm. In Serra, J., Soille, P. (eds.): Computational Imaging and Vision Mathematical Morphology and Its Applications to Image Processing. Dordrecht, the Netherlands. (1994) 281-288
5. Moga, A. N., Viero, T., Gabbouj, M., et al.: Parallel watershed algorithm based on sequential scannings. In Pitas, I. (ed.): Proceedings 1995 IEEE Workshop on Nonlinear Signal and Image Processing, Vol. II. Neos Marmaras, Greece. (1995) 991-994
6. Cramariuc, B., Gabbouj, M.: A parallel watershed algorithm based on rain falling simulation. In Proceedings 12th European Conference on Circuit Theory and Design, Vol. 1. Istanbul, Turkey. (1995) 339-342.