International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:4, 2012

Parallelization and Optimization of SIFT Feature Extraction on Cluster System

Mingling Zheng, Zhenlong Song, Ke Xu, and Hengzhu Liu

***Abstract*—**Scale Invariant Feature Transform (SIFT) has been widely applied, but extracting SIFT feature is complicated and time-consuming. In this paper, to meet the demand of the real-time applications, SIFT is parallelized and optimized on cluster system, which is named pSIFT. Redundancy storage and communication are used for boundary data to improve the performance, and before representation of feature descriptor, data reallocation is adopted to keep load balance in pSIFT. Experimental results show that pSIFT achieves good speedup and scalability.

***Keywords*—**cluster, image matching, parallelization and optimization, SIFT.

I. Introduction

IMAGE matching is a fundamental aspect of many problems in computer vision, including object or scene recognition, solving for 3D structure from multiple images, stereo correspondence, and motion tracking. Image matching is divided into scale- information-based matching and feature-based matching. Scale-information-based matching is simple and easy to realize, but it is compute-intensive and very sensitive to changes in image scale, rotation, deformation etc. Feature-based matching can largely overcome these shortcomings, so it is well applied in computer vision. D. G. Lowe has proposed a method for extracting distinctive invariant features from images， named SIFT [1]. The SIFT features are invariant to image scale and rotation, and also provide robust matching across a substantial range of affine distortion, change in viewpoint, addition of noise, and change in illumination. So SIFT method becomes a hot topic and has been studied in serial optimization [5], parallel optimization [6], [7], and there are many method extensions, such as PCA-SIFT [2], GLOH [3] and CSIFT [4]. It is also applied in detection [8], [9], tracking [10], [11] and registration [12], [13].

A 128-dimensional vector is used to describe the SIFT feature, which is compute-intensive. The method of SIFT can't meet the real-time demand of many applications, such as online object recognition and real-time video processing.

Therefore, parallelization of SIFT has come to the researchers' attention. Parallel SIFT has been implemented on GPU, and extracted about 800 points from a 640x800 video at 10 frames per second (FPS)for the limitation of hardware and OpenGL [14].

Mingling Zheng is with the School of Computer, National University of Defense Technology, Changsha, Hunan, 410073 China (e-mail: minglingzh@ yahoo.com.cn).
Zhenlong Song is with the School of Computer, National University of Defense Technology, Changsha, Hunan 410073 China (e-mail: songzhl@ sina.com).
Ke Xu is with the Hunan Police Academy, Changsha, Hunan,China.
Hengzhu Liu is with the School of Computer, National University of Defense Technology, Changsha, Hunan, 410073, China.

Then the SIFT was parallelized on GPU with the speed of 20 FPS [15]. Qi Zhang et al. implemented parallel SIFT on the dual 4-core server and gain the speed of 45 FPS, that met the demand of real-time video stream with the speed 30 FPS [16]. For the High Definition Television (HDTV) image (size of 1920x1080), the speed of 10 FPS on the 16-core machine can't meet the requirement of HDTV. So Reference [17] adopted a 64-core Chip MultiProcessor (CMP) simulator and achieved 33 FPS, which met the real-time requirement of HDTV. With GPU、SMP and CMP platform, these studies basically meet the requirements of real-time applications with small image. But for the application of aerophotogrammetry and remote sensing, the image is massive and the speed of parallel SIFT on these platform can't meet the real-time demand. Seth Warn et al. carried on the parallel experiment of large image and achieved the speedup about 2x on dual 4-core SMP machine [7]. Feng et al. implemented the parallel SIFT on 32-processor cluster which has been named DDP-SIFT, but the speedup is about 10x. The two parallel methods have low speedup for large images.

This paper focus on the parallelization and optimization of SIFT feature extraction on cluster system for large images. The remainder of this paper is organized as follows. Section II describes the extraction of SIFT feature. In section III, we propose a parallelized and optimized algorithm of SIFT feature extraction. Section IV gives the experimental results and Section V concludes our work.

II. Sift Feature Extraction

SIFT method was proposed by D. G. Lowe in 2004, which remained invariant to rotation and change in image scale, and also showed to provide robust matching across a substantial range of affine distortion, change in viewpoint, addition of noise, and change in illumination [1]. Three major stages are used to extract SIFT feature in this paper. They are generation of Gaussian pyramid, scale space extrema detection and descriptor representation.

A. Generation of Gaussian Pyramid

Two-dimensional image I(x, y) at different scale space, can be expressed as follows:

$$L(x,y,\delta)=G(x,y,\delta)*I(x,y) \qquad (1)$$

Where * is the convolution operation of the x and y, two-dimensional Gaussian function G (x, y, δ) is defined as follows:

$$G(x,y,\delta)=\frac{1}{2\pi\delta^2}e^{-(x^2+y^2)/2\delta^2} \qquad (2)$$

The Gaussian pyramid (GSS pyramid) is computed by the convolution of a variable-scale Gaussian function with the input image. There are o octaves and each octave has s images in the GSS pyramid. For the first octave of scale space, the image (the input image is up-sampled by a factor of 2) is repeatedly convolved with Gaussian to produce the set of scale space images. After that, the Gaussian image is down-sampled by a factor of 2, and the process is repeated for other octaves.

B. Scale Space Extrema Dectection

In 1999, Lowe proposed to detect extrema at Difference-of-Gaussian (DOG) scale space [19]. Adjacent images in GSS pyramid are subtracted to produce the DOG images:

$$\begin{aligned} D(x,y,\delta) &= (G(x,y,k\delta) - G(x,y,\delta)) \\ &= L(x,y,k\delta) - L(x,y,\delta) \end{aligned} \quad (3)$$

In order to detect the local maxima and minima of D(x,y,δ), comparison between each sample point with its eight neighbors in the current image and nine neighbors in the scale above and below is performed. Twenty six comparison operations are needed for a sample point during this step. The next step is to perform a detailed fit to the nearby data for location, scale, and ratio of principal curvatures. This information is used to reject points that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge.

C. Descriptor Representation

Firstly, it is needed to determine the keypoint's location, scale and orientation. In the section B, the location and scale is ready. For each image sample, L(x,y), the gradient magnitude, m(x,y), and orientation, θ(x,y), are computed using pixel difference:

$$m(x,y)=\sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2} \quad (4)$$

$$\theta(x,y)=\tan^{-1}((L(x,y+1)-L(x,y-1))/(L(x+1,y)-L(x-1,y))) \quad (5)$$

In actual process, the gradient magnitude and orientation are computed in a region around the keypoint location, and these samples are accumulated into orientation histogram. The peak of the histogram is the main orientation and others above 80% of the peak are the auxiliary orientation. Then the coordinates of the descriptor and the gradient orientations are rotated to the main orientation to keep the invariability to rotation. A Gaussian window with the center at the keypoint is selected and these samples in the window are accumulated into orientation histograms over 4x4 subregions with 8 orientation bins. The descriptor is formed a vector containing the value of all orientation histogram entries, corresponding to the gradient magnitudes. So the descriptor vector contains 4x4x8=128 elements.

III. Parallelization And Optimization Of SIFT

The serial processing time of three steps is shown in table 1 and the platform is a dual 6-core server. The serial code is provided by Andrea Vedaldi [20]. The time of extrema detection is little and it occupies less the 5% of the total execution time. The time spent on the generation of GSS pyramid increases rapidly with the increasing size of image, and when the image size is 2048x2048, time spent on GSS pyramid is much more than that on the descriptor representation. In the generation of GSS pyramid, the base image (the bottom image in each octave) of current octave is from the last octave, so it is difficult to parallelize totally among octaves. How to parallelize and optimize the generation of GSS pyramid is one of the difficulties in parallelism and optimization of SIFT. Time of descriptor computation is affected by the number of keypoints and the computation of descriptors can be parallelized totally.

TABLE I
Serial Execution Time of SIFT Feature Extraction (Unit: Second)

Image Size	number of keypoints	GSS pyramid	Keypoint detection	Descriptor presentation
800x640	4513	0.8949s	0.076s	1.066s
1024x1024	14346	1.9899s	0.115s	3.122s
2048x1024	33409	5.1603s	0.278s	7.051s
2048x2048	43341	13.229s	0.452s	9.817s

Reference [18] proposed a parallel SIFT feature extraction algorithm, named DDP-SIFT. In the DDP-SIFT, the base images of each octave are produced in serial, referenced Pyramid Octave Base (POB) bellow. Then POB is divided and allocated to all task nodes to do the following steps. The top layer of POB is divided into one block. Next layer is divided into four blocks and the rest may be deduced by analogy. For example, when there are 4 octaves, the ratio of blocks number from top to bottom of POB is 1:4:16:64. The generation of POB can't be parallelized, so the time spent on this procedure is invariable when the system scale enlarged. This procedure results in bad speedup, and according to Amdahl's law, the speedup will get worse with the increasing number of computing nodes.

We propose a parallel algorithm with input image, named pSIFT. The flow of pSIFT is shown in Fig. 1. After the input image is partitioned and assigned to all nodes, the GSS pyramid is generated and local extrema is detected using local data on each node. All nodes send the number of keypoints to the master node(master node is the node whose identifier is 0 in this paper). The master node calculates the mean value according to the number of keypoints and nodes and broadcasts to all nodes. All nodes send the keypoints with a small region around that exceeds the mean value to the master node and the master node reallocates these data to the nodes whose number of keypoints is less than the mean value. All nodes compute the descriptor vector using local data in parallel.

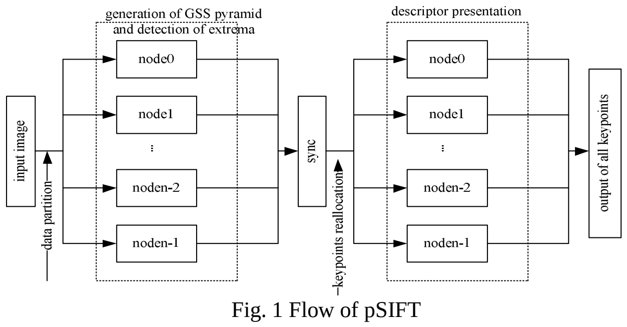

Fig. 1 Flow of pSIFT

A. Data Partition

During the generation of GSS pyramid, the base image of current octave is from last octave, so the procedure of octave generation can't be completely parallelized. According to the number of nodes, input image is allocated to all nodes evenly and all nodes compute GSS pyramid on the local image data. The partition of input image is diverse, but only two are in common use, as shown in Fig. 2.

If the boundary data is redundant, two partition methods shown in Fig.2 have no effect on the parallel algorithm. But for data communication, two methods will have different impact on performance. For the partition (a) in Fig. 2, one node will communicate with two adjacent nodes at most. For the partition (b) in Fig. 2, one node will communicate with up to four nodes, which brings additional communication overhead and adds the complexity of program. So the partition (a) is adopted in this paper.

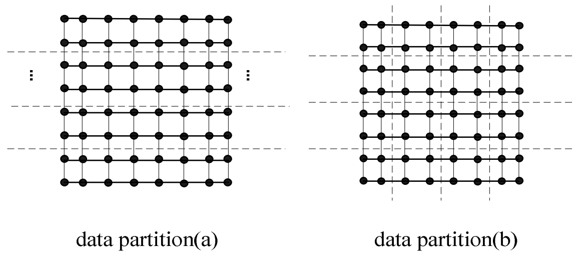

Fig. 2 Diagram of data partition

B. Processing of Boundary Data

The main problem brought by data partition is the processing of boundary data. After the partition of input image, the middle part of the input image may be the boundary data of a node and keypoints may be extracted from this area, just like data roped by the coil shown in Fig. 3(a). Different procedure needs different boundary data and the following discusses boundary data needed in SIFT feature extraction.

1. For the procedure of GSS pyramid generation, a boundary pixel needs w rows of data on adjacent node(2w+1 is the template size of Gaussian filter).
2. For the procedure of extrema detection, a pixel will be compared with 8 neighbors on the same scale, so one row of pixel on adjacent node is needed.
3. For the representation of descriptor, a keypoint needs a 16x16 block (not including the row and column of keypoint), so a boundary keypoint needs 8 rows of pixels on adjacent node.

There are two methods for the boundary data. One is redundant storage and another is communication. In the generation of GSS pyramid, the base image of current octave is from last octave, so if fully redundant storage is used, there will be large volume of data to be redundant and additional computation will also be brought. The following gives an example with o octaves and s images for each octave.

To get the base image of current octave, the Gaussian image of last octave that has twice the initial value of δ (δ is the deviation of Gaussian function) is resampled by taking every second pixel in each row and column [1]. If w rows of boundary pixels need to be redundant for the oth octave, 2w rows are needed for o-1th octave. The rest may be deduced by analogy, and 2^{o-1}w rows are needed to be redundant for the first octave. So there will be much additional computing for this method. And the volume of redundant data is invariant when the system scale enlarged. For 1024x1024 image on 32 CPU cores, when the template size of filter, w, is 8 and number of octave, o, is 4, the local data for each node are 1024x32 and the redundant data is 1024x2^{4-1}x8=1024x64. That is to say, the redundant data are twice as the local data, which has affected the performance seriously. Above figures only take into account the unilateral data redundancy, bilateral data needs to be redundant in fact.

Redundancy and communication are adopted for boundary data in this paper. For the first octave, 2(w+8) rows are redundant for each node, shown in Fig. 3(b), so the first octave is computed by local data and the redundant data does not bring additional computation. In the following octaves, each node obtains the boundary data by communication as shown in Fig. 3(c). Data movement consists of two directions: up and down data transfer. In the communication, end nodes (node 0 and node n-1 in Fig. 3(c)) send and receive once, and other nodes send and receive twice.

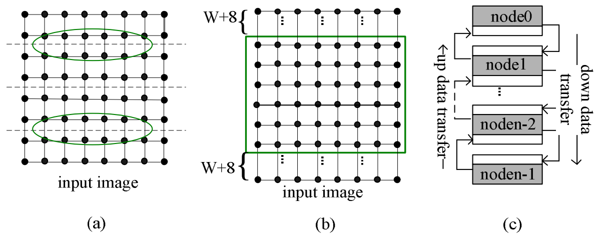

Fig. 3 Processing of boundary data

C. Data Synchronization

After the detection of extrema, all nodes compute the descriptors using local data directly, which can reduce the overhead of communication. However, the number of keypoints on each node is not equal and the load imbalance is variable with difference input image. So there is data synchronization before descriptor computation. All nodes send the number of keypoints to the master node, and the master node calculates and broadcasts the mean value based on the total number of nodes and keypoints. All nodes compare the number of keypoints with the mean value. The keypionts that exceeds the mean value are sent to the master node with position, scale, gradient orientation and a small region around. The master node distributes these keypoints to the nodes whose number of keypoints is less than the mean. Each node computes the descriptor in parallel.

IV. Experimental Results and Analysis

We evaluate the performance of pSIFT on the cluster developed by National University of Defense Technology (NUDT). The configuration of cluster system is as follows:

1. Computing node is configured with 48GB memory and dual 6-core Intel Xeon CPU X5670 which has 2.93GHz clock and 12MB L3 cache.
2. Luster file system is used.
3. Bandwidth of network is 160Gbps in dual.

Fig. 4 shows the speedup of pSIFT and DDP-SIFT, and we can see that the performance of pSIFT is better than DDP-SIFT, for pSIFT avoids the serial process. With the increasing number of CPU cores, the performance of DDP-SIFT becomes worse and the speedup almost reaches the peak with 24 cores. For the image with size of 1024x1024, the speedup of pSIFT is about 20x and it is only about 10x for DDP-SIFT with 32 cores.

Fig. 4 also gives the experimental result of image with size of 2048x2048 and the super-linear speedup is achieved. For large images, the GSS pyramid is too large to cache and cache miss leads to the overhead of memory access. For the first octave with 5 images, there are 80MB data and it’s hardly to put into cache totally. So when the scale of system enlarges, the probability of cache miss will be reduced and the performance will be improved which has nothing to do with the computation or communication.

In Fig. 4, pSIFT:1024 and pSIFT:2048 are the pSIFT experimental results with image size of 1024x1024 and 2048x2048. DDP-SIFT:1024 is the DDP-SIFT experimental result with image size of 1024x1024. The same is true of Fig.5.

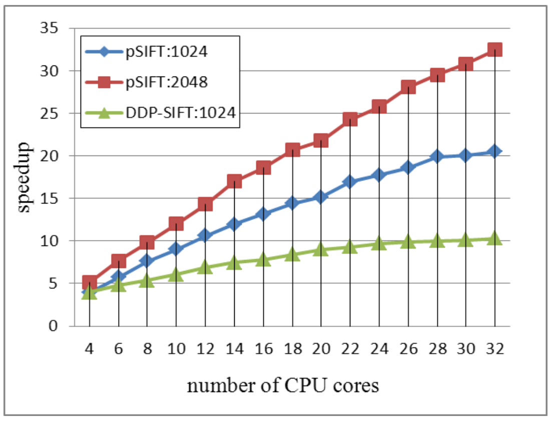

Fig. 4 Speedup of pSIFT and DDP-SIFT

Synchronization is executed before keypoints descriptor computation to keep load balance. Synchronization keeps the load balance, and at the same time it brings communication overhead. With the enlarging of system, the ratio of computation with communication will reduce and the overhead of communication will affect the performance more apparently. We carry on the experiment for the synchronization. The synchronization is canceled before computation of descriptor, so the descriptor is directly computed after the keypoint detection and this process is named nsSIFT (non-synchronization SIFT). Fig. 5 shows the experimental result of pSIFT and nsSIFT on 32-core cluster, and pSIFT is superior in performance to nsSIFT.

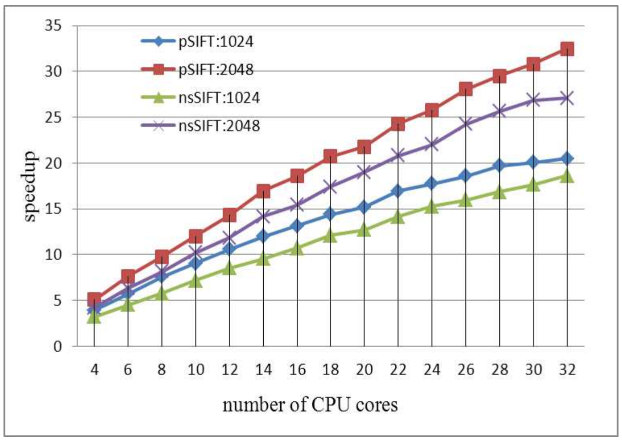

Fig. 5 Performance comparison between pSIFT and nsSIFT

V. Conclusion And Future Work

We proposed pSIFT which implemented the parallel SIFT on the cluster. The pSIFT method gains good scalability and speedup compared to DDP-SIFT. This is mainly for the data partition of pSIFT that avoids the serial execution time. In this paper, boundary data is dealt with carefully, which avoids the loss of feature and improves the performance.

Images with the size of 2048x208 are used for the experiment. The experimental results show that the volume of GSS pyramid data has great effect on performance for large images. So how to improve the performance of SIFT to large images will be a challenge and it is one of our future works.

Acknowledgment

The authors are grateful to the reviewers for their important contribution to improve the paper clarity. And the authors also want to give thanks to Andrea Vedaldi for his source code of serial SIFT.

References

[1] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision, vol. 60, pp. 91–110, 2004.
[2] A.Y.Ke and R.Sukthankar, “PCA-SIFT: A more distinctive representation for local image descriptors,” In Proc. 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’04), pp.506-513.
[3] Mikolajczyk, K., Schmid, C., “A performance evaluation of local descriptors,” IEEE Trans. Pattern Analysis and Machine Intelligence. Vol.27, pp.1615-1630, Augst 2005.
[4] Alaa E. Abdel-Hakim and Aly A. Farag, “CSIFT: A SIFT Descriptor with Color Invariant Characteristics,” in proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06).
[5] Geoffrey Treen, and Anthony Whitehead, “Efficient SIFT Matching from Keypoint Descriptor Properties,” 2009 Workshop on Applications of Computer Vision(WACV), pp1-7.
[6] Vanderlei Bonato, Eduardo Marques, and George A. Constantinides, “A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection,” IEEE Trans. Circuits and Systems for Video Technology, VOL.18, pp1703-1712, 2008.
[7] Seth Warn, Wesley Emeneker , Jackson Cothren,Amy Apon, “Accelerating SIFT on Parallel Architectures,” In Proc. 2009 IEEE Int. Conf. Cluster Computing and Workshops(CLUSTER’09), pp.1-4.
[8] Marc Lalonde, David Byrns, Langis Gagnon, Normand Teasdale, Denis Laurendeau, “Real-time eye blink detection with GPU-based SIFT tracking,” In Proc. 4th Canadian Conference on Computer and Robot Vision(CRV'07), pp.481-487,2007.

[9] Sirmacek, B., Unsalan, C., “Urban-Area and Building Detection Using SIFT Keypoints and Graph Theory,” IEEE Trans. Geoscience and Remote Sensing, Vol.47, pp.1156-1167, 2009.
[10] Gangqiang Zhao, Ling Chen, Jie Song, Gencai Chen, “Large head movement tracking using SIFT-based registration,” In Proc. 15th international conference on Multimedia, PP: 807-810, 2007.
[11] Jiang, R.M., Crookes, D., Luo, N., Davidson, M.W., “Live-Cell Tracking Using SIFT Features in DIC Microscopic Videos,” IEEE Trans. Biomedical Engineering, Vol.57, pp: 2219-2228, 2010.
[12] Goncalves, H., Corte-Real, L., Goncalves, J.A., “Automatic Image Registration through Image Segmentation and SIFT,” IEEE Trans. Geoscience and Remote Sensing, Vol.49 pp.2589-2600, 2011.
[13] Yi, Z., Zhiguo, C., Yang, X., “Multi-spectral remote image registration based on SIFT,” IEEE Electronics Letters, Vol.44 pp. 107-108,2008.
[14] Sudipta N. Sinha, Jan-Michael Frahm, Marc Pollefeys, and Yakup Genc, “Feature Tracking and Matching in Video Using Programmable Graphics Hardware,” Machine Vision and Applications, Vol.22, pp.207-217, March 2007.
[15] S. Heymann, K. Muller, A. Smolic, B. Froehlich, and T. Wiegand, “SIFT implementation and optimization for general-purpose GPU,” In Proc. WSCG’07, 2007.
[16] Q. Zhang, Y. Chen, Y. Zhang, and Y. Xu, “Sift implementation and optimization for multi-core systems,” IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008), pp. 1–8, 2008.
[17] H. Feng, E. Li, Y. Chen, and Y. Zhang, “Parallelization and characterization of sift on multi-core systems,” IEEE International Symposium on Workload Characterization (IISWC’08), pp. 14–23, 2008.
[18] Guiyuan Jiang, Guiling Zhang and Dakun Zhang, “A Distributed Dynamic Parallel Algorithm for SIFT Feature Extraction,” 3rd International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), pp.381-385, 2010.
[19] Lowe, D.G., “Object recognition from local scale-invariant features,” In Proc. IEEE Int Conf. Computer Vision, pp. 1150-1157, 1999.
[20] Andrea Vedaldi, SIFT source code, download from http://www.vlfeat.org/~vedaldi/assets/ siftpp/versions/.