Acceleration of the Speed of Tissue Characterization Algorithm for Coronary Plaque by Employing GPGPU Technique

TAKANORI KOGA
Tokuyama College of Technology
Dept. of Computer Science and Electrical Eng.
Gakuendai, 745-8585 Shunan
JAPAN
koga@tokuyama.ac.jp

SHOTA FURUKAWA
Yamaguchi University
Graduate School of Science and Eng.
1677-1 Yoshida, 753-8512 Yamaguchi
JAPAN
n016vc@yamaguchi-u.ac.jp

NORIAKI SUETAKE
Yamaguchi University
Graduate School of Science and Eng.
1677-1 Yoshida, 753-8512 Yamaguchi
JAPAN
suetake@sci.yamaguchi-u.ac.jp

EIJI UCHINO*
Yamaguchi University
Graduate School of Science and Eng.
1677-1 Yoshida, 753-8512 Yamaguchi
JAPAN
uchino@yamaguchi-u.ac.jp

Abstract: The general purpose computation technique on Graphics Processing Unit (GPGPU) has got into the limelight recently. The authors have proposed the multiple k-nearest neighbor (MkNN) classifier for the tissue characterization of coronary plaque. Its characterization performance is highly evaluated. The purpose of this paper is to accelerate the speed of MkNN classifier aiming for it to be actually used in the medical practice. It has been confirmed that its speed is drastically accelerated enough for the practical use.

Key–Words: Pattern Classification, Pixel Classification, Tissue Characterization, Multiple k-Nearest Neighbor, GPGPU Technique

1 Introduction

An intravascular ultrasound (IVUS) method [1] is a tomographic imaging technology, which is often used for the diagnosis of acute coronary syndromes (ACS) [2, 3] in the field of cardiology. The IVUS method provides thousands of two-dimensional cross-sectional images of plaque in coronary artery. A medical doctor diagnoses the coronary plaque by carefully observing the B-mode images [4], which are constructed by the intravascular ultrasound signals.

In order to realize a precise tissue characterization of coronary plaque to support the medical doctors, we have proposed a multiple k-nearest neighbor (MkNN) classifier [5, 6]. The MkNN classifier classifies coronary plaque pixel by pixel. The MkNN classifier considers the spatial continuity of distribution of the data, not only in the feature space but also in the observation space [5, 6], and thus performs a fine classification even if the distributions of data for each class overlap with each other in the feature space.

However, the MkNN classifier takes too much time for classification. Therefore, the speed-up of the MkNN classifier is a supreme order to make it used in the medical practice.

Recently, the general purpose computation technique on Graphics Processing Unit (GPGPU) has got into the limelight in various fields of science and technology [7]. The calculation performance of the latest graphics processing unit (GPU) is extremely higher than that of central processing unit (CPU). Moreover, by using the Compute Unified Device Architecture (CUDA) library supplied by NVIDIA Corporation [7], it is easy to develop a general purpose program processed on GPU with C language.

In this paper, in order to realize a speed up of the multiple k-nearest neighbor (MkNN) classifier [5, 6], it is implemented by employing GPGPU technique. First, we verify its performance using a benchmark problem. We then apply it to the real tissue characterization problem of coronary plaque in the IVUS image. This is our main concern.

*Eiji Uchino is a chairman of Fuzzy Logic Systems Institute (FLSI), 680-41 Kawazu, 820-0067 Iizuka, JAPAN.

ISBN: 978-960-474-283-7

2 MkNN Classifier

Fig.1 shows the pixel classification procedure of the multiple k-nearest neighbor (MkNN) classifier for an example of a two-class classification problem. In Fig.1, the observation space is an observed image, and the feature vector space consists of the feature vectors obtained at each pixel of the observed image. In general, the feature vectors are calculated usually, *e.g.*, by principal component analysis of an image, Fourier transformation, and so on.

In this paper, for the tissue characterization of coronary plaque, the feature vectors are calculated by the Fourier transformation of radio frequency (RF) signal, which constitutes the IVUS B-mode image [6]. The training feature vectors (prototypes) are fed to the MkNN classifier in the same manner as to the ordinary kNN classifier.

Now, suppose that a set of N pairs $\{(\boldsymbol{v}_n, i_n); n = 1, \cdots, N\}$ are given, where $\boldsymbol{v}_n$ is a training feature vector which belongs to class $i_n \in \{1, \cdots, C\}$. C is the number of classes.

The classification scheme and the procedure of the MkNN classifier are as follows. That is, in the MkNN classifier, the classification is carried out for the feature vector obtained at each pixel $\boldsymbol{p} = (x_1, x_2)$ of interest (POI) in the observation space as shown in Fig. 1. Note that k represents "the number of neighboring pixels $\boldsymbol{p}_m$ of $\boldsymbol{p}$ (POI) in the observation space," and k' represents "the number of training feature vectors $\boldsymbol{u}_m$ in the feature space corresponding to pixels $\boldsymbol{p}_m$ in the observation space."

The substantial difference between the ordinary kNN classifier and the proposed MkNN classifier is as follows. In the MkNN classifier, not only the training feature vectors corresponding to the pixel of interest $\boldsymbol{p}$ (the pixel to be classified) but also those of neighboring pixels $\boldsymbol{p}_m$ of $\boldsymbol{p}$ in the observation space (*i.e.*, on the image) are used for the classification.

In the MkNN classifier, $(k \times k')$-nearest training feature vectors are selected for the training of $\boldsymbol{p}$ (POI). Finally, the majority decision considering the class labels of all the selected training feature vectors is made in order to classify $\boldsymbol{p}$ (POI). On the other hand, the ordinary kNN classifier only uses the class labels of k'-nearest prototypes corresponding to $\boldsymbol{p}$ (POI) in the majority decision making.

The following is a brief description of the procedure of the MkNN classifier.

Step1: Select the neighboring pixels $\{\boldsymbol{p}_m; m = 1, \cdots, k\}$ of $\boldsymbol{p}$ (POI) in the observation space. Calculate $\boldsymbol{u}_m$ for $\boldsymbol{p}_m$.

Step2: Calculate the Euclidean distances $\|\boldsymbol{v}_n - \boldsymbol{u}_m\|$ between each $\boldsymbol{u}_m$ and all the training feature vectors $\{\boldsymbol{v}_n; n = 1, \cdots, N\}$.

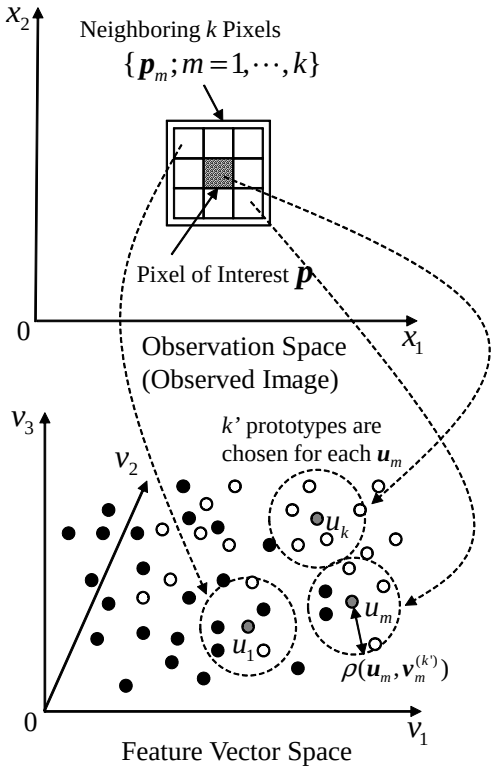

Figure 1: Overview of the MkNN classifier.

Step3: The training feature vectors $\boldsymbol{v}_n$ which satisfy the following condition:

$$\|\boldsymbol{v}_n - \boldsymbol{u}_m\| \leq \rho\left(\boldsymbol{u}_m, \boldsymbol{v}_m^{(k')}\right), \qquad (1)$$

are selected for each $\boldsymbol{u}_m$, where $\boldsymbol{v}_m^{(k')}$ is the k'-th nearest training feature vector around $\boldsymbol{u}_m$. $\rho(\boldsymbol{u}_m, \boldsymbol{v}_m^{(k')})$ is an Euclidean distance between $\boldsymbol{u}_m$ and $\boldsymbol{v}_m^{(k')}$ in the feature space.

Step4: The feature vector for pixel $\boldsymbol{p}$ in the observation space is classified into class c as follows:

$$c = \arg\max_j \sum_{m=1}^{k} \sum_{n=1}^{N} U(\boldsymbol{u}_m, \boldsymbol{v}_n, i_n, j), \qquad (2)$$

$$U = \begin{cases} 1, & \|\boldsymbol{v}_n - \boldsymbol{u}_m\| \leq \rho\left(\boldsymbol{u}_m, \boldsymbol{v}_m^{(k')}\right) \text{ and } i_n = j \\ 0, & otherwise. \end{cases} \qquad (3)$$

Eq. (3) is a function for a majority decision making of a class label.

ISBN: 978-960-474-283-7

In the MkNN classifier, the spatial continuities both in the observation and feature spaces are utilized in the procedure of **Step3**. That is, in the MkNN classifier, $(k \times k')$-nearest training feature vectors are selected for each feature vector corresponding to $\boldsymbol{p}$ (POI) based on the spatial relationship between $\boldsymbol{p}$ and $\boldsymbol{p}_m$ in the observation space. This means that the characteristic of MkNN is to utilize the information on the spatial continuities in both the observation and feature spaces. The MkNN classifier thus realizes a fine pixel classification even for the B-mode image with heavy noises and/or measuring errors [5, 6].

3 Experiments

3.1 Experimental Conditions

The program codes for GPGPU using CUDA library [7] are constituted by a host program and a GPU kernel function. The host program consists of the program routines for data transfer to GPU and its control routines. The GPU kernel function consists of the program routines to be processed on the GPU.

In the experiments, the MkNN classifier is programmed in the GPU kernel function. In programming of MkNN classifier, paper [8] is referred, which describes an implementation of the ordinary kNN classifier by using GPGPU.

In order to compare the computational costs for various implementations, the following seven types of implementations are achieved. Below is a brief explanation of each implementation with an abbreviation label of each implementation.

- MATLAB: Program described by MATLAB script
- MATLAB+PCT: Program described by MATLAB script using Parallel Computing Toolbox (execution in parallel on CPU)
- MATLAB+PCT+BLAS: Program described by MATLAB script using Parallel Computing Toolbox, in which Basic Linear Algebra Subprograms (BLAS) library like distance calculation is performed
- C: Program described by C language
- CBLAS: Program described by C language and BLAS library
- CUDA: Program described by C language and CUDA
- CUBLAS: Program described by C language, CUDA, and CUBLAS（BLAS library optimized for CUDA)

Table 1: Specification of Tesla C1060 GPU computing processor board used in the experiments.

Peak Processing Performance	933 [GFLOPs]
Number of Processor Cores	240
Clock Speed	1,296 [MHz]
Memory Size	4 [GB]
Memory I/O	GDDR3 512 [bit]
Memory Clock	800 [MHz]

MATLAB is a numerical computation software produced by Mathworks. MATLAB's Parallel Computing Toolbox (PCT) is a library to produce functions for parallel computing on multi-core CPU. BLAS is a library for linear algebraic processing with respect to vectors and matrices. CUBLAS is an optimized BLAS library for GPGPU [7].

Note that Basic Linear Algebra Subprograms (BLAS) library like distance calculation in the MATLAB+PCT+BLAS makes the following reduction in terms of the distance calculation of Eq.(1).

The Euclidian distance between a feature vector $\boldsymbol{u}_m$ and a training vector $\boldsymbol{v}_m$ is calculated by $\|\boldsymbol{v}_n - \boldsymbol{u}_m\|^2 = \boldsymbol{v}_n^T\boldsymbol{v}_n - 2\boldsymbol{u}_m^T\boldsymbol{v}_n + \boldsymbol{u}_m^T\boldsymbol{u}_m$. Here, $\boldsymbol{u}_m^T\boldsymbol{u}_m$ can be neglected as an unnecessary term in the evaluation of distance. Thus, the Euclidian distance between the feature vector and the training vector can be obtained only by calculating $\boldsymbol{v}_n^T\boldsymbol{v}_n - 2\boldsymbol{u}_m^T\boldsymbol{v}_n$. In the program code of the MATLAB+PCT+BLAS, this leads to a big reduction of calculation.

The computer used in the experiments has Intel Core i7 920 2.67GHz (with 4 cores) CPU and NVIDIA Tesla C1060 GPU computing processor board. The specification of Tesla C1060 computing processor board is described in Table 3.1. Operating system (OS) is Microsoft Windows XP 32-bit. The GPU kernel function is described by CUDA. MATLAB's version is 2009b. Compiler of C language is Microsoft Visual Studio 2008.

3.2 Performance Validation by Using Randomly-Generated Data Set

For performance validation of GPGPU, an artificially-generated random data set is used. The feature vector is 16-dimensional real-valued vector. The number of classes is 11. The number of the training feature vectors is 3,000. The training feature vectors are sampled from 11 classes of data set with equal probability.

Under the above conditions, the computational costs are evaluated for the following 3 cases.

ISBN: 978-960-474-283-7

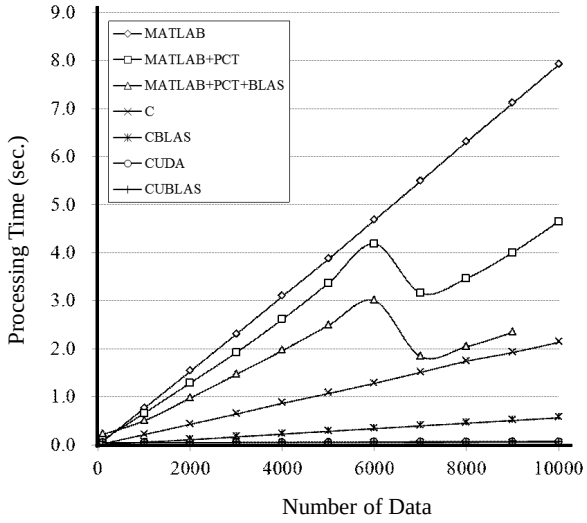

Figure 2: Computational costs versus the number of data (pixels) to be classified. Experimental conditions: $k=9$ (3×3), and $k'=9$.

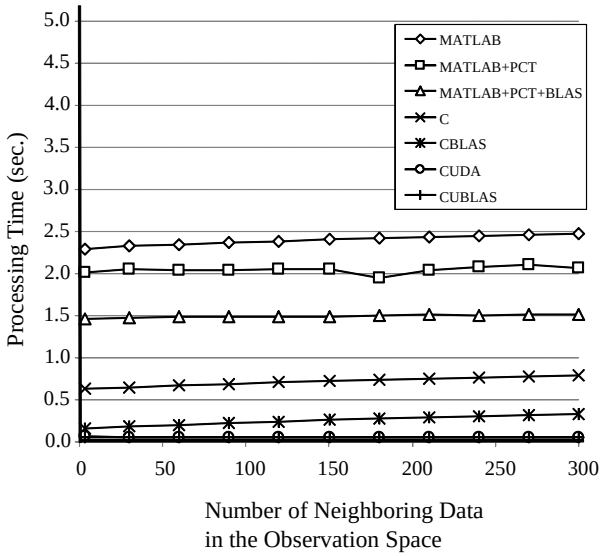

Figure 3: Computational costs versus the number of neighboring data (pixels) in the data observation space to be classified. Experimental conditions: the number of data to be classified is 3,000, and $k'=9$.

- Case 1: the number of data (pixels) to be classified is changed
- Case 2: the number of neighboring data (pixels) in the data observation space is changed
- Case 3: the number of neighboring vectors in the feature vector space is changed

The computational cost is evaluated by an average time of 100 iterations for each case. The parameters of the MkNN classifier are shown in the captions of Figs.2, 3 and 4, respectively.

Fig.2 shows the experimental results for case 1 versus the number of data (pixels) to be classified. In general, the computational cost monotonically increases with the number of data. On the contrary, in cases with CUDA and CUBLAS, the computational cost is very small independent of the number of data. The program using CUBLAS is a little bit faster than that using CUDA.

In the cases of MATLAB implementations using PCT, there can be seen sudden drops of computational cost around 7,000. It is thought that these are caused by the internal processes of allocation for parallel processing. Furthermore, in case of MATLAB+PCT+BLAS, there is a missing of result around 10,000. This is because a memory allocation was impossible due to a hardware restriction. That is, this comes from too huge memory area was tried to be allocated automatically for the parallel processing by PCT.

Fig.3 shows the experimental results for case 2 versus the number of neighboring data (pixels) in the data observation space. It can be seen that the computational cost is independent of the neighboring data. Those results imply that each implementation method can process the MkNN classifier in approximately the same calculation time with the ordinary kNN classifier. It means that the MkNN classifier is an effective classifier because it can use the information in the data observation space in approximately the similar calculation time of the ordinary kNN classifier.

Fig.4 shows the experimental results for case 3 versus the number of neighboring vectors in the feature vector space. The computational cost is in general independent of the neighboring data. The programs using CUDA and CUBLAS are a little bit influenced by the number of neighboring data. This is because the sorting algorithm is involved in the MkNN classifier to obtain the neighboring order in the feature vector space, which is not suitable for parallel processing. The similar phenomena have been reported in [8].

In all cases, it was confirmed that the classification results were not affected in all the experiments with Tesla C1060, although Tesla C1060 computing processor board only supports single precision.

3.3 Real Application to Tissue Characterization of Coronary Plaque

Here we confirm the performance of the GPGPU technique applied to the tissue characterization of coronary plaque in the real IVUS image of a patient.

The region of interest (ROI) surrounded by two white lines in Fig.5(a) is classified into lipid tissue, fi-

ISBN: 978-960-474-283-7

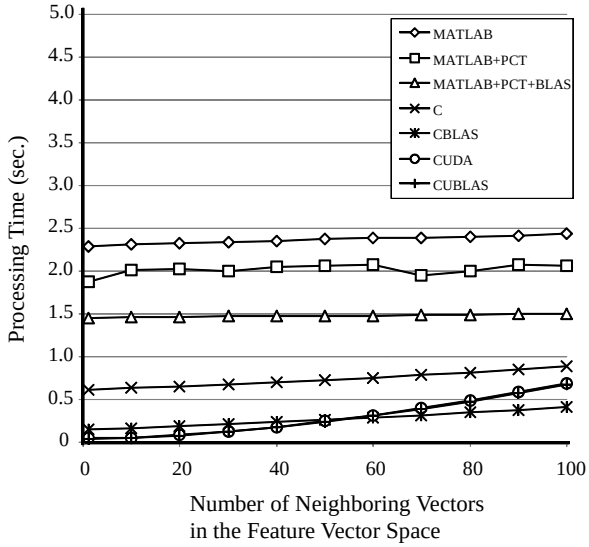

Figure 4: Computational costs versus the number of neighboring vectors in the feature vector space. Experimental conditions: the number of data to be classified is 3,000, and $k = 9(3 \times 3)$.

brofatty tissue, and fibrous tissue. In the experiments, the feature vector obtained at each pixel of a B-mode image is a power spectrum calculated by the short-time discrete FFT of a radio frequency (RF) signal in radial direction [5, 6].

The short-time discrete FFT was carried out by shifting the window of a size of 32 pixels in depth direction of a radial line. The feature vector is 17-dimensional real-valued vector concerning spectrum. The feature vectors are normalized in the range of $[0, 1]$. The number of class is 3. The number of the training feature vectors is 195. The training feature vectors are sampled from 3 classes of data sets with equal probability.

In the experiments, approximately 70,000 samples (pixels) per IVUS B-mode image are classified. In case of the MkNN classifier, the number of neighboring data in the feature vector space k' is set to be 9, and the number of neighborhood data in the observation space k is set to be 9 (3×3).

Table 2 shows the comparison of processing time by each implementation. In this regard, experimental result by CUBLAS could not be obtained partly because memory allocation was impossible due to a hardware restriction. The calculation time employing GPGPU (CUDA) is drastically reduced, which is good enough for a medical practice.

It was also confirmed as in the simulation experiments using artificial data set in section 3.2 that the classification results were not affected in all the experiments with Tesla C1060, although Tesla C1060 computing processor board only supports single precision.

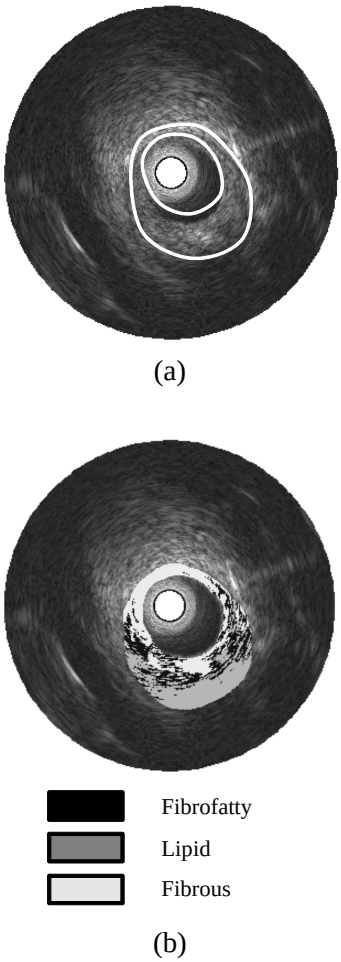

Figure 5: Tissue characterization of coronary plaque in IVUS B-mode image. (a) The area surrounded by two white lines is a region of interest (ROI) to be classified. (b)Tissue characterization results by the MkNN classifier.

With those experiments, tissue characterization of coronary plaque by the MkNN classifier has proposed in [5, 6] reached almost near to the practical use.

4 Conclusion

In this paper, we have implemented the MkNN classifier, a tissue characterization algorithm for coronary plaque, by using GPGPU technique. The speed of the MkNN classifier has been drastically accelerated. The possibility of the practical use of the tissue characterization by MkNN classifier has been greatly increased. Future work is to further reduce the computing time by using GPU clusters aiming at the real practical use soon.

ISBN: 978-960-474-283-7

Table 2: Computational cost for each implementation in classification of IVUS data.

Implementation method	Calculation time [sec.]
MATLAB	13.12
MATLAB+PCT	6.00
MATLAB+PCT+BLAS	4.56
C	1.24
CBLAS	0.53
CUDA	0.07

Acknowledgement

This work was supported by the strategic program for promoting research of Yamaguchi University.

References:

[1] J. B. Hodgson, S. P. Graham, A. D. Savakus, S. G. Dame, D. N. Stephens, P. S. Dhillon, D. Brands, H. Sheehan and M. J. Eberle, "Clinical percutaneous imaging of coronary anatomy using an over-the-wire ultrasound catheter system," *Int. J. Cardiac Imaging*, Vol.4, pp.187–193, 1989.

[2] E. Falk, "Why do plaques rupture?," *Circulation*, Vol.86 (Suppl III), pp.30–42, 1992.

[3] E. Falk, P. K. Shah and V. Fuster, "Coronary plaque disruption," *Circulation*, Vol.92, pp.657–671, 1995.

[4] J. D. Klingensmith, D. G. Vince, B. D. Kuban, R. Shekhar, E. M. Tuzcu, S. E. Nissen and J. F. Cornhill, "Assessment of coronary compensatory enlargement by three-dimensional intravascular ultrasound," *Int. J. Cardiac Imaging*, Vol.16, pp.87–98, 2000.

[5] R. Kubota, E. Uchino and N. Suetake, "Hierarchical k-nearest neighbor classification using feature and observation space information," *IEICE Electronics Express*, Vol.5, No.3, pp.114–119, 2008.

[6] E. Uchino, N. Suetake, R. Kubota, T. Koga, G. Hashimoto, T. Hiro and M. Matsuzaki, "An ROC performance validation of hierarchical k-nearest neighbor classifier applied to tissue characterization using IVUS-RF signal," *Proc. of 2009 Int. Workshop on Nonlinear Circuits and Signal Processing*, pp.333–336, 2009.

[7] `http://www.nvidia.com/object/cuda_home_new.html.`

[8] V. Garcia and E. Debreuve, "Fast k nearest neighbor search using GPU," *Proc. of the CVPR Workshop on Computer Vision on GPU*, 6 pages, 2008.