2009 International Conference on Computer Engineering and Applications
IPCSIT vol.2 (2011) © (2011) IACSIT Press, Singapore

Cache-conscious Feature Construction for Breast Abnormality Detection

Visanu Changniam[1,2], Juggapong Natwichai[2]
[1]Information System, Faculty of Business Administration and Liberal Art
Ragamangala University of Technology Lanna, Chiang Mai, Thailand
[2]Computer Engineering Department Faculty of Engineering
Chiang Mai University, Chiang Mai, Thailand

Abstract. Breast cancer has been a cause of women-mortality for many decades. Mammogram is one of the best means to help radiologists to detect the calcification, an indicator for the abnormality, because a very small calcification (1-3 millimeters) can even be detected. There are many classification methods proposed for such the detection task, however, its efficiency in data preparation and training phases needs to be addressed. In this paper, we propose to investigate the possibilities to solve the efficiency problem using cache-conscious approach, without any compromise on the effectiveness. We apply two cache-conscious techniques, i.e. data re-blocking, and loop fusion to improve the efficiency on a sub-process of the classifier building. The main aim of such the techniques is to fetch the cpu-needed data to suit the cache size as much as possible. Achieving the aim will result in a high cache-hit-rate, and thus improve the efficiency. A preliminary experiment is conducted on the Sobel edge detection sub-process to validate the efficiency our approach. From the results, we can reduce the time used in the process by 68%, with the L2 cache-miss-rate decreased by 10%.

Keywords: Feature Extraction, Cache Conscious, Breast Abnormality Detection, Sobel Edge Detection

1. Introduction

Breast cancer has been a cause of women-mortality for many decades. Since, the cause and the treatment which can guarantee 100% of recovery have yet to be discovered. Therefore, one of the best possible way to deal with such the problem is to detect the cancer as soon as possible. The sooner the cancer can be discovered, the more opportunity the cancer can be cured. Though, breast abnormality can be detected by selfexamination. However, such the method can only detect 2-5 centimeter calcification, which is an important indication of breast abnormalities [1]. An effective method to detect smaller calcification is mammograms, as it can facilitate radiologists to detect 1-3 millimeter-calcification, so called microcalcification. Typically, a radiologist takes approximately 15 minutes to diagnose one mammogram. As a number of mammograms to be diagnosed could be high, the ability in detecting microcalcification will decrease over time. Therefore, there are many attempts have been proposed to alleviate this situation by computer-based abnormality detection [2], [3], [4], [5]. From such the attempts, one of the effective approaches is using fuzzy-based classification. In [5], authors reported that the fuzzy rule-based classification can achieve up to 90.36% accuracy rate with only 4.73 false-positives. Typically, the classifiers to detect abnormality in mammograms are built in two steps. Firstly, the features needs to be extracted, then, the classifiers are trained from the set of extracted data. Here, we focus on the feature extraction process. In [2], where the authors presented the fuzzy rule-based classification for such the task, four features are extracted as follows. Firstly, the edge of each given mammogram is detected to separate objects of interest from the background. Because the previous output from the first step may have some noise, therefore, a few noise filters such as median filter needs to be applied. Subsequently, boundary following algorithm is applied to locate the position of the objects. Then, the characteristics of the objects are then transformed into features, such as B-descriptor and D-descriptor using Fourier transformation, or average intensity difference between inside and outside boundaries. Although, such the feature set can be used well for the microcalcification detection task, its efficiency issue need to be addressed. There are a few work attempted to solve the issue [6], [7]. In this paper, we propose to address the problem of efficiency using cache-conscious approach. To the best of our knowledge, this is the first attempt to use such the approach for the task. Cache-conscious algorithm/techniques, in fact, have been so far used to improve the efficiency of computational problem in the area of databases/datamining. For example, in [8], a cache characteristics, i.e. its capacity, block (cache line) size, and associativity, have been parameterized, and considered to improve the B+ tree operations. Also, the frequent pattern mining problem has been addressed cache-consciously in [9]. In order to illustrate the possibility to apply such the approach for the problem, we select the Sobel edge detection [4], which is one of the most time-consuming processes in calcification detection, as an example sub-process to be improved using two cache-conscious techniques, i.e. data re-blocking, and loop fusion. We have also conducted the preliminary experiments to demonstrate and validate our work. The organization of the paper is as follows. The details of the feature extraction process are presented in Section 2. In Section 3, we review some key concepts of cache-conscious optimization, and propose techniques to be applied to our work. Subsequently, we validate our proposed work by experiments in Section 4. Finally, the conclusion and future work are presented in Section 5.

2. Feature Extraction for Calcification Detection

In this section, we present the details of feature extraction process presented in [2]. From such the work, the classification takes a set of mammograms as the training data. For each image, it is scanned with a sliding window with 35x35 pixels window from the top to the bottom, and from the left to the right of the image in order to obtain four features, the window is slid by 10 pixels each time. Such the features obtained are B-descriptor, D-descriptor, average intensity inside boundary, and intensity difference between inside and outside boundary of the center of the processed window. Subsequently, the set of rules is derived from such the features.

2.1. Edge Detection

Firstly, each scanned window from a mammograms is detected to find its edge. Here, the Sobel edge detection [4] is applied. The concept is to find the area which the gray level changes dramatically. The Sobel edge detector uses two 3x3 convolution masks, i.e. horizontal mask, and vertical mask as shown in Figure 1b), and 1c) respectively. Then, Equation 1 and 2 can be used to compute the gradient of the input window as positioned in Figure 1a). Finally, the approximate gradient is computed from Equation 3

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figure 1. Image Region, Horizontal Sobel mask G_x, and Vertical Sobel mask G_y

$$G_x = (x_7 + 2x_8 + x_9) - (x_1 + 2x_2 + x_3) \quad (1)$$
$$G_y = (x_3 + 2x_6 + x_9) - (x_1+2x_4+x_7) \quad (2)$$
$$|G| = |G_x| + |G_y| \quad (3)$$

2.2. Noise Suppression

After the edge of each window has been detected, noise suppression is then proceeded. Typically, the median filter replaces the center pixels inside the window with the median gray level. From the running example, the result of this step is shown in Figure 2c). Subsequently, the threshold gray level of the window is used to separate object and background image. Figure 2d) presents the output of the running example after this step has been proceeded.

2.3. Image Boundary Following

As Fourier descriptors are to be used as features, the image boundary needs to be computed. It is performed by following its outside boundary in the clockwise direction and recording the coordinate sequence [10]. The sequence of points in this system is downsampled to produce m points, which usually m is set at 50. The boundary result of the running example is shown in Figure 2e).

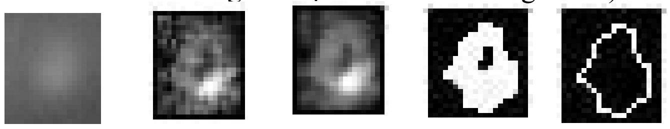

a) b) c) d) e)

Figure 2. Original Image, Sobel Edge Detection, Median Filter, Threshold Image, and Contour Image

2.4. Fourier Descriptor

After the boundary of each window is obtained, we extract two out of four features using Fourier transformation, i.e. B-descriptor and D-descriptor [11] as follows. Firstly, the coordinate x_i, and y_i of the ith boundary point is represented as a complex value u_i, where $u_i = x_i + jy_i$. Then, the u_i which is in the time domain can be changed to the coefficient in the frequency domain using Equation 4:

$$a_k = \frac{1}{m}\sum_{i=0}^{m-1} u(n)e^{-j2\pi k\frac{i}{m}} \qquad (4)$$

where m is the number of points of the boundary, and k = $-\frac{m}{2}$,…, $\frac{m}{2}-1$. Finally, the B-descriptor of the window can be computed using Equation 5.

$$b(k) = \frac{(a_{1+k} * a_{1-k})}{a_1^2} \quad \textit{for each } \text{n} = 0, 1, \ldots, \text{m-1}. \quad (5)$$

Meanwhile, the D-descriptor of the given window can be computed using Equation 6.

$$d(k) = \frac{(a_{1+k} * |a_1|^k)}{a_1^{k+1}} \quad \textit{for each } \text{n} = 0, 1, \ldots, \text{m-1}. \quad (6)$$

Note that B-descriptors (bk) are invariant in size, angular orientation, position, and the start position of the contour sequence. While, D-descriptors (dk) are independent of translation and dilation.

2.5. Average Intensity Inside Boundary

One of the other two features used in [2] is the average intensity inside the boundary. Generally, after we get the boundary of the window, this feature can be computed using Equation 7.

$$g_{ave\_in} = \frac{1}{npx}\sum_{i=1}^{npx} g_i \qquad (7)$$

where g_i is the gray level of the ith pixel which is inside the boundary, and npx is number of pixels inside the boundary.

2.6. Different Intensity Between Inside and Outside Boundary

Finally, the last feature is the different intensity between inside and outside boundary of the given window. We compute this feature using Equation 8.

$$Diff\_Int = g_{ave\_in} - g_{ave\_out} \qquad (8)$$

where gave in and gave out is the average intensity inside,and outside the boundary respectively.

3. Efficiency Improvement using Cache-consciousness

In this section, we will present a short overview of cache memory, subsequently, cache-conscious techniques which can be used to improve the efficiency. Finally, we briefly present the possibilities to apply such the techniques for our task.

3.1. Cache Memory Overview

Cache memories are small, fast static RAM designed to improve program-running-efficiency by prioritizing recently referenced data. Modern architectures typically employ two levels of cache (L1 and L2) between the CPU and main memory. While the L1 cache can perform at CPU speed, the L2 cache and main memory accesses normally introduce latencies in the order of 10 and 100 cycles respectively. Cache memories can reduce the memory latency only when the requested data is found in the cache. This mainly depends on the memory access pattern of the application. Memory references satisfied by the cache are called hits, these references will be processed at processor speed. However, those unsatisfied references, called misses, will incur a cache miss penalty, And they have to be detached from the corresponding cache block, then they have to be fetched from the main memory instead. Thus, unless special care is taken, memory latency could become a performance bottleneck significantly.

3.2. Cache-conscious Improvement Techniques

Typically, the main aim of the improvement techniques is to ensure that there are as few cache misses occur as possible, also this should be done without significantly increasing the number of instructions to be executed. There are a few techniques commonly applied [12], we list the two main techniques used in this paper as follows.

- **Data Re-blocking**
  Intuitively, the algorithm efficiency can be improved if the data to be processed is stored in the cache rather than the hard disk, or main memory. This technique divides the data to be processed into chunks which can fit in the cache. Let us take an example of a naive nested loops computing. From the non-reblocking version in Figure 3, we can see that the a[i], and b[j] needs to be read from the main memory separately. However, if we process the array b in the block-manner as shown in the re-blocking version in the same figure, the efficiency can be improved. The technique starts with determining the block size of the cache (BKSZ in Figure 3). Then, the loop is divided in the manner that the data in each block of the b array is processed for the whole block at once. Which will ensure that the elements of arrays b in a block will almost always be in the cache provided BKSZ is less than the cache size, thus significantly improving the cache performance.

Non-reblocking on arrays:	**Re-blocking on array b:**
for (i = 0; i < M; i++)	**for** (bkNo = 0; bkNo < N/BKSZ; bkNo++)
for (j = 0; j < N; j++)	**for** (i = 0; i < M; i++)
process(a[i],b[j]);	**for** (j = bkNo*BKSZ; j < (bkNo+1)*BKSZ; j++)
	process(a[i],b[j]);

Figure 3. Data Re-blocking and Non-reblocking

- **Loop Fusion**
  Usually, separated loops operating on the same data structure can be merged. This can improve the locality of references for such the structure. For example, in Figure 4, combining the creation of a key pointer in the array with the insertion into a hash table will improve the likelihood that a[i] could be in the cache when it is inserted in the hash table.

Non-loop fusion:	**Loop fusion:**
for (i = 0; i < N; i++)	**for** (i = 0; i < N; i++)
a[i].key = relation[i].key;	a[i].key = relation[i].key;
a[i].ptr = relation[i].ptr;	a[i].ptr = relation[i].ptr;
for (i = 0; i < N; i++)	insert in hashtable(a[i]);
insert in hashtable(a[i]);	

Figure 4. Loop and Non-loop Fusion.

3.3. Cache-conscious Calcification Detection

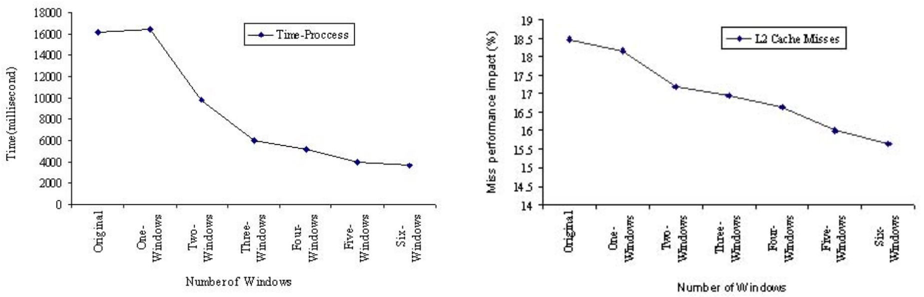
In general, almost all the sub-processes needed to detect the calcification can benefit from the cache-conscious approach. Since the data to be processed on all the steps are already the fixed size block with 35x35 pixels. Therefore, the most intuitive technique to be applied is the re-blocking the data to fit the cache size. Instead of time spending for reading data from the main memory or hard disk every time, we can tailor the size of the reading block to suit the cache size. Though the other data outside each 35x35 pixels windows might not be processed in a current loop, it will stay in the cache for the next coming loop. The loop fusion technique can also be applied by considering the sub-processes or tasks in a sub-process which its input is single-independent windows. As the feature extraction process presented in Section2 for a single window are all independent, we can combine all the loops of each process into one obviously.

4. Experiments

In this section, we present a preliminary experiment results of the proposed approach. As mentioned above, the Sobel edge detection processing is selected. Each mammogram image size 1024x1536 pixels. The experiments are conducted on an 1.73 GHz Intel Pentium M with 512 megabytes main memory running Microsoft Window XP. The processor has an 8-way set associative 32KB L1 data cache, an 8-way set associative 2048KB unified L2 cache on the chip. The cache-line sizes are 64 bytes for the L1 and L2 cache. From the cache setting, the size of block that can fit into our cache at once is 180x180 pixels. The system bus runs at 133MHz. Both the cache-conscious and non-cache-conscious version of the Sobel edge detection are implemented using JDK 5.0. We use the *Inte® VTune^{TM} Performance Analyzer* to collect the performance results. This tool profiles programs execution at the level of source code, and provides performance characteristics for each function in the implementation. The following analysis presents average processing time and the L2 miss rate measured over five runs. There are two parameters to be controlled, i.e. the size of the block which results from the re-blocking technique, and the number of window to be processed at the same time which results from the loop fusion technique.

4.1. Loop Fusion

We vary the size of the number of window to be processed in the same loop using the loop fusion technique in this experiment. For validation, the processing time, and the L2 cache-miss-rate are presented. The size of the block is set at 180x180 pixels. The cache-conscious version algorithm will read the required window(s) to the block. While, the original version algorithm will process each window directly. From Figure 5a), we can see that the processing time of the original algorithm is less that the time of the onewindow version slightly. This is because the one-window version needs more time to perform fetch the window into the block. Afterward, the processing time decreases when more loops are fused together. Also, the increasing of the windows decreases the miss rate as shown in Figure 5b).

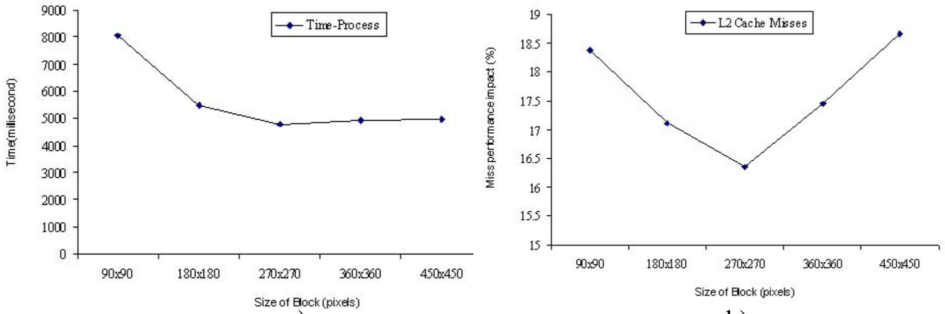


a) b)

Figure 5. Time Process and L2 Cache Misses per number windows

4.2. Data Re-blocking

In this experiment, we vary the size of the block to be read into main memory, such that block data may remain in the cache. The size of the blocks to be experimented are 90x90, 180x180, 270x270, 350x350, and 450x450 pixels. Here, to present the effect of the data re-blocking technique, we fix the number of window at 4. The result of applying the technique is presented in Figure 6a). We can see that when the size of the block is increased, the running is decreased. Noticeably, the running is dropped sharply almost half when we increase the block size from 90x90 pixels to 180x180 pixels which is fit to the cache size. From that setting, the running time is almost constant, but still much faster than the processing time of the 90x90 pixels-block. Also, we can see that the lowest running time occurred when we set the block size at 270x270 pixels. This can be described using Figure 6b) which it can be seen that the miss rate is the lowest.



a) b)

Figure 6. Time Process and L2 Cache Misses per size of block

5. Conclusion Remarks

The idea of cache-consciousness can improve processing time, and can increase the efficiency of the cache usage. If the data can be partitioned, and the loops are fused together properly, reading data form slow hard disks or main memory to CPU can be lessen. In this paper, we investigate the possibilities to apply the approach to address the efficiency problem of feature extraction for the classification task on the breast abnormality detection. Two cache-conscious techniques, i.e. data re-blocking, and loop fusion, are applied to improve the efficiency on a sub-process of the building of the classifier. From the preliminary experiment, the fourwindows unison can decrease the processing time by 68% approximately. Also, the setting can decrease the L2 cachemiss-rate by 10% comparing with the non-cache-conscious version. In our future work, we will further improve the whole processes of fuzzy-based classification by the proposed approach. Loop fusion between processes will also be investigated.

6. References

[1] Paul C. Stomper, Joseph Geradts, S.B.E., Levine, E.G.: Mammographic predictors of the presence and size of invasive carcinomas associated with malignant microcalcification lesions without a mass. *American Journal of Roentgenology* 6 (2003) 1679–1684.

[2] Sansanee Auephanwiriyakul, Siripen Attrapadung, S.T., Theera-Umpon, N.: Breast abnormality detection in mammograms using fuzzy inference system. In: *Proceedings of The 2005 IEEE International Conference on Fuzzy Systems*., IEEE (2005) 150–160.

[3] Netsch, T., Peitgen, H.: Scale space signatures for the detection of clustered microcalcifications in digital mammograms. *The 1999 IEEE Transactions on Medical Imaging* 18 (1999) 744–748.

[4] Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (2002).

[5] Sutasinee Thovutikul, S.A., Theera-Umpon, N.: Microcalcifi-cation detection in mammograms using interval type-2 fuzzy logic system. In: *The 2007 IEEE International Conference on Fuzzy Systems*., IEEE (2007) 1–5.

[6] Setnes, M., Roubos, H.: Ga-fuzzy modeling and classifi-cation: complexity and performance. *IEEE Transactions on Fuzzy Systems*. 8 (2000) 509–522.

[7] Hahn-Ming Lee, Chih-Ming Chen, J.M.C., Jou, Y.L.: An efficient fuzzy classifier with feature selection based on fuzzyentropy. *IEEE Transactions on Systems, Man, and Cybernetics, Part B:Cybernetics*. 31 (2002) 426–432.

[8] Rao, J., Ross, K.A.:Making b+- trees cache conscious in main memory. SIGMOD Rec.29 (2000) 475–486.

[9] Ghoting, A., Buehrer, G., Parthasarathy, S., Kim, D., Nguyen,A., Chen, Y.K., Dubey, P.: Cache-conscious frequent pattern mining on a modern processor. In*: The 31st International Conference on Very Large Data Bases*, VLDB Endowment (2005) 577–588.

[10] Janakiraman, P.A.: Robotics and Image Processing. New Delhi, Tata McGraw-Hill (1995).

[11] James M.Keller, Zhanqi Cheng, P.D., Hocaoglu, A.: Fourier descriptors features for acoustic landmine detection. 4742 (2002).

[12] Shatdal, A., Kant, C., Naughton, J.F.: Cache conscious algorithms for relational query processing. In: Proceedings *of the 20th International Conference on Very Large Data Bases*, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1994) 510–521.