在具有超廣播匯流排網狀連結計算機上設計最佳時間之平行影像模板匹配演算法

Time-Optimal Parallel Image Template Matching Algorithms on the Mesh-Connected Computers with Hyperbus Broadcasting *

蔡 鴻 仁† Horng-Ren Tsai 洪 西 進‡ Shi-Jinn Horng 蔡 樹 山‡ Shun-Shan Tsai
李 信 興§ Shung-Shing Lee 高 宗 萬¶ Tzong-Wann Kao

†私立僑光商專資訊管理科
Department of Information Management, The Overseas Chinese College of Commerce
E-mail: hrt@rs2.occc.edu.tw
‡國立台灣科技大學電機系
Department of Electrical Engineering, National Taiwan University of Science and Technology
E-mail: horng@mouse.ee.ntust.edu.tw
§私立復興工商專電子科
Department of Electronic Engineering, Fu Shin Institute of Technology and Commerce
¶私立光武工商專電子科
Department of Electronic Engineering, Kuang Wu Institute of Technology and Commerce

摘 要

本論文所發展的演算法完全植基在具有超廣播匯流排網狀連結計算機的架構上。給一$N \times N$數位影像和一$m \times m$模板(或視窗)，我們首先將分別設計兩個最佳時間的平行演算法來解影像模板匹配的問題：第一個是使用m^2N^2處理器且每個處理器包含$O(1)$記憶體時，可執行$O(\log m)$時間；另一個則是使用N^2處理器且每個處理器包含$O(m)$記憶體時，可執行$O(m^2)$時間。然後，基於以上這兩個演算法，我們將推導出使用p^2N^2處理器且每個處理器包含$O(\frac{m}{p})$記憶體時，執行$O(\frac{m^2}{p^2}+\frac{m}{p}\log p)$時間的一般化平行演算法來，其中$1 \leq p \leq m$。很清楚地，我們的演算法在時間上達到了最佳化和最佳速度比。
關鍵字-影像模板匹配、平行演算法、影像處理、具有超廣播匯流排之網狀連結計算機。

Abstract

The computation model on which the algorithms are developed is the mesh-connected computers with hyperbus broadcasting (abbreviated to MCCHB*). For an $N \times N$ digitized image and an $m \times m$ template (or window), we first design two optimal parallel algorithms for the image template matching problem; one runs in $O(\log m)$ time using $O(m^2N^2)$ processors with each processor containing $O(1)$ restricted memory, and the other runs in $O(m^2)$ time using N^2 processors with each processor containing $O(m)$ restricted memory, respectively. Then, based on these two proposed algorithms, an $O(\frac{m^2}{p^2}+\frac{m}{p}\log p)$ time generalized parallel algorithm is derived using p^2N^2 processors with each processor containing $O(\frac{m}{p})$ restricted memory, where $1 \leq p \leq m$. Clearly, our algorithm is both optimal and optimal speed-up in time complexities.*
Key Words- *image template matching, parallel algorithm, image processing, mesh-connected computers with hyperbus broadcasting (*MCCHB*).*

1 Introduction

With the great computation power of parallel processing systems, parallel computations for image processing and computer vision have received considerable attention during the last few years. Because of its regularity in architecture and simple interconnection network [4, 8], the mesh-connected computer (abbreviated to MCC) is one of famous parallel processing systems and has been widely applied in image processing and computer vision. Unfortunately, the large communication diameter is the inherent drawback of the MCC. In order to overcome such a drawback, the MCC has recently been enhanced with various faster communication mechanisms by many researchers [1, 2, 9, 10, 15]. These include the mesh-connected computers with multiple broadcasting (abbreviated to MCCMB) [15], the generalized mesh-connected computers with multiple buses (abbreviated to GMCCMB)[3] and the reconfigurable parallel processing systems [11, 16, 17, 18, 25]. For the sake of reducing the system diameter, the MCCMB was constructed by embedding the global

*This work was supported by National Science Council under the contract no. NSC-86-2213-E011-058.

buses to each row and column of the MCC [15]. For increasing the parallelism of the MCCMB, the GMCCMB was proposed by partitioning the MCCMB into several individual modules and all modules were connected by global buses only [3]. For improving the fixed architecture and local communication mechanism of the MCC, the reconfigurable parallel processing system was proposed by equipping it with a reconfigurable bus system whose configuration could be dynamically changed by properly setting local switches within each processor [16, 17, 18, 25]. Recently, based on the advanced VLSI technique, Horng [9, 10] extended the mesh-connected computers with hyperbus broadcasting (abbreviated to MCCHB). Such an extension led this machine to be more powerful than other enhanced MCCs. For example, compared to the MCCMB and GMCCMB, it not only efficiently promoted the communication ability but also kept the simplicity and regularity in its architecture. Besides, Li and Maresca [16, 17] had shown that the local switches in each processor of the reconfigurable parallel processing system would take about 20% silicon area of the system. On the contrary, the silicon area needed for the global buses was far less than that needed for the local switches. It would be quite efficient and save so much silicon area by embedding the global buses to the parallel processing system instead of embedding the local switches to it.

The image template matching problem is one of the fundamental problems and can be used to solve many practical image processing and computer vision problems [22]. It can be defined as follows. Given an $N \times N$ digitized image and an $m \times m$ template (or window), the image template matching problem is to compare the template with all possible windows of the image, and the result of each window operation is stored in the top left corner of the window. This problem has been studied extensively by several researchers using various parallel computation models in the literature [5, 6, 7, 11, 12, 13, 14, 19, 20, 21, 23].

In this paper we are interested in using the mesh-connected computers with hyperbus broadcasting to solve the image template matching problem. We first design two parallel algorithms: one runs in $O(\log m)$ time using m^2N^2 processors with each processor containing $O(1)$ restricted memory; and the other runs in $O(m^2)$ time using N^2 processors with each processor containing either $O(m)$ or $O(1)$ restricted memory, respectively. Then, based on these two algorithms, an $O(\frac{m^2}{p^2} + \frac{m}{p}\log p)$ time generalized algorithm is developed by using p^2N^2 processors with each processor containing $O(\frac{m}{p})$ restricted memory, where $1 \le p \le m$. Clearly, our algorithms are better than those results proposed in the literature [5, 6, 7, 14, 19, 20, 21, 23].

2 The Computation Model

A *k-dimensional* (k-D) MCCHB of size M contains M processors arranged in a k-D grid. That is, the processors can be thought of as logically arranged as in a k-D array $A(n_{k-1}, n_{k-2}, \ldots, n_0)$, where n_j, $0 \le j < k$, is the size of the j^{th} dimension and $M = n_{k-1} \times n_{k-2} \times \ldots \times n_0$. Usually, assume $n_{k-1} = n_{k-2} = \ldots = n_0 = N = 2^n$.

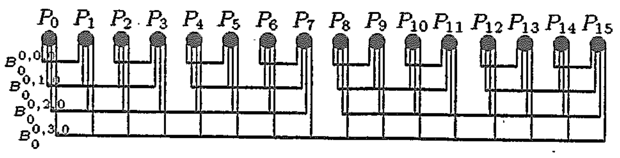
P_0 P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_8 P_9 P_{10} P_{11} P_{12} P_{13} P_{14} P_{15}
$B^{0,0}_0$
$B^{0,1}_0$
$B^{0,2}_0$
$B^{0,3}_0$

Figure 1: A 1-D MCCHB of size 16.

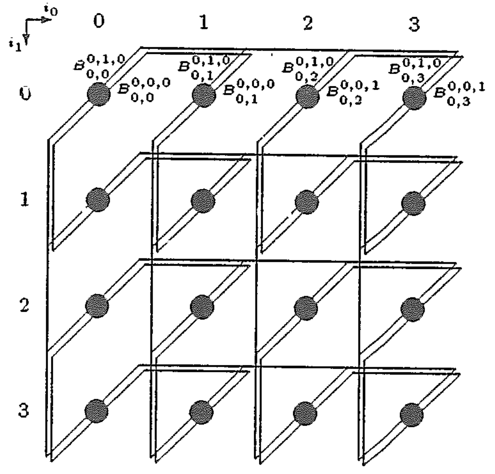

Figure 2: A 2-D MCCHB of size 4×4.

Then, $M = N^k = 2^{kn}$. Each processor is identified by a *k-tuple* unique index $(i_{k-1}, i_{k-2}, \ldots, i_0)$, $0 \le j < k$, $0 \le i_j < n_j$. The processor with index $(i_{k-1}, i_{k-2}, \ldots, i_0)$ is denoted by $P_{i_{k-1}, i_{k-2}, \ldots, i_0}$. Conventionally for the MCC, $P_{i_{k-1}, \ldots, i_j, \ldots, i_0}$ is connected to $P_{i_{k-1}, \ldots, i_j \pm 1, \ldots, i_0}$, $0 \le j < k$, provided $P_{i_{k-1}, \ldots, i_j \pm 1, \ldots, i_0}$ exists. That is, each processor (except the boundary processors) is connected to its $2k$ nearest neighbors by its local buses. The MCCMB is constructed by embedding the MCC with global buses [15]. In addition to the global bus for each row and each column as proposed in the MCCMB [15], we also extend the bus capacity for each processor to $k \times n$ global buses but without local buses. That is, instead of using $2k$ local ports for the local buses, each processor has $k \times n$ global ports for the global buses.

Conceptually, a group of processors is connected by a global bus if its size is a power of 2. For a given size 2^s (s is an integer) on a specific dimension of the k-D MCCHB, group $g(s)_0$ is beginning at processor 0, then group $g(s)_1$ is beginning at processor 2^s, then group $g(s)_2$ is beginning at processor 2 2^s, ... and so on along that dimension. Thus it follows that the number of local buses of the MCC is equal to that of global buses of the MCCHB. That is, the link (local bus or global bus) complexity of the MCCHB is the same as that of the MCC. Also the local buses of the MCC can be simulated by the global buses of the MCCHB. That is, we can use the global bus of group $g(s)_i$ to simulate the local bus to connecting two neighboring processors of group $g(s-1)_{2i}$ and group $g(s-1)_{2i+1}$ for any $s > 1$, $0 \le i < N/2^s$. Hence, all data routing operations of the MCC can be simulated by the MCCHB without any extra cost. Assume $N = 16$ and $k = 1$. We show an example for a 1-D MCCHB of size 16 in Figure 1 by establishing the global connection $B_{i_0}^{0,\,l,\,\omega}$ for each processor, where $0 \leq l < 4$ and $0 \leq \omega < \frac{16}{2^{l+1}}$. Assume $N = 16$ and $k = 2$. A 2-D MCCHB of size 4×4 is also shown in Figure 2 by establishing the global connection $B_{i_1,\,i_0}^{j,\,l,\,\omega}$ for each processor, where $0 \leq j < 2$, $0 \leq l < 2$, $0 \leq \omega < \frac{4}{2^{l+1}}$, and $0 \leq i_0,\ i_1 < 4$, respectively.

An MCCHB is operated in an SIMD (single instruction stream, multiple data streams) model. For a unit of time, assume each processor can either perform arithmetic and logic operations or communicate with others by broadcasting data on a bus. It allows multiple processors to broadcast data on the different buses simultaneously at a time unit, if there is no collision.

3 Basic Operations

Several data manipulation operations will be proposed in this section. These data manipulation operations will be used to derive the image template matching algorithm in the next section. Without loss of generality, assume $N = 2^n$, $m = 2^k$ and $p = 2^q$ also $N \geq m \geq p$ to be used in the following sections for some integers n, k and q.

3.1 The Multiple Block Summations

Let $A = (a_{im+\beta})$, $0 \leq \beta < m$, $0 \leq i < \frac{N}{m}$, be a data matrix of size N with $\frac{N}{m}$ segments, where m is the length of the segment. The multiple segment summations are to compute the summation of each segment. That is,

$$ss_{im} = \sum_{\alpha=0}^{m-1} a_{im+\alpha}, \tag{1}$$

where $0 \leq i < \frac{N}{m}$.
Based on the semigroup operation as proposed by Horng [10], Eq. (1) can be computed in $O(\log m)$ time using N processors by the binary tree technique. This leads to the following lemma.

Lemma 1 *[23] The multiple segment summations can be computed in $O(\log m)$ time on a 1-D N MCCHB.* □

Let $A = (a_{im+\alpha,\ jm+\beta})$, $0 \leq \alpha,\ \beta < m, 0 \leq i,\ j < \frac{N}{m}$, be an $N \times N$ data matrix which is partitioned into $\frac{N}{m} \times \frac{N}{m}$ blocks each of size $m \times m$. The multiple block summations are to compute the summation for each of these $\frac{N}{m} \times \frac{N}{m}$ blocks and store the result in the top left corner of each block. That is,

$$bs_{im,\ jm} = \sum_{\alpha=0}^{m-1}\sum_{\beta=0}^{m-1} a_{im+\alpha,\ jm+\beta}, \tag{2}$$

where $0 \leq i,\ j < \frac{N}{m}$.
By Lemma 1, the multiple block summations for computing Eq. (2) can be easily derived. This leads to the following lemma.

Lemma 2 *[23] The multiple block summations can be computed in $O(\log m)$ time on a 2-D $N \times N$ MCCHB.* □

3.2 The Adjacent Segmented Rotation

Let $A = (a_{0,\ im+\beta})$, $0 \leq \beta < m$, $0 \leq i < \frac{N}{m}$, be a $1 \times N$ data matrix which is partitioned into $\frac{N}{m}$ segments with each segment containing m consecutive data items. A is initially stored in the first memory location of the 0th row of an $m \times N$ matrix, where each element of the matrix contains $O(\frac{N}{m})$ memory location. For each memory location t, $0 \leq t < \frac{N}{m}$, the adjacent segmented rotation is to rotate the data items of all $\frac{N}{m}$ data segments of the 0th-row to left or right by t segments. That is,

$$a'_{0,\ im+\beta}[t] = a_{0,\ ((i\pm t)m+\beta) \bmod N}[0], \tag{3}$$

where $0 \leq \beta < m$ and $0 \leq t,\ i < \frac{N}{m}$.

Eq. (3) can be computed in $O(\frac{N}{m})$ time on a 2-D $m \times N$ MCCHB. At each rotation, we rotate the data items of each segment to left or right by one segment simultaneously, and store it into its corresponding memory location. After repeating $\frac{N}{m} - 1$ times, each memory location has owned the data item of its adjacent $\frac{N}{m} - 1$ segments. To rotate the data in constant time at each iteration, the rotation operation can be done as follows. Let the $m \times N$ matrix be viewed as $\frac{N}{m}$ blocks each of size $m \times m$. We first copy the data items within each block to their corresponding diagonal processors along i_1-dimension, then rotate them to left or right by one segment along i_0-dimension, and then copy them back to their corresponding rows along i_1-dimension. Hence, this leads to the following lemma.

Lemma 3 *The adjacent segmented rotation can be computed in $O(\frac{N}{m})$ time on a 2-D $m \times N$ MCCHB.* □

3.3 The Consecutive Segmented Rotation

Let $A = (a_{0,\ im+\beta})$, $0 \leq \beta < m$, $0 \leq i < \frac{N}{m}$, be a $1 \times N$ data matrix which is partitioned into $\frac{N}{m}$ segments with each segment containing m consecutive data items. A is initially stored in the 0th row of an $m \times N$ matrix. For $0 \leq t < m$, the consecutive segmented rotation is to rotate the data items of all segments of the 0th row to left or right by t positions and put it to the tth row. That is,

$$a'_{t,\ im+\beta} = a_{0,\ (im+\beta\pm t) \bmod N}, \tag{4}$$

where $0 \leq t,\ \beta < m$ and $0 \leq i < \frac{N}{m}$.

By the doubling recursive technique, the consecutive segmented rotation can be computed as follows. Initially, set $a'_{0,\ im+\beta} = a_{0,\ im+\beta}$, $0 \leq \beta < m$, $0 \leq i < \frac{N}{m}$. At the first iteration, rotate $a'_{0,\ (im+\beta\pm\frac{m}{2}) \bmod N}$ to $a'_{\frac{m}{2},\ im+\beta}$. Then, at the second iteration, rotate $a'_{0,\ (im+\beta\pm\frac{m}{4}) \bmod N}$ and $a'_{\frac{m}{2},\ (im+\beta\pm\frac{m}{4}) \bmod N}$ to $a'_{\frac{m}{4},\ im+\beta}$ and $a'_{\frac{m}{2}+\frac{m}{4},\ im+\beta}$ simultaneously. Following this way, the tth row which comes from the initial matrix can be obtained after $\log m$ iterations. Thus, Eq. (4) can be formulated by the following recursive form.

Initially,

$$a'_{0,\ im+\beta} = a_{0,\ im+\beta}, \qquad 0 \le \beta < m,\ 0 \le i < \frac{N}{m}.$$

For each iteration l, $1 \le l \le k$,

$$a'_{\frac{m}{2^{l-1}}\cdot t+\frac{m}{2^l},\ im+\beta} = a'_{\frac{m}{2^{l-1}}\cdot t,\ (im+\beta\pm\frac{m}{2^l}) \bmod N}, \tag{5}$$

where $0 \le t < 2^{l-1}$, $0 \le \beta < m$, $0 \le i < \frac{N}{m}$.
It means that at each iteration l there are 2^{l-1} rows to rotate their data items to left or right by $\frac{m}{2^l}$ positions, simultaneously.

The main idea for computing Eq. (5) can be described as follows. At the l^{th} iteration, the $m \times N$ matrix can be viewed as 2^{l-1} horizontal slices each of size $\frac{m}{2^{l-1}} \times N$ and each slice can be further partitioned into $2^{l-1} \cdot \frac{N}{m}$ sub-blocks each of size $\frac{m}{2^{l-1}} \times \frac{m}{2^{l-1}}$. There are three major steps for each iteration. The data items of each sub-block are first copied to their diagonal processors along i_1-dimension, then all of them are rotated to left or right by $\frac{m}{2^l}$ positions along i_0-dimension, finally all of them are copied back to their corresponding rows along i_1-dimension. Hence, this leads to the following lemma.

Lemma 4 *The consecutive segmented rotation can be computed in $O(\log m)$ time on a 2-D $m \times N$ MC-CHB.* □

3.4 The Window Broadcasting

Let $B = b_{\alpha,\ \beta}$, $0 \le \alpha,\ \beta < m$, be a template data matrix of size $m \times m$, where the data matrix B is stored in the top left corner of an image matrix of size $N \times N$. The window broadcasting operation is to distribute the template of size $m \times m$ over the $N \times N$ image matrix. That is,

$$b_{im+\alpha,\ jm+\beta} = b_{\alpha,\ \beta}, \tag{6}$$

where $0 \le \alpha,\ \beta < m$ and $0 \le i,\ j < \frac{N}{m}$.

By the power of the broadcasting buses, Eq. (6) can be computed in $O(1)$ time on a 3-D $m \times N \times N$ MCCHB. Hence, this leads to the following lemma.

Lemma 5 *The window broadcasting can be computed in $O(1)$ time on a 3-D $m \times N \times N$ MCCHB.* □

Since the data items of each template row are broadcasted independently, Lemma 5 can be easily modified to run in $O(\frac{m}{p})$ time when there are only pN^2 processors to be used for $1 \le p \le m$. This leads to the following corollary.

Corollary 1 *The window broadcasting be computed in $O(\frac{m}{p})$ time on a 3-D $p \times N \times N$ MCCHB, where $1 \le p \le m$.* □

4 The Image Template Matching

Let $A = (a_{i_1,\ i_0})$ represent an $N \times N$ image and $B = (b_{\alpha,\ \beta})$ represent an $m \times m$ template, where $0 \le \alpha,\ \beta < m$, $0 \le i_1,\ i_0 < N$ and $m \le N$. Assume both the image and the template are with the same domain D. The image template matching problem is to compare the template with all possible windows of the image, and the result of each window operation is stored in the top left corner of the window. That is, let $C = (c_{i_1,\ i_0})$, $0 \le i_1,\ i_0 < N$, be the result of each window operation. Then, $c_{i_1,\ i_0}$, $0 \le i_1, i_0 < N$, can be formulated by

$$c_{i_1,\ i_0} = \sum_{\alpha=0}^{m-1}\sum_{\beta=0}^{m-1} a_{(i_1+\alpha) \bmod N,\ (i_0+\beta) \bmod N} \times b_{\alpha,\ \beta}. \tag{7}$$

It is clear that the lower bound of Eq. (7) is $\Omega(m^2N^2)$ time in a unit processor system.

4.1 Algorithm TMA-1

Let the $N \times N$ image data items $a_{i_1,\ i_0}$, $0 \le i_1,\ i_0 < N$, be partitioned into $\frac{N}{m} \times \frac{N}{m}$ sub-images each of size $m \times m$ (i.e., $a_{im+\alpha,\ jm+\beta}$, $0 \le \alpha,\ \beta < m$, $0 \le i,\ j < \frac{N}{m}$). Then, for the vertical and horizontal displacements s and t relative to the top left corner of the image, Eq. (7) can be rewritten as

$$c_{im+s,\ jm+t} = \sum_{\alpha=0}^{m-1}\sum_{\beta=0}^{m-1} a_{(im+s+\alpha) \bmod N,\ (jm+t+\beta) \bmod N} \times b_{\alpha,\ \beta}, \tag{8}$$

where $0 \le s,\ t < m$, $0 \le i,\ j < \frac{N}{m}$.

For each pair of s and t displacements specified in Eq. (8), we can assign the $(sm+t)^{th}$ layer to compute the $c_{im+s,\ jm+t}$ (there are $\frac{N^2}{m^2}$ terms to be computed). Totally, it requires m^2 layers to compute Eq. (8). As for the m^2 layers to be simultaneously computed, we must rotate the image data items and broadcast the template data items to their corresponding layers (i.e., $a'_{sm+t,\ im+\alpha,\ jm+\beta} = a_{(im+\alpha+s) \bmod N,\ (jm+\beta+t) \bmod N}$ and $b'_{sm+t,\ im+\alpha,\ jm+\beta} = b_{\alpha,\ \beta}$). Then, Eq. (8) can be reformulated as

$$c_{sm+t,\ im,\ jm} = \sum_{\alpha=0}^{m-1}\sum_{\beta=0}^{m-1} a'_{sm+t,\ im+\alpha,\ jm+\beta} \times b'_{sm+t,\ im+\alpha,\ jm+\beta}, \tag{9}$$

where $0 \le s,\ t < m$, $0 \le i,\ j < \frac{N}{m}$.

Based on Eq. (9), the image template matching can be computed in $O(\log m)$ time on a 3-D MCCHB using m^2N^2 processors. Let a 3-D MCCHB of size $m^2 \times N \times N$ consist of m^2 2-D MCCHB each of size $N \times N$. Each 2-D MCCHB is a layer of the 3-D MCCHB and denoted as MCCHB$_{i_2}$, $0 \le i_2 < m^2$. Initially, the image and template are stored in the local variables $a(0,\ im+\alpha,\ jm+\beta)$ and $b(0,\ \alpha,\ \beta)$, $0 \le \alpha,\ \beta < m$, $0 \le i,\ j < \frac{N}{m}$, of 2-D MCCHB$_0$, respectively. Finally, the results are stored in the local variable $c(0,\ im+\alpha,\ jm+\beta)$ of the 2-D MCCHB$_0$. Thus, there are m^2 layers for all pairs of s and t displacements; each layer has $\frac{N}{m} \times \frac{N}{m}$ blocks and each block has $m \times m$ product terms for computing each $c_{sm+t,\ im,\ jm}$ of Eq. (9). That is, we can assign the $(sm+t)^{th}$ layer (i.e., MCCHB_{sm+t}) to compute $c_{sm+t,\ im,\ jm}$ of Eq. (9) with a fixed pair of s and t displacements. The image template matching algorithm (TMA-1) consists of the following five steps. Step 1, broadcast the template data items of the MCCHB_0 over all MCCHB_α, $0 \le \alpha < m^2$. Step 2, rotate the image data items of the MCCHB_0 up to the MCCHB_{sm} by s displacements. Step 3, rotate the image data items of the MCCHB_{sm} left to the MCCHB_{sm+t} by t displacements. Step 4, compute the product terms and accumulate the sum of products. Step 5, copy $c(sm+t,\ im,\ jm)$ of the MCCHB_{sm+t} back to its corresponding position of MCCHB_0.

Theorem 1 *Algorithm TMA-1 can be computed in* $O(\log m)$ *time on a* 3-D $m^2 \times N \times N$ MCCHB. □

4.2 Algorithm TMA-2

Based on the communication ability of the broadcasting buses and the concept as stated by Horng et al. [12], the image template matching problem can be solved in $O(m^2)$ time on the 2-D MCCHB. Assume that each processor contains $O(m)$ restricted memory. The image template matching algorithm (TMA-2) for computing Eq. (7) consists of three steps. Step 1, rotate the image data items left to each processor along each row of the i_0-dimension such that each processor stores the right $m-1$ consecutive data items (i.e., rotate $a'_{i_1,\ i_0}[\beta] = a_{i_1,\ (i_0+\beta) \bmod N}$ for $0 \le \beta < m$). Step 2, sequentially broadcast the template data to all processors, then multiply it with the corresponding image data item, and then accumulate the partial sum of each row along i_0-dimension. That is, for each α, $0 \le \alpha < m$, compute $d_{i_1,\ i_0}[\alpha]$ by

$$d_{i_1,\ i_0}[\alpha] = \sum_{\beta=0}^{m-1} a'_{i_1,\ i_0}[\beta] \times b_{\alpha,\ \beta}, \qquad (10)$$

where $0 \le i_1,\ i_0 < N$.
Finally, Step 3, accumulate the partial sum of each column along i_1-dimension by

$$c_{i_1,\ i_0} = \sum_{\alpha=0}^{m-1} d_{(i_1+\alpha) \bmod N,\ i_0}[\alpha], \qquad (11)$$

where $0 \le i_1,\ i_0 < N$.

Theorem 2 *Algorithm TMA-2 can be computed in* $O(m^2)$ *time on a* 2-D $N \times N$ MCCHB *with each processor containing* $O(m)$ *restricted memory.* □

While each processor contains only $O(1)$ restricted memory, algorithm TMA-2 can be easily modified to run in $O(m^2)$ time. The interesting reader can refer to [24] for details. Hence, the following corollary is hold.

Corollary 2 *The image template matching problem can be solved in* $O(m^2)$ *time on a* 2-D $N \times N$ MCCHB *with each processor containing* $O(1)$ *restricted memory.* □

4.3 Algorithm TMA-3

By combining algorithm TMA-1 and algorithm TMA-2, a more efficient image template matching algorithm which runs in $O(\frac{m^2}{p^2} + \frac{m}{p}\log p)$ time will be derived in this subsection, where each processor contains $O(\frac{m}{p})$ restricted memory for $1 \le p \le m \le N$. Let the $N \times N$ image data items $a_{i_1,\ i_0}$, $0 \le i_1,\ i_0 < N$, and the $m \times m$ template $b_{\alpha,\ \beta}$, $0 \le \alpha,\ \beta < m$, both be partitioned into $\frac{N}{p} \times \frac{N}{p}$ sub-images and $\frac{m}{p} \times \frac{m}{p}$ sub-templates each of size $p \times p$ (i.e., $a_{i'p+\gamma,\ j'p+\delta}$ and $b_{up+\gamma,\ vp+\delta}$, $0 \le \gamma,\ \delta < p$, $0 \le i',\ j' < \frac{N}{p}$, $0 \le u,\ v < \frac{m}{p}$). Then, for any pair of vertical and horizontal displacements s' and t' relative to the top left corner of the image, Eq. (7) can be rewritten by

$$c_{i'p+s',\ j'p+t'} = \sum_{u=0}^{\frac{m}{p}-1} \sum_{v=0}^{\frac{m}{p}-1} \sum_{\gamma=0}^{p-1} \sum_{\delta=0}^{p-1} a_{(i'p+s'+up+\gamma) \bmod N,\ (j'p+t'+vp+\delta) \bmod N} \times b'_{up+\gamma, vp+\delta}, \qquad (12)$$

where $0 \le s',\ t',\ \gamma,\ \delta < p$, $0 \le u,\ v < \frac{m}{p}$ and $0 \le i',\ j' < \frac{N}{p}$.

The main idea for computing Eq. (12) is as follows. Based on the idea of algorithm TMA-1, we first rotate the image data items located on the 0th layer to all p^2 layers such that each processor of the $(s'p+t')^{th}$ layer stores the image data item up by s' vertical displacement and left by t' horizontal displacement corresponding to the original image data located on the 0th layer. Then based on the idea of algorithm TMA-2, each layer computes $c_{i'p+s',\ j'p+t'}$ of Eq. (12), simultaneously. That is, there are $\frac{N}{p} \times \frac{N}{p}$ terms to be computed in each layer. Let $a'_{s'p+t',\ i'p+\gamma,\ j'p+\delta} = a_{(i'p+s'+\gamma) \bmod N,\ (j'p+t'+\delta) \bmod N}$ and $b'_{s'p+t',\ up+\gamma,\ vp+\delta} = b_{up+\gamma,\ vp+\delta}$, where $0 \le s',\ t',\ \gamma,\ \delta < p$, $0 \le i',\ j' < \frac{N}{p}$ and $0 \le u,\ v < \frac{m}{p}$. By rearranging the summation order, Eq. (12) can be rewritten as

$$c_{s'p+t',\ i'p,\ j'p} = \sum_{u=0}^{\frac{m}{p}-1} \sum_{\gamma=0}^{p-1} \sum_{\delta=0}^{p-1} \sum_{v=0}^{\frac{m}{p}-1} a'_{s'p+t',\ (i'p+up+\gamma) \bmod N,\ (j'p+vp+\delta) \bmod N} \times b'_{s'p+t',\ up+\gamma,\ vp+\delta}, \qquad (13)$$

where $0 \le s',\ t' < p$, $0 \le i',\ j' < \frac{N}{p}$.

By the data manipulation operations proposed in the previous section, Eq. (13) can be also computed in $O(\frac{m^2}{p^2} + \frac{m}{p}\log p)$ time on a 3-D $p^2 \times N \times N$ MCCHB. The interested reader can refer to [24] for details. Hence, this leads to the following theorem.

Theorem 3 *The image template matching can be computed in* $O(\frac{m^2}{p^2} + \frac{m}{p}\log p)$ *time on a* 3-D $p^2 \times N \times N$ MCCHB *with each processor containing* $O(\frac{m}{p})$ *restricted memory, where* $1 \le p \le m$. □

5 Concluding Remarks

The architecture of the MCCHB is regular and the number of links of it is less than that of the mesh of tree and the hypercube. Hence, it is quite possible to implement such an architecture constructed by thousands of processors through currently VLSI technology and suitable for image processing.

The image template matching is a basic operation in image processing. Following the algorithms proposed in this paper, we find the results are far better than those derived before [5, 6, 7, 14, 19, 20, 21, 23]. Furthermore, the result derived in Tsai et al. [23] is a special case of this paper. Our result achieves the same time complexity with that of Tsai et al. [23] in that case but reduces the number of processors by a factor of m. Clearly, our algorithm is both optimal and optimal speed-up in time complexities.

References

[1] A. Aggarwal, "Optimal Bounds for Finding Maximum on Array of Processors with k Global Buses," *IEEE Trans. on Comput.* 35 62-64 (1986).

[2] S. H. Bokhari, "Finding Maximum on an Array Processor with a Global Bus," *IEEE Trans. on Comput.* 33 133-139 (1984).

[3] K. L. Chung, "Prefix Computations on a Generalized Mesh-Connected Computers with Multiple Buses," *IEEE Trans. on Parallel and Distributed Systems* 6 196-200 (1995).

[4] T. Y. Feng, "A Survey of Interconnection Networks," *Computers* 12-27 (1981).

[5] Z. Fang, X. Li and L. M. Ni, "Parallel Algorithms for Image Template Matching on Hypercube SIMD Computers," *IEEE Workshop on Comput. Architect. for Patt. Anal. and Image Database Manag.* 33-40 (1985).

[6] Z. Fang and L. M. Ni, "Parallel Algorithms for 2D Convolutions," *Int. Conf. on Parallel Processing* 262-269 (1986).

[7] Z. Fang and L. M. Ni, "On the Communication Complexity of Generalized 2-D Convolution on Array Processors," *IEEE Trans. on Comput.* 38(2) 184-193 (1989).

[8] K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*, Reading, New York: McGraw-Hill, 1984.

[9] S. J. Horng, "Generalized Mesh-Connected Computers with Hyperbus Broadcasting for a Computer Network," *IEICE Trans. on Inform. and Syst.* E79-D(8) 1107-1115 (1996).

[10] S. J. Horng, "Semigroup Computation and Its Applications on Mesh-Connected Computers with Hyperbus Broadcasting," *Int. Conf. on Parallel and Distributed Systems* 34-39 (1995).

[11] S. J. Horng, "Constant Time Algorithm for Template Matching on a RAP," *The Computer Journal* 36(2) 246-253 (1993).

[12] S. J. Horng, W. T. Chen and M. Y. Fang, "Optimal Speed-up Algorithms for Template Matching on SIMD Hypercube Multiprocessors with Restricted Local Memory," *Information Processing Letters* 38(1) 29-37 (1991).

[13] J. F. Jenq and S. Sahni, "Reconfigurable Mesh Algorithms for Image Shrinking, Expanding, Clustering, and Template Matching," *Int. Parallel Processing Symp.* 208-215 (1991).

[14] V. K. P. Kumar and V. Krishnan, "Efficient Image Template Matching on Hypercube SIMD Arrays," *Int. Conf. on Parallel Processing* 765-771 (1987).

[15] V. K. P. Kumar and C. S. Raghavendra, "Array Processor with Multiple Broadcasting," *Journal of Parallel and Distrib. Comput.* 2(4) 173-190 (1987).

[16] H. Li and M. Maresca, "Polymorphic-Tours Architecture for Computer Vision," *IEEE Trans. on Patt. Analys. and Machine Intell.* 11 233-243 (1989).

[17] M. Maresca and H. Li, "Connection Autonomy in SIMD Computers: a VLSI Implementation," *Journal of Parallel and Distrib. Comput.* 7 302-320 (1989).

[18] R. Miller, V. K. P. Kumar, D. Reisis and Q. F. Stout, "Parallel Computations on Reconfigurable Meshes," *IEEE Trans. on Comput.* 42 678-692 (1993).

[19] X. Qu and X. Li, "Parallel Template Matching Algorithms," *Int. Conf. on Parallel Processing* 223-225 (1988).

[20] S. Ranka and S. Sahni, "Image Template Matching on SIMD Hypercube Computers," *Int. Conf. on Parallel Processing* 84-91 (1988).

[21] S. Ranka and S. Sahni, "Convolution on SIMD Mesh Connected Multicomputers," *Int. Conf. on Parallel Processing* 212-217 (1988).

[22] L. J. Seigel, H. J. Seigel and A. E. Feather, "Parallel Processing Approaches to Image Correlation," *IEEE Trans. on Comput.* 31(3) 208-217 (1982).

[23] S. S. Tsai, S. J. Horng and T. W. Kao, "Optimal Template Matching on Mesh-Connected Computers with Hyperbus Broadcasting," *Int. Conf. on Computation and Information* 336-348 (1995).

[24] H. R. Tsai, S. J. Horng, S. S. Tsai, S. S. Lee and T. W. Kao, "Time-Optimal Parallel Image Template Matching Algorithms on the Mesh-Connected Computers with Hyperbus Broadcasting," *Tech. Rept.*, Dept. of Electrical Eng., National Taiwan University of Sci. and Tech., 1996.

[25] B. F. Wang, G. H. Chen and F. C. Lin, "Constant Time Sorting on a Processor Array with a Reconfigurable Bus System," *Information Processing Letters*, 34 187-192 (1990).