Massively Parallel Approach to Pattern Recognition

Wu Wang *S. Sitharama Iyengar* *Jianhua Chen*

Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803

ABSTRACT

Template matching is concerned with measuring the similarity between patterns of two objects. This paper proposes a massively parallel approach to pattern recognition with a large template set. A class of image recognition problems inherently needs large template sets, such as the recognition of Chinese characters which needs thousands of templates. The proposed algorithm is based on the SIMD-SM-R machine or the SIMD machine with broadcasting abilities, which is the most popular parallel machine to date, using a multiresolution method to search for the matching template. The approach uses the pyramid data structure for the multiresolution representation of templates and the input image pattern. For a given image it scans the template pyramid searching for the match. Implementation of the proposed scheme is described.

1. INTRODUCTION

Pattern recognition is concerned primarily with the description and analysis of measurements taken from physical or mental processes. Finding appropriate properties of the object and matching the found properties are both computationally intensive and error-prone. Much work has been done on this area [1, 2, 7, 19, 20]. However, few methods seem to be satisfactory in real-time application, especially when there is a large number of known classes. In this paper, we propose a massively parallel algorithm for pattern recognition based on the SIMD-SM-R machines or the SIMD machines with broadcasting abilities.

An SIMD (Single-Instruction-Stream-Multiple-Data-Stream) machine consists of an array of identical processors. In the SIMD-SM model, reading and writing conflicts are not allowed. In SIMD-SM-R, simultaneous reading (but not writing) is allowed. In SIMD-SM-RW, simultaneous reading and writing is allowed [14]. If an SIMD machine has the broadcasting abilities, then such an SIMD machine is functionally equivalent to an SIMD-SM-R machine. Therefore, we present our approach in terms of these two types of SIMD model interchangeably. By nature, the SIMD model is very suitable for application in image processing. However, parallelization has barely explored on the recognition level. In this paper, we propose an algorithm which matches an input image with a large template set. Little work on dealing with large template sets can be found in literature.

Li, et al. [13] describe an experiment using Ramapriyan's [15] template tree approach and propose a tree generating procedure. Their experiment was not run on a parallel machine. The tree search method compares an image with each of the child nodes and selects the one with the greatest similarity to continue the search and stops when a sufficient match is found with a leaf node (true template). However, it is hardly a scheme which can guarantee that every branch examined during its search leads to the appropriate template leaf. Moreover, the template tree takes up twice as much the memory space as the templates themselves and the template tree is not easy to update.

There is no general way to search memory for the best match without examining every element of memory. Instead of a brute force linear search through K templates in **M**, we propose an algorithm on the SIMD-SM-R machine to match all K templates using multiresolution techniques. Our approach has $O(\log N)$ times comparisons where the image is composed of $N \times N$ pixels. We will explain what "a comparison" means shortly.

The SIMD machine in our focus has two types of communication facilities: the processors are connected by a mesh into a grid network, allowing rapid direct communication between neighboring processors, and by the global broadcasting mechanism which provides a **SEND** instruction that takes a broadcasting cycle to reach any destination processor from a source processor. A broadcasting cycle is considered as a time unit. Some machines with these abilities are described in [4, 5, 8].

Based on the facilities of the SIMD-SM-R machine, the primitive operation *plus-scan* can be defined [9, 21]. The *plus-scan* operation can be implemented in microcode and it takes $\log_2 K$ time units to *scan* a vector of length K. Figure 1 shows the *plus-scan* operation.

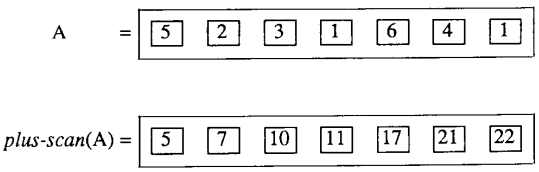


Figure 1: *An Example of* Plus-Scan

Our approach makes use of the hierarchical pyramidal data structure [19] to perform the multiresolution matching. The pyramid of arrays described by Tanimoto [19] is a sequence $\{M(L), M(L-1), ..., M(1)\}$ of arrays where $M(L)$ represents an original image and $M(i-1)$ is a version of $M(i)$ at half the resolution of $M(i)$ with $M(1) = 2 \times 2$ pixels. A pyramid for an image is constructed from an image array using an averaging rule:

$$M(i)[p][q] = \left\lceil \frac{\sum_{x=0}^{x=1}\sum_{y=0}^{y=1} M(i+1)[2p+x][2q+y]}{4} \right\rceil, 0 \le p, q \le 2^i - 1 \quad (1.1)$$

where p and q denote the coordinates of the pixels and $1 \le i \le L$. The whole array $M(i)$ can be obtained by averaging all 2×2 components of $M(i+1)$ simultaneously via neighbor communications in several time units. Theorem 1.1 follows immediately.

Theorem 1.1. *M(i)* can be computed from $M(i+1)$ in constant time.

If the SIMD-SM-R machine has $N' \times N'$ processors, one operation on every element of an $N \times N$ array can be done in $\left\lceil \frac{N \times N}{N' \times N'} \right\rceil$ steps [21]. Hereafter, we assume that the SIMD-SM-R machine has $N \times N$ processors where $N = 2^L$, and the templates and image patterns are $N \times N$ arrays of pixels. We will ease the assumption that the machine has $N \times N$ processors in the end of Section 3.

For a template T and an image P of the same size $m \times m$, the comparison operation COMP(T,P) returns a measure s of dissimilarity between T and P where the measure s is a numerical value, which is used to determine the confidence level of T matching P. Obviously, there are many specific implementations of COMP operations according to the image recognition tasks, and the time complexity $O(f(m))$ of such COMP's is dependent on the specific comparison scheme selected. In this paper, we are concerned with neither what specific scheme used in COMP's nor what the complexity $O(f(m))$ of such an operation is.

Definition 1. The execution of a parallel version of COMP(T,P) on an SIMD machine is considered one *comparison operation.*

Assumption 1. Assume that the COMP is implemented in such a way that if a region of the processor array is assigned to execute COMP(T',P) and another region of the processor array is assigned to execute COMP(T'',P), then COMP(T',P) and COMP(T'',P) can be carried out under one single instruction stream, i.e., COMP(T',P) and COMP(T'',P) can be executed simultaneously if there are enough processors on the SIMD machine to allocate the (T',P) pair and the (T'',P) pair at the same time.

Assumption 1 is the basis of our algorithm. The assumption is reasonable since some comparison schemes satisfy this assumption. We will show one example in Section 4.

This paper is organized as follows. Section 2 describes our approach and gives algorithms for constructing the template database and recognizing an image. In Section 3, we discuss the performance of our approach. Section 4 gives some simulation results.

2. PROPOSED APPROACH

An SIMD-SM-R machine with massively parallel processors and high connectivity can be suitable for image recognition tasks [21]. In the proposed approach, a database **M** of templates is constructed and the image recognition is carried out using the facilities provided by the SIMD-SM-R machine.

For our system, the database is a set of pyramids of templates. Let $N = 2^L$, then each pyramid for a template is composed of $N \times N, \frac{1}{2}N \times \frac{1}{2}N, ..., 2 \times 2$ arrays. We use $m_k(i)$ to denote the array of level i of the kth template pyramid and K to denote the number of templates, i.e. $|\mathbf{M}| = K$. The structure of the database is also appropriate for learning new templates after the recognition process is started since the database is a set of pyramids. Learning is simply adding a pyramid to the database.

Constructing a pyramid for template m_i has a time complexity of $O(\log N)$. A pyramid needs at most $\frac{4}{3}N^2$ pixels of memory [21]. The construction of the database of our approach is simply to learn all templates in **M**. Constructing the database has a computational complexity of $O(K \log N)$ to construct the database and the database needs $\frac{4}{3}N^2K$ pixels of memory where K = |M|.

We also need to construct a pyramid for the input image pattern before the recognition process proceeds. The procedure for constructing the input pattern pyramid is the same as the one for constructing template pyramids.

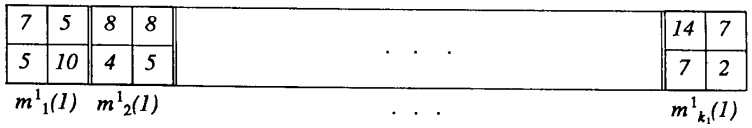
After all pyramids of templates and the pattern are ready, the multiresolution matching process can be started. The idea of multiresolution matching method is the following. The system has a set of thresholds, each of which is associated with every level of pyramids. If COMP(m,P) at level i is greater than *threshold(i)*, it does not necessarily compare template m with the pattern at levels $i+1, ..., L$. We have the assumption that COMP(m,P) is a parallel procedure on an SIMD machine. If the data array is smaller than the physical processor array, then there will be some processors working on the data array while the rest are left idle. On the other hand, if the data array is larger, the SIMD-SM-R machine can work on the data array portion by portion. Since the array size at level i of a pyramid is $2^i \times 2^i$ and at level $i+1$ is $2^{i+1} \times 2^{i+1}$, at level i this comparison needs only one-fourth processers as at level $i+1$ on the SIMD-SM-R machine.

Under assumption 1, one can see that in one comparison operation we can compare more templates at higher levels than at lower levels of the pyramids if we can make use of all processors. In order to maximize the utilization of processors and speed up the matching process, we first construct a pixel plane consisting of all top level arrays of K template pyramids and another pixel plane consisting of K top level arrays of the input pattern pyramid as shown in Figure 2.

Theorem 2.1. A pixel plane consisting of a set of smaller arrays can be constructed in one operation.

Proof. The SIMD-SM-R machine or the SIMD machine

Note: $m^1_q(l)$ is the level l array of the pyramid of template m_q which is selected for level l comparison. k_l is the number of selected templates for l th level comparison.



Note: P (l) is level l of the input pattern pyramid.

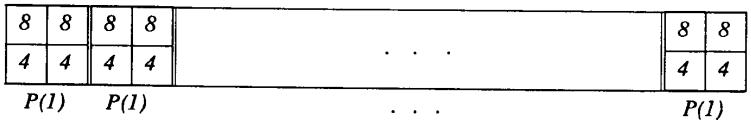


Figure 2: *Pixel Planes for Matching*

with broadcasting abilities allows any processor to communicate directly with any other processor in unit time, while other processors also communicate concurrently. The factor that the pixel plane may be larger than the processor array is discussed in the performance analysis in Section 3. □

By assumption 1, the comparisons COMP(m_i^1(1),P(1)) for $1 \le i \le k_1$ can be simultaneously executed on the processor array regions corresponding to m_i^1(1) and P(1). Therefore, the machine compares all template m_i^1(1)'s with pattern P(1) in one comparison operation. This results in a vector as shown in Figure 3 which is dissimilarity factors.

$$S_1 = (s[1], s[2], \ldots, s[k_1])$$

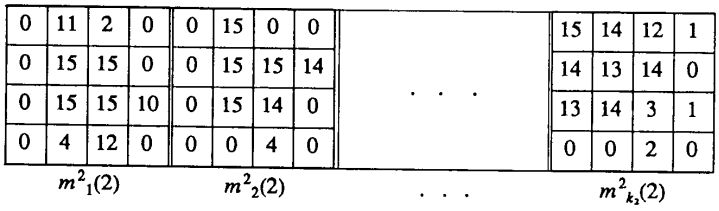
Figure 3: *Vector of the Regions Dissimilarities*

Theorem 2.2. The dissimilarity vector **S** can be computed in one operation.

Proof. The proof follows from the above illustration.

Suppose there are k_2 elements of the dissimilarity vector **S** less than the *threshold*(1). Then we use level 2 arrays of these k_2 pyramids to construct a template pixel plane and a pattern pixel plane consisting of k_2 level 2 arrays of pattern pyramid. Figure 4 shows the template and pattern pixel planes.

Repeat the process of constructing pixel planes and comparing pixel planes for level 2 , level 3, and so on. On the bottom level we will get all matchings whose dissimilarity measures are less than *threshold(L)*. The idea is that when the array size we need to compare increases, the number of templates



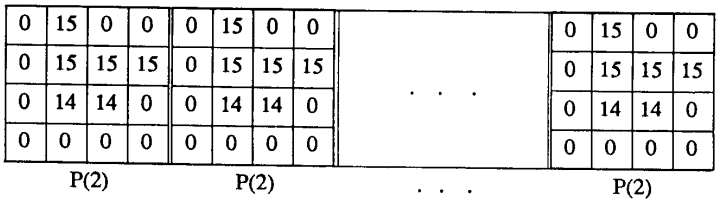


Figure 4: *Template and Pattern Pixel Planes for Level 2*

we need to compare probably decreases, if *thresholds* are appropriate. We expect that the pixel plane size for comparison is at most several times that of the SIMD-SM-R machine processor array size.

Theorem 2.3. Selecting templates for next level comparison according to the dissimilarity vector S can be done in one *plus-scan* operation with a time complexity of $O(\log k_{i+1})$ where k_{i+1} is the number of templates selected for next level comparison.

Proof. Introduce 2 vectors *model*[1] and *model*[2] as indicators. *Model*[2][p] indicates the pyramid of the corresponding template for comparison in region p where S_i[p] is obtained.

Step(1). Set *model*[1][p] = 1 if S_i[p] < *threshold(i)* , else *model*[1][p] = 0.

Step(2). If *model*[1][p] = 1 then the processor containing it is set to active, else set to inactive.

Step(3). k_{i+1} = Plus-Scan(*model*[1]). After the *plus-scan* operation , as shown in Figure 1, *model*[1] looks like this (1 2 3 4 5 ... k_{i+1}) for origin as (1 1 ... 1), where the inactive processors are omitted while their contents are remained as 0's.

Step(4). If *model*[1][p] ≠ 0, set *model*[2][*model[1][p]*] = *model*[2][p].

The time complexity $O(\log k_{i+1})$ is due to step 2. □

The formal description of the recognition procedure can be given as algorithm 2. Algorithm 1 is to initialize the system by constructing the database, initialize model[2] and setting up the first template pixel plane MBP for the sake of efficiency.

Algorithm 1

Input: A template set **M**.

Output: A database as a set of pyramids of templates, a template pixel plane for top level comparison, and vector model[2] initialized pointing to all templates.

```
1  procedure INITIALIZATION(M);
2  begin
3    BASE(M);   /* construct the template database */
4    par  1 ≤ i ≤ K,  0 ≤ p, q ≤ 1  do
5       model[2][i] = i; /* denotes the template needed to
         be matched */
6       MBP[i][p][q] = m_i(1)[p][q];   /* first template pixel
         plane */
7    endpar
8  end
```

Algorithm 2

Input: A database of template pyramids, a vector model[2] of pointers pointing to the template pyramids that should be compared, a template pixel plane MBP for the first level comparison, an input image pattern P, and a set of *thresholds*.

Output: Matched templates with similarity confidences, or the input pattern is learned if no matching is found.

```
1  procedure RECOGNITION;
2  begin
3    PATTERN(P); /* construct the pattern pyramid */
4    size = 2; /* size indicates the region size for com-
       parison */
5    m = | M |; /* m contains the content of k_1, k_2, ..., k_L
       */
6    for level = 1 to L do /* matching the templates */
7      par 1 ≤ i ≤ m, 0 ≤ p, q ≤ size - 1 do /* construct pat-
         tern pixel plane as shown in theorem 2.1 */
8        PBP[i][p][q] = P(level)[p][q];
9      endpar
10     par 1 ≤ p ≤ m do
11       s[p] = COMP( PBP[p], MBP[p] ) /* PBP[p]
           represents the pth region of the pattern pixel plane,
           and MBP[p] represents the pth region of the template
           pixel plane, refer to theorem 2.2 */
12       Select templates for next level comparison; Number
           of selected templates is in m, i.e. m = k_{i+1}; Pointers
           pointing to selected templates are in model[2]; /*
           refer to theorem 2.3 */
13     endpar
14     if level = L then exit endif;
15     size = 2 * size; /* increase the region size for next
         level */
16     par 1 ≤ i ≤ m, 0 ≤ p, q ≤ size - 1 do /* constructs
         template pixel plane for next level */
17       MBP[i][p][q] = M_{model[2][i]}(level+1)[p][q]
18     endpar
19   endfor /* from statement 6 */
20   if m = 0 then LEARN(P(L))
21   else CONFIDENCE(m, model) /* assign confidence
       level of matching */
22   endif
23 end    { RECOGNITION }
```

---

3. PERFORMANCE ANALYSIS

The time complexity of the database construction for the system is $O(K \log N)$, and space needed is $\frac{4}{3} N^2 K 2^t$ where K is the number of templates, an image is composed of $N \times N$ pixels and each pixel needs 2^t bits memory. The time complexity and the space overhead is reasonable since the time lower bound for the database construction is $O(K)$ and the space lower bound for the template set **M** is $O(N^2 K 2^t)$. Since the comparison operation COMP(T,P) is the dominant factor in template matching process, we analyze our algorithm in terms of times of comparisons.

Theorem 3.1. Algorithm 2 needs $O(\log N)$ comparison operations in general while an image is composed of $N \times N$ pixels and is $O(K)$ comparison operations in the worst case where

$K = |\mathbf{M}|$.

Proof. Statement 3 has a computation complexity $O(\log N)$. Statements 6 to 19 form a loop of $\log N$ times computation, by which the computational complexity is decided. Let's analyze statements 7 to 18.

Each statement from 7 to 18 except for 11 and 12 can be done in one operation. Each *plus-scan* in Statement 12 takes $O(\log K_i)$ time units. Thus, in worst case the time complexity for Statement 12 is $O(\log K)$ in each iteration. Since the dominant factor in computation of template matching is the comparison in Statement 11, let's focus on the analysis of Statement 11.

Since our approach is to select the $m_k \in \mathbf{M}$ such that $S(m_k(i),p(i)) < threshold(i)$ to continue matching on level $i+1$. For $1 \le i_j \le k_i$, if $S(m_{i_1}(i),p(i)) < threshold(i)$, $S(m_{i_2}(i),p(i)) < threshold(i)$, ..., $S(m_{i_{k_i}}(i),p(i)) < threshold(i)$, only choose $m_{i_1}, m_{i_2}, \ldots, m_{i_{k_i}}$ for matching on level $i+1$. Therefore, $|\mathbf{M}| = k_1 \ge k_2 \ge \ldots \ge k_{L-1} \ge k_L$ where L is the height of the pyramid, i.e. $L = \log N$.

If the SIMD-SM-R machine has $N \times N$ processors, statement 11 on level i will require $\left\lceil \frac{2^i \times 2^i \times k_i}{N \times N} \right\rceil$ comparison operations since the array size of template pyramid on level i is $2^i \times 2^i$ and there are k_i templates needed to be compared. One can see the worst case for statement 11 is that we still need to match $|\mathbf{M}|$ templates on the bottom level, i.e. $k_1 = k_2 = \ldots = k_L$. However, this case shouldn't happen unless the *thresholds* associated with each level are too high or the templates are all too similar since if $S(m_p(i),p(i)) < threshold(i)$ and $S(m_q(i),p(i)) < threshold(i)$, then $\frac{S(m_p(i),m_q(i))}{2} < threshold(i)$. Thus, the selection of *thresholds* is critical to the process [21].

Only the bottom level array of each template pyramid needs one comparison operation if the image size is equal to the machine processor array size. One comparison operation can compare 4 template arrays on level $(L-1)$, and one comparison operation can compare 4^2 template arrays on level $(L-2)$ and so on. On level 1 , one comparison operation can compare $\frac{N \times N}{2 \times 2}$ template arrays. If N = 256, i.e. a 64K-processor SIMD-SM-R machine, it can compare 16K template arrays on level 1 in one comparison operation or compare 2^{10} template arrays on level 2 in one comparison operation. Practically, the number of templates is seldom that large. Thus one comparison operation can usually compare all the selected template arrays on an upper level. By selecting the good *thresholds*, statement 11 will only take more than one comparison operation on the several lowest levels. Suppose statement 11 needs $C_i + 1$ comparison operations to compare the pixel planes on level i . Then statement 11 has $(C_1 + 1 + C_2 + 1 + \ldots + C_{L-1} + 1 + C_L + 1) = (C_1 + C_2 + \ldots + C_{L-1} + C_L) + \log N$ comparison operations in algorithm 2. As discussed before, C_1 and C_2 are usually equal to zero or no more than 1, and C_{L-2}, C_{L-1} and C_L are usually small numbers. Hence, on average the algorithm has $O(\log N)$ comparison operations

Theoretically, the worst case happens when we need to compare all template arrays on every level. All template arrays on level L need $|\mathbf{M}| = K$ comparison operations to compare and all template arrays on level $(L-1)$ need $\frac{1}{4}K$ comparison operations to compare and so on. Therefore, the algorithm totally needs

$$K + \frac{1}{4}K + \frac{1}{16}K + \ldots + \frac{1}{4^L}K \le \frac{4}{3}K$$

comparison operations to compare all template arrays on all levels, i.e. $O(K)$ comparison operations. □

One can see that the working space for the template pixel plane and the pattern pixel plane also has a large overhead in the algorithm above if the worst case occurs. In practice, we can use some simple techniques to keep a pattern pixel plane no large than the processor array since the SIMD-SM-R machine actually operates on a large array as on several smaller arrays. Similarly, we can construct the template pixel plane as large as the pattern pixel plane for every comparison. This revised implementation will not cause more overhead than the algorithms above. On the other hand , it constructs smaller pixel planes and hence decreases the overhead of communications and the space. We keep our algorithms in Section 2 "inefficient" because they are more easily understood.

4. SIMULATION

A program has been implemented on the VAX 11/780 to simulate the recognition of binary images. The comparison operation COMP(T,P) is to return the sum of all elements of the exclusive or of T and P. Since the purpose of the experiment was to verify the time complexity, the program simply uses an integer array to simulate the SIMD-SM-R machine processor array. The program simulates an "SIMD-SM-R machine" with $N \times N = 64 \times 64$ processors. Binary images of size 64×64 are used to run the program. English letters are used to test the program in the experiments.

The number of levels of the pyramid representing an image is six. All templates are upper case letters. Table 1 gives the number of nodes needed at each level, i.e. at statement 11 of the RECOGNITION algorithm, for recognizing each letter using the following values for thresholds

$$threshold = [1\ 2\ 4\ 8\ 32\ 64]^T$$

Since 'J' and 'I' are very similar, they are discriminated at the last level (using full resolution). This can be considered the case of similar templates and distorted images of one input pattern. Low values of thresholds would lose some distorted input images and high values would cause redundant matchings and degrade the performance. A low threshold will let the system treat some noises as the extra parts of the object so that the system rejects some possible templates. On the other hand, a high threshold will let the system treat some extra parts of the object as noises so that the system selects too many unmatched templates for higher-resolution matchings and thus degenerate the performance, especially at the higher-resolution levels where the processor array can not cover too many template arrays. Thus, selection of threshold values always involves tradeoffs.

Table_1: *Number of Nodes Needed at Each Level, i.e. number of elements of the pixel plane array needed to be mapped onto the 64 × 64 physical processor array*

Template	Level					
	1	2	3	4	5	6
'A'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'B'	26 × (2 × 2)	6 × (4 × 4)	4 × (8 × 8)	2 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'C'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	2 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'D'	26 × (2 × 2)	6 × (4 × 4)	4 × (8 × 8)	2 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'E'	26 × (2 × 2)	6 × (4 × 4)	5 × (8 × 8)	2 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'F'	26 × (2 × 2)	9 × (4 × 4)	5 × (8 × 8)	2 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'G'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'H'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'I'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	2 × (16 × 16)	2 × (32 × 32)	2 × (64 × 64)
'J'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	2 × (16 × 16)	2 × (32 × 32)	2 × (64 × 64)
'K'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'L'	26 × (2 × 2)	2 × (4 × 4)	2 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'M'	26 × (2 × 2)	2 × (4 × 4)	2 × (8 × 8)	2 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'N'	26 × (2 × 2)	2 × (4 × 4)	2 × (8 × 8)	2 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'O'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	3 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'P'	26 × (2 × 2)	4 × (4 × 4)	3 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'Q'	26 × (2 × 2)	2 × (4 × 4)	2 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'R'	26 × (2 × 2)	4 × (4 × 4)	4 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'S'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	2 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'T'	26 × (2 × 2)	4 × (4 × 4)	4 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'U'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'V'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'W'	26 × (2 × 2)	6 × (4 × 4)	1 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'X'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'Y'	26 × (2 × 2)	13 × (4 × 4)	13 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)
'Z'	26 × (2 × 2)	6 × (4 × 4)	5 × (8 × 8)	1 × (16 × 16)	1 × (32 × 32)	1 × (64 × 64)

Note: *Each element $k \times (s \times s)$ means k templates are selected to compare with the input pattern and the size of each template array at current level is $(s \times s)$. Thus, the comparison at each level needs* $\left\lceil \frac{k \times (s \times s)}{64 \times 64} \right\rceil$ *comparison operations.*

5. CONCLUSION

Recognition is one of the tasks a machine vision system must accomplish, i.e. the system needs some scheme to assign an object to one of a number of known classes. The main contributions of this paper are summarized as follows.

The approach uses a multiresolution method to search for the matching template. The pyramid data structure is used for the multiresolution representation of templates and the input image pattern. In many cases, the correct template can be found in a medium level of the pyramid. The number of templates, K, is usually not more than thousands and the image size $N \times N$ is generally greater than 128×128. That is , at levels 1 and 2, we usually can compare all templates in one comparison operation. Thus, to recognize an image of size $N \times N$ using a set of reasonable thresholds has $O(\log N)$ times comparison according to theorem 3.1. On the SIMD-SM-R Machine with $N' \times N'$ processors, a comparison to an $N \times N$ image needs $\left\lceil \frac{N \times N}{N' \times N'} \right\rceil = c$ comparison operations. In that case, recognizing an image would have $O(c \log N)$ comparison operations using our approach. A linear search through K templates requires $O(c\ K)$ comparison operations. If templates are organized into a balanced binary tree, recognition of an image on the SIMD-SM-R machine has $O(c \log K)$ comparison operations. On the other hand, the template tree method has difficulty in deciding the thresholds for different branching points and balancing the tree.

Furthermore, since the time complexity of the comparison operation depends on the region size which it operates on, in

our algorithm one comparison operation on a high pyramid level has much less time than on a low pyramid level. The linear search method or the tree search method has the same complexity for every comparison operation which is the function of image size. Our approach needs only $\log N$ thresholds. When compared to other methods published in the literature [6, 13, 19], the thresholds used in our approach are not necessarily very accurate. In addition, the template tree needs $(2K-1)N^2 2^t$ bits of memory but our database needs only $\frac{4}{3}K N^2 2^t$ bits. Moreover, our database is very easy to update.

This method can be useful for that class of applications that need large template sets. One such problem is that of Chinese characters recognition. Matching the five to seven thousand Chinese characters in daily use requires not only a large template set but also a great amount of time, depending on the algorithm used. By using the method described in this paper, the problem can be solved quickly and precisely with little overhead in database updates as the matching progresses.

References

[1] Rodney Allen Brook, *Model-Based Computer Vision*, UMI Research Press, 1984.

[2] M. J. B. Duff (ed.), *Intermediate-Level Image Processing*, Academic Press. 1986.

[3] J. A. Feldman and D. H. Ballard, Computing With Connections, in *Human and Machine Vision* (Jacob Beck, et al. eds.). Academic Press, 1983.

[4] Karen A. Frenkel, Evaluating Two Massively Parallel Machines, *Commun. ACM* Vol. 29, No. 8, pp. 752-759, 1986.

[5] T. J. Fountain, K. N. Matthews and M. J. B. Duff, The CLIP7A Image Processor, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 3, pp. 310-319, 1988.

[6] E. S. Gelsema and L. N. Kanal (eds.), *Pattern Recognition in Practice II*, Elsevier Science Pub. Co., 1986.

[7] E. L. Hall, *Computers Image Processing and Recognition*, Academic Press, 1979.

[8] W. Daniel Hillis, *The Connection Machine*, The MIT Press, 1985.

[9] W. D. Hillis and G. L. Steele Jr., Data Parallel Algorithms, *Commun. ACM* Vol. 29, No. 12, pp. 1170-1183, 1986.

[10] Berthold Klaus Paul Horn, *Robot Vision*, The MIT Press, 1986.

[11] L. H. Jamieson, D. Gannon and R. J. Douglass, *The Characteristics of Parallel Algorithms*, MIT Press,1987.

[12] S. Levialdi (ed.), *Integrated Technology for Parallel Image Processing*, Academic Press, 1985.

[13] Xiaobo Li, M. Ferdousi, M. Chen and T. T. Nguyen, Image Matching With Multiple Templates, *Proceedings of IEEE 1986 Computer Vision and Pattern Recognition*, pp. 610-613, 1986.

[14] Abha Moitra and S. Sitharama Iyengar, Parallel Algorithms for some Computational Problems, *Advances in Computers*, Vol. 26, 1987.

[15] H. K. Ramapriyan, A Multilevel Approach to Sequential Detection of Pictorial Features, *IEEE Transactions on Computers*, Vol. c-25, No. 1, pp. 66-78, 1976.

[16] Thomas A. Rice and Leah H. Jamieson, Parallel Processing for Computer Vision, in *Integrated Technology for Parallel Image Processing* (S. Levialdi, ed.). Academic Press, 1985.

[17] A. Rosenfeld (ed.), *Multiresolution Image Processing and Analysis*, Springer-Verlag, 1984.

[18] Kathryn T. Spoehr and S. W. Lehmkuhle, *Visual Information Processing*, W. H. Freeman and Company, 1982.

[19] S. Tanimoto and A. Klinger (eds.), *Structured Computer Vision*, Academic Press, 1980.

[20] Shimon Ullman and W. Richards (eds.), *Image Understanding*, Ablex Pub. Corp., 1984.

[21] Wu Wang, S. S. Iyengar and L. M. Patnaik, Memory-Based Reasoning Approach for Pattern Recognition of Binary Images, *Pattern Recognition*, Vol. 22, No. 5.

[22] Satoshi Watanabe, *Pattern Recognition: Human and Mechanical*, Wiley, 1985.