S19-62
78
185139

Parallel Processing for Digital Picture Comparison

H.D. Cheng and L.T. Kou
University of California, Davis
Davis, CA 95616

CC 907079

ABSTRACT

In picture processing an important problem is to identify two digital pictures of the same scene taken under different lighting conditions. This kind of problem can be found in remote sensing, satellite signal processing and the related areas. The identification can be done by transforming the gray levels so that the gray level histograms of the two pictures are closely matched. The transformation problem can be solved by using the "packing" method. In this paper, we propose a VLSI architecture consisting of m x n processing elements with extensive parallel and pipelining computation capabilities to speed up the transformation with the time complexity O(max(m,n)), where m and n are the numbers of the gray levels of the input picture and the reference picture respectively. If using uniprocessor and a dynamic programming algorithm, the time complexity will be $O(m^3 \times n)$. The algorithm partition problem, as an important issue in VLSI design, is discussed. Verification of the proposed architecture is also given.

Index terms: Digital picture comparison, packing algorithm, very large scale integration (VLSI), algorithm partition, VLSI architecture verification.

I. INTRODUCTION

The technique of dynamic programming has wide applications in computer science [6,7] for solving mathematical problems arising from multistage decision processes. Based on the dynamic programming path-finding algorithm, the technique of dynamic programming is both mathematically sound and computationally efficient. The recent advent of very-large-scale integration (VLSI) technology has triggered the thought of implementing some algorithms directly in hardware with extensive parallel and pipelining computation capabilities. The use of VLSI architectures to implement dynamic programming procedures has been investigated for several applications. Guibas et al. [8] describes a VLSI architecture for a class of dynamic programming problems characterized by optimal parenthesization. Chu and Fu [9] describe VLSI architectures for recognition of context-free and finite-state languages. Chiang and Fu [10] describe a VLSI implementation of Early's algorithm for parsing general context-free languages. Cheng and Fu [11] describe algorithm partition and parallel recognition of general context-free languages using fixed-size VLSI architecture. Liu and Fu [12] describe a VLSI implementation for string-distance computation. Clarke and Dyer [15] describe four VLSI architectures for line and curve detection. Cheng and Fu [13,14] propose VLSI architectures for pattern-matching and hand-written symbol recognition. In this paper, we propose a VLSI architecture for identifying digital pictures if they are taken from the same scene under different lighting conditions. This is a very important problem related to remote sensing, satellite signal processing and other areas. As an important issue in VLSI design, the algorithm partition problem is discussed. The backtracking procedure is also discussed in much detail, and the formal verification of the proposed architecture is given. An example is used to illustrate the work of the proposed VLSI architecture.

II. PRELIMINARY

The image matching technique has been used extensively for many applications such as curvature sequences detection [2], template matching and pattern matching [1], character recognition, target recognition, aerial navigation and stereo mapping, picture matching, earth resource analysis, missile guidance, intelligence gathering systems, and robotics [2,3].

There are many situations in which we want to match or register two pictures with one another, or match some given pattern with a picture [2]. For example:

(a) Given two or more pictures of the same scene taken by different sensors, we want to determine the characteristics of each pixel with respect to all of the sensors and then we can classify the pixels.

(b) Given two pictures of scenes taken from different times, we want to determine the points at which they differ and then can analyze the changes that have taken place.

(c) Given two pictures of a scene taken from different positions, we want to identify corresponding points in the pictures and then determine their distances from the camera to obtain three-dimensional information from the scene.

(d) We want to find places in a picture where it matches a given pattern.

In this paper we want to discuss another very important aspect in picture processing which is to identify two digital pictures of the scene taken under different lighting conditions. These kinds of problems arise from many areas such as remote sensing, satellite signal processing, and etc. The identification can be done by transforming the gray levels so that the gray level histograms of the two pictures are closely matched.

Mathematically, a picture is defined by a function of two variables $F(x,y)$, where $F(x,y)$ is the brightness, or K-tuples of brightness values in several spectral bands, [2,4,5] and x and Y are the coordinates in the image plane. In the black and white case, the values are called gray levels. These values are real, non-negative, and bounded. The pictures are represented as matrices with integer elements which are the pixels. A gray level histogram of an image is a function that gives the frequency of occurrence of each gray level in the image. Where the gray levels are quantized from 0 to n, the value of the histogram at a particular gray level p, denoted $H(p)$, is the number of fraction of pixels in the image with that gray level [5]. When pictures of the same scene are obtained under different lighting conditions, different histograms are gained. For identifying these pictures, we can transform their gray level scales so that their histograms would closely match each other.

Assume that H_1 and H_2 are histograms of two pictures obtained from the same scene with m and n gray levels, respectively. An algorithm is proposed to "reshape" H_1 (i.e. rescale its gray levels) so that it has the minimal deviation from H_2. The mathematical problem is defined by:

$$Z = \min_{(X_0,\ldots,X_n)} \sum_{j=1}^{n} |H_2(j) - \sum_{i=P}^{Q} H_1(i)|$$

where $P = X_{j-1}$ and $Q = X_j - 1$ subject to

$$1 = X_0 < X_1 < \ldots < X_n = m+1 \tag{1}$$

X_i = integer, for $i = 1,\ldots,n$.

It will transform the gray levels $X_{j-1},\ldots,X_j-1$ in one picture into gray level j in the other picture, for suitably chosen X_{j-1} and X_j, $j = 1,\ldots,n$.

This problem can be interpreted as a packing problem: to pack m objects of sizes $\{H_1(1),\ldots,H_1(m)\}$ into n boxes of spaces $\{H_2(1),\ldots,H_2(n)\}$ in such a way that

(i) if the ith object has been placed in the jth box, the (i+1)th object is not allowed to be packed into the kth box for any $K < j$, and

(ii) the accumulated error due to space over-packed of leftover is minimized.

Such a problem can be solved by using dynamic programming techniques. Let $S_j(i)$ be the minimal accumulated error caused by transforming the gray levels $1,\ldots,i$ into the gray levels $1,\ldots,j$. The recursive formula is given by

$$S_j(i) = \min_{0<u<i} \{S_{j-1}(u) + |H_2(j) - \sum_{v=u+1}^{i} H_1(v)|\} \tag{2}$$

for $i=1,\ldots,m$ and $j=1,\ldots,n$

where the initial conditions are $S_0(0) = 0$, $S_0(i) = \sum_{v=1}^{i} H_1(v)$ for all $i = 1,\ldots,m$ and $S_j(0) = \sum_{u=1}^{j} H_2(u)$ for all $j = 1,\ldots,n$. If $i>j$, then $\sum_{k=i}^{j} H_1(k) = 0$. The minimal accumulated error, $S_n(m)$, can be computed.

The straight forward execution of this procedure would obtain the optimal solutions for all (i,j) pairs with time complexity $O(m^3 x n)$ by using uniprocessor. In this paper, we want to propose a m x n VLSI array to speed up the computation. The time complexity for the proposed architecture is $O(\max(m,n))$.

III. VLSI DIGITAL PICTURE COMPARATOR

3.1 The algorithm and its VLSI implementation

We will propose a VLSI architecture based on the space-time domain expansion approach [14,15], which has a very natural and regular configuration and can be implemented easily by applying today's VLSI technology. Another important issue in VLSI design - algorithm partition problem is also solved by using the proposed VLSI architecture. The proposed VLSI architecture can speed up the digital picture comparison procedure greatly by using extensive parallel and pipelining techniques. Before discussing the VLSI architecture in detail, we propose the following algorithm.

Let H_1 and H_2 be the histograms of two pictures taken at the same scene with m and n gray levels, respectively.

Algorithm 1: The algorithm for digital picture comparison:

```
begin
    $S_0(0)$ := 0;
    for i := 1 to m do
        begin
          $S_0(i)$ := 0;
          for k := 1 to i do
          $S_0(i)$ := $S_0(i)+H_1(k)$
        end;

    for j := 1 to n do
        begin
          $S_j(0)$ := 0;
          for k := 1 to j do
          $S_j(0)$ := $S_j(0)+H_2(k)$
        end;

    for i := 1 to m do
      for u := 0 to i-1 do
        begin
          v := u + 1;
          T(v) := 0;
          for k := v to i do
          T(v) := $H_1(k)$ + T(v)
        end;

    for i := 1 to m do
      for j := 1 to n do
        begin
          $T'$ := $S_{j-1}(i)$ + $H_2(j)$;
          I := i (index channel value);
          T := 0;
          for u := 1 to i-1 do
            begin
              v := u +1;
              s := 0;
              for k := 1 to v do
                begin
                  s := s + $H_1(k)$;
                  T := $|H_2(j) - s|$;
                T := $s_{j-1}(u)$ + T;
                If $T' < T$ then
                t' := T and output v to the index channel by letting I := v
              end
          end;
      end
```

Append index-pair (I,j-1) to index-pair (i,j), when the identification signal arrives, and form ((I,J-1), (i,j)).

end.

We can build a VLSI array with m x n processing elements. Each processing element has a subtracter which will produce the absolute value of the two inputs difference, a comparator which will compare two input values and output the smaller value with the corresponding index to the next processing element below it. The functions performed by the (i,j)th processing element are as follows:

Input: $H_2(j)$, outputs of (i-1,j)th processing element, $\sum_{v=1}^{i} H_1(v)$, index-pair, $S_{j-1}(i-1)$ and $S_{j-1}(i)$.

Output: $S_j(i)$ and index-pair to the right element when the identification signal arrives, and the intermediate results to the processing element below.

Operations: Each processing element has a local connection to the processing element beneath it which will accept the intermediate results including the accumulated errors and the index-pairs, and has a local connection to the right processing element which will receive $S_j(i)$ and index pair (i,j) when the identification signal arrives. Each processing element can perform accumulation, $|a-b|$, and comparison operations, and requires one time unit. The adder uses the combinational circuit, which will not require the time unit, or its delay is much smaller than a time unit. The data will move among the processing elements, one processing element per time unit.

1) Input data of H_1 arrives at the (i,j)th PE and performs the accumulation of each for one time unit. $|H_2(j)-\sum_{v}^{i} H_1(v)|$ needs one time unit.

2) $S_{j-1}(i-1)$ arrives at the (i,j)th processing element and it performs $S_{j-1}(i-1) + |H_2(j)-H_1(i)|$ operation which requires one time unit. The result is delayed one time unit.

3) $0<u<i-2\{S_{j-1}(u)+|H_2(j)-\sum_{v=u+1}^{i} H_1(v)|\}$ arrives at the (i,j)th processing element from the (i-1,j)th processing element and compares with the result of step 2) which requires one time unit. At the same time, the identification signal arrives and the result will compare with $S_{j-1}(i)+H_2(j)$ which will require one time unit. Then $S_j(i)$ and the index-pair will be sent to the (i,j+1)th processing element.

Algorithm 2 VLSI implementation of Algorithm 1

Input: Gray levels of the input picture $-H_1(i)$, and of the reference picture $-H_2(j)$ (for $1<i<m$, $1<j<n$); indices, index pairs; initial conditions: $S_0(0)$, $S_0(i)$, and $S_j(0)$ (for $1<i<m$, $1<j<n$); and identification signals.

Output: The accumulated error $S_j(i)$ and corresponding index pairs

Move the gray levels $H_2(j)$ of the reference picture, the identification signals, and the index j from the top to the bottom one processing element per time unit. Move the gray levels $H_1(i)$ of the input picture and index i from the left to the right of the VLSI array one processing element per time unit. The identification signals will be sent at the fifth time unit and will move down one processing element per two time units and move to the right one processing element per time unit. When the identification signal arrives, it will open the connection channel to the comparator which connects the right processing element, and $S_j(i)$ will be sent to the processing element (i,j+1). To obtain the 'packing' sequence, we have to perform a backtracking procedure which can be done in several ways as follows.

1) Output the accumulate error matrix S and/or the index-pairs to the host machine which will perform the backtracking procedure.

2) Attach another VLSI module and use the tag of the index pair as the search key to perform the backtracking procedure.

3) Expand the 'append' operation to the one which appends the index into the index list of its ancestor.

An index list is formed by appending an index or an index list. We can use index (m,n) as the tag to find the 'packing' sequence. This will change the backtracking procedure into forward and speed up the computation, but it requires a large output channel capacity, especially for the processing elements located at the upper-right corner. The upper bound of the channel capacity for the (i,j) processing element will be (i+j+1).

4) Add an index register to each processing element which consists of two parts, the first part for the first index and the second part for the second index. The second part of the index pair register will compare with the tag. If they are matched, the second part is output into the output channel and also output into the first part as the tag to its top and left side neighbors. The tag will move up until it match with another index pair. The procedure will be continued. We need one index register for each processing element. At the (2i+j+3)th time unit, send a backtracking signal which moves along the channels connecting to the left neighbor and the one on the top of it, each processing element per time unit. The index (m,n) is used for the tag of the (m,n)th processing element. It needs at most (m+n) time units to complete this procedure.

From the above discussion, we can conclude that the proposed architecture can compare two digital pictures by transforming the gray levels. In many applications, only the summation error is required. In such cases, we can simplify the structure of the processing element and the entire VLSI architecture further. If there are P digital pictures which are compared with the reference picture, or an input digital picture compared with P reference digital pictures, we can make a P-time expansion. The time complexity will be O(max(Pxm,n)). If using uniprocessor, the time complexity will be O(Pxm^{3}xn). For indicating the most matched digital picture, we number the digital pictures and add a register consisting of two parts. One part is for the summation error, the other is for the index of the numbered digital pictures. We also add a counter which is initially set to zero and starts at (2m+n+3)rd time unit.

The operation of the register is as follows:

```
begin
  error.register:=∞;
    if error.register > error array
      then begin
            error.register:=error.array
            index.register:=counter
          end end
```

The final result of index.register indicates the index of the most matched digital picture. If we use a three dimensional array (Pxmxn processing elements), the time complexity will be reduced to O(max(P,m,n)). The detail will be omitted here.

IV. VERIFICATION OF THE PROPOSED ALGORITHM

To verify algorithm 2, we need the following lemmas and theorem.

Lemma 1: The identification signal arrives at the (i,j)th processing element at the (2i+j+2)th time unit.

Proof: The identification signal is sent at the fifth time unit and it needs 2(i-1) time units to reach the ith row, it then needs j time units to arrive at the (i,j)th processing element. Totally, 5+2i-2+j-1=2i+j+2 time units.

Lemma 2: $\sum_{v}^{i} H_1(v)$ will be computed at the (v,j)th processing element at the (i+v+j-2)th time unit, for all 1<v<i,1<i<m and 1<j<n.

Proof: First consider the j=1 case. From the data arrangement in Fig. 3, the first input of the vth row will arrive at the boundary of the array at 2(v-1)th time unit, then (i-v+1) time units are needed to compute $\sum_{v}^{i} H_1(v)$. Totally, 2(v-1)+(i-v+1)=i+v-1 time units are needed. Since the computation of the (v,k)th processing element will start one time unit earlier than one of the (v,k+1)th processing elements, the time units needed for the (v,j)th processing element will be i+v+j-2 to produce the summation.

Theorem: After receiving the inputs, $S_j(i)$ will be produced at the (2i+j+3)th time unit, for all 1<i<m and 1<j<n.

Proof: We prove the theorem by induction on i and j.

Basis: First we consider i=j=1 case. Since $S_0(0)$ and $S_0(1)$ are fixed values which exists already, $H_1(1)$ the inputs into the processing element (1,1) and it performs the accumulation which requires one time unit.

Then $|H_2(1)-H_1(1)|$ is performed by spending another time unit. It will be added to $S_0(0)$ and delayed one time unit. At the forth time unit, the result will be compared with $S_1(0)$. When the identification signal arrives at the fifth time unit, the result of the comparator will compare with $S_0(1) + H_2(1)$ and output $S_1(1)$ which needs one more time unit, $6=2x1+1+3=2xi+j+3$.

Induction Step: Our induction hypothesis is that all $(p,q)^{th}$ processing elements can produce the outputs and the index-pairs at the $(2xP+q+3)^{th}$ time unit, for all $1<p<i$ and $1<q<j$.

Now consider the $(i+1,j)^{th}$ processing element. According to lemma 2 and the hypothesis, $\sum_{v}^{i+1} H_1(v)$ will be computed at the $(v,j)^{th}$ processing element, $(i+1+v+j-2)^{th}$ time unit, for all $1<v<i$, $1<i<m$ and $1<j<n$ and the comparators are connected in a pipeline version, so $M = \min_{0<u<i} (S_{j-1}(u)+|H_2(j)-\sum_{v=u+1}^{i+1} H_1(v)|)$ will be output from the $(i,j-1)^{th}$ processing element, $(2i+j-1+2+3)^{th}$ time unit. Also $S_{j-1}(i+1)$ will be input at the $2x(i+1)+j-1+3^{th}$ time unit. At the same time $N=S_{j-1}(i+1)+H_2(j)$ will be computed. According to lemma 1, the identification signal arrives at the $(i+1,j)^{th}$ processing element at the $2(i+1)+j+2^{th}$ time unit. Then M and N will be compared, the minimum $(M,N)=S_j(i+1)$ will be sent to the $(i+1,j)^{th}$ processing element at the $(2i+j+5)^{th}$ time unit. Since $S_{j+1}(i+1)$ will be one time unit later than $S_j(i+1)$, $S_{j+1}(i+1)$ will be obtained at the $(2i+j+6)=2(i+1)+(j+1)+3$ the time unit. Therefore the proof is completed.

Corollary 1: The accumulated error and the index pairs can be obtained at the $(2m+n+3)^{th}$ time unit.

Proof: Follow the theorem and let $i=m$ and $j=n$.

V. ALGORITHM PARTITION

We could use a one-dimensional array or a two-dimensional array with size different to the problem size by performing time expansions following the partition rule.

A. Using the One-Dimensional Array

First we assume that the size of the array is m. We can consider it as an m-space expansion along the x_1 direction. The input channels will form the queues. The register will hold the initial value and the result from the CR output which will input into the register by the control signal. The control signal is sent at the $(m+1)^{th}$ time unit and moves down per two time units and one processing element. The input will repeat n times. The time complexity will be $O(m \times n)$.

B. Using the Two-Dimensional Array with the Dimensions Kxl

If $k=m$ and $\ell=n$, it is the case which has already been discussed. We now consider the other cases. According to the partition rule we have to make an $\lceil m/k \rceil$ - time expansion and an $\lceil n/l \rceil$ - time expansion. There are also queues for feedback of the data. The lengths of the queues will be varying with the values of m and n to make the right data meet at the right processing element at the right time. This will cause much difficulty to the control system and the queue structures. Hence, we either use a sufficiently large size VLSI architecture or use a one-dimensional array to solve the partition problem.

VI. CONCLUDING REMARKS

We have proposed an VLSI architecture for digital picture comparison. The time complexity will be $O(\max(m,n))$ by using a two-dimensional $m \times n$ array, where m is the gray level of the input digital picture and n is the ray level of the reference digital picture. With a uniprocessor, the comparison process will have the time complexity $O(m^3xn)$ if using the straightforward computation approach. If there are p reference pictures using the proposed architecture, the comparison process will be solved in time $O(\max(mxp,n))$; and using a uniprocessor the time complexity will be $O(mxpxn)$. If using a three-dimensional array, this problem can be solved in time $O(\max(m,n,p))$. One important issue, the algorithm partition of the VLSI design is discussed and formal verification of the proposed VLSI architecture is given. The proposed architecture will be useful for remote sensing, satellite signal processing, and other related areas. It can also be useful for other 'packing' related tasks and for real-time digital picture processing.

LIST OF REFERENCES

1. K. S. Fu, Syntactic Pattern Recognition with Application, Prentice-Hall, Inc., 1982.

2. A. Rosenfeld and A. C. Kak, Digital Picture Processing, Vol. 2, Academic Press, New York.

3. R. J. Offen, VLSI Image Processing, Collins Professional and Technical Books, William Collins Sons & Co. Ltd., 1985.

4. T. Pavlidis, Algorithms for Graphics and Image Processing, Computer Science Press, 1982.

5. D. H. Ballard and C. M. Brown, Computer Vision, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

6. R. E. Bellman, Dynamic Programming, Princeton, NJ: Princeton Univ. Press, 1975.

7. K. Q. Brown, "Dynamic programming in computer science," CMU Tech. Rep., February 1979.

8. L. J. Guibas, H. T. Kung and C. D. Thompson, "Direct VLSI implementation of combinational algorithms," Caltech. Conf. on VLSI, January 1979.

9. K. H. Chu and K. S. Fu, "VLSI architectures for high speed recognition of general context-free languages and finite-state languages," Proc. 9th Annual Int'l. Symp. Comput. Arch., Austin, TX, April 1982.

10. Y. T. Chiang and K. S. Fu, "Parallel parsing algorithms and VLSI implementations for syntactic pattern recognition," IEEE trans. on Pattern Anal. Machine Intell., May 1984.

11. H. D. Cheng and K. S. Fu, "Algorithm partition and parallel recognition of general context-free languages using fixed-size VLSI architecture," Pattern Recognition, Vol. 19, No. 5, 1986.

12. H. H. Liu and K. S. Fu, "VLSI arrays for minimum-distance classifications," in VLSI for Pattern Recognition and Image Processing, edited by K. S. Fu, Springer-Verlag, Berlin, Heidelberg, 1984.

13. H. D. Cheng and K. S. Fu, "VLSI architectures for string matching and pattern matching," to appear in Pattern Recognition, Vol. 20, No. 1, 1987.

14. H. D. Cheng and K. S. Fu, "VLSI architecture for dynamic time-warp recognition of hand-written symbols," IEEE Trans. Acoust., Speech, Signal Processing, Vol. 34, No. 3, June 1986.

15. M. J. Clarke and C. R. Dyer, "Curve Detection in VLSI," VLSI for Pattern Recognition and Image Processing, edited by K. S. Fu, Springer-Verlag, Berlin, Heidelberg, 1984.

16. W-M Chow and L. T. Kou. "Matching Two Digital Pictures," Proc. International Computer Symposium, December 1978.