VLSI Algorithm for Euclidean Distance Transform

Tomio Hirata

Graduate School of Information Science, Nagoya University

hirata@nuee.nagoya-u.ac.jp

Distance maps of binary images contain, for each pixel, the distance between that pixel and the pixel of value 0 closest to it. They have important uses in computer vision, pattern recognition, morphology and robotics. Many algorithms have been proposed for computing distance maps for a variety of distances such as the Manhattan(L_1) and chessboard(L_∞) distances. Though the Euclidean distance(L_2) is the most natural one for many applications, Euclidean distance maps have not been used because of their high computation cost. The time complexity of Euclidean distance transform algorithms was $O(n^3)$ for $n \times n$ input images, while transform algorithms for other distances are of $O(n^2)$ time. Recently the time complexity for Euclidean distance transform was improved to $O(n^2)$, but today's applications, such as robot vision, require real-time processing speed.

In this presentation, we propose a hardware algorithm using a systolic array that performs the Euclidean distance transform. It is designed so that hardware resources, such as multipliers and comparators, are reduced, and also it achieves very high processing speed. Our algorithm performs the Euclidean distance transform for an $n \times n$ binary input image in $3n-1$ clocks, and the size of the required hardware resources is $O(n^2)$.

Let $B = \{b_{i,j}\}$ ($b_{i,j} \in \{0,1\}$) be an $n \times n$ binary image. We denote by (i,j) a pixel in the ith row and the jth column of B. A Euclidean transformation of B is denoted as $D = \{d_{i,j}\}$ and defined as follows.

$$d_{i,j} = \min_{0 \le p,q \le n-1} \{\sqrt{(i-p)^2 + (j-q)^2} | b_{p,q} = 0\}$$

The proposed algorithm consists of two transformations, T_1 and T_2. T_1 executes the Euclid distance transform in each column of the input image. We denote by $G = \{g_{i,j}\}$ the output of T_1.

$$g_{i,j} = \min_{0 \le p \le n-1} \{|i-p| | b_{p,q} = 0\}$$

T_2 then receives the output $G = \{g_{i,j}\}$ of T_1 and calculates Euclidean distance value for each pixel.

$$d_{i,j} = \min_{0 \le q \le n-1} \{\sqrt{(j-q)^2 + g_{i,q}^2}\}$$

Figure 1 shows the block diagram of the systolic array. A systolic array proposed by H. T. Kung in 1982 is a VLSI architechture in which simple computation units are located regularly and data exchange is performed between adjacent units simultaneously at each clock. The input B is fed to the array row by row and computation proceeds in pipeline fashion. We have implemented the algorithm in a VLSI chip and verified the performance.

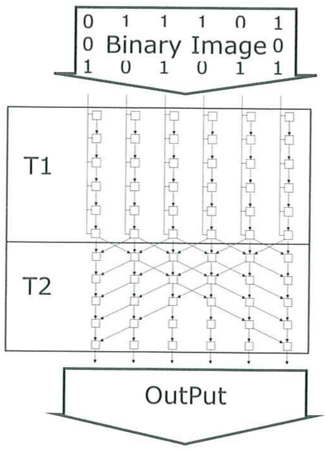


Figure 1. Systolic array

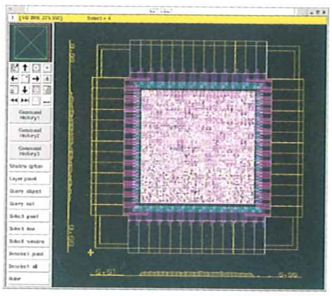


Figure 2. Design of VLSI chip

VLSI Algorithm for Euclidean Distance Transform

1. Euclidean Distance Transform

Input: Binary image
Output: Distance to the nearest 0-pixel for each pixel

1	1	1	0	1
1	1	1	1	1
1	1	1	0	1
1	1	0	0	1
0	0	1	1	1

⇒

3	2	1	0	1
$2\sqrt{2}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
2	$\sqrt{2}$	1	0	1
1	1	0	0	1
0	0	1	1	$\sqrt{2}$

2. Background

- Applications
  Image processing, Morphological filter,
  Pattern Recognition, Robot vision, etc.

- In Robot vision, especially.
  Real-time speed is required → VLSI implementation

⇒ *We proposed the algorithm suitable for VLSI.*

3. Basic algorithm

T1: Distance calculation in each column
T2: Calculation of the Euclidean distance

1	1	1	0	1
1	1	1	1	1
1	1	1	0	1
1	1	0	0	1
0	0	1	1	1

T1 ⇒

4	4	3	0	∞
3	3	2	1	∞
2	2	1	0	∞
1	1	0	0	∞
0	0	1	1	∞

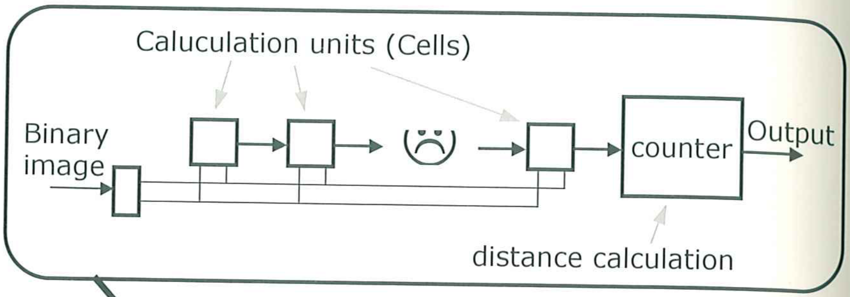
T2 ⇒

3	2	1	0	1
$2\sqrt{2}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
2	$\sqrt{2}$	1	0	1
1	1	0	0	1
0	0	1	1	$\sqrt{2}$

VLSI Algorithm for Euclidean Distance Transform

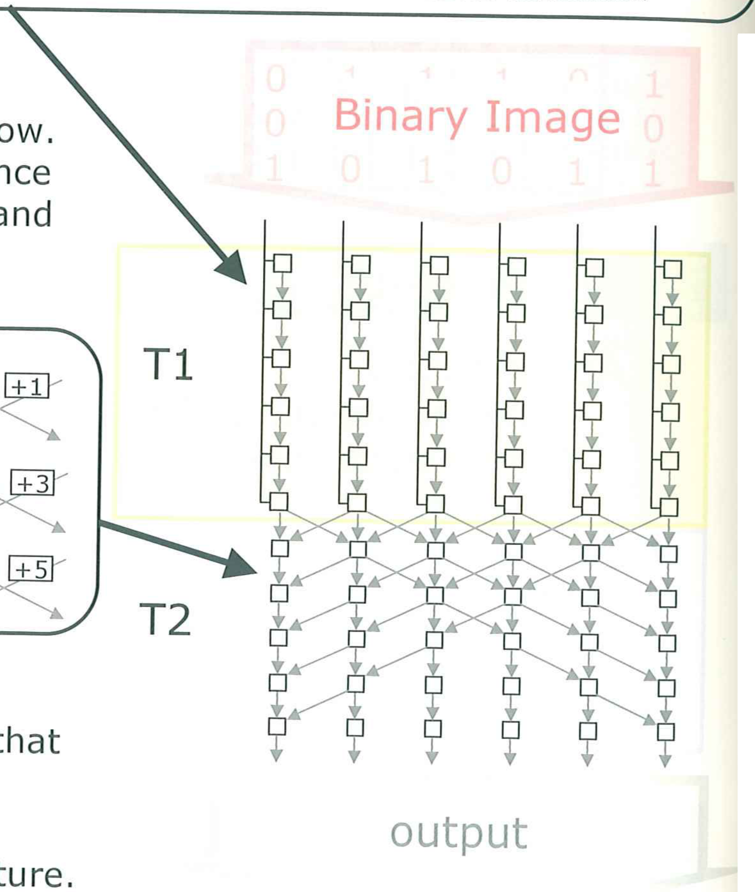
4. VLSI algorithm

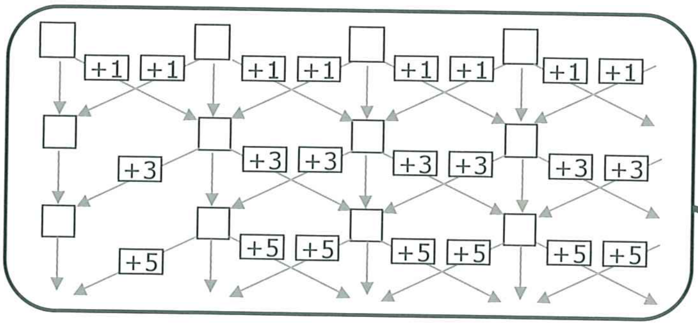
Our algorithm uses a systolic array.



T1: Binary image is fed into the first array row by row. It computes column distance.

T2: Output from T1 is fed into the second array again row by row. It calculates the Euclidean distance by propagating data diagonally and downward.





Merits

Data Stream is one direction so that calculation proceeds in pipeline fashion.

Processor array has simple structure.

We need not use multipliers.

For $N \times N$ binary image, this algorithm runs in $3N-1$ time units on $2N^2-N$ cells.

5. Parformance

Verification by simulation

Process : ROHM CMOS 0.35um(VDEC)
Area : $2.45 mm^2$
Speed : Testing