

Computing Hough Transforms on Hypercube Multicomputers*

SANJAY RANKA

School of Computer and Information Science, 4-116 CST, Syracuse University, Syracuse, NY 13244-4100

SARTAJ SAHNI

Department of Computer Science, University of Minnesota, Minneapolis, MN 55455

(Received January 1989; final version accepted January 1990.)

Abstract. Efficient algorithms to compute the Hough transform on MIMD and SIMD hypercube multicomputers are developed. Our algorithms can compute p angles of the Hough transform of an $N \times N$ image, $p \leq N$, in $O(p + \log N)$ time on both MIMD and SIMD hypercubes. These algorithms require $O(N^2)$ processors. We also consider the computation of the Hough transform on MIMD hypercubes with a fixed number of processors. Experimental results on an NCUBE/7 hypercube are presented.

Key words: Hough transform, MIMD and SIMD hypercube multicomputers, complexity.

1. Introduction

The Hough transform is used to transform edges to another space, called the Hough space, so that the desired group of edges forms a cluster in the transformed space. Let $I[0 \dots N - 1, 0 \dots N - 1]$ be an $N \times N$ image such that $I[x, y] = 1$ iff the image point $[x, y]$ is a possible edge point. $I[x, y] = 0$ otherwise. The p angle Hough transform of I to detect straight lines in an image is the array H such that

$$H[i, j] = |\{(x, y) | i = \lfloor x \cos \theta_j + y \sin \theta_j \rfloor, \theta_j = \frac{\pi}{p}(j + 1) \text{ and } I[x, y] = 1\}|. \quad (1)$$

j takes on the integer values $0, 1, \dots, p - 1$. These correspond to the p angles $\theta_j = \frac{\pi}{p}(j + 1)$, $0 \leq j < p$. Hence $0 < \theta_j \leq \pi$. For θ_j in this range and x and y in the range $0 \dots N - 1$, $\lfloor x \cos \theta_j + y \sin \theta_j \rfloor$ is in the range $-\sqrt{2}N \dots \sqrt{2}N$. Hence H is at most a $2\sqrt{2}N \times p$ matrix.

The general equation of a straight line can be given by the parametric equation

$$x \cos \theta + y \sin \theta = r, \quad (2)$$

where θ is the angle that the normal, to the line given by Equation (2), makes with the x axis and r is the length of the normal. Any edge point (x_i, y_i) on this line satisfies the equation

$$x_i \cos \theta + y_i \sin \theta = r. \quad (3)$$

*This research was supported by the National Science Foundation under grants DCR84-20935 and 86-17374. All correspondence should be mailed to Sanjay Ranka.

The above equation represents a sinusoidal curve in the (r, θ) space. Any point on this curve corresponds to a line passing through (x_i, y_i) . Thus, the curves corresponding to all the points on a line in the (x, y) space must intersect at the same point in the (r, θ) space. Each edge point contributes 1 to the (r, θ) cells given by Equation (3) and the cells corresponding to the local maxima give the desired lines.

The generalized Hough transform can be used to recognize curves of arbitrary shapes [Ballard 1981]. It has been used successfully in a wide variety of domains. These include detection of tumors in chest films, recognition of objects in aerial images, and detection of human hemoglobin fingerprints [Ballard and Brown 1982].

The serial algorithm to compute H has complexity $O(N^2p)$. Parallel algorithms to compute H have been developed by several researchers. Rosenfeld et al. [1988], Cypher et al. [1987], and Guerra and Hambruch [1987] consider mesh connected multicomputers; Fishburn and Highnam [1987] consider scan line array processors; Ibrahim et al. [1986] consider SIMD tree machines; and Chandran and Davis [1987] consider the use of the Butterfly and NCUBE multicomputers to compute the Hough transform.

In this paper we develop algorithms to compute the above Hough transform on hypercube multicomputers. First, in Section 2, we describe our model for fine-grained MIMD and SIMD hypercubes and how to perform certain fundamental data movement operations on a hypercube. These are used in our subsequent development of hypercube algorithms for the Hough transform. In Section 3 we describe our Hough transform algorithm for the MIMD hypercube. The case of an SIMD hypercube is considered in Section 4. Section 5 considers the computation of the Hough transform on a medium-grained MIMD hypercube. Experimental results on an NCUBE/7 hypercube are also presented in this section.

2. Preliminaries

2.1. Hypercube Multicomputer

Block diagrams of an SIMD and MIMD hypercube multicomputer are given in Figures 1a and 1b, respectively. The important features of an SIMD hypercube and the programming notation we use follow:

1. There are $P = 2^p$ processing elements (PEs) connected via a hypercube interconnection network (to be described later). Each PE has a unique index in the range $[0, 2^p - 1]$. We shall use brackets $[]$ to index an array and parentheses $()$ to index PEs. Thus $A[i]$ refers to the i -th element of array A and $A(i)$ refers to the A register of PE i . Also, $A[j](i)$ refers to the j -th element of array A in PE i . The local memory in each PE holds data only (that is, no executable instructions). Hence, PEs need to be able to perform only the basic arithmetic operations (that is, no instruction fetch or decode is needed).
2. There is a separate program memory and control unit. The control unit performs instruction sequencing, fetching, and decoding. In addition, instructions and masks are broadcast by the control unit to the PEs for execution. An *instruction mask* is a boolean function used to select certain PEs to execute an instruction. For example, in the instruction

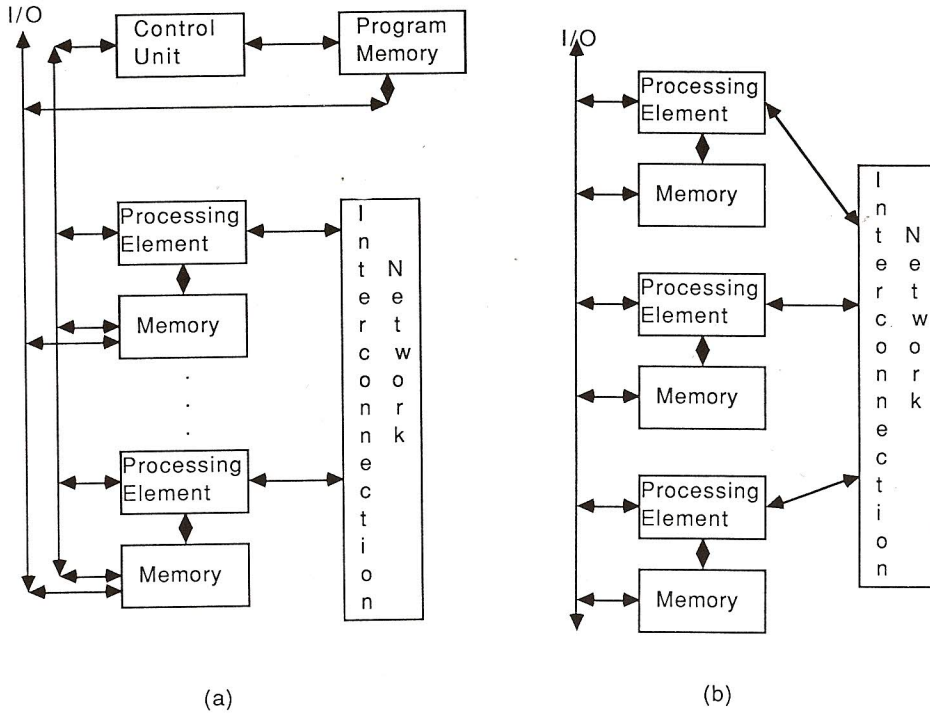


Figure 1. Hypercube multicomputers: a) SIMD hypercube, b) MIMD hypercube.

$$A(i) := A(i) + 1, (i_0 = 1).$$

$(i_0 = 1)$ is a mask that selects only those PEs whose index has bit 0 equal to 1; that is, odd indexed PEs increment their A registers by 1. Sometimes we shall omit the PE indexing of registers. The above statement is therefore equivalent to the statement

$$A := A + 1, (i_0 = 1).$$

3. The topology of a 16-node hypercube interconnection network is shown in Figure 2. A p -dimensional hypercube network connects 2^p PEs. Let $i_{p-1}i_{p-2} \dots i_0$ be the binary representation of the PE index i . Let \bar{i}_k be the complement of bit i_k . A hypercube network directly connects pairs of processors whose indices differ in exactly one bit; that is, processor $i_{p-1}i_{p-2} \dots i_0$ is connected to processors $i_{p-1} \dots \bar{i}_k \dots i_0$, $0 \leq k \leq p - 1$. We use the notation $i^{(b)}$ to represent the number that differs from i in exactly bit b .
4. Interprocessor assignments are denoted using the symbol \leftarrow , while intraprocessor assignments are denoted using the symbol $:=$. Thus the assignment statement

$$B(i^{(2)}) \leftarrow B(i), (i_2 = 0)$$

