

Constant Time BSR Solutions to L_1 Metric and Digital Geometry Problems

ROBERT A. MELTER

Department of Mathematics, Long Island University, Southampton, NY 11968, USA

IVAN STOJIMENOVIĆ

Computer Science Department, University of Ottawa, Ottawa, Ont K1N 9B4 Canada

Abstract. In this paper we solve several geometric and image problems using the BSR (broadcasting with selective reduction) model of parallel computation. All of the solutions presented are constant time algorithms. The computational geometry problems are based on city block distance metrics: all nearest neighbors and furthest pairs of m points in a plane are computed on a two criteria BSR with m processors, the all nearest foreign neighbors and the all furthest foreign pairs of m points in the plane problems are solved on three criteria BSR with m processors while the area and perimeter of m isooriented rectangles are found on a one criterion BSR with m^2 processors. The problems on an $n \times n$ binary image which are solved here all use BSR with n^2 processors and include: histogramming (one criterion), distance transform (one criterion), medial axis transform (three criteria) and discrete Voronoi diagram of labeled images (two criteria).

Keywords: BSR model, digital geometry, parallel computation

1 Introduction

Models of computation typically provide a simple abstraction of a computing device and a set of primitive operations which are assumed to be executed in unit time. This allows great simplification in designing algorithms and in their analysis. However, it does not mean that these primitive operations really take unit time when implemented. One most evident example is Knuth's famous assumption in the serial random access machine (RAM) model that memory access takes constant time, which is impossible to implement despite very close practical realizations. In the parallel random access machine (PRAM) models, memory references are again assumed to take unit time, although in the practical implementations this time is non constant, and in best cases is $O(\log n)$ where n is the number of processors. This may lead to algorithms that are efficient on the model but inefficient in practice. The scan model (cf. [9]) has for primitives some operations that practically have performances similar to memory reference, and are treated as unit time operations. It has lead to much simpler algorithms and/or improving the time complexities of solutions to many problems. The new set of primitives includes the scan operation, also known as the parallel prefix or partial sum operation. The other primitives in the scan

model are elementwise arithmetic and logical primitive operations and arbitrary permutations. It is shown in [9] that the scan model of parallel computation is particularly suitable for solving problems on binary images.

Broadcasting with selective reduction (BSR) is a model of parallel computation introduced by Akl, Guenther and Lindon. The model is more powerful than any of the PRAM models, yet one and two criteria BSR do not require more hardware to implement than PRAM. Besides being powerful, the model allows one to write very short solutions to a variety of problems. For many problems BSR offers constant time solutions which were not possible on any PRAM. BSR is also more powerful than the scan model of parallel computation.

The BSR model is described in section 2. The first solution presented here is for the histogramming problem (section 3). It is chosen because of the very simple solution, containing merely one broadcasting instruction on a one criterion BSR. It serves as an illustration for the model and introduces other more complex solutions.

The distance $d(p_1, p_2)$ between two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ can be defined in various ways, although a family of L_p metrics seem to be most commonly used. It is defined by $L_p((x_1, y_1),$

$(x_2, y_2)) = (|x_1 - x_2|^p + |y_1 - y_2|^p)^{1/p}$. Special cases of the L_p metrics which are most often used are Euclidean distance L_2 and the Manhattan metric (or city block distance) L_1 . In this paper we adopt the L_1 metric, and denote the city block distance $d(p_1, p_2)$ between two points P_1 and P_2 as follows:

$$d(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|.$$

We consider the computational geometry problems that are based on city block distance. In these problems the input is a set of m points or rectangles in the plane, and a BSR with m processors is used to achieve constant time solutions. Section 4 presents solutions to all nearest and furthest neighbor problems for planar point sets. We consider three different output requests (for each point), and for each give a solution on a two criteria BSR:

- (i) find the distance to the nearest/furthest point,
- (ii) find the index of the nearest/furthest point,
- (iii) find the color (or label or any other type of information) in the nearest/furthest point.

Section 5 deals with the distance transform problem for $n \times n$ images. A one criterion BSR with n^2 processors is used to answer, for each pixel, the following queries in constant time (it is similar to the previous case):

- (i) find the distance of each (white) pixel to the nearest black pixel,
- (ii) find the row and column indices of the nearest black pixel,
- (iii) find the color (usually called the label, especially in multiple component problems) or other information for the nearest black pixel.

Section 6 gives a constant time solution on a three criteria BSR with n^2 processors solution to the medial axis transform problem.

Area and perimeter are studied in section 7. The problem is trivial for binary images in the usual representation, but sometimes images are given via their medial axis transforms (MATs). A constant time algorithm using a one criterion BSR with m^2 processors is given for the corresponding problem of computing the area and perimeter of m isooriented rectangles.

Finally, section 8 gives a solution to the problem of constructing discrete Voronoi diagrams for labeled images.

2 BSR

A mathematical notation is employed to describe the multiple criteria BROADCAST instruction. Let us define the following symbols:

n : number of processors

p_i : processor $i, l \leq i \leq n$

d_i : datum broadcast by $p_i, l \leq i \leq n$

σ_h : selection operation, $l \leq h \leq k$, taken from the following set $\{<, \leq, =, \geq, >, \neq\}$

$t(i, h)$: tag broadcast by p_i for criterion $\sigma_h, l \leq h \leq k, l \leq i \leq n$

$l(j, h)$: limit value broadcast by p_j for criterion $\sigma_h, l \leq h \leq k, l \leq j \leq n$

\mathcal{R} : binary associative reduction operation, where $\mathcal{R} \in \{\sum, \prod, \wedge, \vee, \oplus, \min, \max\}$ denoting sum, product, AND, OR, XOR, maximum, and, minimum respectively

x_j : result of a multiple criteria BROADCAST instruction, $l \leq j \leq n$.

The k criteria BSR BROADCAST instruction is denoted by:

$$x_j \leftarrow \mathcal{R}_{l \leq i \leq n} d_i \mid \bigwedge_{l \leq h \leq k} t(i, h) \sigma_h l(j, h),$$

for $l \leq j \leq n$.

When the ranges of the variables are understood, this can be abbreviated as:

$$x_j \leftarrow \mathcal{R} d_i \mid \bigwedge t(i, h) \sigma_h l(j, h).$$

The above notation can be interpreted as follows. If $t(i, h) \sigma_h l(j, h)$ is satisfied for each $h, l \leq h \leq k$, then d_i is "accepted" by location x_j . The set of all data accepted by x_j is reduced to a single value by means of the binary associative operation \mathcal{R} , and stored in x_j . If no data are accepted by a given memory location x_j then x_j receives the value of the neutral element for operation \mathcal{R} . If only one datum is accepted, then x_j is assigned the value of that datum. The operation is performed for all $j, l \leq j \leq n$, in parallel.

3 Histogramming

Suppose that an $n \times n$ image is given by its gray level values I_{ij} , $0 \leq I_{ij} < g$, $1 \leq i, j \leq n$. Thus each pixel has one of g possible integer values, denoting its gray level. The histogram of an image is a list of all gray values paired by their counts, i.e. the number of pixels having such gray value.

In [16], Tanimoto describes an algorithm for histogram computation on a pyramid which involves one $O(\log n)$ steps "pass" from bottom of the pyramid to the top for each gray level value to be computed. [6] presented an algorithm for the hypercube which computes the complete histogram in $\log(n)$ time independent of the range of gray level values.

Assume that the image is stored one pixel per processor on a one criteria BSR with n^2 processors. We will present a constant time algorithm for computing the histogram on the model.

For simplicity, assume that pixel indices are in the range from 1 to n^2 (in rowmajor or other order). The number of pixels s_j having the same gray value as the pixel j is obtained by a single call to the broadcasting instruction.

$$s_j \leftarrow \sum 1 \mid g_i = g_j.$$

If a list of all different gray level values with their counts is wished then pixels can be sorted by their counts s_j , with repetitions eliminated.

4 All Nearest Neighbors and Furthest Pairs

Given a set of m points $p_i = (x_i, y_i)$, the all nearest neighbors problem is to find the nearest point for each point. We describe a constant time solution to the problem on a BSR with m processors.

The definition of BSR does not allow the use of absolute value functions. Without the function, the distance of p_i from p_j depends on whether p_i is above or below, to the left or right of p_j . Therefore there are four different definitions of the distance depending of the mutual positions of two points. A BSR operation with two criteria can be applied for each of these positions, and the minimum nn_j of obtained values is then selected. The distance of a point p_i which is above and to the right of p_j is $d(p_j, p_i) = x_i + y_i - x_j - y_j$. Thus the closest point in the quadrant is at distance $ar_j - s_j$ where $s_j = x_j + y_j$ and ar_j is the minimal of values $s_i = x_i + y_i$ for points in the quadrant. The closest

pixels in other quadrants are found in similar way. The algorithm can be written as follows.

$$\begin{aligned} s_j &\leftarrow x_j + y_j \\ t_j &\leftarrow x_j - y_j \\ ar_j &\leftarrow \min s_i \mid x_i \geq x_j \wedge y_i > y_j \\ bl_j &\leftarrow \max s_i \mid x_i \leq x_j \wedge y_i < y_j \\ al_j &\leftarrow \max t_i \mid x_i < x_j \wedge y_i \geq y_j \\ br_j &\leftarrow \min t_i \mid x_i > x_j \wedge y_i \leq y_j \\ nn_j &\leftarrow \min(s_j - bl_j, t_j - l_j, -t_j + br_j, -s_j + ar_j). \end{aligned}$$

The algorithm finds the distance of the nearest neighbor point (problem (i) in the introduction). The index in_j of the nearest neighbor point for point p_j can be found by using additional two criteria BSR instructions (if there are several points at the same nearest neighbor distance, only one of them is selected). The points are assumed to be sorted by their x -coordinates in the increasing order (this can be done in constant time on a one criteria BSR with n processors [2]).

$$\begin{aligned} iar_j &\leftarrow \max i \mid y_i > y_j \wedge s_i = ar_j \\ ibl_j &\leftarrow \min i \mid y_i < y_j \wedge s_i = bl_j \\ ial_j &\leftarrow \min i \mid y_i \geq y_j \wedge t_i = al_j \\ ibr_j &\leftarrow \max i \mid y_i \leq y_j \wedge t_i = br_j \\ \text{if } nn_j = -s_j + ar_j &\text{ then } in_j \leftarrow iar_j \\ \text{if } nn_j = s_j - bl_j &\text{ then } in_j \leftarrow ibl_j \\ \text{if } nn_j = t_j - al_j &\text{ then } in_j \leftarrow ial_j \\ \text{if } nn_j = -t_j + br_j &\text{ then } in_j \leftarrow ibr_j \end{aligned}$$

In the above solution we note that one criterion, comparison along x -coordinates, is omitted. This is possible because if a nearest neighbor is in a given quadrant then the choice of max or min guaranties that a point from the quadrant is selected (not from the another one corresponding to the failure of comparison along x -coordinates).

Finally, if the information co_i contained in the nearest neighbor point i for point j is desired (problem (iii) in the introduction) then the following simple broadcasting instruction can be used to obtain it (obviously min can be replaced with max or sum, since there is exactly one successful candidate).

$$inf_j \leftarrow \min co_i \mid i = in_j.$$

The all furthest pairs problem is to find the furthest point for each point in a given set. It can be obtained in a similar way as the nearest neighbor. More precisely, it suffices to interchange all references to min and max.

Closest pair of points is defined by two points having the minimal possible distance. It can be obtained by

minimizing all nearest neighbor distances. Similarly the diameter is the distance between two furthest points, and is derived by maximizing all furthest pair distances.

In [10] the following modifications of the closest point and all nearest neighbor problems is considered. Let S be a set of m colored points. In the closest foreign pair problem one has to find a closest foreign pair, i.e. a bichromatic pair of points which are closest. In the all nearest foreign neighbors problem one has to find for each point in the configuration S a nearest neighbor with different color. [10] presented plane sweep algorithms for solving the closest foreign pair and all nearest foreign neighbor problems in L_1 and L_∞ metrics in optimal $O(n \log n)$ sequential time.

Using the city block metric, we may solve both problems on a BSR with m processors in constant time. The solution is obtained by a modification to the closest pair and all nearest neighbor algorithms (which are special cases of the problems with all points colored in different colors) by adding one more criterion to all BSR instructions so that points with the same color are ignored. For example, the instruction

$$ar_j \leftarrow \min s_i \mid x_i \geq x_j \wedge y_i > y_j$$

becomes

$$ar_j \leftarrow \min s_i \mid x_i \geq x_j \wedge y_i > y_j \wedge c_i \neq c_j$$

where c_i is the color of pixel i .

Similarly one can define the all furthest foreign pair and the foreign diameter problems and appropriate solutions.

5 Distance Transform

A binary image is usually represented by a $n \times n$ matrix I such that $I_{ij} = 0$ if the pixel in i -th row and j -th column is white and $I_{ij} = 1$ if the pixel is black, $1 \leq i, j \leq n$. The distance transform problem is to find the nearest black pixel for each white pixel in the image. The distance transform is introduced by Rosenfeld and Pfaltz [14]. Important applications of distance transforms are expanding and shrinking objects (by thresholding the transform), constructing shortest paths and computing shape factors, reconstructing objects from parts of the boundary, and skeletization (cf. [15]).

Various metrics were used in literature to find the distance transforms, mostly belonging to the family of L_k metrics. The distance transform for the L_1 metric can be found sequentially in $O(n^2)$ time. In [15] it is

shown that distance transform cannot be computed on a pyramid in polylogarithmic time for any L_k metric. For the mesh of trees model, [15] gave an $O(\log n)$ time algorithm for computing the distance transform under the L_1 metric.

Since the image has n^2 pixels, we assume a BSR model with n^2 processors, one per pixel. The problem can be solved by modifying the algorithm for finding all nearest neighbors (see previous section), where white pixels set the values of s_j and t_j at infinity (positive or negative, chosen such that they are ignored in the computation). However, such a solution requires a two criteria BSR model, which needs more hardware than a one criterion BSR.

We now give an alternate solution to the distance transform problem, which requires a one criterion BSR. The algorithm for finding the distances to the nearest black pixels is based on the following simple property of binary images. Suppose that b' is the closest black pixel to a white pixel a' . Let u be the pixel that lies in the same row as a' and same column as b' . Then obviously b' is the closest black pixel to u among black pixels from the same column to which u belongs. This property follows from the monotonicity of distance functions with respect to absolute values of differences in x - or y -coordinates of two pixels.

This property suggests a simple algorithm for finding the distance transform. Let processors be numbered by their row and column indices (corresponding to the pixel indices), from 1 to n . The algorithm consists of a columnwise scan followed by a rowwise one.

In the columnwise scan, for each pixel (j, k) (black or white) find the distance r_{jk} to the closest black pixel from the same column as (j, k) , if exists. Each processor (i, k) carrying a white pixel writes 0 ($n+1$, respectively) to a designated location b_{jk} . Processor (i, k) carrying a black pixel writes their row index $b_{ik} = i$. Each pixel (j, k) finds the closest black pixel above (below, respectively) itself (in the same column), if it exists, chooses the closer one, and stores the distance to it in location r_{jk} of corresponding processor. The closest black pixel (i, k) to pixel (j, k) is at distance $|i - j|$. For each column k ($1 \leq k \leq n$) a one criterion BSR model with n processors is applied to compute r_{jk} as follows. Indices j and i are used as in the BSR definition, while k is considered a constant for given column.

$$\begin{aligned} b_{ik} &\leftarrow i \text{ if } I_{ik} = 1 \text{ (i.e. } (i, k) \text{ is a black pixel),} \\ &\quad \text{otherwise } b_{ik} \leftarrow n + 1 \\ r'_{jk} &\leftarrow \min b_{ik} \mid i \geq j \\ \text{if } I_{ik} = 0 \text{ then } b_{ik} &\leftarrow 0 \end{aligned}$$

$$\begin{aligned}
r''_{jk} &\leftarrow \max b_{ik} \mid i \leq j \\
r_{jk} &\leftarrow 3n \\
\text{if } r'_{jk} < n + 1 &\text{ then } r_{jk} \leftarrow r'_{jk-j} \\
\text{if } r''_{jk} > 0 &\text{ and } j - r''_{jk} < r_{jk} \text{ then } r_{jk} \leftarrow j - r''_{jk}
\end{aligned}$$

The index in_{jk} of the selected black pixel which is at distance r_{jk} from pixel (j, k) is determined as either $j + r'_{jk}$ or $j - r''_{jk}$, depending on whether the selected pixel is above or below (j, k) , respectively. The information contained in the selected black pixel (like a label) can be broadcasted as shown in the previous section for the all nearest neighbor problem.

In the rowwise scan, each pixel (k, j) finds the minimum value $|i - j| + r_{ki}$ which is the distance to its closest black pixel. For each row k ($1 \leq k \leq n$) a one criterion BSR with n processors is used.

$$\begin{aligned}
c'_{ki} &\leftarrow r_{ki} + i \text{ for each pixel } (k, i) \\
s'_{kj} &\leftarrow \min c'_{ki} \mid i \geq j \\
c''_{ki} &\leftarrow -i + r_{ki} \text{ for each pixel } (k, i) \\
s''_{kj} &\leftarrow \min c''_{ki} \mid i \leq j \\
s_{kj} &\leftarrow \min(s'_{ki} - j, j + s''_{ki})
\end{aligned}$$

In order to find the index in_{kj} of the pixel that "supplied" indirectly the nearest black pixel, two candidates are found by the following broadcasting operations:

$$\begin{aligned}
rin_{kj} &\leftarrow \max i \mid c'_{ki} = s'_{kj} \\
lin_{kj} &\leftarrow \min i \mid c''_{ki} = s''_{kj}
\end{aligned}$$

and one of them, corresponding to the choice for s_{kj} , is taken to become in_{kj} . The information from the nearest black pixel (its indices or component label) is then available via processor (k, j) (we have described how to read information from a particular processor), since it can be collected in the previous columnwise step.

The algorithm for distance transform can be adapted in straightforward way to calculate the furthest black pixel for each pixel in the image.

In this solution we used columnwise and rowwise scans and considered each column or row to be a separate BSR model with n processors. However, we assumed merely that we have a unique BSR with n^2 processors, such that a broadcasting operation selects data from all processors in order to extract a datum for a particular processor. There are two ways to resolve the problem. One is to add additional criteria, requesting data from a particular row or column only (where row and/or column indices are given for purpose of selection); this solution increases the number of needed criteria which is not a desirable property. A better solution

is to sort all data in either column major or row major order (depending on the type of scan) and modify the algorithm accordingly. For example, in the columnwise scan for the distance transform algorithm, pixels are assigned indices from 1 to n^2 and each pixel finds the nearest black pixel with a greater and a smaller index; if the obtained indices exceed the appropriate bounds for a given column the corresponding result is ignored and information that no pixel is found is recorded. For the rowwise scan we can introduce one more "trick": since all meaningful distances after the first step are between 1 and n , it is possible to increase all r_{ki} values by kn ($1 \leq k \leq n$). In this way there is a considerable gap between neighboring rows which does not allow one to select data from a pixel in a different row. Similar modifications can be made for other algorithms on binary images presented in this paper.

6 Medial Axis Transform

The medial axis transform (MAT) is an image representation scheme proposed by Blum [5]. The essential idea in the MAT is to find a minimal set of upright squares whose union corresponds exactly to the regions in I that have value 1 (i.e. cover black pixels). In passing, we note that the problem of finding the minimum number of rectangles that cover the 1's in a binary image is n - p hard [11].

Sequential $O(n^2)$ algorithms for this problem are given in [12, 7]. In [7], an $O(n)$ time n processor CREW PRAM algorithm for the MAT is obtained while in [12] two PRAM algorithms were developed, both using $O(n^2)$ processors and having a run time $O(\log n)$. Also, [12] gave a $O(\log^2 n)$ time complexity solution on a hypercube with n^2 processors.

We follow the sequential solution in [7, 12] to develop a constant time algorithm using n^2 processors for a three criteria BSR. Assume that the top left corner of the top left pixel is indexed $(1, 1)$, and the bottom right corner of bottom right pixel is indexed $(n + 1, n + 1)$. Let $M[i, j]$ be the height of the largest square with top left corner $[i, j]$, all of whose image values are 1. Obviously $M[n + 1, j] = M[j, n + 1] = 0, 1 \leq j \leq n$. In fact, to simplify we assume that additional white pixels $(n + 1, j)$ and $(j, n + 1)$ are added, $1 \leq j \leq n$. The top left corners of the squares in the MAT are marked by value $T[i, j] = \text{true}$ while other pixels have $T[i, j] = \text{false}$; and $T[i, j] = \text{true}$ iff $\max(M[i, j - 1], M[i - 1, j], M[i - 1, j - 1]) \leq M[i, j]$. This assumes that $M[0, j] = M[j, 0] = 0, 1 \leq j \leq n$.

The sequential and parallel algorithms in [7, 12] have more definitions and computations which we do not need to introduce for solving the MAT problem on a BSR model. Thus we have two steps in our algorithm:

- Step 1. Compute $M[i, j]$.
- Step 2. Compute $T[i, j]$.

Here we use a different approach to compute $M[i, j]$ on a three criteria BSR model. We use the L_∞ metric, defined as $L_\infty((x_1, y_1), (x_2, y_2)) = \max(|x_1 - x_2|, |y_1 - y_2|)$. Let us observe that all pixels which are to the right and below pixel (i, j) and are at L_∞ distance $\leq M[i, j]$ must be black (otherwise the square of size $M[i, j]$ with top left corner (i, j) contains a white pixel, contrary to the definition of $M[i, j]$). Our algorithm on BSR is based on finding the white pixel to the right and below that is closest to the pixel (i, j) , and its distance to (i, j) will determine $M[i, j]$.

Suppose that a white pixel (k, l) is the closest pixel to the pixel (i, j) among pixels to the right of and below pixel (i, j) . Then $k \geq i$ (to the right), $l \geq j$ (below) and the distance is $M[i, j] = \max(k - i, l - j)$. The latter distance is $k - i$ if $k - i \geq l - j$ i.e. $k - l \geq i - j$, and $l - j$ otherwise (see the figure). On three criteria BSR $M[i, j]$ and $T[i, j]$ can be found as follows (the ranges of k and l are between 1 and $n + 1$):

$$\begin{aligned} &\text{if } I_{kl} = 0 \text{ then } \{k' \leftarrow k; l' \leftarrow l; kl \leftarrow k - l\} \text{ else} \\ &\{k' \leftarrow n + 1; l' \leftarrow n + 1; kl \leftarrow 3n\} \\ &ij \leftarrow i - j \\ &c'_{ij} \leftarrow \min k' \mid k \geq i \wedge l \geq j \wedge kl \geq ij \end{aligned}$$

$$\begin{aligned} c''_{ij} &\leftarrow \min l' \mid k \geq i \wedge l \geq j \wedge kl \leq ij \\ M[i, j] &\leftarrow \min(c'_{ij} - i, c''_{ij} - j) - 1 \end{aligned}$$

$$\begin{aligned} T[i, j] &\leftarrow \text{true if and only if } \max(M[i, j - 1], \\ &M[i - 1, j], M[i - 1, j - 1]) \leq M[i, j]. \end{aligned}$$

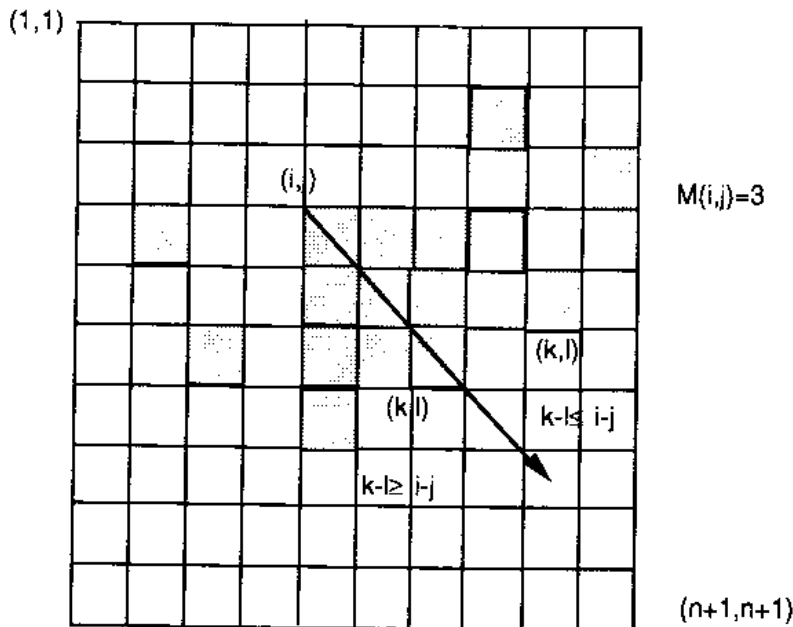
7 Area and Perimeter of Isooriented Rectangles

The area and perimeter of a binary image can easily be calculated (using no criteria BSR, which more or less correspond to the scan model of computation [9]) from the matrix I_{ij} in the following way:

$$\begin{aligned} \text{area} &\leftarrow \sum I_{ij} \\ b_{ij} &\leftarrow 1 \text{ iff } I_{ij} = 1 \text{ and at least one neighbor of} \\ &\text{pixel } (i, j) \text{ is a white pixel; otherwise } b_{ij} \leftarrow 0 \\ \text{perimeter} &\leftarrow \sum b_{ij} \end{aligned}$$

Note that all pixels receive the area and perimeter information by the above instructions.

It is more difficult to compute the area and perimeter from MAT representation of the image. Known algorithms compute, in fact, the area and perimeter of the union of m upright (or isooriented) rectangles. $O(m \log m)$ serial algorithms were given in [18]. In [19] $O(m)$ time algorithms for both CREW PRAM and mesh-connected computers to compute the area, perimeter and contour of a set of n upright rectangles were developed. Their PRAM algorithm uses $O(m)$ processors, whereas their mesh-connected computer algorithms use $O(m^2)$ processors. [8] obtained



$O(\log m)$ time CREW PRAM algorithms for the area and perimeter of m rectangles, using m processors. [13] develop an $O(\sqrt{m})$ parallel algorithm that computes the area and perimeter of m rectangles on an m processor mesh-connected computer. This algorithm is easily modified to run on an m processor hypercube in $O(\log^2 m)$ time. It can be further modified to compute the perimeter in the same time and processor bounds. In [12] $O(\log m)$ time algorithms to compute the area and perimeter on an m^2 processor hypercube are presented. [21] describes an $O(\log m \log \log m)$ time with $O(m / \log \log m)$ processors CREW PRAM algorithm.

Here we present the first constant time algorithms for computing the area and perimeter of m isooriented rectangles. We use a one criterion BSR with m^2 processors.

We use the concept of slabs as in [13, 8, 12]. The plane is subdivided into approximately $2m$ horizontal slabs by extending horizontal sides of each rectangle. By sorting the y -coordinates of horizontal sides the slabs receive their borders. The computation proceeds separately for each slab, and an n processor one criterion BSR is assigned to each slab. In each slab, it is first checked whether each of rectangles intersects the slab or not. Each rectangle either has empty intersection (which can be detected by having both horizontal sides of the rectangle either above or below the slab borders) or the intersection is a rectangle with vertical sides determined by x -coordinates of the rectangle and filling the slab vertically completely. Therefore, each slab contains up to m intervals, and the area in the slab is determined by the product of the width of the slab and the measure of the union of intervals determined by the rectangles. The later can be computed in constant time on an m processor one criterion BSR [4]. Thus the area of m upright rectangles can be determined in constant time with m^2 processors on a one criterion BSR (by summing areas in each slab).

The perimeter can be similarly computed, with some modifications. The endpoints in each interval in a slab (recall intervals are intersections of the slab with rectangles) and for each endpoint it is determined whether it is the border point of the union of the intervals (this is a part of the measure of the union algorithm in [4]). The number of such endpoints is multiplied by the width of the slab to find the contribution of the slab toward the perimeter of the union of rectangles.

Often, the area and perimeter is computed for each object in an image. The algorithms presented above consider only the case that the image contains only one object. However, it is not difficult to design BSR

algorithms to compute each object's area and perimeter. It suffices to add one more condition to each of above algorithms, which will separate computation for each object. The details are omitted.

8 Discrete Voronoi Diagram for Labeled Images

Image processing often requires separating an image into components, which are defined as components of a graph that is composed of edges joining neighboring black pixels. An image where each (black) pixel is assigned a label such that two black pixels have the same label if and only if they are in the same component is referred to as a labeled image.

Labeling an image is a fundamental problem in image processing, and it is studied extensively on various models of parallel computation. The best PRAM solution is in $O(\log n)$ time with n^2 processors, and is obtained basically by the method of recursive doubling. The method easily applies on a one criterion BSR. It is an open problem to find a constant time algorithm for labeling an image on a BSR with any number of criteria and processors.

Often in the literature the image is assumed to be already labeled, and then various questions about the image are asked. We consider here one such question as an example. We will give a construction of a discrete Voronoi diagram in constant time on a two criteria BSR with n^2 processors.

Assuming that the black pixels of a picture F are divided into p connected components C_1, C_2, \dots, C_p , a set of pixels which are strictly closer to C_r than to any other component is called the tile T_r of the connected component C_r . More precisely, T_r contains black pixels from C_r and exactly those white pixels x for which $d(x, C_r) < d(x, C_i)$ for $i \neq r$, where $d(x, C_i)$ denotes the minimal distance from pixel x to a pixel from C_i . The set of all tiles is called the discrete Dirichlet tessellation $T(F)$. The discrete Voronoi diagram is defined as the set of all pixels which are located on borders of tiles (i.e. have neighbors that do not belong to the same tile) and all pixels which belong to no tile (or pixels which have more than one closest component to them). The definitions are taken from [17] where some other neighboring relations among figures are studied.

The first task is to compute, for each white pixel, the closest and second closest components. The closest component is found by applying the discrete transform operation, which finds the closest black pixel for each white pixel (if a pixel is black then it is at distance

0 to the closest component, which is the component containing the pixel).

The second closest component for each white pixel can be found by modifying the distance transform operation. In the columnwise scan, each pixel (i, k) disregards the pixels which have the same component label as the pixel selected in the columnwise scan of distance transform algorithm, by adding one more criteria on BSR broadcasting operations. Therefore each pixel has selected two closest black pixels from the same column, belonging to two different components.

The rowwise scan has more modifications. The corresponding steps in the distance transform operation are performed twice: once with the closest and once with the second closest black pixels from the columnwise scan. Each time one more criterion is added, asking for pixel candidates to be from different components than the component selected in the regular distance transform algorithm. The repetition of steps is needed because some candidates come from the already selected component while their appropriate alternatives may come from second closest component. The better candidate from the two obtained candidates is selected.

The discrete Voronoi diagram is now easily constructed from its definition. Each white pixel may compare distances to its two closest components. If it is the same, or it has a neighbor which belongs to a different tile then the pixel belongs to the discrete Voronoi diagram.

9 Open Problems

There are a number of other problems in digital geometry or in computational geometry using city block distance metrics which can be investigated. For example, eccentricity and moment of inertia [7]. It appears that the Voronoi diagram of a set of points in plane based on the L_1 metric can be constructed in constant time using a three criteria BSR with n processors using the algorithm [20] (the algorithm is omitted to avoid tedious repetitions from [20]).

It is open problem to check whether the area of m rectangles can be computed in constant time on a BSR with less than m^2 processors.

Acknowledgments

The authors acknowledge the financial support by NATO Collaborative Research Grant CRG 900840 and

Natural Sciences and Engineering Research Council of Canada operating grant OGPIN007, and a Faculty Research Grant from Long Island University.

References

1. S.G. Akl, L. Fava Lindon, and G.R. Guenther, "Broadcasting with selective reduction on an optimal PRAM circuit," *Technique et Science Informatiques*, Vol. 4, pp. 261–268, 1991.
2. S.G. Akl and G.R. Guenther, "Broadcasting with selective reduction," in *Proc. of 11th IFIP Congress*, San Francisco, 1989, pp. 515–520.
3. S.G. Akl and G.R. Guenther, "Application of BSR to the maximal sum subsegment problem," *Int. J. High Speed Computing*, Vol. 3, No. 2, pp. 107–119, 1991.
4. S.G. Akl and I. Stojmenovic, "Multiple criteria BSR: An implementation and applications to computational geometry problems," Technical Report No. 93-351, Department of Computing and Information, Queen's University, Kingston, Ontario, Canada, 1993.
5. H. Blum, "Models for perception of speech and visual form," Cambridge, MA: MIT Press, pp. 362–380, 1967.
6. T. Bestul and L.S. Davis, "On computing complete histograms of images in $\log(n)$ steps using hypercubes," *IEEE T-PAMI*, Vol. 11, No. 2, pp. 212–213, 1989.
7. S. Chandran, S. Kim, and D. Mount, "Parallel computational geometry of rectangles," *Algorithmica*, Vol. 7, pp. 25–49, 1992.
8. S. Chandran and D. Mount, "Shared memory algorithms and the medial axis transform," in *Proc. IEEE Workshop CAPAMI*, 1987, pp. 44–50.
9. B. Djokic and I. Stojmenovic, "Constant time digital geometry algorithms on the scan model of parallel computation," *Proc. SPIE Vol. 1832 Vision Geometry*, 1992, pp. 162–170.
10. T. Graf and K. Hinrichs, "Algorithms for proximity problems on colored point sets," in *Proceedings of Fifth Canadian Conference on Computational Geometry*, Waterloo, 1993, pp. 420–425.
11. M.R. Garey and D.S. Johnson, "Computers and Intractability," W.H. Freeman, San Francisco, 1979.
12. J. Jenq and S. Sahni, "Serial and parallel algorithms for the medial axis transform," *IEEE T-PAMI*, Vol. 14, No. 12, pp. 1218–1224, 1992.
13. M. Lu and P. Varman, "Optimal algorithms for rectangle problems on a mesh-connected computer," *J. Parallel Distributed Computing*, Vol. 5, pp. 154–171, 1988.
14. A. Rosenfeld and J.L. Pfalz, "Sequential operations in digital picture processing," *Journal of the ACM*, Vol. 13, No. 4, pp. 471–494, Oct. 1966.
15. O. Schwarzkopf, "Parallel computation of discrete transforms," *Algorithmica* Vol. 6, pp. 685–697, 1991.
16. S.L. Tanimoto, "Sorting, histogramming, and other statistical operations on a pyramid machine," in *Multiresolution Image Processing and Analysis*, (A. Rosenfeld, ed.) New York: Springer-Verlag, 1982, pp. 136–145.
17. J. Toriwaki and S. Yokoi, "Voronoi and related neighbors on digitized two-dimensional space with applications to texture analysis," in *Computational Morphology* (G.T. Toussaint, ed.), North-Holland, 1988, pp. 207–228.
18. A.Y. Wu, S.K. Bhaskar, and A. Rosenfeld, "Computation of geometric properties from the medial axis transform in $O(N \log N)$

- time," *Comput. Vision Graphics Image Processing*, Vol. 34, pp. 76-92, 1986.
19. A.Y. Wu, S.K. Bhaskar, and A. Rosenfeld, "Parallel computation of geometric properties from the medial axis transform," *Comput. Vision Graphics Image Processing*, Vol. 41, pp. 323-332, 1988.
 20. Y.C. Wee and S. Chaiken, "An optimal parallel L_1 metric Voronoi diagram algorithm," in *Proc. 2nd Canadian Conference on Computational Geometry*, 1990, pp. 60-65.
 21. M. Zubair, "An optimal speedup algorithm for the measure problem," *Parallel Computing*, Vol. 13, pp. 61-71, 1990.



Robert A. Meltzer received the Ph.D. in Mathematics from the University of Missouri at Columbia in 1962. He originally worked

on Boolean Algebras and Ring Theory and later turned to Graph Theory. His more recent work has been in Digital Geometry and other aspects of Vision Geometry. Since 1971 he has been teaching at the Southampton Campus of Long Island University where he is now a Professor of Mathematics.



Ivan Stojmenović received B.S. and M.S. degrees in 1979 and 1983 from the University of Novi Sad and Ph.D. Math. degree in 1985 from the University of Zagreb. In 1980 he joined the Institute of Mathematics, University of Novi Sad. He was visiting Professor at the Electrotechnical Laboratory, Tsukuba, Japan (winter 1985/86), Washington State University, Pullman, USA (Fall 1987), and University of Miami, Miami, USA (Winter 1988). In Fall 1988, he joined the faculty of the Computer Science Department at the University of Ottawa (Ottawa, Canada), where currently he holds the position of an Associate Professor. His research interests are computational geometry, parallel computing, combinational algorithms, and multiple-valued logic. He is currently an editor of two journals: *Parallel Processing Letters*, and *Parallel Algorithms and Applications*.