Fast Algorithms for Block-Based Medial Axis Transform on the LARPBS

Yuh-Rau Wang
Department of Computer Science and Information Engineering
St. John’s University
Taipei, Taiwan
yrwang@mail.sjsmit.edu.tw

Abstract - Many parallel algorithms have been proposed for computing the two-dimensional block-based medial axis transform (2D_BB_MAT). Unfortunately, almost all of them are dedicated for solving 2D_BB_MAT problem and are very difficult to be extended for solving the 3D_BB_MAT problem. In this paper, an O(1) time algorithm for solving the 2D_BB_MAT of a binary image \( \mathcal{P} \) of size \( N \times N \) on an LARPBS is first developed. The running time of this algorithm has a smaller constant factor compared with those of all the other previous proposed O(1) time algorithms. Then this algorithm is extended for computing the 3D_BB_MAT of a binary image \( \mathcal{V} \) of size \( N \times N \times N \) in O(1) time on an LARPBS. The running time of this algorithm has a small constant factor also. To the best of our knowledge, this is the first parallel algorithm proposed for solving the 3D_BB_MAT problem known.

Keywords: Medial axis transform, image processing, parallel algorithm, LARPBS.

1 Introduction

The Medial axis transform (MAT) is an image representation scheme proposed by Blum [1]. It is a powerful shape descriptor that has good properties for data reduction and allows a full reconstruction of the original shape. There are two types of MATs [2]. One is the distance-based MAT (DB_MAT), which is defined as a recovering of the object by maximal digital balls (disks) included in the object in a 3D (2D) space. A maximal digital ball (disk) is a digital ball (disk) that contains in no other digital ball (disk). If every voxel (pixel) of the medial axis of an object is labeled with the radius of its corresponding maximal sphere, then the object can be exactly rebuilt using the information of its axis [3]. Very few parallel algorithms have been developed for the 3D_DB_MAT [4], [5]. In [4], the task of computing the 3D_DB_MAT of a binary image of size \( N \times N \times N \) can be performed in \( O(\log n) \) time on common CRCW with \( \frac{N^2}{\log n} \) processors, or in \( O(\log \log n) \) time on EREW with \( \frac{N^2}{\log n} \) processors. In [5], the task of computing the 3D_DB_MAT of a binary image of size \( N \times N \times N \) can be performed in \( O(1) \) time on an AROB with \( N^{3+\delta} \) processors, where \( \delta = \frac{1}{k} \), \( k \) is a constant and a positive integer. To the best of our knowledge, [4] and [5] are the best previous results for the 3D_DB_MAT.

The other type is the block-based MAT (BB_MAT), which is a recovering of the object by maximal 1-cubes (1-squares) in a 3D (2D) space. A maximal 1-cube (1-square) is a 1-cube (1-square) that contains in no other 1-cube (1-square). The BB_MATS [2] are defined as follows. For an \( N \times N \) binary image \( \mathcal{P} \), the 2D_BB_MAT of the binary image \( \mathcal{P} \) is defined as a problem to find a minimal set of upright 1-squares whose union corresponds exactly to the 1-pixels in \( \mathcal{P} \). The computation of the 2D_BB_MAT problem has been well studied both in the sequential and parallel domains [2], [6], [7], [8], [9], [10], [11], [12], [13]. Unfortunately, almost all of them are dedicated for solving the 2D_BB_MAT problem only and are very difficult to be extended for solving the 3D_BB_MAT problem. Up to now, no parallel algorithm has been developed for the 3D_BB_MAT. For an \( N \times N \times N \) binary image \( \mathcal{V} \), the 3D_DB_MAT of the binary image \( \mathcal{V} \) is defined as a problem to find a minimal set of upright 1-cubes whose union corresponds exactly to the 1-voxels in \( \mathcal{V} \). In this paper, we will first focus on devising an \( O(1) \) time 2D_BB_MAT algorithm which can be extended for solving the 3D_BB_MAT problem, and then devising an \( O(1) \) time 3D_BB_MAT algorithm based on the computed 2D_BB_MAT.

The remainder of this paper is organized as follows. In Section 2, we introduce the LARPBS model. In Section 3, we introduce the definitions, notations and properties of 2D_BB_MAT and 3D_BB_MAT. In Section 4, we describe our algorithm named Algorithm 2D_BB_MAT in detail. In Section 5, we describe our algorithm named Algo-
algorithm 3D_BB_MAT in detail. Finally, some concluding remarks are included in the last section.

2 The LARPBS model

In this paper, we will base on the linear array with a reconfigurable pipelined bus system (LARPBS) [14] to devise our algorithms. In the LARPBS, each processor with a local memory is identified by a unique index and denoted as $P_i$, where $0 \leq i \leq N - 1$. Optical buses offer numerous advantages including high bandwidth and low interference; however, the two most relevant properties of optical buses are their unidirectional propagation and predictable propagation delay per unit length, which enable synchronized concurrent access to an optical bus in a pipelined fashion [15]. The optical bus of the LARPBS is constructed from three identical waveguides. They are message, select and reference waveguides. The message waveguide is used for sending data, and the select and reference waveguides are used for sending address information. Each waveguide is conceptually divided into two segments, the transmitting segment and the receiving segment. Each processor $P_i$ is connected to the optical bus with two couplers. One coupler, which is a $1 \times 2$ optical switch, is used to write data on the transmitting segment of the bus. The other, which is a $2 \times 1$ optical switch, is used to read the data from the receiving segment of the bus. These switches are also called reconfigurable switches due to their function. Due to reconfigurability, an LARPBS can be partitioned into $k$ (where $k \geq 2$) independent subarrays. All these subarrays can operate independently to solve different subproblems. As in many other synchronous parallel computing systems, an LARPBS computation is a sequence of alternate global communication and local computation steps. Since the message communication time on the whole optical pipelined bus is compatible with the computation time of any arithmetic or logic operation, we can assume that a bus cycle as well as a processor cycle as one step and each step takes $O(1)$ time. This assumption for time complexity measure has been shown and adopted widely in the literature [5], [14], [16], [17]. In this paper, we adopt the same time complexity measure. The coincident pulse technique [18] is used on the LARPBS model for addressing.

2.1 Some primitive operations of the LARPBS model

Lemma 1 summarizes some primitive operations of the LARPBS model. See [14], [16] for the definitions and detailed implementations of these primitive operations. Unlike many theoretical models, such as PRAM, the LARPBS is more realistic and is likely to become feasible architecture in the near future [14], [16]. The powerful primitive operations plus the property of reconfigurability make the LARPBS model very attractive in solving the problems that are both computation and communication intensive such as MATs.

Lemma 1 [14], [16]. One-to-one communication, broadcasting, multicasting, multiple multicasting, binary prefix sum, binary summation, and compression can be done in $O(1)$ bus cycles on an LARPBS.

2.2 The general minimum finding algorithm

Lemma 2 [19]. $N$ items can be sorted in $O(1)$ time on an LARPBS of size $N^2$.

Lemma 2 implies Lemma 3. Here we name Lemma 3 as the basic minimum finding algorithm.

Lemma 3 (Algorithm BFA). The minimum value of $N$ data items can be computed in $O(1)$ time on an LARPBS of size $N^2$.

Then, based on Lemma 3 and [20], we devise a scalable $O(1)$ time minimum finding algorithm, named Algorithm MFA as described in Lemma 4. Here we omit the proof.

Lemma 4 (Algorithm MFA). The minimum value of $N$ data items can be computed in $O(1)$ time on an LARPBS of size $N^{1+c}$, where $0 < c = \frac{1}{2^{\log_2{2}} - 1} \ll 1$, $c$ is a constant.

Throughout this paper, $\frac{1}{2^{\log_2{2}} - 1}$ is denoted as $\epsilon$. Clearly, Lemma 3 is a special case of Lemma 4 (when $c = 0$). The detailed steps of this algorithm are similar to those in [20]. Note that in each step of Algorithm MFA, all processors are always fully utilized and $c$ converges toward zero very fast. It implies that the number of processors needed will reduce dramatically even if $c$ increases very slowly. If we invoke Lemma 3 (i.e., $c = 0$ in Lemma 4), it will take $N^2$ processors for us to find the minimum of these $N$ data items in $O(1)$ time; however, if we invoke Lemma 4 and choose $c = 3$, then it only takes $N^{\frac{11}{5}}$ processors for finding the minimum in $O(1)$ time. Although $c$ can be any non-negative integer, to fully take the advantage of this algorithm, we suggest that $c \geq 3$ or 5.

3 Definitions, notations and properties of BB_MAT

In this paper, a 3D (or 2D) binary image of size $N \times N \times N$ (or $N \times N$) is represented by $V$ (or $P$). A voxel (or pixel) is used to represent an image point in $V$ (or $P$). A voxel (or pixel) is represented by its Cartesian coordinates. A black/white voxel (or pixel) is also denoted as a 1/0-voxel (or pixel).
3.1 Definitions, notations and properties of 2D_BB_MAT

Let the origin of the 2D coordinate system be at the top-left corner of the image and be represented as (0, 0). The X-coordinate increases downwards and the Y-coordinate increases towards right. A pixel (i, j) is in row i and column j of the image P. The value of a pixel (i, j) is denoted as I[i, j]. For example, the notation I[i, j] = 1 represents that there exists a black pixel at coordinates (i, j).

**Definition 1** Let the number of contiguous 1-pixels in column j starting at pixel (i, j) be denoted as LC[i, j]. Let the number of contiguous 1-pixels in row i starting at (i, j) be denoted as LR[i, j]. For a pixel (i_0, j_0) ∈ {(0, +)} ∪ (0, 0), let the set of pixels (i_0 + h, j_0 + h), where i_0 + h < N, j_0 + h < N and h = 0, 1, 2, ..., be defined as the diagonal of (i_0, j_0) and be denoted as Diag[i_0, j_0]. Let the length of Diag[i_0, j_0] be denoted as |Diag[i_0, j_0]|. Let the number of contiguous 1-pixels in Diag[i_0, j_0] starting at (i, j) be denoted as LD[i, j]. Let min{|LD[i, j], LR[i, j], LC[i, j]|} be denoted as M_1D[i, j].

**Lemma 5** For all the pixels (i, j) of a fixed column j of P, where 0 ≤ i < N, their corresponding LC[i, j] can be computed in O(1) time using an LARPS of size N. Similarly, for all the 1-pixels (i, j) of a fixed row i of P, where 0 ≤ j < N, their corresponding LR[i, j] can be computed in O(1) time using an LARPS of size N. For all the 1-pixels (i, j) in a diagonal of (i_0, j_0) (i.e., Diag[i_0, j_0]), their corresponding LD[i, j] can be computed in O(1) time using an LARPS of size |Diag[i_0, j_0]|.

**Definition 2** A square image is defined as a 2D shape if all the pixels in this square image are 1-pixels. Let M_1[i, j] be the maximal square with 1-pixel (i, j) as its top-left corner, and M_1[i, j] be the side-length of M_1[i, j]. Clearly, M_1[i, j] = 0 if I[i, j] = 0.

**Lemma 6** Assume that M[i, j] = h. Then, M_1D[i, j] ≥ h, M_1D[i + 1, j + 1] ≥ h − 1, M_1D[i + 2, j + 2] ≥ h − 2, ..., M_1D[i + h − 1, j + h − 1] ≥ h. In other words, M_1D[i, j] ≥ h, M_1D[i + 1, j + 1] ≥ h, M_1D[i + 2, j + 2] ≥ h, ..., M_1D[i + h − 1, j + h − 1] ≥ h.

**Lemma 7** Assume that M_1D[i, j] = m and min{M_1D[i, j], M_1D[i + 1, j + 1], M_1D[i + 2, j + 2], ..., M_1D[i + m − 1, j + m − 1]} = m, where m ≥ h. Then, M[i, j] = h.

The 2D_BB_MAT of the binary image P is defined as a problem to find a minimal set of upright maximal 1-squares whose union corresponds exactly to the 1-pixels in P. This can be implemented by first computing M_1[i, j] and M[i, j] based on Lemma 7, and then make sure that the M_1[i, j] is not included by any other M_1[i_o, j_o], where i ≥ i_o and j ≥ j_o. If M_1[i, j] is not included by any other M_1[i_o, j_o], set T[i, j] = 1; otherwise, set T[i, j] = 0. The T[i, j] can be computed based on Lemma 8 as follows. Then, 2D_BB_MAT can be represented by the union of all the M_1[i, j] with T[i, j] = 1.

**Lemma 8** [6, 9, 12]. Set T[i, j] = 1 if max{M[i − 1, j − 1], M[i − 1, j], M[i, j − 1]} ≤ M[i, j]; otherwise, set T[i, j] = 0.

3.2 Definitions, notations and properties of 3D_BB_MAT

Let the origin of the 3D coordinate system be at the top-left-corner of the image and be represented as (0, 0, 0). The X-coordinate increases downwards. The Y-coordinate increases towards right. The Z-coordinate is perpendicular to the xy-plane and onto it. That is, the plane of the paper is the plane with Z-coordinate = 0 (i.e., the Z0-plane). A voxel (i, j, k) is located at X-coordinate i, Y-coordinate j and Z-coordinate k of V. The value of a voxel (i, j, k) is denoted as I[i, j, k]. For example, the notation I[i, j, k] = 1 represents that there exists a black voxel at coordinates (i, j, k).

**Definition 3** For a voxel (i, j, k), the side-length of a 2D (upright) maximal 1-square with (i, j, k) at its top-left corner on (i, j, k) plane (i.e., the ZK-plane) is denoted as M_2[i, j, k]. The side-length of a 2D maximal 1-square at its top-left corner (i.e., the ZK-plane) is denoted as M_1[i, j, k]. The side-length of a 2D maximal 1-square with (i, j, k) at its left-near corner on (i, j, k) plane (i.e., the ZK-plane) is denoted as M_1[i, j, k]. For a voxel (i_0, j_0, k_0) ∈ {0, +} ∪ {0, +} ∪ {0, +}, the set of voxels (i_0 + h, j_0 + h, k_0 + h), where i_0 + h < N, j_0 + h < N and h = 0, 1, 2, ..., is defined as the diagonal of (i_0, j_0, k_0) and denoted as Diag[i_0, j_0, k_0]. Let the length of Diag[i_0, j_0, k_0] be denoted as |Diag[i_0, j_0, k_0]|. Let the number of contiguous 1-voxels in Diag[i_0, j_0, k_0] starting at (i, j, k) be denoted as PD[i, j, k]. Let min{M_1[i, j, k], M_1[i, j, k], PD[i, j, k]} be denoted as M_2D[i, j, k].

**Definition 4** A cube image is defined as a 1-cube if all the voxels in this cube image are 1-voxels. Let M_1[i, j, k] be the maximal cube with 1-voxel (i, j, k) at the top-left-near corner, and M[i, j, k] be the side-length of M_1[i, j, k]. Clearly, M[i, j, k] = 0 if I[i, j, k] = 0.

**Lemma 9** Assume that M[i, j, k] = h. Then, M_2D[i, j, k] ≥ h, M_2D[i + 1, j + 1, k + 1] ≥ h − 1, M_2D[i + 2, j + 2, k + 2] ≥ h − 2, ..., M_2D[i + h − 1, j + h − 1, k + h − 1] ≥ h. In other words, M_2D[i, j, k] ≥ h, M_2D[i + 1, j + 1] ≥ h, M_2D[i + 2, j + 2] ≥ h, ..., M_2D[i + h − 1, j + h − 1] ≥ h.

**Lemma 10** Assume that M_2D[i, j, k] = m and min{M_2D[i, j, k], M_2D[i + 1, j + 1, k + 1], M_2D[i + 2, j + 2, k + 2]} = m, where m ≥ h. Then, M[i, j, k] = h.
The 3D_BB_MAT of the binary image \( \mathcal{V} \) is defined as a problem to find a minimal set of maximal 1-cubes whose union corresponds exactly to the 1-voxels in \( \mathcal{V} \). This can be implemented by first computing \( \mathcal{M}_{i,j,k} \) and \( \mathcal{M}[i,j,k] \) based on Lemma 10, and then check if the \( \mathcal{M}_{i,j,k} \) is included by any other \( \mathcal{M}_{i',j',k'} \) or not, where \( i \geq i_0, j \geq j_0 \) and \( k \geq k_0 \). If \( \mathcal{M}_{i,j,k} \) is not included by any other \( \mathcal{M}_{i',j',k'} \), set \( T[i,j,k] = 1 \); otherwise, set \( T[i,j,k] = 0 \). The \( T[i,j,k] \) can be computed based on Lemma 11 as follows. Then, 3D_BB_MAT can be represented by the union of all the \( \mathcal{M}_{i,j,k} \) with \( T[i,j,k] = 1 \).

**Lemma 11** Set \( T[i,j,k] = 1 \) if \( \max\{\mathcal{M}[i-1,j-1,k-1], \mathcal{M}[i-1,j,k-1], \mathcal{M}[i,j-1,k-1], \mathcal{M}[i-1,j,k-1], \mathcal{M}[i,j,k] \} \leq \mathcal{M}[i,j,k] \); otherwise, set \( T[i,j,k] = 0 \).

### 4 Algorithm 2D_BB_MAT

In this section, we propose an efficient \( O(1) \) time algorithm for computing 2D_BB_MAT. Although the computation of the 2D_BB_MAT has been well studied in the parallel domains; unfortunately, almost all of them are dedicated for solving the 3D_BB_MAT problem only and cannot be extended for solving the 3D_BB_MAT problem in parallel. To the best of our knowledge, up to now, this is the first parallel 2D_BB_MAT algorithm that can easily be extended for solving the 3D_BB_MAT problem.

**Algorithm 2D_BB_MAT**

**Input:** A 2D \( N \times N \) binary image, each pixel being represented by \( (i,j) \), where \( 0 \leq i, j < N \).

**Output:** 2D_BB_MAT.

**Step 1:** For each \( I[i,j] = 0 \) in \( \mathcal{P} \), set \( \mathcal{M}[i,j] = 0 \). For each \( I[i,j] = 1 \) in \( \mathcal{P} \), compute \( LC[i,j], LR[i,j] \) and \( LD[i,j] \) by performing binary prefix sum on column \( j \), row \( i \) and the diagonal which covers \( LD[i,j] \). Then each \( I[i,j] = 1 \) computes \( M[i,j] = \min\{LD[i,j], LR[i,j], LC[i,j]\} \) by invoking Lemma 4.

**Step 2:** This step is for all \( I[i,j] = 1 \) to compute their corresponding \( M[i,j] \) in parallel. Assume that \( M_{1D}[i,j] = m_{i,j} \). Then, we allocate \( m_{i,j}^{1+\epsilon} \) for a single \( I[i,j] = 1 \) to compute its \( M[i,j] \). Assume that \( \min\{M_{1D}[i,j], m_{1D}[i+1,j+1], m_{1D}[i+2,j+2], \ldots, m_{1D}[i+m-1,j+m-1] + (m-1)\} = h_{i,j} \). Then, based on Lemma 7, \( M[i,j] = h_{i,j} \).

**Step 3:** Based on Lemma 8, this step checks if each \( M[i,j] \) is included by other \( M[i_0,j_0] \) or not, where \( i \geq i_0 \), and \( j \geq j_0 \). If \( \max\{M[i,j-1], M[i-1,j], M[i-1,j-1]\} \leq M[i,j] \), then we assign 1 to \( T[i,j] \) to represent that \( M[i,j] \) is not included by any other maximal 1-square. Then, 2D_BB_MAT can be represented by all the \( \mathcal{M}_{i,j} \) with their corresponding \( T[i,j] = 1 \).

### 4.1 Time complexity analysis

Based on Lemma 5, in Step 1, all the \( N \) columns of \( LC[i,j] \) can be computed in \( O(1) \) time using an LARPBS of size \( N^2 \). Similarly, all the \( N \) rows of \( LR[i,j] \) can be computed in \( O(1) \) time using an LARPBS of size \( N^2 \). All the \( LD[i,j] \) can be computed in \( O(1) \) time using an LARPBS of size \( N^2 \) based on the prefix sum on the \( 2N-1 \) diagonals which cover \( LD[i,j] \). Then, it takes an LARPBS of size \( N^2 \) for all the 1-pixel \( (i,j) \) in \( \mathcal{P} \) to compute their corresponding \( M_{1D}[i,j] = \min\{LD[i,j], LR[i,j], LC[i,j]\} \). In Step 2, assume that \( M_{1D}[i,j] = m_{i,j} \). Then, for each \( I[i,j] = 1 \), it takes an LARPBS of size \( m_{i,j}^{1+\epsilon} \) to compute its \( M[i,j] \). For all the \( I[i,j] = 1 \), assume that \( \sum_{i,j=0}^{N-1} m_{i,j}^{1+\epsilon} = S_2 \), then it takes an LARPBS of size \( S_2 \) for all the \( I[i,j] = 1 \) to compute their corresponding \( M[i,j] \). In the worst case, \( S_2 = 1N^{1+\epsilon} + 3(N-1)1+\epsilon + 5(N-2)^2+\epsilon + \ldots + (2N-1)(N-(N-1))1+\epsilon = \sum_{i=1}^{N} (2i-1)(N-i+1)1+\epsilon \), i.e., the number of 1-pixels is \( O(N^2) \) and \( m_{i,j} = O(N) \), and hence, \( S_2 = N^{3+\epsilon} \). However, in the best case, the number of 1-pixels is 0 and \( m_{i,j} = 0 \), and hence we set their corresponding \( M[i,j] = 0 \) directly and no processors are needed for the \( M[i,j] \) computation. In general, \( 0 < S_2 < N^{3+\epsilon} \). In Step 3, it takes an LARPBS of size \( N^2 \) for all the \( M_{i,j} \) of 1-pixel \( (i,j) \) in \( \mathcal{P} \) to compute their corresponding \( T[i,j] \). From the above analysis, it takes an LARPBS of size \( \max\{N^2, S_2\} \) for \( \mathcal{P} \) to compute its corresponding 2D_BB_MAT in \( O(1) \) time. This concludes the following result.

**Theorem 1** The 2D_BB_MAT of a binary image of size \( N \times N \) can be computed in \( O(1) \) time on an LARPBS of size \( \max\{N^2, S_2\} \), where \( 0 < S_2 < N^{3+\epsilon} \), \( 0 < \epsilon = \frac{1}{1+\epsilon} \).

### 5 Algorithm 3D_BB_MAT

The main contribution of this paper is to propose an efficient \( O(1) \) time algorithm for computing 3D_BB_MAT. To the best of our knowledge, up to now, this is the first parallel algorithm for solving the 3D_BB_MAT problem.

**Algorithm 3D_BB_MAT**

**Input:** A 3D \( N \times N \times N \) binary image, each voxel being represented by \( (i,j,k) \), where \( 0 \leq i, j, k < N \).
Output: 3D_BB_MAT.

Step 1: For each $I[i, j, k] = 0$ in $\mathcal{V}$, set $M[i, j, k] = 0$. For each $I[i, j, k] = 1$ in $\mathcal{V}$, compute $M_b[i, j], M_j[i, k], M_k[i, j]$ by invoking Steps 1 and 2 of Algorithm 2D_BB_MAT and compute $PD[i, j, k]$ by performing binary prefix sum. Then compute $M_{2D}[i, j, k] = \min(M_b[i, j], M_j[i, k], M_k[i, j], PD[i, j, k])$ by invoking Lemma 4.

Step 2: This step is for all $I[i, j, k] = 1$ to compute their corresponding $M[i, j, k]$ parallel. Assume that $M_{2D}[i, j, k] = m_{i, j, k}$. Assume that $\min(M_{2D}[i, j, k], M_{2D}[i + 1, j + 1, k + 1] + 1, M_{2D}[i + 2, j + 2, k + 2] + 2, ..., M_{2D}[i + m - 1, j + m - 1, k + m - 1] + (m - 1)} = h_{i, j, k}$. Then, based on Lemma 10, $M[i, j, k] = h_{i, j, k}$.

Step 3: Based on Lemma 11, this step checks whether $M[i, j, k]$ is included by other $M[i_0, j_0, k_0]$ or not, where $i \geq i_0, j \geq j_0$ and $k \geq k_0$. If $\max(M[i - 1, j - 1, k - 1], M[i - 1, j, k], M[i - 1, j - 1, k - 1], M[i - 1, j, k], M[i - 1, j - 1, k - 1], M[i, k, j - 1], M[i, j, k], M[i, j - 1, k], M[i, j, k - 1]) \leq M[i, j, k]$, then we assign 1 to $T[i, j, k]$ to represent that $M[i, j, k]$ is not included by any other maximal 1-cube. Then, 3D_BB_MAT can be represented by all the $M[i, j, k]$ of 1-voxel $I[i, j, k] \in \mathcal{V}$ with its corresponding $T[i, j, k] = 1$.

5.1 Time complexity analysis

In Step 1, it takes an LARPBS of size $\max\{N^2, S_{2k}\}$ for all 1-voxel $(i, j, k) \in \mathcal{Z}_i$-plane, where $0 \leq k < N$ to compute Steps 1 and 2 of their corresponding 2D_BB_MAT in $O(1)$ time. Clearly, $\mathcal{V}$ is composed of $N$ $\mathcal{Z}_i$-planes. Hence, it takes an LARPBS of size $\max\{N^3, S_{3k}\}$, where $S_{3k} = \sum_{i=0}^{N-1} S_{2k}$, for all $I[i, j, k] = 1$ in $\mathcal{V}$ to compute their corresponding $M_b[i, j]$. Similarly, it takes an LARPBS of size $\max\{N^3, S_{3j}\}$, where $S_{3j} = \sum_{j=0}^{N-1} S_{2j}$, for all $I[i, j, k] = 1$ in $\mathcal{V}$ to compute their corresponding $M_j[i, k]$ and an LARPBS of size $\max\{N^3, S_{3k}\}$, where $S_{3k} = \sum_{i=0}^{N-1} S_{2i}$, for all $I[i, j, k] = 1$ in $\mathcal{V}$ to compute their corresponding $M_k[i, j]$. The all $3N^2 - 2$ diagonals, $PD[i, j, k]$, can be computed in $O(1)$ time using an LARPBS of size $N^3$. Then, the $M_{2D}[i, j, k]$ can be computed in $O(1)$ time using an LARPBS of size $N^3$ by finding the minimum in the set $\{M_b[i, j], M_j[i, k], M_k[i, j], PD[i, j, k]\}$. In summary, Step 1 takes an LARPBS of $\max\{N^3, S_{3k}, S_{3j}, S_{3\ell}\}$ processors, where $0 \leq S_{3\ell} \leq N^3 + \epsilon$. Step 2, assume that $M_{2D}[i, j, k] = m_{i, j, k}$. Then, for each $I[i, j, k] = 1$, it takes an LARPBS of size $m_{i, j, k}^{i,j,k}$ to compute its $M_{i, j, k}$ and corresponding $M[i, j, k]$. For all the $N - 1$, $I[i, j, k] = 1 \in \mathcal{V}$, let $\sum_{i, j, k} m_{i, j, k}^{i,j,k} = S_3$, then it takes an LARPBS of size $S_3$ for all $I[i, j, k] = 1 \in \mathcal{V}$ to compute their corresponding $M[i, j, k]$. In the worst case, the number of 1-voxels is $O(N^3)$ and $m_{i, j, k} = O(N)$, and hence, $S_3 = O(N^4)$. However, in the best case, the number of 1-voxels is 0 and $m_{i, j, k} = 0$, and hence we set their corresponding $M[i, j, k] = 0$ directly and no processors are needed for the $M[i, j, k]$ computation. In general, $0 \leq S_3 \leq N^4$. In Step 3, it takes an LARPBS of size $N^3$ for all the $M_{i, j, k}$ of voxels $(i, j, k) \in \mathcal{V}$ to compute their corresponding $T[i, j, k]$. From the above analysis, it takes an LARPBS of size $\max\{N^3, S_3, S_{3j}, S_{3\ell}\}$ for $\mathcal{V}$ to compute its corresponding 3D_BB_MAT in $O(1)$ time. This concludes the following result.

Theorem 2 The 3D_BB_MAT of a binary image of size $N \times N \times N$ can be computed in $O(1)$ time on an LARPBS of size $\max\{N^3, S_3, S_{3j}, S_{3\ell}\}$, where $0 \leq S_3, S_{3j}, S_{3\ell} \leq N^4 + \epsilon, 0 < \epsilon = \frac{1}{2m - 1} \ll 1$. 

6 Concluding remarks

In this paper, we develop an $O(1)$ time algorithm for computing the 2D_BB_MAT of a binary image of size $N \times N$ on an LARPBS with $\max\{N^2, S_2\}$ processors, where $0 \leq S_2 \leq N^3 + \epsilon, 0 < \epsilon = \frac{1}{2m - 1} \ll 1$, and an $O(1)$ time algorithm for computing the 3D_BB_MAT of a binary image of size $N \times N \times N$ on an LARPBS with $\max\{N^3, S_3, S_{3j}, S_{3\ell}\}$ processors, where $0 \leq S_3, S_{3j}, S_{3\ell} \leq N^4 + \epsilon, 0 < \epsilon = \frac{1}{2m - 1} \ll 1$. The worst case happens only when all $(i, j, k) \in \mathcal{V}$ (all $(i, j) \in \mathcal{P}$) are 1-voxels (1-pixels). For an image with sparse 1-voxels (1-pixels), the proposed algorithms might be very efficient and take $N^2$ processors to solve the 2D_BB_MAT problem or $N^3$ processors to solve the 3D_BB_MAT problem in $O(1)$ time. Besides, the running time of the 2D_BB_MAT algorithm has a smaller constant factor compared with all the other previous proposed $O(1)$ time 2D_BB_MAT algorithms. To the best of our knowledge, the former is the first 2D_BB_MAT algorithm that can be extended for solving the 3D_BB_MAT problem and the latter is the first parallel 3D_BB_MAT algorithm proposed for solving the 3D_BB_MAT problem.

7 Acknowledgments

This work was partially supported by National Science Council under the contract number NSC-93-2213-E-129-011.
References


