PARALLEL THINNING ON RMESH

John Jenq  
Computer Science Department  
Montclair State University  
Upper Montclair, NJ 07043, USA  
jenqj@mail.montclair.edu  

Wingning Li  
Department of Computer Science and  
Engineering  
University of Arkansas  
Fayetteville, AR 72701  
Wingning@mail.uark.edu  

Abstract  
Thinning operation is a fundamental operation in image processing. It is a typical preprocessing stage for pattern recognition and data compression. In this report we proposed an algorithm that performs thinning operation by using reconfigurable mesh RMESH multiprocessors. The time and space complexities are both $O(1)$ and therefore optimal.

1. INTRODUCTION

Thinning operation is a fundamental operation in image processing. It is a typical preprocessing stage for pattern recognition and data compression. Thinning and skeletonization are synonymous on many articles. Usually one refers to skeleton as the result of the thinning. The thinning operation makes further image processing and pattern recognition task easier by reducing both the time and space complexities. There are many applications that thinning can be applied. In the area of biomedical, for example, it can be used for analysis of white blood cells[5], the tissue cells[1], and for detecting malaria parasites[4]. It can also be used in finger print analysis [13]. In the area of optical character recognition, there are many articles published such as [10], [11], [12], [15] amongst others.

It is important that the thinning operation retains geometrical and topological information so that the thinned result can be used in future pattern recognition. Therefore for an effective thinning algorithm, it shall reduce or eliminate noise, reduce data size, retain essential features of the object such as junction points, and end points. It also shall eliminate distortions that often introduced from the thinning operations themselves.

The common method to extract the skeleton by using iterative thinning method is to remove all the contour pixels at a time, until ideally when all the lines and curves are unit width. One can classify the thinning operations into either sequential or parallel. If the $n$th iteration of the thinning algorithm depends on $(n-1)$th iteration alone then it is parallel. If the $n$th iteration also depends on pixels already processed in $n$th iteration then it is a sequential algorithm. Basically speaking, our mission is to find medial axis of the object or the centerline of the object. In a sense, the distance from contour to the medial axis of the object shall be equal as much as possible in local neighborhood. Imagine by using maximal disks to cover the whole object. The union of the center points of these maximal disks gives us the medial axis. The original medial axis transform represent scheme was proposed by Blum[2]. In many reports researchers used a window of 3 x 3 to determine, in each of the iteration, if a pixel shall be removed. For better results, bigger windows such as 5 x 5 or 7 x 7 can of course be used. If a pixel is on the contour then it is “deletable”. Since there are no exact definition of what a skeleton shall be by giving an object, using different thinning methods often generate different results. Most of the algorithms differ in how these thinned lines and curves pieces, generated by the algorithms, are connected. Furthermore, by using different algorithm, in general it will generate different distortions. There are numerous articles published that proposed different thinning or skeleton operation[3], [6], [7], [9]. Due to page limitation, Interesting reader please refers to these articles directly.

In this paper we present a parallel thinning method that run on RMESH multiprocessors. The RMESH multiprocessors are the one that used by Jenq and sahn[8]. Our algorithm obtains the skeleton in $O(1)$ time for a given $N \times N$ image and use only constant storage. So our algorithm is optimal for the RMESH multiprocessors. We organize the remaining sections as the following. In section 2, the preliminaries of RMESH architecture is
presented. In section 3, Basic data manipulation operations are presented. These operations are used as basic building blocks that are used in section four to deal with the thinning on RMESH. Section 4 discusses the parallel thinning algorithm on RMESH. We conclude our report in section 5.

2. PRELIMINARIES

The particular reconfigurable mesh architecture that we use in this paper is called RMESH. It employs a reconfigurable bus to connect together all processors. Figure 2 shows a 4x4 RMESH. By opening some of the switches, the bus may be reconfigured into smaller buses that connect only a subset of the processors.

![Figure 2. A 4 x 4 RMESH](image)

The important features of a RMESH are:

1. An $N \times M$ RMESH is a 2-dimensional mesh connected array of processing elements (PEs). Each PE in the RMESH is connected to a broadcast bus which is itself constructed as an $N \times M$ grid. The PEs are connected to the bus at the intersections of the grid. Each processor has up to four bus switches that are software controlled and that can be used to reconfigure the bus into subbuses. The ID of each PE is a pair $(i, j)$ where $i$ is the row index and $j$ is the column index. The ID of the upper left corner PE is $(0,0)$ and that of the lower right one is $(N-1,M-1)$.

2. Associated with a PE there are up to four switches that are labeled E (east), W (west), S (south) and N (north). Notice that the east (west, north, south) switch of a PE is also the west (east, south, north) switch of the PE (if any) on its right (left, top, bottom). Two PEs can simultaneously set (connect, close) or unset (disconnect, open) a particular switch as long as the settings do not conflict. The broadcast bus can be subdivided into subbuses by opening (disconnecting) some of the switches.

3. Only one processor can put data onto a given sub bus at any time.

4. In unit time, data put on a subbus can be read by every PE connected to it. If a PE is to broadcast a value in register $I$ to all of the PEs on its subbus, then it uses the command broadcast($I$).

5. To read the content of the broadcast bus into a register $R$, the statement $R = \text{content(bus)}$ is used.

6. Row buses are formed if each processor disconnects (opens) its S switch and connects (closes) its E switch. Column buses are formed by disconnecting the E switches and connecting the S switches.

3. BASIC DATA MANIPULATION OPERATIONS

In this section we define several basic data manipulation algorithms for RMESH multiprocessors. These operations are used as building blocks in the next section to develop the parallel algorithm for thinning operations. These operations can be found in [8].

3.1 Row(Column) Bus Broadcast

This operation will broadcast a datum that is initially in a PE to all other PEs that are connected to the same sub bus (either row or column). It is a special case of window broadcast operation, this operation can be done in $O(1)$ time.

3.2 Shift

Each PE has data in its $A$ variable that is to be shifted to the $B$ variable of a processor that is $S$, $S > 0$, units to the right but on the same row. One of the variant of the shift is the operation of circular shift which performs shift
with wrap around feature. These operations can be done in $O(s)$ time. If $s=1$ then the time becomes $O(1)$.

3.3 Sub MESH Bus Broadcast

This operation will broadcast a datum that is initially in a PE to all other PEs that are connected to the same sub MESH. It is similar to Row bus broadcast operation.

4. PARALLEL THINNING ON RMESH

In this section, we present the algorithm that performs thinning operation on RMESH. One of the advantages of the RMESH is its broadcast capability that we mentioned in section 2. By forming broadcasting sub-buses, for example, the geometrical or topological information on one side of the object image can be propagated to the other side of the object in constant time. This capability gives us a broader picture of the scene, and therefore allows us to make clever decision. Recall most of the pixel based algorithms use only local information to make decision on whether to delete a particular pixel under consideration. We can roughly partition our algorithm into 9 steps. The outline of the algorithm is in Figure 2.

Step 1 Shift operations to gather neighborhood information.
Step 2 identify boundary type and boundary type if it is a boundary pixel.
Step 3 if boundary pixel then form row and column buses to broadcast its $(x,y)$ coordinates and its boundary type.
Step 4 compute horizontal and vertical run lengths.
Step 5 identify center pixels for vertical and horizontal segments.
Step 6 identify center pixels with junction boundary type.
Step 7 connect junction pixels locally to form junction region group.
Step 8 connect junction region group.
Step 9 remove unwanted center points.

Figure 2 Parallel Thinning on RMESH

In Step 1, boundary pixels shall self identify themselves. A pixel can be either an internal pixel or a boundary pixel. As soon as a pixel found out it is a boundary pixel, the pixel then starts to determine its boundary type in Step 2. The boundary types could be an end point, a junction of two segments, part of a top (or bottom) of vertical segment such as in Figure 3(b), part of the left end (or right end) of a vertical segment as in Figure 3(d). Figure 3(a) and (c) are junction boundary types. There are other types that we skip the details here. Note the pixel that was under consideration is the pixel in the center of the 3 x 3 windows. Black pixels are pixels belong to the object and the whites are the background pixels. The accuracy of the identification of boundary type may be compromised by noise. Yet one can improve the accuracy if we enlarge the window size from 3 x 3 to say 5 x 5. Of course the time to process the whole picture would then be increased. This step can be achieved by using a constant time shift operations to gather information from neighbors.

Figure 3 Examples of different boundary types

The Step 3 is to propagate the information computed from Step 2 to the interior pixels and then reaches the pixels in the other end of the horizontal and vertical segments. For an edge point, such as the example in Figure 3(b), the information will then pass to both vertical and horizontal segment above and left to it. So for each individual pixel on the image plane, total of four pieces of data will it receives. Each data set contains both of the sender’s $x$ and $y$ coordinates, and the sender’s boundary type. First iteration propagates information to the pixels above if it is a bottom boundary pixel. The second iteration is to pixels below if it is a top boundary pixel. Similar operations can be done for propagating to the right and to the left of horizontal segments. All these operations can be achieved by using row or column sub buses broadcasting. Step 4 then uses the data set received during the propagation phase to compute the length of the segment. The run
length will be used in Step 9 when unwanted center points will be removed.

In Step 5, we determine if the pixel is in the center of either vertical or horizontal segment. Recall our main task is to find the centerline pixels of the object. This step also determines the distance of these pixels to the top, bottom, left, and right boundary pixels. Note it is possible that we might have two pixels claim that they are in the center for a particular segment when the length of the segment is of even. We can then arbitrary choose one. In Step 6, those centered pixels with junction boundary property such as Figure 3(a) identified in Step 2 will mark themselves for further connection task. The main purpose is to make sure the thinned skeleton connect all the lines and curve segments that generated from Step 5. In Step 7, junction boundary pixels that are locally close will be connected together. A threshold can be used to determine how close the junction pixel shall be group together. The main purpose for this step is to reduce the unwanted spur points of the end thinned result. It can be done by the combinations of shifting and sub-mesh broadcasting operations. Step 8 can be subdivided into two sub operations. The first operation allows the region pixel groups to across the junctions to connect the line segments that are suppose to belong to the same segment before the thinning. The result can be used in the recognition phase when the thinning is done. This step can be achieved by iterate through different kind of junction boundary types. By forming sub buses simult aneously and then followed by broadcast operations. The second sub step is to turn on the pixels that are between these junction region groups. At this point, the junctions are connected so are all these line segments. The thinned result is connected. Step 9 is called to remove unwanted center line pixels based on the run length that we received from Step 4. It can be achieved by horizontal and vertical sub buses broadcasting to make sure the deletion of the pixels do no harm to the connectivity of the image. All the steps takes constant time and time space, therefore the complexities are linear.

5. CONCLUSION

A constant time algorithm that performs thinning operation on $N \times N$ RMESH has been presented. The space complexity is also constant. Both time and space complexities are therefore optimal.

6. REFERENCES