# Generalized Algorithm for Two Dimensional Digital Image Skeletonization

P Srinivasa Rao
ECE Dept
CVR College of Engineering
Hyderabad, India

M Madhavi Latha
Director-ITC
JNTUH, Kukatpally
Hyderabad, India

## ABSTRACT

This paper proposes a novel algorithm for the skeletonization of binary and gray digital images. We are proposing an algorithm which can be easily implemented on Field Programmable Gate Arrays (FPGA).In this paper , we are proposing an algorithm for two dimensional digital image skeletonization, which can be easily extendible to three dimensional images also. No complex commutations, large set of structuring elements are used, which makes FPGA implantation efficient. Resulting skeletons for many objects are presented in the paper.

## Keywords

Skeleton, Thinning, shape representation, FPGA

## 1. INTRODUCTION

Representing and understanding shapes play central roles in graphics and vision applications. These applications benefit from some form of shape descriptors that lies within a solid object. The skeleton of a 2D object consists of (1D) linear elements where as skeleton of a 3D object may consists of a 2D object known as surface skeleton or 1D known as curve skeleton.[1] A skeleton captures essential topology and shape information of the object in a simple form is extremely useful in solving various problems such as character recognition, model matching and retrieval and medical image analysis[2]. Given an input binary image or gray image, skeletonization changes non skeletal object pixels in to back ground pixels. Regardless of the scheme adopted to perform skeletonization, the resulting skeleton is a union of curves placed symmetrically with respect to the boarder of the object. The literature on 2D skeletonization is very rich, but most of the literature is devoted for binary image skeletonization. There exist few algorithms which work on gray images. In most of the gray image skeletonization, gray image is first converted in to binary and binary image skeletonization algorithms are applied. Converting gray image into binary, some information may be lost. To overcome the above limitation we are proposing a generalized algorithm which works on both binary and gray images also.

## 2. IMAGE SKELETONIZATION METHODS

There are a variety of methods proposed for image skeleton extraction in the literature. In general, they can be classified in 3 categories[4]:

a) Boundary peeling or iterative method

b) Euclidean Distance coding/transform

c) Distance transform - Voronoi Diagram

### 2.1 Boundary peeling or iterative method

This method iteratively peels off the boundary pixel-by-pixel, layer-by-layer from outside to inside of the object where removal does not affect the topology of the object. This is a repetitive, time intensive process of testing and deletion each pixel. The difficulty of this method is that the set of rules defined for removing pixel is dependent highly on the type of image and that different set of rules will be applied for different type of images. However, this method is good for connectivity preservation

### 2.2 Euclidean Distance coding/transform

This method is based on the distance of each replace to the boundary, and tries to directly extract skeleton replaces by finding the local maximum replaces (replaces in the center of object). The distance coding is based on Euclidean distance or the approximation to Euclidean distance. This method is faster and can be done in high degree of parallelism; however the output is not guaranteed to preserve connectivity The main problem of this method is connectivity. In order to connect local maxima using uphill climbing rules, they added saddle points, which are local minima along a skeleton, to the skeletal point set. However, direct extension of this algorithm to 3D is difficult because there are not necessarily unique sequences of replaces around a given replace.

### 2.3 Distance transform - Voronoi Diagram

This method , theoretically, guarantees connected skeletons, is best suited for polygonally defined objects. However, real image data such as CT, MRI with large amount of pixels and noise, cause Voronoi diagram to be very dense and the computation time to be expensive because they generate too many nodes and lines. Computing by this method is also impossible for small computer with small memory since the whole image data (with 256 x 256 x 128) requires at least 8M of memory to load the image. So far, no algorithm works well with a general image, nor satisfies the accuracy and time performance requirements. Voronoi method produces good result, but the computing time is large, Border erosion method is faster than Voronoi, but is also less accurate. Distance Transform method gives the best performance, but also is difficult in implementation and programming to preserve connectivity. Connectivity step (or how to connect all clusters) is also the most difficult step that many authors have proposed algorithms to solve but ended up either too complicated or inefficient.

Out of these methods, our algorithm comes under the category of boundary peeling method. The advantage of this method is that, it can be easily implemented on FPGA, another advantage of this method sis that it requires less number of computations. The mail disadvantage of this method is that, if noise existed in the given input image then the skeleton obtained will not truly represents shape of the object. The skeleton of an object must satisfy the following four requirements

1. Centeredness satisfaction: Skeleton is geometrically centered within the object boundary or as close as possible.
2. Connectivity preservation: The output skeleton should have the same connectivity as the original object and should not contain any back ground pixels.
3. Topology must remain constant
4. As thin as possible:1 pixel thin is the requirement for the 2D skeleton and in 3D as thin as possible (4 voxel thin )

## 3. TERMINOLGY

### 3.1 Binary Image:

The term binary image refers to the image with all pixels carrying the value of 0 or 1. In image visualization, 0's will be displayed as black and 1's will be displayed as white.

### 3.2 Gray Image:

The term gray image refers to the image with all pixels carrying a values in between 0 to 255. In image visualization, 0's will be displayed as black and 255's will be displayed as white. Some cases back ground pixel values may not be zero, in that case from the image histogram , we can able decide the back ground pixel value. In image histogram, the minimum value will be taken as back ground pixel value.

### 3.3 Notations used:

**For binary images:**

Back ground pixel: A pixel with value =0, black
Object pixel: A pixel with value =1,white
Boundary pixel: A pixel with at least one of its neighbor value =0, black

**For gray images:**

Back ground pixel: A pixel with value =minimum (from histogram)

Object pixel: A pixel with value = Maximum (from histogram)

Boundary pixel: A pixel with at least one of its neighbor value = minimum (from histogram)

## 4. PROPOSED ALGORITHM

Step1: Raster scan the given grey image by a 3x3 window.

Step2: Identify whether pixel under processing is corner pixel or boarder pixel.

Step3: If it is corner pixel, retain the pixel else delete the pixel

Step4: Remove the outer layer by retaining the corner pixels

Step5: Repeat the process until idempotent is reached

### 4.1 Conditions for identifying the corner pixels

In each iteration, take a 3x3 sub-matrix from input, if the following conditions are satisfied then they are corner pixels.

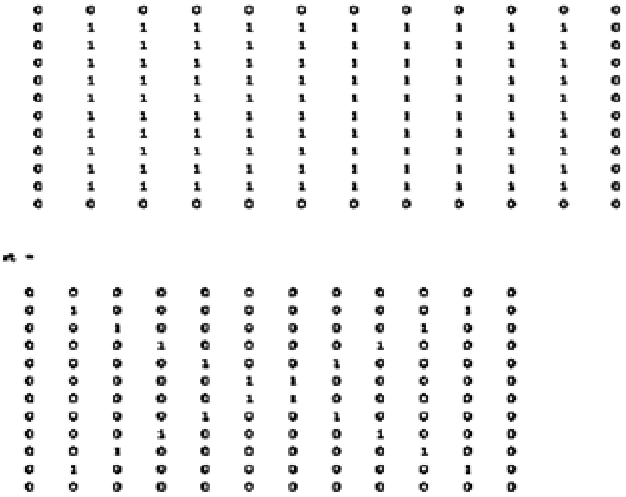
```
if (d(1,1)= =0 && d(3,3)= =1 || d(1,1) = =1 && d(3,3)= =0)
                    or
if(d(1,3)= =0 && d(3,1)= =1 || d(1,3) = =1 && d(3,1)= =0)
                    or
if (d(1,2)= =0 && d(3,2)= =1 || d(1,2) = =1 && d(3,2)= =0)
                    or
if  (d(2,1)= =0 && d(2,3)= =1 || d(2,1) = =1 && d(2,3)= =0)
                    or
if  (sum of elements of d <6)
```

### 4.2 Skeletonization of even sized objects:

In this proposed algorithm, we are using 3x3 mask is selected; it cannot detect two element structures. In any image, if there exists two element thick objects, its skeleton will be disconnected as shown the results. To overcome this disadvantage, we are using 5x5 mask processing the digital image.

## 5. RESULTS

We tested the proposed algorithm on number of test images as well as real time images. It is working satisfactory. The proposed algorithm is implemented in MATLAB version R2010a.The algorithm is executed on Intel core 2 duo, 3 GHz processor with 2 GB RAM.

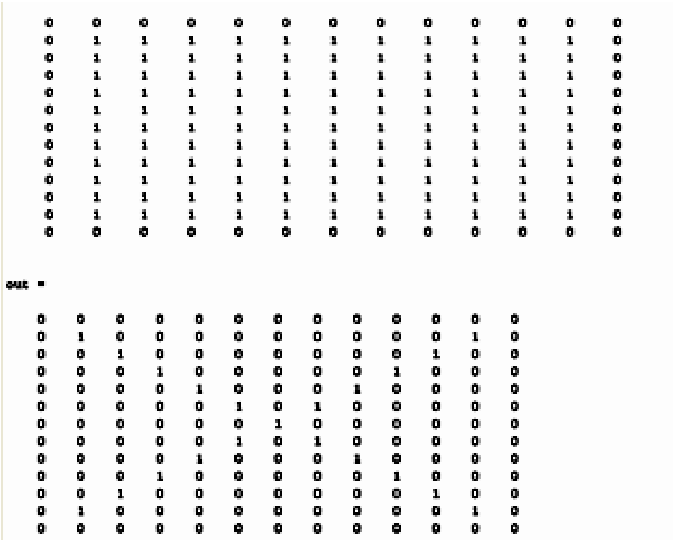


```
     0     0     0     0     0     0     0     0     0     0     0     0
     0     1     1     1     1     1     1     1     1     1     1     0
     0     1     1     1     1     1     1     1     1     1     1     0
     0     1     1     1     1     1     1     1     1     1     1     0
     0     1     1     1     1     1     1     1     1     1     1     0
     0     1     1     1     1     1     1     1     1     1     1     0
     0     1     1     1     1     1     1     1     1     1     1     0
     0     1     1     1     1     1     1     1     1     1     1     0
     0     1     1     1     1     1     1     1     1     1     1     0
     0     1     1     1     1     1     1     1     1     1     1     0
     0     1     1     1     1     1     1     1     1     1     1     0
     0     0     0     0     0     0     0     0     0     0     0     0

st =

     0     0     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     1     0
     0     0     1     0     0     0     0     0     0     1     0     0
     0     0     0     1     0     0     0     0     1     0     0     0
     0     0     0     0     1     0     0     1     0     0     0     0
     0     0     0     0     0     1     1     0     0     0     0     0
     0     0     0     0     0     1     1     0     0     0     0     0
     0     0     0     0     1     0     0     1     0     0     0     0
     0     0     0     1     0     0     0     0     1     0     0     0
     0     0     1     0     0     0     0     0     0     1     0     0
     0     1     0     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     0     0     0     0
```

**Figure1: 12x12 binary input image and it's Skelton**

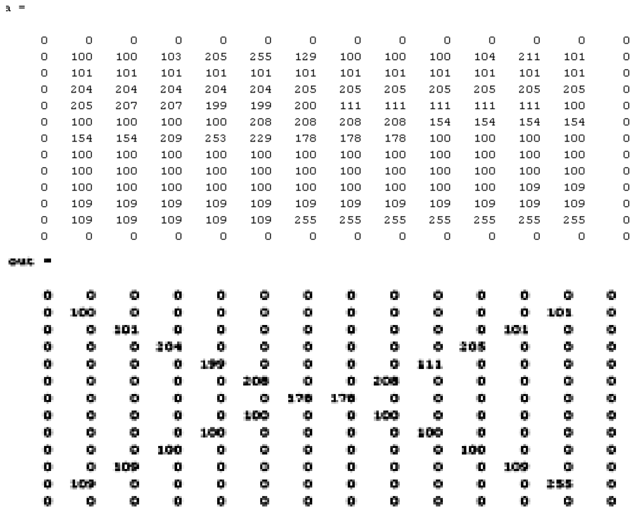


```
   0    0    0    0    0    0    0    0    0    0    0    0    0
   0    1    1    1    1    1    1    1    1    1    1    1    0
   0    1    1    1    1    1    1    1    1    1    1    1    0
   0    1    1    1    1    1    1    1    1    1    1    1    0
   0    1    1    1    1    1    1    1    1    1    1    1    0
   0    1    1    1    1    1    1    1    1    1    1    1    0
   0    1    1    1    1    1    1    1    1    1    1    1    0
   0    1    1    1    1    1    1    1    1    1    1    1    0
   0    1    1    1    1    1    1    1    1    1    1    1    0
   0    1    1    1    1    1    1    1    1    1    1    1    0
   0    1    1    1    1    1    1    1    1    1    1    1    0
   0    1    1    1    1    1    1    1    1    1    1    1    0
   0    0    0    0    0    0    0    0    0    0    0    0    0

out =

   0    0    0    0    0    0    0    0    0    0    0    0    0
   0    1    0    0    0    0    0    0    0    0    0    1    0
   0    0    1    0    0    0    0    0    0    0    1    0    0
   0    0    0    1    0    0    0    0    0    1    0    0    0
   0    0    0    0    1    0    0    0    1    0    0    0    0
   0    0    0    0    0    1    0    1    0    0    0    0    0
   0    0    0    0    0    0    1    0    0    0    0    0    0
   0    0    0    0    0    1    0    1    0    0    0    0    0
   0    0    0    0    1    0    0    0    1    0    0    0    0
   0    0    0    1    0    0    0    0    0    1    0    0    0
   0    0    1    0    0    0    0    0    0    0    1    0    0
   0    1    0    0    0    0    0    0    0    0    0    1    0
   0    0    0    0    0    0    0    0    0    0    0    0    0
```

**Figure2: 13x13 binary input image and it's Skelton**



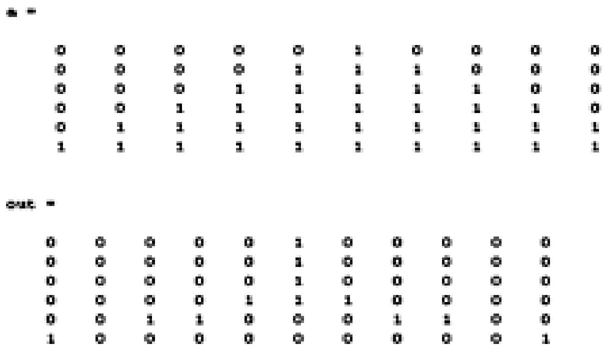
```
a =

   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0  100  100  103  205  255  129  100  100  100  104  211  101    0
   0  101  101  101  101  101  101  101  101  101  101  101  101    0
   0  204  204  204  204  204  205  205  205  205  205  205  205    0
   0  205  207  207  199  199  200  111  111  111  111  111  100    0
   0  100  100  100  100  208  208  208  208  154  154  154  154    0
   0  154  154  209  253  229  178  178  178  100  100  100  100    0
   0  100  100  100  100  100  100  100  100  100  100  100  100    0
   0  100  100  100  100  100  100  100  100  100  100  100  100    0
   0  100  100  100  100  100  100  100  100  100  100  109  109    0
   0  109  109  109  109  109  109  109  109  109  109  109  109    0
   0  109  109  109  109  109  255  255  255  255  255  255  255    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0

out =

   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0  100    0    0    0    0    0    0    0    0    0    0  101    0
   0    0  101    0    0    0    0    0    0    0    0  101    0    0
   0    0    0  204    0    0    0    0    0    0  205    0    0    0
   0    0    0    0  199    0    0    0    0  111    0    0    0    0
   0    0    0    0    0  208    0    0  208    0    0    0    0    0
   0    0    0    0    0    0  178  178    0    0    0    0    0    0
   0    0    0    0    0  100    0    0  100    0    0    0    0    0
   0    0    0    0  100    0    0    0    0  100    0    0    0    0
   0    0    0  100    0    0    0    0    0    0  100    0    0    0
   0    0  109    0    0    0    0    0    0    0    0  109    0    0
   0  109    0    0    0    0    0    0    0    0    0    0  255    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

**Figure3: 12x12 gray input image and it's Skelton**



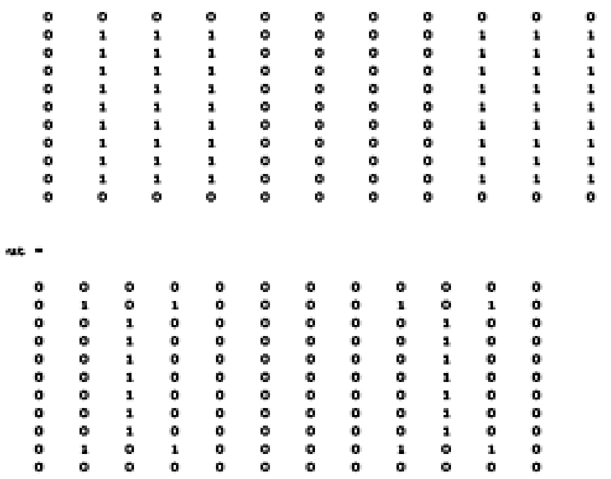
```
a =

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

out =

   0    0    0    0    0    1    0    0    0    0    0
   0    0    0    0    0    1    0    0    0    0    0
   0    0    0    0    0    1    0    0    0    0    0
   0    0    0    0    1    1    1    0    0    0    0
   0    0    1    1    0    0    0    1    1    0    0
   1    0    0    0    0    0    0    0    0    0    1
```

**Figure4: 11x6 binary input image and it's Skelton**



```
   0    0    0    0    0    0    0    0    0    0    0
   0    1    1    1    0    0    0    0    1    1    1
   0    1    1    1    0    0    0    0    1    1    1
   0    1    1    1    0    0    0    0    1    1    1
   0    1    1    1    0    0    0    0    1    1    1
   0    1    1    1    0    0    0    0    1    1    1
   0    1    1    1    0    0    0    0    1    1    1
   0    1    1    1    0    0    0    0    1    1    1
   0    1    1    1    0    0    0    0    1    1    1
   0    1    1    1    0    0    0    0    1    1    1
   0    0    0    0    0    0    0    0    0    0    0

out =

   0    0    0    0    0    0    0    0    0    0    0    0
   0    1    0    1    0    0    0    0    1    0    1    0
   0    0    1    0    0    0    0    0    0    1    0    0
   0    0    1    0    0    0    0    0    0    1    0    0
   0    0    1    0    0    0    0    0    0    1    0    0
   0    0    1    0    0    0    0    0    0    1    0    0
   0    0    1    0    0    0    0    0    0    1    0    0
   0    0    1    0    0    0    0    0    0    1    0    0
   0    0    1    0    0    0    0    0    0    1    0    0
   0    1    0    1    0    0    0    0    1    0    1    0
   0    0    0    0    0    0    0    0    0    0    0    0
```

**Figure5: two objects binary input image and its Skelton**

**Table1: Execution of different types of images**

| Image type | Grid size | Shape of the object | Execution time (in seconds) |
|---|---|---|---|
| Binary | 12x12 | square | 0.29 |
| Binary | 13x13 | rectangle | 0.15 |
| Binary | 11x6 | triangle | 0.07 |
| Binary | 12x11 | two objects | 0.07 |
| Gray | 12x12 | square | 0.07 |

Most of the algorithms proposed will works on specific type of images, where are our algorithm will work on all types of images as well as it works on both binary and gray images also. We have tested our algorithm on square, rectangle, triangle, two objects in one image. We have also tested our algorithm on both binary and gray images. Figures 1 to 5 are results of standard test images.We have also tested our algorithm on real time images. Figure 7 is the sketon of the figure 6.

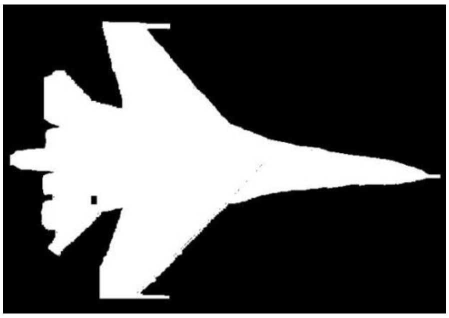


**Figure 6 : Sukoi binary input image**

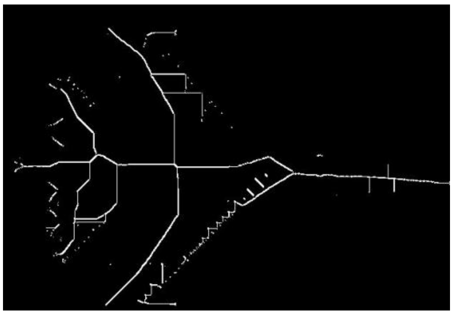


**Figure 7 : skeletonized output image**

From the above figures, we can conclude that our algorithm works test and real time images. It also works on binary and gray images also.

## 6. FUTURE SCOPE

In this paper we used 3x3 mask for skeletonization, the limitation with this mask is that it will not detect two element thick objects, this is solved by taking 5x5 mask for the skeletonization. We are also planning to extend this algorithm to 3 dimensional images also.

## 7. REFERENCES

[1]. Tao Ju, Matthew L. Baker, and Wah Chiu "Computing a family of skeletons of volumetric models for shape escription" Comput Aided Des. May 2007; 39(5): 352–360.

[2]. Lakshmi, , Punithavalli, M.A Survey on Skeletons in Digital Image Processing Digital Image Processing, 2009 International Conference on DIP 260 – 269

[3] F. Leymarie, M.D. Levine, "Simulating the Grass Fire Transform Using an Active Contour Model", IEEE Trans on Pattern Analysis and Machine Intelligence, vol.14, no. 1, Jan 1992, pp. 56-75.

[4] Y.F. Tsao and K.S. Fu, "A Parallel Thinning Algorithm for 3-D Pictures", CGIP no. 17, 1981, pp. 315-331.

[5] C.M. Mao and M. Sonka, "A Fully Parallel 3D Thinning Algorithm and Its Applications", Computer Vision and Image Understanding, vol. 64, no. 3, 1996, pp. 420-433.

[6] S.Lobregt, P.W. Verbeek, and F.C.A. Groen, "Three-Dimensional Skeletonization: Principle and Algorithm", IEEE Transaction on PAMI, vol. 2, 1980, pp.75-77.

[7] C. Arcelli and G. Saniti di Baja, "Finding Local Maxima in a Pseudo-Euclidean Distance Transform", Computer Vision, Graphics and Image Processing, vol. 43, 1988, pp. 361-367.

[8] Y. Zhou, A. Kaufman, and A.W. Toga, "3D skeleton and Centerline Generation Based on an Approximate Minimum Distance Field", The Visual Computer, vol.14, no. 7, 1998, pp 303-314.

[6] L. Dorst, "Pseudo-Euclidean Skeletons", Proc. Eighth Int'l Conf. Pattern Recognition, 1986, pp. 286-289.

[7] R.L. Ogniewicz and O. Kubler, "Hierarchic Voronoi Skeletons", Pattern Recognition, vol. 28, no. 3, 1995, pp. 343-359.

[8] R.L. Ogniewicz and M. Ilg, "Voronoi skeletons: Theory and applications", Proc. Conf. On CVPR, 1992, pp. 63-69.

[9] R.L. Ogniewicz, "Skeleton-Space: a Multiscale Shape Description Combining Region and Boudary Information", Proc. CVPR, 1994, pp. 746-751.

[10] G. Borgefors, "Distance Transformation on Digital Images", Computer Vision Graphics Image Processing, vol 34, 1986, pp. 344-371.

[11] Itoh T, Yamaguchi Y, Koyamada K, "Volume thinning for automatic isosurface propagation. IEEE Proceeding of Visualization'96, San Francisco, CA, Assoc. for Computing Machinery, New York, NY, 303-310.

[12] Payne BA, Toga AW, "Distance field manipulation of surface models", IEEE Comput Graph Appl 121992, pp.65-71.

[13] C. W. Niblack, P.B. Gibbons, D. W. Capson, "Generating Skeletons and Centerlines from the Distance Transform," CVGIP, vol. 54 no. 5, sept. 1992, pp. 420-437.

[14] N. Gagvani, "Skeleton and Volume Thinning in Visualization" MS Thesis, Dept. of Electrical and Computer Engineering, Rutgers Univ., New Brunswick, N.J. June, 1997

[15] J.L. Helman, L. Hesselink, "Visualization of Vector Field Topology in Fluid Flows", IEEE Computer Graphics and Application, vol. 11, no.3, 1991, pp. 36-46.

[16] Y. Zhou, A.W. Toga, "Efficient Skeletonization of Volumetric Objects", IEEE Trans on Visualization and Computer Graphics, vol.5, no. 3, 1999, pp 196-209.