2D/3D Skeletonization Using Sequential/Parallel Thinning

ECE AIP Project 3, April 15, 2009

Wei Lu 00649439

1. Introduction

Skeletonization is a morphology operation that erodes an object iteratively until it is unit pixel/voxel wide while following the geometry preservation and the connectivity preservation.

In this project, we apply sequential thinning method [1] to deal with 2D case and parallel thinning method [2] to deal with the 3D case. Schemes to detect end points and junction points are proposed respectively.

2. Basic Morphology Operations

Binary Erosion
The erosion of object **X** by structuring element **B** composes of all the elements belong to **X** such that there $\exists b \in \mathbf{B}$: the shift-by-b version of those elements also belong to **X**. Mathematically,

$$\mathbf{X} \ominus \mathbf{B} = \{p = x + b \in X \mid x \in \mathbf{X}, b \in \mathbf{B}\} \tag{1}$$

Hit-or-Miss Transformation
The hit-or-miss transformation (HMT) of object **X** by structuring element $\mathbf{B} = (\mathbf{B_1},\ \mathbf{B_2})$ composes all the elements belong to **X** such that for $\forall\, b_1 \in \mathbf{B_1}$ and $\forall\, b_2 \in \mathbf{B_2}$: the shift-by-$b_1$ version of those elements belong to **X** while the shift-by-b_2 version of that element must belong to the exterior of **X**. Mathematically,

$$\mathbf{X} \otimes \mathbf{B} = \left\{ \begin{matrix} x \in \mathbf{X} \mid x + b_1 \in \mathbf{X}\, for\ \forall\, b_1 \in \mathbf{B_1}; \\ x + b_2 \in \mathbf{X^c}\, for\ \forall\, b_2 \in \mathbf{B_2};\ \mathbf{B} = (\mathbf{B_1},\ \mathbf{B_2}) \end{matrix} \right\} \tag{2}$$

It can be shown that

$$\mathbf{X} \otimes \mathbf{B} = (\mathbf{X} \ominus \mathbf{B_1}) \cap (\mathbf{X^c} \ominus \mathbf{B_2}) \tag{3}$$

Thinning Operation
Thinning, one application of hit-or-miss transformation, is defined as

$$\mathbf{X} \oslash \mathbf{B} = \mathbf{X} \backslash (\mathbf{X} \otimes \mathbf{B}) \tag{4}$$

3. Proposed 2D Skeletonization Method and End Points, Bifurcations Detection

In our 2D skeletonization method, we iteratively apply sequential thinning operation with Golay alphabet of structuring elements 'L' and 'E' (see **Fig. 1**) followed with a group of correction structuring elements 'C' (see **Fig. 2**). Note, thinning with structuring elements 'L' and 'E' alone wouldn't work with some object structures as shown in **Fig. 2** but can be thinned by our correction structuring elements.

The detection of end points is realized by HMT with a group of structuring elements 'I' shown in **Fig. 3**. Finally, bifurcations are detected by HMT with a group of structuring elements 'Y' and 'T' which are shown in **Fig. 4**. The generic outline of this skeletonization as well as the detection algorithm is given in **Algorithm 1**. Note, structuring element 'L' is actually a deleting element, element 'E' is used to smooth the obtained skeleton, and the 'L' operation right after the 'E' operation is necessary because there exists the case that some one-pixel elements prevent the correct deletion provided by the 'L' operation.

```
Program 2D _Skeletonization
   Skeletonized_Object = 2D_Sequential_Thinning (Object);
   End_Points = 2D_End_Points_Detection (Skeletonized_Object);
   Bifurcations = 2D_Bifurcations_Detection (Skeletonized_Object);
   Display ();
End 2D _Skeletonization

Program 2D_Sequential_Thinning
   Do
       Object = Thinning (Object, 'L');
   While not convergent
   Do
      Object = Thinning (Object, 'E');
      Object = Thinning (Object, 'L');
      Object = Thinning (Object, 'C');
   While not convergent
End 2D_Sequential_Thinning

Program 2D_End_Points_Detection
   End_Points = HMT (Skeletonized_Object, 'I');
End 2D_End_Points_Detection

Program 2D_Bifurcations_Detection
   Bifurcations = HMT (Skeletonized_Object, 'T' & 'Y');
End 2D_ Bifurcations _Detection
```

Algorithm 1. 2D skeletonization with end points, bifurcations detection

4. Proposed 3D Skeletonization Method and End Points, Junction Points Detection

In our 3D skeletonization method, we apply Ma and Sonka's parallel thinning method [2] with structuring elements A, B, C, and D (most of them are 5-by-5-by-5 masks, see **Fig. 5** for details). The detection of tail points is achieved with the following rules defined in [2]:

- p is called a line-end point if p is 26-adjacent to exactly one object point,

- p is called a near-line-end point if p is 26-adjacent to exactly two object points which are:
  - either s(p) and e(p), or s(p) and u(p) but not both;
  - either n(p) and w(p), or u(p) and w(p) but not both;
  - either n(p) and d(p), or e(p) and d(p) but not both;
- p is called a tail point if it is either a line-end point near-line-end point; otherwise it is called a "non-tail point", where e(p), w(p), n(p), s(p), u(p) and d(p) are the east, west, north, south, up, and down neighbors of p, respectively.

Note the process of tail point detection and 3D skeletionization are indeed (recursively) embedded with each other. Just see that tail point detection will generate new tail points by using the current result of 3D skeletonization, while in each thinning step of 3D skeletonization, we need to keep those known tail points as undeletable elements.

When it comes to junction point detection, the explicit enumeration of all possible structuring elements in 3D would become extremely clumsy. Just imagine the case of extending 2D structuring elements 'Y' and 'T' to 3D would hopefully yield 80 more times structuring element. One possible rescue as well as exploiting the 2D elements is to respectively project 26-adjancent neighboring (N26) elements to xy, yz, zx plane, and then apply 2D bifurcations detection scheme we mention in **Section 3**. Note here the projection resembles a logical addition.

Well, unfortunately that's not the whole story of 3D structuring elements. Some special structures are not simply the extension of their 2D versions (See **Fig. 6**). What we did in this project only exploited the simple extension from 2D structuring elements. Therefore further improvement can be made if better junction point detection scheme is used. The generic outline of this skeletonization as well as the detection algorithm is given in **Algorithm 2**.

```
Program 3D _Skeletonization
    Skeletonized_Object = 3D_Parallel_Thinning (Object);
    Tail_Points = 3D_Tail_Points_Detection (Skeletonized_Object);
    Junction_Points = 3D_Junction_Points_Detection (Skeletonized_Object);
    Display ();
End 3D _Skeletonization

Program 3D_Parallel_Thinning
    Do
        Tail_Points = 3D_Tail_Points_Detection (Object);
        Object1 = Thinning (Object, 'A');
        Object2 = Thinning (Object, 'B');
        Object3 = Thinning (Object, 'C');
        Object4 = Thinning (Object, 'D');
        Object  = Intersection (Object1, Object2, Object3, Object4);
        Restore_Tail_Points (Object,Tail_Points);
    While not convergent
End 3D_Parallel_Thinning

Program 3D_Tail_Points_Detection
    For each voxel V of current skeletonized Object
        N = Count_Point#_In_N26 (Object,V);
        If (N == 1 or (N == 2 and the neighbors are positioned in a near-line-end way))
            V is a tail point;
        End if
    End for
End 3D_Tail_Points_Detection

Program 3D_Junction_Points_Detection
    Nx = N26_Projection_X (Skeletonized_Object);
    Ny = N26_Projection_Y (Skeletonized_Object);
    Nz = N26_Projection_Z (Skeletonized_Object);
    For each projection slice S
         Junction_Points += 2D_Bifurcation_Detection (S);
    End for
    Junction_Points += 3D_Structures_Exception_Detection (Skeletonized_Object);
End 3D_Junction_Points_Detection
```

Algorithm 2. 3D skeletonization with end points, junction points detection

5. Experiment Results

Some primary experiment results are shown in **Fig. 7**.

6. Conclusion

Originated from some ideas in literature, we develop a 2D/3D skeletonization method based on sequential/parallel thinning operation. It can preserve the geometric and connectivity characteristics to some extent.

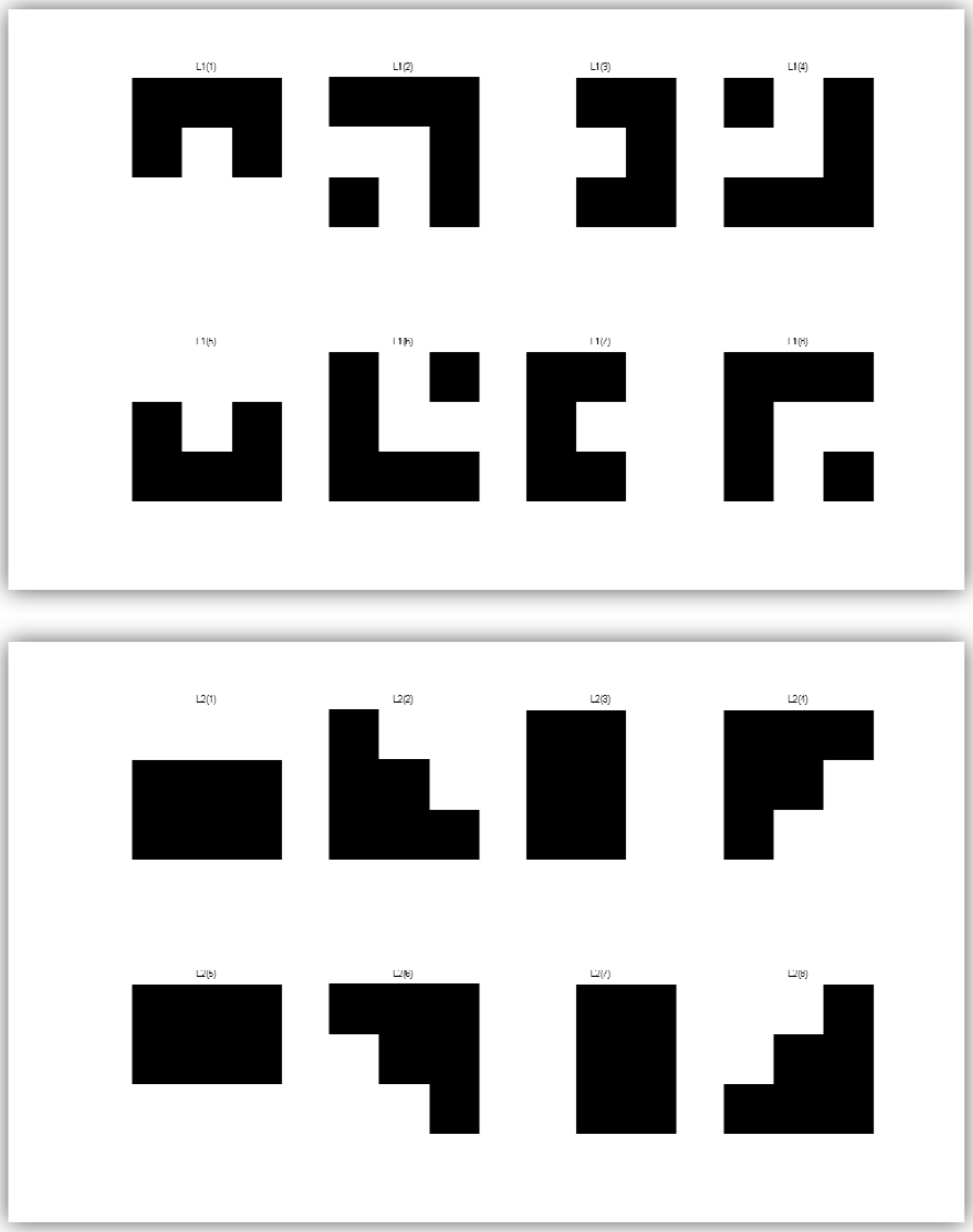


Fig. 1 2D Structuring Element L: the white block indicates 1 while black one indicates 0. The first piece is L_1, and the second piece is L_2 where $L = (L_1, L_2)$. It follows the same rule in the next few 2D figures.

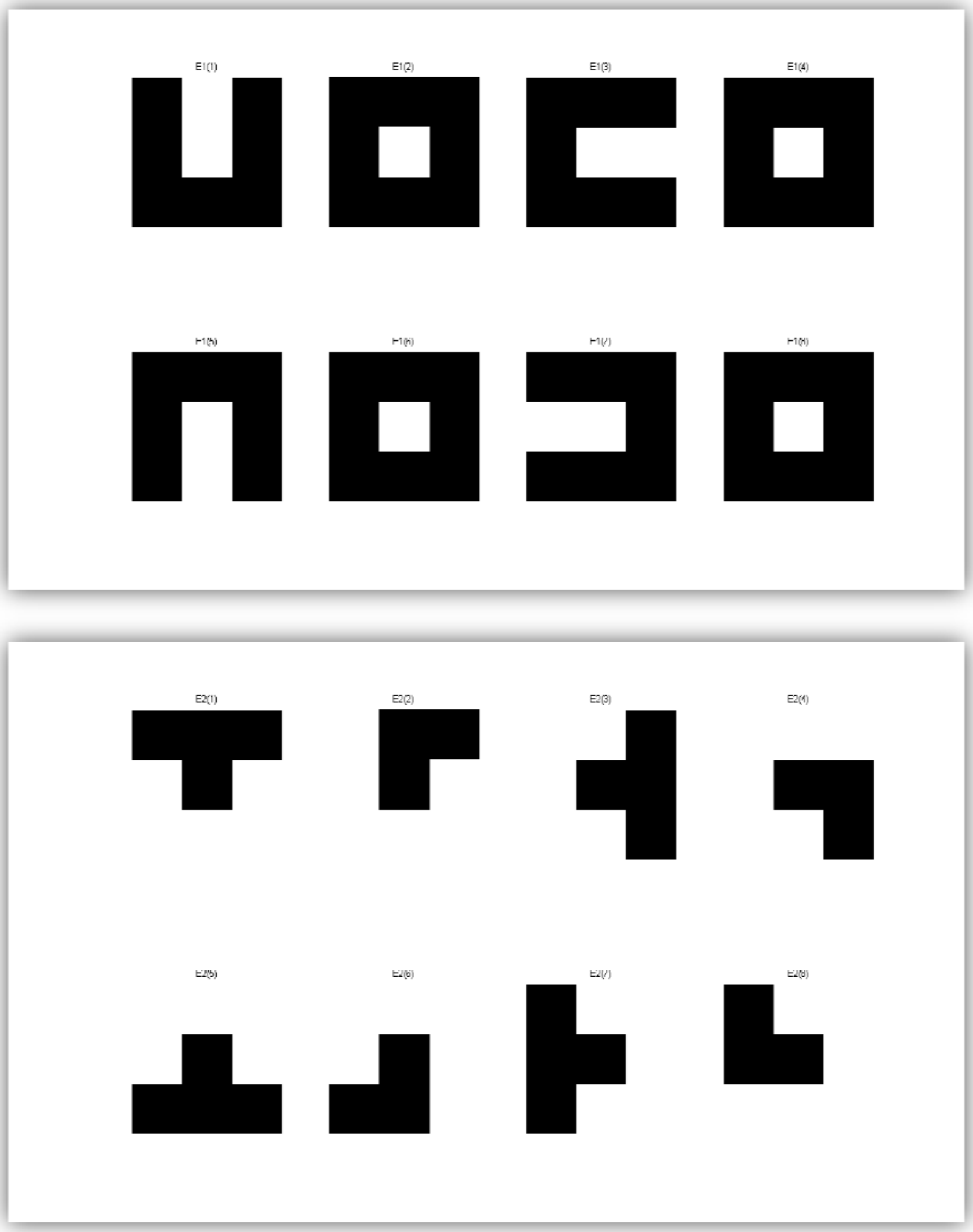


Fig. 1 2D Structuring Element E

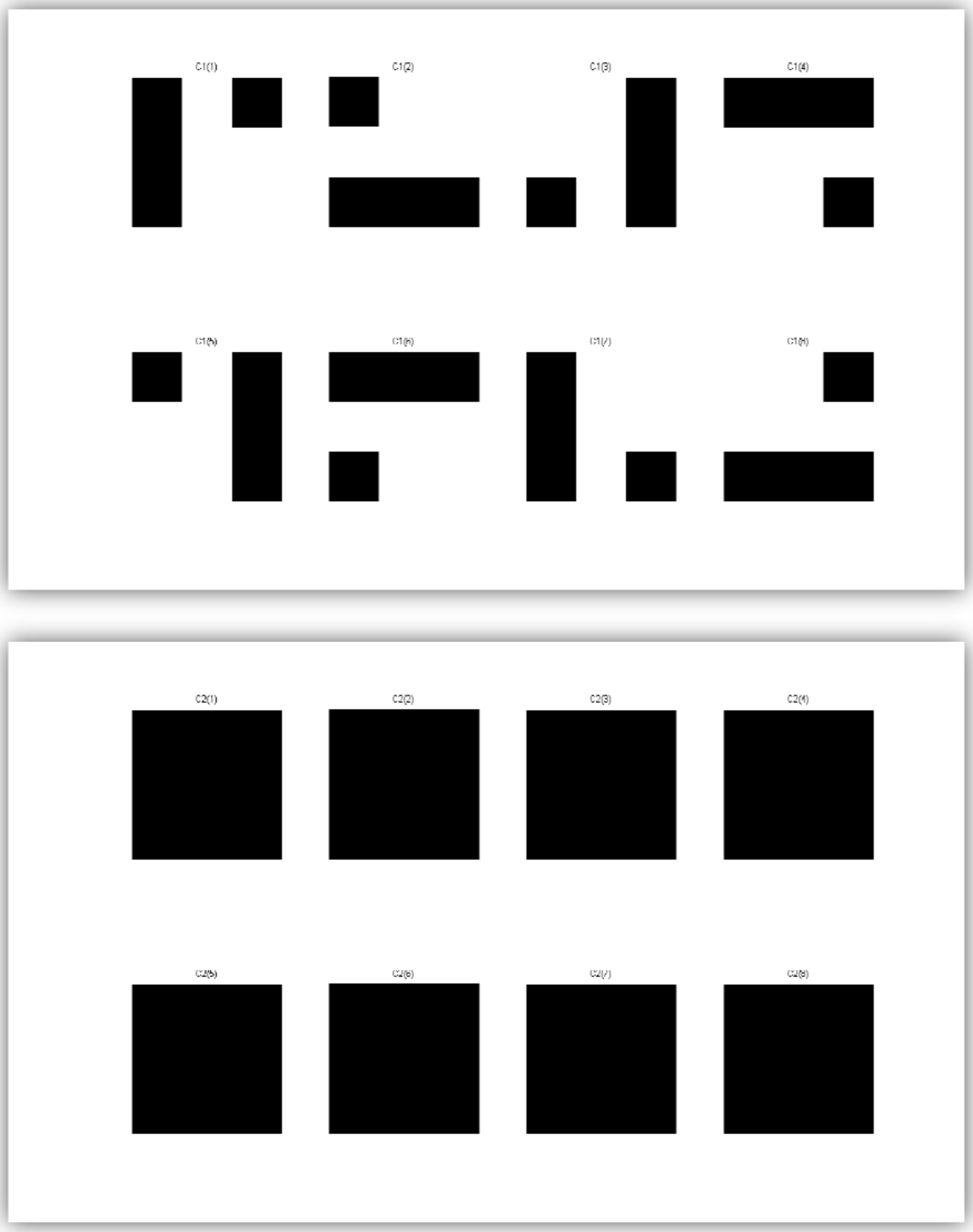


Fig. 2 2D Structuring Element C

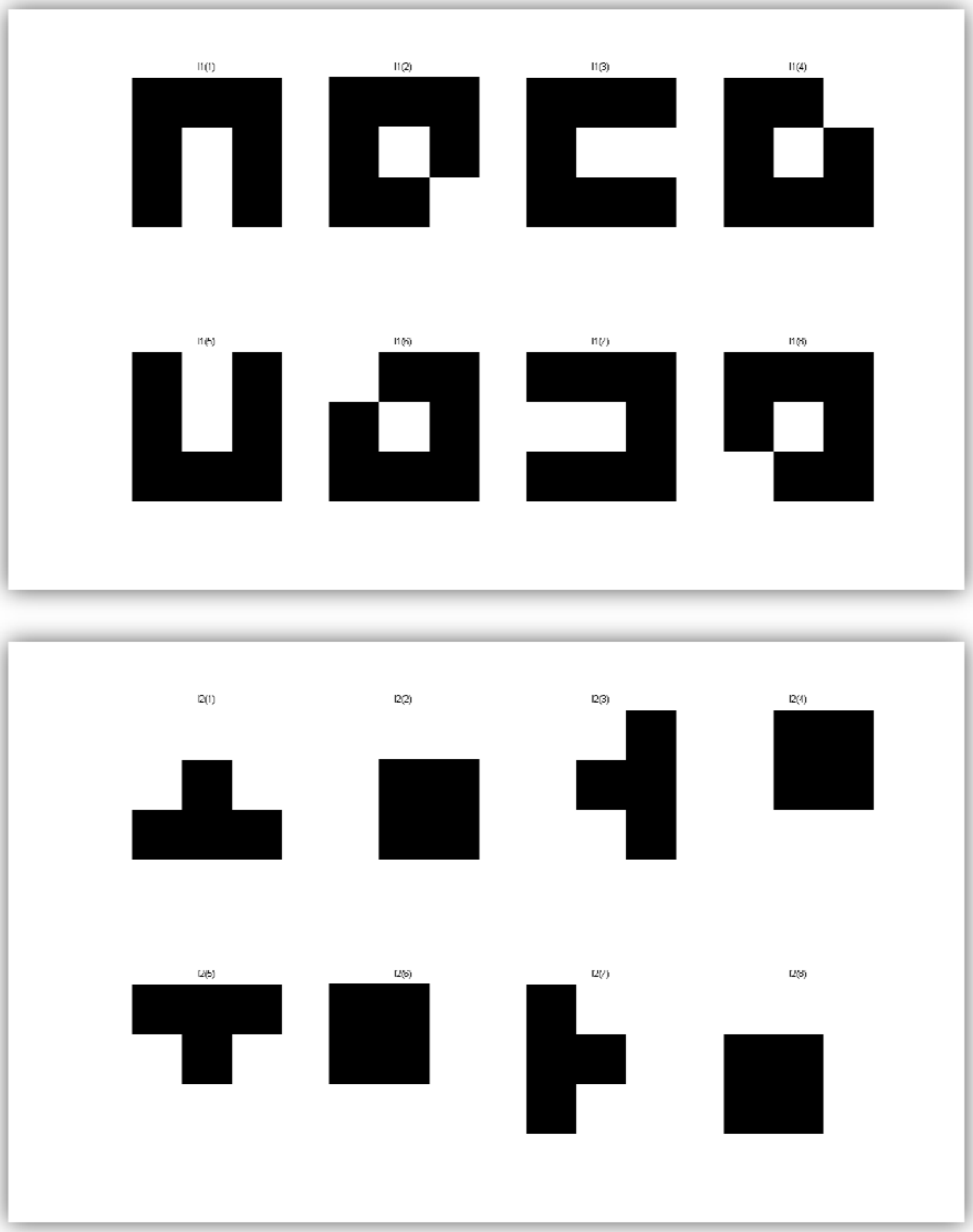


Fig. 3 2D Structuring Element I

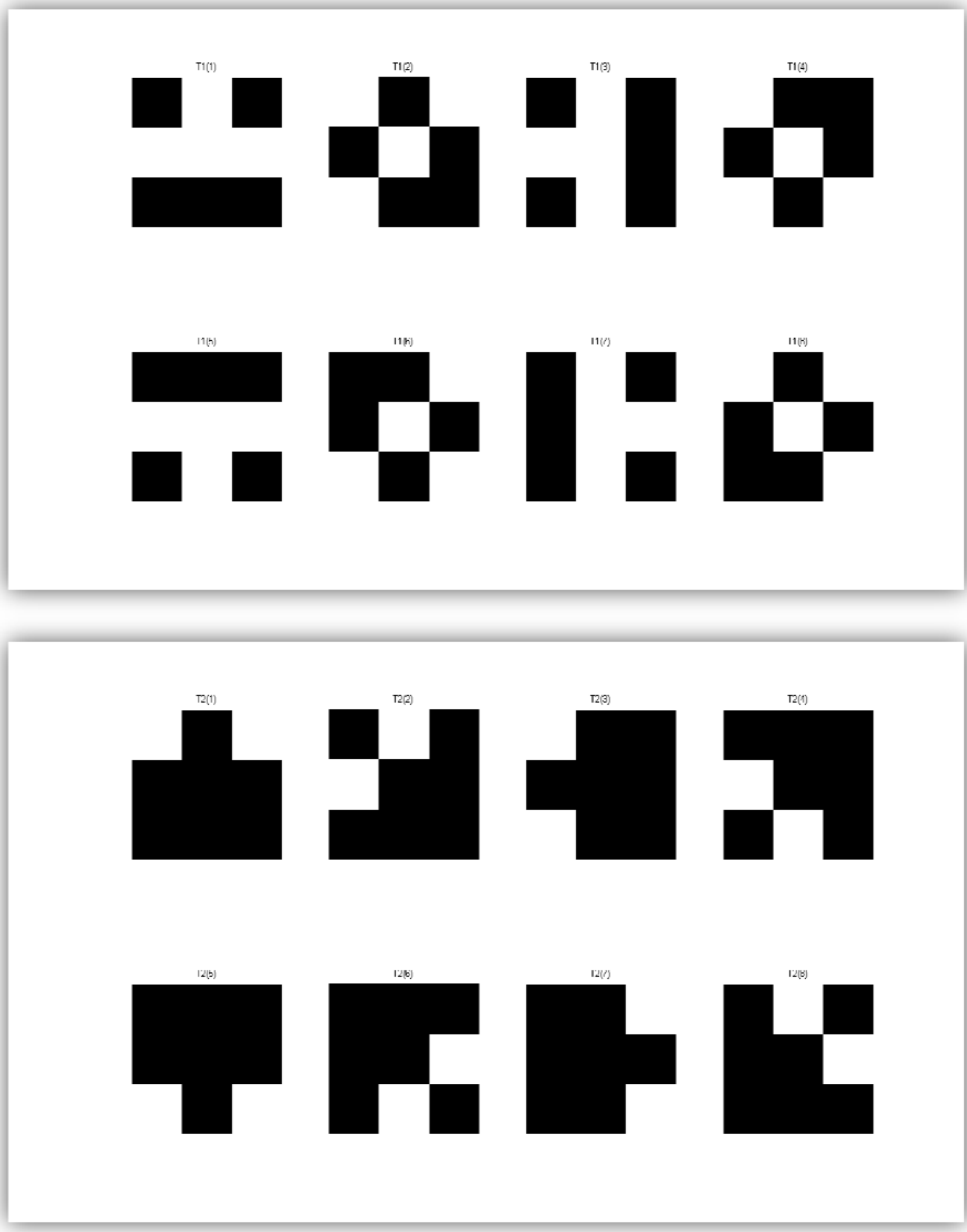
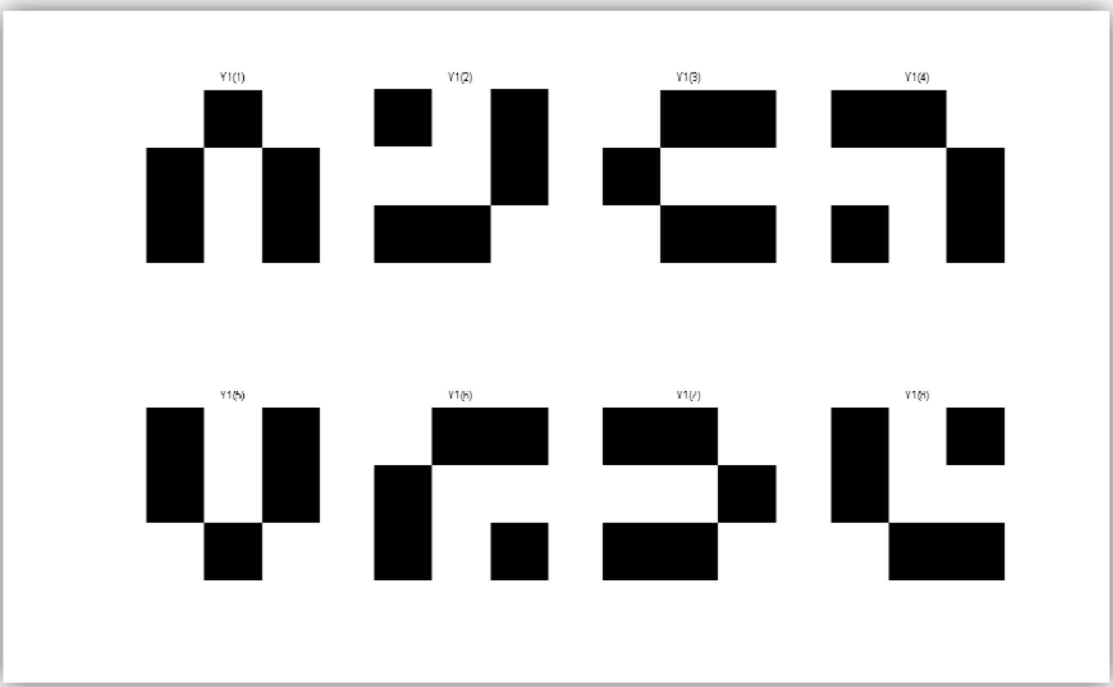


Fig. 4 2D Structuring Element T



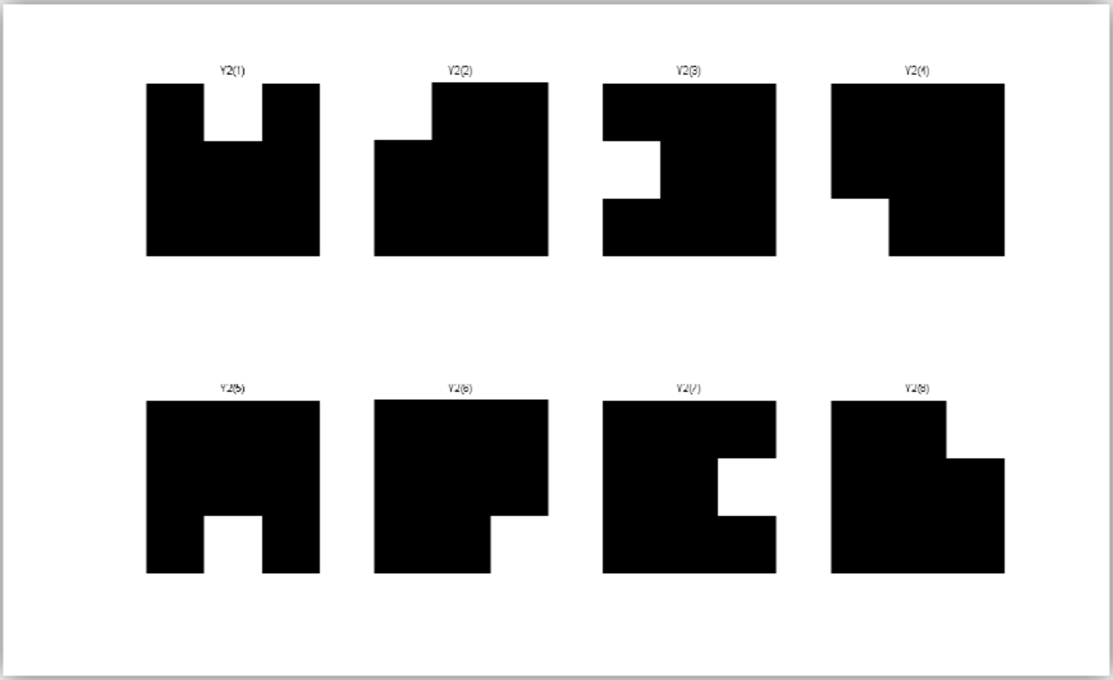


Fig. 4 2D Structuring Element Y

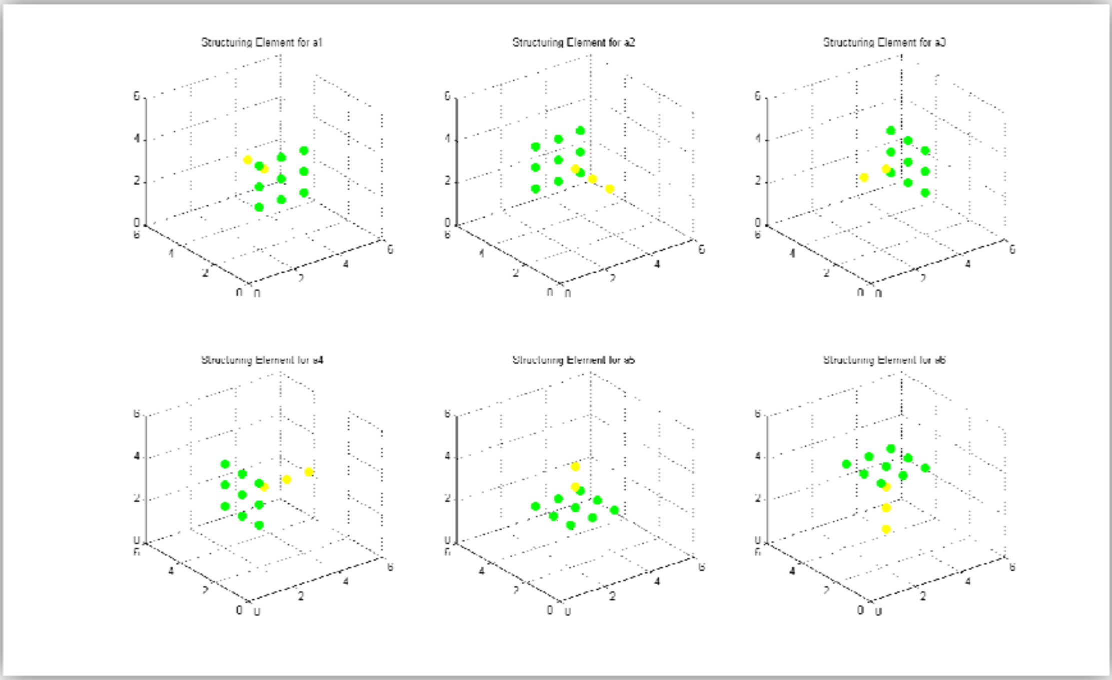


Fig. 5 3D Structuring Element A

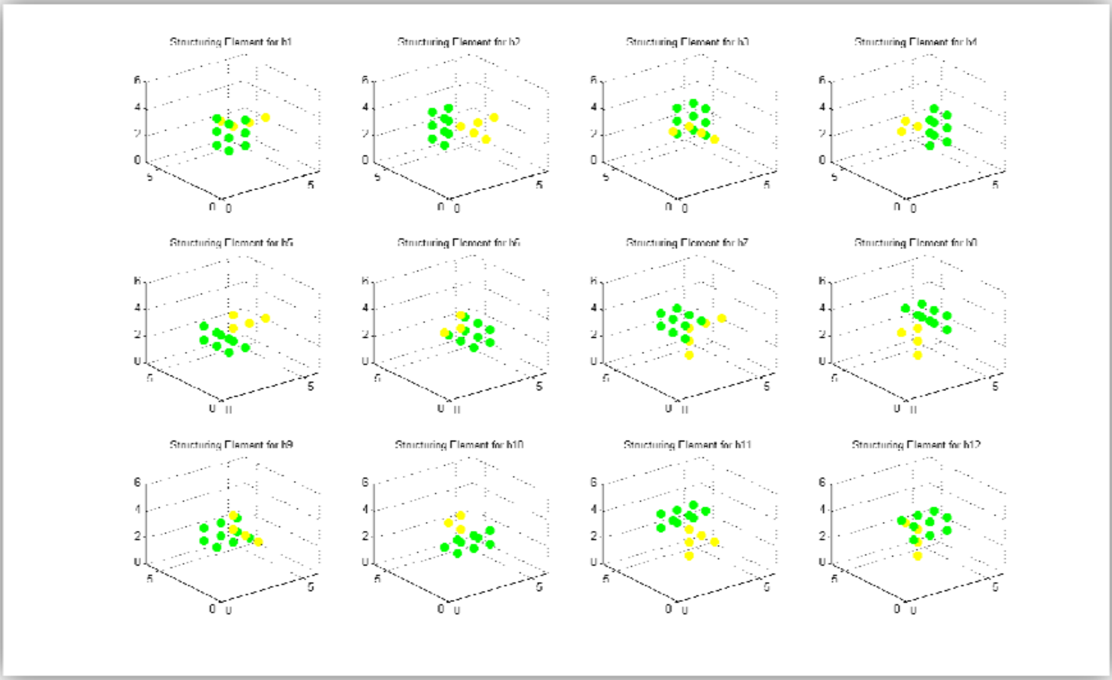


Fig. 5 3D Structuring Element B

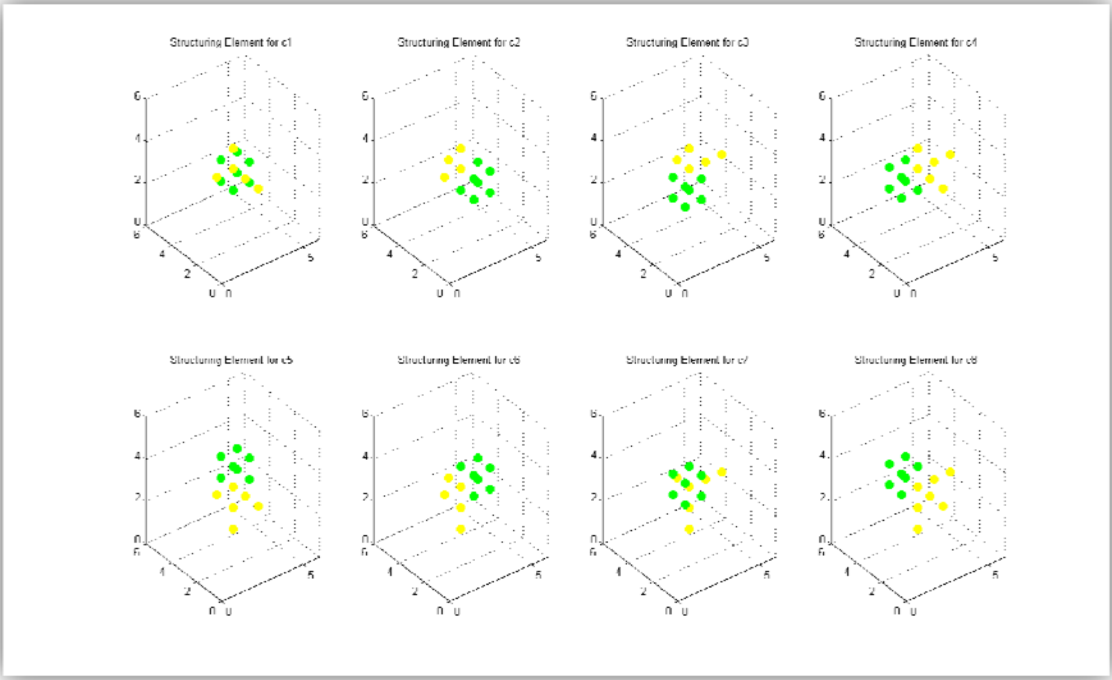


Fig. 5 3D Structuring Element C

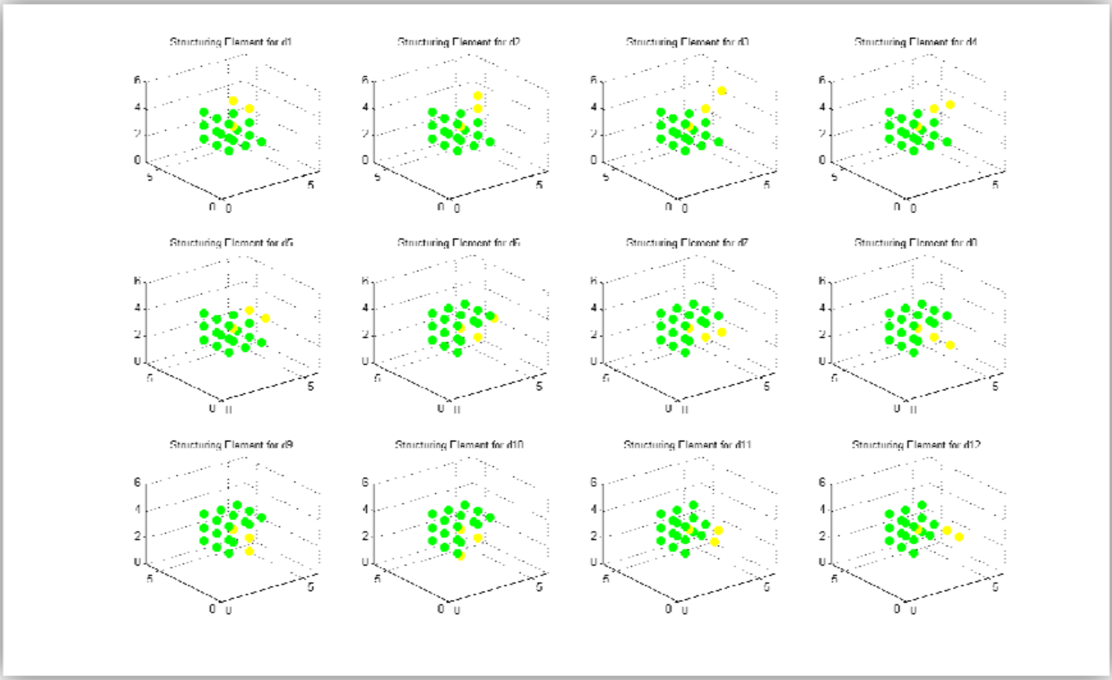


Fig. 5 3D Structuring Element D

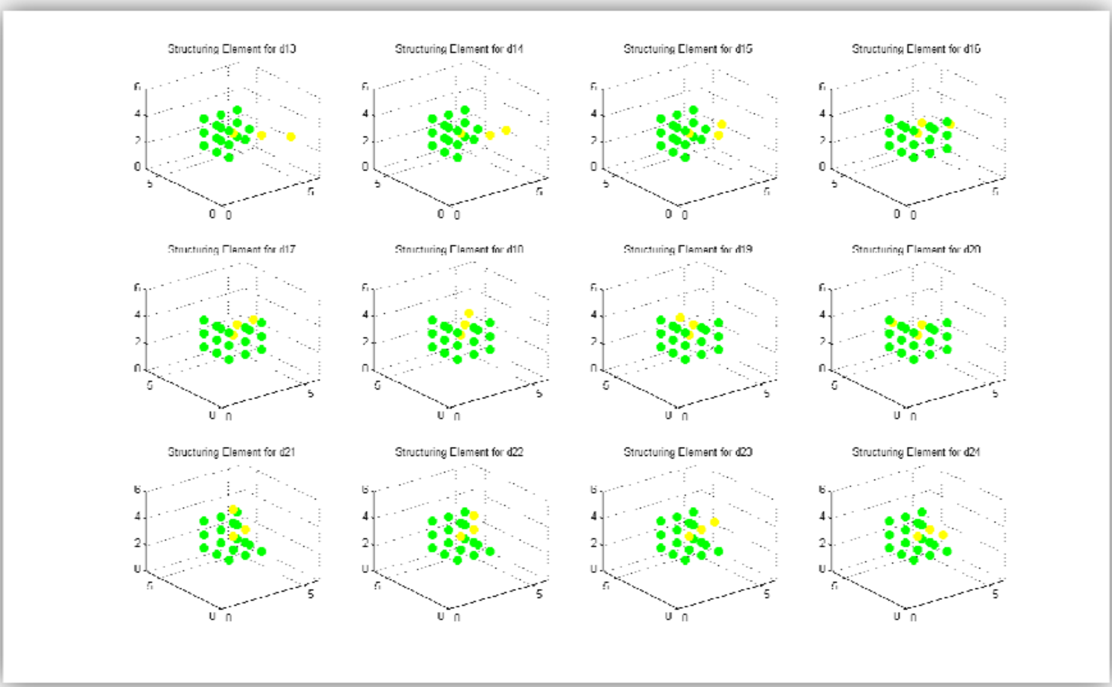


Fig. 5 3D Structuring Element D (cont'd)

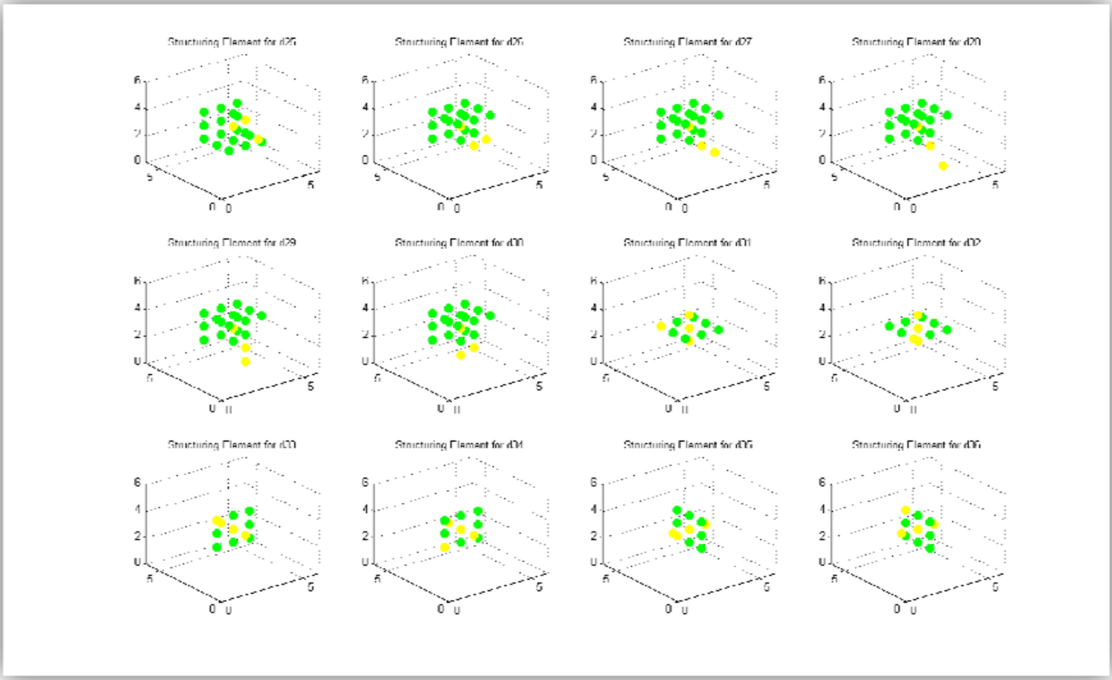


Fig. 5 3D Structuring Element D (cont'd)

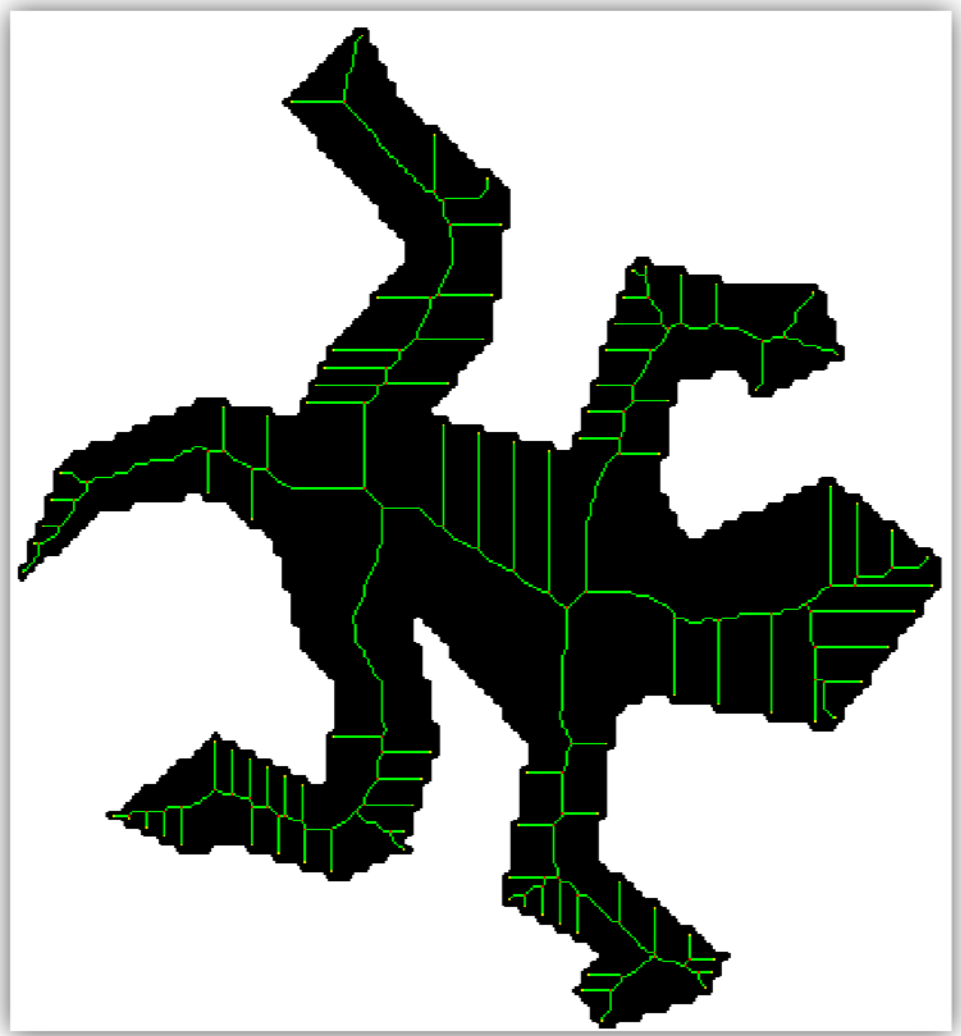
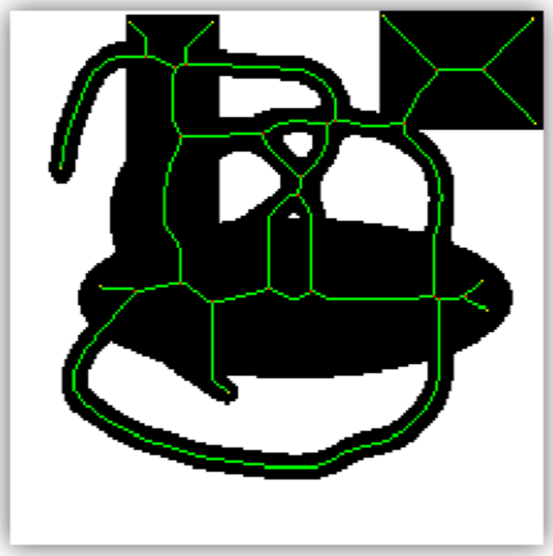



Fig. 7 2D skeletonization result (end points are in yellow and junctions are in red)

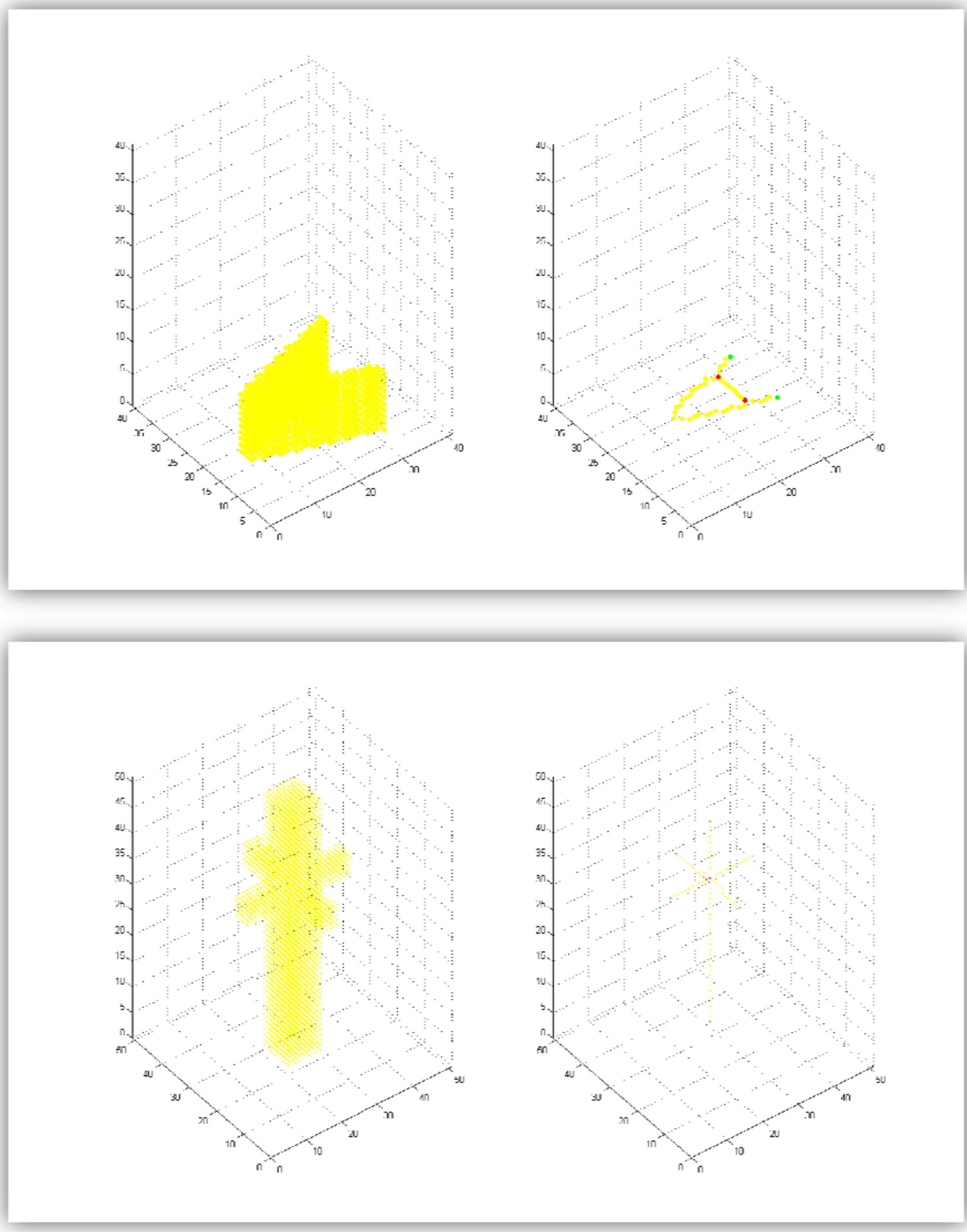


Fig. 7 3D skeletonization result (end points are in green and junction points are in red)

Reference:

[1] M. Sonka et al, Image Processing, Analysis, and Machine Vision (3rd ed, 2008)

[2] C. Ma and M. Sonka, A Fully Parallel 3D Thinning Algorithm and Its Applications, in *Computer Vision and Image Understanding*, 1996