An Approximation of the Maximal Inscribed Convex Set of a Digital Object

Gunilla Borgefors[1] and Robin Strand[2]

[1] Centre for Image Analysis, Swedish University of Agricultural Sciences,
[2] Centre for Image Analysis, Uppsala University,
Lägerhyddsvägen 3, SE-75237 Uppsala, Sweden
{gunilla, robin}@cb.uu.se

Abstract. In several application projects we have discovered the need of computing the maximal inscribed convex set of a digital shape. Here we present an algorithm for computing a reasonable approximation of this set, that can be used in both 2D and 3D. The main idea is to iteratively identify the deepest concavity and then remove it by cutting off as little as possible of the shape. We show results using both synthetic and real examples.

1 Introduction

Shape manipulation and description in 2D and 3D are of great importance when analyzing many kinds of digital images. Many shape descriptors have been introduced, to capture various aspects of shape. Global descriptors, e.g., bounding box and compactness (P2A), are generally useful, as they make comparisons between objects easier than a set of local descriptors. However, they can be difficult to compute, especially in 3D images.

A popular tool for shape description is the *convex hull* (CH), i.e. the minimal area convex set that includes an object [1]. A concept that should be equally useful, but is much harder to compute, is the *maximal area/volume convex subset* (MACS), i.e. the largest convex set enclosed in the object. Finding the MACS is sometimes called "the potato-peeling problem."

Very few algorithms for finding MACS in digital images have been published. Often people have been content to find the maximal enclosed disc/ball. This can easily be found by computing the distance transform of the object. The maximal distance value is the centre of the maximal inscribed disc/ball [2,3]. An overview of the state of the MACS art can be found in [4], where only 2D is discussed. A discrete algorithm for finding the convex subset that minimizes the Hausdorff distance between the object and the subset can be found in [5]. It generates a reasonable approximation of MACS. In the case where the object is bounded by n-sided polygon (all discrete objects can be easily transformed to such a polygon) a $O(n^7)$ solution is found in [6]. It can only be used for very small objects. In [4] an approximation of this solution is given for star-shaped n-sided polygons. The approximation presented here is much coarser than those mentioned above,

F. Roli and S. Vitulano (Eds.): ICIAP 2005, LNCS 3617, pp. 438–445, 2005.

but it can be used for arbitrary connected objects in both 2D and 3D, i.e., the objects can have holes (2D) or cavities and tunnels (3D).

The CH can be used to identify concavities of an object by computing the convex deficiency, i.e. by subtracting the object from its convex hull. Similarly, the MACS can be used to identify protrusions. It can also be found to identify the main "body" of the object. A typical application is to find the body of a cell and ignore (or identify) any thin protrusions. In our case, the need for computing the MACS came up in two applications: finding the main body of pores in paper and the main body of proteins consisting of several protruding subparts [7].

An object with a large convex deficiency can be rather close to being convex, close in the sense that the MACS includes most of the object area (think of a convex object with a few single thin protrusions or some holes near the edges). Our MACS approximation is intended for objects in 2D or 3D having a large MACS. In these cases it will usually produce good approximations. If the object is, e.g., snakelike, the approximation can become far from the true MACS, but it is hard to think of applications where identifying the MACS would be valuable for such objects.

Our algorithm is based on a simple idea: use an approximation of the CH to identify concavities in the object and then iteratively remove them, starting with the deepest one. The depths of the concavities are computed using a distance transform of the convex deficiency from the background, constrained by the object. The CH approximation and constrained distance transform can be computed by simple local operations. The deepest concavity is removed by cutting the object into two parts using a straight line. The cutting is done so as to remove as little of the object as possible. This cutting is not local, but only simple operations are needed. The remaining part of the object is "more convex" than the original object. The process is repeated until no concavities remain. The remaining part of the object is the approximation of the MACS.

2 Notations

Since it will always be clear from the text if 2D or 3D images are considered, both pixels (2D) and voxels (3D) are denoted *elements*. Let I be a binary image containing the two sets B and W, where B is the set of black elements (the object) and W is the set of white elements (the background).

In 2D, 4-connectivity is used for W and 8-connectivity is used for B. Two elements are 4-connected if they share an edge and 8-connected if they share an edge or a vertex. In 3D, 6-connectivity is used for W and 26-connectivity for B. Two elements are 6-connected if they share a face and 26-connected if they share a face, an edge, or a vertex. Pairwise connected elements are referred to as *neighbours*. A set of elements is connected if, for any two elements a and b in the set, there is a path between a and b consisting of neighbouring elements all belonging to the set.

We assume that B is a connected set of elements. If this is not the case, each connected component must handled separately in the algorithm.

A white element sharing a side (2D) or a face (3D) with at least one black element is referred to as a *border element*. A border element will be defined as being in a local concavity by the configuration of its neighbouring black elements.

To find a globally deepest concavity, we compute the convex hull (CH). It is approximated by a *covering polygon* (2D) or a *covering polyhedron* (3D). The term *envelope* will be used to denote either covering polygon or covering polyhedron. The elements in the convex deficiency, i.e., the elements in the envelope that are not object (black) will be marked as gray, G.

The set of element(s) that are found to be the deepest concavity is denoted C. These corresponds to the reflex points in [4].

The end result of the algorithm is an approximation of the largest convex set enclosed in the object, i.e., the maximal area/volume convex subset (MACS).

3 Computing the Envelope

There are many algorithms for computing the CH of a digital object. We will use the one in [8], that approximates the CH by a covering polygon (polyhedron). It is simple and uses only local operations. The envelope is obtained by iteratively filling local concavities. The resulting shape is convex (coarse approximation) or very nearly convex (better approximation where some very shallow concavities may remain). Brief descriptions of the [8] algorithm in 2D and 3D are given in the subsections of this section.

The number of possible orientations of the sides (faces) of the envelope depends on the size of the neighbourhood from which curvature information is derived: the larger neighbourhood, the larger number of sides (faces) of the envelope. For example, in 2D, if curvature information is derived from the 3×3 neighbourhood of a border element, eight orientations are possible for the sides of the envelope, so what is actually computed is the minimal covering octagon. In 3D, using the $3 \times 3 \times 3$-neighbourhood will result in the smallest covering rhombicubeoctahedron (an Archimedian solid with 26 faces). If curvature information is gathered from a larger neighbourhood, many more orientations are possible (in fact, infinitely many, but still far from *all* orientations). The extension to larger neighbourhoods is achieved by a labeling procedure using 3×3-neighbourhoods ($3 \times 3 \times 3$-neighbourhoods).

After the computation of the envelope the image will be ternary: object elements (black), B, elements in the envelope that are not object elements, i.e. the convex deficiency, (gray), G, and background elements (white), W.

Computing the 2D Convex Hull. The 2D approximation of the convex hull is computed as follows. We give the two versions used in this paper.

3×3 Case

1. Iteratively change border elements with more than three non-white neighbours to gray.

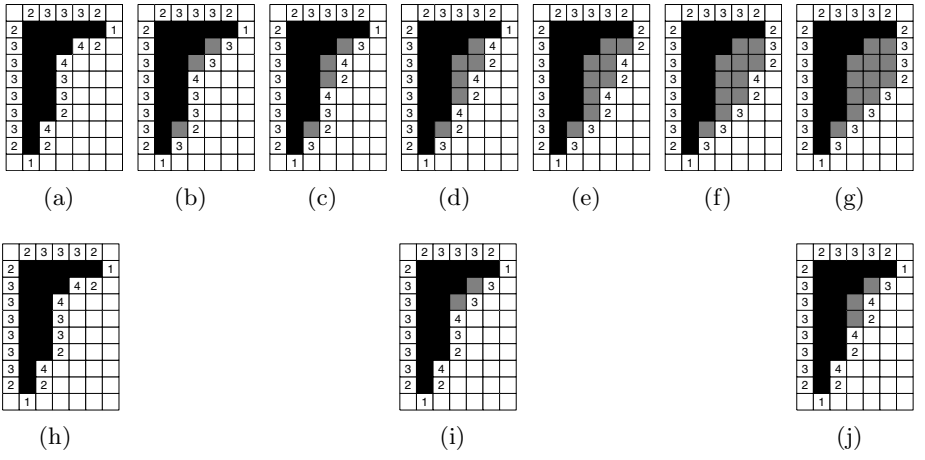

Fig. 1. Computing the covering polygon of an object using the 3×3-rule, (a)-(g); and using the 7×7-rule (h)-(j)

7×7 Case

1. Each border element is labeled with the number of its non-white neighbours.
2. Border elements labeled 3 with no neighbour labeled less than 3 are marked.
3. Change the following border elements to gray: elements labeled more than 4; elements labeled 4 with at least one neighbour labeled more than 3, or two neighbours labeled 3, or one marked neighbour.
4. Repeat from (1) until no changes occur.

A small example of how these two algorithms work is found in Figure 1.

Computing the 3D Convex Hull. The 3D approximation of the convex hull is computed as follows. We give the two versions used in this paper.

$3 \times 3 \times 3$ Case

1. For each border element, its non-white face and edge neighbours in the x-, y-, and z-planes are counted, respectively. If the maximum of these three sums is larger than 3, the element is marked.
2. Marked elements are changed to gray.
3. Repeat from (1) until no changes occur.

$5 \times 5 \times 5$ Case

1. For each border element, its non-white face and edge neighbours in the x-, y-, and z-planes are counted, respectively. These three sums, denoted Σ_x, Σ_y, and Σ_z, are stored as labels of the border element.

2. Change the following border elements to gray: Elements with at least one $\Sigma_k > 4$, ($k \in \{x, y, z\}$); elements with one $\Sigma_k = 4$ and having, in the same k-plane, at least one neighbour with $\Sigma_k > 2$.
3. Repeat from (1) until no changes occur.

4 Finding the Deepest Concavity

Here we describe how to find the places where the object must be cut to make it convex. To find the deepest concavity C, we use a constrained distance transform. In a distance transform, each element in one set is labeled with the distance to the closest element in another set. In a constrained distance transform [9], a third set acts as a barrier for the distance propagation. In our case, we will compute the distance transform of the convex deficiency, G, from the background, W, with the object, B, as the constraint.

We use *weighted* distance transforms. These are simple to compute and use, while being reasonably rotation independent distance, see, e.g., [2] for 2D and [3] for 3D. The distance between two elements is defined by the shortest path between them, using only steps between neighbours and weighting the steps according to the neighbouring relation. In 2D, we use the weights 3 and 4 for edge- and vertex-neighbours, respectively; and in 3D we use the weights 3, 4, and 5 for face-, edge- and vertex-neighbours, respectively. The actual computation is done by the efficient chamfer algorithm, see [2,3], again. The number of necessary scans depends on the object configuration.

The constrained distance transform of the gray pixels in the small example from Figure 1 is found in Figure 2. The deepest concavity C is simply the element with the largest distance label, in Figure 2(a) 8 and in Figure 2(b) 3.

Often neighbouring elements gets the same distance label, therefore connected components (using the background connectivity, as the gray elements are part of the original background) with the same distance label are considered together, so C consists of more than one element. If 6 were the maximum label in Figure 2(a), C would consist of the two elements labeled 6. If there is more than one connected component with elements having maximal distance label, as in Figure 2(b), anyone of them can be arbitrarily chosen as C.

Fig. 2. Constrained distance transforms of the gray elements in Figure 1(g),(j): each gray element is labeled with the shortest distance to a white element

If there are holes (cavities) in the object they should be treated before the concavities. Each hole is treated as a connected set. Compute the weighted distance transform of the object *and* the holes from the non-black elements outside the object. The depth of each hole is the maximum distance value of any of the elements in the hole. The deepest hole (cavity) becomes C.

In 3D, there can also be tunnels. They do not need any special treatment, as they will become part of G when computing the envelope and will thus get distance values. In this case, C may not be border elements, as the elements farthest from the background may be in the interior of a tunnel. However, when this C is cut off, the tunnel will be broken and next time C will be a "normal" concavity.

5 Removing a Concavity

The (set of) deepest concavity element(s), C, has been found. It is removed by cutting the object into two (or more) parts, using straight lines (planes) through C. One part will be discarded and the other(s) will be kept for further processing. The aim is to remove as little of the object as possible, therefore cuts are made in all of a fixed number of directions and the one leaving the largest number of elements is chosen. In 2D we use 8 directions (angles $n\pi/4, n = 0, \ldots, 7$), and in 3D 18 directions (defined later). This is in agreement with envelopes based on the 3×3-rule ($3 \times 3 \times 3$-rule.)

In 2D, the line that separates the object and C must satisfy one of the following equations $y = m$, $y = x + m$, $x = m$, or $y = -x + m$. The values of m are computed such that one connected component of the object is on one side of the line and C is on the other. In Figure 2(a), C is the element with distance label 8. In Figure 3 all eight possible cuts removing C are shown.

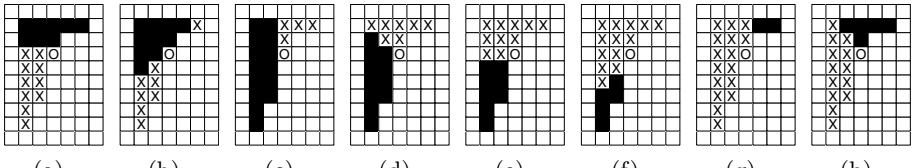

Fig. 3. The cuts induced by the eight directions applied to the running example. The element marked "O" is the deepest concavity. The elements removed by each cut are marked "X", while black elements are the ones remaining.

In some cases, the cut splits the object into two connected components. In this case, the size of part on the "C-side" measures how much is removed. In other cases, where the object "protrudes" on both sides of C the line splits the object into three (or more) connected components, see Figure 3(b). In these cases, only the smallest of the components on the C-side is removed and its size is the measure of the cut. The protrusion on the other side of C will probably

be removed at a later stage, but not necessarily *all* of it, so removing only one component can increase the size of the final MACS.

After cuts have been made in all eight directions, the cut removing the smallest connected component is chosen to remove C. In Figure 3 this is the cut in (b); only 1 element is eliminated.

In 3D, the cutting plane must satisfy one of the nine equations $x = m$, $y = m$, $z = m$, $y = x+m$, $y = -x+m$, $z = x+m$, $z = -x+m$, $z = y+m$, or $z = -y+m$. The number of different planes that must be tested becomes 18. The cutting is done exactly as in the 2D case – for each direction m is determined and the smallest connected component after the cut is considered for removal. In the end the best of the 18 possible cuts is chosen to remove C.

6 Finding the Maximal Convex Subset

By sequentially repeating the algorithms described in Section 3-5, MACS is computed. In each iteration, first the envelope is computed using the 3×3-rule ($3 \times 3 \times 3$-rule) and the deepest concavity C is found. If the distance value of $C > 3$, C is removed by making the best cut.

If, however, the distance value of $C = 3$, the object is possibly already convex. Remember that the envelopes using the $3 \times 3(\times 3)$ neighbourhoods overestimate the true CH, because of the limited number of directions of edges (faces). Thus, if $C = 3$ we recompute the envelope, using the more accurate 7×7-rule ($5 \times 5 \times 5$-rule). If any concavities do remain, using these rules, they must be removed in the usual way.

Note that, the envelope should not be computed using the larger neighbourhoods. Experimentation has shown that, in most cases, using more directions when computing the envelope than when cutting will remove more than is necessary of the object. However, using the larger neighbourhoods at the very end can "save" many object elements in the MACS, as each "pseudo-concavity" can lead to the cutting off of many elements.

7 Examples and Conclusion

In Figure 4 there are a number of examples of how the algorithm works, both synthetic and real. In Figure 4(a) we have small fish with an eye-hole and in (b) its MACS. We mentioned cells with thin extensions as suitable for the algorithm. Figure 4(c) shows a nerve cell with axons from a stem cell project at our lab. The cell body is found by computing the MACS, Figure 4(d). Figure 4(e) shows a synthetic 3D object, a "potplant" and (f) its MACS. Not all of the pot remains, as the number of cutting plane directions is limited, but most of it is there. Figure 4(g) shows a pore (void) segmented from a 3D image of a piece of milk cartoon. The pore shape is very irregular with deep concavities, more than is apparent from the image. Finding the body of such pores was one of the motivations of this work. The MACS of the pore is shown in Figure 4(h).

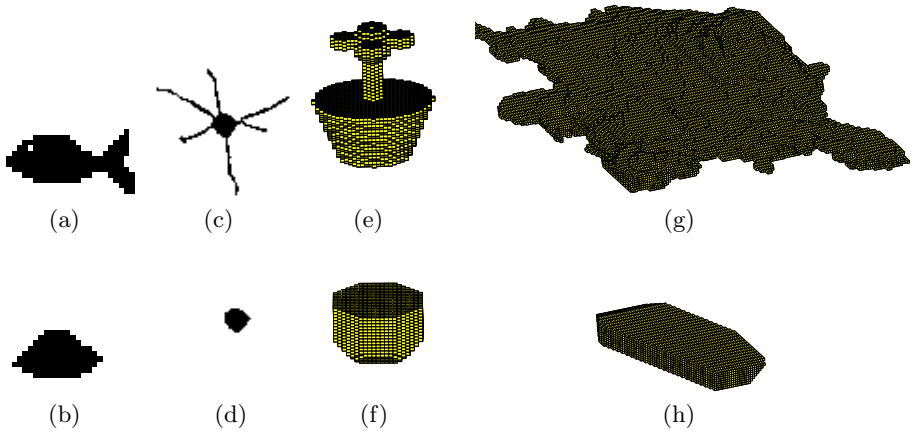

Fig. 4. Examples of maximal inscribed convex sets in 2D and 3D. See text

Even if the proposed algorithm is not perfect, it is shown to be a useful tool for finding an approximation of the largest inscribed convex set. It is simple to implement and uses mostly local operations, so it is reasonably fast even in 3D.

References

1. Preparata, F.P., Shamos, M.I.: Computational Geometry an Introduction. Springer-Verlag, New York (1985)
2. Borgefors, G.: Distance transformations in digital images. Computer Vision, Graphics, and Image Processing **34** (1986) 344–371
3. Borgefors, G.: On digital distance transforms in three dimensions. Computer Vision and Image Understanding **64** (1996) 368–376
4. Chassery, J.M., Coeurjolly, D.: Optimal shape and inclusion. In Ronse, C., Najman, L., Decencière, E., eds.: Mathematical Morphology: 40 Years On. Computational Imaging and Vision, Springer, Dordrecht (2005) 229–248
5. Chassery, J.M.: Discrete and computational geometry approaches applied to a problem of figure approximation. In Pietikäinen, M., Röning, J., eds.: Proc. 6th Scandinavian Conference on Image Analysis, Oulo, Finland (1989) 856–859
6. Chang, J.S., Yap, C.K.: A polynomial solution for the potato-peeling problem. Discrete & Computational Geometry **1** (1986) 155–182
7. Sintorn, I.M.: Segmentations methods and shape descriptions in digital images – applications in 2D and 3D microscopy. PhD thesis, Swedish University of Agricultural Sciences (2005)
8. Borgefors, G., Sanniti di Baja, G.: Analyzing nonconvex 2D and 3D patterns. Computer Vision and Image Understanding **63** (1996) 145–157
9. Piper, J., Granum, E.: Computing distance transformations in convex and non-convex domains. Pattern Recognition **20** (1987) 599–615