The University of Kansas

Information and Telecommunication Technology Center

A Technical Report of the
Intelligent Systems and Information Management Laboratory

Experiments and Algorithmic Analysis of the Restricted Growing

LeenKiat Soh
Costas Tsatsoulis

ITTC-FY99-TR-11810-05

May 1999

Project Sponsor:

Naval Research Laboratory

Experiments and Algorithmic Analysis of the Restricted Growing Concept

Leen-Kiat Soh and Costas Tsatsoulis

Abstract

This report defines a new concept called the Restricted Growing Concept (RGC) for object separation and provides an algorithmic analysis of its implementations. Our concept decomposes the problem of object separation into two stages. First, we achieve separation by *shrinking* the objects to their cores while keeping track of their originals as masks. Then we *grow* the core within the masks obeying the guidelines of a restricted growing algorithm. Hence, we achieve both goals better than conventional object separation techniques. In this report, we investigate various issues via different implementations of core and mask images. We compare morphological operators to probabilistic labeling—a technique we establish to obtain cores and masks based on neighborhood confidence—in the remote sensing domain, particularly for Synthetic Aperture Radar (SAR) sea ice imagery. We review a skeletonization algorithm and demonstrate how it can be converted to grow objects instead of thinning them. Further, we describe a dozen sequential algorithms differing in scanning order and pixel selection, even employing distance transform, blob coloring, and jumps. We evaluate the algorithms in visual judgment of the results and computational speeds. In conclusion, we present general observations and offer design recommendations for the concept.

1 Introduction

When two gray level objects touch with shared boundaries, it makes shape analysis and recognition difficult. For example, in industrial vision applications [8], touching objects hinder the recognition of the object's shape and complicate the task of defect inspection. In aerial image and terrain analysis, where the objective is to identify items such as airplanes on an airfield [21], failure to obtain isolated objects creates misunderstanding in the resulting image context. In shape analysis where geometric descriptors are computed for each object, treating a conglomerate of touching objects as one object leads to incorrect measurements [11]. Hence, separating objects is important in object-based image analysis.

In object separation, our objectives are (1) to achieve object separation, and (2) to preserve (or approximate as closely as possible) the object's original shape and size. The fact that the two goals cannot be optimized simultaneously further adds to the complexity of this task. This dilemma is illustrated in Fig. 1.

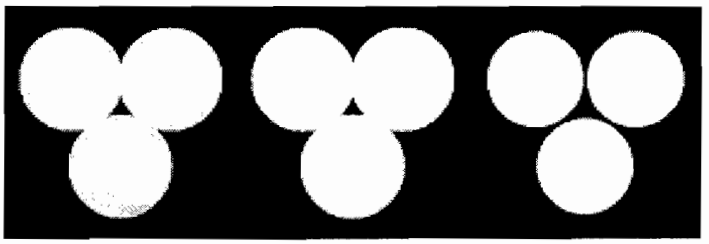

Fig. 1 The conflict between separation and preservation in object separation. The raw image consists of three circles. The image in the middle shows the segmentation result of using a low threshold value. The right image shows the segmentation result of using a high threshold value.

The figure shows an image with three touching objects, all having pixels with lower intensity values towards the boundary. One simple way of obtaining the objects would be thresholding. Using a low threshold, we would be able to extract the objects with complete shape and size preservation; however, the objects might, as a result, be connected. Using a high threshold, on the other hand, we would be able to separate the objects, but would lose the boundary and some layers of interior object pixels. This tradeoff between separation and preservation of size and shape is inherent in all object separation algorithms. To address this problem, we have designed an object separation technique based on a concept called the Restricted Growing Concept (RGC). This concept divides the task and the accomplishments of the two objectives into two tiers. First, it achieves the separation. Second, it re-establishes the sizes and shapes of the objects lost or

distorted during the separation process by performing restricted growing. In our research, we use reversed skeletonization to grow objects. In mathematical morphology [27][28], skeletonization (or thinning) is used to obtain the skeleton of a region while preserving its connectivity. We reverse that process: instead of deleting pixels from the region, we add pixels to the region, and instead of preserving connectivity, we preserve separation. This novel use of skeletonization is the key to achieve separation and size and shape preservation sequentially in the framework of RGC.

In this report, we present the restricted growing concept and discuss various issues involved in achieving the objectives of object separation. We probe into the alternatives in obtaining cores and masks of the growing process. We address the issues of preserving details through different designs of masks. We investigate the use of morphological reconstruction and h-domes in extracting cores, and compare the differences between the performance of the morphological operators in synthetic and remotely sensed images. After obtaining the appropriate designs for obtaining cores and masks for our Synthetic Aperture Radar (SAR) sea ice domain, we adapt a skeletonization algorithm to guide the growth of object pixels in the image. Given the algorithm, we further present twelve different designs and examine their weaknesses and advantages when applied to remotely sensed imagery.

Object extraction is important in SAR sea ice image analysis, in which individual ice floe identification and their outlines are important for examining both large- and small-scale processes in sea ice such as classification [11] and floe size measurements [9][16][26], and feature analysis [7]. We have applied our algorithm to SAR sea ice analysis for segmentation and floe size distribution [29][31][33].

2 Background

In morphology [27][28], opening usually eliminates thin protrusions and breaks narrow bridges between objects. On the other hand, closing usually fills small holes, strengthens linear structures, and eliminates gaps in the contour of an object. However, once an object has been closed or opened, further applications of closing or opening using the same structuring element will not modify the object—connections between objects that were not broken will not be broken at a later iteration. Usually, iterative erosion alone is used to achieve object separation.

[2] used a method called erosion-propagation (EP) algorithm coupled with clustering about principal curves [10] to identify objects in satellite images. The EP algorithm morphologically

erodes a pixel from object to non-object if any of its 8-neighboring pixels is non-object. As the edge of an object is gradually eroded during the iterative process, the locations of the edge pixels are propagated toward the interior of the object. This information is later used in the clustering about principal curves to determine whether merging of objects is necessary to remedy the effects of excessive erosion. This object separation technique suffers from several disadvantages: (1) the number of iterations required to achieve separation has to be determined manually for each image, (2) objects smaller than $(2i + 1) \times (2i + 1)$ pixels, where i is the number of iterations, will be eliminated, and (3) objects do not preserve their original size.

In another approach [5], a tagging algorithm was used to separate objects with weak connections. At each growing iteration, every object pixel that is untagged and an 8-neighbor to the existing grown object pixels is a candidate. A candidate is then tagged as belonging to a feature if (1) it has at least two 8-neighbor pixels that are either the candidates of previous iteration or pixels of the feature, and (2) the two pixels must be 4-neighbors to each other. The process grows each object individually; unlike the EP algorithm that applies the same number of iterations to all objects non-discriminatingly, this tagging algorithm is thus able to retain small objects. However, the authors' implementation separated only regions connected by corners or by one-pixel bridges, rendering it incapable of achieving separation when stronger connections occur.

[30] proposed an interesting approach to separate objects. With the assumption that touching objects create sharp turns or corners in the boundaries where the objects meet, the authors designed a set of constraints to decide whether two corners should be connected. The technique first locates a chain of corners on the boundary and then applies the constraints (which were based on geometric heuristics and semantic properties of the boundaries), and separation among objects is finally created by linking corners of different boundaries. This approach is able to achieve good separation and complete preservation. However, since it relies on the definition of a corner, it is noise-sensitive and limited to domains in which the aforementioned assumption holds.

In this report, we present a technique that is robust in identifying individual objects in remotely sensed images where speckle noise effects are not negligible. Our technique (1) does not require user interactions once the initial object-background segmentation is obtained, (2) preserves the original sizes and shapes of the features in the image, and (3) eliminates small

objects only minimally (as a result of morphological cleaning to remove noise effects), and thus addresses the weaknesses associated with other object separation techniques.

3 The Restricted Growing Concept

To achieve separation and preservation, we design a concept called the Restricted Growing Concept (RGC). The main idea of this concept is to decompose the object separation problem into two modules, each dedicated to achieve one of the objectives as outlined in the sections above. The first objective is separation. This is accomplished by *shrinking* objects such that each object is separated from its touching neighbors. The second objective is preservation of size and shape. This is accomplished by growing the shrunk objects to restore the size and shape. However, the two objectives are contradictory, as illustrated in Fig. 1. To ensure that the separation that has been established (after the first stage) is not disturbed, our growing process is *restricted*. This gives rise to our RGC.

Now we define the components of RGC. A *core object* is a version of the original object such that its linkages to neighboring objects are disconnected, satisfying our first objective of object separation. Such an object is reduced in size, but it usually captures the general shape of its original version. An image with core objects is thus a *core image*. A *mask object* is a version of the original object such that the original size of the object is preserved. Mask objects are usually interconnected and could encompass one or more core objects. An image with mask objects, which will serve as constraints on the growing of their enclosed core objects, is a *mask image*. Note that a less constrained version of the definition is to allow the mask object to be a close approximation of the original object. Finally, a *restricted growing algorithm* grows from a core object within the boundary of its corresponding mask object while preserving the object's separation from its neighbors. This definition implies that the growing process stops either when the boundary of the object has been reached or when further growing will damage the object's separation from its neighbors. Thus, conventional region growing [37] or morphological dilation schemes are not restricted growing algorithms.

Fig. 2 demonstrates an execution of RGC. The original image in this example consists of touching sea ice floes (objects) observed in satellite imagery of the Arctic. Our task is to identify each individual object. First, we generate the mask image, which is a binary segmentation of the original image that identifies the object and background pixels.

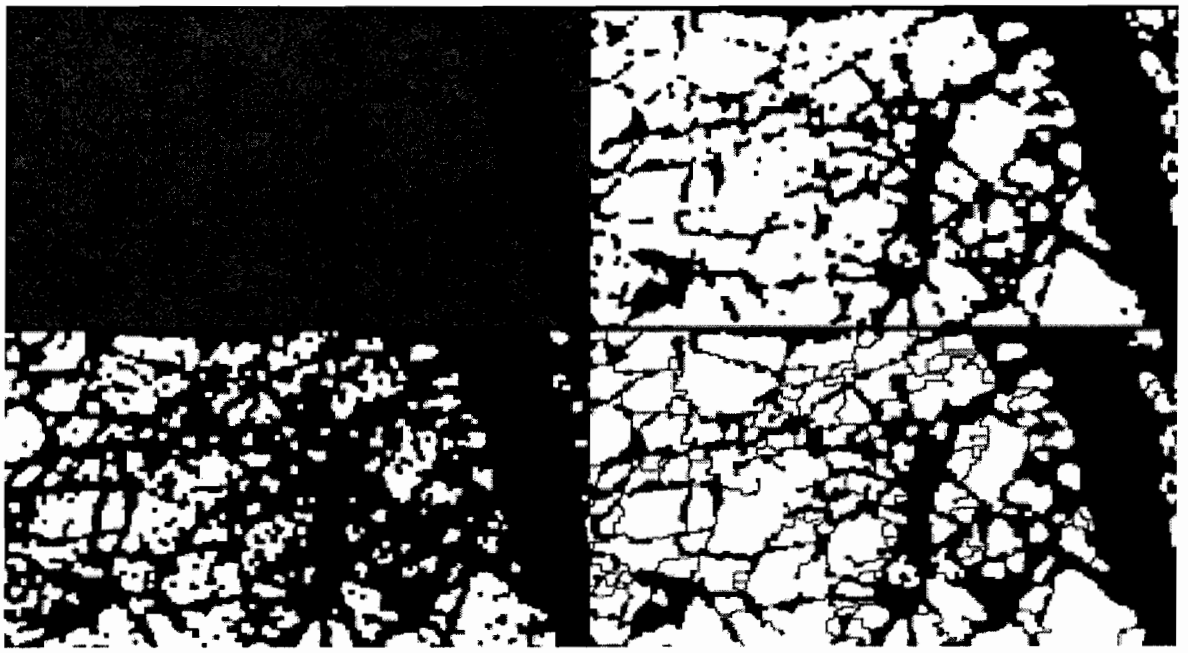

Fig. 2 An execution of RGC: The upper left image shows an aircraft STAR-2 SAR sea ice imagery (obtained from Dr. J. Comiso of NASA Goddard SFC). The upper right image is the mask image. The lower left image is the core image. The lower right image is the restricted growing result.

As noted from Fig. 2, objects are now observed as a network of interconnected entities, making individual object analysis impossible. Second, we generate the core image. Similar to the mask image, the core image is a binary version of the original image. However, the objects are now separated from their touching neighbors, isolated through a shrinking process. Comparing the mask and core images, one observes that mask objects always enclose one of more core objects in corresponding areas. During the restricted growing process, a core object will not be grown outside of the enclosure. In addition, a pixel will not be grown from non-object to object class when such growth will destroy the existing separation status of the area.

4 Generating Mask Image

The objective of having a mask object is to contain the growth of the core objects that it encloses. A mask object should retain its original shape and size. In practice, a mask image is binary, with its pixels divided into object and background pixels. Our implementation basis for the mask image is gray level thresholding, which is a general segmentation approach in image processing. In the examples and illustrations of this report, all mask images have been obtained using global thresholding. However, in actual SAR sea ice applications, we have employed dynamic local thresholding object separation [33].

4.1 Probabilistic Labeling

We use probabilistic labeling to analyze the neighborhood of a pixel to decide whether the pixel is an object pixel in the mask image. The concept of our probabilistic labeling is similar to that of relaxation [24] which is based on incremental improvement of the confidence of a pixel being an object (or a non-object) pixel. Conventionally, a relaxation technique first assigns to each pixel an initial classification or labeling, then computes the confidence of each pixel with its label, relaxes the current classification either stochastically or deterministically, and iterates until all pixels converge to a stable labeling state. Our design of probabilistic labeling uses threshold slices as the changing environment on which the assessment of the confidence of a pixel's being an object is based. A threshold slice, S_t, is obtained by thresholding the image at intensity t. The higher the intensity, the stricter the environment that its threshold slice imposes on a pixel's neighborhood, effectively lowering the probability of a pixel surviving as an object pixel as its neighbors are eroded gradually. By observing the neighborhood of a pixel in various environments, we determine whether it deserves to be labeled as an object pixel.

The set of environments or threshold slices, $\Omega(T, I, N)$, is a three-tuple, where T is the starting threshold, I is the interval between successive threshold slices, and N is the number of threshold slices. In our implementation:

$$\Omega_{mask} = \Omega(t(i,j),2,3) = \{S_{t(i,j)}, S_{t(i,j)+2}, S_{t(i,j)+4}\} \qquad (1)$$

where $t(i,j)$ is the threshold computed at pixel (i,j) during the segmentation process (either global thresholding or dynamic local thresholding), $I_{mask} = 2$, and $N_{mask} = 3$. To obtain the accumulated confidence of a pixel at (i,j) being an object pixel, we first compute the confidence of the pixel being an object at threshold slice S_t as the ratio of the neighbors in the pixel's 3×3 neighborhood that have survived the slicing of S_t. This confidence we denote as $c_{S_t}[(i,j) = object]$. Next, we sum all confidence values for each pixel as follows.

$$C_{mask}(i,j) = \sum_{S_t \in \Omega_{mask}} c_{S_t}[(i,j) = object] \qquad (2)$$

Finally, to label each pixel in the mask image, we compare $C_{mask}(i,j)$ to a pre-specified threshold, T_{mask}: if the $C_{mask}(i,j)$ of a pixel at (i,j) is greater than or equal to T_{mask}, then that pixel is an object pixel in the mask image. T_{mask} has been experimentally determined and set as

0.75 for Synthetic Aperture Radar (SAR) sea ice images, and it has been used as a constant in all examples presented in this report. Note that the above algorithm has three important parameters, i.e., T_{mask}, I_{mask}, and N_{mask}, that system designers can adjust to accommodate their specific needs and domains of applications during the development phase of their object separation software. Once the parameters are determined, the technique runs without human interaction or supervision.

4.2 Morphological Cleaning

Since remotely sensed images are usually corrupted with speckle noise, we apply morphological cleaning to eliminate the noise effects. This cleaning operation comprises a sequential execution of closing and opening. Note that closing is a combination of first erosion and then dilation on an image while opening is first dilation and then erosion. Opening generally trims off peninsulas, break bridges, and solidifies channels. Closing generally closes inlets, absorbs islands, and fills lakes. As a result, visually, the features after morphological cleaning are better in shape and content definition, providing more definitive boundary.

4.3 Different Mask Images

The definition of Eq. (2) destroys a tiny, unrecoverable portion of the original size and shape because pixels at a region's boundary may be lost due to lack of confidence. Thus, we devised two alternative implementations: (1) one using the cleaned, directly thresholded result, and (2) one using the uncleaned, directly thresholded result. Fig. 3 shows the results of using different mask images for our restricted growing algorithm. The original image in Fig. 3 is a STAR-2 aircraft-SAR sea ice image. The upper-right corner shows the result of using the originally defined mask image. Details were lost, and the floes were slightly shrunk. The lower-left corner shows the result of using the cleaned, directly thresholded result as the mask image; the lower-right corresponds to that of uncleaned, directly thresholded result as the mask image. Comparing the results visually, the last option yielded the most accurate result as small details and wiggliness of the boundaries have been completely preserved. However, since it uses the directly thresholded result as the mask image, it suffers from noise effects, as evident in the interior of the large ice floe. All three segmentation methods are: if the image is very noisy, we recommend the mask image as computed in Eq. (2); if the details of the boundaries are

important, we recommend the directly thresholded result as the mask image; if the image is noisy and details are important, we recommend the cleaned thresholded result as the mask image.

5 Generating Core Image

To generate core objects, we have investigated several techniques. Of particular interest is one that utilized morphological reconstruction [35] because of its similarity to the underlying idea of RGC. In [35], the author extracted as core objects h-domes of regions in an image through reconstruction. The process is as follows. First, subtract the original image by h for all pixels to obtain the minus-h image (where all negative values are equated to zeros). Second, reconstruct the features in the minus-h image to obtain regional maximum—structures analogous to highland plateaus. Third, subtract the original image by the reconstructed minus-h image; the leftover features are *domes*. The reconstruction process is similar to RGC but it does not address the separation issue. In both cases, the reconstruction of the core image is restricted by the mask image. However, separated core objects that are connected in the mask image will become connected since the reconstruction process does not have a provision to recognize and preserve the separation. RGC, on the other hand, is able to recognize such separation and preserve it.

5.1 Extracting h-Domes As Cores

Moreover, [35] proposed a sequential reconstruction algorithm for either grayscale or binary images, which we have implemented to analyze SAR sea ice images for core. The algorithm is as follows. First, obtain the mask image, M. Second, obtain the core (or marker [35]) image, C. Third, repeat until the core image converges the following sequential operations: (1) Scan the core image rasterly and, for each nonzero pixel, p, in C, assign to it the value of $\max\left\{C(q), q \in N_G^+(p) \cup \{p\}\right\} \wedge M(p)$, where $N_G^+(p)$ is the collection of pixels in the 8-neighhorhood of p already visited (rasterly) before arriving at p. (2) Scan the core image anti-rasterly and, for each nonzero pixel, p, in C, assign to it the value of $\max\left\{C(q), q \in N_G^-(p) \cup \{p\}\right\} \wedge M(p)$, where $N_G^-(p)$ is the collection of pixels in the 8-neighhorhood of p already visited (anti-rasterly) before arriving at p. For binary images, the operator $\wedge$ is logical AND; for grayscale images, it is the minimum operator. Fig. 4 shows an original image and two binary h-domes images; Fig. 5 shows the examples of a cross section of

the minus-h images and h-domes images. As can be observed from both figures, the choice of h effects the results greatly. If h is too small, then the algorithm produces overly reduced cores which might not be representative of their original sizes and shapes, as shown in Fig. 4(b). If h is too large, the reconstructed plateau might be too low, and all regions would be flooded and be considered as a single region, as shown in Fig. 4(c).

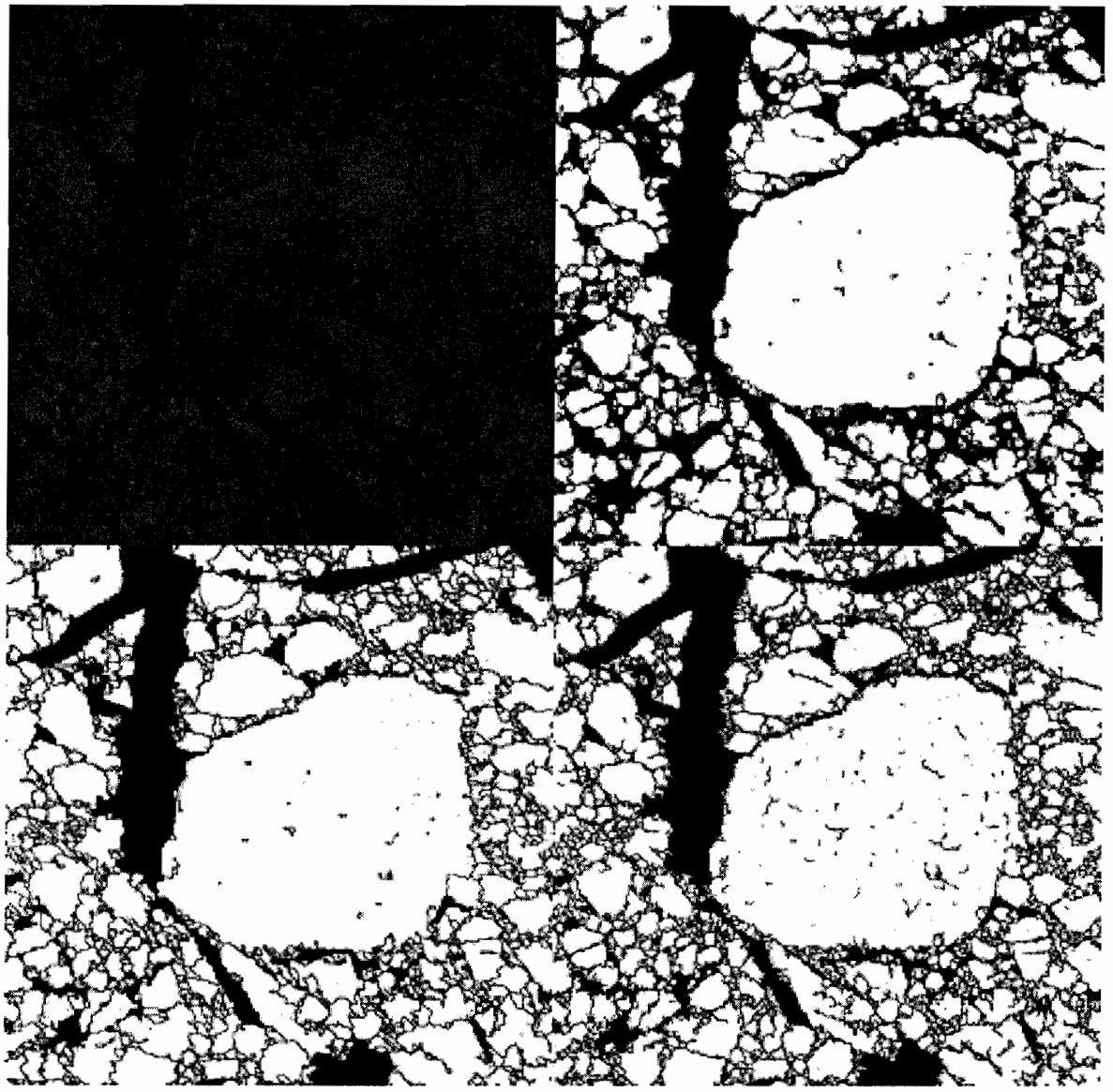

Fig. 3 Experiments on different mask image designs. The raw image shows a STAR-2 aircraft SAR sea ice image (obtained from Dr. J. Comiso). The upper-right corner shows the result of using our object separation technique with the mask image as defined in Eq. (2); the lower-left corner shows the result with the morphologically cleaned, direct thresholded result of the original image; the lower-right corner shows the result with the uncleaned, directly thresholded result of the original image.

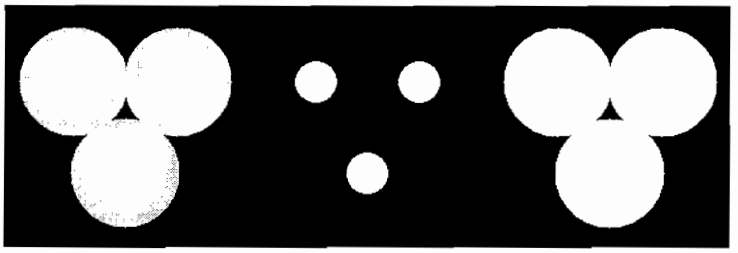

Fig. 4 From left to right: (a) Original image of three touching circles. (b) Binary h-domes image of the original image where $h = 20$. (c) Binary h-domes image of the original image where $h = 60$.

Reconstruction and h-domes can be applied to well-behaved or synthetic images to extract cores. However, for remotely-sensed imagery such as SAR sea ice images, due to inherent speckle noise, such application is not desirable. Referring to Fig. 6, one see that the noise effects and intrinsic heterogeneity within each region have forbidden the reconstruction to flood and form good quality plateaus. This results in many isolated, trivial cores. Fig. 7 shows a cross section taken from the image. The first graph reflects the original image; the second the reconstructed minus-h image with flooded plateaus; the third the h-domes. As observed, the application resulted in many individual domes and thus many cores.

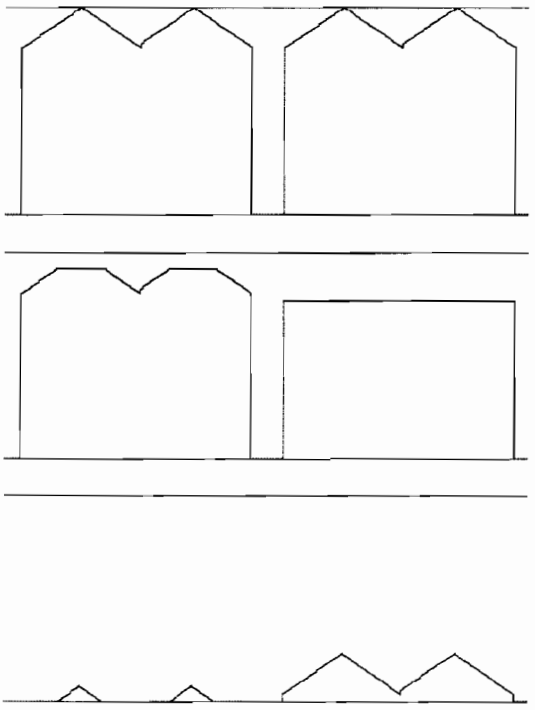

Fig. 5 Column (1) Top to bottom: Cross sections of (a) the original image of Fig. 4: two cones. (b) the reconstructed minus-h image ($h = 20$) of the original image. (c) the h-domes image. Column (2) Top to bottom: Cross sections of the (a) Same as (1)(a). (b) the reconstructed minus-h image ($h = 60$) of the original image. (c) the h-domes image.

5.2 Extracting Cores Using Probabilistic Labeling

Even though reconstruction and h-domes are useful morphological methodologies, they do not readily apply to remotely-sensed images because of speckle noise and heterogeneity within features such as those found in SAR sea ice images. Thus, we have again turned to the probabilistic labeling technique discussed above. To obtain core object pixels, we use:

$$\Omega_{core} = \Omega\left(t(i,j),2,5\right) = \left\{S_{t(i,j)}, S_{t(i,j)+2}, S_{t(i,j)+4}, S_{t(i,j)+6}, S_{t(i,j)+8}\right\} \qquad (3)$$

and $I_{mask} = 2$, and $N_{mask} = 5$. The sum of confidence is

$$C_{core}(i,j) = \sum_{S_t \in \Omega_{core}} c_{S_t}\left[(i,j) = object\right] \qquad (4)$$

Similar to the treatment of the pixels in the mask image, if the $C_{core}(i,j)$ of a pixel at (i,j) is greater than or equal to T_{core}, then that pixel is an object pixel in the core image. T_{core} has been experimentally determined to be 0.50 for SAR sea ice images.

In [32], we evaluated probabilistic labeling-based core extraction on various synthetic images to measure its ability to (1) separate touching objects with N number of shared boundary pixels, and (2) preserve sizes and shapes under various percentages of added noise. We also evaluated cores generated by morphological erosion and found that the technique described in this subsection to be superior. Due to the complexity of those experiments, we do not discuss them here. Briefly, the current implementation of core extraction was found to be able to separate touching objects with up to 25 shared boundary pixels and obtain primary cores in 10%-15% noise-corrupted images.

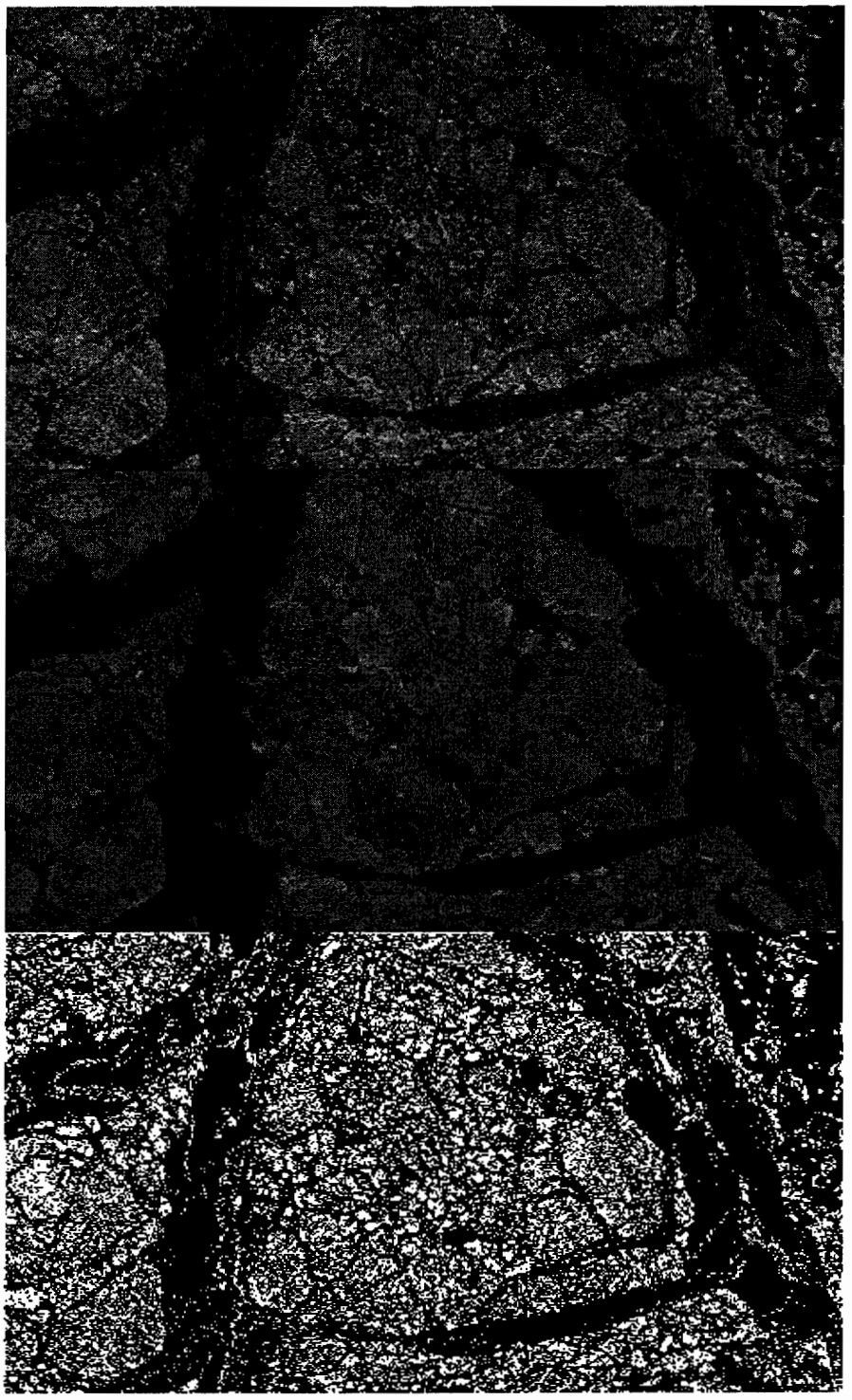

Fig. 6 Applying morphological reconstruction and *h*-domes directly to satellite sea ice image. (a) Original image is a STAR-2 aircraft SAR sea ice image (obtained from Dr. J. Comiso) (b) Reconstructed minus-*h* image (h = 40) (c) Binary *h*-domes.

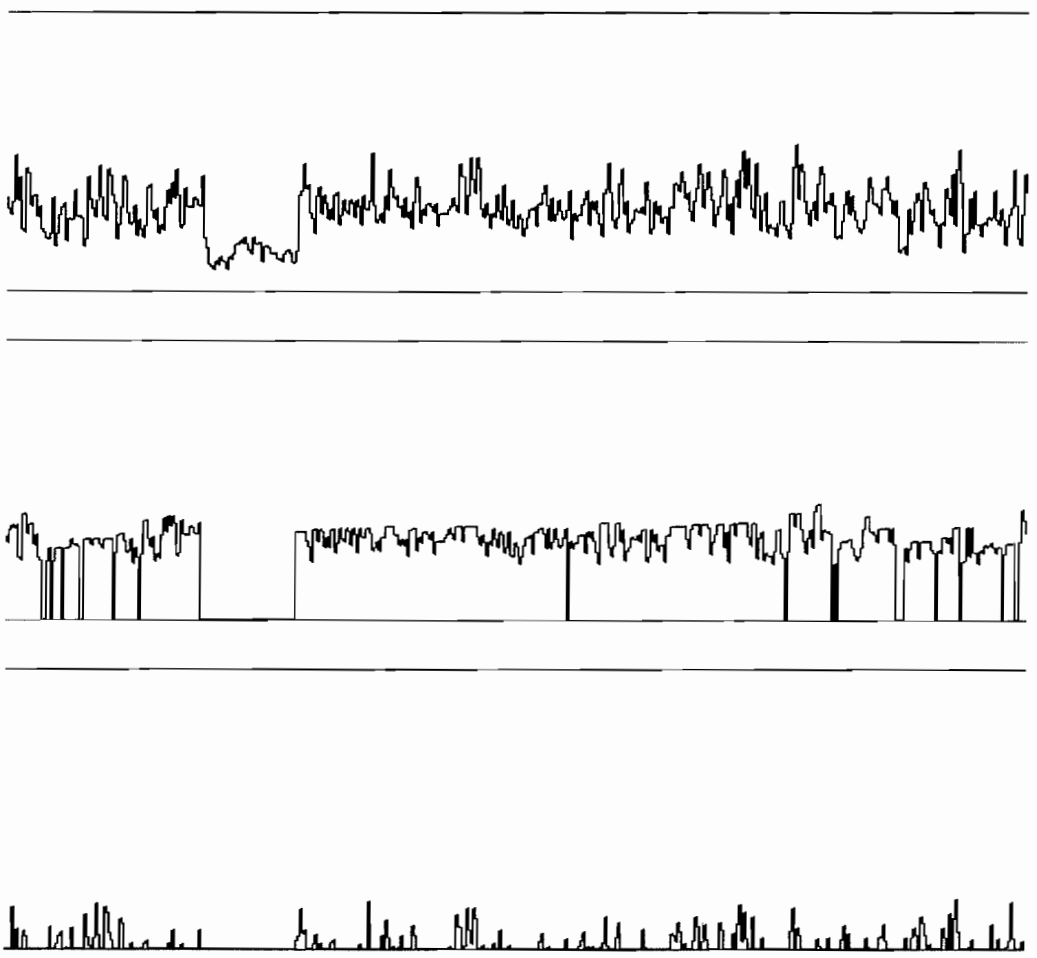

Fig. 7 Cross sections of (a) the original image. (b) the reconstructed minus-h image (h = 40) of the original image. (c) the h-domes image.

6 The Restricted Growing Algorithm Using Reversed Skeletonization

Our restricted growing algorithm earns its novelty with its use of skeletonization in a reversed manner. Note that the objective of the restricted growing algorithm is to grow core objects within the boundary of their corresponding mask objects while preserving existing separation among the core objects. In our definition, a growing algorithm (1) expands the core object to its original shape and size, and (2) introduces no additional connectivity to its neighbors. Skeletonization, in contrast, is a process that (1) reduces an object to its skeleton, and (2) ensures connectivity of skeletal branches within the object. That these two processes are closely related prompted us to design the restricted growing algorithm by modifying a skeletonization algorithm. Formally, a skeletonization algorithm reduces a binary object into a one-pixel thick skeleton, which should (1) preserve the object's topology, (2) be central to the object, or (3)

enable the original object to be recoverable from the skeleton [19]. The definition of a skeleton was first introduced in [4] as the medial axis, obtained by setting up grassfires simultaneously at all fronts. Numerous designs of thinning algorithms have since appeared in literature [14][15], aimed at improving the rate of convergence, accuracy, and connectivity [23] and preservation of skeletal legs [12].

6.1. *A Skeletonization Algorithm*

Here we describe a skeletonization algorithm proposed by [36]. The object pixels are assigned 1 and non-object pixels 0, and a boundary pixel is a pixel of an object that has at least one non-object 8-neighbor. This algorithm consists of iterations of two basic steps applied to object pixels. Fig. 8 shows the 8-neighborhood numbering.

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

Fig. 8 The numbering sequence of the 8-neighborhood of p_1.

During the first step, a boundary pixel p_1 is flagged if the following conditions are satisfied:

(1) $2 \le N(p_1) \le 6$, (2) $S(p_1) = 1$, (3) $p_2 \cdot p_4 \cdot p_6 = 0$, and (4) $p_4 \cdot p_6 \cdot p_8 = 0$,

where $N(p_1)$ is the number of nonzero neighbors of the boundary pixel p_1, and $S(p_1)$ is the number of 0-1 transitions in the ordered sequence of the neighbors of p_1. During the second step, a boundary pixel p_1 is flagged if

(1) $2 \le N(p_1) \le 6$, (2) $S(p_1) = 1$, (3) $p_2 \cdot p_4 \cdot p_8 = 0$, and (4) $p_2 \cdot p_6 \cdot p_8 = 0$.

One iteration of the algorithm includes applying the first step to flag boundary pixels for deletion, deleting the flagged pixels, applying the second step to flag remaining boundary pixels for deletion, and deleting the flagged pixels; and it iterates until no further pixels are deleted.

The first condition is violated when p_1 has only one or seven 8-neighbors valued as object pixels. A boundary pixel having only one such neighbor implies that it is situated at the end of a skeletal leg, and therefore should not be deleted. On the other hand, if it has seven such neighbors, the deletion of the pixel would penetrate into the object and thus is not allowed. The second condition is violated when the neighborhood contains a one-pixel thick line ($S(p_1) > 1$).

A deletion of such a pixel is forbidden since it would introduce a break to the connectivity of the object. To satisfy the last two conditions of the first step, a pixel must be an east or south boundary pixel or a northwest corner pixel in the boundary. Correspondingly, during the second step, a pixel must be a north or west boundary pixel or a southeast corner pixel in the boundary. Any point matching any of these four patterns should be removed.

5.2. *The Restricted Growing Algorithm*

Our design of the restricted growing algorithm was based closely on the thinning algorithm described above. Note that a pixel is deleted if it satisfies all conditions mentioned in the above subsection during the thinning process. In restricted growing, a pixel is grown or rejected based on the following sequential tests.

Test 1: Potential Growing Condition

If a pixel in the core image is a non-object pixel and its corresponding pixel in the mask image is an object pixel, then the pixel is qualified for further test.

This test selects only non-object core pixels for potential growth. This is how we guarantee that we only grow core objects within the boundary of mask objects. In addition, pixels that are non-object in both core and mask images are deemed as true non-object and are rejected from growing.

Test 2: Isolation Condition

If the pixel in the core image does not have an object pixel in its core image as an 8-neighbor, then the pixel is disqualified.

This condition avoids erroneous separation within an object. For example, small dark specks in an object would be eroded to non-object pixels during the generation of the core image. If allowed to grow, these non-object holes could become individual object regions within the object that encloses them. With this test, a small hole within an object will instead be consumed by the encroaching object during the growing process.

Test 3: Connectivity Condition 1

If the pixel in the core image has seven or eight object pixels in its core image 8-neighborhood, then the pixel is grown.

If the 8-neighborhood of a pixel has seven or more object pixels, that means all object pixels in that neighborhood are connected. Hence, the growth of the pixel from non-object to object does not damage the existing (or non-existing) separation. Note that Tests 2 and 3 correspond collectively to the first condition of the skeletonization algorithm.

Test 4: Connectivity Condition 2

If the pixel in the core image has no or one 1-0 transition in its core image 8-neighborhood, then the pixel is grown.

If no or one 1-0 transition ($S'(P_1) \leq 1$) is found, that means all object pixels in the area are connected and further growth does not alter the separation. This condition is analogous to the second condition of the skeletonization algorithm.

Test 5: Connectivity Condition 3

If the 8 neighborhood of the pixel in the core image matches one of the four corner patterns, then the pixel is grown.

The *corner patterns* are shown in Fig. 9. Any of these patterns could have two or more 1-0 transitions yet have all its object pixels connected. Thus, a pixel growth will not alter the separation. This test is a combination of the last two conditions of both steps of the thinning algorithm.

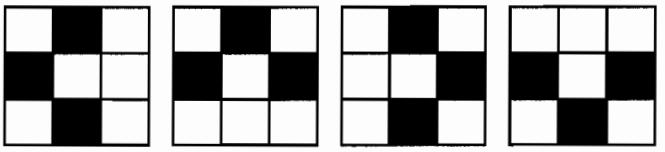

Fig. 9 Four patterns examined in Test 5 of the restricted growing algorithm. Dark pixels denote object pixels; unshaded pixels denote "don't care" pixels.

7 Analysis of Different Algorithms of RGC

So far, we have discussed RGC and addressed various issues regarding its three main components. We have also investigated the use of reconstruction and h-domes for core extraction, and subsequently determined the probabilistic labeling approach to core and mask extraction suitable for our domain, i.e., remotely sensed imagery. Also, we have reviewed a thinning algorithm and converted it to a restricted growing algorithm. Now, we arrive at the stage of joining all these three components together.

7.1 Algorithms

The algorithm of the basic RGC can be expressed in the following pseudocode:

algorithm RGC
(1) Generate the mask image.
(2) Generate the core image.
(3) Scan the core image in *some manner*, and for each pixel encountered:
  (a) Apply the tests (as described in Section 6).
  (b) **If** the pixel passes the tests, **Then** convert it to an object pixel.
  (c) **If** no change, **Then** move on to the *next* pixel.
(4) **Repeat** step (3) **Until** the core image converges.

The two keys to the design of the above algorithm are (1) how one scans the image and (2) how one selects the next pixel for processing. The scanning order influences the final processing outcome since the algorithm is implemented sequentially. In addition, the selection of the next pixel hinges upon whether one desires the growth of the current pixel to immediately affect the qualification of the next pixel since the growth is constantly updated to the core image where pixels are examined. Here below we present twelve different algorithms.

algorithm RGC_BASIC
(1) Generate the mask image.
(2) Generate the core image.
(3) Scan the core image *from top to bottom, left to right,* and for each pixel encountered:
  (a) Apply the tests (as described in Section 6)
  (b) **If** the pixel passes the tests, **Then** convert it to an object pixel, and
    move to the next row and column of the current pixel location.
  (c) **If** no change, **Then** move on to the next pixel in the raster scan.
(4) **Repeat** step (3) **Until** the core image converges.

The above algorithm uses the basic raster scanning. After each growth, the raster simply moves to the next column and row to reduce horizontal (or vertical) growth tendency in the image.

algorithm RGC_JUMP
(1) Generate the mask image.
(2) Generate the core image.
(3) Compute the JUMP_STEP as 10% of the minimum dimension of the image.
(4) Scan the core image *from top to bottom, left to right*, and for each pixel encountered:
  (a) Apply the tests (as described in Section 6).
  (b) **If** the pixel passes the tests, **Then** convert it to an object pixel, and
    jump JUMP_STEP rows and columns ahead to get the next pixel.
  (c) **If** no change, **Then** move on to the next pixel in the raster scan.
(5) **Repeat** step (4) **Until** the core image converges.

The above algorithm jumps to a distant pixel for continued processing; thus the growth of the current pixel does not immediately affect the qualification of the next pixel. Also, a region will not be grown continuously to a point where it dominates its neighboring regions by absorbing their pixels. The resulting growth pattern will be more balanced.

algorithm RGC_BLOB_COLORING
(1) Generate the mask image.
(2) Generate the core image.
(3) Blob color the core objects: each core object is tagged with a unique ID.
(4) Scan the core image *from top to bottom, left to right*, and for each pixel encountered:
  (a) Apply the tests (as described in Section 6).
  (b) **If** the pixel passes the tests, **Then** convert it to an object pixel, and
    move on to the next column and next row.
  (c) **If** the pixel fails the tests, **Then** check all its object neighbors;
    (c.1) **If** they all share the same ID, **Then** convert the pixel to an object pixel;
      and *label the new object pixel with the same ID,*
      and *move on to the next column and next row.*
  (d) **If** no change, **Then** move on to the next pixel in the raster scan.
(5) **Repeat** step (4) **Until** the core image converges.

In RGC_BASIC and RGC_JUMP, a pixel is always prevented from becoming an object pixel when it is between, for example, two separated object neighbors since a growth would connect the two neighbors, destroying the separation. However, there are scenarios where a region might have an internal linear break. If the break is significant in size, the algorithm treats it as a genuine feature and it remains in the final result. If the break is negligible, we want the algorithm to absorb it into the region. The inclusion of the blob-coloring filter offers that

absorption. If all tests fail to grow the pixel, we look at the neighbors of the pixels; if they are labeled with the same ID, then there is no harm in growing the pixel and connecting the two parts of the same region. This was the motivation behind the above design. Blob coloring is an image processing technique used to identify contiguous object pixels as regions and label them each with a unique ID. The pseudocode for an 8-neighbor blob coloring is as follows:

algorithm BLOB_COLORING
(1) Scan the image *from top to bottom, left to right*, and for each *object* pixel encountered:
  (a) **If** none of the $N_G^+(p)$ of the pixel is an object pixel, **Then**
    assign a new ID to the pixel.
  (b) **If** only one neighbor in the $N_G^+(p)$ of the pixel is an object pixel, **Then**
    assign the ID of that neighbor to the current pixel.
  (c) **If** there are more than one object neighbors in the $N_G^+(p)$ of the pixel, **And**
    each has the same ID, **Then** assign the ID to the current pixel.
  (d) **If** there are more than one object neighbors in the $N_G^+(p)$ of the pixel, **And**
    as a whole have more than one IDs, **Then**
    (d.1) assign one of the IDs, Λ, the current pixel.
    (d.2) list all other IDs of the neighbors.
    (d.3) convert all (so-far-labeled) pixels with IDs on the list to Λ.

Remember, from Section 5.1, that $N_G^+(p)$ is the collection of pixels in the 8-neighhorhood of p already visited (rasterly) before arriving at p. We also assign to the newly grown pixel the ID of its neighbors so the blob grows consistently.

algorithm RGC_BLOB_COLORING_JUMP
(1) Generate the mask image.
(2) Generate the core image.
(3) Compute the JUMP_STEP as 10% of the minimum dimension of the image.
(4) Blob color the core objects: each core object is tagged with a unique ID.
(5) Scan the core image *from top to bottom, left to right*, and for each pixel encountered:
  (a) Apply the tests (as described in Section 6).
  (b) **If** the pixel passes the tests, **Then** convert it to an object pixel, and
    move on to the next column and next row.
  (c) **If** the pixel fails the tests, **Then** check all its object neighbors;
    (c.1) **If** they all share the same ID, **Then** convert the pixel to an object pixel;
      and *label the object pixel with the same ID,*
      and *move on to the next column and next row.*
  (d) **If** no change, **Then** move on to the next pixel in the raster scan.
(6) **Repeat** step (5) **Until** the core image converges.

Similar to RGC_JUMP, the above implementation is a combination of using jump steps and blob coloring for selecting the next pixel for inspection.

algorithm RGC_DISTANCE
(1) Generate the mask image.
(2) Generate the core image.
(3) Generate the 8-distance transform of the mask image.
  (a) Obtain the maximum distance value, D_{max}.
(4) **Repeat** from $d = D_{max}$ to $d = 1$:
  (a) Scan the core image *from top to bottom, left to right,* **And**
  for each pixel whose 8-distance transform value equal to d:
    (a) Apply the tests (as described in Section 6)
    (b) **If** the pixel passes the tests, **Then** convert it to an object pixel, and
      move to the next row and column of the current pixel location
    (c) **If** no change, **Then** move on to the next pixel in the raster scan

The above implementation utilizes 8-distance transform, a representation that delineates the shortest distance of an object pixel from a non-object pixel [26]. The objective of this distance-based design is to grow each ring of a region at a time for all regions equally from the innermost pixels outward to discourage overgrowth or undergrowth. The following pseudocode facilitates the 8-distance transform:

algorithm 8_DISTANCE_TRANSFORM
(1) Scan the image *from top to bottom, left to right*, and for each pixel encountered:
  (a) assign to the pixel directly the value of $\min\{M(q), q \in N_G^+(p) \cup \{p\}\}$.
(2) Scan the image *from bottom to top, right to left*, and for each pixel encountered:
  (a) assign to the pixel directly the value of $\min\{M(q), q \in N_G^-(p)\}$.

Remember that $N_G^-(p)$ is the collection of pixels in the 8-neighhorhood of p already visited (anti-rasterly) before arriving at p, and that M is the mask image.

algorithm RGC_DISTANCE_BLOB_COLORING
(1) Generate the mask image.
(2) Generate the core image.
(3) Blob color the core objects: each core object is tagged with a unique ID.
(4) Generate the 8-distance transform of the mask image.
  (a) Obtain the maximum distance value, D_{max}.
(5) **Repeat** from $d = D_{max}$ to $d = 1$:

(a) Scan the core image *from top to bottom, left to right,* **And**
for each pixel whose 8-distance transform value equal to d:
(a.1) Apply the tests (as described in Section 6).
(a.2) **If** the pixel passes the tests, **Then** convert it to an object pixel, and *move on to the next column and next row.*
(a.3) **If** the pixel fails the tests, **Then** check all its object neighbors;
(a.3.1) **If** they all share the same ID, **Then**
(a.3.1.1) convert the pixel to an object pixel.
(a.3.1.2) label the object pixel with the same ID.
(a.3.1.3) move on to the next column and next row.
(a.4) **If** no change, **Then** move on to the next pixel in the raster scan.

The above implementation combines the 8-distance transform and blob coloring in growing object pixels. So far, we have described six different implementations of our RGC concept with one raster scan per iteration. In morphology, a two-scan iteration is often used to obtain balanced consideration for all pixels in the image in both directions (such as that mentioned in Section 5.1). By always looking at previously visited 8-neighborhoods ($N_G^+(p)$ and $N_G^-(p)$), we avoid including not-yet-processed neighbors when examining the current pixel; and by scanning from two directions sequentially in one iteration, we reduce directional biases that might occur otherwise. The following is the pseudocode of the algorithm RGC_BASIC using a two-scan iteration:

algorithm RGC_BASIC2
(1) Generate the mask image.
(2) Generate the core image.
(3) **Repeat** the following **Until** the core image converges:
(a) Scan the core image *from top to bottom, left to right,* and for each pixel encountered:
(a.1) **If** the $N_G^+(p)$ of the current pixel has at least one object pixel, **Then**
(a.1.1) Apply the tests (as described in Section 6)
(a.1.2) **If** the pixel passes the tests, **Then** convert it to an object pixel, and move to the +1 row and +1 column of the current pixel location.
(a.1.3) **If** no change, **Then** move on to the next pixel in the scan.
(b) Scan the core image *bottom to top, right to left,* and for each pixel encountered:
(b.1) **If** the $N_G^-(p)$ of the current pixel has at least one object pixel, **Then**
(b.1.1) Apply the tests (as described in Section 6)
(b.1.2) **If** the pixel passes the tests, **Then** convert it to an object pixel, and move to the -1 row and -1 column of the current pixel location.
(b.1.3) **If** no change, **Then** move on to the next pixel in the scan.

Note that (a.1) and (b.1) are similar to the maximum operators in the reconstruction algorithm discussed in Section 5.1 when the image is binary. In the same manner, we have also designed RGC_JUMP2, RGC_BLOB_COLORING2, RGC_BLOB_COLORING_JUMP2, RGC_DISTANCE2, and RGC_DISTANCE_BLOB_COLORING2. In all, we designed and implemented twelve different RGC algorithms for our experiments.

7.2 Results

Fig. 10 shows an example of the original images that we have used to test the different implementations discussed above. It is a portion of a Synthetic Aperture Radar (SAR) sea ice image taken by the first Earth Resource Satellite (ERS-1). Dark regions are ice floes; bright are water. The task was to separate the large piece of ice floe from its neighbors. Figs. 11-13 show the object separation results of the twelve algorithms.

Fig. 10 A portion of an ERS-1 SAR sea ice image taken on Aug 24 at Beaufort Sea (Copyright ESA).

After applying the implementations to numerous SAR sea ice images, we observe the following:

- Compared to the one-scan iteration design, the two-scan iteration installation offers better balance in regional growth. The regions are less complicated and they expand more fully towards the masks. However, the two-scan iteration designs are generally slower than the one-scan iteration designs.
- Implementations that include jumps to avoid immediate growth effects fair better in terms of shape definition. However, jumps also introduce breaks into the regions. On the other hand,

those without jumps are able to establish fuller regions by absorbing more pixels from neighboring regions.

Fig. 11 RGC results of Fig. 10. Top left: Result of RGC_BASIC. Top right: Result of RGC_JUMP. Bottom left: Result of RGC_BLOB_COLORING. Bottom right: Result of RGC_BLOB_COLORING_JUMP.

- By utilizing blob coloring in the restricted growing algorithm, we obtain regions that close better. 'Hairline' effects that are sometimes evident because of breaks within regions are reduced. This results in more compact regions. However, with blob coloring, non-object pixels are more sporadic and less connected. So, if one is interested in signifying the non-object breaks

and background pixels, then an implementation without blob coloring should produce more desirable results.

Fig. 12 RGC results of Fig. 10. Top left: Result of RGC_DISTANCE. Top right: Result of RGC_DISTANCE_BLOB_COLORING. Bottom left: Result of RGC_BASIC2. Bottom right: Result of RGC_JUMP2.

- The implementations with distance transform do not perform as well as those without such facility. There are two possible reasons. First, SAR sea ice images are noisy and results in holes in a region. These holes could be seen as lakes and thus the topology of the region in terms of the shortest distance to a non-object pixel is no longer in a uniform ring radiating outward from the

center of the region. This retains noise effects. Second, the 8-distance transform is not truly representative of the actual physical distance between pixels. To illustrate, the 4- neighbors of a pixel are each 1 unit away from the pixel; and the diagonal neighbors in an 8-neighborhood are each $\sqrt{2}$ units away from the center pixel. In our implementation, we did not account for this difference.

Fig. 13 RGC results of Fig. 10. Top left: Result of RGC_BLOB_COLORING2. Top right: Result of RGC_BLOB_COLORING_JUMP2. Bottom left: Result of RGC_DISTANCE2. Bottom right: Result of RGC_DISTANCE_BLOB_COLORING2.

- For SAR sea ice images because of its inherent noise, we do not recommend using distance transform alone, i.e., RGC_DIST, because the distance transform dominates the growth patterns and retains noise effects. Collaborating the distance transform with blob coloring or two-scan iteration improves the results. In addition, we do not recommend using two-scan iteration, blob coloring, and jumps (i.e., RGC_BLOB_COLORING_JUMP2) together. This combination extracts regions of blocky shapes as shown in Fig. 13. The two-scan iteration allows the tracking of a pixel and, combined with jumps, creates blocks of regions, which are further enhanced by the merging ability of the blob coloring.
- In general, most of the twelve algorithms yield good object separation results. They are able to tolerate speckle noise in object separation and reduce noise effects in object definition. Our evaluation of the results indicates that the algorithm RGC_BLOB_COLORING is the most consistent. We attribute the performance to two reasons: (1) the uni-directional scan and without jumps, and (2) the ability of the blob coloring-based approach to absorb negligible noise effects.

In terms of computational speeds, Table 1 exemplifies the performances of the algorithms. We have identified five speed groups: fast, moderately fast, average, slow, and very slow. Members of each group require roughly the same amount of time to accomplish the object separation task. In general, algorithms that implement blob coloring perform faster; those that implement distance transform perform more slowly; and finally those that implement jumps perform even more slowly. This is because of the skipping of pixels per iteration, due to either distance transforms or jumps, and the consequent additional number of iterations needed to converge the core image.

8 Discussions and Conclusions

The restricted growing concept consists of three main procedures: generating the mask objects, obtaining the core objects, and restricted growing. All can be implemented differently from what we have presented here. In addition to the three alternatives of generating the mask image that we have provided, one can experiment with other types of segmentation process as long as it yields objects with preserved shapes and sizes. Approaches, such as relaxation [6][13][24] and multiple resolution [3][34], that are able to erode an object based on its boundary intensities and neighborhood suitable. Spatial and textural statistics can also be used instead of probability of intensities in the neighborhood for determining the status of each pixel as a core or mask pixel.

Similarly, one can design a different restricted growing algorithm as long as it grows core objects within the boundary of mask objects and does not destroy the existing separation during the process. A host of thinning algorithms designed for different applications [17][18][19] or faster computational speed [1][21][22] can be modified to improve the convergence rate of the growing process. Because of the generality and modularity of the restricted growing concept, one can design his or her own object separation algorithm to accommodate specific applications and imagery requirements.

SPEED GROUPS	ALGORITHMS	AVERAGE (ON A 450×600 IMAGE)
Fast	RGC_BASIC RGC_BLOB_COLORING RGC_BLOB_COLORING_JUMP RGC_BASIC2 RGC_BLOB_COLORING2	9.05 sec
Moderately Fast	RGC_DISTANCE RGC_DISTANCE2	22.18 sec
Average	RGC_DISTANCE_BLOB_COLORING RGC_DISTANCE_BLOB_COLORING_JUMP RGC_DISTANCE_BLOB_COLORING_JUMP2	52.00 sec
Slow	RGC_JUMP	148.57 sec
Very Slow	RGC_JUMP2	343.90 sec

Table 1 Computational speeds of the implementations of RGC. The programs were executed on a SGI Challenge L/6, 512 Mb RAM machine.

We have investigated the use of morphological reconstruction and h-domes in extracting cores for object extraction in Synthetic Aperture Radar (SAR) images. We have realized that, due to the inherent speckle noise, the two operators do not work well for our domain imagery. This has compelled us to use probabilistic labeling to extract mask and core images for our technique. This approach is more robust and noise-resistant. We have demonstrated in steps how to convert a skeletonization algorithm to a restricted growing algorithm. This novel utility of skeletonization algorithms allow system designers to design and implement RGC with readily knowledge compiled over the past decades on issues regarding the mathematics and applications of skeletonization or thinning. In addition, we have explored a variety of scanning orders and features such as blob coloring and distance transform to enhance or improve the design of RGC. We have concluded that most these implementations yield satisfactory results. We have further identified the most consistent implementation (which combines one-scan iteration and blob

coloring), and rejected two sub-par designs, in addition to pointing out weaknesses and advantages of each design.

In conclusion, we have presented a concept that is able to achieve both object separation and preservation of size and shape of objects. This concept is both intuitive and general. It helps solve the problem of object separation by addressing its two most important issues: separation among objects and preservation of objects. The restricted growing concept is a framework with which scientists can design image processing techniques tailored to specific domains and applications such as pattern recognition, feature extraction, object solidification, segmentation, and other tasks. Our implementation is able to handle a variety of images of the same domain (i.e., SAR sea ice) without any parameter adjustments after they have been determined. On the other hand, the parametric design allows flexibility in adapting to other image domains.

9 References

[1] C. Arcelli and G. Sanniti di Baja, "A Width-Independent Fast Thinning Algorithm," *IEEE Trans. PAMI*, vol. 7, no. 4, pp. 463-474, 1985.

[2] J. Banfield and A. E. Raftery, "Ice Floe Identification in Satellite Images Using Mathematical Morphology and Clustering about Principal Curves," *J. Amer. Stat. Assoc.*, vol. 87, no. 417, pp. 7-16, 1992.

[3] F. Bergholm, "Edge Focusing," *IEEE Trans. PAMI*, vol. 9, no. 6, pp. 726-741, 1987.

[4] H. Blum, "A Transformation for Extracting New Descriptors of Shape," in *Models for the Perception of Speech and Visual Form*, W. Wathen-Dunn (ed.)., Cambridge, MA: MIT Press, 1967.

[5] J. Daida, R. Samadani, and J. F. Vesecky, "Object-Oriented Feature-Tracking Algorithms for SAR Images of the Marginal Ice Zone," *IEEE Trans. GARS*, vol. 28, no. 4, pp. 573–589, 1990.

[6] A. J. Danker and A. Rosenfeld, "Blob Detection by Relaxation," *IEEE Trans. PAMI*, vol. 3, no. 1, pp. 79-92, 1981.

[7] M. Fily and D. A. Rothrock, "Opening and Closing of Sea Ice Leads: Digital Measurements from Synthetic Aperture Radar," *J. Geo. Res.*, vol. 95, no. C1, pp. 789–796, 1990.

[8] R. C. Gonzalez, "Industrial Computer Vision," in *Advances in Information Systems Science*, J. T. Tou (ed.), pp. 345-385, Plenum, New York, 1985.

[9] R. T. Hall and D. A. Rothrock, "Photogrammetric Observations of the Lateral Melt of Sea Ice Floes," *J. Geo. Res.*, vol. 92, no. C7, pp. 7045–7048, 1987.

[10] T. Hastie and W. X. Stuetzle, Principal Curves, *J. Amer. Stat. Assoc.*, vol. 84, pp. 502–516, 1989.

[11] D. Haverkamp, L.-K. Soh, and C. Tsatsoulis, "A Comprehensive, Automated Approach to Determining Sea Ice Thickness from SAR Data," *IEEE Trans. GARS*, vol. 33, no. 1, pp. 46-57, 1995.

[12] C. J. Hilditch, "Comparison of Thinning Algorithms on A Parallel Processor," *Image and Vision Computing*, vol. 1, no. 3, pp. 115–132, 1983.

[13] R. A. Hummel and S. W. Zucker, "On the Foundations of Relaxation Labeling Processes," *IEEE Trans. PAMI*, vol. 5, no. 3, pp. 267–287, 1983.

[14] B.-J. Jang and R. T. Chin, "Analysis of Thinning Algorithms Using Mathematical Morphology," *IEEE Trans. PAMI*, vol. 12, no. 6, pp. 541–551, 1990.

[15] L. Ji and J. Piper, "Fast Homotopy-Preserving Skeletons Using Mathematical Morphology," *IEEE Trans. PAMI*, vol. 14, no. 6, pp. 653–664, 1992.

[16] R. Korsnes, "Quantitative Analysis of Sea Ice Remote Sensing Imagery," *Int. J. Rem. Sen.*, vol. 14, no. 2, pp. 295–311, 1993

[17] L. Lam, S. W. Lee, and C. Y. Suen, "Thinning Methodologies—A Comprehensive Survey," *IEEE Trans. PAMI*, vol. 14, no. 9, pp. 869–885, 1992.

[18] L. Lam and C. Y. Suen, "An Evaluation of Parallel Thinning Algorithms for Character Recognition," *IEEE Trans. PAMI*, vol. 17, no. 9, pp. 914–919, 1995.

[19] P. A. Maragos and R. W. Schafer, "Morphological Skeleton Representation and Coding of Binary Images, *IEEE Trans. ASSP*, vol. 34, pp. 1228–1244, 1986.

[20] D. M. McKeown, W. A. Harvey, and J. McDermott, "Rule-Based Interpretation of Aerial Imagery," *IEEE Trans. PAMI*, vol. 7, no. 5, 1985, pp. 570–585, 1985.

[21] L. O'Gorman, "$k \times k$ Thinning," *CVGIP*, vol. 51, no. 2, 1990, pp. 195–215, 1990.

[22] J. Piper, "Efficient Implementation of Skeletonization Using Interval Coding," *Pattern Rec. Let.*, vol. 3, pp. 389–397, 1985.

[23] A. Rosenfeld, "Connectivity in Digital Pictures," *J. ACM*, vol. 17, no. 1, pp. 146–160, 1970.

[24] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene Labelling by Relaxation Operations," *IEEE Trans. SMC*, vol. 6, pp. 420–433, 1976.

[25] A. Rosenfeld and J. L. Pfaltz, "Distance Functions on Digital Pictures," *Pattern Rec.*, vol. 1, pp. 33-61, 1968.

[26] D. A. Rothrock and A. S. Thorndike, "Measuring the Sea Ice Floe Size Distribution," *J. Geo. Res.*, vol. 89, no. C4, pp. 6477–6486, 1984.

[27] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.

[28] J. Serra, "Introduction to Mathematical Morphology," *CVGIP*, vol. 35, pp. 283–305, 1986.

[29] L.-K. Soh, D. Haverkamp, and C. Tsatsoulis, "Separating Ice-Water Composites and Computing Floe Size Distributions," *Proc. IGARSS '96*, Lincoln, Nebraska, pp. 1532–1534, 1996.

[30] L.-K. Soh and C. Tsatsoulis, "A Multistage Feature Extraction Technique for Synthetic Aperture Radar (SAR) Sea Ice Imagery, Part I: Combination of Algorithmic and Heuristic Methods," *CECASE TR 9710-02*, The University of Kansas, Lawrence, Kansas, 1993.

[31] L.-K. Soh and C. Tsatsoulis, "A Feature Extraction Technique for Synthetic Aperture Radar (SAR) Sea Ice Imagery," *Proc. IGARSS '93*, Tokyo, Japan, pp. 632-634, 1993.

[32] L.-K. Soh and C. Tsatsoulis, "The Restricted Growing Concept for Object Separation," *ITTC-FY98-TR-11810-02*, ITTC, University of Kansas, Lawrence, KS, 1998.

[33] L.-K. Soh, C. Tsatsoulis, and B. Holt, "Identifying Ice Floes and Computing Ice Floe Distributions in SAR Images," in *Recent Advances in the Analysis of SAR Sea Ice Data*, C. Tsatsoulis and R. Kwok (eds.), pp. 9-34, Berlin: Springer-Verlag, pp. 9–34, 1998.

[34] D. Terzopoulos, "Image Analysis Using Multigrid Relaxation Methods," *IEEE Trans. PAMI*, vol. 8, no. 2, pp. 129–139, 1986.

[35] L. Vincent, "Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms," *IEEE Trans. IP*, vol. 2, no. 2, pp. 176-201, 1993.

[36] T. Y. Zhang and C. Y. Suen, "A Fast Parallel Algorithm for Thinning Digital Patterns," *CACM*. vol. 27, no. 3, pp. 236–239, 1984.

[37] S. W. Zucker, "Region Growing: Childhood and Adolescence," *CGIP*, vol. 5, pp. 382–399, 1976.