2002

A Set of New Sea Ice Feature Descriptors for SAR Images

Leen-Kiat Soh
University of Nebraska, lsoh2@unl.edu

University of Nebraska-Lincoln, Computer Science and Engineering
Technical Report # TR-UNL-CSE-2002-0004

A Set of New Sea Ice Feature Descriptors for SAR Images

Leen-Kiat Soh
Computer Science and Engineering Department
University of Nebraska
Lincoln, NE 68588-0115
Tel: (402) 472-6738 Fax: (402) 472-7787
E-mail: lksoh@cse.unl.edu

Abstract—In this paper, we present a set of algorithms for describing sea ice features in SAR images. These algorithms we have implemented and incorporated in the ARKTOS software package, an intelligent sea ice classification. These algorithms have unique characteristics. Some are extensions or adaptations of existing image processing techniques to the specific problem domain of satellite sea ice classification, while some are innovative designs, inspired by the aforementioned problem domain. These feature descriptors may be generalized to other remote sensing applications.

I INTRODUCTION

In our ARKTOS project [21], we label sea ice objects with different classes by analyzing their feature descriptors: geometric, textural, intrinsic, relational, and so on [22]. That is, the underlying methodology is to classify a group of pixels based on its appearance as a feature. This differs from pixel-based approaches in which a classifier analyzes each pixel individually and attempts to assign a classification directly to that pixel based on its spatial or spectral characteristics. Our ARKTOS methodology is motivated by the fact that sea ice experts look for feature-based visual cues in an image during analysis. Here in this paper, we present a set of new sea ice descriptors for SAR images. These descriptors are used to construct facts about features extracted from SAR images. Some facts are domain-specific such as “the object is an ice lead” or “the object is a blob.” Some facts are general such as “the object is irregular” or “the object is jagged.” ARKTOS classifies the objects based on these facts.

In the following, we first briefly discuss some background on remote sensing image analysis in general and on SAR sea ice classification in particular. In Section III, we present the algorithms for the new set of se ice feature descriptors. Subsequently in Section IV, we apply these descriptors in the context of SAR sea ice classification, from the viewpoint of ARKTOS. Finally, we conclude the paper.

II BACKGROUND

In general, the analysis of remote sensing imagery of natural scenes presents many unique problems. It differs from the shape-based analysis of urban, commercial or agricultural areas, and from medical and industrial imagery taken in usually controlled environments. Natural scenes (forests, mountains, sea ice, clouds, etc.) are not structured and cannot be represented easily by regular rules or grammars. In contrast to artificial structures, natural features do no obey strict positioning rules (for example, harbor structures will always be next to water, while coniferous forests can be found in very large geographical and elevation ranges, and their positioning rules are much weaker). Finally, the shape of natural features can vary greatly based on the dynamic environments that the features are in such as weather, currents, and so on. On the other hand, as these features take on different shapes in response to their environments, scientists have been able to make use of shape-based analysis to correlate to, or predict, or classify what the environments are and what the features are.

In this paper, we focus our discussion on shape-based analysis of sea ice features in SAR images. First, remote sensing of the polar regions has important applications in meteorology and global climate. For example, the thickness of sea ice influences the heat flux between the atmosphere and water surface; thus the classification and temporal tracking of sea ice can be used as an indicator in global climate monitoring. In addition, localization and classification of ice floes are important to commercial, scientific, and military navigation of the polar regions such as offshore and deep-sea fishing, oil drilling, marine biology, and geology explorations. Second, with better remote sensing, image collection and distribution capabilities and availability, image processing has become increasingly useful and important for examining both large- and small-scale sea ice geophysical processes [2, 3, 19]. Research issues in the areas of segmentation and classification [12, 13, 20], registration and motion tracking [4, 28], floe size measurements [15, 25], floe extraction [1], melt analysis [10, 26], lead analysis [5], ice edge localization and so on are of great interest. In particular, some of the approaches are ice-feature-dependent (or shape-based). Sea ice features are extracted from the image and geometric descriptors are computed for each

feature to be analyzed. This paper presents a set of algorithms for computing a set of new sea ice feature descriptors.

Even though the focus of our discussions here is with satellite sea ice classification, the feature descriptors introduced may be generalized to other remote sensing and image processing applications that involve shape-based or feature-based analyses: change detection, matching and registration, temporal tracking, classification, noise filtering, object identification, and so on [18]. Moreover, shape-based or feature-based approaches have been used increasingly for content-based image indexing and retrieval [27], in addition to color- and texture-based approaches. Shape-based analysis has also been involved in identification and change detection for river sandbars, lake areas, forest burn areas, and other natural objects in remote sensing images.

III METHODOLOGY

We have applied the above two measures to a sea ice classification system. This expert system extracts sea ice features from images and describes each feature with a set of measurements. It then converts the measurements into discrete, symbolic facts through simple and composite thresholding. During the classification phase, the system matches these facts against a set of rules. Rules that match will be fired with assertions—each assertion is a classification of the feature with a confidence value. The system combines all the assertions into a final, consistent classification for each feature. Segmentation of some sort to obtain feature and background.

A. Perimeter Porosity

To compute for the perimeter porosity of a feature, we first construct the chain code [6, 7] of the feature's boundary. In this algorithm, we use a 4-neighbor chain code. It contains an initial point $\langle x_{ini}, y_{ini} \rangle$, an end point $\langle x_{end}, y_{end} \rangle$, and a set of 4-directions leading from the current point to the next on the boundary. Given an image feature with its pixels labeled with a unique value *o*, the following algorithm computes the perimeter porosity:

Algorithm Perimeter_Porosity

1. ***Algorithm Outer_Perimeter***
a. Find the topmost, leftmost pixel with the label *o*. This is the initial point of the chain code.
b. Construct the chain code by moving to the leftmost 4-neighbor that shares the same label until the next neighbor has reaches the initial point of the chain code.
c. The number of directions or moves of the chain code is the *outer_perimeter*.
2. ***Algorithm Perimeter***
a. Scan from left to right, top to bottom: if a pixel with the label *o* has a non-*o* 8-neighbor, then add 1 to the *perimeter* count.
3. Compute:

$$perimeter\_porosity = \frac{\max(outer\_perimeter, perimeter)}{\min(outer\_perimeter, perimeter)}$$

*Note that if we use, instead, an 8-neighbor chain code, then we look for non-*o* 4-neighbors.

End Algorithm

Note that the algorithm is based on two different ways of computing perimeters. The value of *perimeter* is the actual boundary pixels of the feature. The value of *outer_perimeter* is the number of moves one needs to take in order to traverse the entire boundary of the feature—that may actually require moving to boundary pixels that have been traversed before.

Figures 1(a) and 1(b) show two examples related to structural weakness of a feature: 1-pixel-thick protrusions. In the first example, the *outer_perimeter* value is 11, following the 4-neighbor chain code, while the *perimeter* is only 10. Thus, the *perimeter_porosity* is 14/10 or 1.4. In the second example, the *outer_perimeter* is 14, while the *perimeter* is 14. Thus, the *perimeter_porosity* is 14/14 or 1.0. In the third example, we can see that the *outer_perimeter* is 14, the *perimeter* is 14, and thus the *perimeter_porosity* is once again 1.0. Finally, for the last example, the *perimeter* and the *outer_perimeter* are the same at 14, and the *perimeter_porosity* is 1.0.

Figures 1(c) and 1(d) show two examples related to the porosity of a feature: holes within the feature. In general, the size of a hole and how the holes are distributed in a feature affect the perimeter porosity value. Also, even though the total size of holes in Figure 1(c) and Figure 1(d) are the same—that is, 4 pixels—the perimeter porosity values are different. However, Figure 1(c) results in 1.55 and Figure 1(d) in 1.73.

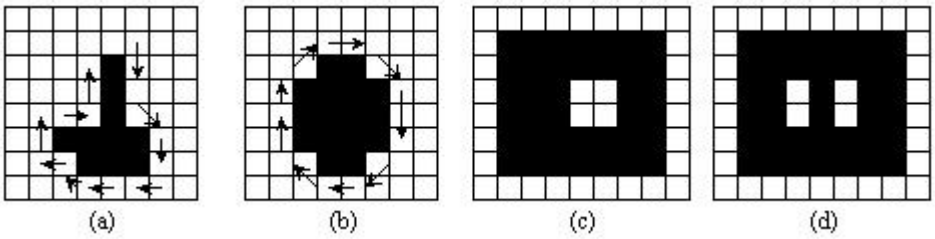

Figure 1 (a) *perimeter_porosity* = 14/10 = 1.4, (b) *perimeter_porosity* = 14/14 = 1.0, (c) *perimeter_porosity* = 34/22 = 1.55, and (d) *perimeter_porosity* = 38/22 = 1.73.

Note that if we had used just the number of feature pixels and the number of "hole" pixels as a porosity measure, Figure 1(c) and Figure 1(d) would have yielded the same porosity values. Thus, our design for perimeter porosity has a bias: a feature with many small holes is in general more porous than a feature with a single large hole. This corresponds intuitively to what we expect of a porosity measure.

B. Area Porosity

The underlying principle for this measure is based on the best-fitting bounding rectangle of a feature. The idea is that a *structurally compact* feature should fill out its bounding rectangle nicely. One that does not is considered to have a high area porosity measure.

We compute for the bounding rectangle indirectly by computing for the size of its area, instead of the actual corner points of the rectangle. We first find the orientation of the feature, to gauge how it aligns in the image, based on the feature's first- and second-order moments [14]:

Algorithm Orientation

a. Compute:

$$orientation = \frac{1}{2}\tan^{-1}\left[\frac{2\overline{\mu}_{1,1}}{\overline{\mu}_{2,0} - \overline{\mu}_{0,2}}\right]$$

where $\overline{\mu}_{0,2} = \sum_{all\,y}(y-\mu_y)^2$, $\overline{\mu}_{2,0} = \sum_{all\,x}(x-\mu_x)^2$, $\overline{\mu}_{1,1} = \sum_{all\,x}\sum_{all\,y}(x-\mu_x)(y-\mu_y)$, x is the column value of a pixel located at $\langle x, y\rangle$, y is the row value of a pixel located at $\langle x, y\rangle$, μ_x is the mean value of all values of x, and μ_y is the mean value of all values of y.

End Algorithm

Given the orientation value, we can then identify the two principal axes as the maximum length and width of the feature [14]:

Algorithm Maximum Length and Width

a. For each pixel $\langle x, y \rangle$, compute its

$$\alpha = x\cos(orientation) + y\sin(orientation)$$
$$\beta = -x\sin(orientation) + y\cos(orientation)$$

b. Identify the maximum α as $\alpha_{\max}$; identify the maximum β as $\beta_{\max}$; identify the minimum α as $\alpha_{\min}$; identify the minimum β as $\beta_{\min}$.

c. Compute the lengths of the principal axes:

$$d_\alpha = \alpha_{\max} - \alpha_{\min}, \text{ and}$$
$$d_\beta = \beta_{\max} - \beta_{\min}.$$

d. Compute

$$max\_length = \max(d_\alpha, d_\beta), \text{ and}$$
$$max\_width = \min(d_\alpha, d_\beta).$$

End Algorithm

Finally, the algorithm for computing for area porosity is:

Algorithm Area_Porosity

1. *Algorithm Orientation*
2. *Algorithm Maximum Length and Width*. This yields *max_length* and *max_width*.
3. Compute the size of the feature by counting all the *o*-labeled pixels. This is *area*.
4. Compute:

$$area\_porosity = \frac{max\_width \bullet max\_length}{area}$$

End Algorithm

Figure 3 shows an example related to area porosity. In Figure 3(a), the size of the bounding rectangle is 9×5 = 45. The actual area of the feature is 17 pixels. Thus, the area porosity of the feature is 45/17 = 2.65. In Figure 3(b), the size of the bounding rectangle is the same as the area of the feature: 9. Thus, the area porosity of the feature is 1.0.

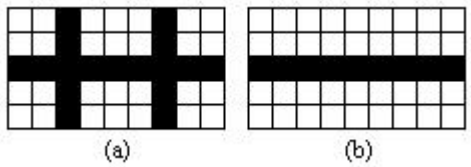

Figure 3 (a) *area_porosity* = 45/17 = 2.65, and (b) *area_porosity* = 9/9 = 1.0.

C. Irregularity

We define irregularity as a complex measurement based on the two porosity measures:

$$irregularity = area\_porosity \cdot perimeter\_porosity.$$

That is, a feature is highly irregular if it has a high area porosity measure and a high perimeter porosity measure.

D. Mottledness

Traditionally, sea ice texture has been difficult to capture and many approaches have been proposed [23]. Most research has studied sea ice texture at two resolutions: (1) the texture on the surface of a sea ice feature or feature, and (2) the texture exhibited by a collection of sea ice features. Thus, the first resolution is local, while the second is global or regional. Textural algorithms such as gray level co-occurrence matrices (GLCM) [11], gray level run length [8], Fourier power spectrum [17], texture spectrum [29], and semivariograms [16] are among the common approaches in the remote sensing and image processing literature.

We propose here a new approach to the surface texture of a sea ice feature. The goal is not to measure the degree of texture or identify the type of texture on the surface of a sea ice feature. Instead, we look to determine whether there is texture on the surface of the feature. This algorithm is fast, simplistic and effective.

Algorithm Mottledness

1. Obtain the maximum difference between two horizontal, neighboring pixels of the feature, $\max_{horizontal}$.
2. Obtain the maximum difference between two vertical, neighboring pixels of the feature, $\max_{vertical}$.
3. Compute:

$$mottledness = \left(\max_{vertical} + \max_{horizontal}\right) \bullet \frac{average\_intensity}{255}.$$

*The value 255, the maximum intensity value, is used as a normalization factor.

End Algorithm

The strategy of the algorithm is to detect the intensity differences in a feature. If we detect high intensity differences, then we say the feature is mottled. Also, we tolerate less difference for bright features than we do for dark features to capture the observation that a slight change at the bright end of the intensity spectrum is more significant than one at the dark end. This treatment allows us to effectively associate mottledness to the ridged surface of a piece of ice floe and the streaked surface of a lead-like feature. For

example, an ice floe may only appear to be ridged partially due to noise or radar limitations, or due to actual physical makeup. A feature that looks like a lead may be open water if its surface is smooth. At times, such a feature has streaks of brighter features within its area, and may help classify the feature as young or new ice. In both cases, a texture algorithm that aggregates the entire surface may not result in a high texture value. However, our mottledness algorithm is based on existence of texture, and thus is able to give a high texture value. Also, we do have a bias for brighter features. That is, it is easier for a brighter feature to have a higher mottledness value. This is because it is more difficult to see the intensity difference caused by a ridge, for example, on a brighter sea ice surface.

Figure 4 shows an example related to mottledness. Both features have the same maximum intensity difference at 127. But the second feature has bright ridges, and that translates into a higher average intensity, which ultimately gives the second feature a higher mottledness value. So, indirectly, our algorithm does compensate to some extent for the "degree" of texture on a surface.

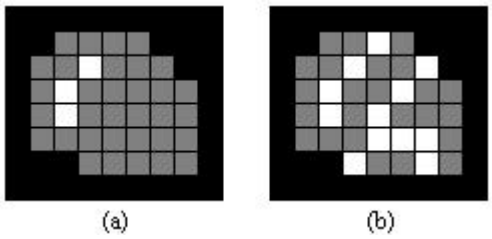

Figure 4 Black, gray, and white pixels' intensity values: 0, 128, 255, respectively. Black pixels are background pixels while the others are feature pixels. (a) *mottledness* =(127·138.58)/255 = 69.02, and (b) *mottleness* =(127·170.33)/255 = 84.83.

In addition to mottledness, we have designed two different "roughness" measures: *average_roughness* and *new_roughness*. Both approaches aggregate surface texture. The first approach is to utilize local variances while the second approach combine the global variance with the local one. We use these two measures in our comparative studies.

Algorithm Average_Rouhness

1. Divide the feature into overlapping 5x5 areas.
2. Compute the variance of each area.
3. Sum all the variances.
4. Divide the sum with the feature's *area* (size).

End Algorithm

Algorithm New_Rouhness

1. Compute the variance of the feature. This yields *variance.*
2. Algorithm Average_Roughness. This yields *average_roughness.*
3. Compute:

$$new\_roughness = \frac{variance}{average\_roughness}$$

End Algorithm

Suppose a feature has a low *average_roughness* but a high *variance*. That means locally, the feature is uniform; but globally, the feature has a larger intensity range and wider intensity distribution. This implies two possible conditions: (1) the feature has a significant shift in intensity values from one part of the feature to another part, or (2) the feature is patchy (for example, a checker board pattern).

E. Eccentricity

Traditionally, one measures the circularity or roundness of a feature using the following algorithm.

Algorithm Roundness

1. Identify all boundary pixels of the feature.
2. Compute the centroid of the feature $\langle \mu_x, \mu_y \rangle$.
3. For each boundary pixel *p*, compute its distance from the centroid:

$$D\left(p, \langle \mu_x, \mu_y \rangle\right) = \sqrt{\left(x_p - \mu_x\right)^2 + \left(y_p - \mu_y\right)^2}$$

4. Compute:

$$roundness = std\_dev\left[D\left(p, \langle \mu_x, \mu_y \rangle\right)\right]$$

End Algorithm

In the above algorithm, the roundness measure is the standard deviation of all the distances between a boundary pixel and the centroid of the feature. This is an aggregate measure of the "degree" of roundness. Once again, we propose a new feature that targets the existence of eccentricity—whether there is a portion of the boundary that does not conform to the roundness of the feature. We define that a feature is eccentric if it has boundary pixels that are close to its centroid and boundary pixels that are far from its centroid.

Algorithm Eccentricity

1. Identify all boundary pixels of the feature.
2. Compute the centroid of the feature $\langle \mu_x, \mu_y \rangle$.
3. For each boundary pixel *p*, compute its distance from the centroid:
$$D(p, \langle \mu_x, \mu_y \rangle) = \sqrt{(x_p - \mu_x)^2 + (y_p - \mu_y)^2}$$
4. Identify the maximum distance, $d_{\max}$, and the minimum distance, $d_{\min}$.
5. Compute:
$$eccentricity = \frac{d_{\max}}{d_{\min}}$$

End Algorithm

Thus, we can see that a circle is not eccentric because all boundary pixels are equidistant from the centroid. An N-pointed star is eccentric, however, because the boundary pixels at its points are farther away from the centroid than the pixels at the valley between points.

Figure 5 shows an example related to eccentricity. Figure 5(a) is basically a round object while Figure 5(b) is round except for a crack extending to its centroid from its perimeter. The traditional roundness measure does reflect this crack, as the value rises from 0.32 to 0.77. However, the eccentricity value is able to illuminate this crack further, with an increase of 2.55 times, from 1.24 to 3.16. For a larger object, the difference would be even larger as the large boundary would prevent the roundness to increase too much due to a small crack in the boundary.

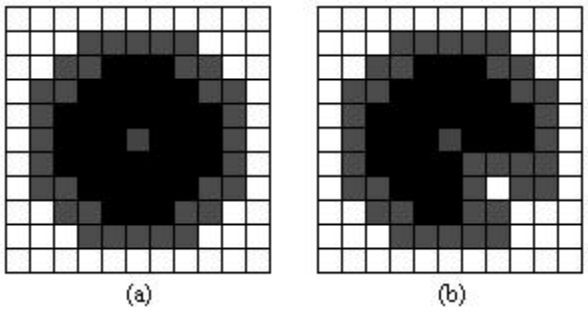

Figure 5 Black and red pixels are object pixels. Red pixels are boundary pixels. Blue pixels are centroids. (a) *roundness* = 0.32, *eccentricity* = 1.24, and (b) *roundness* = 0.77, *eccentricity* = 3.16.

Note also that the eccentricity measure introduced here is different from that described in [14]. In [14], the eccentricity measure is based on the best-fit ellipse which takes into account the average shape of the feature.

G. Jaggedness

This measures how jagged the boundary of a feature is. The underlying approach to this descriptor is the traversal of the chain code of the boundary of the feature. As we traverse the boundary, we take the absolute difference between the previous direction that brought us to the current pixel and the current direction that brings us to the next pixel. In a 3×3 8-connected neighborhood, the maximum change in directions is 4; the minimum is 0. As we move along the boundary, we add the magnitude of each directional difference to a counter. *jaggedness* is defined to be the average of all such direction changes, i.e., the value of the counter divided by *outer_perimeter*. The higher the value of *jaggedness*, the more jagged the boundary of the feature.

Algorithm Jaggedness

1. Given the chain code of the outer perimeter (boundary), set *prev_dir* to the first direction value, set *curr_dir* to the second direction value, and initialize *sum_diff* to 0.
2. Do until the *curr_dir* is the last direction value:
   a. *diff* = | *prev_dir* – *curr_dir* |
   b. add *diff* to *sum_diff*
   c. *prev_dir* = *curr_dir*
   d. *curr_dir* = next direction value
3. Compute:

$$jaggedness = \frac{sum\_diff}{outer\_perimeter}$$

End Algorithm

Figure 6 shows some examples of corners on the boundary. As one can see, Figure 6(a) is not a corner and thus it does not contribute to the jaggedness of the boundary. However, Figure 6(f) is a sharp, 180-degree-turn corner and adds the maximum difference of 4 to the boundary's jaggedness. Note that if the difference in the row and column coordinates were used as the basis for the jaggedness measures, we would not be able to distinguish Figure 6(f) as a sharp corner.

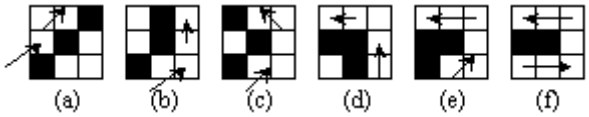

Figure 6 Examples of jaggedness. The center pixel is the current boundary pixel. The arrows show the direction of the boundary. (a) previous direction = 1, next direction = 1, difference = 0, (b) previous direction = 1, next direction = 2, difference = 1, (c) previous direction = 1, next direction = 3, difference = 2, (d) previous direction = 2, next direction = 4, difference = 2, (e) previous direction = 1, next direction = 4, difference = 3, (f) previous direction = 0, next direction = 4, difference = 4.

IV APPLICATION AND DISCUSSIONS

We have implemented and incorporated the above algorithms in ARKTOS [21, 24], an intelligent sea ice classification. ARKTOS (Advanced Reasoning using Knowledge for Typing Of Sea ice) performs fully automated classification of sea ice images by mimicking the reasoning process of sea ice experts and photo-interpreters; that is, feature-based analysis. We use features in a sea ice image as the visual cues about which experts reason to classify and use rules to represent how experts apply their knowledge. To extract features, we use a Watershed merging algorithm [9] to segment the image to obtain individual objects. Then, we describe each feature, computing a set of measurements such as area, roundness, jaggedness, and so on. Currently, there are about 30 different measurements computed for each feature. Subsequently, to classify a feature, we use a Dempster-Shafer rule-based system in which each rule acts upon a set of facts associated with the feature. Hence, these measurements or descriptors previously computed must be converted to discrete, symbolic facts. Here we discuss some of the heuristics that we use to convert measurements for the descriptors introduced in Section III to facts. Also, some descriptors are combined to give birth to *composite* facts. We also use thresholds as the discretization points. In the following discussion, a threshold of a descriptor is denoted as $T_{descriptor}$.

A. Lead

First, we determine whether a feature is an *ice lead*. To be considered as an ice lead, a feature must be elongated and irregular. Note that *elongation* is computed as the ratio of *max_length* over *max_width* of the feature.

Heuristic Lead

If $elongation > T_{elongated}$ and $irregularity > T_{irregular}$ then *lead=true*
else *lead=false*.

Given the above heuristic, we see a lead as a feature that is both elongated and irregular.

Figure 7 shows an example of an ice floe whose porosity measures are close to 1.0. This floe feature was classified as old ice with a confidence level of 0.99. Figure 8 shows an example of an ice lead whose area porosity measure is much greater than 1.0, significantly contributing to its irregularity. This lead feature was classified as first year ice with a confidence level of 0.66.

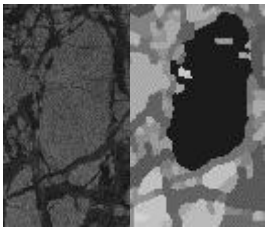

Figure 7 (a) A portion of an original image. Copyright ESA. (b) The segmented image with feature #1535 colored blue. Measurements: area = 2321, perimeter = 251, outer_perimeter = 208, orientation = 0.1974, maximum length = 81.21, maximum width = 42.95, area_porosity = 1.54, perimeter_porosity = 1.21, irregularity = 1.86.

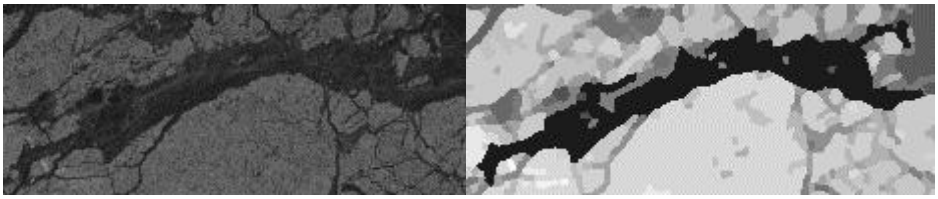

Figure 8 (a) A portion of an original image. Copyright ESA. (b) The segmented image with feature #2264 colored blue. Measurements: area = 4012, perimeter = 724, outer_perimeter = 644, orientation = -0.2253, maximum length = 226.20, maximum width = 57.27, area_porosity = 3.23, perimeter_porosity = 1.13, irregularity = 3.63.

B. Blob

Second, we determine whether a feature is a *blob*. A blob is a very large, shapeless feature. This could be due to (1) a large area of old ice covering most of the image, with holes within the area, or (2) an image artifact due to poor image quality that hinders effective extraction of individual features. This is based on the measurements *area*, *irregularity*, and *eccentricity*. To obtain the fact, we have:

Heuristic Blob
If $area > T_{size,2}$ and ($irregularity > T_{irregular}$ or $eccentricity > T_{eccentric}$) then *blob=true*
else *blob=false*.

Figure 9 shows an example of a blob whose perimeter porosity measure is above 2.0, significantly contributing to its irregularity. This blob was classified as old ice with a confidence level of 0.98.

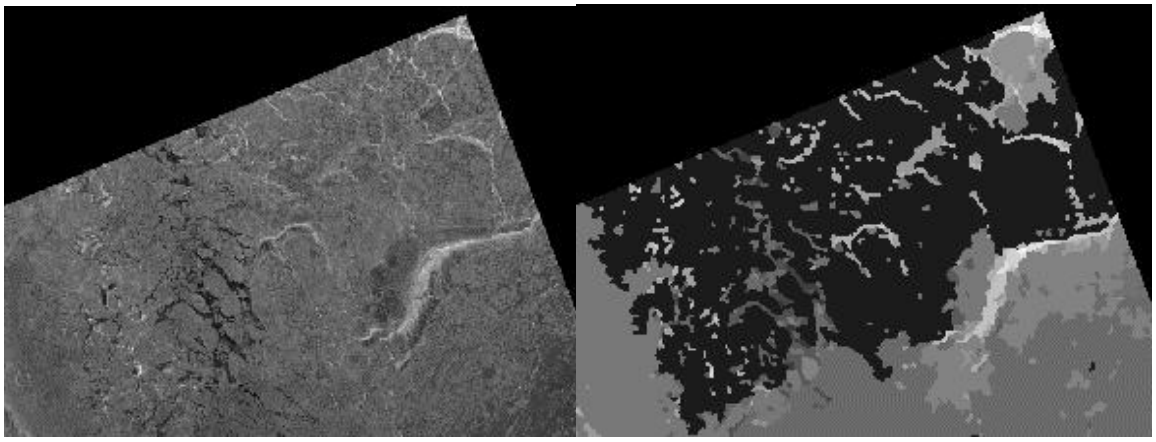

Figure 9 (a) A portion of an original image. Copyright ESA. (b) The segmented image with feature #22 colored blue. Measurements: area = 72025, perimeter = 7213, outer_perimeter = 3577, orientation = -0.3722, maximum length = 453.86, maximum width = 255.37, area_porosity = 1.61, perimeter_porosity = 2.02, irregularity = 3.25.

Note that due to the ability of the two porosity measures to capture the inherent "porosity" characteristics of the features, they play a significant role here in helping classify the images in Figures 7, 8, and 9.

C. Mottled or Smooth

This fact denotes the surface mottledness or smoothness of a feature. It is based on the measurement *mottledness*.

Heuristic Mottled/Smooth
`If` $mottledness > T_{mottled}$ `then mottled=true else smooth=true.`

Note that instead of using the fact mottled=false, we use smooth=true because the latter is more intuitive. Figures 10 and 11 show two examples of mottledness at work. Figure 10 shows an ice object with a mottled surface. This fact triggers directly three classification rules that allow ARKTOS to determine that the object is an old ice with a confidence of level of 0.998. Figure 11 shows an object with a smooth surface. The fact that it is smooth triggers directly two rules resulting in a classification of first year ice with a confidence level of 0.67.

Note that if we take the ratios of the two sets of measurements for average_roughness, new_roughness, and mottledness, we have, respectively: 1.13, 1.38, and 2.38. The ratio for the mottledness is the greatest among the three descriptors. This has a very useful application. This means that the difference between a smooth object and a mottled object, along the dimension of mottledness, is large and distinguishable. This allows experts and software designers more flexibility in determining the value of $T_{mottled}$ that converts the measurements into facts.

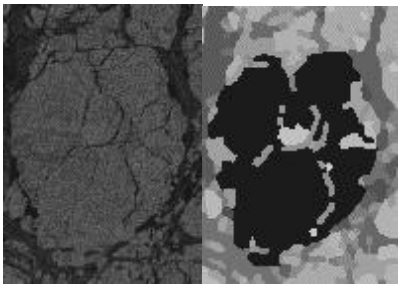

Figure 10 (a) A portion of an original image. Copyright ESA. (b) The segmented image with feature #595 colored blue. Measurements: area = 5673, average_intensity = 80, average_roughness = 59.68, new_roughness = 0.1816, mottledness = 31.37.

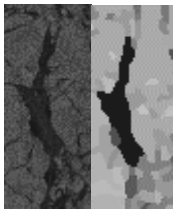

Figure 11 (a) A portion of an original image. Copyright ESA. (b) The segmented image with feature #449 colored blue. Measurements: area = 502, average_intensity = 41, average_roughness = 52.94, new_roughness = 0.1320, mottledness = 13.18.

D. Irregular

This denotes whether a feature has an irregular shape. In addition to *irregularity*, we also use *eccentricity* to help define this attribute. This is thus also a composite fact. Two thresholds are used.

Heuristic Irregular

If ($irregularity > T_{irregular}$ or $eccentricity > T_{eccentric}$) then irregular=true
else irregular=false.

Figure 12 shows an example of regular object, while Figure 13 shows an example of an irregular object. The object is Figure 12 is subsequently classified as old ice by ARKTOS with a confidence level of

0.9995, while that in Figure 13 as first year ice with a confidence level of 0.97. Note that since both features do not have too many holes and protrusions, the irregularity measurements do not play as significant a role as do the eccentricity measurements. The percentage of difference in the former is only 32.6% while the latter is 237.8%.

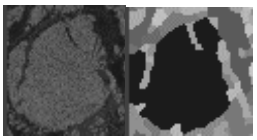

Figure 12 (a) A portion of an original image. Copyright ESA. (b) The segmented image with feature #1217 colored blue. Measurements: area = 1847, average_intensity = 89, perimeter = 224, outer_perimeter = 224, orientation = 0.3662, maximum_length = 59.30, maximum_width = 54.41, elongation = 1.09, eccentricity = 2.35, irregularity = 1.75.

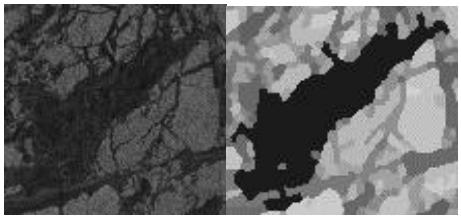

Figure 13 (a) A portion of an original image. Copyright ESA. (b) The segmented image with feature #1922 colored blue. Measurements: area = 3040, average_intensity = 40, perimeter = 413, outer_perimeter = 400, orientation = -0.7598, maximum_length = 128.45, maximum_width = 53.23, elongation = 2.41, eccentricity = 7.94, irregularity = 2.32.

E. Jagged

This denotes whether the boundary of a feature is jagged. It is based on the measurement *jaggedness*. One threshold is used:

Heuristic Jagged
`If` $jaggedness > T_{jagged}$ `then jagged=true else jagged=false.`

Figure 14 shows an object with a jagged boundary. This fact helps ARKTOS in classifying the object as first year ice with a confidence level of 0.62. Figure 15 shows an object with a non-jagged boundary. ARKTOS classifies the object as old ice with a confidence level of 0.996.

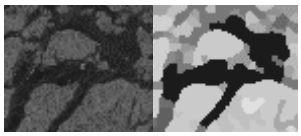

Figure 14 (a) A portion of an original image. Copyright ESA. (b) The segmented image with feature #2030 colored blue. Measurements: area = 1138, perimeter = 284, outer_perimeter = 284, jaggedness = 0.9648.

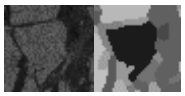

Figure 15 (a) A portion of an original image. Copyright ESA. (b) The segmented image with feature #129 colored blue. Measurements: area = 450, perimeter = 82, outer_perimeter = 82, jaggedness = 0.7195.

F. Thresholds

Table 1 shows the threshold values that have been determined by sea ice experts during the refinement and evaluation phase of the ARKTOS project.

Descriptor	**Possible Facts**	**Thresholds Used**
lead	`lead=true`, `lead=false`	Observes values between 1.0 and 25.0. $T_{irregular} = 3.10$, Observes values between 1 and 4. $T_{elongated} = 1.3$
blob	`blob=true`, `blob=false`	$T_{size,2} = 25{,}000$, $T_{irregular} = 3.10$ Observes values between 2.0 and 20.0. $T_{eccentric} = 4.50$
mottled or smooth	`mottled=true`, `smooth=true`	Observes values between 5.0 and 60.0. $T_{mottled} = 31.0$
irregular	`irregular=true`, `irregular=false`	$T_{irregular} = 3.10$, $T_{eccentric} = 4.50$
jagged	`jagged=true`, `jagged=false`	Ranges between 0 and 4. $T_{jagged} = 0.74$

Table 1 Facts, values, and thresholds used in ARKTOS.

V CONCLUSIONS

We have introduced a set of new sea ice descriptors for SAR images: perimeter_porosity, area_porosity, irregularity, mottledness, eccentricity, and jaggedness. For each descriptor, we have specified an algorithm for its design and also demonstrated its characteristics through examples and comparisons with other traditional image processing techniques. In general, we have found that traditional descriptors tend to focus on the overall, "aggregate" sense of a feature's shape, which is not always applicable to remote sensing applications such as sea ice classification. For example, in sea ice, an object that exhibits a small

portion of ridged surface is considered to be textured even if a large portion of its surface is smooth. Thus, our designs have utilized the "existential" sense of a feature's shape, emphasizing extrema more than average. We have also applied these new descriptors in ARKTOS, an intelligent sea ice classification system. Using these descriptors, we are able to generate symbolic, composite facts such as `leads=true` and `blob=true`, and simple facts such as `mottled=false`, `smooth=true`, `irregular=false`, and `jagged=true`. Subsequently, ARKTOS fires its classification rules matching these facts, and thus classifies each feature with a particular confidence level. We have shown through actual examples from our sea ice research how these features help in the classification process. These feature descriptors may also be generalized to other remote sensing domains and applications.

VI ACKNOWLEDGMENT

The ARKTOS project was conducted at the University of Kansas, with PI Dr. Costas Tsatsoulis. The research work was supported in part by a Naval Research Laboratory research grant, contract number N00014-95-C-6038.

VII REFERENCES

[1] J. D. Banfield and A. E. Raftery, "Ice floe identification in satellite images using mathematical morphology and clustering about principal curves," *JASA*, vol. 87, no. 417, pp. 7-16, 1992.

[2] B. A. Burns, D. J. Cavaileri, M. R. Keller, W. J. Campbell, T. C. Grenfell, G. A. Maykut, and P. Gloersen, "Multisensor comparison of ice concentration estimates in the marginal ice zone," *J. Geophysical Res.*, vol. 92, pp. 6843-6856, 1987.

[3] W. J. Campbell, P. Gloersen, E. G. Josberer, P. S. Johannessen, P. S. Guest, N. Mognard, R. Shuchman, B. A. Burns, N. Lannelongue, and K. L. Davidson, "Variations of mesoscale and large scale sea ice morphology in the 1984 marginal ice zone experiment as observed by microwave remote sensing," *J. Geophysical Res.*, vol. 92, pp. 6805-6824, 1987.

[4] J. Daida, R. Samadani, and J. F. Vesecky, "Feature-oriented feature-tracking algorithms for SAR images of the marginal ice zone," *IEEE Trans. Geosci. & Remote Sensing*, vol. 28, no. 4, pp. 573-589, 1990.

[5] M. Fily and D. A. Rothrock, "Opening and closing of sea ice leads: digital measurements from synthetic aperture radar," *J. Geophysical Res.*, vol. 95, no. C1, pp. 789-796, 1990.

[6] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Transactions*, vol. 10, pp. 260-268, 1961.

[7] H. Freeman, "Computer processing of line-drawing images," *ACM Computing Surveys*, vol. 6, pp. 57-97, 1974.

[8] M. M. Galloway, "Texture analysis using gray level run lengths," *Comput. Graph. Image Processing*, vol. 4, pp. 172-179, 1975.

[9] J. M. Gauch and S. M. Pizer, "Multiresolution analysis of ridges and valleys in grey-scale images," *IEEE Trans. Pattern Analysis & Machine Intell.*, vol. 15, no. 6, pp. 635-646, 1993.

[10] R. T. Hall and D. A. Rothrock, "Photogrammetric observations of the lateral melt of sea ice floes," *J. Geophysical Res.*, vol. 92, no. C7, pp. 7045-7048, 1987.

[11] R. M. Haralick, K. Shanmugan, and I. H. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 610-621, May 1973.

[12] D. Haverkamp, C. Tsatsoulis, and S. Gogineni, "The combination of algorithmic and heuristics methods for the classification of sea ice imagery," *Rem. Sensing J.*, vol. 9, no. 2, pp. 135-159, 1994.

[13] D. Haverkamp, L.-K. Soh, and C. Tsatsoulis, "A comprehensive, automated approach to determining sea ice thickness from SAR data," *IEEE Trans. Geosci. & Remote Sensing*, vol. 33, no. 1, pp. 46-57, 1995.

[14] A. K. Jain, *Fundamentals of Digital Image Processing*, Englewood, CA: Prentice Hall, 1989.

[15] R. Korsnes, "Quantitative analysis of sea ice remote sensing imagery," *Int. J. Rem. Sensing*, vol. 14, no. 2, pp. 295-311, 1993.

[16] R. M. Lark, "Geostatistical description of texture on an aerial photograph for discriminating classes of land cover," *Int. J. Remote Sensing*, vol. 17, no. 11, pp. 2115-2133, 1996.

[17] G. O. Lendaris and G. L. Stanley, "Diffraction pattern sampling for automatic pattern recognition," *Proc. IEEE*, vol. 58, pp. 198-216, Feb, 1970.

[18] S. Loncaric, "A survey of shape analysis techniques," Pattern Recognition, vol. 31, no. 8, pp. 983-1001, 1998.

[19] D. A. Rothrock and A. S. Thorndike, "Measuring the sea ice floe size distribution," *J. Geophysical Res.*, vol. 97, no. C11, pp. 17729-17738, 1984.

[20] A. J. Sephton, L. M. J. Brown, J. T. Macklin, K. C. Partington, N. J. Veck, and W. G. Rees, "Segmentation of synthetic-aperture radar imagery of sea ice," *Int. J. Rem. Sensing*, vol. 15, no. 4, pp. 803-825, 1994.

[21] L.-K. Soh, "ARKTOS: An intelligent system for satellite sea ice image analysis," *TR-CSE-UNL-06-13-2002*, Department of Computer Science and Engineering, University of Nebraska, 2002.

[22] L.-K. Soh, "Image processing techniques for describing sea Ice features," *Proc. 2002 Int. Geosci. & Remote Sensing Symp. (IGARSS'02)*, Toronto, Canada.

[23] L.-K. Soh and C. Tsatsoulis, "Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices," *IEEE Trans. Geosci. & Rem. Sensing*, vol. 37, no. 2, pp. 780-795, 1999.

[24] L.-K. Soh and C. Tsatsoulis, "ARKTOS: a knowledge engineering software package for satellite sea ice," in *Proc. 2000 Int. Geosci. & Remote Sensing Symp. (IGARSS'00)*, Honolulu, Hawaii, pp. 696-698, 2000.

[25] 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.), Berlin: Springer-Verlag, pp. 9-34, 2000.

[26] M. Steele, "Sea ice melting and floe geometry in a simple ice-ocean model," *J. Geophysical Res.*, vol. 99, pp. 22413-22424, 1992.

[27] C. C. Venters, and M. Cooper, "A review of content-based image retrieval systems," *JISC Technology Applications Programme (JTAPS) Technical Report 054*, University of Manchester, UK, 2000.

[28] J. F. Vesecky, R. Samadani, M. P. Smith, J. M. Daida, and R. N. Bracewell, “Observation of sea-ice dynamics using Aynthetic Aperture Radar images: automated analysis,” *IEEE Trans. Geosci. & Remote Sensing*, vol. 26, no. 1, pp. 38-48, 1988.

[29] L. Wang and D.-C. He, “A new statistical approach for texture analysis,” *Photogramm. Eng. Remote Sensing*, vol. 56, pp. 61-66, 1990.