

Toward an Automated Ship and Wake Detection System

C.C. Wackerman^a, W. Pichel^b, X. Li^b and C. Jackson^b

^aGeneral Dynamics AIS, P.O. Box 990, Ypsilanti, MI, USA 48197

email: chris.wackerman@gd-ais.com

^bNOAA/NESDIS, Camp Springs, MD, USA

ABSTRACT

An automated, constant false alarm rate, detection algorithm for a ship hard target signature in a SAR image is presented based on a K-distribution model for the background clutter. It is validated against a number of SAR images that contained ships at known locations, many with known lengths. The algorithm detects ~90% of all the ships, although it is shown that some of the ships are not actually where they should be and thus account for some of the missed detection. Ships with lengths as small as 10m are detected with a probability of false alarm of ~1E-8.

A ship wake detection algorithm is also presented based on a supervised classification algorithm that finds the optimal linear separation between classes for classification. It has not yet been validated, but a limited number of examples are shown.

Keywords: SAR, automated ship detection, automated wake detection

1 INTRODUCTION

Under the Alaska SAR Demonstration Project, General Dynamics has been developing automated tools to detect and classify ships from commercial synthetic aperture radar (SAR) imagery. The goal is to automatically generate a list of ship locations (in latitude, longitude) and ship lengths, then post this onto a web site for use in fisheries management and coastal monitoring. As is well known, ships in SAR imagery often appear as bright dots against the ocean clutter background and in the case of fisheries can often have dark wakes behind them (usually caused by the damping of the small ocean waves by fish products disposed of by the processing ship). Figure 1 shows an example of these where a blow-up of a region containing fishing vessels is shown on the right that contains both hard target returns (the bright dots) and a slick wake (the dark, curved, band).

The approach we have taken to ship detection for the hard target is to determine if a pixel response in the SAR image is statistically different than the ocean background clutter, and if so to flag it as a target pixel. We then combine target pixels together that come from a single ship to generate the final ship signatures. We assume that the ocean background clutter in a SAR image can be modeled by a K-distribution probability density function and find the background statistics we need to characterize the K-distribution parameters by moving a local window through the SAR image, eliminating potential target pixels from the window, then calculating the statistics. This allows the background statistics to vary over large spatial scales within an image while still allowing us to keep a constant probability of false alarm (PFA) for the whole image. A description of the algorithm is given in Section 2, and in Section 3 we show how well the algorithm has performed against ground truth test sets.

The approach we have taken to wake detection is based on work we did for automatically finding oil spills in a SAR image. It is essentially a supervised classification algorithm based on feature vectors derived from image regions. We first threshold the image to find all of the “dark” pixels. We then form “blobs” from connected dark pixels and characterize the blobs using their shape, edge contrast, variation in the edge

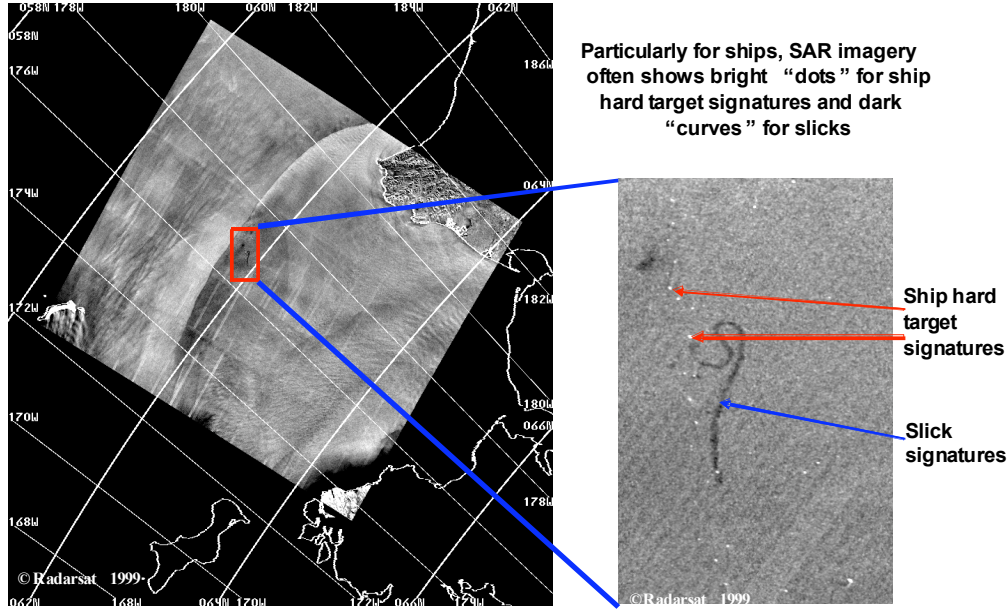


Figure 1: Example of ship hard target returns (white dots) and ship wake signatures (dark bands) from a blow-up (right) of a fishing fleet in a SAR image (left).

location, etc. These blob characteristics then become the elements of the feature vector, and we find the projection of these vectors that maximize the distance between them (as measured by the Fischer Discriminant). This algorithm is described in Section 4, and results are shown. Section 5 then summarizes the paper.

2 ALGORITHM FOR DETECTING THE SHIP HARD TARGET

The algorithm is based on the assumption that the background ocean SAR image samples are distributed with a probability density function known as the K-Distribution [1] and is derived from an earlier approach based on the ratio of means to standard deviations [2]. The K-distribution is derived by assuming that the radar cross section of the material (in this case the ocean) varies spatially in a random way that can be characterized by the gamma distribution. These radar cross section values are then imaged by a SAR, which imposes multi-looked speckle noise onto the radar cross section values. Multi-looked speckle can also be described as a gamma distribution. By putting these processes together, the variations of the SAR image intensities, I , can be shown to be distributed according to the K-Distribution

$$p(I) = \frac{2}{I} \left(\frac{LvI}{\langle I \rangle} \right)^{(L+v)/2} \frac{1}{\Gamma(L)\Gamma(v)} K_{v-L} \left[2 \left(\frac{LvI}{\langle I \rangle} \right)^{1/2} \right] \quad (1)$$

where $\langle I \rangle$ is the mean image intensity, the symbol $\langle \rangle$ represents the expected value of what is inside, L is the number of looks in the SAR sensor, v represents the order parameter of the gamma process that describes the underlying radar cross section fluctuations (the variance of the radar cross section values is $1/v$), Γ is the gamma function, and K_j represents a modified Bessel function of integer order j . Note that the K-Distribution in Eq. (1) is a three parameter density function; $\langle I \rangle$, L and v must be specified.

It can be shown [1] that the moments about the origin of the K-Distribution are

$$\langle I^m \rangle = \langle I \rangle^m \frac{\Gamma(L+m)\Gamma(v+m)}{L^m v^m \Gamma(L)\Gamma(v)}. \quad (2)$$

Some authors use a two parameters K-distribution model which can be derived from Eq. (2) by letting $L=1$ and making the substitution

$$b = \frac{2\sqrt{v}}{\sqrt{\langle I \rangle}} \quad (3)$$

to form the density function

$$p_o(I) = \frac{b}{\sqrt{I}\Gamma(v)} \left(\frac{b\sqrt{I}}{2} \right)^v K_{v-1}(b\sqrt{I}) \quad (4)$$

In what follows we will use the more general three parameters form in Eq. (1).

We will remove one parameter by dividing all the statistics by the mean of the background. This has the added advantage of removing any dependencies on scale factor changes between images. Thus without loss of generality we can set $\langle I \rangle = 1$ in Eqs. (1) and (2) and be down to a two parameter problem; L and v . Also note that interchanging L and v in Eqs. (1) and (2) makes no difference, so also without loss of generality we can assume that $v \geq L$. We will make the further assumption that L and v are integers so that we do not need to deal with non-integer order modified Bessel functions.

Thus under our assumptions we can parameterize the background statistics using the integers L and v . For a given probability of false alarm (PFA) we can loop over various values of L and v and calculate the threshold required to achieve that PFA by numerically integrating Eq. (1), and we can determine the second and third order moments that the background statistics need to have using Eq. (2). All of this can be calculated once and stored for all time. Thus we have generated threshold tables for PFA = 1.0e-10, 1.0e-09, 1.0e-08, 1.0e-07, 1.0e-06 and 1.0e-05. For each PFA we generate 3 tables: (1) the threshold that generates the given PFA; (2) the normalized second moment about the origin ($E[x^2] * 0.5/E[x]$); and (3) the normalized third moment about the origin ($E[x^3] * 0.3333/E[x]$). These tables are generated by looping over values of L and v , calculating the normalized statistics for those values of L and v , and then numerically integrating the density function to generate the threshold. We do not loop evenly over values of L and v since we need to be more finely sampled for small values than for larger ones. Thus we increment L and v by 1 in the range from 1 to 10, and then every 2 thereafter.

We should note that some care needs to be taken in performing the numerical integration of the density function in Eq. (1). If we just use the standard routines for modified Bessel functions (such as are in Numerical Recipes) we run into some significant problems in getting down to such small PFA values. In essence, the Bessel functions for such large values of L , v , and I are so small that they get set to zero by the approximations, whereas the gamma function terms are still quite large and would offset the small Bessel function values. We have countered this by using approximations not to $K_j(x)$ but rather to $\exp(x)K_j(x)$ and then putting a term $(\exp[-x])^{2/(L+v)}$ inside of the first set of parenthesis in Eq. (1). We also calculate the gamma functions and the term inside the first set of parenthesis together so that as the gamma function terms get larger, they are offset by the smaller valued term within the parenthesis. Putting these together, we have re-written the density function as:

$$p(I) = \frac{1}{I} \prod_{k=1}^{L-1} \left(\frac{\alpha \exp\left[\frac{-2\alpha}{(L-1) + (v-1)}\right]}{L-k} \right) \prod_{k=1}^{v-1} \left(\frac{\alpha \exp\left[\frac{-2\alpha}{(L-1) + (v-1)}\right]}{v-k} \right) (\exp[2\alpha] K_{v-L}[2\alpha]) \quad (5)$$

where

$$\alpha = \sqrt{\frac{LvI}{\langle I \rangle}}, \quad (6)$$

the last term on the right is calculated with a polynomial approximation which does not get set to zero now because of the additional exponential term, and the product terms are well-behaved because the small exponential term offset the large α term. Implementation in the form of Eq. (5) has allowed us to generate accurate PFA values for very small numbers.

In general, the detection code proceeds as follows. For an image pixel the local background statistics $\langle I \rangle$, $\langle I^2 \rangle$ and $\langle I^3 \rangle$ are calculated (how this is done specifically is discussed below). The normalized statistics

$(\langle I^2 \rangle^{*0.5}) / \langle I \rangle$ and $(\langle I^3 \rangle^{*0.3333}) / \langle I \rangle$ are computed and compared to the entries in the L, v tables for these quantities (which were pre-calculated using Eq. (2)). The values of L and v that generate normalized statistics “closest” to these values (in the sense of minimizing the root-mean-squared error calculated over the two normalized statistics) are assumed to describe the statistical distribution of the background. The pre-calculated threshold for these values of L and v is then compared to the image pixel divided by $\langle I \rangle$. If this value is above the threshold then this pixel is assumed to be over a target. This is done for every pixel in the image. The individual pixels that were deemed to be over a target are then combined into single signatures. This is done by combining all detections that are within some distance constraint. The user specifies the largest size for a ship target, and then a window of slightly larger size is moved over the image. All detected pixels within the window are assumed to belong to a single target. This could cause a problem if two ships are closer than the assumed target size, but in practice this is hardly ever true. The mean and standard deviation of the energy over all the detected pixels in a target are calculated to generate a metric for the signature. In addition, the spatial shape of the target is calculated as follows. A line is fit to the locations of detections within the target, and the projection of the locations along this line determines the length of the target. The mean and standard deviations of the projections orthogonal to the best fit line is used to estimate the target width (the mean) and how jagged the target is (the standard deviation divided by the mean).

The background statistics are pre-calculated using large blocks within the image in order to capture potential spatial changes in the local statistics. The user specifies the block size to use to calculate the background statistics, the image is divided up into contiguous blocks of this size, and the normalized statistics calculated for each block. For higher resolution imagery we have found that large, bright ship signatures can in fact skew the background statistics within a block even for very large block sizes (300 X 300). To address this we median filter the image before calculating the statistics over the large block size. To save time we only perform a one-dimensional median filter on each image line where the size of the filter is determined by twice the largest ship we expect to find (we used 41 samples). Since we are only trying to remove very bright ship signatures, we compare the image sample in the center of the median filter window with the median. If the sample is larger than 5 times the median, we replace it with the median. Otherwise we leave it alone. This allows us to maintain the correct background statistics when there are not any bright ship targets. However, it also means that we are wasting effort in the sense of performing a median filter just to throw the median value away. Making this more efficient is an ongoing area of research.

In order to provide a smooth transition of background means and thresholds from image block to image block, we actually assign each image pixel a background mean and threshold that is derived from a bi-linear interpolation of the means and thresholds calculated from each block. We assign the middle pixel within each block to its mean and threshold and use the distance of the image pixel from the block centers to do the interpolation. For the edges of the image we just replicate the block statistics around the image edges.

In summary, the algorithm is as follows. The user determines the PFA to use, and inputs the pre-calculated threshold values for this PFA as well as the values of $(\langle I^2 \rangle^{*0.5}) / \langle I \rangle$ and $(\langle I^3 \rangle^{*0.3333}) / \langle I \rangle$. The image has its bright points removed via the one-dimensional “median filter and replace” operation described earlier. The image is then divided up into contiguous large blocks, the same normalized statistics are calculated for each block, the values of the pre-calculated statistics that are closest to the set of normalized statistics are determined, and their corresponding threshold assigned to this block of image values. A bi-linear interpolation is done on the background means and thresholds from each image block to assign a background mean and threshold for every pixel in the image. The pixel value of the original image (i.e. without the “median filter and replace” operation) is divided by the background mean and compared to its threshold. It is flagged as a detected pixel if it is larger than the threshold. Finally, the flagged image pixels are combined into single targets by assigning every flagged pixel within the user-defined ship-size box as coming from one target, and the characteristics of the target (energy and shape) are outputted into an ascii file. Figure 2 graphically illustrates the algorithm approach.

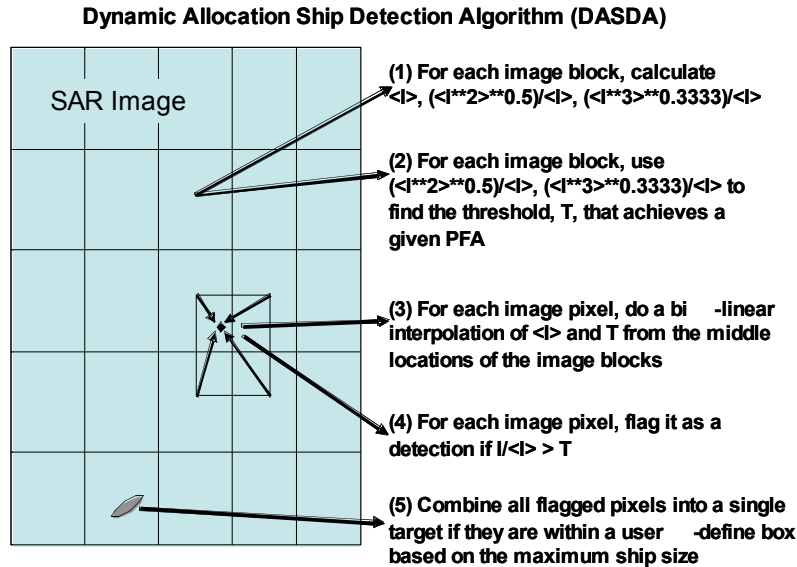


Figure 2: Flowchart for the ship hard target detection algorithm.

This algorithm has a number of advantages:

- (a) it is truly a CFAR algorithm and uses the K-distribution to describe the background responses;
- (b) it only uses image pixels that are over the target to characterize the target and therefore utilizes all of the target energy without any contamination from background pixels;
- (c) it allows an estimate of target shape in addition to detecting the target, which is the first step toward classification; and
- (d) it actually runs quite fast due to all the pre-calculation that can be done before analyzing each image pixel.

3 SHIP DETECTION ALGORITHM PERFORMANCE ESTIMATION

There was a European program being run by the Joint Research Center called Detection, Classification and Identification of Marine Traffic From Space (DECLIMS) whose purpose was to combine and analyze automated algorithms for detecting vessels in remote sensing imagery. To look at SAR algorithms, two test set were compiled. The first benchmark test set consisting of 10 SAR images from the Canadian RADARSAT sensor which contained 149 ships with known locations (although other ships were also in the imagery) of which 112 had known ship lengths. Almost all of the images had a spacing of 25m, but one image had a spacing of 12.5m. The second benchmark consisted of 12 images containing 175 ships with known locations, of which 151 had known lengths. Most of these images had a sample spacing of 12.5m, but some had 6.25m.

The ship detection algorithm was run on both data sets. Because the images contained more ships than were known in the ground truth, the first step was to connect detections with known targets. To do this, we made a list of all the detections in an image and a list of all the known target locations. We then calculated the distance from each detection to each target. In this array of distances we found the smallest distance between a target and a detection, attached this detection to that target, removed that line and column from the array, and repeated the process. In this way, we attached detections to the targets that were closest to them throughout any image. However, since there were more detections than targets in any image, this process always attached some detection to every target. What determined whether that detection really was that target was the distance between them. We had to allow some distance between targets and detections due to the fact that there was an unknown time difference between when the ship reported its location and the SAR image was collected, there was some error in the ship location information, and there was some error in the SAR image registration accuracy. To determine what the threshold on this distance should be

to connect a detection to a target, we examine two approaches. First, we looked at all the ships that had lengths larger than 80 m (i.e. that should be very obvious in the image) and determined the largest distance for these ships between targets and detections (for ships that we could manually find, see below). This generated about 3500m for the distance threshold. Second, we generated histograms of the number of ships detected versus the threshold on the distance between detections and targets and determined when these curves flattened out (i.e. we detected very small amounts of new ships for larger distances). This also generated approximately 3500m, so this was the threshold we used. If the target and detection would closer than this, we determined that the target had been detected. This resulted in detecting 94% of all the known ships in BMK1, and 87% of all the known ships in BMK2.

However, we also know that ship length can be an important parameters, since ships that are much smaller than the SAR image sample spacing may not be detectable in the image. Thus for the ships with known lengths, we binned them into bins of 10m lengths and determined the percentage of ships detected within each bin. These results are shown in Figure 3 for both BMK1 (red line) and BMK2 (black line). For BMK1, we detect all but one of the ships whose lengths are greater than 30m. However we only detect about half of the ships that have lengths between 20 and 30m; just around the sample spacing of most of the imagery. This indicated to us that the algorithm did well if the ship length was greater than the image sample spacing.

For BMK2, we missed 8 ships whose lengths were greater than 80m. This was a little surprising, since in BMK2 the image sample spacing was 12.5 m or less. When we looked at the image locations for these 8 ships, there was no visually obvious ship signature anywhere near the locations where the ships were supposed to be. Although we can not say definitely that these ships are missing, it does appear that they may not have been where they were suppose to be since they were such large ships and should have been readily visible in the imagery. If we assume that these 8 ships are not really there (in which case our overall detection performance becomes 90%), then in BMK2 we detected all of the ships with lengths greater than 30 m. Similar to BMK1, we see a decrease in performance for the 20-30m ship lengths but not nearly as much as in BMK1. Then, interestingly, we get improved performance for the smaller ships; detecting all of the ships with lengths less than 10 m (although this may be a result of having only 3 such ships). In general though, it appears that with the smaller image sample spacing in BMK2 we do get

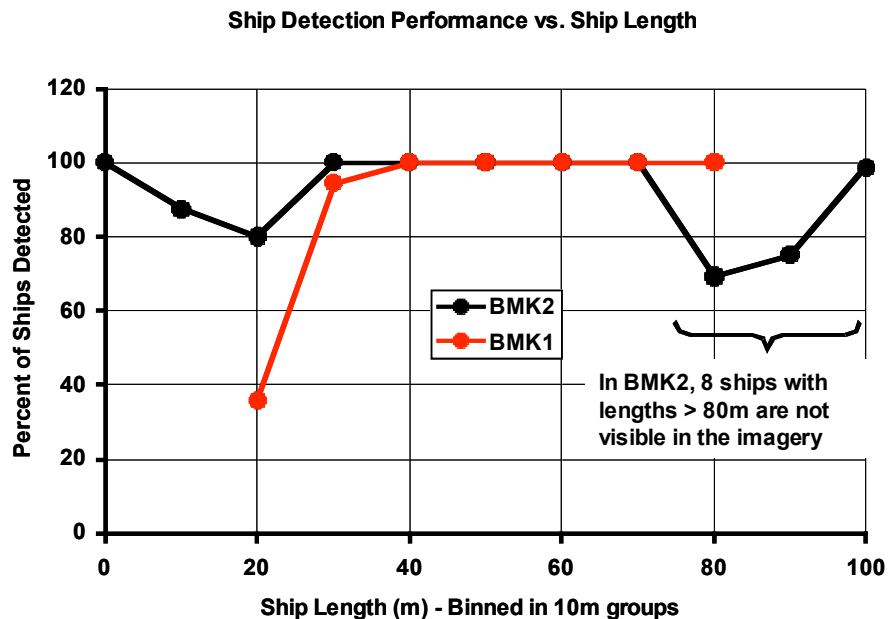


Figure 3: Detection results for the two DECLIMS benchmarks (BMK1, red; BMK2, black). For each benchmark the ground truth is binned based on ship lengths into bins of 10m each. The percent of ships detected in each length bin are plotted. Note that in BMK2 some very large (>80m in length) ships are not visually evident in the image, and we conclude that these ships are not in their given locations. This may also be true for the smaller ship results, unfortunately we can not determine this visually.

improved performance in detection of the smaller ships, and in fact appear to be able to detect ships robustly down to the sample spacing of the imagery (6-12m). However, since we know that some of the larger ships are not where they should be, it is possible that some of the smaller ships are also not where they should be, making these results very conservative in terms of performance.

It is also important to examine the probability of a false alarm (PFA) for these detection results, since we could detect all of the ships by having a very high PFA. To generate estimates of PFA, we generated images where we embedded a white triangle just below the location of a detection (so that the tip of the triangle should point at the detection) and then manually examined whether there were any triangles for which a user would have said that there was no ship above it. Overall, for individual images in BMK1 we estimated PFA to be from 2E-09 to 1E-08. This was for a setting on the K-Distribution thresholds that theoretically achieved a PFA of 1E-10. However, many of the image sizes were such that a single false alarm in an image would generate a PFA of 1E-08, so that we could generate either 0 or 1E-8 for manually generated PFA; nothing in between. These results are therefore consistent with a setting of 1E-10, but the images sizes were too small to verify that this was actually what we were achieving.

4. SHIP WAKE DETECTION ALGORITHM AND EXAMPLES

The ship wake detection algorithm approach was based on previous work done for ice classification in the marginal ice zone [3] and for terrain classification using dual-antenna airborne SAR systems [4]. It is a supervised classifier that first gets trained on examples of the types of classes to be generated in order to create a series of classification vectors, then applies these classification vectors to separate the image regions into classes.

Assume that there are N classes desired, and the user has specified a window size, $M_1 \times M_2$, that will be used to classify the image. The window must be big enough to contain the tone and texture information that can differentiate between the classes, yet small enough to generate sufficient spatial resolution for the resulting classification. The user has also specified a series of algorithms that are applied to the image samples within the $M_1 \times M_2$ window and that generate measures of the tonal or texture information within the window. That is, each algorithm inputs the image values within the window and outputs a scalar quantity that measures, for example, the statistics, texture, or correlation properties of the image within the window. The scalar outputs from all of the algorithms are assembled into a feature vector, \vec{f} , that will be used to classify the samples within the window. From the training of the algorithm, we will generate the mean feature vector for each class: \vec{m}_k where k goes from 1 to N . Also from the training we will generate a classification vector for each pair of classes, \vec{c}_{kj} , and a scalar threshold value, T_{kj} , where k goes from 1 to N and j goes from 1 to N . The classification process is then as follows. For each pair of classes, the scalar p_{kj} is formed via

$$p_{kj} = (\vec{f} - \vec{m}_k) \cdot \vec{c}_{kj} \quad \text{for } k \neq j \quad (7)$$

where \cdot represents the vector dot product. If $p_{kj} < T_{kj}$ for all $j = 1$ to N , $j \neq k$, then the image samples within the window are assign to class k . In essence, the algorithm assumes that in the space of the image features represented by the feature vector \vec{f} , each class occupies a unique convex region. The classification vectors, \vec{c}_{kj} , define hyperplanes that separate pairs of classes, where the hyperplane is orthogonal to the classification vector and located at a distance T_{kj} from the location of the mean feature vector from class k along the classification vector. A feature vector is assigned to class k if it is on the ‘‘correct’’ side of each hyperplane between class k and each other class, which implies that p_{kj} from Eq. 1 is $< T_{kj}$ for all $j = 1$ to N , $j \neq k$. Note that we can always normalize the length of \vec{c}_{kj} such that

$$(\vec{m}_j - \vec{m}_k) \cdot \vec{c}_{kj} = 1 \quad (8)$$

so that the threshold values, T_{kj} , are always between 0 and 1.

The classification vectors are generated in the training session by finding the vector direction that maximizes the distance between the two classes in the feature space. Specifically, if we let s_k represent the scalar values generated from the dot product of the feature vectors from class k and the classification

vector \bar{c}_{kj} , and likewise let s_j represent the dot product values between the class j feature vectors and \bar{c}_{kj} , then we want to define \bar{c}_{kj} such that we maximize the distance metric, d , defined as

$$d = \frac{\left(E[s_j] - E[s_k] \right)^2}{\left(\text{var}[s_k] + \text{var}[s_j] \right)} \quad (9)$$

where $E[\cdot]$ and $\text{var}[\cdot]$ represent the mean and variance, respectively of the values within the brackets. Note that we can re-write Eq. (9) as

$$d = \frac{\bar{c}_{kj}^T M \bar{c}_{kj}}{\bar{c}_{kj}^T (C_k + C_j) \bar{c}_{kj}} \quad (10)$$

where M is a matrix defined as

$$M = (\bar{m}_j - \bar{m}_k)(\bar{m}_j - \bar{m}_k)^T, \quad (11)$$

C_j is the covariance matrix for the feature vectors from class j and C_k is the covariance matrix for the feature vectors from class k . We can re-write Eq. (10) as an eigenvalue problem

$$(C_k + C_j)^{-1} M \bar{c}_{kj} = d \bar{c}_{kj} \quad (12)$$

where the -1 superscript indicates matrix inversion. The classification vector we want is the eigenvector that generates the maximal eigenvalue and thus maximizes the distance metric d . Since the matrix M has unit rank, Eq. (12) has only a single eigenvector solution which can be written as

$$\bar{c}_{kj} = (C_j + C_k)^{-1} (\bar{m}_j - \bar{m}_k) \quad (13)$$

which generates a distance metric value of

$$d = (\bar{m}_j - \bar{m}_k)^T (C_j + C_k)^{-1} (\bar{m}_j - \bar{m}_k). \quad (14)$$

Since the feature vector values are usually not independent, the matrix inverses in Eqs. (12) through (14) need to be performed using the standard pseudo-inversion process. Note that this classification approach is based on standard discrimination theory [5].

The training of the algorithms proceeds as follows. The user generates a database of example image subsets from the various classes, from which are generated example feature vectors. The mean feature vector and covariance matrix of the feature vectors are generated for each class from the database of example feature vectors. For each pair of classes, the classification vectors are generated using Eq. (13). Finally, the threshold values, T_{kj} , are generated by calculating the statistics $E[s_k]$, $\text{var}[s_k]$, $E[s_j]$, $\text{var}[s_j]$ as defined above, assuming the density functions of these variables are Gaussian, and finding the location where the two density functions intersect. This generates all of the parameters we need for the classification: the mean feature vectors for each class, the classification vectors for each pair of classes, and the thresholds for each pair of classes.

Since we are using a $M_1 \times M_2$ window to perform the classification of the image, for each placement of the window we assign the resulting class to every image sample in the window. If we move the window by one sample each time and re-classify, we actually classify each image sample $M_1 M_2$ times. In the algorithm we keep track of these individual classifications for each image sample, and when we have finally left the image sample, we assign it the class that occurred most often. This helps to significantly clean up the classification results near edges within the image.

The classification process as defined is not guaranteed to classify every image sample. It may be that a feature vector resulting from some placement of the window does not fall on the "right" side of all the hyperplanes for any single class. If this happens, the algorithm puts the sample into a special "No Class" category. Note that the final classification will only be "No Class" if this occurs most often for a given image sample over every placement of the window.

Because the classification vectors are eigenvectors, they can be very sensitive to the image values in the training set. Thus this approach will only work well if the training set is sufficiently large to be an adequate

representation of the statistical properties of each class; that is, there must be enough samples from each class to generate statistically accurate entries in the covariance matrices.

Finally, note that if $C_k = C_j$ for any pair of classes, then this approach is equivalent to a Normal distribution likelihood ratio test.

The power of this detection approach lies in choosing appropriate feature vector elements for the application. In the case of detecting ship wakes, we first threshold the image to generate a binary output image with 1's for dark regions and 0's for bright regions. We then clump all the 1's into connected blobs by labeling as a single blobs all pixels with a value of 1 that are within a user specified distance of each other (in all of these results we used a distance of 3 samples). We then characterize each blob by specifying its shape (how elliptical, how filled in it is, how smooth the edges are, etc.) and the image intensity values in the blob location (the mean image intensity within the blob, the average gradient of the change in image intensity at the edge of the blob, etc.). These blob characteristics become the feature vector elements, and we derived the classification vectors as described above. We can then play with the thresholds in the resulting hyperplanes to generate the "optimal linear" algorithm (discussed above), to minimize the total error (i.e. both missed detections and false alarms) within the training set, or to make sure that all the oil examples in the training set pass, regardless of other errors. We tried these three approaches, and found that the "optimal linear" approach worked the best. Figure 4 summarizes the ship wake detection algorithm.

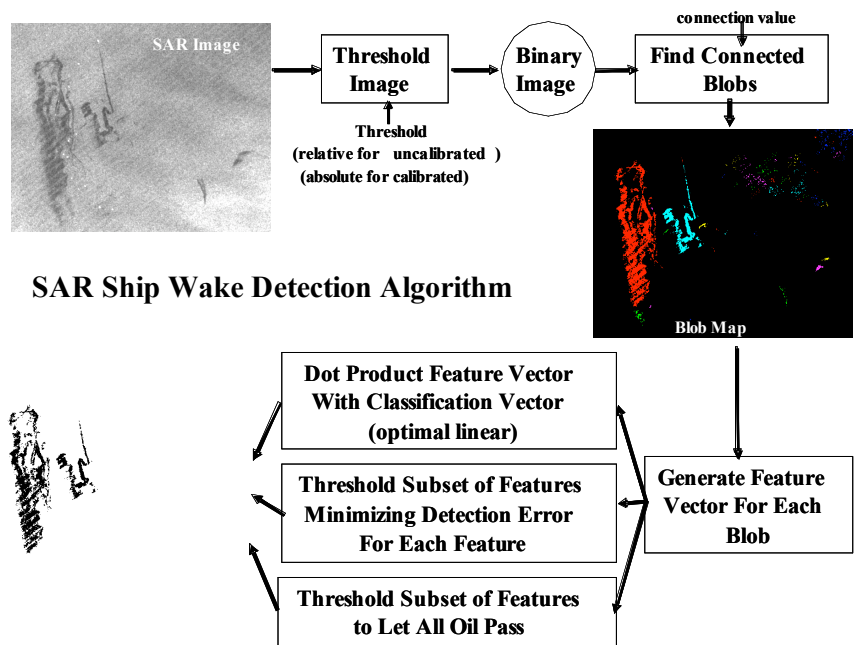


Figure 4: Flowchart for the SAR wake detection algorithm. It is based on a supervised classification algorithm where the classification vectors are generated from a training set of known wake signatures. The classification is based on a feature vector derived from the SAR image. In this case the features come from characterizing dark “blobs” in the SAR image via their shape and image intensities.

The wake detection algorithm has not been validated to date; it has only been run on a small number of examples and compared to manual interpretations. Figure 5 shows some of the results. Validation is ongoing.

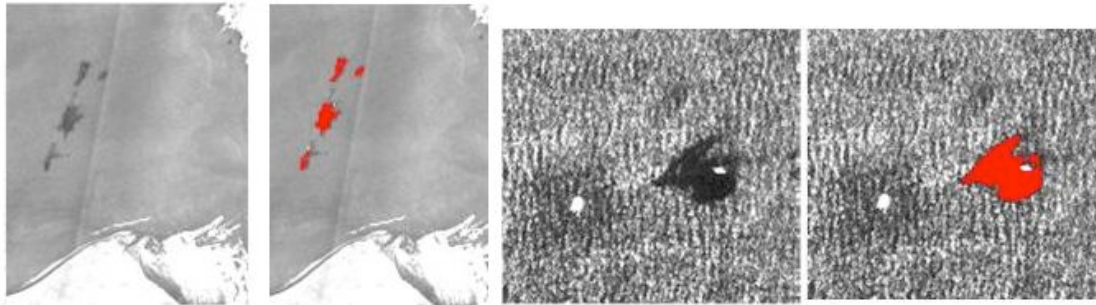


Figure 5: Example results from the wake detection algorithm. The original SAR image is on the left and the detected region is shown in red on the right. So far the algorithm has only been run on a small number of cases.

5. SUMMARY

We present two algorithms for use in automated ship detection in SAR imagery. One locates the hard target response of the ship and is a constant false alarm algorithm based on a K-Distribution model for the background clutter. Its performance is validated using two sets of SAR images with known ship locations and ship lengths, where it is shown that it can detect ~90% of all the known ships, even down to 10m in ship length. The second algorithm locates a ship wake and is based on a supervised classification algorithm that first finds the image “blobs” that represent the wake, and then classifies it based on the shape and image intensity characteristics of the blob. This algorithm has not been validated to date; some example results are shown.

6. REFERENCES

- [1] OLIVER, C.J., 1993: Optimum texture estimators for SAR clutter, *J. Phys. D: Appl. Phys.* 26, pp. 1924-1835.
- [2] WACKERMAN, C.C., FRIEDMAN, K.S., PICHEL, W.G., and CLEMENTE-COLÓN, P., 2001: Automatic detection of ships in RADARSAT-1 SAR imagery, *Canadian J. Remote Sens.* 27, pp. 568-577.
- [3] WACKERMAN, C., and MILLER, D., 1996: An automated algorithm for sea ice classification in the marginal ice zone using ERS-1 Synthetic Aperture Radar, *ERIM Technical Report 252000-25-T*, Ann Arbor, MI, USA.
- [4] WACKERMAN, C., 1997: Use of an interferometric SAR for terrain classification, *Proc. 26th AIPR Workshop*, SPIE vol. 3240, pp. 75-86.
- [5] LACHENBRUCH, P.A., 1975: *Discriminant Analysis*, Hafner Press, USA.