Automatic Target Recognition in Synthetic Aperture Radar Images

Oren Laskin (laskin@andrew.cmu.edu)
Elaine Ramundo (eramundo@andrew.cmu.edu)
Andrea Okerholm (amo@andrew.cmu.edu)

Digital Communications and Signal Processing Systems Design
Department of Electrical Engineering
Carnegie Mellon University
18-551 Prof. Casasent
Spring 2000

Introduction:

Automatic Target Recognition (ATR) of synthetic aperture radar (SAR) images is an area of ongoing research by all branches of the military and large research institutions. These images are being processed to locate interesting objects within them, such as enemy military vehicles – which can be classified by type. Another possible application of this technology is mapping; where landmarks could be identified and automatically placed on a map at the correct coordinates.

Systems such as SAR generate very large amounts of data often in the form of two-dimensional images. Due to the large scale of these images and the relatively small size of targets within them, finding the targets via ATR becomes a serious issue. Because of the research in the field of ATR, the basic methodologies are well studied and can be divided into three main groups; statistical based systems, model based systems, and neural network based systems. Since we rely on an outside source for our two-dimensional image database, we cannot work with a true model based system. We approached the problem of ATR from the statistical basis so that we could focus primarily on the cascades of filters needed to perform the image processing rather than generating and teaching a complex neural network. While such statistical systems are documented, implementation varies from system to system depending on hardware and software optimization. We attempted to reproduce the accuracy percentages of other experiments on the laboratory's Texas Instruments C67000 DSP boards.

The Database:

For this project we relied heavily upon synthetic aperture radar images provided by MSTAR. (Throughout this project we discovered that while these images were very useful, they were also somewhat limited). Although the MSTAR CDs that were ordered during the semester were better documented than the original image database we received from Professor Kumar, we still faced difficulties in understanding the precise nature of the library's contents. In general each MSTAR CD contains 2 to 3 categories of Russian armored vehicle that contain the vehicle from different angle positions overlooking the vehicle. Within each folder designated by angle, the same target is captured from multiple orientations and directions, in a series of passes by the radar onboard an aircraft. Therefore while we knew that each folder contained roughly 196 images we were uncertain where one pass began and another ended. Another issue we had was that these images would always have the vehicles roughly centered in the frame on a background of what we assumed to be a grassy or similarly flat, featureless, terrain. This type of surface provided only minimal clutter. There was nothing that could be mistaken for a target. The lack of clutter was a problem because it made us shift our focus from generating a full fledged ATR system, to focusing on classification of the vehicles. (Without "real" clutter in the images, the CFAR and Discrimination stages of the design were relatively unnecessary and rather trivial). We went ahead and implemented both stages under the assumption that eventually we would have enough clutter in our images to need these stages but were never able to find a good test image with both target and clutter present.

The CFAR function:

The concept and design of the CFAR filter for our ATR system came directly from ideas outlined in the Lincoln Laboratory Journals from 1996. We attempted to follow their design ideals in that we used the margin pixels to estimate image statistics of standard deviation of image clutter, and average intensity of background clutter. These two values are combined to generate a threshold value for searching the image for "target" pixels using the equation:

$$\frac{\left|I_{Pixel} - I_{avg}\right|}{I_{std}} > \text{Threshold}$$

Where I_{Pixel} is the intensity of the current pixel being tested, I_{avg} is the average intensity of clutter, and I_{std} is the standard difference of the clutter. We selected our threshold via trial and error. In the end we achieved satisfactory results by setting it equal to one sixth of the average intensity of a tank (a constant set over all images equal to roughly 43) and then manipulated per image by the generated average intensity and standard differentiation. The CFAR filter generates the threshold value once per image, then uses it to search through every pixel in the 128x128 image searching for target pixels, if the current pixel passes threshold test then it is checked against the location of previously found "target" pixels, whose locations are stored in an array of (x, y) pairs. If the current location is within 20 pixels of any previously found pixel then it is not added onto the array, if it is a new and distinct point the array is updated accordingly. The value of 20 was chosen because we chose the size of 40x40 for the minimum size to capture a tank from our database so "half an image" would be 20 pixels. In Matlab the function was be implemented by calls to *diff* and *sum* followed by a for-loop through image. For the EVM board we implemented our own versions of *sum* and *diff of array* in order to maintain simplicity of code. We initially wrote the function to return a 40x40 image based around one of its “target” pairs for debugging purposes, but then changed it to return an array of “coord” pairs for the discriminator function to examine. Since each of the for loops occurs separately the function resolves in linear time with 3,904,963.0 cycles when not optimized and 6,206,811 cycles with optimization flags turned on. The main issues we had with building this function was not in implementation but in the inflexibility of our database. Tests were run on images with clutter and with tanks, but there were no images with tanks in fields of clutter. The results of CFAR on a standard “tank” image and then on a standard “clutter” image are attached to the back of this report.

Discrimination:

The purpose of the discriminator is to reduce the number of false alarms from the CFAR stage. The CFAR output is a series of coordinate pairs designating possible targets. These targets could be tanks, buildings, trees, etc. In order to rule out candidates containing trees and other “natural-clutter” objects, we ran a series of statistical tests using our knowledge of an average tree and an average tank. The discriminator output is fed directly into the classifier, so it is important to eliminate as many images as possible without eliminating any actual tanks. We also desire to eliminate “man-made clutter”: man-made objects that are not targets. This discrimination is beyond the scope of our project, but deserves some mention here. Man-made clutter would include buildings and other types of vehicles.

Tests we researched and/or used in our final demo were fractal dimension, Sobel and Gaussian edge detection, weighted rank fill, and standard deviation. These tests each have shown a statistical difference in their results for natural-clutter objects and for target objects. To calculate threshold values for each test, we ran the tests on 10 target images and averaged their results. If an image passed one edge detection, the standard deviation, and one of two weighted-rank fills at different thresholds, it was declared to be a tank and returned. As in other sections of this project, we first implemented the discrimination algorithms in Matlab, then in C, then on the EVM.

Threshold Values:

Standard Deviation of top 50%	Sum of the Sobel of top 50%	Sum of the Gaussian of top 50%	Power in top 10%	Power in top 50%
13.0	80.0	70.0	35.0	75.0

Fractal Dimension:

Fractal Dimension is a measure of how close together the pixels in a binary image are. The pixels in a man made object are generally closer together than the pixels in a natural-clutter object, thus making fractal dimension an appropriate test to use in natural-clutter detection. We threshold the image, taking only the most powerful pixels, and then calculate their fractal dimension. A natural-clutter object will have a fractal dimension of less than 1; whereas a target object will have a value between 1 and 2.

One of the many methods of calculating fractal dimension is the box counting method. It counts the number of boxes needed to cover a one pixel wide border, increasing the size of the boxes from 2x2 pixels up. The fractal dimension is the negative of the slope of the line (boxes vs box size) on a log-log plot.

We found code in C implementing the algorithm from "A FAST ALGORITHM TO DETERMINE FRACTAL DIMENSION BY BOX COUNTING", by Liebovitch and Toth, Physics Letters A, 141, 386-390 (1989). at http://life.bio.sunysb.edu/morph/soft-out.html. This uses a form of the box counting method. We were able to integrate this into our C code, however were not able to get it working on the EVM. The EVM would infinite loop in the middle of a radix sort done prior to the box counting. Because of these problems, the fractal dimension function was left out of our final demo. A solution to the problem would have been to either move the sorting algorithm to the PC side, or to rewrite the sort completely so that it worked.

Other Tests:

Sobel edge detection used two kernels, one for each of the x and y directions. The x kernel was [1/8,0,-1/8,2/8,0,-2/8,1/8,0,-1/8]; and the y kernel was [1/8,2/8,1/8,0,0,0,-1/8,-2/8,-1/8]. The kernel used for the Gaussian edge detector was [1/32,4/32,1/32,4/32,12/32,4/32,1/32,4/32,1/32]. We summed the returned image to get a characteristic number, which could then be compared against the training threshold. //Our Sobel and Gaussian edge detectors did not make it successfully to the EVM due to an unexplainable floating-point problem.

Weighted rank fill returned an image with only the top n% brightest pixels.

The percentage of the power of the image contained in these pixels also gave us insight into the discrimination of the image.

Standard deviation of the image gives a measure of the contrast of the image. We know that in general, natural clutter has a low standard deviation and target images have a high standard deviation.

//We also researched a Canny edge detector but were not able to port it from Matlab to C. Code was found in the IEEE IUE image processing library but it was written in C++ and we never successfully ported it to C or the EVM.

Result:

In the end, the discriminator successfully passed our tank images with a X% accuracy. It completed the task in 7,313,967 cycles not optimized and in 3,881,510 cycles with optimization flags on. We did not test for rejection of natural-clutter images.

Classifier:

Certain assumptions had to be made for the classifier that was designed to function correctly. Due to the MSTAR database that we received, roughly 196 images of three types of vehicles existed. We chose to train using 1/4 or 1/8 of the 196 images and the rest was used for testing purposes. After converting the MSTAR info into raw 128x128 at 8bpp we were ready to attempt to classify the images. Our first attempt was all in software on the Unix environment.

Due to work I was doing with a partner in Image and Video Processing (18-798), I chose to attempt to classify by using the Hough transform after doing thresholding to the original image. The thresholding was done by blurring the image slightly and removing all parts that were below the mean+1.5*standard deviation. The result of this was reasonably good for most images but ran into difficulties when shadows fell upon the objects so you could run into an object that looked like it had been bisected.

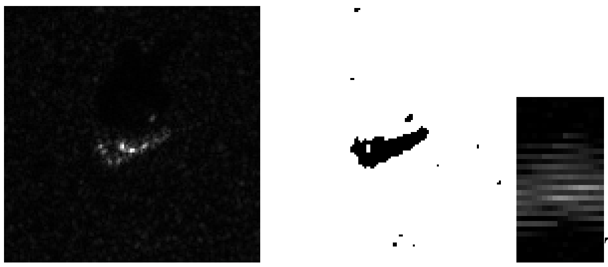
The Hough transform is a method of converting straight lines in an image into slope-intercept form. Using this transform gains you the ability to detect where lines meet and comparing training images with the test images gives a good classification result. The rho and theta used for the Hough transform result were both 32. Therefore the resulting picture was a 17x32 image at 8bpp since negative rho values were ignored.

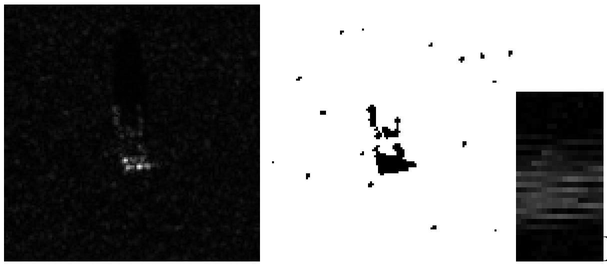
After doing Hough transforms on all the testing and training images a very simple nearest neighbor algorithm based on using the mean squared error was used. This has the advantage of being computationally very simple to use and gave surprisingly good results so we continued to use it.

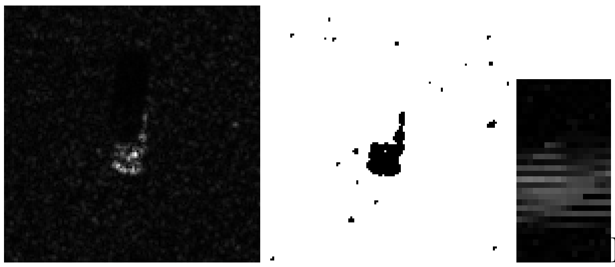
Certain things were tried and dropped while creating the classifier for this project. The first was a Sobel filter that was used to extract the outer lines of the image before they were sent to the Hough transform. This was removed when no noticeable gains were coming from it. Also, the computation for a Sobel filter is very high when done on a large sized image. The second major change was that the Hough transform needed to be rewritten in the last week. At first, we believed C code downloaded from the Internet had been working but when the dataset was changed from Professor Kumar's images to MSTAR images, the classifier broke. Therefore, Matlab Hough transform code that was proven to work became the basis of a new C function that we wrote and implemented for classification.

As stated earlier the final results were good enough that we moved the code over to the DSP early. Our original images were based on the one Professor Kumar handed out for his Pattern Recognition Theory (18-794) class and the original classifier was based off that. Using 1/4 as training images and having Sobel edge detection and the original Hough Transform, we received an 89% correct recognition rate. However, Professor Casasent wanted our classifier to be based directly from using the MSTAR database images. This was accomplished by getting the database CDs, and necessary Unix tools, from the MSTAR home page. A simple shell script was created and the MSTAR database was converted to a series of numbered images that contained magnitude information of the 128x128 images and ranged in values from 0-255. After the modification of the code I mentioned earlier, removing Sobel and replacing Hough Transform, we had a 96% correct identification rate. We decided to test for robustness by making only 1/8 of our images be for training and our accuracy rate only fell to 92%. This 92% correct classifier was what was shown in the final demo.

Many improvements can still be made to this classifier. First of all, the thresholding on the image needs to be improved so that under any condition we can accurately discriminate the entire vehicle and not have major shadows remove portions of vehicle and at the same time prevent clutter noise from reducing our accuracy. I believe this can be done with some work on constant values and removing parts of image that still remain that are too small to be our target. However, now you must choose whether to trade accuracy for speed. The Hough transform is a major part of this classifier. This can be unrolled to some degree and optimized for greater parallelism and thus more speed without removing accuracy. The rho values and theta values used should be adjusted to see what is optimal. The values of 32 were chosen arbitrarily, and we stayed with it due to the good results that we had obtained. What the optimal value is needs be determined and if you go too high it will make the next step, the MSE, a bit slower. Finally the classifier is a simple nearest neighbor method, taking the MSE of the training set compared to test image that is minimal. This is optimal for simplicity of code and speed but for greater accuracy you would want to take the closest n neighbors and see which of the classes is most represented in that set. This will remove outlying cases from causing errors. These are simple improvements that could be made to follow on to this project to make it more robust and faster. The Hough transform ran in 77,880,430 cycles with optimization flags turned on.


T-72


BMP-2


BTR-70

Conclusions:

Now that our project is over we have mixed feelings about the actual results. While the project was completed and demoed successfully there were several items that we had really hoped to accomplish. We had hoped to implement a full system where the CFAR, discriminator, and classifier had all worked together rather than demoing them separately. Given more time we believe we could have better information about managing on board memory, parallelization, and other optimizations. Also, for better testing we need images that not include just clutter or just images, but both so we have a full test for the system.

A wish list for future classes includes… better description of EVM bugs. An EVM emulator on Unix so a full demo could be given without hardware bugs causing problems. We think the checkpoints should be arranged in this manner… proposal followed by C code on Unix and this followed by the code working with this mystical EVM emulator before the final demo is shown. Also, we believe this class lacks a focus and should learn from other capstone design classes that have a stronger focus such as 18-545 and 18-778. The lectures given them become appropriate for everyone rather than a small minority. We believe that more could have been learned from this class. While the hands-on project experience was valuable we felt the majority of our time was not directed at the DSP problems we chose to investigate, but at what should've been simple EVM board and Code Composer Studio interactions.

Bibliography:

http://life.bio.sunysb.edu/morph/soft-out.html

https://www.mbvlab.wpafb.af.mil/public/sdms/tools/index.htm

"A FAST ALGORITHM TO DETERMINE FRACTAL DIMENSION BY BOX COUNTING", by Liebovitch and Toth, Physics Letters A, 141, 386-390 (1989).

Dudgeon, Lacoss, "An Overview of Automatic Target Recognition". Lincoln Laboratory Journal, vol 6, number 1, 1993. p 3-9.

Novak et al, "Performance of a High Resolution Polarimetric SAR Automatic Target Recognition System". Lincoln Laboratory Journal, vol 6, number 1, 1993. p 11-23.
Kreithen et al, "Discriminating Targets from Clutter". Lincoln Laboratory Journal, vol 6, number 1, 1993. p 25-51.

Verbout, et al, "Improving a Template-Based Classifier in a SAR Automatic Target Recognition System by Using a 3-D Target Information" Lincoln Laboratory Journal, vol 6, number 1, 1993. p 53-71.