Proceedings of the Institute of Acoustics

TARGET INJECTION FOR SAS CAD/CAC TRAINING AND TESTING

A Putney Applied Signal Technology, Inc., Torrance, California, USA
J K Harbaugh Applied Signal Technology, Inc., Torrance, California, USA
M A Nelson Applied Signal Technology, Inc., Anaheim, California, USA

1 INTRODUCTION

Synthetic aperture sonar (SAS) processing can provide the spatial resolution required to detect and classify mines at long ranges. Several SAS systems have demonstrated along-track resolution of 2.5 cm, both at short ranges from unmanned undersea vehicles (e.g., the Hydroid REMUS and Bluefin vehicles) and at ranges of several hundred meters from the towed Slow Speed SAS system. Outside the US, the Norwegian Defense Research Establishment is testing a SAS system on the Hugin unmanned undersea vehicle. Compared to conventional sidescan sonars, SAS has the potential to increase area coverage rate (ACR) at classification resolutions by an order of magnitude. With this increase in both ACR and resolution comes a dramatic increase in image data volume and rates. The expected data rates are likely to overwhelm human operators, thus such systems will require computer aided detection and classification (CAD/CAC) to extract targets from mission data.

At present, automated mine detection and classification is used in a limited capacity with side-scan mine hunting sonar systems such as the AQS-20 and the AQS-14, and is also being tested on unmanned platforms such as the REMUS 100[1,2]. The foremost work in the field is in the fusion of multiple CAD/CAC algorithms to decrease the false alarm rate[3,4,5]. CAD/CAC systems typically select regions of interest (ROI) and evaluate the probability of the presence of a target through template matching, statistical matches of features, or neural network matching of features. All require extensive training using data collected by the system under normal operating conditions. Currently fielded CAD/CAC systems are designed to work with conventional sidescan sonar data and need to be extended to utilize the SAS imagery that will be collected by future mine countermeasure assets. Preliminary successful work in SAS CAD/CAC training has thus far been conducted by Dobeck[6].

The key to this "data-based" CAD/CAC approach (vice the model-based approach used in synthetic aperture radar, e.g., MSTAR[7]) is training. Extensive image training sets are utilized for both the design of templates and the development of classification engines that exhibit high probabilities of classification and low false alarm rates. To properly train the CAD/CAC system on a given mine, representative imagery must be collected for various target geometries and all requisite system, operational and environmental characteristics. The time required to obtain these various data may delay the deployment of a specific sonar's CAD/CAC engine for many years.

A more efficient method for creating training images is required to support the growing use of CAD/CAC. Applied Signal Technology, Inc. (AST; formerly Dynamics Technology, Inc.) is creating a sonar image simulator that will enable concurrent sonar system development and CAD/CAC training. The simulator will also support CAD/CAC upgrades to operational systems, such as adding new targets to the CAD/CAC database. Simulated target images alone are not adequate for training due to the absence of realistic seabed clutter that can influence CAD/CAC system performance. Fortunately, existing simulation tools are capable of producing high-fidelity synthetic target data, and the collection of extensive libraries of varied seabed data is a relatively inexpensive undertaking. Our basic approach is to inject simulated targets, or target-like clutter, into images of real bottom clutter to create intricate simulated data sets suitable for CAD/CAC training.

2 APPROACH

2.1 Target Injection

Our approach, shown schematically in Figure 1, is to extract a small section of seabed data from a large swath of imagery, inject an object into that small patch, and then replace it in the larger swath, thus the software's name, "Injector." This large data set with an injected target can then be run through a CAD/CAC system as normal. The object is injected into the scene with ray-tracing target simulation software that includes interactions between the target model and seabed objects in the scene. AST has designed this software to use a section of clutter imagery as a bottom map and allow a virtual target to interact with the bottom through multiple scattering paths. This technique allows the target image to take on characteristics of both target surface roughness and the local seabed reflectivity features.

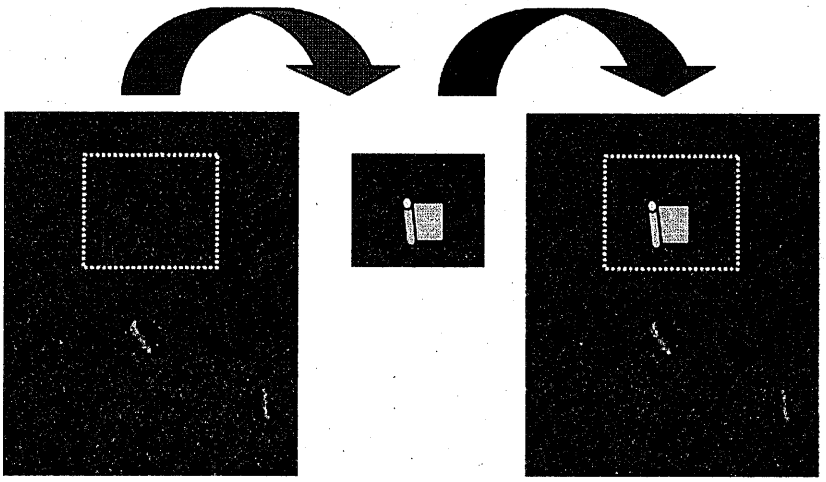

Figure 1. Schematic of injection methodology. Left panel shows the original swath of sonar imagery with a small region selected for use in the simulation. Middle panel shows the extracted region with an injected object. Right panel shows the swath of data with the injected object.

We are using SAS data from the Small Synthetic Aperture Minehunter (SSAM) system for our initial injection validation. The available data contain a variety of bottom types and objects that are appropriate for the testing of the simulator. There are sufficient targets present that we can inject simulated targets nearby for validation.

The work presented here is in the first validation phase. This means that the injected object must pass the "by eye" test – it must appear reasonably similar to the image of a real target to an experienced analyst. Future work will include simple quantitative tests such as measurement of length, width, and intensity statistics of the injected object and its shadow, which will be compared to measurements of actual images of the objects they are intended to mimic. The ultimate validation is for a CAD/CAC to respond to the injected object image as it would the image of a similar object collected during sonar operations.

2.2 Image Formation

Injector forms images directly via ray-tracing, meaning either a SAS or real-aperture sonar (RAS) image is formed directly, rather than simulating the raw element level data which would then be processed into a RAS or SAS image. This approach was chosen for two reasons. First, the input bottom clutter imagery may only be available as SAS or RAS data (although the method does not preclude using element level data). Second, by operating in a data domain characterized by very narrow beamwidths, the computational load of the simulation is dramatically reduced. This method

is valid in the optical limit, where the wavelength is assumed to be short compared to the target size.

In this approach, the simulator inputs are more closely related to the final image characteristics than they are to the specifications of the sonar used for data collection. Thus neither the number of receiver sonar hydrophone elements nor their beamwidth is needed, while the spatial grid size of the SAS or RAS image, the sonar signal center frequency, and bandwidth are required. This is a different way to think about sonar simulations, particularly for SAS, but it is appropriate when simulating image rather than sensor data.

The ray-tracing image formation methodology allows for the accurate projection of shadows, including target self-shadowing. We have tested the Injector software with several geometric shapes (planes, spheres, cylinders) and are developing additional objects (e.g., cones, truncated cones, toroids). In operation, the simulator reads a bottom image, typically from collected data, and uses it as an intensity map to allow the mine-like object's acoustic response to be influenced by the bottom.

The Injector simulator we have created comprises two software components. The first traces the rays and calculates raw acoustic impulses as would be seen by the sonar. The second converts the impulses into an image. The current configuration produces SAS images, but with modifications RAS images could be formed.

3 MODELING

3.1 Overview of Model

The sonar is modeled as a co-located acoustic source and receiver. The sonar transducer is square shaped, can point in an arbitrary orientation, and has a specified center frequency and bandwidth. The source/receiver pair moves along a linear, horizontal track. At equally spaced intervals along this path, the acoustic return of an infinitely narrow vertical receiver beam is computed using a fan of rays varying in elevation angle (see Figure 2, left panel). The narrow beam approximates the narrow beamforming produced by SAS processing. The rays are distributed uniformly in elevation angle. Sufficient elevation angles are generated to adequately sample the scene at all ranges of interest.

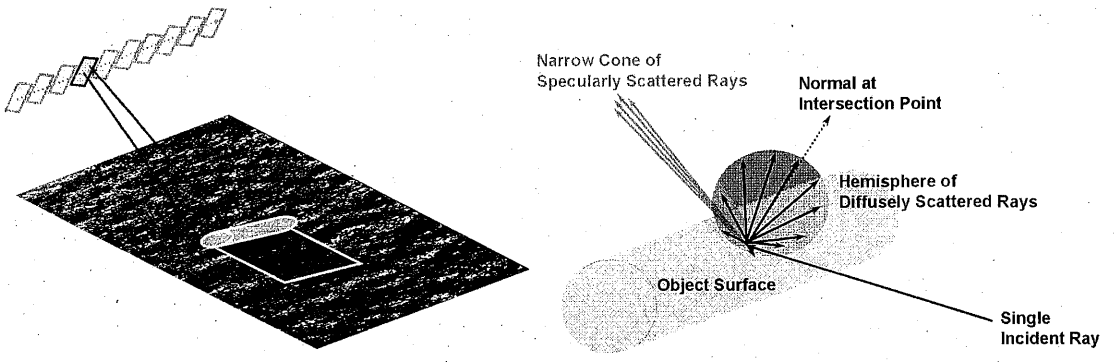

Figure 2. Schematics of Injector model. Left panel demonstrates the position by position approach in developing the image. Right panel shows the rays spawned after an interaction of the ray with an object.

Each ray generated by the receiver is propagated forward, ultimately to determine if it returns to the source. Because of time-reversal symmetry, tracing rays from the receiver to source is equivalent to tracing from source to receiver; however, fewer rays need to be calculated when tracing from the receiver to the source, a feature exploited by Injector (see Section 3.2). A ray that intersects an

object then spawns additional rays, depending on the surface properties of that object. These additional rays represent both specular reflections and diffuse scatter from the intersection point on the object. The number of reflections, or recursions, for a given run is set at the start. Rays are spawned if the number of recursions is below the input recursion limit.

For each set of rays that form a complete path from the receiver to source, the path lengths and complex strengths of the return impulses are computed. The convolution of the sum of the impulses with the sonar's pulse shape results in the return-scatter time series.

The objects in the scene are modeled as geometric shapes (see Section 3.3). All possible intersection points between each ray and each object are computed, and geometrically excluded intersections are discarded. For example, a ray can intersect a sphere at one, two, or no points. However, only the intersection point closest to the origin of the ray is actually used, thus creating a model of self-shadowing of the far side of the object. Scattered rays then spawn from the intersection (see Figure 2, right panel). Specular scatter is modeled by creating a few rays in a narrow cone in the specular direction. Diffuse scatter is modeled by many rays filling the hemisphere tangent to the surface at the intersection. The scattered rays are weighted with a generalized Lambertian scattering coefficient.

3.2 Sonar Model

The sonar model, as currently implemented, represents a SAS sensor. An idealized SAS sensor has an infinitely narrow, horizontal beam. The receiver portion of the sonar is modeled as a point source with a fan of rays extended in a vertical plane. The source portion of the sonar is modeled as a finite rectangle with a finite beam pattern, in order to allow rays to complete a reflection circuit (an infinitely small source would preclude any rays intersecting it). The rays propagate forward and those that are reflected back to the sonar are used to compute the backscatter signal. Each completed circuit of rays (from the sensor, into the scene, and back to the sensor) has a corresponding impulse.

Each impulse has a complex amplitude and time delay. The time delay is simply the time of flight of sound along the entire path:

$$t_{\text{ray}} = \sum_k \ell_k / c_k \,,$$

where the summation is over all rays in the completed path, ℓ_k is the length of the ray and c_k is the speed of sound through the medium in which the ray is propagating (assumed to be uniform). The complex amplitude is the product of reflection coefficients of each intersected surface and a phase factor depending on path length:

$$A_{\text{ray}} = A_{\text{sonar}} \prod_k \rho_k w_k \exp\left(i f_{\text{sonar}} \ell_k / c_k\right) \,,$$

where the product is over all rays in the completed path, ρ_k is the reflection coefficient of the surface at the origin of the ray (unity for rays originating at the receiver), w_k is the weighting of the ray (according to the fraction of the solid angle sampled by the ray). The amplitude intensity and the center frequency of the sonar are A_{sonar} and f_{sonar}, respectively.

The impulses are then convolved with a Gaussian pulse in order to generate a range-compressed time series:

$$s(t) = \sum_k A_k G(t - t_k) \,,$$

where the summation is over all impulses, and A_k and t_k are the complex amplitude and time delay of each impulse. The Gaussian pulse envelope, $G(t)$ has a width appropriate for the bandwidth of the sonar.

The receiver and source are then moved to the next position in the track, and the procedure repeats itself. The track is assumed to be both level and linear. Ultimately, a two-dimensional SAS image is produced.

3.3 Analytic Geometry Models

Arbitrary shapes can be modeled in the software, provided two conditions are met. First, the intersection of an arbitrary ray and the object's surface must be calculable. Second, the surface normal at the intersection point must also be calculable (see Figure 3). Injector has the equations for simple geometric shapes (e.g., cylinder, sphere, plane) and the user can create more complex shapes through the superposition of multiple simple shapes. This approach easily allows for object self-shadowing.

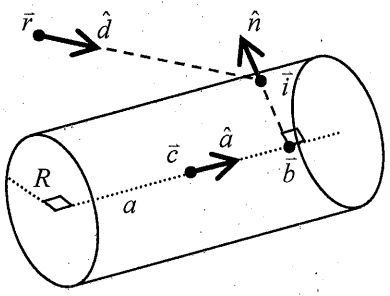

Figure 3. Schematic description of cylinder. Its center is $\vec{c}$, radius R, length $2a$, and axis in direction $\hat{a}$. The intersection of ray $\vec{r}$ with the cylinder is $\vec{i}$, with a normal $\hat{n}$.

Every item in a scene is considered an object and modeled geometrically, although different attributes are given to the object. A sound velocity profile can be entered into Injector by creating a series of refractive planes. The clutter imagery used for the bottom map is considered a plane with an intensity map. At present the forward and backward scattering coefficients associated with this intensity map are directly proportional to the intensity. A planned upgrade to the code will use the bottom intensities for the back scatter and an appropriate model to calculate the forward scattering.

3.4 Convergence, Run Time, and Image Quality

Injector is under development, thus the version presented herein has a few limitations. The quality of the final image is largely based on the number of rays that make the complete circuit from receiver to transmitter. In order to fully populate a scene, millions of rays must be used. Every interaction must spawn thousands of additional rays (both specular and diffuse). Run time is directly proportional to the number of rays. Initial run times were prohibitive when multiple recursions were used. Simply reducing the number of rays used and spawned creates imagery that does not pass the "by eye" test criterion. In order to balance the image quality and run time, we are using a Monte Carlo method to statistically sample the initial and spawned rays.

With the Monte Carlo method, a single-interaction run of Injector is about 20 CPU days which we split amongst 10 CPUs for a reasonable total of 2 calendar days. A two-interaction run is approximately 10 days with 10 CPUs. A three-interaction run is estimated to take over a CPU year, which is impractical given the current implementation and program objectives. We are investigating multi-CPU methodologies, such as the peer to peer architecture used in the Berkeley Open Infrastructure for Network Computing (BOINC[8]), and plan to put that into operation.

Image quality is also affected by combining data, which contains noise, and a simulated object, which is modeled as a noiseless reflector. Thus we can end up with a realistic bottom and "perfect" object. This is not optimal since the injected object should blend naturally with the background image. The combination of the two also limits the ways in which we can make the object more realistic since adding noise to selected scene features is not feasible with the current simulation

methodology. One method we use to make the scene more realistic is to use a coherent sinc convolution in the cross-range direction on the complex-valued results to simulate realistic SAS point response functions. We are currently investigating other methods including introducing more multipath through adding an air-water surface, which is currently lacking in the figures shown in Section 4.

4 RESULTS

Over the course of development, we have run multiple test simulations to verify the basic functionality of the Injector simulator. Thus far, Injector has passed all tests. Here we present four key demonstrations of the Injector status:

1. An object on a flat, Lambertian bottom with only one interaction per ray (Figure 4);
2. A comparison of a sonar image of a seabed to an image of the same seabed reproduced by Injector (Figure 5);
3. An object injected into a sonar image snippet with only one interaction per ray (Figure 6); and
4. The same scene except allowing up to two interactions per ray (Figure 7).

We have simulated all scenes using an isovelocity sound velocity profile (SVP) and an infinitely deep ocean (no air-water surface). The sonar frequency was set to 100 kHz with 30 kHz bandwidth and the sonar located 10 m off the bottom and approximately 50 m range from the object.

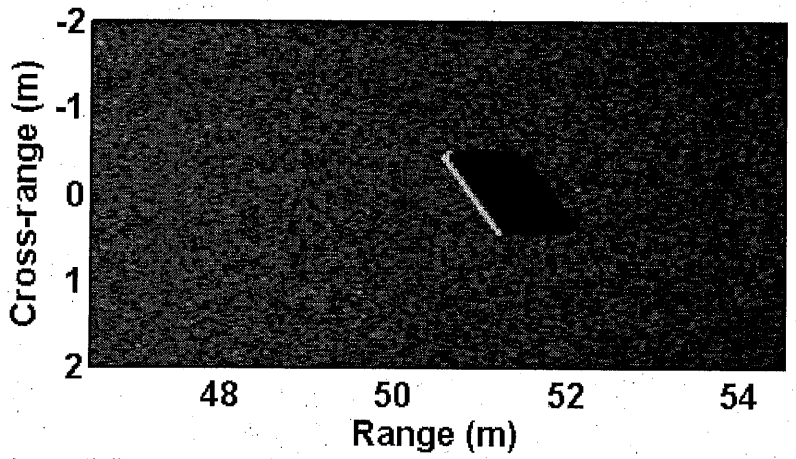

Figure 4. Cylinder partially buried in a plane. Cylinder is 1.2 m long with 0.1 m diameter and buried 0.067 m into the plane (uniform, Lambertian surface).

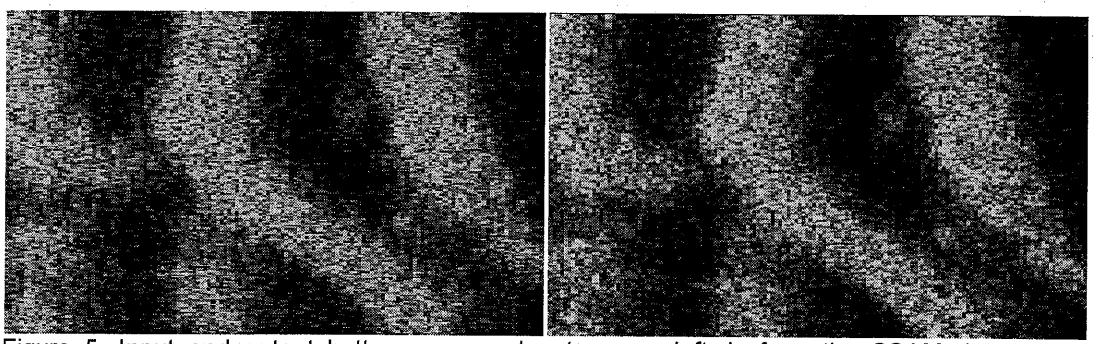

Figure 5. Input and output bottom scenes. Input scene, left, is from the SSAM data taken in November 2005 during trials off NURC. The output scene, right, is the result of Injector.

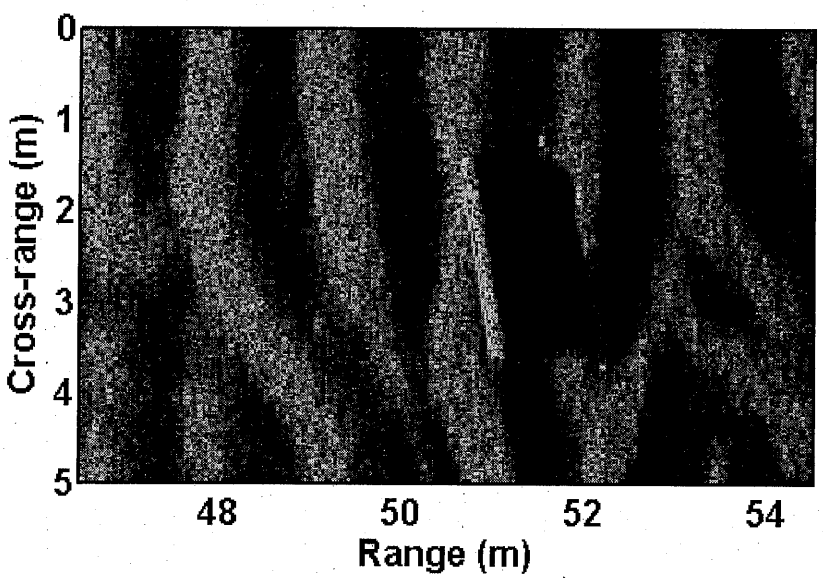

Figure 6. Cylinder Injected into an SSAM data snippet. Cylinder is 2.3 m long, 0.3 m diameter, spherical endcaps, and buried 0.1 m into the bottom. Only one interaction per ray was permitted. Data snippet was obtained from November 2005 sea trials off NURC.

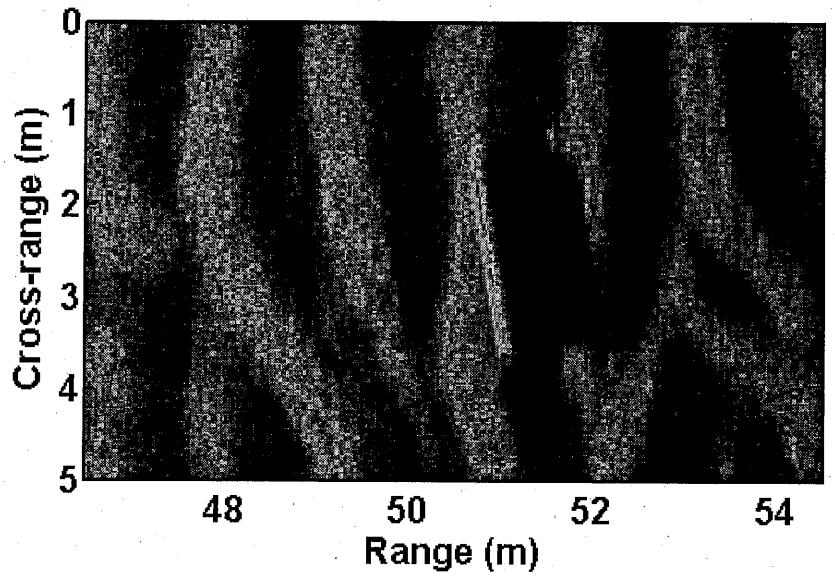

Figure 7. Cylinder Injected into SSAM data snippet for two interactions per ray. The same scene as for Figure 6, except that one additional interaction was permitted. Overall appearance is similar to the single-interaction case, indicating that the second recursion level introduces subtle refinements to the image in this particular simulation.

5 CONCLUSIONS

We have developed an acoustic ray tracing simulator, called Injector, which injects objects into collected sonar imagery. The results presented indicate that this is a promising technique for creating CAD/CAC training data sets. Simulations suitable for training will be available after the system has been appropriately validated.

Injector requires significant computing resources; however, the simulation methods naturally scale to multiple compute nodes, and run times decrease linearly with additional nodes. It is expected that a computer cluster of several dozen machines will yield satisfactory simulation throughput. Incorporation of automated job migration into Injector is in progress.

The Injector framework was devised to be easily extendable and support improvements incrementally. By allowing multiple interactions of an object with itself or other surfaces (bottom, air-water interface, refractive surfaces for changing sound speeds), we can test their effects easily. The choice of creating objects from simple geometric formulae allows us to create complex objects via superposition and therefore does not limit our choices to a few well known shapes. At present the code is validated for an analyst "by eye" demonstration with more quantitative tests planned for later this year.

6 ACKNOWLEDGEMENTS

This work has been supported by the Office of Naval Research, contract number N00014-05-C-0073.

7 REFERENCES

1. von Alt, C., 2003, "REMUS 100 Transportable Mine Countermeasures Package," Oceans 2003 Conference proceedings, p 1925-1930.H.J.M. Steeneken and T. Houtgast, The temporal envelope spectrum and its significance in room acoustics, Proc. 11th ICA, Vol. 7, 85-88. Paris (1983).
2. Dobeck, G.J., 2004, "Algorithm Fusion: An Overview - Combining multiple detection and classification algorithms," Proceedings of the 6th Monterey International Symposium on Technology and the Mine Problem, Monterey, 10-13 May 2004.
3. Dobeck, G.J., 2001, "Algorithm Fusion for Automated Sea Mine Detection and Classification," Oceans 2001 Conference Proceedings (Escondido, CA: Holland Enterprises), p. 130-134.
4. Aridgides, T., Fernández, M., & Dobeck, G.J., 2001, "Fusion of Adaptive Algorithms for the Classification of Sea Mines Using High Resolution Side Scan Sonar in Very Shallow Water," Oceans 2001 Conference Proceedings (Escondido, CA: Holland Enterprises), p. 135-142.
5. Ciany, C.M. & Huang, J., 2000, "Computer Aided Detection/Computer Aided Classification and Data Fusion Algorithms for Automated Detection and Classification of Underwater Mines," Oceans 2000 Conference Proceedings (Escondido, CA: Holland Enterprises), p. 277-284.
6. Dobeck, G.J., 2003, private communication.
7. Wissinger, J., Ristroph, R., Diemunsch, J., Severson, W., & Freudenthal, E., 1999, "MSTAR's Extensible Search Engine and Model-based Inferencing Toolkit," SPIE Conference on Algorithms for Synthetic Aperture Radar Imagery VI, Orlando, Florida, April 1999, SPIE vol 3721, p 554-570.
8. Anderson, D.P. & Fedak, G. 2006, "The Computational and Storage Potential of Volunteer Computing," IEEE/ACM International Symposium on Cluster Computing and the Grid, Singapore, May 16-19, 2006.