

SIMULATION OF A DISTRIBUTED TARGET RECOGNITION SYSTEM WITH VARIABLE OPERATING CONDITIONS

Saptarshi Adhikari
James Fang
Guenevere Lindgren
Shadi Wadie
Michael D. DeVore

Department of Systems and Information Engineering
University of Virginia
mdevore@virginia.edu

ABSTRACT

This paper details a simulation tool for assessing the effect of variable operating conditions on an automatic target recognition system. The simulation tool helped to assess the impact of variable operating conditions and hardware capability on the performance of distributed target recognition systems. Its user interface allowed for real-time monitoring of the best object classification, its associated likelihood value, and the time required to make the classification. Testing revealed that the available network bandwidth and the number of processing nodes used to compute most greatly influenced system performance, followed closely by the level of resolution of the target model images.

1 INTRODUCTION

Automatic target recognition systems are often conceived as an aid to soldiers to help them classify observations as friend or foe. Object recognition has been a topic of study for many years. Complex algorithms and detailed radar imagery are two of the most critical aspects of any automatic target recognition system. Thus, an automatic target recognition system's ability to accurately classify images, both quickly and consistently, have improved. While various automatic target recognition systems have been constructed to date, the intensive computational needs put on such systems have impaired progress (Hummel 1999). Unfortunately, because of their computational burden and slow-speed, the most sophisticated and best performing systems employing these methods are relegated to research-only activities and have not been adapted to field use.

These enormous demands can be satisfied by implementing a distributed system. The automatic target recognition system developed by our team uses recognition algorithms based on a conditionally complex Gaussian model (DeVore 2001). Further, this system relies on parallel

processing across a distributed network comprised of multiple computational servers that are part of a Linux cluster. Modules like a target model database, a Management Module and a Process Monitor reside on this cluster. See Figure 1 for a block diagram of the system.

Using MPI, a message passing interface, for reliable communication with each other, the nodes within this cluster work collaboratively to classify each image within a continuous stream of images. The likelihood servers compute the likelihood of a sequence of hypotheses generated by the Management Module. These hypotheses are responsible for testing the likelihood that one or more user-specified target models (from the Target Model Database) matches a specified image to be examined. With each target model retaining levels ranging from 1-9, a larger value here implies a higher resolution level for that model. Specifically, the level of any given target model, n , specifies that there are 2^{n-1} angles at which pictures of the sought object was taken. From here, the Management Module reports the best of the results, along with resource utilization information to the user through the Process Monitor.

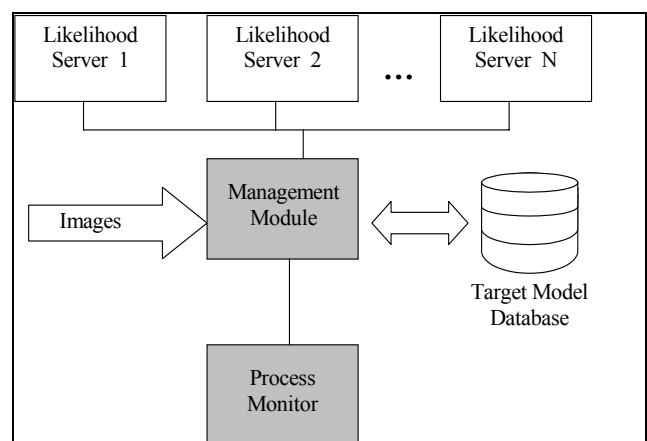


Figure 1. Design of a distributed Automatic Target Recognition System

Functionality of the recognition system simulation includes being able to specify three variables required by the Management Module: 1) the number of target models to be sent to the Likelihood Servers for comparison, 2) the level of resolution desired for the specified target models, and 3) the number of likelihood servers available for computation.

When an image is constructed and received by the system, the variances between the dark and light areas of the image are closely examined (DeVore & O'Sullivan 2001). The recognition algorithms perform a preliminary search that focuses on the variation over the orientation of the image, and its broad class of objects. Eventually, all comparisons are tried and the confidence in the recognition can be high.

One of the most sophisticated target recognition systems was developed through a project called Moving and Stationary Target Acquisition and Recognition (MSTAR) conducted by Wright Laboratories under DARPA funding. The goal of the MSTAR project was to advance the state of the art by developing a research system based on formal models of the objects to be recognized. Figure 2, below, displays sample images produced from of the MSTAR project.

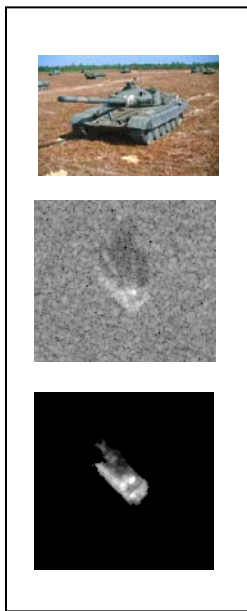


Figure 2.

Top: A photograph of a tank

Middle: A synthetic aperture radar image of the tank

Bottom: Pixel-by-pixel variance estimated from the middle image and several other images

In the distributed system outlined in Figure 1, the Management Module is largely responsible for the transfer of information throughout the system. More importantly, this module is capable of simulating changing operational conditions while these recognition processes are carried out. Specifically, the Management Module's configuration can be manipulated by the user to simulate a different number of likelihood servers available for computation, the number of models used for comparison, the resolution level of the models, and the available network bandwidth.

Future enhancements to the system will allow us to specify alternate management algorithms and to simulate slower processing units used for target identification.

After completing a single recognition calculation, the Management Module provides the user with: 1) the target model that most closely matches the radar image input, 2) the estimated angle orientation of the target model which most closely resembles the image, 3) a log-likelihood value of how well the target model matches the image, and 4) a rejection entropy value. Additionally, a timing mechanism tracks the time taken to transfer images, models, and classification results within the system. While the likelihood value assumes there is a match to the model in the image and reports the best model approximation, the rejection entropy value examines if the model found really is a good fit.

Unlike the Management Module that reports the results and findings of the actual target recognition process, the Process Monitor provides a single screen output of pertinent statistics regarding system resources consumed. Specifically, the Process Monitor allows users to monitor recognition tasks and the following statistics of each recognition process performed: 1) CPU memory usage, and 2) time required to complete the classification computations.

The literature on target recognition tends to focus on one of three technical areas of target recognition systems. The majority of the papers describe accuracy results obtained from taking sensor data and applying various algorithms. Others discuss the difficulties experienced with target recognition regarding accuracy as a function of operating conditions. A third group of papers explains how the accuracy of the system is dependent upon the complexity of the models used by the system.

Understanding that performance and reliability are critical aspects to the operation of any military, the goal of our project was to determine if an automatic target recognition system was feasible and efficient (Gelenbe 2000) without compromising performance integrity. To do so, our simulation tool collects statistical information specific to accuracy and performance measures for a given instance of target recognition. Tests were then subsequently conducted to investigate how varying the level of model resolution may influence the previously mentioned accuracy and performance statistics.

The remainder of this paper discusses the simulation process, testing of the system, and the results obtained. Ultimately, the result of the study determined the impact of operating conditions on the performance of an automatic target recognition system.

2 SIMULATION AND DATA COLLECTION

The simulation process begins at the Management Module where the user is asked to input the name of a file specifying the number of likelihood servers to be used in

the target recognition process. By modifying the number of likelihood servers on the network used to run the simulation, users can see how the simulation's performance is affected.

Specifically, the user can create a text file that selects any or all of the likelihood servers that reside on the network, and thus run the simulation on a selective group of servers by typing the text file's name at the given prompt. The simulation's versatility in running on a variable number of servers makes it a great tool to evaluate the sensitivity of an automatic target recognition system's performance to the availability of fewer or more likelihood servers on the network.

Upon providing this input, the user is then prompted to input the image and target model(s) they wish to be processed. Providing these as runtime parameters in the form of filenames to be read by the system, it is here that the user must specify the resolution level of the target model(s) they wish to have compared. It is important to note that the current Management Module applies the same resolution level to all target models being used for a single simulation run.

Next, the Management Module determines how many computations the system needs to process for comparing the user specified image with each of the target models in the user specified target model database folder. After determining the number of likelihood calculations to be carried out for that particular simulation run, the Management Module then allocates these computations across the number of likelihood servers the user specified at runtime.

As each likelihood server receives the user-specified image and target model(s), two things occur: 1) the Management Module goes into its "receive and send" state, where it awaits 2) the first likelihood value to be returned from one of the likelihood servers. Dealing with the ladder of these, the first likelihood server computes a likelihood of the target model being in the image and returns this value to the Management Module. This likelihood value is computed by superimposing the target model on the image in every possible position and then comparing them pixel by pixel. Depending on the model and the image, a certain orientation will provide the greatest likelihood estimate. This likelihood estimate represents how likely it is that the model is present in the image.

With respect to the first item in the previous paragraph, the Management Module goes into a "receive and send" state while each likelihood server receives its image and model pair. In essence, each likelihood server's "send and receive" state complements the Management Module's "receive and send" state. When the likelihood servers have computed their likelihood values and are ready to send them, the Management Module is in its receive state waiting to retrieve the new information.

Specifically, the Management Module waits to receive any computation result from any of the likelihood

servers. When returning a likelihood result that was computed to Management Module, that likelihood server goes into a "send and receive" state. In its "send and receive" state, each likelihood server sends the result of its most recent computation, and subsequently goes into its "receive" state where it waits for a new image and target model pair from the Management Module.

After the first likelihood server has sent the result of its calculations to the Management Module, the Management Module must perform a quick check on these values before sending a new model and image pair to the second likelihood server. Before sending a new pair, the Management Module compares the obtained result with any previous results. If the likelihood of this most recent result is greater than that of previous ones, it computes a rejection value for it and prints both the result characteristics and its rejection value to the screen. This sequence continues until each likelihood server has been sent an image and model pair for analysis.

In computing it, the simulation compares the distributions of the image and model that produced the most recent likelihood result. If the rejection value is too high, then the image should be rejected as representing an unknown target class. The upper limit of acceptable rejection values is an arbitrary issue that needs to be decided by the user of the system. Setting the limit too low can result in accepting false classifications as true, while setting the threshold too high can result in rejecting true classifications.

Thus, the simulation compares user-specified target models with any given image the user wishes to find a match for. Through such measures, this simulation simultaneously provides a running update of which model is most likely to be in the image and timestamps key events in seconds, such as: start up, the time of each new best result, and shutting down the recognition process. Once the simulation cycles through and compares to all the target models in the user specified model database folder, the simulation shuts down. While target recognition at this fundamental level can be very powerful, the evaluation of this simulation needed to analyze the sensitivity of system performance to both: a slower distributed network, and a network with fewer or more likelihood servers.

To simulate a target recognition system's operation on a slower network, the simulation code was modified with two minor changes. A sleep function, `usleep(x)` was embedded, where 'x' represents the number of microseconds the system pauses until the next operation in the function is called. The user can replace each 'x' with any integer and the simulation will add a corresponding number of seconds to the time taken to transfer an image, a model, their classification result. In this way, the simulation can pretend to run using a decreased network bandwidth.

Evaluators can easily test the simulation with a number of different use-cases by changing runtime parameters and examining the data visible through the user interface.

They can test the impact of a slower network or a network with fewer or more servers on the simulation's performance, and the impact of resolution of the target database models.

While any recognition process is running, a process-monitoring module displays the distributed system's information. Here, the user is able to observe what step of the simulation is being carried out, in addition to the amount of system resources being consumed while a particular task is being carried out. Specifically, the system statistics shown to the user include the amount of CPU time being used by a task, the amount of memory being used by that task, and the number of bytes required to store the task's data.

While there are a number of commercially available tools available to monitor programs, we found that monitoring the performance and resources consumed by a distributed system were important elements not addressed by the previously developed programs. It is for this reason that our simulation includes a customized process-monitoring program. By delving into the operation of existing programs and how they perform their monitoring tasks, system statistics were gathered to reflect the resources utilized accordingly.

Within Linux, a virtual proc folder exists that contains a number of virtual files used for recording up-to-date system information. Having gained access to the files in these folders, this allowed for an output of the system information desired. The information contained in these proc folders include details on the hardware employed on the cluster in addition to information on the usage of memory and CPU. In addition to the files that retain the overall system performance, the proc folder contains subdirectory folders that describe the processes running on the cluster (each named by the process ID it monitors). The files and folders listed under proc can usually be accessed without any special access permission.

Using several functions intrinsic to each CPU for collecting system information, one monitoring process per CPU was created for display purposes. Due to the use of multiple functions, several MPI routines were also used to report system performance.

The process monitoring module is a program that is essentially separated from the rest of the simulation process. Its execution is initiated by the Management Module, which does so by using the `MPI_Comm_spawn` function. This function starts the monitoring program from within the simulation by specifying the process the user wishes to monitor. It begins on one node and continues to each node in a round robin fashion until it has instantiated the number of copies of the program specified by the user.

After spawning the monitor program, `MPI_Send` and `MPI_Receive` functions are used to transfer the process ID of the simulation process to the monitoring process. Upon receiving the process ID of the simulation process, the

monitor process can then access simulation process's files in the proc folder, read in the simulation related information and then display them on the output screen.

When outputting the information, a system function time is displayed to show the time during which the resources were consumed. Furthermore, the process monitor is set on a continuous loop so that continues to output the most current information until the simulation process is finished. To account for the trade off between resource consumption and the frequency of monitoring the system, another function called `sleep()` was used in the monitoring process to allow the user to specify how frequently the data will be updated on the screen.

Through the methodology described above, the process-monitoring module is able to show the CPU usage and memory usage of the simulation process at the frequency specified by the user. The information gathered can be used in conjunction with the simulation results, potentially lending insight into the possible bottlenecks of this distributed system.

3 RESULTS AND ANALYSIS

Ultimately, performance was measured by focusing on the impact of the number likelihood servers available, level of model resolution, number of models used for comparison, and available bandwidth on elapsed time. Our statistical calculations and comparisons led us to determine that the impact of the available bandwidth followed by the number of likelihood servers were the most significant.

To determine how the resolution of the images processed simultaneously affected the performance of the system, we created a test case for target recognition on one image compared with two models, using two likelihood servers at 100% network bandwidth. We found that at level seven, the images were classified at a time comparable to levels five, six and eight, but significantly faster than at level nine. In addition, at that level, the image was consistently classified correctly and estimated the angle to within 5 degrees of the true angle. With the best prediction angle for the image at level eight, level seven produces a sufficiently farther angle estimate of the image, and therefore, the trade-off of time for higher level computations should be examined. See Table 1. Here, all target model resolution levels returned an angle within 5 degrees of the true angle, thus resulting in a small standard deviation.

Table 1. The level of resolution dictated how close the angle approximation could be to the actual image angle of 210 degrees. The time quantifies how long it took the system to calculate its best likelihood of the predicted model using 100% bandwidth and comparing with two models.

level	angle	Average time
5	213.75	3
6	208.125	5
7	205.313	11
8	209.531	19
9	208.828	43

As seen from Table 1, above, the time taken to complete the recognition process increased exponentially as a function of object model resolution. The average time required was 3 seconds, 5 seconds, 11 seconds, 19 seconds, and 43 seconds for levels five, six, seven, eight, and nine respectively. Constraints on system resources do not affect rate of an incorrect image classification if time allows for all of the calculations to run. However, having fewer resources does have a significant impact on the performance time of the system.

The impact was tested by removing system components and comparing elapsed time of classification and rate of incorrect image classification. See Figure 3 below. What we concluded was that the system should not be built with fewer than two processors. As seen in Figure 3, the performance of our system using any number of servers proved comparable up to level seven, where results produced show a significant increase in time when using a single likelihood server. Specifically, a t-score of 3.02 indicates that there is a significant difference at the .01 level of alpha in the time required to complete the calculations with only one server versus with two or three servers.

From the resulting data, we concluded that use of one server can be maintained when performing calculations for target models with a resolution level less than seven, as there is no substantial difference in computing time for using additional servers thought there may be in classification accuracy over many trials. As it clearly demonstrates an increased speed in target recognition above level seven, using two or three likelihood servers would be our recommendation. . Consequently, the difference between having two and three servers is not significant at the .05 level of alpha.

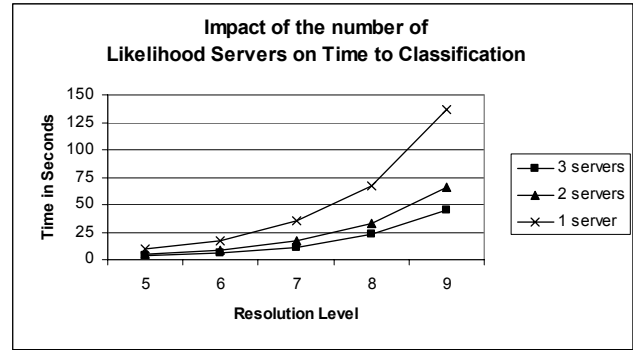


Figure 3. The above graph shows the impact of removing likelihood servers. The graph shows that the greater the resolution of the model images, the more impact the number of likelihood servers has on the system performance as measured by elapsed time of classification.

While the available network bandwidth affected the elapsed time to classification as expected, it did not affect rate of incorrect image classification or system resource usage. The greater the network bandwidth available, the quicker the results were obtained. The effect of bandwidth was greater when using fewer number of likelihood servers.

With a reduced network bandwidth, the tests run with only two likelihood servers executed slightly faster than the tests run using all three likelihood servers. While this was somewhat unexpected, tests run with only one likelihood server available took significantly longer than when more servers were used. See Figure 4.

The reductions in network bandwidth correspond to the number of microseconds the system paused for before transferring images, models, and their classification results. For example, pausing each transfer operation when comparing an image with 45 models would result in an overall increase of 135 seconds to the total transfer time. Dividing the total number of bytes sent by this new transfer time would yield the new bandwidth whereas dividing the old transfer time not including pauses by the same number of bytes would yield the old bandwidth. Computing a percentage difference between the two yields the percentage difference in bandwidth.

It is imperative to note that all bandwidth calculations and figures in this work are relative to the throughput provided by a 1 Gbps switched network. Thus, unlike the previous test varying the number of likelihood servers, the altered measure of bandwidth has led us to conclude that when performing recognition with target models above level six, at least 25.4% bandwidth should be available. If this is not satisfied, the linear trend between elapsed time and available bandwidth begins to take on a more exponential relationship. As a result, when the available network bandwidth was reduced, the time required for one server was still the greatest while three servers required the least. However, the relative increases in time were different de-

pending on the number of servers. For example, when 75% of the bandwidth was unavailable, the time to classification increased 5-fold, 4-fold, and 3-fold for 3 servers, 2 servers, and 1 server respectively. See Table 2. Therefore, while the relative multiplicative increase in time for one server was the least significant, with three servers the most dramatic.

Table 2. The above values were found by averaging the quotients of the times it took for the system to classify the image with the limited bandwidth and the times it took the system with full bandwidth available. The relative increase is greatest for 3 servers and least when only 1 server is used.

Bandwidth	3 Servers	2 Servers	1 Server
7.2% Available (Decrease by factor of 14.3)	21.33282	15.22303	11.41268
25.4% Available (Decrease by factor of 3.9)	5.558448	4.057278	3.58182

Thus, it appears that the more likelihood servers used in the system, the greater the impact the available bandwidth has on the total elapsed time required for classification. Since there is a strong correlation between available network bandwidth and elapsed time to classification, there is no optimal rate. The greater the available network bandwidth, the better. The conclusion drawn from these tests is that at least two likelihood servers are needed to optimize the time being used during the recognition process. In addition, since this variable does not affect rate of incorrect image classification, the engineers that design the system can determine how important the speed of classification is to the ultimate end-user rate and choose the available bandwidth appropriately.

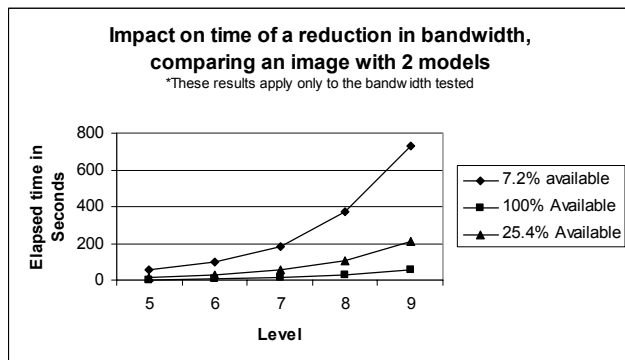


Figure 4. Each level or resolution requires a different amount of time to obtain the best likelihood value of the predicted vehicle type. In addition, the more bandwidth available, the faster the best likelihood was found.

Using the maximum bandwidth available, comparing the radar images with two models from the database took an average of 20.7 seconds for the algorithm to obtain the best likelihood value. Comparing the image with three models took an average of 31.6 seconds and comparing the image with four models took an average of 46.1 seconds. The graph below shows the results at each level, see Figure 5.

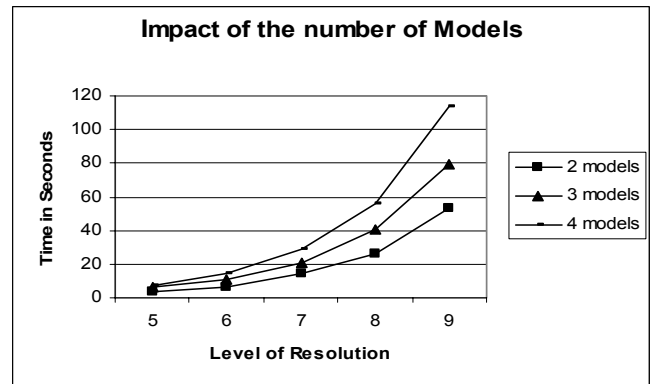


Figure 5. Each level of resolution requires a different amount of time to obtain the best likelihood value of the predicted vehicle type. In addition, the more models the system compares the image to, the longer the classification takes.

As shown in Figure 5, the greatest difference in times occurs at level nine. Again, the trade-off for a prediction with only a slightly higher log-likelihood value is not justified at all times. From the table on page five, resolution levels seven and eight also accurately classify the object and estimate the approximate angle as well. Regardless of how many models the system is comparing, operators of the system can decide to their liking whether level seven or eight resolution is the optimal resolution for minimizing time to process an image, without compromising accuracy. At level seven, the algorithm returned accurate results in an amount of time significantly shorter than level nine, and not significantly greater than levels five and six.

For all variations on the models being compared and the number likelihood servers available, the t-scores comparing the times between levels seven, eight, and nine are significant at the .05 level of alpha. In addition, most of the t-scores comparing the times between levels five, six, and seven are not significant. This means we can reject the null hypothesis that levels seven, eight, and level nine require the same amount of time in favor of the alternative that level seven computations require less time to complete. Similarly, for most operating conditions, we fail to reject that the level seven computations require a greater amount of time to complete than levels five and six.

Using the results of the tests conducted on the simulation, we would recommend that automatic target recognition systems be built with at least two processors. In addi-

tion, the target model database should concentrate on images of resolution level seven in order to minimize elapsed time of classification, and rate incorrect image classification.

4 CONCLUSIONS

Overall, the simulation was able to imitate a true system in receiving an image, assigning a recognition algorithm and image to likelihood servers, computing recognition algorithm statistics, and returning the object classification and angle to the user. Its functionalities included being able to compare the unknown radar image with multiple model objects, simulating a limited bandwidth, being able to specify the resolution level, and allowing the user to specify how many likelihood servers were available for computation.

By varying the component availability, rate of communication, and model resolution, the effect on system performance was through elapsed time, angle accuracy, and image classification accuracy. We found that the available bandwidth and the number of processors available for computation have the greatest impact on system performance. The significance of this impact is followed closely by the resolution level specified for the model objects. The higher the resolution, the greater amount of time that is required to complete the computations. We determined that the resolution of seven ensured image and angle accuracy while it did not sacrifice a significant amount of time. In addition, we recommend that the distributed system employ at least two likelihood servers at all times because reducing the number of servers to one is detrimental to optimizing time to best likelihood classification.

Our testing allows us to be able to report with confidence that similar automatic target recognition systems should be investigated and potentially employed. Being able to classify radar images quickly and accurately are characteristics that are desirable to soldiers and other military personnel and a system similar to ours could deliver that functionality.

RECOMMENDATIONS

A real system should be built using at least two likelihood servers. Further testing should be done to study the impact of greater than three likelihood servers on the system performance time. In addition, the model database should focus on level seven resolution in order to minimize time needed for classification, maximize probability of an accurate classification, and reduce the memory required to store the target models. The time-accuracy trade-off may not be justified to increase the resolution to level eight or nine. Furthermore, as students and experts attempt to study the system further, we recommend that more testing is completed with additional likelihood servers to find the

optimal number of servers to minimize time and impact of limited bandwidth availability. Another aspect that could be further examined in future study is the effect of algorithm complexity on the system performance and the optimal algorithm complexity given certain operating conditions.

ACKNOWLEDGEMENTS

This project was sponsored by the Office of Naval Research. The authors wish to thank Dr. Alan Van Nevel of the Naval Air Warfare Center for his help and guidance throughout the year.

REFERENCES

- DeVore & O'Sullivan. (2001, April). "Probabilistic approach to model extraction from training data." Algorithms for Synthetic Aperture Radar Imagery VIII, Proc. of SPIE, 358-366.
- Gelenbe, Erol (2000). System Performance Evaluation: Methodologies and Applications. Washington D.C.: CRC Press.
- Hummel, Robert (1999). SAR Signature Processing for ATR. Defense Advanced Research Projects Agency. <http://www.hpcmo.hpc.mil/Htdocs/Challenge/FY99/21.html>

BIOGRAPHIES

SAPTARSHI ADHIKARI – Originally from Vienna, VA, Rishi majored in Systems & Information Engineering and triple minored in Applied Mathematics, Computer Science, and Electrical Engineering. He is currently undecided about his plans for next year.

JAMES FANG - James will graduate with a double major in Systems & Information Engineering and Economics with a minor in Computer Science Engineering. Originally from Taipei, Taiwan, James is joining AMS in Washington D.C. next year.

GUENEVERE LINDGREN – Guenevere will graduate with a BS in Systems & Information Engineering with a concentration in Management and minor in Engineering Business. She is a member of the Society of Women Engineers, Tau Beta Pi, and INCOSE. Originally from Bay Village, OH, Guenevere is undecided about her plans for next year.

SHADI M. WADIE – Shadi is a fourth-year undergraduate student majoring in Systems and Information Engineering. Originally from Alexandria, VA, Shadi is entering

graduate school next year in the Systems and Information Engineering Department, studying and researching under Professor James H. Lambert.

MICHAEL D. DEVORE – Dr. DeVore is an Assistant Professor of Systems Engineering at the University of Virginia. He was the technical advisor for this project. He received his B.S., and M.S., degrees from the University of Missouri-Columbia, and D.Sc. degree from Washington University in St. Louis. He has held a visiting assistant professor in the electrical engineering department at Washington University in St. Louis and prior to that managed system integrity and operations at Amdocs, Inc. He is a member of IEEE and SPIE. His email address is <mdevore@virginia.edu>.