PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Explanation mode for Bayesian automatic object recognition

Thomas L. Hazlett
Rufus H. Cofer
Harold K. Brown

SPIE.

Explanation mode for Bayesian automatic object recognition

Thomas L. Hazlett, R. H. Cofer, Harold K. Brown

Department of Electrical / Computer Engineering
Florida Institute of Technology
150 West University Boulevard
Melbourne, FL 32901-6988
(407) 768-8000, ext. 8818 / FAX: (407) 984-8461

ABSTRACT

One of the more useful techniques to emerge from AI is the provision of an explanation modality used by the researcher to understand and subsequently tune the reasoning of an expert system. Such a capability, missing in the arena of statistical object recognition, is not that difficult to provide.

Long standing results show that the paradigm of Bayesian object recognition is truly optimal in a minimum probability of error sense. To a large degree, the Bayesian paradigm achieves optimality through adroit fusion of a wide range of lower informational data sources to give a higher quality decision - a very "expert system" like capability.

When various sources of incoming data are represented by C++ classes, it becomes possible to automatically backtrack the Bayesian data fusion process, assigning relative weights to the more significant datums and their combinations. A C++ object oriented engine is then able to synthesize "English" like textural description of the Bayesian reasoning suitable for generalized presentation. Key concepts and examples are provided based on an actual object recognition problem.

2. INTRODUCTION

In order to effectively use the Bayesian approach to Automatic Target Recognition (ATR), multisource data must be collected and fused. This fusion process is complex in the sense that it must probabilistically consider and coalesce a myriad of intertwining possibilities. The current state of the art is such that the designer must hierarchically dissemble the mathematics of the evidence fusion process into a computation tree which is both probabilistically meaningful and computationally feasible. To simplify the design process, it becomes desirable to create an object oriented C++ engine to codify the evidence fusion design and to provide an explanation mode capable of verbalizing the reasons for evidence fusion within any specific recognition decision.

By taking an object oriented approach using C++, the mathematical properties affecting the Bayesian paradigm can be incorporated into classes. Instances of these fundamental classes can then be used to capture various levels of the probabilistic evidence fusion tree. At the same time, by using classes, a description of operations taken at particular levels of processing can be directly associated with C++ objects. Using the ability to overload operators in C++, an existing operator can be redefined to specifically link higher level pdfs with the pdfs of lower levels. This then also allows a graphical representation of the tree structure together with the presentation of "English" like sentences developed as the descriptions of objects (evidence) at higher levels are linked to those of objects at lower levels. The final result is an interface allowing the user to descend through a complete explanation of an specific recognition decision.

3. BAYESIAN AUTOMATIC TARGET RECOGNITION THEORY

In the development of high performance Automatic Target Recognition (ATR) systems, there exists the need for final evidence accumulation from a variety of sources: incoming images; models of targets, scene background, and sensors; battlefield support data; inputs from target cuers, etc. The Bayesian approach has long been known to be the optimal decision process in the sense of minimizing the weighted cost of all recognition decision outcomes against a system mission. As theoretically shown in Van Tree[1] and other references[2,3], it functions by finding and comparing the various probabilities, $P(D \mid H_i)$, of observing the incoming image, D, when conditioned upon the various target recognition hypothesis, H_i, shown in Figure 1. Speaking simply and all other things equal, the Bayesian process states that it is reasonable to choose (or recognize) target H_i when the image, D, itself is most likely, i.e. $argmax P(D \mid H_i)$, to have resulted from the presence of target H_i in the scene.

 0-8194-0865-4/92/$4.00

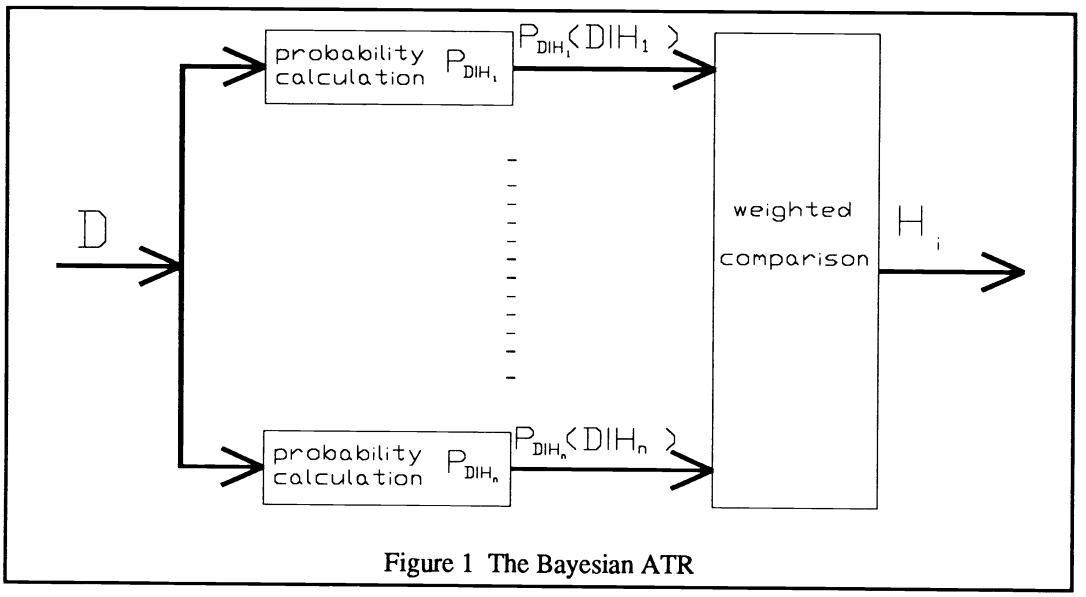

Figure 1 The Bayesian ATR

Obviously, any theoretical difficulty in developing a Bayesian ATR system centers on determining the various probability density function (pdf) forms, $P_{D|H_i}$. Reflection shows that these probabilities arise from underlying scene/sensor randomizations which in varying conditional ways can exhibit high degrees of independence and mutually exclusiveness. One of the authors[4,5,6,7] has recently spent considerable effort in identifying the natural pdf fracture points and weaving the results into a constructive process akin to that proposed by Pearl[8]. The natural fracture points include

1 The Mutual exclusiveness rule which introduces some new problem specific variable, C, to allow the decomposition of a pdf, $P_{A|B}$

$$P_{A|B} = \sum_i P_{A|C_i,B} P_{C_i|B}$$

into more simple constituents, $P_{A|C_i,B}$ and $P_{C_i|B}$ on the basis of the mutual exclusiveness of the values of C. It is invoked at the expense of developing $P_{C_i|B}$ from auxiliary data.

2 The Independence rule which allows the pdf of some composite data, A, to be decomposed

$$P_{A|B} = \prod_i P_{a_i|B}$$

in terms of the more elementary pdf's, $P_{a_i|B}$, of its constituents, a_i, i = 1, 2, ..., m, on the basis of the high degree of independence of each process a_i and a_j, $i \neq j$.

3 The Irrelevance rule which says that the probability of some variable A is not dependent on the value of some other variable, B, i.e.

$$P_{A|B,C} = P_{A|C}$$

Irrelevance occurs when the data B in no way influences the value of A. An example is the noise level, B, of a sensor which in no way influences the position, A, of a target in a scene.

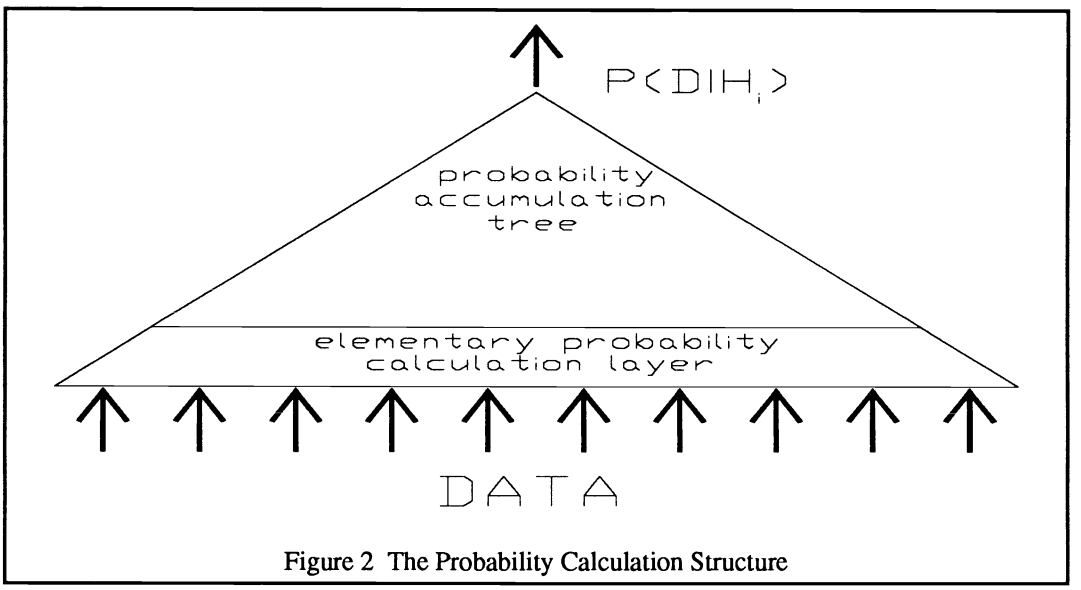

Figure 2 The Probability Calculation Structure

Adroit iterative use of these rules can result in substantial simplification of the complexity of the pdfs to be determined. However, while the overall problem is simplified, the intricacy of the computational structure grows, as shown in Figure 2. As a result, the Bayesian ATR designer needs access to an explanation mode[9]. This explanation mode should allow ready access to intermediate results throughout the computation to determine validity and tuning of the pdf accumulation process, efficient enhancement via subtree prunings and reorganization, sensitivity analysis as to quality of incoming information, etc. The remainder of this paper is devoted to the development of such an explanation mode based on the real life example of the next section.

4. TANK RECOGNITION USING THE EXPLANATION MODE

4.1. Underlying theory

During final evidence accumulation, the Bayesian decision process functions by comparing the probabilities, $P_{D|T_i}(D \mid T_i)$, of seeing the incoming image data, D, conditioned respectively on presence of the various target types, T_i, i = 1, ..., n.

Now a target, T_i, can generally be located at any of j = 1, 2, ..., m mutually exclusive locations, with probability $P_{\tau_j|T_i}(\tau_j \mid T_i)$. This probability of target location can be found from a variety of earlier ATR functions, e.g. target cueing subsystem. Mathematically, these considerations can be written as

$$P_{D|T_i}(D \mid T_i) = \sum_{j=1}^{m} P_{D|\tau_j,T_i}(D \mid \tau_j, T_i) P_{\tau_j|T_i}(\tau_j \mid T_i)$$

In many cases the location, τ_j, of the target is independent of the target type, T_i, resulting in the further simplification

$$P_{D|T_i}(D \mid T_i) = \sum_{j=1}^{m} P_{D|\tau_j,T_i}(D \mid \tau_j, T_i) P_{\tau_j}(\tau_j)$$

The problem now reduces to the determination of $P_{D|\tau_j,T_i}(D \mid \tau_j, T_i)$, i.e. the probability of seeing relevant image data, D, given a specific target type, T_i, at position τ_j. Begin by letting $D_t(i,j)$ represent the region of the image containing the target, T_i, at position τ_j and $D_b(i,j)$ the remainder of the image, i.e. the scene background of the target. Since the scene background and target appearance are generally independent, one can write

$$P_{D|\tau_j,T_i}(D \mid \tau_j, T_i) = P_{D_t(i,j)|\tau_j,T_i}(D_t(i,j) \mid \tau_j, T_i) P_{D_b(i,j)|\tau_j,T_i}(D_b(i,j) \mid \tau_j, T_i)$$

However, the background data contained within $D_b(i,j)$ is generally independent of target type and position leading to the further reduction

$$P_{D|\tau_j,T_i}(D \mid \tau_j, T_i) = P_{D_t(i,j)|\tau_j,T_i}(D_t(i,j) \mid \tau_j, T_i) P_{D_b(i,j)}(D_b(i,j))$$

In more sophisticated ATR designs the image data, D, can itself arise from multiple sensors. For instance, D can arise from LADAR (target range) and IR (target temperature) registered sensors producing images D^{LA} and D^{IR} respectively. In general the target's temperature and range data are independent as are the internal IR and LADAR sensor processes. This leads to the following breakdown of the $P_{D_t(i,j)|\tau_j,T_i}(D_t(i,j) \mid \tau_j, T_i)$ calculation

$$P_{D_t(i,j)|\tau_j,T_i}(D_t(i,j) \mid \tau_j, T_i) = P_{D_t^{LA}(i,j)|\tau_j,T_i}(D_t^{LA}(i,j) \mid \tau_j, T_i) P_{D_t^{IR}(i,j)|\tau_j,T_i}(D_t^{IR}(i,j) \mid \tau_j, T_i)$$

Image data on the target T_i at location τ_j is composed of pixels, d_k, k = 1, 2, ..., $l(i,j)$. In the LADAR (range case), once target type and position is known, the ranging data of pixel d_u, is generally independent of pixel, d_v, $u \neq v$. The reason of course is that scene randomness has been conditioned out, leaving only the randomness of the LADAR sensor. To a first order then, ranging pixels are independent, leading to the computational simplification

$$P_{D_t^{LA}(i,j)|\tau_j,T_i}(D_t^{LA}(i,j) \mid \tau_j, T_i) = \prod_{k=1}^{l(i,j)} P_{d_t^{LA}(i,j,k)|\tau_j,T_i}(d_t^{LA}(i,j,k) \mid \tau_j, T_i)$$

For targets at sufficient range, $P_{d_t^{LA}(i,j,k)|\tau_j,T_i}$ can often be represented as Normal with the modeled range as the mean and the LADAR ranging equation determining the variance.

In the case of IR, the target image can be affected by its state of operational readiness, r_g, g = 1, 2, ..., s. For instance, a simple case can be g=1: engine running and g=2: engine off. The probability $P(r_g)$ of the operational readiness can, for instance, be based upon auxiliary battlefield reports leading to the following probability breakdown

$$P_{D_t^{IR}(i,j)|\tau_j,T_i}(D_t^{IR}(i,j) \mid \tau_j, T_i) = \sum_{g=1}^{2} P_{D_t^{IR}(i,j)|r_g,\tau_j,T_i}(D_t^{IR}(i,j) \mid r_g, \tau_j, T_i) P_{r_g}(r_g)$$

While the probability breakdown of $P_{D|T_i}(D \mid T_i)$ can be continued, the above portion is sufficient to explain this example.

4.2. Hierarchical representation of the probability breakdown

Figure 3 illustrates the hierarchical structure resulting from the breakdown of section 4.1. The top level (A) represents the probability of an image existing when given the target type. This level encompasses a range (summation over j possibilities) of positions where the target (for our example a tank) may be in the image. The mutually exclusive rule is applied to the probability at each j. The result being the two probabilities on level B. The left branch represents the probability of an image given the target and the position of the target. The right branch supports the probability that the target is at position j. Level C fractures the left branch of level B into the region of the image containing the target and the region of the image containing background. The target region probability is broken into its LADAR and IR sensor data probabilities shown at level D. Level E has two fractures associated with it. The first is the product of the LADAR probabilities coming from the image pixels themselves (level F). This fracture is due to the independence between pixels from the LADAR sensor image data. The IR branch from level D consists of two probabilities which are mutually exclusive of each other: engine on and engine off. The number of IR levels can be taken down further, however this provides enough information to support the example code generation given in the next section.

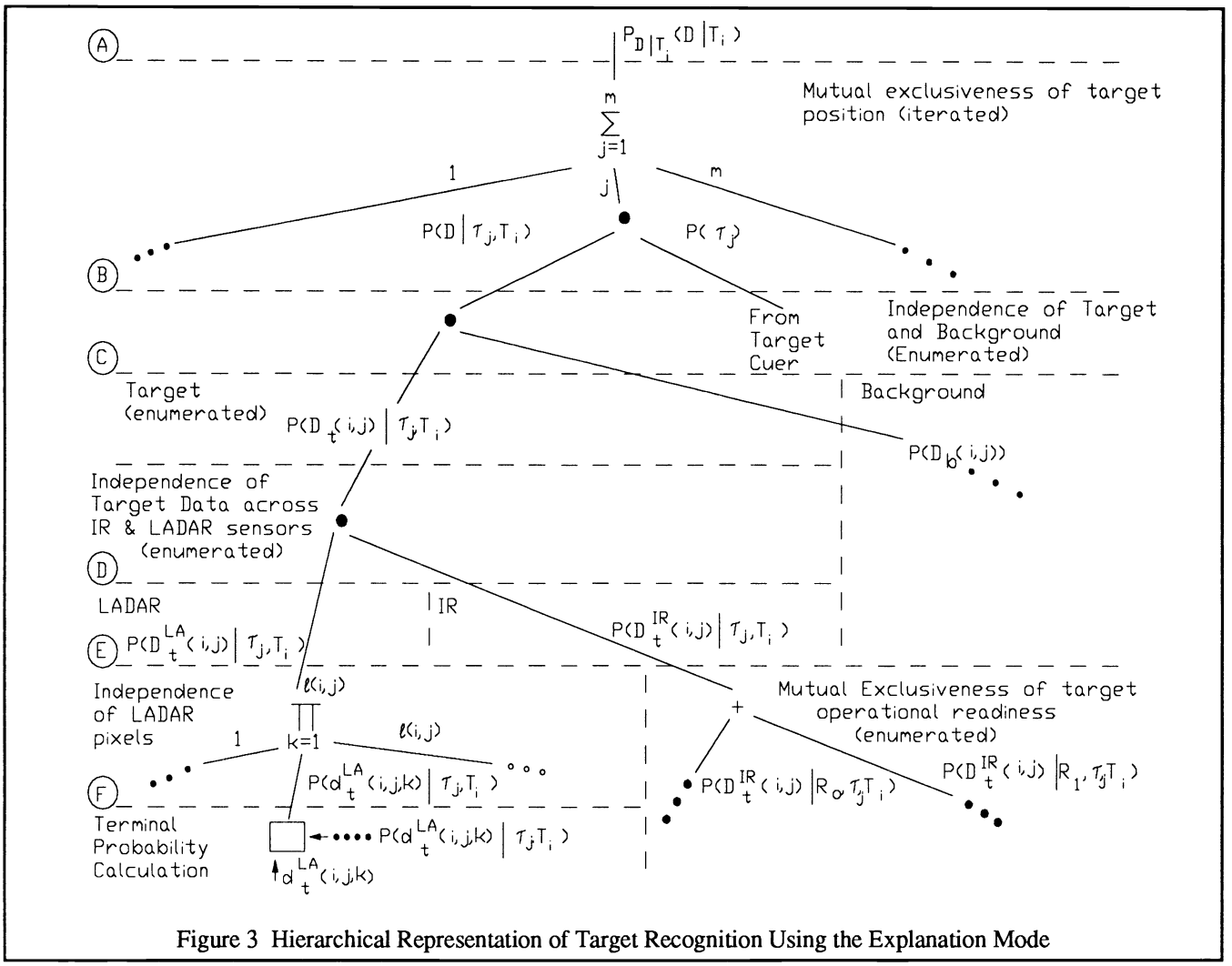

Figure 3 Hierarchical Representation of Target Recognition Using the Explanation Mode

5. OBJECT ORIENTED IMPLEMENTATION IN C++

5.1. Code generation

The C++ object oriented approach defines a single base class providing the fundamental information required by each of the additional classes. Other classes derived from this base class take on more specific information[10]. For our example, a base class called TerminalPdf is created containing information which is constant for the sensor systems being used (i.e. LADAR and IR). The position and the target type for this example are the same for the LADAR and IR sensor systems, therefore the TerminalPdf base class will contain data members associated with each of these data. However, other characteristics of these systems may be entirely different. The LADAR system can contain additional data pertaining to the relative distance of the target, whereas the IR system can contain data associated with the target's temperature. New derived classes are created by inheriting the base class TerminalPdf and adding additional data members related to the characteristics specific to the sensor system being used. For our example, the LADAR and IR systems have derived classes of TerminalPdfLadar and TerminalPdfIr respectively. Three additional classes are derived from the TerminalPdf base class, with additional features added for the division of higher level pdfs and the creation of the recognizer explanation. The following code segment is used to initialize both sensor systems and create a probability tree beginning from the bottom by utilizing iFLOW Technology[1], the sensor data, and by incorporating the three fracture points (probability rules).

1 iFLOW Technology and iFLOW are trademarks of Harold K. Brown

```
1    TerminalPdfLadar Ladar ("tank");
2    TerminalPdfIr Ir (Ladar);
3
4    // Create objects for iterative products and sums
5
6    Pi LadarTargetPixel(1D,"Pixel",Ladar.k);
7    Sum TargetPosition(1D,"Location",Ladar.j);
8
9    // Create all necessary object of type Result
10
11   Result LadarTarget("LADAR",Ladar);
12   Result ConditionalSensorEvidence("Was Present At",Ladar);
13   Result TargetProbability("Probability of 'Target'",Ladar);
14   Result Cuer("Being At Location 'j'",Ladar);
15   Result EngineOn("Engine On",Ladar);
16   Result EngineOff("Engine Off",Ladar);
17   Result IrTarget("IR",Ladar);
18   Result TargetRegion("Silouhetted Image Data",Ladar);
19   Result BackgroundRegion("Surrounding Background Region",Ladar);
20
21   // Determine the final probability
22
23   ((((((Image >> Ladar >> LadarTargetPixel >> LadarTarget) *
24    ((Image >> Ir >> ... >> EngineOn) + (Image >> Ir >> ... >> EngineOff))
25     >> IrTarget) >> TargetRegion) *
26      ( ... >> BackgroundRegion)) >> ConditionalSensorEvidence) *
27       ( ... >> Cuer)) >> TargetPosition >> TargetProbability;
28
29   // The ellipses (...) denote other portions of the decision tree
       which are not presented and are not valid syntax
```

In lines 1 and 2 of this code segment, two objects are created, Ladar and Ir, representing the two sensor systems LADAR and IR used in determining the existence of a target within an image. The Ladar object is constructed within a database with a target type of "tank." In order to make certain the target types are the same for each of the sensor systems, the object Ir is constructed using the Ladar object. When line 2 is executed, Ir is linked to the database for the target "tank" through Ladar. If other sensor systems were to be used, additional sensor system objects would be constructed the same way, and could use any of the previously created TerminalPdf derived objects as their initializer.

For this example only two iterative processes were required. The classes for these processes are Pi and Sum. The Pi class contains member functions which generate a table of pdfs within the database corresponding to a series of lower level pdfs which will be multiplied together. Sum's member functions create a table of pdfs within the database corresponding to the range of pdfs at the next lower level. Lines 6 and 7 create objects of these two processes. LadarTargetPixel is the object created using the Pi class and is initialized with a one-dimensional representation, an explanation of "Pixel", and the range of pixel values given by the LADAR sensor system which are found in a data member of the Ladar object. The object TargetPosition, of class Sum, is also one-dimensional (for this example), initialized with an explanation of "Location", and is given a range of position values given by the LADAR sensor system. Although this example provides a more descriptive hierarchy to the LADAR sensor system, it should be pointed out that the TargetPosition object could also be constructed with the IR sensory data.

The objects created in lines 11 through 19 from the Result class are all enumerated, not iterative like the Sum and Pi classes. The code-implemented tree structure in Figure 5 illustrates this point. The objects of type Result are responsible for retrieving information from the database and storing the results of either the sum or product of two probabilities. These two probabilities may be contained within other objects of Result class type or from the probabilities resulting from the evaluation of the data in the tables created by the LadarTargetPixel or TargetPosition objects. A member function within the Result class accumulates the data from the tables and formulates a single probability, thus assuring the one-dimensionality of these objects. Each object of class Result is initialized with a unique explanation depending on the level for which the object is defined. The Ladar object is also passed to each Result type objects' constructor in order to initialize the correct position and target type of the data members of objects defined within the database.

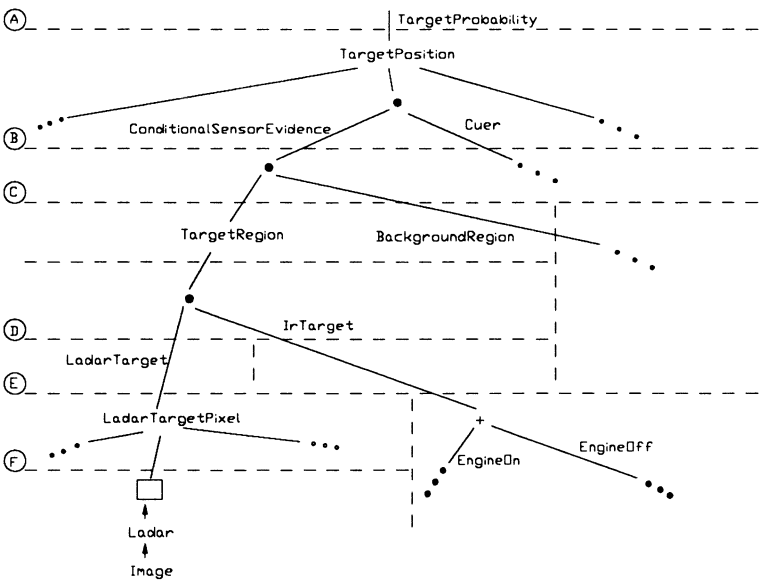

Figure 5 Hierarchical Representation Using Code Notation

The code segment located on lines 23 through 27 is essentially a single line of code within the program. This code generates the tree structure outlined in Figure 5 from the bottom starting with the raw data from an image. To achieve this, the ">>" (put-to) operator is overloaded, to accomplish upward linking between objects while at the same time creating a smaller database within the global database responsible for storing the explanations as they are linked together. The first upward put-to operator ">>" in line 23, detects that Image is raw data and sets an entry of the source list of Ladar to point to data. Next the Ladar member function for upward linking is called and updates the database for the output within Ladar. The output for the Ladar object would simply be a pdf resulting from the Image data. The next forward put-to operator separating Ladar and LadarTargetPixel in Line 23 detects that Ladar is a class derived from TerminalPdfLadar, and updates the source list of LadarTargetPixel. The source *list* is more appropriately used here considering the LadarTargetPixel object accumulates data from multiple sources. As before, the LadarTargetPixel member function for upward processing is called, and the database is updated. This process is continued until the output within the Result object LadarTarget is updated.

The same operations as described previously are implemented on the succeeding lines. The ellipses represent the further dissection of pdfs within the hierarchy. The operations on line 24 deal directly with the IR subsystem. The outputs generated within the Result classes EngineOn and EngineOff are a result of a number of lower level pdfs which are not shown for simplicity of the example. Once these outputs are updated, the addition operation is performed, resulting in a temporary Result object which is automatically created, and the upward put-to operator in line 25 detects that the type to the left is of Result (temporary), and once again the upward member function is called and the output within IrTarget is updated. The objects LadarTarget and IrTarget are now multiplied and the output is updated within TargetRegion.

The last two lines of the code segment operate on the higher levels of the tree structure, however the operations are the same and the database is updated in the same way. The TargetRegion and BackgroundRegion are multiplied and the upward put-to operator updates the output within ConditionalSensorEvidence. The output is eventually updated within TargetPosition, and once the last upward put-to operator is reached, TargetPosition is detected as a Sum having multiple sources and the output within TargetProbability is updated.

The links between the levels of pdfs are achieved through the development of tables within the database and not through the use of linked lists. This decision is based entirely on the amount of overhead required by the usage of pointers. However, the excessive amounts of data provided by sensor systems, coupled with the large number of levels the probability tree can be taken to, will inherently overload even high memory computer systems. Due to memory limitations not all of the probability tree can be stored, thereby making it necessary to eliminate some of the data. Considering that the higher levels of pdfs are derived from the lower level pdfs, the lower levels become useless due to their ability to be dissected from the higher levels. Therefore, as the probability tree is being generated, when the amount of memory becomes an issue the lower levels stored in the database are replaced by the higher levels being generated.

To expand the Bayesian technique to multi-processing, the put-to operations are converted to communications functions. When an TerminalPdf is created, as in line 1 and 2, a query is made to the processor manager to find an under-utilized processor. A command is given to the selected processor to create an image based on the TerminalPdf invoked. From that point on, each of the operators generates communication requests that interconnect the various objects. By using this technique, a system can be constructed utilizing a standard C++ compiler that produces executable code on each of the processors. The processors do not have to be of the same make. This type of processor management is contained within the Florida Institute of Technology's Open Parallel Architecture Design Specification (OPADS)[11], shown in Figure 4.

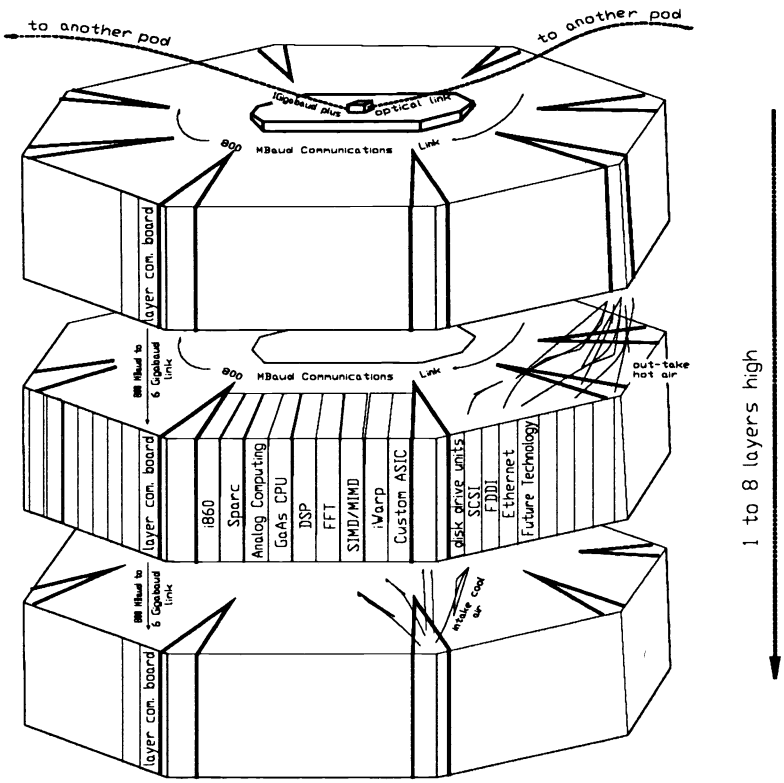

Figure 4 OPADS Development Platform for Implementation of Object Oriented Bayesian Recognizer

The OPADS platform allows technologies from various vendors (i.e. Sparc, i486, Josephson Junction, Analog, i860) to be integrated together to from a more effective system. When the code for the Bayesian recognizer is executed on a host system (i.e. Sun, PC or Macintosh) the program manager will search for the most efficient nodes, or sets of nodes, to execute the code onto. For instance, the platform may support a number of Digital Signal Processing nodes providing high speed implementation of the Bayesian recognition algorithm, with the program manager directing code execution to these nodes.

5.2. Explanation mode utilization

The following figures are representations of what the user might see when attempting to discover the reasons for an existing probability density function. Figure 6 shows the three views that the user would see once the program is executed. The image window contains a gray-level image of the target and any obstructions that may exist within the background scene. The probability tree window contains a tree structure similar to that in Figure 5 with probability values present at each of the interconnecting nodes. The data window shows the histograms of probabilities at various levels of the hierarchy. The use of these three windows allows the user to have a wide range of control over the ability to receive and understand the information. Figures 7 through 12 correlate directly with levels A through F of Figure 5, following a unique path to develop an explanation for the influences of a specific region of the target in the LADAR image to the overall probability. When the user selects the exclusive-or symbol, ⊕, below the histograms within the data window, the supporting level of probability development is shown.

This next level of development is displayed as both a histogram, below the higher levels, in the data window and as another branch in the probability tree window. When the user selects the probabilities at various locations on the histograms, windows will appear giving verbal, "English like" descriptions of the probability at that specific level.

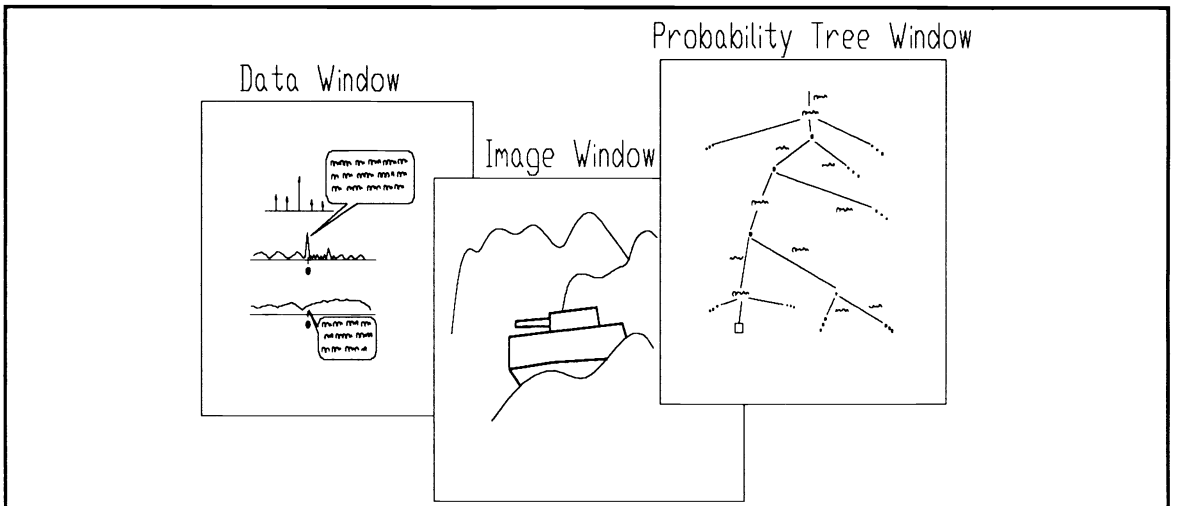

Figure 6 Three Views for Explanation Mode Monitoring

The example path development chosen by this sequence of figures is based on a hypothetical user's desire to determine the cause for the probability that the target was a tank (Figure 7). This is accomplished by selecting the high probability in Figure 8 which represents the probability of seeing the tank given it was present at a certain location. The actual location and an additional phrase associated with the location are added within the explanation window. The bottom histogram of Figure 8 represents the Cuer branch on Figure 5.

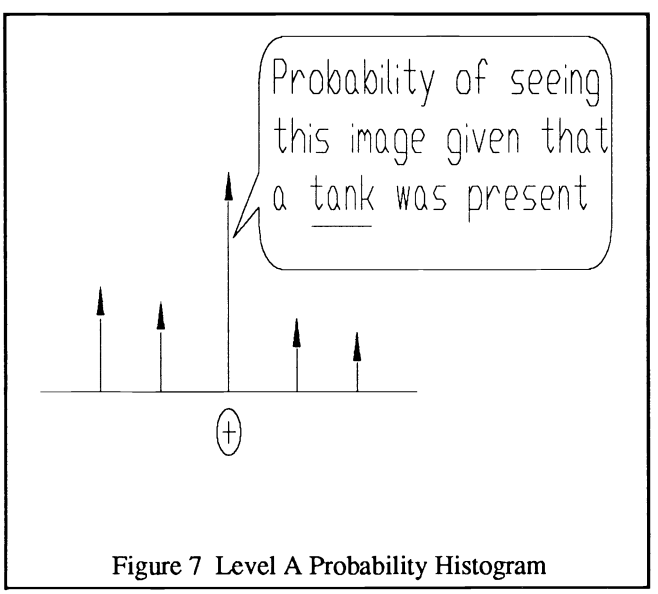

Figure 7 Level A Probability Histogram

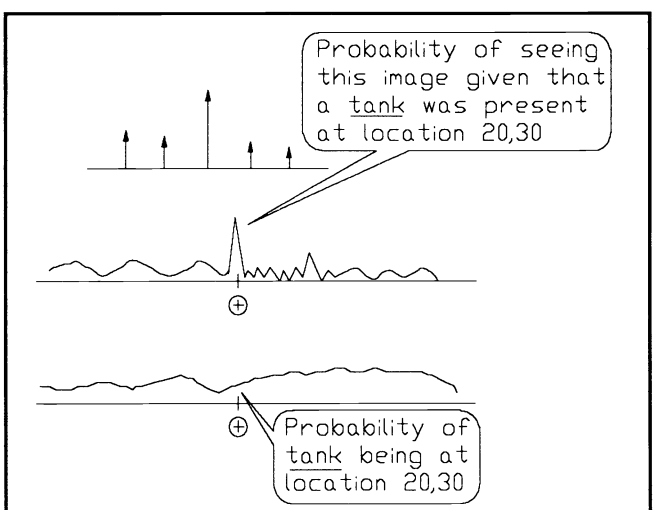

Figure 8 Level B Probability Histograms

The user selects to descend to a lower level and again the higher probability is selected in Figures 9. The histogram at this level represents the target and background regions. Illustrated are two spikes representing the probabilities of each of the two regions. The user then selects the target region probability and an explanation is given. The two probabilities in Figure 10 represented on the histogram are for the LADAR and IR sensor systems. When the LADAR probability is selected, the LADAR subsystem is included into the new explanation.

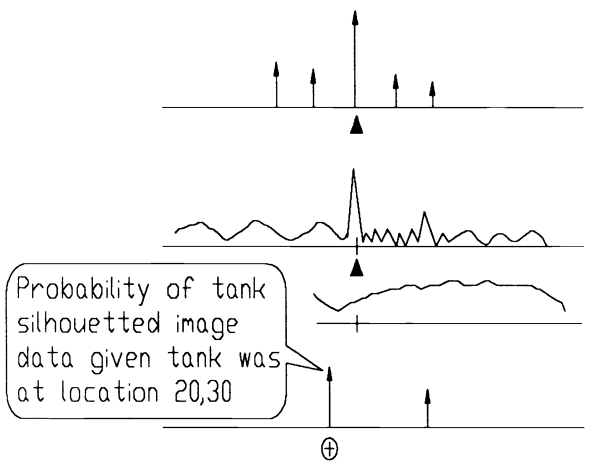

Figure 9 Level C Probability Histograms

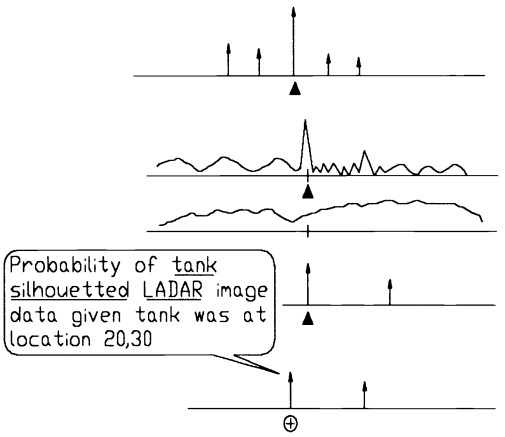

Figure 10 Level D Probability Histograms

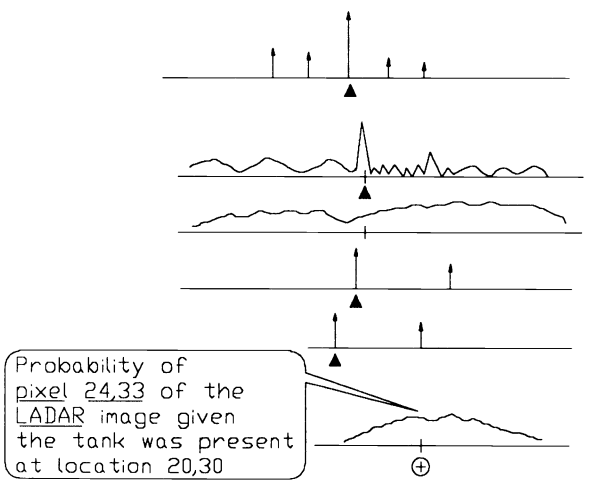

Figure 11 Level E Probability Histograms

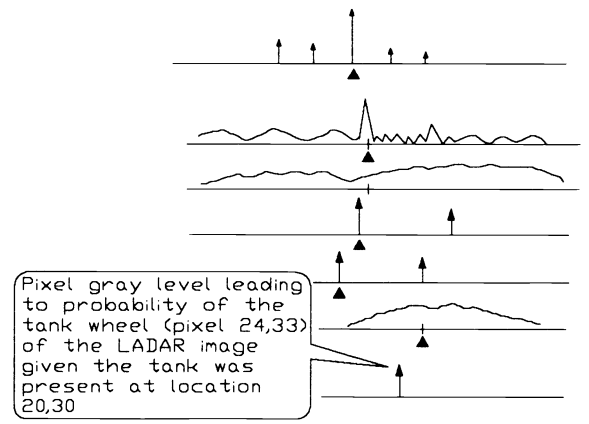

Figure 12 Level F Probability Histogram

As the user descends, the explanation becomes more descriptive as shown in Figures 11 and 12. Figure 11 provides an explanation of the pdf down to the level of the pixel location on the raw Image data. The final figure gives the most descriptive explanation, including the probability of the gray level of the image at a pixel location on the image. At this point the user may select any point on any of the above histograms, thereby causing the lower level histogram(s) to disappear and a new, different lower level histogram to reappear.

6. CONCLUSION

In order to optimize Bayesian ATR systems, it has become necessary to develop a means to verbalize the processes in the fracturing of pdfs using rules of mutual exclusiveness, irrelevance, and independence. By using these natural fractures, the problems can be simplified a great deal over the many iterations. A problem arises in that there are simply too many iterations to follow due to the indices involved with the problem. Two such indices are the target type and the position of the target. Together with a set of sensor systems, Bayesian recognition of targets can be implemented. However, even with these few datums, the problem of determining locations of questionable probabilities remains difficult.

Using object oriented programming in C++, code can be developed that will create a hierarchical structure, linking the levels of pdf decomposition. At the same time, explanations can be associated with the objects within the code segment in relation to their position within the probability tree structure. By overloading C++ operators, upward processing of data can be done automatically with an entire hierarchy being generated in a single line. This approach to implementation will also enable the recognition process to occur in a parallel system. With a small modification of the code, the Bayesian recognition using an explanation mode could be implemented over a system such as the OPADS platform.

User interface to the program will involve the selection of points in various probability histograms in order to receive a verbal description of the probability and supporting lower level probability histograms. This type of interface will thus allow the user to, in an orderly user friendly manner, understand key factors in how the probabilities of an ATR problem came to exist. This would allow a user to locate problem areas within a sensor network by simply traversing different levels of the tree until the invalid probability is located.

7. ACKNOWLEDGEMENTS

We would like to thank Tim Floore for the many drawings, and also for his work in editing this paper. This paper is a collaborative effort by individuals in two different areas of expertise, image recognition and object oriented C++.

8. REFERENCES

1. Van Trees, Harry L., Detection, Estimation, and Modulation Theory, Vol. 1, John Wiley, New York, 1968.

2. S. James Press, Bayesian Statistics, John Wiley, 1989.

3. Jain, Anil K., Fundamentals of Digital Image Processing, Prentice Hall, 1989.

4. Cofer, R. H., "Probability Determination for Range Imagery Bayesian ATR," Internal Report, USAF-UES Summer Faculty Research Program (1989).

5. Cofer, R. H., "Probability Event Spaces for ATR," Internal Report, USAF-UES Summer Faculty Research Program (1989).

6. Cofer, R. H., "LADAR Target Detection and Recognition," Final Report, USAF-UES Summer Faculty Research Program (1989).

7. Cofer, R. H., "Probabilistic IR Evidence Accumulation," Final Report, USAF-UES Summer Faculty Research Program (1990).

8. Pearl, J., Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, Palo Alto, 1988.

9. Rich, Elaine and Knight, Kevin, Artificial Intelligence, McGraw-Hill, 1991.

10. Ellis, M. A., Stroustrup, B., The Annotated C++ Reference Manual, Addison-Wesley, Reading MA, 1990.

11. Burgess, L., "An Open Supercomputer Emerges," Military & Aerospace Electronics, Westborough MA, Vol. 3, No. 1, p. 2, January/February 1992.