Understanding Ocean Surface Temperature Features

Technical Paper Number:
Submitted as part of the University of Cape Town,
Department of Computer Science Honours Project 2005

Nemanja Spasic
University Of Cape Town

Jared Tilanus
University Of Cape Town

Anet Potgieter
University Of Cape Town

Abstract

The aim of this project was to develop a prediction system that uses Artificial Intelligence, machine learning using training data and Image Processing (AI) to extract training data from Sea Surface temperature (SST) images to predict the ocean surface, temperature features around the coast of the Southern African region.

Region growing and histographic algorithms were used in the image processing section to extract thermal fronts as training data from the available SST images. A Temporal Bayesian Network was developed as the prediction model which used approximate stochastic learning and inference algorithms based on the Maximum Likelihood Algorithm (MLE). User-Centered Design (UCD) and Human-Computer Interaction (HCI) methods were used to develop user-friendly and easy to understand Graphical User Interfaces (GUI).

Results and evaluations of the project revealed that a generally successful prototype implementation of a prediction system that used AI, machine learning and image processing was developed.

Categories and Subject Descriptors:
G.3 [**Mathematics of Computing**]: Probability Statistics – *Probabilistic Algorithms, Stochastic Processes, Time Series Analysis*; H2.4 [**Database Management**] System – *Relational Database;* H5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Graphical User Interfaces (HCI), User-centered design*; I 2.4 [**Artificial Intelligence**]: Knowledge Representation Formalism and Methods – *Temporal logic;* I 2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving – *Inference Engine*; I 2.6 [**Artificial Intelligence**]: Learning – *Parameter Learning;* I 4.6 [**Image Processing and Computer Vision**] Segmentation – *Region Growing, Pixel Classification;* I 5.1 [**Pattern Recognition**]: Models – *Statistical;* I 5.3 [**Pattern Recognition**]: Clustering – *Similarity Measure*

General Terms:
Design, Theory, Human-Factors

Additional Key Words and Phrases:
Bayesian Network, Temporal Bayesian Network, Approximate Stochastic Learning, Approximate Stochastic Inference, Poisson Distribution, Human-Computer Interaction, Region Growing, Histographic, Segmentation.

1. INTRODUCTION

"Prediction is very difficult, especially if it's about the future."
Nils Bohr, Nobel laureate in Physics

Although the above quote, from Nils Bohr [10], perfectly describes the reality of prediction, it is one of the most common activities that is done in all aspects of today's world. Some examples of prediction include predicting the weather, predicting sports results, predicting people's behavior, predicting the age of matter using carbon dating, predicting experimental results, etc.

The uncertain or unforeseen factors that are involved in prediction make prediction so difficult.

There have been several attempts at minimizing the effect of such uncertain and unforeseen factors to make prediction more accurate. Some of the attempts include using experience, using scientific reasoning and using previous observations or data, known as "training data", as a guideline to the prediction. Lately, the use of statistics to model future predictions based on probabilities has gained interest within the researching and Computer Artificial Intelligence (AI) communities. An example of using statistics to aid with prediction can be found in Nielsen's work [11], which uses statistics to predict the risk in a wall condensation technique.

Although prediction can be narrowed down to a small prediction error rate using such techniques, for example prediction of weather, one can never be 100 percent certain that the prediction that was made is an accurate prediction. That is the main driving forcing behind finding better prediction techniques that will give better predictions about desired events.

This project focuses on detection and prediction of ocean surface, temperature features that are present in the Atlantic and Indian oceans around the Southern African Region using image processing, Artificial Intelligence (AI), machine learning and stochastic algorithms. Furthermore, the project presents the use of User-Centered Design (UCD) and Human Computer Interaction (HCI) to design user-friendly and easy to use interfaces for the prediction system.

The project was taken in collaboration with the Zoology Department at the University Of Cape Town and a French company named de l'Institut de Recherche pour le Dévelopement.

2.MOTIVATION AND BACKGROUND

2.1 Motivation

The collaborating companies expressed the following problem areas, which created the project domain:

- A visualization and prediction tool for SST images (including daily, 5 day and monthly).
- Identify persistent and re-occurring features
- Develop a prediction system that will be user-friendly and easy to use and interpret by non-expert users.

In addition to the above problems, the fishermen and anglers in the Southern African region could benefit form a prediction system that could monitor and predict ocean surface, temperature feature. The reason being, as Dan Rudnick suggests, "Oceanic weather influences the distribution of phytoplankton, which other marine life feeds upon." [13] Hence, by monitoring and predicting the ocean surface, temperature features the fishermen and anglers could predict where the most likely accumulation of phytoplankton would be found, thus leading to the most likely accumulation of marine life. Furthermore, biologists who study the migration of fish to monitor whether or not the ocean surface temperature features influence the migration of fish could also benefit from the project prediction system. Thus, the project would add to the value of Artificial Intelligence and computer science in today's world.

Finally, as Semtner A. [15] suggests that "only the most advanced parallel computers are fast enough to produce high-quality ocean simulations and accurate global climate predictions of temperature and precipitation." Furthermore, Wang P. *et al* [17] support the views of Semtner in terms of computationally expensive ocean models. Thus, there is a need for a prediction system that is not computationally expensive and can be utilized on a personal computer (PC), which what the targeted users of the system will have access to.

2.1.1 High-Level Project Requirements

From the stated problem areas, the following overall project requirements were established:

- Design software to detect the actual positions of temperature features from SST images
- Design a prediction model for prediction of ocean surface features over a time series period – the feature being: fronts, filaments and gyres
- Design a prediction model that will be able to identify ocean surface, temperature features that are persistent features and features that are re-occurring features.
- Design a prediction system, which will be able to use existing data as training data
- Design a prediction model that uses minimum system requirements in its prediction of ocean surface features
- Design a prediction model that will be user-friendly and easy to use for non-expert users

2.1.2Division of project work

The project was divided into two sections, which are detailed below: Image Processing and Understanding of the available ocean surface temperature data and Creation of a prediction tool, using AI and machine learning, which can predict ocean surface temperature features over time, using available data as training data.

2.2 Background

2.2.1. Threshold Algorithms

Thresholding is used to distinguish classes of pixels in an image based on their intensity value [16]. In the simplest (binary) case an image is segmented by assigning pixels to one class if they are below the threshold and to the other class if they are above or equal to it.

One can extend this to more than two classes of pixels, where a class is defined by an upper and a lower threshold. Where k is the number of classes, there are k-1 thresholds, and a class is defined as all pixels with an intensity value of between k_i and k_{i-1}.

Histographic methods are designed to automatically identify the optimal threshold to use in an area. A histogram of the pixel values in the area is created and examined for the existence of 2 peaks.

[12] describes a method where one segments the image using a threshold and evaluates the "goodness" of segmentation for each of the possible threshold levels for the image. If the maximum "goodness" value (the value corresponding to the best threshold level) exceeds a given limit, the area is said to have 2 populations.

The goodness measure is defined to be the ratio of the total variance to the within class variance.

[3] describe a very similar approach for applied to the identification of SST fronts. They apply this method on local, overlapping "windows" within the image. The size of the windows is a critical tuning parameter in this approach, as is the amount of overlap between these windows.

If a window is identified as having two populations the special compactness of the populations is evaluated. This compactness is measured as the ratio of edges between

pixels of the same class to the total number of edges between pixels.

A more statistically correct method for histographic threshold identification is to fit 2 Gaussian distributions to the histogram and take the intersection between these two distributions as the threshold value.

2.2.2. Region Growing
Region growing as the name suggests is the process of merging neighboring areas into larger regions to segment an image.

A region can be defined as follows: let R be and image and R, R_2, … R_n such that:

1. $\cup_{i=1}^{n} \mathbf{R}_i = \mathbf{R}$,
2. $\mathbf{R}_i$ is a connected region, $i = 1, 2, \ldots, n$,
3. $\mathbf{R}_i \cap \mathbf{R}_j = \emptyset$ for all i and j, $i \neq j$,
4. $P(\mathbf{R}_i) =$ TRUE for $i = 1, 2, \ldots, n$, and
5. $P(\mathbf{R}_i \cup \mathbf{R}_j) =$ FALSE for $i \neq j$,

Where P(R) is a logical predicate over the points in set R_i and Ø is the null set.
One can group them these methods into two main classes, seeded and unseeded.

Seeded region growing [1] starts from some set of points selected either manually or automatically and grows regions around these points. Pixels bordering on these regions are iteratively added to the seeded regions until all the pixels in the image are included in one of the regions. Thus in the final segmentation each region will contain exactly one of the seed points.

Unseeded region growing starts from many small regions and merges them based on some measure of similarity to form larger regions. These methods usually begin by dividing the image up as blocks. Split and merge [6] methods make these initial blocks large and then split them recursively until the blocks meet some homogeneity criteria. Other approaches include making each pixel in the original image a region, or making regular blocks of a small number of pixels. These small regions are then merged until some criteria such as the number of regions or a threshold in the difference between regions is reached.

2.2.3 Bayes' Rule
Due to the rule of symmetry in probability the conditional probability can be expressed without using the joint probability P (A,B). Hence, an alternative way to calculate P(A|B) is possible, and is known as Bayes' Rule. The rule can be modeled with the following equation, using *e* as an evidence variable and *H* as the hypothesis: $P(H \mid e) = P(e \mid H)\, P(H) / P(e)$. "The use of Bayes' Rule underlies all modern AI systems for probabilistic inference" [5]

2.2.4 Bayesian Networks (BN)
A Bayesian Network (BN) is an AI technique that uses probabilistic reasoning to calculate the conditional probability of an event occurring, based on related, existing data about the event, which is known as "training data." The calculations are based on a cause-effect relationship between the events in the network. However, a BN can not model temporal relations between and within the random variables.

A BN can be graphically represented using a directed acyclic graph (DAG) structure which has to conform to the following rules:

1. A set of random variables, either discrete or continuous, or uncertain quantities make up the nodes of the BN.
2. A set of directed links or arrows connects pairs of nodes. This is a representation of a cause and effect relationship.
3. Each node N has a set of conditional probability distributions P(Ni | Cause(Ni)) , which can be stored in a conditional probability table (CPT), that quantify the effect of the cause(s) or parent(s) on the node.
4. The graph has no directed cycles.
5. The root(s) of the network should have a prior probability P(root) in stead of the CPT

The following figure, adopted from [14], gives an example of a Bayesian network. It represents a situation where an alarm responds to a burglary, but also goes off when an earthquake occurs, notifying the police and the fire department.

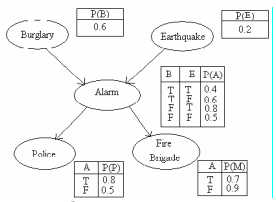

Figure 1: Showing a Bayesian Network

The tables next to each node represent the CPT tables for that particular node. P(X) denotes the probability of X being true. This could be extended to include the conditional probabilities of each node being false, by the simple calculation: 1-p.

2.2.5Bayesian Network Learning/Training
Before one can start using a BN, the network structure and parameters have to be specified from the available training data. Murphy K. [8] suggests that in order to train a BN, two operations have to be performed. Firstly, the structure

of the BN has to be established and then the parameters of the CPT tables have to be evaluated. Leaning the structure of the BN can prove to be an NP-hard problem and be much more difficult than parameter learning, especially when the BN has hidden nodes that are not directly observed from the training data.

There are four cases that have to be examined for learning a BN: when the network structure is known and data variables are observable, when the network structure is known and the data variables are hidden / missing, when the network structure not known but the data variables are observable when the network structure not known and the data variables are hidden/missing.

In the first case where the structure is known and the data variables are fully observable the leaning required would be parameter learning of the CPT. [8] and [14] suggest the use of a maximum-likelihood estimation (MLE) algorithm to do parameter learning in this case.

To illustrate this type of learning, the following example can be used, which was adopted from [8]: where a produces supplies bags of balls to a sport facility, which can contain only be soccer balls or volley balls. The probability distribution of the bags is not known. Hence the probability of getting a bag with only volley balls is θ, and the probability of getting a bag with only soccer balls is 1-θ. The fraction of volley balls in a particular bag can be denoted a hypothesis *h*. Suppose that N bags were received, of which there were (s) soccer balls and (v) volley balls. The following mathematical model can be used to demonstrate this example:

$$P(d \mid h) = \prod_{j=1}^{n} P(d_j \mid h) = \theta^{v} \cdot (1-\theta)^{s}$$

The MLE can be used to do parameter leaning in this case and is given by the value of θ that maximizes the above expression. Using the log-likelihood the above equation can be model as : vlogθ + slog(1-θ). Differentiating the equation with respect to θ one gets:

$$\frac{v}{\theta} - \frac{s}{1-\theta} = 0 \quad \Longrightarrow \quad \theta = \frac{v}{v+s} = \frac{v}{N}$$

Hence, the proportion of volley balls in a given bag is the proportion of observed volley balls from the existing data. This means that this type of learning is a simple count of occurrences over the entire data set. When the data is continuous, a Gaussian distribution can be used to perform MLE instead. However, the MLE algorithm has disadvantages with small data sets. A more detailed elaboration on the MLE algorithm can be found in [14], [8] and [9]

The second case, where the structure of the network is known, but some data variables are hidden or data is missing, according to [14] and [8], would be best suited to the Expectation Maximization (EM) algorithm which uses inference. The EM algorithm can be used to "find a (locally) optimal Maximum Likelihood Estimate of the parameters." [8] An exhaustive explanation of the EM algorithm can be found in [14], [8] and [9].

In the third case, were the network structure is not known, but all the data variables are observable a search through all possible network structures needs to be performed to determine which network structure fits the data best. This problem is an NP-hard problem. Possible solutions to this case are to use greedy algorithms (e.g. scoring metric or hill-climb algorithm) and clustering technique which find the most likely number of clusters. This case is exhaustive explained in [14], [8] and [9]

According to [8], the last case, where both the network structure is not known and the data variables are hidden or data is missing can be solved by using a combination of the EM algorithm and a network structure search. Murphy K. [8] also suggests that introducing hidden variables can make the BN model more compact and can sometimes lead to easier structure learning and better results.

2.2.6 Bayesian Network Inference/Querying

The basic task of inference in a BN is to establish the posterior probability of a set of query variables/ nodes given the state of some observed event / variable. In other words, this means that we wish to know the probability of and effect variable given the current state of the cause variables. Inference in a BN can be either exact or approximate. The choice of inference methods to use is dependent on the network structure. Furthermore, the choice of exact inference or approximate inference depends on the quantity and availability of data. If the data that is available is sparse and only a small quantity of data is available, then approximate inference would be more suitable because to perform exact inference, a large data set is required. Finally, exact inference tends to be more computationally expensive than approximate inference, hence is one is looking for a BN that is less computationally expensive and exact results are not a strict requirement, then approximate inference would be more suited.

Possible BN Learning Algorithms include, for exact learning: Enumeration [14], junction tree [14] [9] [2], variable elimination [14] [9], clustering algorithms [14], linear algebra (for Gaussian nets), Pearl's message passing algorithm (polytrees) [14] and for approximate inference: likelihood weighting [14], [9] and sampling algorithms, e.g. Markov chain Monte Carlo (MCMC) [14], [9]

2.2.7 Dynamic/Temporal Bayesian Network

A Dynamic Bayesian Network (DBN) is an extension of a BN to facilitate the modeling of temporal observations of stochastic random variables over discrete time steps.

Murphy [8] suggests that a "temporal Bayesian network" (TBN) would be a better name than "dynamic Bayesian network", since it is assumed that only the parameters of the model change and the model structure over time. Hence, a DBN will be referred to as a TBN in the rest of the report.

A TBN has to have the following CPTs for each variable: a prior probability, a transitional model CPT from time slice t to t+1 and a sensory model CPT for the attributes of the nodes/variables. These can be specified for the first time slice only as a TBN is assumed to be time-invariant, hence the structure of the TBN is assumed to be constant in each time slice. The first time slice is simply un-wrapped for the required number of time slices when an inference operation is performed. In such a way, a TBN can be converted into a BN. An example of a TBN would be the following example which relates to the project where an ocean temperature feature's (e.g. a front) attributes' change is being predicted over time. Hence, the attributes can either change or not. The attributes that the feature has are temperature and position. The following diagram could be used to model the TBN (unwrapped for 2 time slices):

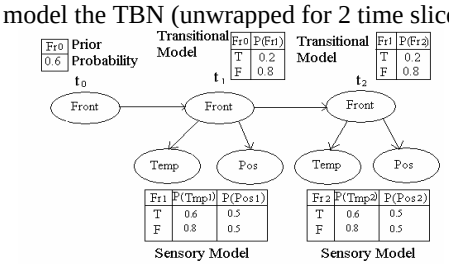

Figure 2: Showing a Temporal Bayesian Network (TBN)

2.2.8 Hidden Markov Models

A Hidden Markov Model (HMM) is a temporal probabilistic model which has only one discrete random state variable per time slice and models how that state variable behaves over time. The state variable is linked to a single discrete hidden node that forms the HMM. Hence, an HMM is the simplest kind of TBN. One has to note that all HMMs are TBN, but not all TBN are HMMs. [14] The example below taken from [8] demonstrates a possible HMM, where Q is the hidden node and Y is the observed node. The model is un-wrapped for four time slices:

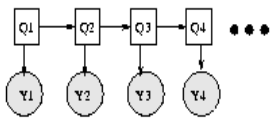

Figure 3: Showing a Hidden Markov Model

2.2.9 Temporal Bayesian Network Learning

Murphy [9] suggests that the learning techniques for TBN are "mostly straightforward extensions of the learning algorithms applied to BN." The only difference between TBN and BN is that online, as well as offline, parameter learning can be applied. This is a great advantage when the training data is continuously being increased or updated. As in a BN, a TBN has structural learning and parameter learning. Hence, most of the previously mentioned learning algorithms for BN are applicable to TBN. A discussion of all the learning algorithms that can be applies to TBN can be found in Murphy [9].

2.2.10 Temporal Bayesian Network Inference

The most common types of inference performs using a TBN include: filtering, prediction and smoothing and these are shown in the diagram below.

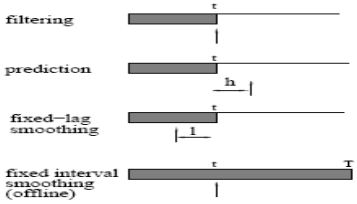

Figure 4: Showing various kinds of inference in a Temporal Bayesian Network [9]

Several algorithms exist to perform inference in a TBN. As in the case of BN, inference can be exact or approximate. The same ideology for choosing between exact inference and approximate inference in a BN applies to a TBN because the TBN is an extension of a BN. Some of the inference algorithms include for exact inference: forward-Backward algorithm Dugad [4] , Murphy [9], [14], Junction tree algorithm Barber [2], Frontier algorithm Murphy [9] and for approximate Inference the Boyen-Koller (BK) algorithm Murphy [9],the factored frontier (FF) algorithm Murphy [9], loopy belief propagation (LBP) Murphy [9], stochastic algorithms Murphy [9]

2.2.11 Poisson Distribution

The Poisson distribution (PD) is a "discrete probability distribution" [18] that expresses the "probability of a number of events occurring in a fixed time if these events occur with a known average rate, and are independent of the time since the last event." [18] There is no limit to the number of values that a Poisson random variable can assume which give it several advantages over other probability distributions such as the binomial distribution. As adopted from [18], the Poisson distribution of k, a poison random variable, is given as: $f(k;\lambda) = \frac{e^{-\lambda}\lambda^{k}}{k!}$, Where: k is a non-negative integer k = 0,1,2..n, *e* is the base of the natural logarithm (e = 2.71828...), k! is the factorial of k, λ is a positive real number that is, according to [18] , equal to the average number of successes occurring in a specified interval. The cumulative Poisson distribution, which is modeled as:

$$\sum_{k=1}^{1+T} f(k;\lambda) = \frac{e^{-\lambda}\lambda^{k}}{k!}$$

And can be used to get the cumulative effect all Poisson random events k0..kn, where kn is the desired Poisson random event.

2.2.12 User-Center Design

User-Centered Design (UCD) is a methodology used to design interactive computer programs for targeted users. UCD places the targeted users at the center of the design stage and builds the final product based on the reaction of the user to the continually evolving prototype. The international standard ISO 13407 [5] is a standard that provides guideline for UCD that should be followed when designing an interactive computer based systems. The standard specified UCD principles that have to be met to ensure a user centered design is achieved and also specifies the activities that should be carried out to ensure that the principles of UCD are met in designing a product.

3. METHOD

3.1. Histographic Algorithm

This algorithm follows the idea of the algorithms described in [3] and [12]. It segments the image by looking at local 25 * 25 pixel "windows" within the image. For each possible threshold level in the window (256 for the 1bit grey scale images used in this project) the optimal threshold is determined. Based on this threshold, it determines whether the window contains 2 distinct populations of pixels and whether these 2 populations are spatially distinct. If both of these criteria are true, a front is determined to exist.

For each pixel intensity level in the image it is assumed that two populations exist, separated by this threshold. This segmentation is then assessed by calculating the variance within the whole population V_T and the variance within each class V_1 and V_2 and calculating the ratio $R = (V_1 + V_2) / V_T$. The threshold that produces the highest value for this ratio is considered to be the optimum threshold.

The variance is defined as

$$\sigma = \sum_{i=1}^{N} P(x_i)(x_i - \mu)^2$$

Where x_i is the value of pixel i and μ is the average intensity of all the pixels in that window.

If the ratio value R corresponding to the optimum threshold is above a given threshold the window is considered to contain two classes. These classes are then checked for their spatial compactness by checking the ratio of pixels that occur on the inter-class boundary. The number of pixels that neighbor on pixels of a different class is calculated by looking at each pixel and its neighbors below and to the left. In this way all the boundaries between pixels are checked. If the ratio of this number to the number of pixels in the smaller class is above a certain ratio, the classes can be said to be in spatially distinct.

If the window passes both these tests, a front exists between the two classes identified. The pixels on the inter-class boundary are marked as edge pixels and

3.2. Region Growing Algorithm

This algorithm is an unseeded region growing algorithm that segments an image by growing homogeneous regions. Regions are defined here as a group of spatially coherent pixels. The image is initially divided into tiny regions, each containing one pixel. At each iteration the two most similar neighboring regions are merged to form a bigger region.

Several similarity measures were implemented in this project. They include one based on the difference between the average pixel intensities of the regions, one based on the edge strength and one based on a combination of these two, with deferent weightings.

On initialization each border between regions is located and an object representing this edge is created and placed in a priority queue. This object stores the difference between the two regions it connects, and thus the most similar regions can be found by taking the edge off the front of the priority queue.

When regions are merged, their point sets are joined. Since the similarity measure is based on the pixels in the region, all the edges which pointed to the old regions have to be updated. This requires removing them from the priority queue and re-inserting them with their new values, a very resource intensive job. Edges that, after a merge, end up joining the same two regions also have to be merged.

The regions are iteratively merged in this way until a stopping criterion is reached. This could be a threshold in the similarity measure, or a number of regions. The latter was implemented in this project.

3.3. Prediction Tool

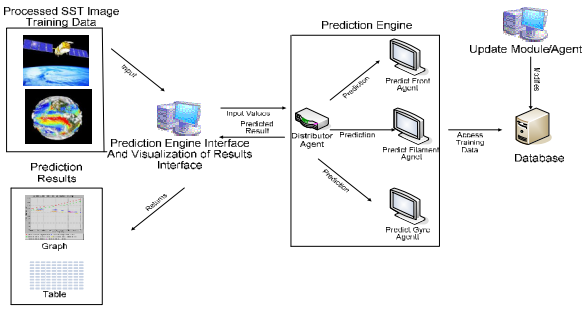

Figure 5: Showing the Prediction Tool

3.3.1 The Prediction Tool

The prediction tool was designed using AI agents for each feature prediction and was modularized to improve performance, database access and testing capability.

3.3.2 Temporal Bayesian Model

The TBN network chosen to represent each ocean surface, temperature feature was the following:

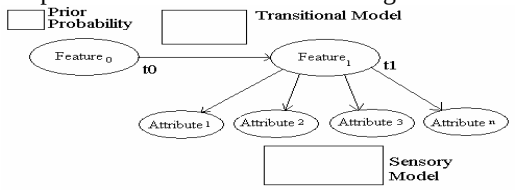

Figure 6: Showing the TBN model

3.3.3 Learning algorithm

An approximate, stochastic, online parameter learning algorithm was applied, which was based on the Poisson distribution. The learning algorithm was adopted from the MLE algorithm and was a by-product of the inference algorithm.

3.3.4 Inference algorithm

Due to the nature of the training data, an approximate, stochastic inference algorithm based on the MLE algorithm was used to perform inference. The current feature attribute was matched to the closest matching training set data to perform the MLE algorithm. In the case where a feature's attribute was a complex one, the Euclidean Distance was used to evaluate the closest training data match.

3.3.5 Prediction Tool Interface

A Prediction tool interface was designed to allow easy access to the prediction tool and to display the prediction results. The visualization of the prediction results was in the form of graphs and tables, which were found to be user-friendly and easy to use. The prediction engine interface was designed to hide the internal workings of the TBN from the end-user.

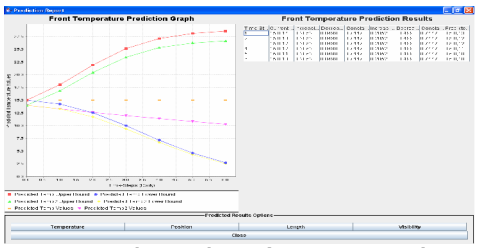

Figure 7: Showing the Prediction engine results

3.3.6 Database selection

A MySQL database server was chosen to store the training data.

3.3.7 Update Module

An update module was designed to allow the users to enter training data into the database and to hide all the internal workings of the database from the end-user.

3.3.8 User-Centered Design (UCD)

UCD was used in the form of participatory and PICTIVE sessions to ensure that users were at the center of the design stages of all interfaces that were created. Iterative prototypes were designed based on these sessions and were continually evaluated by users, then modified until the final user-friendly and easy to use interfaces were established. Human Computer Interaction (HCI) methods were also employed to ensure that a user-friendly interface was developed. In addition, HCI methodologies were used while evaluating the interface to ensure the participants of the sessions were treated with respect.

4. RESULTS

4.1 Image Processing

Due to the nature of the problem, one can not perform a rigorous evaluation of the segmentation. To do that one would require a ground truth set of images to compare to.

From a casual observation one can see that the histograhpic algorithm seems to identify fronts more reliably. It does not, however form continuous lines, as can be seen here

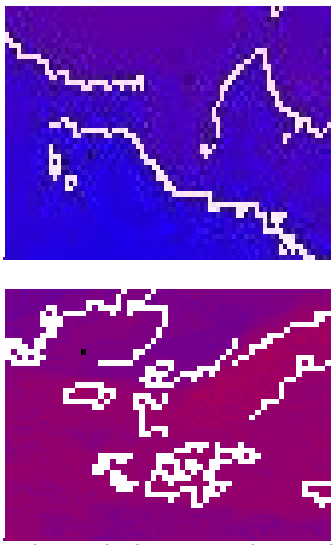

Figure 8: Showing the discontinuous lines produced by the histographuc algorithm

These are usually due to the window level boundaries, where different windows have different optimal thresholds.

The region merging algorithm is particularly sensitive to tuning parameters. This can be seen by the drastic difference between the images produced with different parameters.

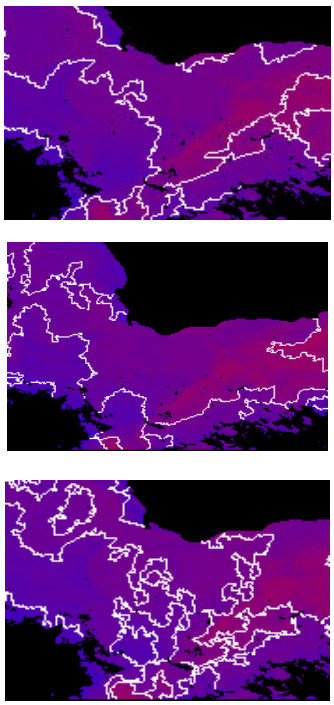

Figure 9: *two different sets of parameters for the region merging algorithm applied to the same image. The first image here is produced using a merging criterion of the difference between average pixel intensity and the strength of the edge, weighted equally. The edge strength is calculated as the maximum strength along the edge. The second image uses the criterion as the edge strength only, defined as the maximum strength of along the edge. The third uses a merging criterion of edge strength only, taken as the average over the whole edge.*

4.2 Prediction Tool

4.2.1 Learning Algorithm

The learnt Poisson rates were compared to Poisson tables and were found to match, hence this result showed that a successful learning algorithm

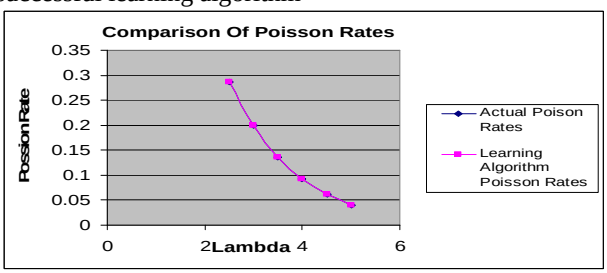

Figure 10: Showing results of multiple-cross validation

4.2.2 Inference Algorithm

The inference algorithm was tested using multiple-cross validation using a sub-set of the training data. Results showed that the error rates for the inference algorithm were varied and can be attributed to tuning of inference parameters using sampled training data. However, after further evaluation of the inference algorithm, results showed that a reasonably acceptable inference algorithm was developed.

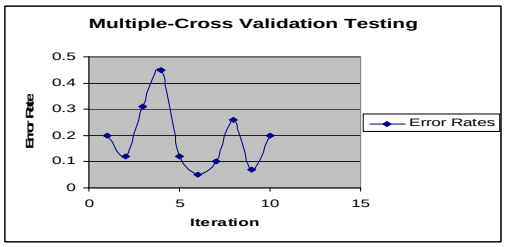

Figure 11: Showing results of multiple-cross validation

4.2.3 Evaluation of GUIs

The evaluations of both the update module and the prediction engine were done by means of questionnaires, and in general confirmed that the interfaces were user-friendly and easy to use by non-experts.

5. CONCLUSION

Comparing the delivered project to the project requirements, one can conclude that almost all the required goals were accomplished, except for the fact that the Image Processing part of the project could only detect thermal fronts, and could not detect filaments and gyres. Besides this requirement, all the other project requirements were met. Although the prediction engine was not tested using real-time data for filaments and gyres, it was tested using thermal front training data and was found to predict reasonably acceptable results. Hence, because the same algorithms were used in all three feature predictions, one can state that the prediction would also work for filaments and gyres, although this would have to be tested further.

Although it was not possible to rigorously evaluate the detection algorithms, they appear to recognize most fronts. The region growing algorithm is very sensitive to tuning parameters, and does not perform as well as the histographic algorithm.

According to the evaluation done, the interfaces designed were user-friendly and easy to use by non-experts, which met the initial project requirement.

Hence, to conclude, through a global view, the project was an implementation of a prediction system which could predict ocean surface, temperature features which used AI techniques, machine learning and Image Processing to extract training data successfully.

6. FUTURE WORK

6.1 Prediction model

The prediction model that was implemented was a naïve model in which all attributes and all feature were assumed to be independent. Hence, a good are for future work would be to extend the model to incorporate feature and

attribute interactions to design a more realistic ocean prediction model.

6.2 Learning algorithm
The learning algorithm that was implemented was a parameter learning algorithm that assumed that the structure of the network was known, As future work, this could be extended to a structure learning algorithm, which attempts to find the best possible network structure for the training data set.

6.3 Inference Model
The inference algorithm that was implemented was a naïve maximum likelihood inference which assumed attribute independence. This could be extended to an inference algorithm that takes into account the interactions with all other attributes and features for a more realistic inference prediction.

7. REFERENCES

[1] Adams, R. and Bischof, L. 1994. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:641–647

[2] Barber, D. 2003. *Probabilistic Modelling and Reasoning Dynamic BayesianNetworks: Discrete Hidden Variables*
Available at : http://anc.ed.ac.uk/~dbarber/pmr/pmr.html

[3] Cayula, J. F., and Cornillon, P. 1992. Edge detection algorithm for
SST images. *Journal of Atmospheric and Oceanic Technology*, 9, 67–80.

[4] Dugad, R et al. 1996. *A tutorial on Hidden Markov Models.* Indian Institute of Technology, Bombay, Signal Processing and Artificial Neural Networks Laboratory, Technical Report SPANN-96.1

[5] ISO 13407:1999 *Human-centred design processes for interactive systems*.

[6] Horowitz, S.L. and Pavlidis, T. 1974, Picture segmentation by a directed
split-and-merge procedure. *In Proc. 2nd. Int. Joint Conf. on Pattern Recognition*, pages 424—433.

[7] Lin, Z., Jin, J. and Talbot, H. 2000. Unseeded region growing for 3D image segmentation. Selected *papers from the Pan-Sydney workshop on Visualization*, p.31-37, December 01, Sydney, Australia.

[8] Murphy K., (1998) A Brief Introduction to Graphical Models and Bayesian Networks [Online]Available at: http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html Date accessed: 20/9/2005

[9] Murphy, P. 2002 .Dynamic Bayesian Network: Representation, Inference and Learning. Available at: www.ai.mit.edu/~murphyk/Thesis/thesis.pdf

[10] Niels Bohr, *Education on the Internet & Teaching History[Online]*Date Accessed:6/9/2005 Available at: http://www.spartacus.schoolnet.co.uk/USAbohr.htm

[11] Nielsen, A., (1995), *Use of statistics for prediction of risk for condensation in a wall construction cold walls radiation technique: the case of quality of air*, International Symposium On Moisture Problems In Building Walls, Porto - Portugal, 11 - 13 September, pp. 1.

[12] Otsu, N. 1979. A threshold selection method from gray level histograms, *Pattern Recognition*, Vol 9, no. 1, pp. 62—66.

[13] Robert Monroe (May/June 2002) *Underwater Weather: From liquid hurricanes to aquatic fronts, scientists seek to unlock the mysteries of oceanic weather*. Weatherwise magazine

[14] Russell S., Norvig P. (2003) *Artificial Intelligence A Modern Approach 2 ed* New Jersey: Pearson Education, Inc.

[15] Semtner A.(April 2000) *Ocean and climate modeling* ACM Press New York, NY, USA

[16] Sahoo, P. K. and Soltani, S. and Wong, A. K.C. and Chen, Y. C. 1988. A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, volume 41, number 2, pp 233—260.

[17] Wang P., Katz D. S., Chao Y (November 1997) *Optimization of a parallel ocean general circulation model* Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)

[18] Wikipedia, the free encyclopedia : Poisson distribution [Online] Available at:
http://en.wikipedia.org/wiki/Poisson_distribution
Date accessed: 25/9/2005