Finding Interesting Features in images using Neural Networks and Support Vector Machines

L. Machowski (0001084K)
Department of Electrical and Information Engineering,
University of Witwatersrand, South Africa.

Abstract
This paper presents a novel use for neural networks to aid in the automatic determination of interesting features in a large image. The network is given a downscaled version of the scene image as training data and the output is used as a high level representation of the scene image. The types of networks compared are MLP, RBF, SVM, Gaussian Approximation for an MLP and HMC for an MLP. The difference between the network output and the original image is thresholded in such a way as to define a user defined percentage of the scene image. This data is then passed through a Shrink Morphological Operator that reduces pixel regions in the output to single pixel values. These represent the positions of the regions of interest in the scene. The results are compared to the results from PCA and it is seen that PCA is much more efficient in terms of time and memory.

Keywords: Morphological Operators, Neural Network, Support Vector Machine, Bayesian.

I. INTRODUCTION

This paper presents a method for automatically finding interesting points in a large scene image so that they can be investigated for the presence of template images. The generalisation ability of neural networks is used to find difficult-to-learn regions of the scene which are categorised as interesting regions. These points can then be used in other template registration techniques to substantially reduce the search space of the registration algorithm.

The paper gives a brief introduction to the image processing field and some insight into neural networks and support vector machines. Next, the method proposed by the paper is described and the results from experimentation are presented. These are analysed and recommendations are made for further work.

A. Image Processing and Scene Analysis

Registering a template image to a scene is a difficult problem, especially if the scene image is large when compared to the template [1]. The shear number of possible combinations for the registration explodes as the number of parameters increase for the transformations of the template. With transformations like scaling, it is possible to fit the template at any location in the scene when the scaling is at one of the high or low extremes. The similarity of certain templates to other objects in the scene and the scaling problem make finding the optimal fit for the template a difficult, non-smooth optimisation problem with many local minima [2][3]. The real problem is that with so many possibilities for the template registration, where does one begin?

When performing typical scene analysis using image registration techniques, it is often the case that we are trying to fit an interesting template to a much larger scene. By interesting, we mean a template that has sufficient variation in its intensity values to give it features that distinguish it from other templates, and characteristics that define it as some object worth looking for in the scene. These templates typically have features that map directly to points (corners), lines (edges) and other features that may be abstracted at a higher level [1][4]. The ability to abstract information into a higher level is an essential tool for understanding and managing the complexity of the world around us [4]. This is one of the most powerful and fundamental abilities of the human mind [4] and it explains why humans have such good visual perception, especially when it comes to identifying objects in various scenes.

A typical problem with machine vision systems is that they are swamped by the shear volume of information that is accessible to them. This is why image processing is so important [1]. One can attempt to mimic the ability to abstract information by using a pyramidal approach to the image registration process [1][5]. The computational cost due to large image sizes is reduced by applying a coarse-to-fine hierarchical strategy to the data being processed [1][5].

This paper presents a method that may be used to find interesting regions in a large scene so that we have a small number of starting points that are likely to contain templates. One may then use optimisation techniques to see if a specific template can be registered near to these starting points [2][3]. It is important to note that the points found are independent of the templates used which means that we are looking purely at the characteristics of the scene. The approach taken is analogous to the way in which a person glances over a complicated scene. Their eyes look for areas in the scene that stand out, and then they determine whether this area contains the object that they are looking for. The interesting points in the scene are found by teaching a neural network what the scene looks like and then identifying the regions that are substantially different to this learned image. The networks performance is compared to when Principal Component Analysis is used on the scene and it is reconstructed with only a few principle components.

B. Principal Component Analysis:

Principal Component Analysis (PCA) is the most commonly used feature extraction and visualisation technique in

practice [6]. It allows us to represent a large amount of data in fewer dimensions so that it can be visualised. By truncating unimportant principle components, we can seemingly represent the data at varying levels of abstraction. This idea is illustrated in Figure 1 which shows the well known image of Lena reconstructed from 1, 2, 4, 8, 16 and 32 principle components. At each increasing level, we are able to distinguish more and more detail.

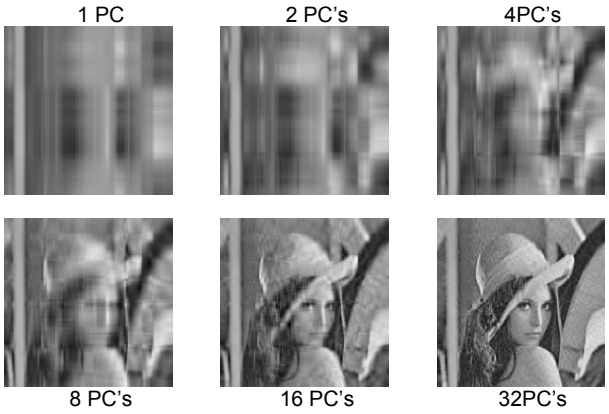

Figure 1 The effects of truncating Principle Components (PC's) in the reconstruction of the image of Lena.

PCA is popular for two reasons [6]: firstly it is fast and easy to compute, and secondly, it retains maximal information amongst all linear projections. The performance of each of the neural networks described in this paper is compared to using PCA for generalising the scene image.

C. Neural Networks

Neural Networks are interconnections of artificial 'neurons' that are greatly simplified versions of the biological neurons found in the human brain [7]. On a computer, the neural network is represented by a labelled acyclic directed graph with a clearly defined set of inputs and outputs [7][8]. The neural network may have any number of hidden units defined in an arbitrary amount of layers. There are several different architectures in existence, each one with its own advantages and disadvantages. The aim of the neural network is to be able to learn complex input-output relationships based on a source of training data, and then be able to make predictions for unseen data [7][8]. The neural network is used for its generalisation abilities and there is always a trade-off between generalisation ability and over-fitting [7]. It is possible to show that perfect generalisation with a neural network is not possible due to the fact that an infinite amount of neurons can produce output that is consistent with the training data [7].

There are three main regularization tools which can improve the performance of the network by ensuring that the network does not over fit data and that it generalises well[9]. The first is early stopping where a sub sample of the training data is kept separate to be used as test data. The progress of how well the network generalises for the test data is monitored while we are training it. Typically a sweep is done and then the network weights that gave the lowest error for the test data are used for the network. This set of weights represents the network that has the best generalisation performance (based on our test data). The problem with early stopping is that we need to keep aside a subset of the training data as test data. For successful training, we need as many training samples as possible so there exists a conflicting situation between the number of training inputs and the size of the test data that is used. Typically, $1/3^{rd}$ of the training data is used as test data. The second regularisation tool is the weight decay factor in the network. This allows the influence of old samples to affect the current weights much less than the current input. The old weight values decay over time. This allows the network to adapt to changes in the system that it is modelling. The problem with this is that the network could forget important information that it has learnt from the earlier samples. The last tool is the use of noise injection. The performance of the network can be improved for noisy and unseen data by training the network with a noise signal imposed on the training inputs or outputs. This makes the network more robust when it comes to predicting values for unseen data. The problem with this method is that we are artificially corrupting the data and therefore we have to ensure that the magnitude of the noise is not too great as to influence our network too much.

Neural networks can be applied to regression or classification problems. Regression involves fitting a curve to some sample data. Classification involves assigning labels to the sample data so that each point lies in either one of *n* classes [4][7][8][10].

Choosing the correct architecture is an important aspect of network design [7], so a brief review of five different types of networks is given below.

D. Multi Layer Perceptron

The Multi Layer Perceptron (MLP) typically consists of two layers of adaptive weights with full connectivity between successive layers [6]. This is shown in Figure 2. The MLP is capable of universal approximation at arbitrary precision of a smooth continuous function from a suitably compact region of the input space [6]. The activation function in the hidden layer is typically a tanh function because it contains both linear and non linear regions. The output activation function is typically linear for regression, logistic sigmoid for classification and softmax for mutually exclusive classes. The logistic sigmoid is used for this application because of its sharper output characteristic that is required to accurately represent image intensity values. The equation of the logistic sigmoid is given as [6]:

$$y_k = \frac{1}{1+\exp(-a_{k'}^{(2)})} \qquad (1)$$

where $a_k^{(2)}$ is the sum of all the weighted inputs and biases into the second layer of the MLP. The MLP is probably the most widely used architecture for practical purposes because

it is easy to train and provides good results for smooth functions. For most applications, there is no compelling reason to choose more general architectures [6].

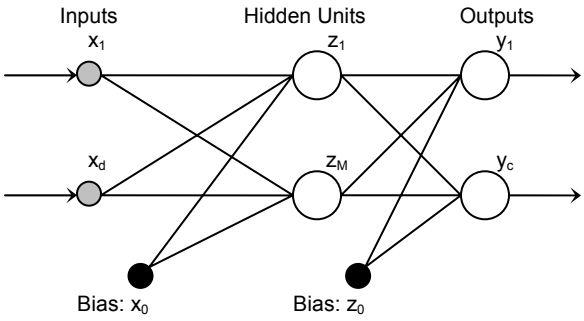

Figure 2 Architecture of a two-layer feed-forward network.

The efficient training of the network is performed by using back-propagation which is a general technique for evaluating derivatives of functions defined in terms of multi-layer feed-forward networks [6][9]. The algorithm is based on successive application of the chain rule of partial derivatives of the error. The error derivative is propagated backward through the network, starting at the output units [6]. The most important result of the back-propagation algorithm is that it allows the required derivatives to be calculated in $O(NW)$ time as opposed to $O(NW^2)$ time. This fact makes the efficient training of the network feasible.

The Hessian matrix is an important quantity that often needs to be calculated in the optimisation of the network weights. The Hessian matrix is defined as the second derivative of the error function with respect to the weights. The matrix plays an important role in several applications of the MLP networks [6]. The inverse Hessian can be used to prune insignificant weights in the network. It is also used to assign error bars to network predictions. The eigenvalues of the Hessian matrix can be used to determine suitable regularisation parameters for the network. The determinant is also used in evaluating the relative probability of different models. All of these reasons make it useful for improving the speed accuracy of the training process for the MLP.

E. Radial Basis Function

The Radial Basis Function (RBF) makes use of a two stage training process and a substantially simplified internal structure. This makes the network significantly faster to train and evaluate and provides a practical alternative to the MLP. The RBF computes the distance between the input vector and a weight vector (or centre) and passes this to an output activation function [6].

In the first stage of training, the centres are chosen either randomly from the training data or by performing some sort of a clustering procedure on the data. What makes the method really attractive is that the clustering procedure does not have to be accurate at all since only the distances to these centres are used for evaluating the input-output relationships. Another attractive aspect of the RBF is that the centres can be chosen in such a way that they have a physical interpretation. After the network has been trained, the network weights can be extracted from the network to investigate the relationships between outputs and the centres.

The speed of training the network is superior to that of the MLP but the accuracy of the network is worse. For this reason, the RBF is typically used as a sub-component in some other process that requires a network [6].

F. Support Vector Machines

Support Vector Machines (SVM's) are a special type of network that have a significantly different structure to the MLP and the RBF. The SVM maps the training data into a VC dimension which is linearly separable [11]. For classification, the training procedure finds a hyper plane that can separate the input data into its various classes. Refer to [11] for the equations on how the inputs are mapped to the outputs through the feature space. There are several kernel functions that are used for the actual mapping into the feature space and they include the Gaussian RBF, Exponential RBF, MLP, Fourier Series, Splines, BSplines, Additive Kernels and Tensor products [11]. The powerful aspect of the SVM is that one may manipulate the degree to which the hyper plane is fitted to the data as well as how sensitive the hyper plane is to scattered inputs. This allows the SVM to classify data that is known to have a random component in it. This ability makes the SVM unique when compared to the MLP and the RBF.

Performing regression with the SVM requires one to introduce a loss function that is affected by the distance of the support vectors from the hyper plane [11]. The e-insensitive loss function is used in this example because it allows one to vary an error bar for the regression which is useful if we have prior knowledge about the data that we are regressing. The loss function has a user definable dead-band which does not influence the optimisation of the line being regressed if the input vectors fall within this dead-band. The output of the training procedure gives us a set of support vectors that are actually used in determining the regressed line. This allows us to see exactly which samples in the input represent redundant data because they are not support vectors.

For this application, we use SVM's with Spline kernel functions because these can fit the image data exactly if the bound is set to infinity and the e-insensitivity is 0. The effect of making the bound closer to zero makes the regressed line more resistant to sharp changes in the line trend. The bound parameter essentially allows us to specify how much we want the SVM to over fit the line to the training data. This is another unique attribute of the SVM because it gives us direct access to the amount of over-fitting there is in the SVM. This is desirable since we do not want our SVM to fit the scene image exactly but to generalise it.

G. Bayesian Neural Networks

The Bayesian framework for Neural Networks poses the problem of identifying weights and biases in the following famous equation:

$$P(w \mid D) = \frac{P(D \mid w).P(w)}{P(D)} \quad (2)$$

where P(w|D) is the posterior probability distribution function, P(D|w) is the likelihood function, P(w) is the prior distribution function, and P(D) is the evidence [12]. Calculating the expectation of the output requires integrating the posterior probability distribution [6]. There are two practical ways in which this can be done. The first is Gaussian Approximation and the second is Hybrid Monte Carlo Sampling (HMC). The HMC method is better than Gaussian Approximation because the probability distribution is calculated directly and does not make use of assumptions. Both methods are capable of giving us the confidence on the prediction in the form of error bounds. This allows us to gauge the quality of the predicted values from the network when it is being tested.

1) Gaussian Approximation

By assuming a Gaussian approximation to the posterior probability of equation (2), we are able to use the evidence framework for finding the most likely hyper parameters for the network [6][12]. The evidence framework can be described as a stepwise process were one first infers weights for the network using an optimisation technique. We then maximise the evidence of the framework given these inferred weights [6][12]. The Scaled Conjugate Gradient (SCG) method is used for the weight inference because of its fast convergences and efficient use of available information [6]. The Gaussian Approximation is seldom used in real life because of the limiting assumption that is made about the underlying distribution. It is however attractive because an analytical solution is easily derived.

2) Hybrid Monte Carlo Sampling

Hybrid Monte Carlo (HMC) is an extension of the Metropolis-Hastings algorithm which defines a Markov Chain where a new state is generated from the previous state [6][12]. The new state is unconditionally accepted if the new state has lower error than the previous error. If this is not the case then the new state is accepted with a negative log likelihood probability. This method ensures that more likely states get accepted more often and therefore we get a representative sample of the underlying distribution. As the number of samples approaches infinity, so does the sample approach the true probability density. HMC makes the sampling of the weights for the MLP network tractable for real-world problems [6]. The mean value of the outputs from the networks optimised using HMC is the same as the value that would be achieved using other methods.

II. METHOD

The method for finding interesting features in images is given with the following steps (Where applicable, the Netlab toolbox is used [6]):

1. Load scene image.
2. Resize image to a smaller size suitable for networks.
3. For each Horizontal Scanline:
    a. Perform a linear regression for each line of the down-scaled image.
    b. Calculate the output scanline from the network and scale it back to the original image size.
4. For each Vertical Scanline:
    a. Perform a linear regression for each line of the down-scaled image.
    b. Calculate the output scanline from the network and scale it back to the original image size.
5. Calculate the overall generalised image by averaging the output from the Horizontal and Vertical scanline images.
6. Find the error in the generalisation by subtracting the learned image from the original image.
7. Perform an optimisation to threshold a certain percentage of the number of pixels in the original image.
8. Fill any holes in the remaining pixels.
9. Shrink the areas of connected threshold pixels until only one pixel remains.
10. The remaining pixels are the coordinates of interesting features in the scene image.

It is important to actually perform a proper image resize to down sample the scene image in step 2. Failure to do so will bring in aliasing problems in the sub-sampled result [5]. Size reduction must go hand in hand with appropriate smoothing [5].

After some experimentation, it was found that using the logistic sigmoid output activation function gives superior results because it has sharp output characteristics. This is important because typical scene images tend to have discontinuities in them. It was also found that 2D regression is not feasible for large images because the regression is poor.

The RBF is an attractive choice for the system because it is very quick to train and the accuracy in the regression that is required for the process is not very high. All that is needed is that the network identifies the general trend of the scene image. The SVM with a spline kernel function is another attractive option because we are able to modify the bound parameter to suite our generalisation needs. It was found, however, that having training data of more than 30 points for the SVM is computationally expensive and takes a lot of time to calculate.

Step 6 extracts the regions of the scene that are different to the generalised model. The reasoning behind this is that the regions that cannot be learnt successfully by a network are the regions that show some interesting characteristics of the image. The difference between the two images is the error in generalisation by the network.

A Nelder-Mead Simplex optimisation is used in step 7 to extract a specified percentage of interesting pixel values from the error image. The threshold value is modified until the desired number of pixels is extracted. Values of 10% of the original image size gave good results.

Steps 7 tends to create regions of pixels that define large areas of interesting regions in the scene. These may have holes within the regions so step 8 fills in these holes.

Step 9 makes use of morphological operators to dilate or erode the remaining pixels until all that is remaining is a single pixel at the object location [5]. The "Shrink" operator is used to remove pixels so that objects without holes shrink to a point [13].

What remains in the image is a set of distinct points that are located at interesting features in the scene image. These points can be used to start an optimisation routine to fit a template to the scene [2][3]. Various templates can be tried to identify if any of them fit the region in the scene.

All of the different networks were tried with the above mentioned method and the results are compared when a similar procedure is done using PCA. First the principle components are calculated for the scene image (which is much faster than training a network) and then another image is reconstructed from five of the most significant principle components. Steps 5-10 are then repeated on the new image. The results are given in the next section.

III. Results

When comparing the different networks, common parameters between methods were kept the same. The input image is a scene of Lena which is 256x256 pixels in size. The down-scaled image is reduced to 30x30 pixels before it is learnt by the specific network. The threshold process extracts as close to 10% of the image as possible before the image is fed into the final Shrink routine.

A. Multi Layer Perceptron

An MLP was trained with 10 hidden units and 50 training cycles using the Quasi-Newton optimisation algorithm. The results are shown in Figure 3.

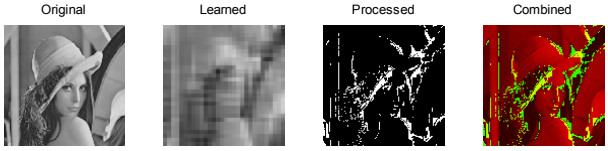

Figure 3 Results for MLP.

The total number of pixels after step 7 is 6554 pixels and this gets reduced to 314 pixels after doing steps 8-10. The routine takes on average 2.5 times longer then the RBF.

B. Radial Basis function

An RBF was trained with 10 hidden units, a Gaussian activation function and the Quasi-Newton optimisation algorithm. The results are shown in Figure 4.

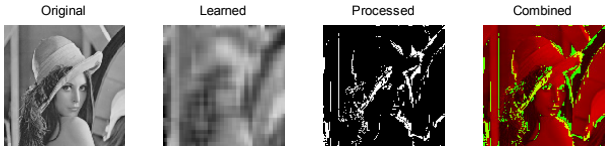

Figure 4 Results for RBF.

The total number of pixels after step 7 is 6554 pixels and this gets reduced to 262 pixels after doing steps 8-10. The time taken for this scene was on average 4.5s.

C. Support Vector Machines

An SVM was trained with a spline kernel function, a bound of C=1000 and an e-insensitivity value of 0. The results are shown in Figure 5.

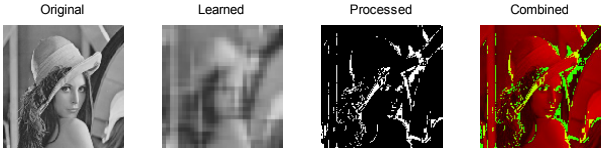

Figure 5 Results for SVM.

The total number of pixels after step 7 is 6554 pixels and this gets reduced to 253 pixels after doing steps 8-10. The routine takes on average 5.5 times longer then the RBF.

D. Gaussian Approximation

An MLP was trained with 10 hidden units, an initial alpha of 0.01, an initial beta of 50, the Quasi-Newton optimisation algorithm, 3 outer loops of 250 iterations for the optimisation. The results are shown in Figure 6.

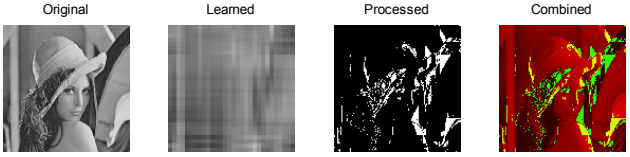

Figure 6 Results for the Evidence Framework.

The total number of pixels after step 7 is 6553 pixels and this gets reduced to 270 pixels after doing steps 8-10. The routine takes on average 71.5 times longer then the RBF.

It is evident that the quality of the learned image is poorer than most other networks. This can be attributed to the approximation in the method.

E. *Hybrid Monte Carlo*

An MLP was trained with 10 hidden units, an initial alpha of 0.001, an initial beta of 100, the Quasi-Newton optimisation algorithm. The step size was set to 0.002 with 50 steps in the trajectory, and the first 10 samples get omitted from each chain. The total number of samples for each scanline is 100. The results are shown in Figure 7.

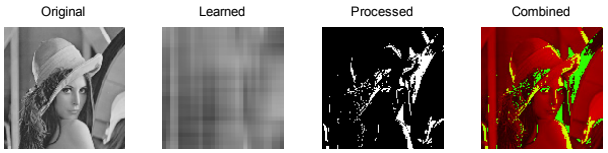

Figure 7 Results for HMC.

The total number of pixels after step 7 is 6554 pixels and this gets reduced to 219 pixels after doing steps 8-10. The routine takes on average 113.3 times longer then the RBF.

The poor quality of the learned image can be attributed to the relatively low number of samples used in the HMC sampling. Typical numbers in the order of 10000 are required but this number is infeasible for our purposes.

F. *Principle Component Analysis*

The reconstructed image was created from the first 5 principle components of the data.
The results are shown in Figure 8.

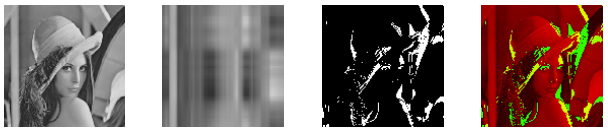

Figure 8 Results for PCA using 5 Principle Components.

The total number of pixels after step 7 is 6554 pixels and this gets reduced to 203 pixels after doing steps 8-10. The routine takes on average 0.7 times longer then the RBF. The PCA is more efficient in both time and memory.

G. *Discussion*

The overall performance of the algorithm works very favourably. In all cases were such features as the mouth, eyes and nose identified. This is a reassuring result that the algorithm is not highly dependent on the underlying error image. This means that similar results can be achieved even with a rough output from the network.

Steps 8 and 9 reduce the number of candidate points from 6000 down to about 250, depending on the scene image and the network used. This is a substantial decrease in the number of candidate points to try if each pixel was being used as a starting position.

The fact that most methods have a final pixel count of 254±40pixels gives a reassuring result that the number of interesting features in the image, no matter which method gets used is more or less the same.

IV. Recommendations

Finding a more accurate measure to compare each individual method should be investigated. The problem lies in quantifying which regions should be identified and how it can be measured.

V. Conclusion

From the results presented, we see that the proposed method for automatically finding interesting points in the image has some potential to be explored. It can help in reducing the number of starting points to try in another template matching routine. The use of neural networks provides a high level abstraction of the scene image so that the features can be extracted by taking a difference between the network-learned image and the original image. It was found that the actual performance of the underlying learned image does not affect the results of the final algorithm substantially. The use of PCA to perform the abstraction stage instead of the networks proves to be a much more efficient solution and therefore should be investigated further. The RBF may however be suitable in cases when scene data is corrupted or missing. Much was learnt about the behaviour of the various networks and a good feel was gained for what each one can achieve.

References

[1] Barbara Zitova, Jan Flusser. "*Image registration methods: a survey*"; Elsevier B.V.; Image and Vision Computing 21 (2003) 977–1000; June 2003.

[2] L. Machowski. "Insights on using Non-Evolutionary Optimisation Methods for Template Based Image Registration"; School of Electrical and Information Engineering; University of Witwatersrand; Unpublished; 2004.

[3] L. Machowski. "Evolutionary Optimisation Methods for Template Based Image Registration"; School of Electrical and Information Engineering; University of Witwatersrand; Unpublished; 2004.

[4] G. F. Luger; W. A. Stubblefield. "Artificial Intelligence and the design of Expert Systems"; The Benjamin/Cummings Publishing Company; California; 1989.

[5] B. Jähne. "Digital Image Processing: Concepts, Algorithms, and Scientific Applications, 4th Ed."; Springer; Heidelberg; 1997.

[6] I. Nabney. "NETLAB: Algorithms for Pattern Recognition"; Springer; London; 2002.

[7] M. Vidyasagar. "A Theory of Learning and Generalization"; Springer-Verlag; London; 1997.

[8] M. I. Jordan, C. M. Bishop. "*Neural Networks*"; MIT Artificial Intelligence Laboratory; 1996.

[9] R. Duin, D. de Ridder. "Neural network experiences between perceptrons and support vectors"; Papers from the BMVC97 Proceedings; 1997. Last Accessed 20/05/2004. http://www.bmva.ac.uk/bmvc/1997/papers/duin/duin.html

[10] R. Hecht-Nielson. "Neurocomputing"; Addison-Wesley Publishing Company; USA; 1990.

[11] S. Gunn. "Support Vector Machines for Classification and Regression"; Faculty of Engineering; Science and Mathematics; School of Electronics and Computer Science; University of Southampton; 1998. http://www.ecs.soton.ac.uk/ Last Accessed: 15/03/2004.

[12] T. Marwala; M. Lagazio. "Modeling and Controlling Interstate Conflict" to be presented in International Joint Conference on Neural Networks 2004, July 25-29, Budapest, Hungary; 2004.

[13] Mathworks Inc. "Image Processing Toolbox User's Guide V3" Mathworks Inc.; 2002. www.mathworks.com