Change Detection by Deep Neural Networks for Synthetic Aperture Radar Images

Frank Liao, Elizabeth Koshelev, Malcolm Milton, Yuanwei Jin, and Enyue Lu

***Abstract* --- In this Research Experience for Undergraduates (REU) project, we develop and implement deep neural network algorithms for change detection of synthetic aperture radar (SAR) images. Deep neural networks represent a powerful data processing methodology that integrates recent deep learning techniques on neural network computing frameworks to undercover underlying features and structures of observational data. The classic change detection method for SAR images is through the difference image analysis method, i.e. filtering the noise in each before-change and after-change image and then identifying the changes between the two images. Although well researched, the difference image method requires significant pre-processing and has difficulty with applications that require high accuracy and flexibility. The proposed deep neural networks create a change detection map from original SAR images directly without generating difference images, thus providing a novel framework for the change detection of complicated SAR images. We conduct numerous experiments on artificial images with added speckle noise and real-world synthetic aperture radar images.**

***Index Terms* --- deep learning, image change detection, neural network, synthetic aperture radar (SAR)**

I. Introduction

Change detection is an important research area in diverse disciplines such as computer networks, computer vision, and communications. In particular, image change detection refers to a process of detecting regions of change in multiple images of the same scene taken at different times. It is of widespread interest due to a large number of applications in remote sensing and surveillance. Synthetic aperture radar (SAR) is an important remote sensing technology due to the fact that radar signals can be used to take geographical images, regardless of time of day or atmospheric disturbances. Change detection in SAR images provides valuable information on terrain change on Earth and other planets. The classic change detection method for SAR images is the difference image (DI) analysis method. The DI method consists of three steps: (1) take two SAR images generated at different times and apply a filtering operation such as geometric correction, noise filtering and registration to align the two images with the same coordinates; (2) generate a difference image (DI) from the two aligned original images; (3) analyze the DI image and obtain a change detection map. However, there exist several challenges in the DI method. First, unlike standard optical images, SAR images contain multiplicative speckle noise. Speckle noise leads to image blurring and reduces spatial resolution, hence, it causes significant false alarms and poor detection performance. Second, the performance of DI method is determined substantially on the quality of the difference image. A good DI is essential to good change detection performance. Finally, how to design an efficient classification algorithm that ensures a high detection rate remains a critical and challenging step.

Neural networks are a biologically-inspired programming paradigm that enable a computer to learn certain features from observational data. On the other hand, deep learning has emerged as a promising data processing method that seeks to exploit unknown structures in the distribution of data in order to discover effective representation using learned features. Deep neural networks (DNN), a merge of neural networks and deep learning, currently provide the best solutions to many problems in image recognition, speech recognition, and natural language processing. In this paper, we utilize deep neural networks to detect change in multitemporal SAR images. The goal of this project is to simplify the processing steps in the conventional change detection algorithm by bypassing the steps of filtering or generating a DI so that the final detection results can be directly obtained from the two original images. To achieve this goal, in this REU project we implement a restricted Boltzmann machine network (RBM) [1, 2] with semi-supervised deep learning algorithms to achieve SAR image change detection. We investigate the impact of speckle noise and other types of noise on the accuracy of the network. We further investigate the parallelization of the RBM network and demonstrate the preliminary

This work was supported in part by the National Science Foundation REU Site: EXERCISE - Explore Emerging Computing in Science and Engineering under Grant CCF-1460900.

REU students F. Liao is with Carnegie Mellon University, Pittsburgh, PA 15213 (e-mail: recs7168@gmail.com). E. Koshelev is with Brandeis University, Waltham, MA 02453 (e-mail: lizkosheleva@gmail.com). M. Malcom is with Salisbury University, Salisbury, MD 21801 (e-mail: mmilton1@gulls.salisbury.edu).

Y. Jin is with University of Maryland Eastern Shore, Princess Anne, MD 21853 (email: yjin@umes.edu).

E. Lu is with Salisbury University, Salisbury, MD 21801 (e-mail: ealu@salisbury.edu).

speed-up of the deep learning algorithm.

II. Related Work

Recent research has shown noticeable advantages of deep learning algorithms for SAR image change detection and classification compared with traditional approaches [3, 4]. In [3], RBMs have been used along with a semi-supervised learning algorithm for multi-layer neural networks, wherein the method of using deep learning as opposed to other methods of creating change detection maps was explored. A restricted Boltzmann machine network was utilized and backpropagation (BP) was used to fine tune the neural network. Before training the network, either clustering algorithms [5, 6] or thresholding methods [7, 8] can be used to classify the pixels.

III. The Proposed Deep Learning Method

In this section, we provide a description of the deep neural network based change detection algorithm. The proposed method consists of three steps: 1) pre-classification for obtaining data with labels of high accuracy to train the network; 2) constructing a deep RBM neural network for the learning of image features with parameter tuning; and 3) using the trained deep neural network for classification of changed and unchanged pixels. Our algorithm is a variant of the work reported in [3].

A. Pre-Classification by Fuzzy C-Means Clustering

The proposed algorithm starts with a joint classification of two original images to avoid generating a DI. The pre-classification step chooses the pixels that are best suited to train the deep neural network. Here we use the Fuzzy C-Means (FCM) classifier [4, 5, 6] to jointly classify the two input images. The FCM clustering method takes gray levels of the image pixels as the inputs and establishes a similarity matrix $S_{ij} = \frac{|I_{ij}^1 - I_{ij}^2|}{I_{ij}^1 + I_{ij}^2}$ from the two input images, I_{ij}^1, I_{ij}^2 representing the gray levels of two pixels at the corresponding positions (i, j) in the two images, where $S_{ij} \in [0,1]$. Then, a global threshold value of similarity T will be applied to the similarity matrix S_{ij} by the iterative threshold method [5]. Next, the gray-level of pixels in the same position of the corresponding two original images are compared to each other to label the pixels. The label of a pixel and its surrounding neighborhood can be used to determine if a pixel is either part of an edge or noise. The results are then passed to the neural network for training. We note that FCM is an image segmentation technique that attempts to discover cluster centers [9]. Instead of determining a threshold of gray-level to separate pixels, FCM identifies the centers of its possible separations by assigning membership to each data point corresponding to each cluster center on the basis of distance between the cluster center and the data point.

Other clustering methods, such as the Otsu's Method [5] involves iterating over all possible thresholds and calculating a measure of spread for the gray-levels on each side of the threshold. The Otsu's Method requires this calculation to be done over a set of possible thresholds given by the user. The best threshold found by this method will be the one with the least within-class variance. Furthermore, brute force region detection [10] or watershed detection [11] can be more reliable, however, they are highly complex and require a significant run time, as well as often involving human intervention. In this project, we focus on the FCM clustering method for pre-classification.

B. Establishing and Training the Deep Neural Networks

After pre-classification, the neighborhood features of each pixel and its corresponding pixel in another image are converted into a vector as inputs to a neural network.

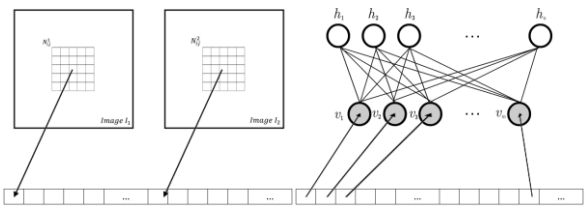

Fig. 1. Left: Vectorization of neighborhood features to be fed into the network. Right: The structure of an RBM. Trained two layers at a time, one visible and one hidden. These layers then become the layers of the Deep Neural Network.

The type of network is a binary restricted Boltzmann machine (RBM), which consists of a greedy "layer by layer" training. The RBM is restricted because a node cannot communicate with other nodes in its given layer. An ordinary RBM adjusts weights and biases in each node using a method called Gibbs sampling. Gibbs sampling attempts to reconstruct an input by feeding forward data to the second layer (called the hidden layer), and then trying to reconstruct the input back to the first layer (called the visible layer) through backpropagation. Then, weights and biases of each hidden node are adjusted through contrastive divergence and then another set of Gibbs sampling is implemented. The hidden layer then becomes the visible layer for the next training set and the process is repeated [12].

In the proposed DNN algorithm, a stack of RBMs is used for training networks using the features of images for change detection. The process involves the following steps. First, take neighborhood features of each pixel at the same location on the two images as inputs to the DNN. In this case, we take a sub-image of size $n \times n$ with the center being a pixel of position (i, j) from each of the two images. The two sub-images are converted to an aggregated feature vector of size $2n^2$ as the input vector to the DNN. Second, each RBM in the stack of RBMs is trained through Gibbs sampling in a phase called pre-training. The RBMs are then unrolled to create a deep

neural network for training. Finally, the DNN is fine-tuned through the backpropagation of error derivatives [12]. The features that the deep network learn can be extracted from the hidden layers. The first few layers learn simple patterns such as lines. Each layer uses the patterns learned from the previous layers to build more complex patterns represented in the input images. This explains why the DNN is able to work with high accuracy despite the presence of speckle noise.

After the training, the deep neural network is established. At this point, two test SAR images can be fed to the network and a change detection map will be produced.

C. Optimization of Network

To create different networks, many parameters will be adjusted, for example, the number of layers, the number of nodes in each layer, the batch size, the number of iterations that each data set trains with, the way the network trains, the training set, and many others. The overarching goal is to find the type of deep neural network best suited to the analysis of SAR images and to understand and see how adjusting each of these parameters affects the performance of the DNN algorithm.

The training set is essential in how the DNN will learn features. Both the size and diversity of training features changes the output of the deep neural network significantly. Choosing an optimal batch size is imperative as well, as it determines which subsets of data are used to train the network. The number of iterations is the number of times Gibbs sampling is utilized to train each layer. In general, more layers equate to higher capability of feature detection of the network. However, having too many or too few nodes for each layer may result in overfitting, i.e. the network may not learn features at all because the network is overly complex for the data set it is analyzing. Other factors to improve the network also include adjusting the learning rate, momentum, weight-decay, and drop-out rate, etc.

D. Understanding Thresholds

It is imperative to understand that the network output is a number between 0 and 1 for each pixel—if the number is closer to 1, the feature is more likely changed to be lighter. If the number is closer to 0, the feature is more likely changed to be darker. Thus, there are 2 thresholds that need to be set in order to create a change detection map. The issue is that these thresholds change depending on how much darker or how much lighter an area became, which makes finding an optimal threshold a challenge. The optimal thresholds also change depending on the image and type of noise. We tested the DNN performance using artificial images and real-world SAR images as preliminary results.

E. Parallelization

The proposed DNN algorithm was implemented as a Matlab program. The Matlab code was parallelized through Matlab's parallel processing toolbox to speed the process of sorting and sifting data to train and test the network. The process of taking ten 300 by 600 images and sorting out 180,000 features from each image, took around 3 hours on a laptop computer. To speed-up the computation, we used the "*parfor"* (parallel for loop) function to accelerate the process of sorting out images' pixels into neighborhood features by splitting the task among several Matlab workers. The original time was 3.27 hours, and the time after parallelization was 1.07 hours. We were able to cut down the time to 32.7% of the original run time just by sorting features. This includes pre-classifying data, sorting data, creating and training a network, and testing the network. The larger the images and the more images there are used to train the network, the more time is saved.

IV. Experimental Studies

We describe the experimental setup to create, train, and test DNNs with SAR images. The first step in training the network is to have a good training set. To understand which order, size, and diversity of labels are ideal to train the network, four groups of features and labels have been tested so far with the regular structure from the network. First, a set of 180,000 features from pre-classification taken as good samples from a set of 10 before-change and after-change 300 x 600 SAR images, a set of 700,000 pixels from 30 sets of images, a controlled set of 90,000 features that have alternating changed and unchanged labels, and another controlled set of 180,000 that has every fourth label changed. Based on results from testing with subsets, an ideal percentage changed and order is to be found.

The next step in training the network is to decide how to train the network. In this project, we used restricted Boltzmann machines. Further, we may adjust the structure or the network. Different sized networks were tested. Initially, we used a 5-layer network with 50, 250, 200,100, and 1 nodes for each layer, respectively, denoted by [50 250 200 100 1]. We then used a 3-layer network with 50, 250, and 1 nodes for comparison. Thus, we were able to compare change detection performance under different sizes of networks.

Once the training set, the training style, and the structure of the network are determined, the last step is to test different parameters of the network such as the batch size and the number of training iterations. The batch size is used to train the network through subsets of data, instead of the full data sets. The batch size of the network was chosen as 100. Training iterations were specified as 50 [3]. Both of these numbers were adjusted in the experiments and the results were then analyzed.

To measure the accuracy of the performance of the networks, a ground truth change image was compared to the change detection map. In each given change detection map, measurements of accuracy are taken as follows: false positives (FP), in which unchanged pixels are detected as changed, false negatives (FN), in which

changed pixels are detected as unchanged, true positives (TP), in which changed pixels are detected correctly, and true negatives (TN), in which unchanged pixels are detected correctly. From these, a percentage correct classification (PCC) is given by:

$$PCC = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

Next, the percent changed labels are tracked as well as the number of labels used to train the initial network. Overall error (OE) is represented by 1-PCC. A snapshot of sample labeled data from the original set of 10 training images is shown as follows:

TABLE I
Statistics from Change Detection Results

Original Set							
Labels	PCC	OE	TP	TN	FP	FN	% change
1000	0.9441	0.0558	165724	0	0	9801	5.50
10000	0.7165	0.2834	116803	8965	48921	836	11.93
20000	0.6363	0.3636	104046	7652	61678	2149	11.82
30000	0.8815	0.1184	145877	8853	19847	948	15.51
40000	0.8586	0.1413	141836	8885	23888	916	15.67
50000	0.9441	0.0558	165742	0	0	9801	12.62
60000	0.9441	0.0558	165742	0	0	9801	11.59
70000	0.9441	0.0558	165742	0	0	9801	10.62
80000	0.9757	0.0558	165742	0	0	9801	9.31
90000	0.9839	0.0242	162741	8535	2983	1266	9.89
100000	0.8231	0.0160	163934	8769	1790	1032	9.67
110000	0.7761	0.1768	135059	9433	30665	368	8.97
120000	0.8055	0.2238	126741	9487	38983	314	8.23
130000	0.9441	0.1944	132374	9019	33350	782	7.62
140000	0.9441	0.0558	165742	0	0	9801	7.08
150000	0.9441	0.0558	165742	0	0	9801	6.64
160000	0.9441	0.0558	165742	0	0	9801	6.25
170000	0.9441	0.0558	165742	0	0	9801	5.89
180000	0.9411	0.0558	165742	0	0	9801	5.58

Two sets of experiments were conducted on a 5-layer neural network [250 200 100 50 1] with 50 iterations, a 100 batch size, trained with a 100,000 features. For the tested artificial images, the lower threshold was set to the value of the minimum network output plus 0.2, and the upper threshold was set to a value of the maximum network output minus 0.005. For SAR images, the threshold changed depending on the image. The best results seemed to be around 0.9999 for the upper threshold, and 0.15 for the lower threshold.

The first experiment was to test the network performance on artificial images with varying levels of speckle noise. Our conjuncture was the higher the noise level, the poorer the DNN performance. To test our conjuncture, we created artificial images before and after changes using grayscale geometric shapes, added noise, and compared the results of the network to the ground truth. A sample of training data is presented in Fig. 2.

The second experiment was to test the network with different types of noise, shown in Fig. 3. In particular, we tested the network with speckle, Gaussian, salt and pepper, and Poisson noise. Speckle noise is a type of multiplicative, uniformly distributed random noise. Gaussian noise is a type of additive white noise. Salt and pepper noise is a type of additive noise with on and off pixels. Poisson noise is a type of additive noise that mimics electronic noise.

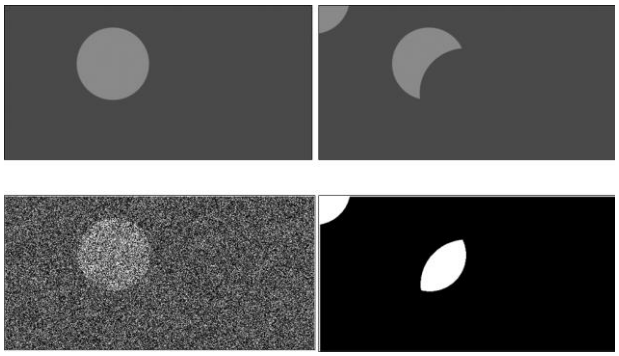

Fig. 2. Top left: image before change. Top Right: Image after change. Bottom left: the before change image with artificial speckle noise of variance 5 added. Bottom right: the ground truth change map.

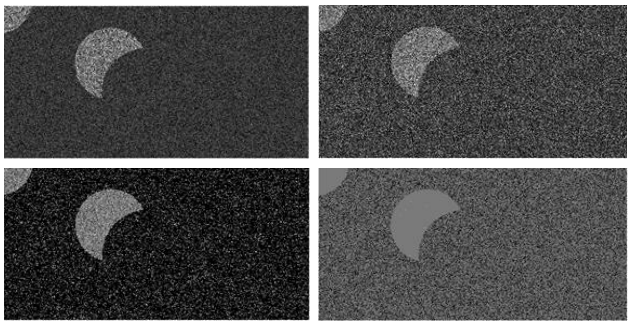

Fig. 3. An artifiial image with different types of noise added to it. Top left has speckle noise, top right has Gaussian noise, bottom left has salt and pepper noise, and bottom right has Poisson noise.

V. Performance Comparison and Analysis

In training the network with different features and labels, the primary issue is that in some cases, the network produces a completely black change detection map. Initially, it was believed that a black change detection map was due to a lack of diversity in the labels. One would expect the greater the training set size, the better a network learns. This is not necessarily true, as we observed for the original 5-layer network trained initially with 10 sets, and then 30 sets of images with little to no progress and more black boxes for change detection maps as the number of training features used increases.

Our results found that in a 5-layer network, there is little to no correlation between the size of the training set and the accuracy in which the network trains. The next correlation to analyze was the diversity of the set versus the PCC. In our test, we observed that for data sets where the percent changed in the training sets were controlled to be 25% and 50%, there were only black boxes and white boxes in the resulting change detection maps. In

training a network, if the percent labels are controlled, it is imperative to adjust the batch size so that the labels are actually being used as intended.

When testing a 3-layer network, the results were a less accurate change detection map but with no black boxes, which implies that features tend to get lost when there were more layers in a network. When the training iterations are lowered, nearly all of the change detection maps were black boxes. Increasing the number of iterations to 100 resulted in higher accuracy, but increasing to 200 resulted in many black boxes. This implies that an optimal iteration number lies between 100 and 200. Overall, the best change detection map came from the original 5-layer network, with a training set of 100,000. Fig. 4 depicts the final test results that include the percentage correct classification, true positives, true negatives, false positives, and false negatives of each image.

	PCC	TP (%)	TN (%)	FP (%)	FN (%)
Puppy (Artificial)	98.21	3.80	94.41	1.78	0.01
Santarem	96.00	1.15	94.86	3.39	0.61
River	97.11	1.10	96.00	2.25	0.64
Santarem 2	87.68	27.79	59.90	6.16	6.16

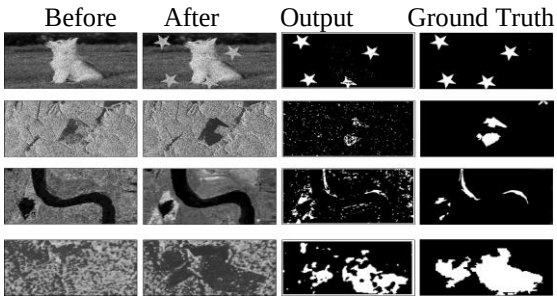

Fig. 4. First row: Puppy image with artificial noise. Second row: Santarem Brazil (June 2007 & May 2008). Third Row: Yellow River Estuary (June 2007 & May 2008). Fourth row: Santarem Brazil (June 2007 & May 2008).

The results of the first experiment with four types of test images were clearly depicted in Fig. 4. The puppy image was the artificial image with artificial speckle noise at a variance of 1.7 from the mean. The Santarem, River, and Santarem 2 images were real-world SAR images available at NASA's Working Group on Validation and Calibration. The before change, after change, output of the DNN, and the ground truth were also plotted. The PCC values were calculated and summarized in the table in Fig. 4. The artificial image without noise shows the best performance of the DNN, PCC at 98.21%, and the network's performance degraded for real-world SAR images.

Next, we showed that the network's performance increased when the amount of noise decreased (see Fig. 5). We added different levels of speckle noise by increasing the variance in nine sets of artificial images and calculated the PCC values. The results confirmed our conjuncture that the DNN performance depended on the level of noise or disturbance contained in the SAR images.

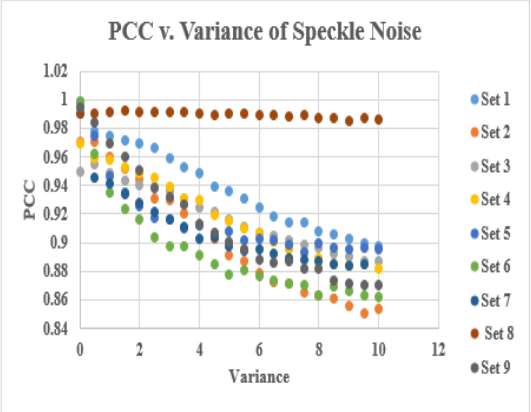

Fig. 5. Accuracy of the network's change detection map after being tested with different levels of speckle variance in the artificial images.

Fig. 6 depicted change detection in an image with different levels of speckle noise power. The top panel image shows the clear image, the middle panel shows the image with mild noise power and the bottom image shows the heavy noise.

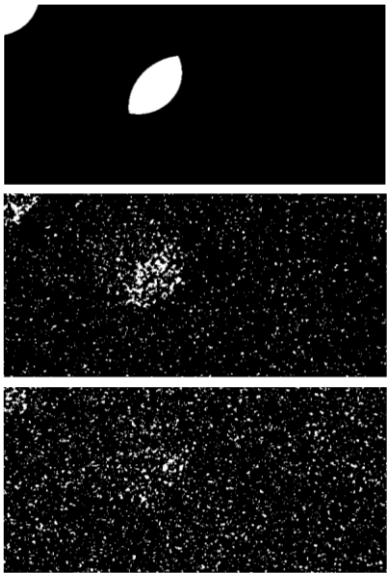

Fig. 6. Top: the test images had a variance of 0 and the PCC of the network's change detection map was 99.5. Middle: the test images had a variance of 5 and the PCC of the network's change detection map was 93.6. Bottom: the test images had a variance of 10 and the PCC of the network's change detection map was 89.9.

We further tested the DNN performance with different types of noise, for example, Gaussian, salt and pepper, and Poisson in addition to speckle noise. A similar trend to Fig. 5 was observed which verified that the performance of the DNN degraded as the power of noise increased for all these types of noises.

VI. Conclusions and future work

We developed and implemented image change detection for SAR images based upon a deep neural network method that involved pre-classifying, feature learning, network training, etc. We tested the algorithms using artificial images and real-world NASA SAR images. For all the noise types tested, when the noise density increased, the network performance degraded. Areas of future work lie in accelerating the training of the neural network. Currently, parallelization affects only iterative image processing. Future work could be in discovering an optimal parallel structure for the training of deep neural networks.

References

[1] H. Larochelle and Y. Bengio, "Classification using discriminative restricted Boltzmann machines," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, Jul. 2008, pp. 536-543.

[2] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, no. 14, no. 8, pp. 1711-1800, 2002.

[3] M. Gong, J. Zhao, J. Liu, Q. Miao, and L. Jiao, "Change detection in synthetic aperture radar images based on deep neural networks," *IEEE Trans. Neural Networks and Learning Systems*. vol. 27, no. 1, pp. 125-138, January 2016.

[4] M. Gong, Z. Zhou, and J. Ma, "Change detection in synthetic aperture radar images based on image fusion and fuzzy clustering," *IEEE Trans. Image Processing*, vol. 21, no. 4, pp. 2141-2151, April 2012.

[5] J. C. Dunn, "A Fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," Journal of Cybernetics, vol. 3, pp. 32-57, September 1973.

[6] J. C. Bezdek, "Pattern recognition with Fuzzy objective function algorithms", Plenum Press, New York, 1981.

[7] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *J. Electronics Imaging*, vol. 13, no. 1, pp. 146-168, 2004.

[8] J. Kittler and J. Illingworth, "Minimum error thresolding," *Pattern Recognition*, vol. 19, no. 1, pp. 41-47, 1986

[9] M. Kerwin, Comparison of Traditional Image Segmentation Techniques and Geostatistical Threshold, School of I.T, Maths & Physics James Cook University Townsville, Australia, 04 July 2006.

[10] C. Siagian and L. Itti, "Biologically-inspired face detection: non-Brute-Force-search approach," IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'04), 1063-6919/04.

[11] L. Najman and M. Schmitt, "Watershed of a continuous function". In Signal Processing (Special issue on Mathematical Morphology), vol. 38, pages 99–112, 1994.

[12] M. Tanaka and M. Okutomi, "A Novel Inference of a Restricted Boltzmann Machine", *International Conference on Pattern Recognition (ICPR2014)*, August, 2014.