12-20-2009

Fast Algorithm for Modeling of Rain Events in Weather Radar Imagery

Anirudh Paduru
University of New Orleans

Fast Algorithm for Modeling of Rain Events in Weather Radar Imagery

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
In partial fulfillment of the
requirements for the degree of

Master of Science
in
Engineering

by

Anirudh Mohan Paduru

B.S. Jawaharlal Nehru Technological University, 2007
M.S. University of New Orleans, 2009

December 2009

Table of Contents

List of Illustrations

Abstract:

Weather radar imagery is important for several remote sensing applications including tracking of storm fronts and radar echo classification. In particular, tracking of precipitation events is useful for both forecasting and classification of rain/non-rain events since non-rain events usually appear to be static compared to rain events.

Recent weather radar imaging-based forecasting approaches [3] consider that precipitation events can be modeled as a combination of localized functions using Radial Basis Function Neural Networks (RBFNNs). Tracking of rain events can be performed by tracking the parameters of these localized functions. The RBFNN-based techniques used in forecasting are not only computationally expensive, but also moderately effective in modeling small size precipitation events.

In this thesis, an existing RBFNN technique [3] was implemented to verify its computational efficiency and forecasting effectiveness. The feasibility of modeling precipitation events using RBFNN effectively was evaluated, and several modifications to the existing technique have been proposed.

Keywords:
Radial Basis Function Neural Network, Forecasting, Nowcasting, Tracking, Rain Events, Non-Rain Events, Gaussian Functions, Mahalanobis Distance, Pyramidal Synthesis, Cascade Synthesis, Coarse Resolution, 3D Elevations

Chapter 1: Introduction

Employment of radars in the field of meteorology can be dated back to early 1940's (World War II) [2]. As a result of advancements in fields such as remote sensing and image processing, it became apparent that conventional radars were particularly useful for tracking and detection of cyclones and other weather related applications. Owing to the introduction of fast and highly efficient computers in 1970's, the processing of large amount of radar data for the purpose of quality control and forecasting became possible.

Currently, dedicated weather radar are designed to detect precipitation (rain events) in the atmosphere. Several countries adopted weather radar networks in order to be able to predict flooding, to provide flood warnings, and to perform water management. Remote sensing using radars is capable of providing adequate coverage and to collect comparatively large amounts of data. Weather radars are capable of covering 200-250 miles from their position. In this work, data from the WSR-88D radar located in Melbourne, Florida has been used. It is important to note that there are numerous challenges involved in using weather radar data. While weather radars are designed to detect precipitation in the atmosphere, there are often echoes resulted from unwanted sources such as birds, earth surface formations, aircrafts and other man-made structures. Moreover, ground clutter also poses a significant problem in radar image data. Most often, all non-rain related echoes should be eliminated prior to data processing.

Tracking of storm fronts presents several challenges. For example, there is a necessity of great computational power, since huge volumes of data need to be processed. Each radar image contains several events, some of which being rain and some of which being non- rain events. Following echo classification and elimination of non-rain events, the remaining precipitation events need to be processed. It can be considered that these rain events are evolving in time in terms of position, size, shape and, intensity. Moreover, new echoes may emerge in the radar images as newly formed precipitation events or events that enter the radar range, while other events may be merged or even fade away from the image, all within a short duration of time. Hence, association of precipitation of rain events in consecutive images is not a trivial problem.

On the other hand, tracking of storm fronts can be successful to certain extent. With help of tracking, quite a few key weather-related applications can be addressed [1], including forecasting. Although forecasting techniques cannot perfectly predict future rain maps, reasonable estimation of certain precipitation characteristics, including the location of rain events, is achievable. In addition, nowcasting [6], i.e., the prediction of the rain events, few or several minutes ahead, can be relatively successful. Based on precious work [3], applications of nowcasting include point-to-point communication in poor weather conditions. For example, accurate prediction of the motion of heavy storms can assist in determining the path attenuation due to precipitation. [1]

As mentioned earlier, separation of rain and non-rain events, such as clutter, is an important application of weather radar imagery. Tracking may be one of the important components used in this application. Separation of precipitation and non-precipitation events (clutter) may in general be considered to be a simpler process compared to forecasting of events. For separation, the paths of the events only need to be tracked and analyzed rather than tracked, analyzed, and predicted. Nevertheless, the association of events from one image to next is non-trivial. However, assuming that there is a solution to the association problem, the paths can assist in determining if the tracked events are static or moving. Non-rain events, such as clutter, are usually static compared to rain events. Techniques used in past to separate rain from clutter [9] do not always take advantage of the temporal characteristics of precipitation events.

There are several ways of tracking rain fields, one of them being tracking of localized functions generated based on modeling of rain fields. Modeling of rain fields is one of the significant components of the overall storm tracking process. Different techniques have been presented in the past for tracking precipitation events [2], [3]. In general, precipitation events can be considered as an assortment of different deformable objects. Out of the different possible ways to track these objects, one technique may attempt to match the outlines between temporally successive images. Alternatively, these objects can be approximated by means of 2D localized Gaussian function. Consequently, every object in the image can be approximated by one or more Gaussian functions. Furthermore, tracking of these localized Gaussian parameters will lead to tracking of the actual events. It is essential to note that localized functions, characterized by a set of parameters, may be used to model deformable events. The measure of deformation can be represented by alteration in these parameters. In the recent past, RBFNN [7], [8] have been used 2D as Gaussian envelopes in order to model precipitation events by determining the envelope's parameters [2], [3]. It was concluded that the usage of RBFNN is computationally expensive, while they appear to be only moderately effective in modeling small precipitation events in weather radar imagery [1]. In order to emphasize the importance of computational efficiency in nowcasting applications, it should be mentioned that if, for instance, a nowcasting technique requires 5 or more minutes to forecast the precipitation paths 5 minutes in the future, employment of this nowcasting technique is meaningless.

In this thesis, we implemented an existing technique, with the all specified conditions, and verify the computational efficiency and modeling accuracy of the algorithm. Changes to certain parameters are also applied to the existing algorithm [3], to acquire appropriate results. Thereafter, modifications are made to improve computational efficiency. Moreover, we evaluate the feasibility of modeling rain events using RBFNN-based approach in an efficient manner.

The proposed modifications primarily deal with two aspects of the RBFNN-based modeling. Firstly, some points on the radar imagery are pre-selected using a simple algorithm, rather than a cold start selection of points, as mentioned in the previous work [3]. These points are used as initial points for training the RBFNN on the radar imagery. Secondly, tracking and prediction are performed only at the coarse image resolution without extrapolating to higher resolution at anytime. In other words, training of RBFNN is performed completely at the coarse image resolution. Nevertheless, it will be shown that full scale resolution images can be obtained directly from the coarse resolution images. It was observed that this modification increased the computational efficiency significantly. In the previous work [3], the final training and prediction are done at full resolution, which results in a computationally expensive algorithm. Furthermore, in order to model precipitation events efficiently at a coarse image resolution, the model parameter initialization should be done considering the existence of small scale events. In particular, small scale events, which are undesirably condensed in size at coarse resolution images, need to be properly handled for precise training and prediction.

The rest of the thesis is organized as follows. Chapter 2 presents details about RBFNNs and previous RBFNN-based forecasting algorithms. Chapter 3 introduces the proposed algorithm and presents a discussion regarding the modifications performed to the existing algorithm. Chapter 4 shows experimental results on actual radar data sequences in order to evaluate the proposed algorithm and compare it with the existing technique. Finally, chapter 5 concludes the thesis, highlighting a potential future evolution of the work.

Chapter 2: Background and Literature Review

2.1 Introduction to the Radial Basis Function Neural Network

As mentioned in the previous section, in the past, RBFNNs have been used to model precipitation events. This section mainly focuses on different types of RBFNNs used for modeling rather than classification. However, it is worth mentioning at this point that RBFNNs are also well-suited for non-linear time-series prediction [10]-[15], which can also be part of a forecasting application. More specifically, RBFNNs have an acceptable prediction performance for stationary time-series, but a poor performance for non-stationary signals. Since real world signals are often highly non-linear and non-stationary, modified versions of RBFNNs may need to be employed [15].

In general, the RBFNN is single hidden layer neural network. The operation of each of the hidden layer nodes is defined by a function characterized by a set of parameters. The input vectors presented to the network are modified by the hidden layer nodes, and the hidden layer node outputs are combined, via a set of weights, in order to produce the network outputs or responses. The response produced by the RBFNN should ideally be equal to a desired response. The difference (error) between the produced and desire responses is used to adopt the network weights and hidden layer node parameters. Next, some more detailed background information regarding the basic RBFNN-based technique for modeling precipitation events is presented. Details regarding the usage of RBFNNs in previous work [3] for the purpose of forecasting are presented in section 2.2.

In general, RBFNNs are used for data classification and function approximation. As mentioned in the previous section, the goal is to approximate the rain events using localized functions, namely 2D Gaussian envelopes. The basic structure of RBFNNs is depicted in Figure 1:

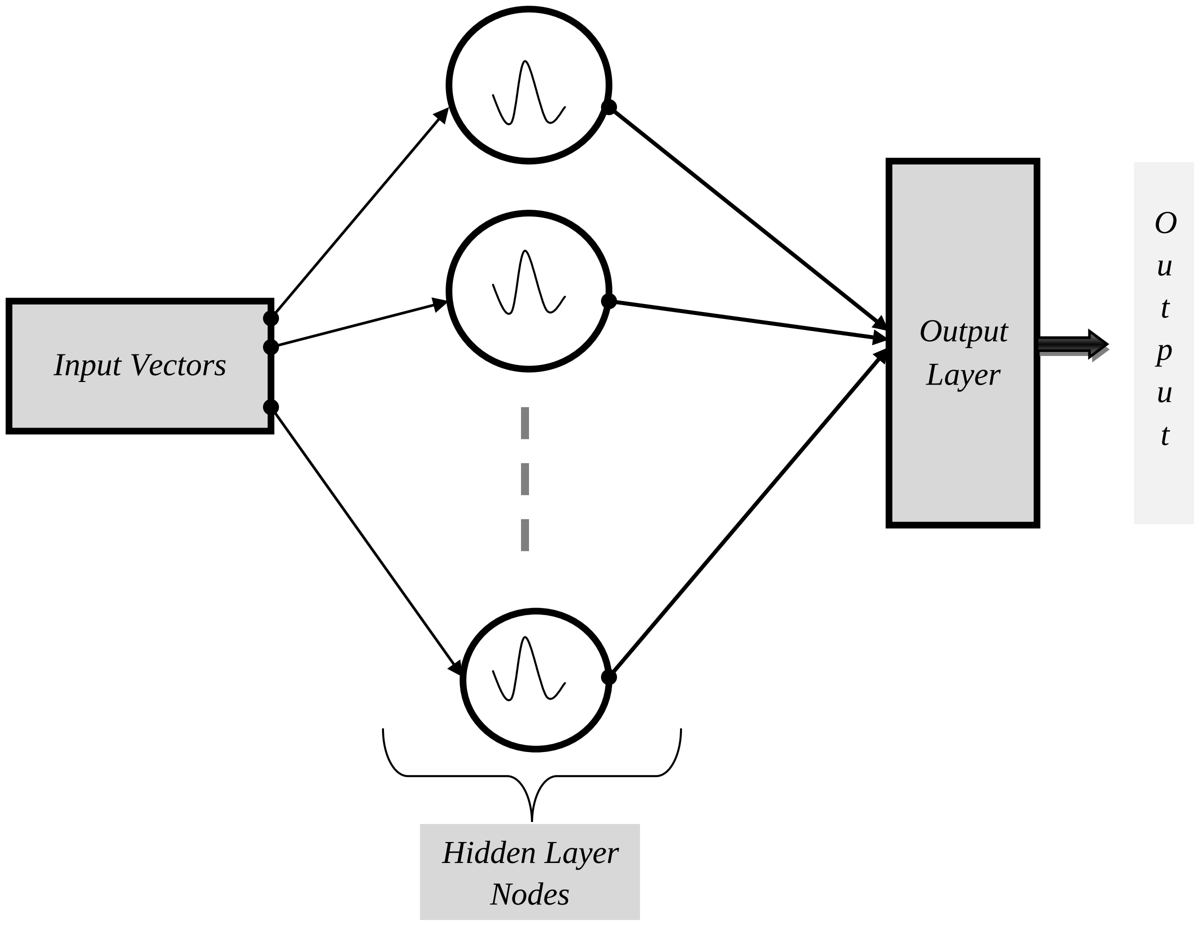


Figure 1: General Structure of Radial Basis Function Neural Network

An RBFNN consists of 2 layers, a hidden layer and an output layer as shown in Figure 1. The input data to each hidden layer node is presented in the form of input vectors. Moreover, each hidden layer node operates as a non-linear function and one response per node is obtained. Outputs from each hidden layer node are multiplied with certain weights and then fed as inputs to the output layer. All the inputs to the output layer are combined accordingly to produce the overall desired output which can be either a scalar or vector. The RBFNN is trained iteratively, attempting to reduce the difference in the desired output and the actual output, thus reducing the output error. For this simple RBFNN, the network parameters involved are hidden-to-output layer weights, and different parameters according to the functions used in each hidden layer node.

As mentioned earlier, RBFNN can be used as function approximator. It has been proposed in the past [2], [3] that RBFNN can be used to model precipitation events as combination of localized functions. Thereafter, the function parameters can be tracked to identify the prediction paths of precipitation events. In the case of rain-map modeling, the *i*th input vector $\overrightarrow{x_i}$ consists of *i*th pixel coordinates, i.e. $\overrightarrow{x_i} = (x_i, y_i)$, $i = 1,2,\dots,N$, where N represents the total number of pixels. All N vectors are used to train the network. The function that represents the output of the *j*th hidden layer node when the *i*th input, $\overrightarrow{x_i}$, is presented to the network is defined as follows :

$$g_j(x_i, y_i) = exp\big(-d_j(x_i, y_i)/2\big) \tag{1}$$

where $d_j(x_i, y_i)$ is the squared distance between the *i*th input and the prototype associated to the *j*th node. The *j*th prototype in this case is defined as the vector representing the center point, $\overrightarrow{p_j} = (p_j^{(x)}, p_j^{(y)})$, of the Gaussian envelope $g_j(x_i, y_i)$. If $d_j(x_i, y_i)$ is considered as the square of the Euclidean distance between the input vector $\overrightarrow{x_i} = (x_i, y_i)$ and the prototype $\overrightarrow{p_j} = (p_j^{(x)}, p_j^{(y)})$, then the function in equation (1) represents an isotropic Gaussian function. This Gaussian function has a standard deviation of unity, and is centered at $\overrightarrow{p_j}$. The squared Euclidean distance $d_j(x_i, y_i)$ can be represented as $d_j(x_i, y_i) = (x_i - p_j^{(x)})^2 + (y_i - p_j^{(y)})^2$. In order to make the RBFNN model more flexible (while simultaneously more complex) for modeling the rain events, the widths of the Gaussian function can be made independent along each direction. This is achieved by replacing the Euclidean with the Mahalanobis distance. By using Mahalanobis distance instead of Euclidean distance in equation (1), the isotropic Gaussians are replaced by directional Gaussians. The square of the Mahalanobis distance is defined as :

$$d_j(x_i, y_i) = (\overrightarrow{x_i} - \overrightarrow{p_j})^T \, K \, (\overrightarrow{x_i} - \overrightarrow{p_j}) \qquad (2)$$

where *K* represents the inverse covariance matrix. As mentioned earlier, the overall output, $\widehat{f}_i(x_i, y_i)$, for the *i*th input should ideally be equal to the actual value of the pixel, $f(x_i, y_i)$, at location (x_i, y_i). The overall output of the system is obtained as a linear combination of each hidden layer node output multiplied by its corresponding weight, w_j. Thus, the overall output $\widehat{f}_i(x_i, y_i)$ for the *i*th input is defined as

$$\widehat{f}_j(x_i, y_i) = \sum_j^{N_j} w_j \; g_j(x_i, y_i) + \theta_j, \qquad (3)$$

where N_j is the total number of prototypes and θ_j is a bias term associated to the *j*th hidden node. Hence, from the above discussion, the training parameters required for precipitation modeling are the weights w_j, the inverse covariance matrix *K*, the prototypes $\overrightarrow{p_j} = (p_j^{(x)}, p_j^{(y)})$ and the bias term θ_j. The training of parameters is performed in an iterative manner. The difference between the obtained output $\widehat{f}_i(x_i, y_i)$ and the desired output $f(x_i, y_i)$ is fed back into the network in order to adopt the network parameters and eventually reduce the error.

As discussed earlier, the RBFNN approximates the precipitation events as mixture of Gaussians. This technique is discussed in detail in section 2.2. In general, network parameters are initialized prior to training. Training is performed in a constructive manner, in the sense that hidden nodes are added and deleted in order to best approximate the rain fields. The initialization criteria, the methodology for training the parameters, and the addition/deletion criteria depend on the objective to be achieved.

It should be mentioned at this point that, in general, the pixel values in weather radar imagery represent reflectivity values, Z. Reflectivity values can be converted to precipitation (rain rate), R, by using a relationship commonly known as Z-R. For example one such relationship is $Z = aR^b$, where a and b are constants. A popular Z-R relationship is the Marshal-Palmer law, which is defined as $Z = 200\ R^{1.6}$ [19].

2.2 Previous work

This section mainly highlights the algorithm and methodology of previous work regarding linear forecasting using RBFNNs. More specifically, parameter adaptation, pyramidal synthesis, cascade synthesis, and finally linear forecasting are discussed.

As a reminder from section 2.1, the i^{th} input vector presented to the network is defined as $\overrightarrow{x_i}$, and consists of the i^{th} pixel coordinates, i.e. $\overrightarrow{x_i} = (x_i, y_i)$, $i = 1,2,\ldots,N$. The output produced by the RBFNN is the rain rate at that point. Accordingly, the algorithm will provide the rain rate at every pixel of the radar image, for every input vector $\overrightarrow{x_i}$ by using 2D mixture of Gaussian functions. Eventually, the difference of the obtained output and desired output is applied to the network, in order to minimize the error. For this purpose, the Mean Square Error (MSE) can be used. Based on the RBFNN output, namely $\hat{f_j}(x_i, y_i) = \sum_j^N w_j\ \ g_j(x_i, y_i) + \ \theta_j$, defined in equation (3), the mean square error is defined as follows [3]:

$$\epsilon = \frac{1}{N_x N_y} \sum_{x,y} (f(x_i, y_i) - \hat{f}(x_i, y_i))^2 \quad (4)$$

It should be mentioned at this point that the MSE definition presented in previous work [3] is not accurate. The network mainly aims to minimize the MSE, by using a competitive algorithm proposed in [2]. Some changes to the original technique in [2] were incorporated in [3]. The technique presented in [3] proposed a modification to the output function of the RBFNN presented in equation (3). More specifically, the output in [3] is defined as:

$$\hat{f}_j(x_i, y_i) = max_j(w_j \; g_j(x_i, y_i) \; + \; \theta_j) \quad (5)$$

In other words, only the output from the hidden node which gives the maximum response for that particular point in the radar image is activated and trained. The selected node is the one that gives the best match between the obtained and desired output. It is important to mention that the parameter, θ_j , was determined to be unnecessary and hence was removed from the training process [3]. By suppressing this parameter, the computational efficiency of the algorithm was improved although its effect on the output was negligible. Hence the output was finally defined to be as follows:

$$\hat{f}_j(x_i, y_i) = max_j(w_j \; g_j(x_i, y_i)) \quad (6)$$

From the above discussion, the RBFNN can be represented as shown in Figure 2.

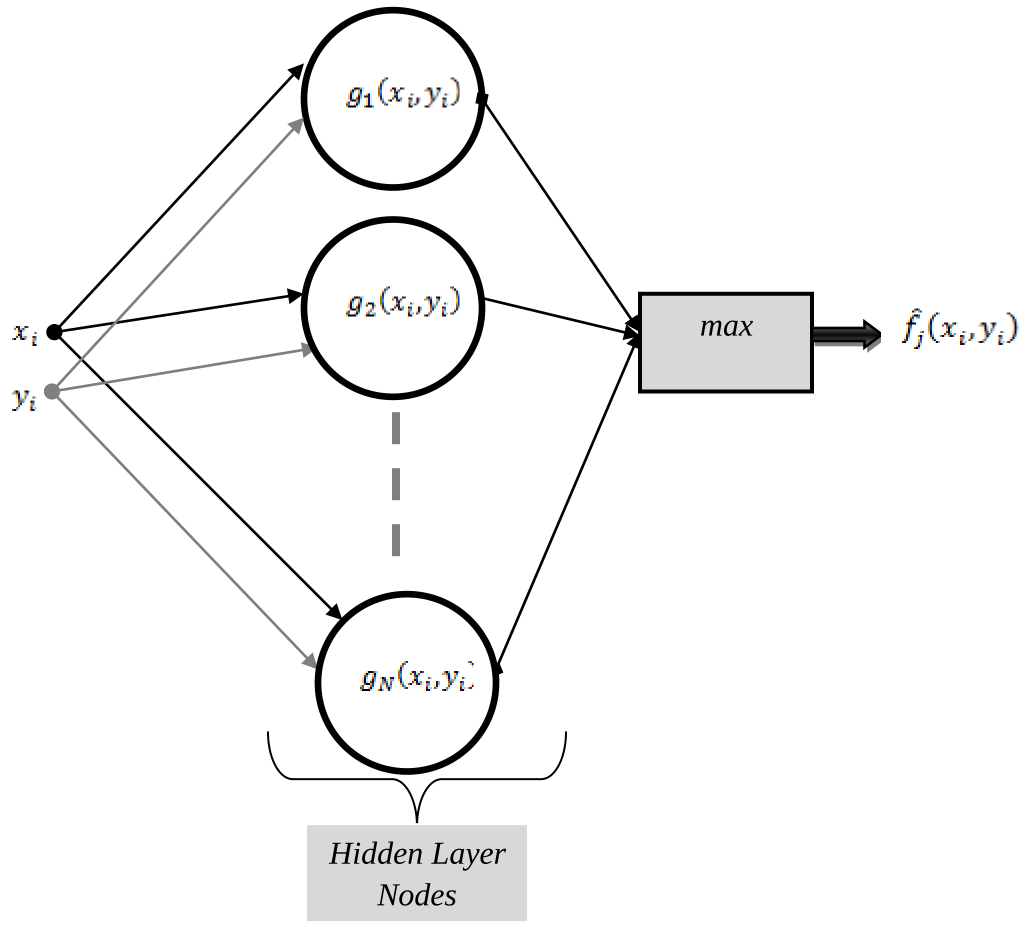


Figure 2: The RBFNN used in previous work [3]. The inputs are the pixel coordinates and the output layer is the max function.

The competitive algorithm used to train the parameters is adopted from [2]. It will be noteworthy to provide some discussion about the Mahalanobis distance before proceeding to the actual parameter learning/updating stage. Mahalanobis distance is defined as mentioned in equation (2), where K represents the inverse covariance matrix. The $(m,n)^{th}$ element of the inverse covariance matrix, K, can be expressed using the marginal standard deviations σ^m and σ^n and the correlation coefficient $h^{m,n}$ as follows [2]:

$$k^{m,n} = \frac{h^{m,n}}{\sigma^m \sigma^n} \tag{7}$$

Where $\sigma^m > 0$ and $\sigma^n > 0$, $h^{m,n} = 1$ if *m*=*n*, else $|h^{m,n}| \leq 1$. Also, $h^{m,n} = h^{n,m}$ for all *m* and *n*. Hence, as an update to what was discussed earlier, the learning parameters are the weights w_j, the prototype centers $\overrightarrow{p_j} = (p_j^{(x)}, p_j^{(y)})$, the standard deviations σ^m and σ^n ,and the correlation coefficient $h^{m,n}$. A quadratic error function is used to train the network in supervised mode, by applying a gradient-decent method [2]. The quadratic error function is defined as $E_j = \frac{1}{2}(f_j(x_i, y_i) - \hat{f}_j(x_i, y_i))^2$. All the parameters are updated using a set of equations presented in [2]. These equations are presented here as well for completeness :

$$\Delta w_j = -\eta \frac{\partial E_j}{\partial w_j} = \eta \ \ (f_j(x_i, y_i) - \hat{f}_j(x_i, y_i)) \ \ g_j(x_i, y_i) \tag{8}$$

$$\Delta p_j^m = -\eta \frac{\partial E_j}{\partial p_j^m} = (\Delta w_j) \, w_j \ \sum_l k_j^{m,l} (x_i^l - p_j^l) \tag{9}$$

$$\Delta \sigma_j^m = -\eta \frac{\partial E_j}{\partial \sigma_j^m} = (\Delta w_j) \ w_j \ \ \sum_l k_j^{m,l} \frac{(x_i^m - p_j^m)(x_i^l - p_j^l)}{\sigma_j^m} \tag{10}$$

$$\Delta h_j^{m,n} = -\eta \frac{\partial E_j}{\partial h_j^{m,n}} = -(\Delta w_j) \ w_j \ \ \frac{(x_i^m - p_j^m)(x_i^n - p_j^n)}{\sigma_j^m \sigma_j^n} \tag{11}$$

where η is the learning step constant. The values for η are defined at a later stage.

As mentioned earlier, for each pixel of the radar image (input), only the hidden node giving the best match between the desired output and the obtained output is activated and trained using the above equations (8-11). If none of the hidden layer nodes give a good match, i.e., if the difference between the obtained and desired output is greater than a minimum threshold, *maxerr* and the minimum Euclidian distance of the point to any nearest node is larger than a threshold distance, *mindist,* a new node is created at that point [3]. Hidden layer nodes which are not activated by minimum number of input vectors for a given number of iterations are deleted. The minimum number of activations required is defined as a parameter *escape.* The deletion condition occurs only after examining all possible input vectors from the radar image. This process is repeated until the MSE, ε, drops below a fixed value, *maxmse,* or until a number of iterations, *maxepoch,* is reached [3].

Once any one of the above conditions is achieved, i.e., $\varepsilon \leq maxmse$ or *maxepoch* is reached, the output image will contain the modeled rain events. These events are characterized/modeled using the localized parameters derived from the input vectors obtained from the original image. By tracking the localized parameters in the modeled image, the rain event paths can be predicted.It is important to emphasize that the values of the above mentioned constants: η, *maxmse, maxepoch, maxerr, mindist*, etc., have been adopted from [2] and [3].

In general, the size of the original radar image is large, thus the training process may be considerably time consuming. In order to improve computational efficiency and optimize the modeling parameters, a pyramidal synthesis technique is used. In pyramidal synthesis, the modeling of events is done at the coarse resolution image. The number of input vectors and hidden layer nodes are considerably reduced using coarse image resolution, which reduces the computational time while achieving the same MSE. Initially, the RBFNN approximation is applied on the coarse image resolution. In order to obtain a coarse resolution image the original image is down-sampled using the following equation:

$$f_k(i,j) = \frac{1}{2^{2k}} \sum_{m.n \epsilon \Omega} f(m,n) \tag{12}$$

where Ω is a 2^k x 2^k squared region around pixel (*i*,*j*) [3] .

The parameters obtained at lower resolutions are extrapolated to higher resolutions in a step-wise manner. Hence, whenever a new training is started on an intermediate resolution image, the parameters from the previously modeled, lower image resolution are considered as a starting point. Thus, as the image gets more detailed at higher resolutions, some more nodes may be added and sometimes deleted.

It is an important observation that higher image resolution just adds small, finer details and allows the network to adjust itself to higher spatial frequency components of the original image. Thus, initially, pyramidal synthesis is applied to the first frame of the sequence, until the highest desired resolution is reached. Cascade synthesis is applied for all subsequent frames after the first frame.

The adaptation of network is performed by training on a sequence of frames which are similar to their next frame. A cold start synthesis on the next frame is computationally expensive. However, since the successive frames are similar to the preceding frame, parameters from the preceding frame can be used as a starting point to train the next frame. Using cascade synthesis, the number of iterations in training a new frame is reduced considerably, thus leading to less computational time. Also the advantage of cascade synthesis is that the same nodes are trained on the subsequent rain maps, and correspondence between nodes is retained [3]. In summary,, training is performed initially by pyramidal synthesis on the first frame, and thereafter by cascade synthesis on the successive frames. Different forecasting techniques have been presented in previous work [3]. In this thesis, a simple linear forecasting approach has been used.

The linear forecasting method is one of the simplest forecasting methods for predicting the parameter values. In this method, the future values are predicted using a simple linear extrapolation of parameter values based on latest two values. For instance, if p_k is the value of the parameter at frame k, and p_{k-1} is the value of the same parameter at the immediately preceding frame, the forecasted value f_{k+n} at frame $k+n$ is given by :

$$f_{k+n} = p_k + n\,.(p_k - p_{k-1}) \tag{13}$$

An alternative to the linear forecasting method is the steady state method. In general, in the steady state method, the rain events are assumed to undergo change in position but not change in size, orientation or intensity. However, RBFNNs are capable of modeling precipitation as a combination of scaled and oriented Gaussian envelopes. Therefore, the available forecasted parameter values include orientation and scaling of precipitation events.

Chapter 3: Proposed work – Fast Algorithm

In the previous work [3], even after using pyramidal synthesis and cascade synthesis, the authors [3] identified that the training process is still relatively time consuming. In general, the radar image contains precipitation events and blank events (no events). The precipitation events are represented by non-zero valued pixels. The regions without the precipitation events are represented by zero-value pixels. In the previous work [3], all pixels are used for training purposes. If the zero-valued pixels are not considered in the training process, arbitrary values are assumed in place of zero values by the Gaussian envelope which approximated that particular event. A Gaussian envelope tries to approximate an event to the best possible approximation. However, if there are no restrictions for the size or width of the Gaussian envelope, it may assume some arbitrary values beyond the event boundaries. On the other hand, if there are certain restrictions for the width of the Gaussian envelope, it may approximate the event without crossing the event boundaries significantly. For example, some zero-valued pixels can be used around the event's boundaries. This is illustrated in the following picture.

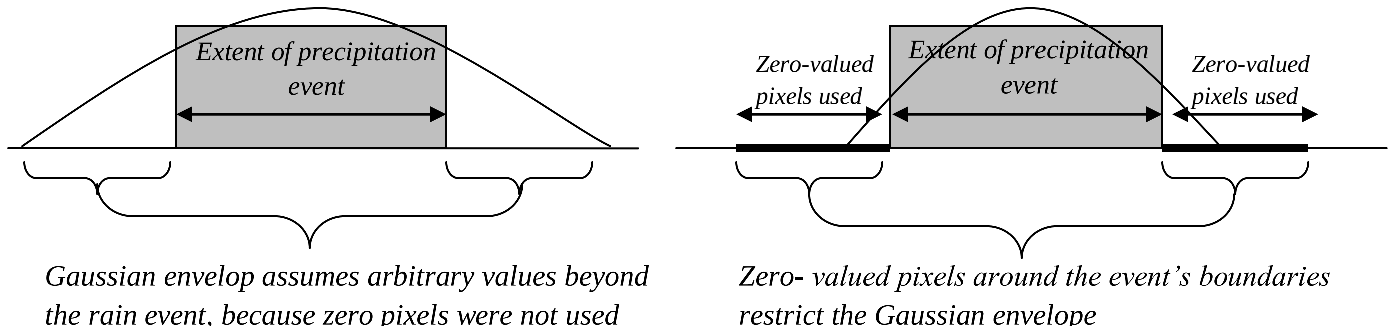

Extent of precipitation event
Gaussian envelop assumes arbitrary values beyond the rain event, because zero pixels were not used
Zero-valued pixels used
Extent of precipitation event
Zero-valued pixels used
Zero- valued pixels around the event's boundaries restrict the Gaussian envelope

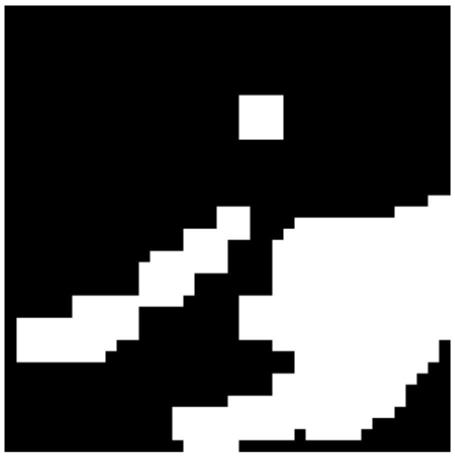
Figure 3: (a) Figure depicting the elongation of the Gaussian envelope beyond the rain event. Thus, assuming arbitrary values in place of zero-valued pixels. (b) The this line indicates the zone of zero values pixels around the boundaries of the rain event. Gaussian envelope is restricted by the zero-valued pixels. Thus very few or no arbitrary values.

If only non-zero valued pixels are, during the training process a divergence of the training parameters has been observed. This is discussed in detail in section 3.1.

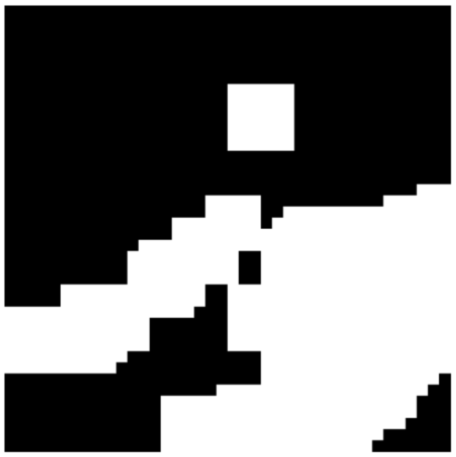
As mentioned in the previous section, different forecasting techniques can be used to track the network parameters. Some of these methods were also discussed in the previous work [3]. More specifically, for the sake of simplicity, a linear prediction technique is used in this thesis. In the previous works [2], [3], focus was laid on modeling of the events and predicting them. However, there has been some problems with the previous [3] modeling techniques. This thesis mainly focuses on efficient modeling of the rain fields. As described in equation (13), the future parameters are predicted using the parameters from the current frame and the immediately previous frame. In the following sub-sections, we propose two modifications of the algorithm proposed in [3], to overcome the above mentioned problems.

3.1 Selection of Pixels for the training of RBFNN

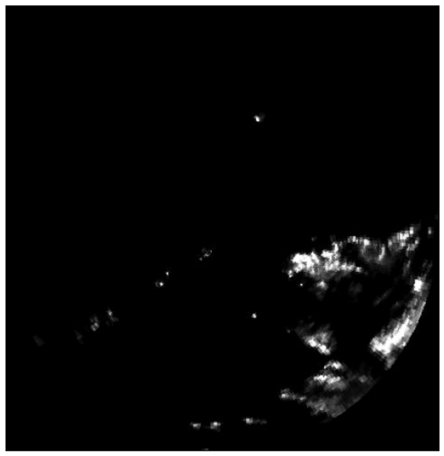
As proposed earlier [3], it may be computationally advantageous to use only non-zero valued pixels for training purposes. However, to solve the problem regarding the divergence of parameters during the training process, as discussed in section 3, a modification in the algorithm is introduced in thesis. All non-zero valued pixels, as well as zones of zero-valued pixels around precipitation events, are used for the training purpose. The following figures illustrate this modification in detail:



(a)



(b)



(c)

Figure 4: (a) The downsampled image of figure 4(c). White regions are the non zero pixels. (b) Image after smoothing the downsampled image in figure 4(a). In this image, white regions are non-zero pixels and zone of zero pixels around them. (c) Original image

By applying a moving average filter on the downsampled image, selecting the non-zero valued pixels in the smoothed image, and excluding the non-zero valued pixels in the non-smoothed image, a zone of zero valued pixels is obtained. More specifically, when a moving average is applied on the precipitation events (non-zero pixels), it blurs the events beyond their borders. Using this technique, all zero-valued pixels required for the training purpose are selected. This ensures that only strategically selected zero-value pixels as well as all the non-zero values pixels are selected for training. This way of pixel selection keeps the number of input vectors presented to the network low. As mentioned earlier, the zone of zero value pixels restricts the Gaussian envelope represented by each hidden layer node mostly within the zone limits (as illustrated in figure 3). Hence, the parameter divergence problem is overcome during the training process.

The localized parameters that are used during the training process namely weight, inverse covariance matrix and its elements, and prototype vectors are adapted as follows as the model evolves from $(it-1)^{th}$ to $(it)^{th}$ iteration:

$$q_{it} = q_{it-1} + m_x \alpha \Delta q \tag{14}$$

where q represents any of the parameters, α is a positive learning parameter, and Δq represents the amount by which the parameter is modified during the training from $(it\text{-}1)^{th}$ to it^{th} iteration. It is important to mention that, in this thesis, the new parameter, m_x, is proposed to represent the multiplicity of the particular input vector. Since only a zone of zero value pixels are used in order to ensure that the Gaussian envelope do not overflow beyond the zone limits, the modeling may be biased in favor of the precipitation pixels (non-zero valued pixels). To avoid such situations, different values of m_x are chosen for zero-valued pixels and non-zero value pixels in the training process. More specifically, to emphasize the zero-value pixels, large values of m_x is chosen. As mentioned earlier, choosing the multiplicity of the input vector m_x, greater than 1 is equivalent to assuming that the same pixel is presented m_x times to the network. More specifically, in this thesis, $m_x = 1$ is used for non-zero pixels and $m_x = 10$ for zero-value pixels.

3.2 Forecasting of image using coarse image resolution

The previous sub-section discussed the approached used regarding the selection of pixels prior to the training process and the conditions for training the parameters. In this sub-section, a method used for increasing the computational efficiency by using coarse image resolution for forecasting is discussed. In general, the characteristic of several events, especially large events, are retained when the image is downsampled to coarse resolution. In this thesis, the network is trained using past and present coarse resolution images in order to predict the future full resolution images. This is in contract to the techniques used in [3], which predicts the future full resolution image from past and present full resolution image. By introduction of this modification, a significant increase in the computational efficiency is achieved. The coarse resolution image is obtained in two steps. First, a low pass filter is applied on the original image for smoothing purposes. Second, the smoothed image is downsampled to the required level. However, it has been observed that most of the small scale events may disappear in the low-resolution image. Therefore, they are more complicated to track and even more difficult to predict than large scale events. More specifically, it has been noticed that some of the isolated small-scale events are not easily distinguishable at coarse resolution.

In coarse resolution image, the peaks (local maximum pixels) within the large scale events may correspond to the points where the Gaussian envelopes (function) one supposed to be centered. In order to determine the peaks within the events, a 3x3 window is used. The pixel with coordinates, $\vec{x}^{lm}$, is a peak if it has the largest value within the 3x3 window centered at $\vec{x}^{lm}$. The initial positions of the prototypes for the training purpose are set at $\vec{x}^{lm}$. Next, the procedure for the proposed work in this thesis is discussed.

As mentioned earlier, this thesis work mainly focuses on precipitation modeling. Assume, for simplicity that, at this point, the parameters for the next image frames have already been estimated. Assume that inverse covariance matrices *K* for the current and future frame, K^t and K^{t+1}, respectively, are also available. The inverse covariance matrix is used in the Mahalanobis distance in equation (1). Similarly, assume the other training parameters i.e. the prototype locations for the current and future frame, $\vec{p}^t$and $\vec{p}^{t+1}$, respectively, are also available. The standard deviations σ_m and σ_n, and the correlation coefficient $h_{m,n}$ are directly associated with *K*, therefore they are not included in this discussion. In order to simplify the discussion, although several prototypes, and thus, covariance matrices are needed to represent the precipitation events included in the rain map, only a single prototype is considered here. Moreover, it is assumed these inverse covariance matrices and prototypes correspond to the exact same event that has evolved from one time instance to the next. The two main characteristics of the Gaussian envelope are the center of Gaussian, and the rotation and scaling parameters. More specifically, the prototypes define the centers and thus are used for determining the translation parameters and, the inverse covariance matrix or the squared inverse covariance matrix defines the rotational and scaling parameters. The following equation defines the association between the future pixel location $\vec{x}_i^{t+1} = (x_i^{t+1}, y_i^{t+1})$ in the image at time instance *t+1* and the corresponding current pixel location $\vec{x}_i^t = (x_i^t, y_i^t)$ at time *t* in the current image:

$$\vec{x}_i^t = (K_1^t)^{-1} K_1^{t+1} (\vec{x}_i^{t+1} - \vec{p}^{t+1}) + \vec{p}^t \qquad (15)$$

Where K_1 is the square root of inverse covariance matrix i.e. $K_1 . K_1 = K$.

Using the above equation (15), the association between the pixel position in future coarse image and current coarse image can be obtained. Although the above equation was evaluated for a single prototype and a single pixel, it can be extended to all prototypes and all pixels in the coarse resolution image. It is noteworthy to mention that the correspondence between the future and coarse resolution images can be transferred to the full resolution images. In other words, the future full resolution image can be obtained by the same equation. To obtain this, the prototype location should be multiplied with the downsampling factor. As mentioned earlier, the above equation is used to obtain the forecasted image once the future parameters have been predicted.

With introduction of the two modifications discussed in sections 4.1 and 4.2, and with the proposed procedure, the modeling of the events is performed with high computational efficiency compared to the work presented in [3]. The illustrations and comparisons between the proposed work and previous work [3] are discussed in detail in the next section.

Chapter 4: Experimental Results

By using the modification proposed in chapter 3, modeling of rain events is performed on a sequence of frames. Also, modeling of rain events is performed using the previous method [3] for the same sequence of frames in order to compare the two algorithms. The central scheme of the algorithm and the values of the parameters are chosen from [2], [3]. After performing the modeling using the appropriate values and conditions for the previous method [3], the results obtained were not satisfactory based on what was expected from the results presented in [3]. Hence, some modifications to the values of the parameters had to be introduced. These parameters are mentioned in the following table. Also, all pixels in the image are used for training purpose and the initial prototype centers are defined randomly at non-zero valued pixels. Two different learning rates were set for prototype centers and all other parameters. They are defined as n_o and n_c respectively.

Resolution	**e**scape	mindist (pixels)	mindist used (pixels)	**m**axsse	maxsee used	n_c	n_o	n_c used	n_o used
40 x 40	3	16	16	0.02	1	0.05	0.01	0.0005	0.0001
80 x 80	5	30	30	0.02	2.5	0.05	0.01	0.0005	0.0001
160 x 160	8	50	50	0.02	3.5	0.05	0.01	0.0005	0.0001

Table 1: Values of escape, mindist, maxsee and learning constants (n_o and n_c) used in previous algorithm [3]. Modified values of the mindist, massee and learning constants (n_o and n_c).

Figures 5 (a), 5(b), and 5(c) show the original images (160 x 160) at time instance 1, 2 and 3, respectively. Figures 5(d)-5(f) are the approximation images for pyramidal synthesis at stages 1, 2, and 3, for time instance 1. Figure 5(g) is the approximated cascade image at time instance 2 at resolution 160 x 160. Figure 5(h) is the predicted image (160 x 160) at time instance 3 using the images from figure 5(f) and 5(h). Figures 5(d) and 5(e) are obtained by downsampling the image in Figure 5(a) by a factor of 4 and 2 respectively.

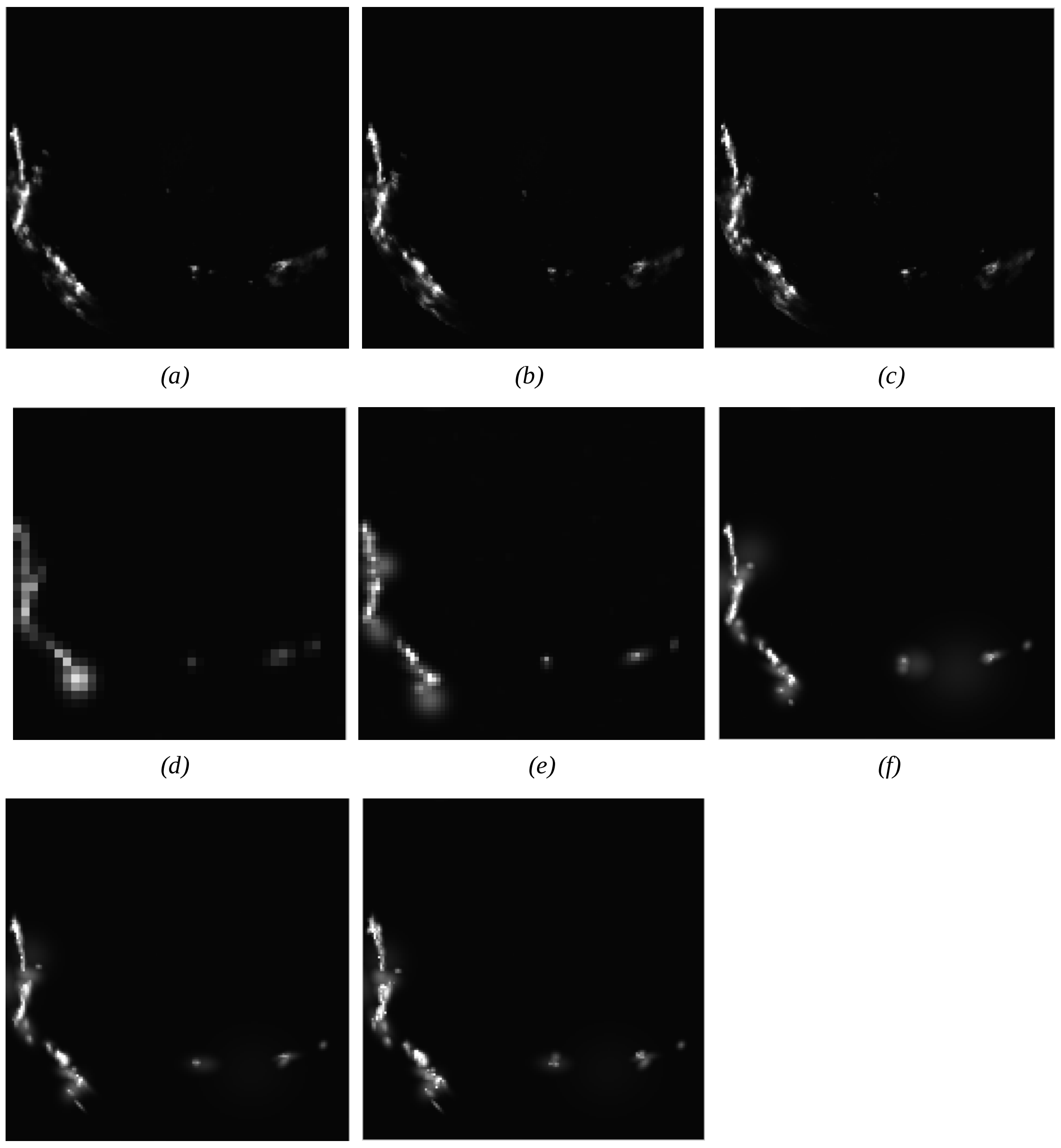
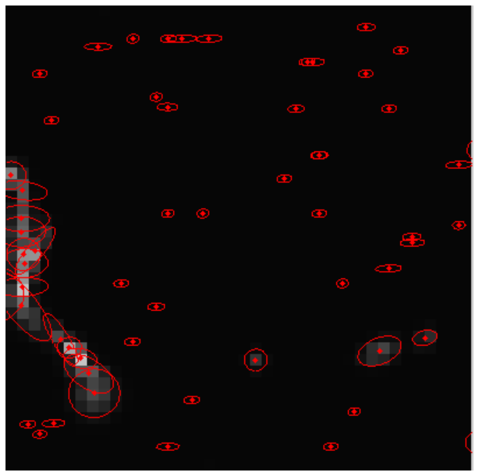
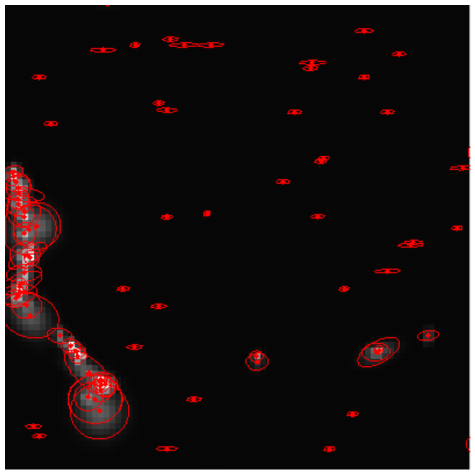


Figure 5: Results obtained using previous algorithm
(a)-(c) are the original images (160 x 160) at time instance 1, 2 and 3.
(d) Pyramidal Stage 1 (40 x 40). (e) Pyramidal Stage 2 (80 x 80). (f) Pyramidal Stage 3 (160 x 160).
(g) Cascade image (160 x 160). (h) forecasted image (160 x 160) using (f) and(g).

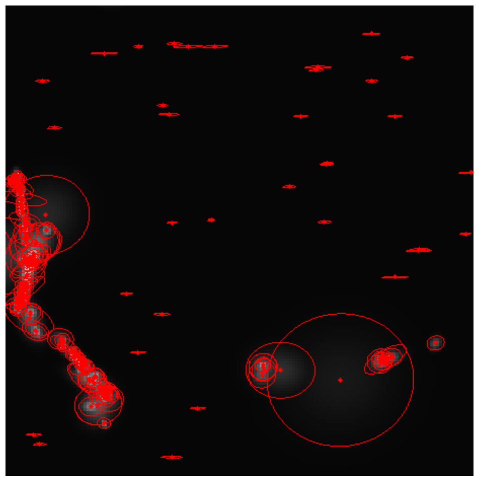
In Figures 6(a)-6(d) the prototype centers are shown. These ellipses are used to illustrate the size and orientation of the Gaussians used to approximate the rain fields in Figures 5(d)-5(g). The size of these ellipses is of course not equal to the extent of the Gaussian envelopes, which is theoretically infinite.



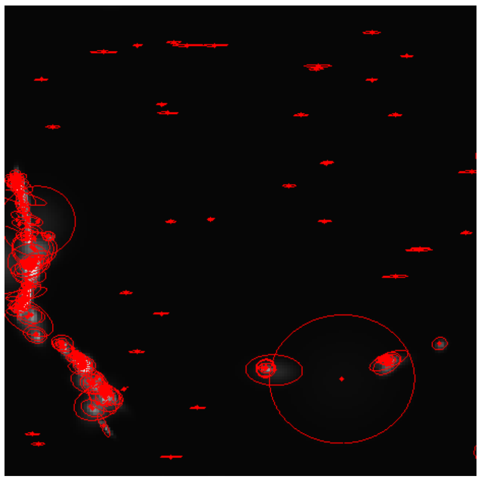
(a)



(b)



(c)



(d)

Figure 6: Images depicting the identified prototypes and ellipses representing the gaussian envelop for images in Figure 5 (d)-(g)
(a) Pyramidal Stage 1 (40 x 40). (b) Pyramidal Stage 2 (80 x 80).
(c) Pyramidal Stage 3 (160 x 160). (d) Cascade image (160 x 160).

Using the same sequence of frames, the modeling of rain events is performed using the proposed method. The new algorithm is different from the previous one in several aspects. The input pixels for the training purposes and the initial centers of the prototypes are strategically selected. Also the prediction is done at a coarse image resolution unlike the previous method where prediction is done on full image resolution. Also the parameters values and conditions used for training are different.

For the same sequence of images used in Figure 5(a)-5(c), the following images are obtained. Figures 7(a)-7(c) are the same images (160 x 160) as in Figures 5(a)-5(c). Figures 7(d)-7(e) are the approximation images of 7(a) and 7(b) respectively. These are of size 40 x 40, since the proposed algorithm performs the modeling and prediction at coarse resolution. Figure 7(f) is the predicted image at full resolution (160 x 160). This prediction is directly obtain from the immediately previous coarse resolution images i.e. Figure 7(d)-7(e). The proposed algorithm does not involve as many steps as in the previous technique [3]. Moreover, the predicted images Figure 5(h) and 7(f) can be compared with the original image in Figure 5(c) or 7(c). It can be noted that, the predicted image in Figure 7(f) looks as a better match than predicted image in Figure 5(h). Also, Figures 7(g)-7(h) provide the prototype information using the ellipses for images in Figures 7(d)-7(e). Additionally, by observing Figures 7(b) and 7(e), it can be concluded that most of the isolated small-scale events, which are not parts of large scale events have a prototype associated with them.

Figure 8 depicts another example. Figure 8(a) is the original image and Figure 8(b) is the approximated image of Figure 8(a). Figure 8(c) is the approximated image in which the prototypes and the oriented ellipses are shown. Figure 8(d) depicts the forecasted image.

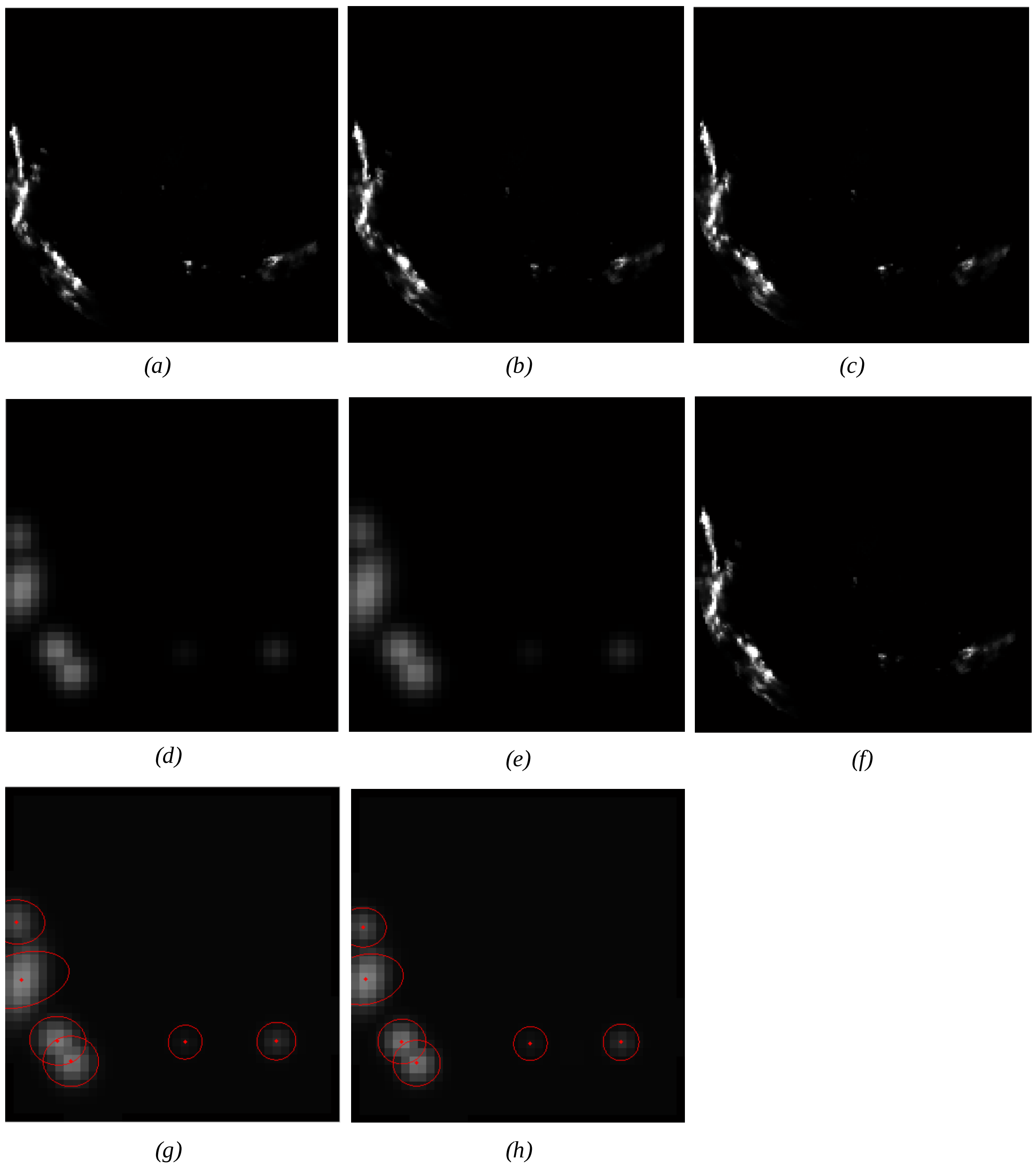
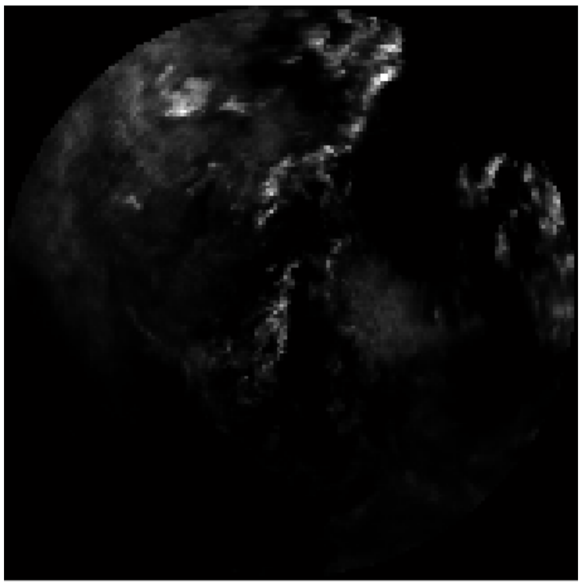
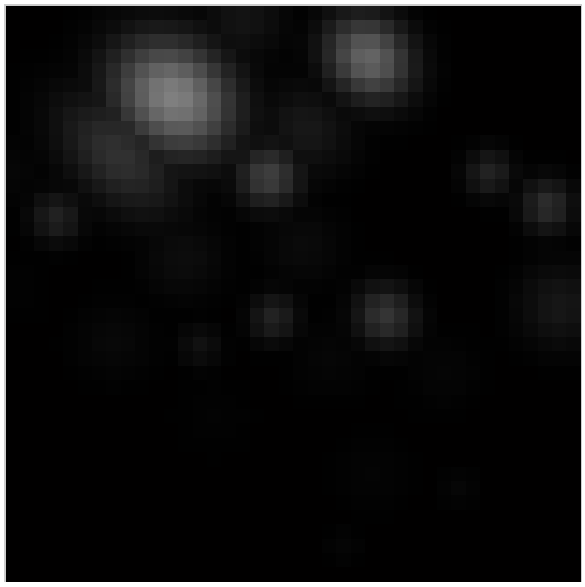
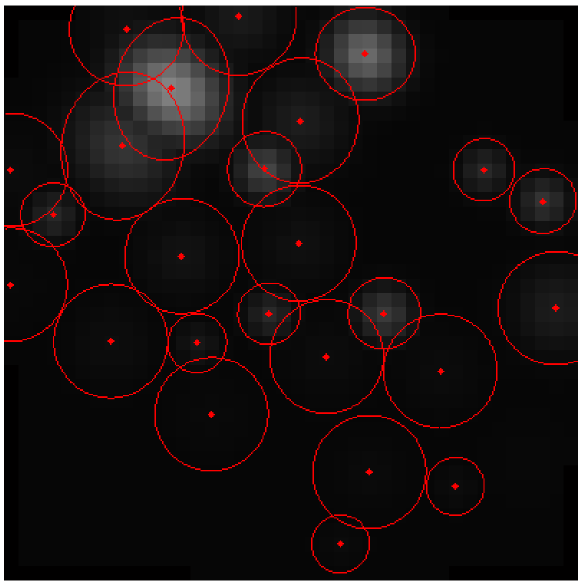


Figure 7: Results obtained using proposed algorithm
(a)-(c) are the original images (160 x 160) at time instance 1, 2 and 3.
(d) approximated image of (a) (40 x 40). (e) approximated image of (b) (40 x 40). (f) forecasted image (160 x 160) using (d) and(e).
(g) and (h) are the images depicting the identified prototypes and ellipses representing the gaussian envelop for images(d) and (e)


(a)


(b)


(c)

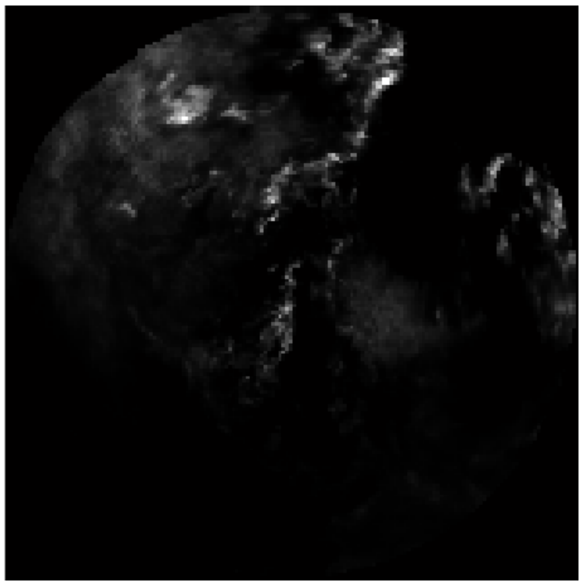

(d)

Figure 8: Results obtained using proposed algorithm
(a) is the original images (160 x 160) (d) approximated image of (a) (40 x 40).
(c) is the image depicting the identified prototypes and ellipses representing the gaussian envelop for image (b). (d) forecasted image (160 x 160) using (d) and(e).

Time and mean square error calculations have been performed for a sequence of frames using the previous algorithm [3] and the proposed algorithm. The average time for each stage of pyramidal synthesis, and cascade synthesis has been calculated using the previous algorithm. Moreover, the total time required for the proposed method is calculated. Since there is a single stage in the proposed algorithm timings are only obtained for this single stage, and therefore correspond to the total timings. The modeling is performed on the first two frames at coarse resolution and the prediction is obtained directly at the original full resolution. These details of the time need for computation is provided in the following table:

Time comparisons Average	**Proposed Method**	**Pyramidal 40 x 40 Resolution**	**Pyramidal 80 x 80 Resolution**	**Pyramidal 160 x 160 Resolution**	**Pyramidal Synthesis Total Time**	**Cascade**	**Total Time for Previous Method**
Seconds	20.67	119.2	330.03	477.53	947.43	282.81	1209.57
Minutes	0.3445	1.98	5.5	7.95	15.7745	4.71	20.14

Table 2: Average time comparisons in seconds and minutes for the previous algorithm and the proposed algorithm. The detailed timings of the each stage of the pyramidal and cascade synthesis are shown. The total time for the proposed algorithm is also shown.

From Table 2, it can be observed that the total time required for the previous algorithm [3] is greater compared to the proposed algorithm. Although cascade synthesis takes less time than pyramidal decomposition, still the proposed technique appears to be more computationally efficient.

The prediction mean square error (mse) for the previous and the proposed work are obtained for a sequence of frames. These mse values are compared with the persistence method mse values. This acts as a reference to evaluate the performance of both previous and proposed algorithms. It can be observed from Table 3 that the mse for the proposed algorithm is smaller in all frames compared to the previous algorithm. Also, the difference in mse between the persistence method and previous method is not significant. However, the proposed method mse is in good standing with the persistence method mse. As mentioned in earlier chapters, this algorithm focuses on better approximation of the rain events. However, the linear prediction technique used in proposed method acts a comparison for evaluating the performances of the two algorithms. Also for information purpose and better understanding, the mse at each stage of the pyramidal synthesis and cascade synthesis for the same sequence is provided in Table 4.

MSE Error / S. No	**Previous Method**	**Proposed Method**	**Persistence Method**
1	6.4386	5.8145	7.3891
2	7.482	6.6214	7.3338
3	12.2844	8.6632	14.2472
4	11.4507	10.1242	14.4324

Table 3: Comparisons of MSE between previous method, proposed method and persistence method for a sequence of frames.

MSE Error / S. No	**Pyramidal 40 x 40 Resolution**	**Pyramidal 80 x 80 Resolution**	**Pyramidal 160 x 160 Resolution**	**Cascade**	**Proposed Method**
1	0.8689	8.3376	2.4748	3.2994	5.8145
2	2.0398	6.2066	2.2944	3.3466	6.6214
3	0.5151	3.0556	2.425	3.4942	8.6632
4	0.4791	2.3515	2.3612	4.9598	10.1242

Table 4: Detailed mse for all stages of pyramidal decomposition, cascade synthesis of the previous method with proposed method for the same sequence of frames used in Table 3

From Tables 2 and 3, it can be observed that the proposed method take very less computational time compared to the previous method and also the mse is less. This is achieved due to the modifications discussed in chapter 3. The decrease in mse is due to strategically selected zero-value input pixels used for the training purpose. Moreover, with a better method for selection of initial positions of the prototype centers and less number input vectors to the RBFNN, the computational time for the proposed method has been considerably reduced. Also, working on the coarse resolution image for prediction purpose, helped in achieving less computational time.

Chapter 5: Conclusion and Future Work

In this thesis, modifications to the existing technique have been proposed and implemented. The focus of the thesis was to improve the quality but mostly the computational efficiency of the precipitation modeling technique. Modeling of rain events, in general, requires more computational time in the overall process of forecasting. However, with the introduction of the modifications presented in this thesis, the computational time has been considerably reduced. More specifically, methods for speeding up RBFNN-based rain field modeling have been proposed. Usage of strategically selected zero-valued pixels for training purpose resulted in a stable approach and less number of input vectors. Also, examples were illustrated to support this stable approach. Moreover, initial selection of prototype centers also helped in better modeling of the rain events. By using the proposed equation (15), it was shown that it is possible to directly obtain a high-resolution forecasted image from downsampled present and past images, without performing the training using any full-resolution images. It is also important to mention that the predicted image is obtained by using the original pixels from the image at time instance 2 and equation (15). These pixel values are extracted by using the predicted localized parameters in equation (15). In the previous method [3], the modeled pixel values are used to obtain the full resolution predicted image. Using the proposed method, to predict a rain map from immediately previous two rain maps it takes approximately 21 seconds, which is less compared to 15 minutes for pyramidal synthesis itself in the previous technique [3]. Hence a greater computational efficiency has been achieved using the proposed technique.

Future work can be lead in many directions. Efficient directional smoothing [16], [17], [18] may be investigated in order to bring out more precipitation characteristics that can assist in the forecasting process. However, the usage of 3D elevation radar map constructed from the regular elevations of the radar is explored. Brief information about the 3D elevation radar map is discussed next.

In general, a single volume scan of radar contains information/images from several elevations. However, the data/pixel-value corresponding to each elevation are obtained at an angle and hence at different height with respect to the ground. More specifically, the pixel value near the radar in the same elevation is at a different height from the pixel value at the farthest position. The technique in [1] and this thesis uses only the first elevation to perform the modeling and prediction, since radar beams associated to higher elevations scans are at a significant height from the ground which is not the area of interest. However, higher radar elevations could be useful for areas located close to the radar.

Future work involves using the techniques presented in [4]. Prior to the modeling and prediction, a 3D volume representation of the elevation is obtained. This 3D volume is used as the input rather than the first elevation for modeling of the rain events. This 3D slice consists of information from the first five elevations of the radar scan. This 3D volume slice is represented as $S(x, y, z)$, where (x, y) are the coordinates of the pixel and z is the height with respect to ground. All the pixels in this 3D slice are considered to be at same height with respect to ground. The procedure for the construction of this 3D volume slice is described next.

First, height z_0 at which the slice is to be build is chosen. The radar rain data is not covered in 3D around the radar, and rain rate values at height z_0 is required to fill the empty grid/slice. Hence, the data from the two elevations which is closest to the (x, y, z_0) position are used. A weighted reflectivity average of the data from the two closest elevations is used to fill the data at position (x, y, z_0). It is important to mention that the center of the 3D slice/grid is radar location. An imaginary line is considered which is perpendicular to the ground and passing through (*x*,*y*), virtually intercepting all the elevations. The point where this imaginary line intercepts the *i-th* elevation is given by (x, y, z_i). The two closest elevations *j* and *k* are considered to be the ones for which the distances $d_j = |z_0 - z_j|$ and $d_k = |z_0 - z_k|$ are the two smaller ones, namely $d_j < d_i$ and $d_k < d_i$, where $i \neq j, k$. Then, the weighted average is computed by [4]:

$$S(x, y, z) = \frac{\left[\frac{1}{(d_j+\delta)} r_j + \frac{1}{(d_k+\delta)} r_k\right]}{\left[\frac{1}{(d_j+\delta)} + \frac{1}{(d_k+\delta)}\right]} \tag{16}$$

where r_j and r_k are respectively the values of the *j-th* and *k-th* elevations at horizontal location *(x,y)*. When the distance d_j or d_k equals zero, the constant δ eliminates the problem. Hence by using this method, all the pixels in the 3D slice are at approximately same height with respect to ground.

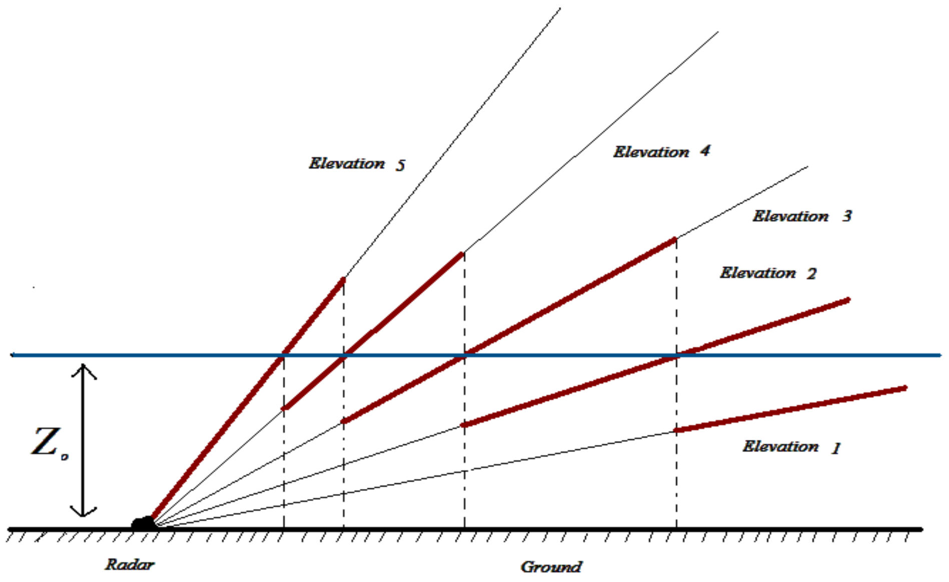


Figure 9: Illustration of building of 3D slice/grid/data representation

The process described above is illustrated in Figure 9. The bold parts of the lines represent the closest regions used by the weighted average for building the corresponding part of the 3D slice. These lines represent the elevations of the radar scan. As it can be noted from Figure 8, pixels from the two closest elevations contribute for building every section/region of the 3D slice. However, it can also be observed that for the region near the radar, the fifth elevation is considered directly. Using this technique, 3D elevations at different heights are obtained as shown in the Figure 8. The pixel coordinates associated to the 3D elevations can be used as the input to the RBFNN rather than the regular elevation. Since the data is a mixture from five different elevations, radar maps are relatively independent with respect to the distance from the radar.

References:

[1] Charalampidis, D., and Paduru, A., "Tracking of storm fronts in weather radar imagery*,"* *Proc. SPIE,* Vol. 7317, 73170C (2009)

[2] Denoeux, T., and Rizand, P., "Analysis of radar images for rainfall forecasting using neural networks*," Neural Computing and Applications,* 3, 50-61 (1995).

[3] Dell'Acqua, F., and Gampa, P., "Pyramidal rain field decomposition using radial basis function neural networks for tracking and forecasting purposes," *IEEE Trans. Geoscience and Remote Sensing,* 41(4), 853-862 (2003).

[4] A. Paduru, D. Charalampidis, "Separation of Rain and Non-Rain events in Radar Imaging using Multiple Elevations and Texture Analysis," *40th Southeastern Symposium on System Theory, New Orleans, 17-* 18 May 2008.

[5] Charalampidis, D., and Paduru, A., "Filtering of weather radar imagery using steerable Gaussian smoothers," Proc. SPIE, Vol. 7308, 730811 (2009)

[6] Li, L., Schmid, W., and Joss, J., "Nowcasting of motion and growth of precipitation with radar over a complex orography," *J. Applied Meteorology,* 34, 1286-1300 (1995).

[7] Chang, E. S., Chen, S., and Mulgrew, B., "Gradient radial basis function networks for nonlinear and nonstationary time series prediction," *IEEE Trans. Neural Networks,* 7, 190-194 (1996).

[8] Guang-Bin Huang, Saratchandran, P., and Sundararajan, N., "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *IEEE Trans. Neural Networks* 16(1), 57-67 (2005)

[9] Charalampidis, D., Kasparis, T., and Jones, L., "Removal of nonprecipitation echoes in weather radar using multifractals and intensity," *IEEE Trans. Geoscience and Remote Sensing,* 40(5), 1121-1131 (2002).

[10] M. A. S. Potts and D. S. Broomhead, "Time series prediction with a radial basis function neural network," *SPIE Adaptive Signal Processing,* Vol. 1565, pp. 225-266, 1991.

[11] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis functions networks," *IEEE Trans. Neural Networks,* vol. 2, pp. 302-309, 1991

[12] M. Casdali, "Nonlinear prediction of chaotic time-series," *Physica D,* vol. 35, pp. 335-356, 1989

[13] D. S. Broomhead and D. Lowe, " Mutlivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321-355, 1988

[14] E. Levin, "Hidden control neural architecture modeling of nonlinear time varying systems and its applications." *IEEE Trans. Neural Networks*, vol. 4, pp. 109-116, 1993

[15] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and control*, Okland, CA: Holden-Day, 1976

[16] Lakshmanan, V., "A Separable Filter for Directional Smoothing," *IEEE Geoscience and Remote Sensing Letters, 1(3), 192-195 (2004).*

[17] Lakshmanan, V., "Speeding up a large scale filter," *Journal of Atmospheric and Oceanic Technology, 17,* 468-473 (2000).

[18] Charalampidis, D., "Efficient Directional Gaussian Smoothers," *IEEE Geoscience and Remote Sensing Letters,* 6(3), 383-387 (2009).

[19] Smith, J.A. and Krajewski, W.F., 1993. "A modeling study of rainfall rate – reflectivity relationships." *Water Resour. Res.*, 29, 2505–2514.

Vita:

Anirudh Paduru was born in Hyderabad, India. He received his undergraduate degree in Electronics and Communication Engineering from J.N.T University, India in May 2007. From Fall-2007 to Fall-2009 he was with Electrical Engineering Department at UNO where he pursued his M.S. in Electrical Engineer and worked with Dr. Dimitrios Charalampidis as Research Assistant. His research interests include Digital Image Processing and Remote Sensing.