Storm-wide Precipitation Retrievals

by

Jessica A. Loparo

B.S., Electrical Engineering (2002)

Case Western Reserve University

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

at the

Massachusetts Institute of Technology

June 2004

Signature of Author [illegible]
Department of Electrical Engineering and Computer Science
May 6, 2004

Certified by [illegible]
David H. Staelin
Professor of Electrical Engineering
Thesis Supervisor

Accepted by [illegible]
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Storm-wide Precipitation Retrievals

by

Jessica A. Loparo

Submitted to the Department of Electrical Engineering and Computer Science
on May 6, 2004 in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Electrical Engineering

ABSTRACT

The monitoring of precipitation is important for scientific purposes, such as the study of world weather patterns, the development of global precipitation maps, and the tracking of seasonal and diurnal variations in precipitation rates. Over time many observation methods have been used to estimate precipitation: rain gauges, ground based radar systems, and visible, infrared, and passive microwave sensors in orbiting satellites. This research project uses data from the Advanced Microwave Sounding Unit (AMSU-A and AMSU-B) which consists of passive microwave sensors that collect data in the opaque water vapor and oxygen microwave absorption bands. This data supports 15-km resolution global precipitation rate estimates. The goal of this research is to develop a computational method that will improve the accuracy of these precipitation estimates by including spatial information in the precipitation retrieval, which is currently pixel based. This spatial information, which consists of the precipitation rate at each pixel of the image, is used to divide the data into separate storms, where a storm is defined as a precipitation region that is separated from other regions by an area of low or zero precipitation. Once storms have been identified, a neural network is used to estimate the integrated precipitation rate over each storm using as input several feature vectors that characterize the initial storm-wide precipitation rate estimates. Then the estimate of integrated precipitation rate is used to adjust the precipitation values of the pixels that correspond to the storm. These methods have resulted in a decrease in the mean-square-discrepancy of the estimate of integrated precipitation rate, as compared to NEXRAD ground-based radar systems, by nearly a factor of two.

Thesis Supervisor: David H. Staelin
Title: Professor of Electrical Engineering

Acknowledgements

I would like to thank my advisor, Professor David Staelin, for his guidance and insight. I would also like to thank Fred Chen for his patience in answering all my questions about his work, and the rest of the Remote Sensing and Estimation Group for their helpful discussions and interesting perspectives.

I want to thank my parents for providing me with unquestioning and unending support. I am thankful for my brother, Joe, for always listening and knowing the right thing to say. I am grateful to Cara, my adorable puppy, for providing much needed study breaks while I was in the midst of writing and editing my thesis.

Finally, I wish to express my heartfelt thanks to Mike for always being patient and supportive, and being able to make me laugh no matter what.

Table of Contents

List of Figures

List of Tables

Chapter One: Introduction and Background

Section 1.1: Introduction

The monitoring of precipitation is important for scientific purposes, such as the study of world weather patterns, the development of global precipitation maps, and the analysis of seasonal and diurnal variations in precipitation rates. Over time many observation methods have been used to estimate precipitation: rain gauges, ground based radar systems, and visible, infrared and passive microwave sensors in orbiting satellites.

This research project uses data from the Advanced Microwave Sounding Unit (AMSU-A and AMSU-B), passive microwave sensors that collect data in the opaque water vapor and oxygen microwave absorption bands. These instruments are located on the NOAA-15 satellite. The goal of this research is to develop a computational method that improves the accuracy of precipitation estimates by including spatial information in the precipitation retrieval, which is currently pixel based. This spatial information, which consists of the precipitation rate at each pixel of the image, will be used to divide the data into separate storms, where a storm is defined as a precipitation region that is separated from other regions by an area of relatively low or zero precipitation. Once storms have been identified, an integrated precipitation rate retrieval will be applied to the area of each storm and the estimate of integrated precipitation rate will be used to adjust the precipitation values of the pixels in the original retrieval.

Compared to other accepted methods, there are many advantages to retrieving precipitation rates from satellite, and specifically passive microwave, data. One advantage of satellites over ground based radar systems is that satellites have uniform global coverage, which allows data to be collected from all over the world rather than just the area surrounding the stationary radar. The Advanced Microwave Sounding Unit is better than other current satellite borne radars because it has a wider swath width, which allows full earth coverage twice per day by a single satellite. Passive microwave sensors have an advantage over visible or infrared systems because of their ability to penetrate non-precipitating clouds. Finally, the combination of AMSU-A and AMSU-B with its opaque and window channels has an advantage over microwave sensors with only window channels because the opaque channels sense precipitation over land without strong surface emissivity effects. With all these advantages over other current methods of sensing precipitation, it becomes important to develop computational methods that can improve precipitation retrievals by including spatial information, which is the major objective of this research effort.

Section 1.2: Background

Section 1.2.1: NOAA-15 (previously known as NOAA-K)

In the spring of 1998, a new series of the National Oceanic and Atmospheric Administration (NOAA) Polar Operational Environmental Satellites (POES) began with the launch of NOAA-15. Its successors include NOAA-16 and NOAA-17; all three satellites have operated well together for more than a year. This family of satellites is designed to improve environmental monitoring in support of NOAA missions.

NOAA-15 was launched May 13, 1998 and is currently in orbit. It is a polar orbiting satellite in a sun-synchronous orbit. The satellite carries many different instruments for environmental monitoring including an Advanced Very High Resolution Radiometer/3 (AVHRR/3), a High Resolution Infrared Radiation Sounder (HIRS), Advanced Microwave Sounding Units A and B (AMSU-A and -B), a Space Environment Monitor (SEM-2) Data Collection System/2 (DCS/2), a Search and Rescue Satellite (SARSAT) Instrument, and a Solar Backscatter Ultraviolet Spectral Radiometer (SBUV/2). As mentioned previously, the instrument of interest for this project is the Advanced Microwave Sounding Unit.

Section 1.2.2: Advanced Microwave Sounding Unit (AMSU)

AMSU consists of two largely independent units: AMSU-A, which observes at 15 frequencies up to 90 GHz with 50-km spatial resolution and AMSU-B, which observes at 5 frequencies between 88 and 191 GHz with 15-km spatial resolution. AMSU-A and AMSU-B images cover the same swath width of approximately 2200 km and are 30 and 90 spots wide, respectively. The AMSU-A unit is a multi-channel microwave radiometer that is used to measure global atmospheric temperature profiles and provide information on atmospheric water in all of its forms except small ice particles, which are transparent at these microwave frequencies. The AMSU-B unit receives and measures radiation from a number of different layers within the atmosphere in order to obtain global data on humidity profiles.

For more information on the NOAA-15 satellite or the Advanced Microwave Sounding Unit, please refer to the NOAA-KLM User's Guide [1].

Section 1.2.3: Description of Precipitation Retrieval Algorithm

AMSU data is in the form of microwave radiances. Because the relationship between microwave radiances or brightness temperatures and precipitation is non-linear, the AMSU precipitation retrieval algorithm uses neural networks trained to retrieve precipitation rates that agree with NEXRAD precipitation estimates [2],[3]. This technique uses opaque oxygen and water vapor channels near 54 and 183 GHz, respectively. These channels are indirectly responsive to vertical wind velocity and absolute humidity, the product of which is proportional to precipitation. The vertical wind is correlated with cell-top altitude, microwave albedo, and hydrometer diameters, all of which influence the microwave spectrum between 50 and 200 GHz independently, and are therefore related to precipitation. The humidity profile is sensed by observations near 183 GHz with the aid of the temperature profile information obtained near 54 GHz.

The 15-km resolution precipitation rate retrieval begins with the determination of which pixels may possibly be precipitating. All 15-km pixels with brightness temperatures at 183 ± 7 GHz that are below a threshold T_7 are flagged as potentially precipitating, where

$$T_7 = 0.667\,(T_{53.6} - 248) + 243 + 6\cos\theta \qquad (1)$$

and where θ is the satellite zenith angle and $T_{53.6}$ is the spatially filtered 53.6-GHz brightness temperature obtained by selecting the warmest brightness temperature within a 7 by 7 array of AMSU-B pixels. If, however, $T_{53.6}$ is below 248 K, the 183 ± 3-GHz band is used to flag potentially precipitating pixels. This is done because a value of $T_{53.6}$ below 248 K can result in the 183 ± 7-GHz flag being mistakenly set by the low surface emissivity in very cold dry atmospheres. In this case, instead of looking at the brightness temperature at 183 ± 7 GHz, the brightness temperature at 183 ± 3 GHz is compared to a threshold T_3, where

$$T_3 = 242.5 + 5 \cos \theta \tag{2}$$

These thresholds are slightly colder than a saturated atmosphere would be, which indicates the presence of a microwave absorbing cloud. Finally, if $T_{53.6}$ is below 242 K, the pixel is assumed to be non-precipitating.

The next step is to use the 50-km resolution 52.8-GHz perturbations to infer the perturbations that might have been observed at 52.8 GHz with 15-km resolution if those perturbations were distributed spatially in the same way as those cold perturbations observed at either 183 ± 3 GHz or 183 ± 7 GHz. This is done by re-sampling the bi-linearly interpolated 50-km AMSU data at the 15-km cell positions. The inferred 15-km radiances are computed by

$$\Delta T_{15,54} = (\Delta T_{15,183} / \Delta T_{50,183})\, \Delta T_{50,54} \tag{3}$$

The perturbation $\Delta T_{15,183}$ near 183 GHz is defined to be the difference between the observed radiance and the appropriate threshold, whether it be T_7 or T_3. The perturbation $\Delta T_{50,54}$ near 54 GHz is defined to be the difference between the observed radiance and the Laplacian-interpolated radiance based on the pixels surrounding the flagged region. Any warm perturbations are set to zero. Limb- and surface-emissivity corrections to nadir for the five 54-GHz channels are obtained as the output of neural networks that are applied to each channel.

The inferred 15-km resolution perturbations at 52.8, 53.6, 54.4, 54.9, and 55.5 GHz are then combined with the 183 ± 1, 183 ± 3, and 183 ± 7 15-km resolution data from AMSU-B along with the leading three principal components that characterize the original five corrected 50-km AMSU-A temperature radiances, and two surface-insensitive principal components that characterize the window channels at 23.8, 31.4, 50, and 89 GHz as well as the channels near 150- and 183-GHz. These thirteen variables, plus the secant of the satellite zenith angle θ, were used as the inputs to a neural network. The neural networks were trained to minimize the root-mean-square (RMS) error between the logarithms of (AMSU + 1-mm/hour) and (NEXRAD + 1-mm/hour). The logarithms reduce the emphasis on the heaviest rain rates, and the addition of 1-mm/hour reduced the emphasis on the smallest rain rates. Eighty different neural networks were trained, and the network with nearly the best performance was selected. The network that was selected had one hidden layer with 5 nodes that had the hyperbolic tangent sigmoid transfer function, and one linear output node. The output of the network is a 15-km resolution retrieval of precipitation rates.

The NEXRAD data consisted of 38 coincident orbits of NOAA-15 AMSU data obtained over the eastern United States and coastal waters during a full year. The data contained significant precipitation, and was seasonally balanced. For the data to be used for training, the NEXRAD precipitation retrievals with 2-km resolution were smoothed to approximate the view-angle distorted 15- or 50-km antenna beam patterns. Because the accuracy of NEXRAD precipitation observations is known to vary with distance, Chen and Staelin [2] used only points beyond 30 km but within 110 km of each NEXRAD radar site to train and test the neural networks.

Section 1.2.4: Precipitation Retrieval Algorithm Results

Early efforts by Staelin and Chen [4] using a simpler algorithm on data from AMSU-A and -B saw reasonable results as follows. For the precipitation retrievals near 183 GHz, the estimates match closely the shape of NEXRAD hurricane and frontal precipitation data that had been smoothed to 15-km resolution. For retrievals using data near both 54 and 183 GHz, comparison with NEXRAD data smoothed to 50-km resolution gave RMS errors for two frontal systems and two passes over hurricane George of ~1.1 mm/hour, and ±1.4 dB for those precipitation events over 4 mm/hour. Only 8.9% of the total AMSU retrieved precipitation was in areas where AMSU saw more than 1 mm/hour and NEXRAD saw less than 1 mm/hour, and only 6.2% of the total NEXRAD rainfall was in areas where NEXRAD saw more than 1 mm/hour and AMSU saw less than 1 mm/hour.

The improved algorithm, as described in Section 1.2.3, yielded even better results [2],[3]. The precipitation rates were classified into the following categories, all in mm/hour: <0.5, 0.5-1, 1-2, 2-4, 4-8, 8-16, 16-32, and >32. The RMS errors for the 3790 pixels that were not used for training are as follows, all in mm/hour: 1.0, 2.0, 2.3, 2.7, 3.5, 6.9, 19.0, and 42.9, respectively. It seems that the precipitation retrieval algorithm neither grossly over- nor under-estimates rainfall. Only 3.3 percent of the AMSU-derived rainfall occurs in areas where AMSU saw greater than 1 mm/hour while NEXRAD saw less than 1 mm/hour and only 7.6 percent of NEXRAD-derived rainfall was in areas where NEXRAD saw greater than 1 mm/hour and AMSU saw less than 1 mm/hour.

The Chen and Staelin [2] precipitation retrieval algorithm has also been used with success to detect artic snowstorms [3],[5],[6]. Artic precipitation has been observed by multispectral passive microwave units aboard the Aqua and NOAA-15, -16, and -17 satellites. Events were considered precipitation if they exhibited appropriate radiometric signatures and morphological evolution over consecutive satellite overpasses at approximately 100 minute intervals. For example, on July 20, 2002 Aqua observed a precipitation event approximately 200 by 1000 km in size moving from the North Pole toward northern Canada which was consistent morphologically with numerical weather prediction. Because the precipitation retrieval algorithm was calibrated for mid-latitude precipitation the polar observation require substantial scaling, but the shape of precipitation events can be detected.

Chapter Two: Motivation

The precipitation retrieval algorithm of Chen and Staelin [2] uses passive microwave measurements in the window channels (10 – 90 GHz) and in the opaque channels (50 – 190 GHz) to estimate precipitation rates over an area using a strictly pixel-based method. In this research project we explore the possibility of improving the estimate of the total precipitation volume of a storm, in units of cubic meters per second, by incorporating spatial information. This new approach is used to estimate total precipitation volume for a convective storm or cluster of storms which will then be used to adjust the precipitation values of the pixels within the storm or cluster of storms. Although it is possible to integrate pixel-based precipitation retrievals to find an approximate total rain volume for a storm, additional physical constraints that apply to storm systems may improve these retrievals.

Section 2.1: Physical Issues in Convective Storms

High-spatial-resolution passive microwave images of storm cells as seen from satellites at frequencies from 30 – 430 GHz appear broader at shorter wavelengths. The physical mechanism behind this tendency offers an opportunity to improve precipitation retrievals for integrated storm systems [7].

A single convective cell can be modeled as consisting of a column shaped updraft with radial inflow near the surface and radial outflow at the top above the freezing level. In this model, the apparent diameter of the cell as seen from above will be determined by the rate at which different size ice particles drop out of the outflow storm cap. Because smaller ice particles fall more slowly they would reside longer at a given altitude and would travel longer distances from the storm center than would the larger particles, which fall more quickly. If there is wind shear above the storm cell, the hydrometer distribution will be elliptical and off center from the convective core, but this does not the change the rate at which particles will fall and where they will fall relative to the core.

The terminal fall velocity of the particles can be estimated using a simple physical model, which shows that particles of 1-mm diameter will fall at half the speed of those having 4-mm diameters, and thus will travel nearly twice as far from the convective storm center before falling out of the outflow region (radial velocity decreases with radius from the core). However, the scattering cross-section of the particles is related to their diameter-to-wavelength ratio. Therefore 1-mm diameter particles falling at 5 ms^{-1} will have radiometric signatures at 200 GHz that resemble those of 4-mm diameter particles falling at 10 ms^{-1} at 50 GHz. As a result, the radiometric diameter of an outflow region near 200 GHz should be roughly twice that of the same region at 50 GHz for constant radial velocity.

In equilibrium the total precipitation from a convective cell approaches the total water vapor (kg s^{-1}) lofted per second, which is proportional to the product of the inflow humidity (kg m^{-3}) and the total lofted air volume (m^3 s^{-1}). The total lofted air volume is directly related to the radial velocity and depth of the outflow region, and total water vapor can be estimated from the oxygen and water vapor band microwave observations in adjacent non-precipitating pixels. The radial reach of any given particle size within the outflow is largely independent of the depth of the outflow region because thicker outflows give particles more time to fall, which directly compensates for the lower outflow velocity. The consequence is that, given the total water vapor, the apparent diameter of the outflow region at any single frequency is sufficient to estimate the equilibrium total rainfall rate (m^3 s^{-1}) of that convective cell. If total water vapor is known, further spectral measurements may not be necessary for this method to work. If this model that links storm microwave radius to vertical air mass and water transport is validated, it will allow a new approach to estimating total cell precipitation rate to be used.

So far a steady state situation has been assumed. It is important to note that the lifetimes of precipitation cells can be on the order of tens of minutes, and this is comparable to the transit time of humid air aloft and to the residence times of smaller hydrometers within the outflow region. Furthermore, because new convective cells will have strong concentrations of large and small hydrometers near their cores, the outflow region may well not yet be fully occupied by hydrometers, possibly resulting in a lower precipitation estimate. At the same time, no rain may yet have reached the ground. Similarly, an exhausted core might leave behind a large outflow region containing only slowly falling small particles, causing an overestimate of the precipitation if it is based on short wavelengths sensitive to small particles. By observing the cell top and outflow diameters at several frequencies these cases would be identified: typically, new cells would have comparable outflow diameters at all wavelengths, while exhausted cells would have larger diameters at the smallest wavelengths and little signature at the longer wavelengths. This offers an opportunity to estimate cumulative and projected lifetime cell precipitation (m^3), as well as instantaneous rates (m^3 s^{-1}).

Section 2.2: Storm-Based Precipitation Retrieval Methods

As discussed previously, estimates of near surface absolute humidity multiplied by the areas occupied by icy hydrometers in cellular outflows should be sufficient to estimate total steady state storm precipitation rates. In addition, the developmental state of the storm could be estimated from the spatial distribution of particle sizes and relative abundances in the cell top. Because the relationships between all observed parameters are non-linear and non-Gaussian, a neural network is a good method to use to retrieve precipitation rates.

In the absence of precise physical models, training the neural network with accurate rainy radar data makes sense. The predicted parameters could include cell rainfall rate, cell total rainfall up to the current time, cell lifetime rainfall, and pixel rain rates. The inputs to the network should include all observables discussed previously.

As a first step to validating these ideas of storm-based precipitation retrieval methods, precipitation rates retrieved from AMSU data will be used to test the idea of storm-wide precipitation retrievals. This will be done by segmenting the precipitation images into separate storm regions, retrieving a measure of rain volume over the storm, and analyzing whether using this estimate of rain volume to adjust the original retrieval improves accuracy.

Chapter Three: Methodology

Section 3.1: Further Introduction to the Problem

The goal of this project is to improve the accuracy of estimates of the integrated precipitation rate by including spatial information; precipitation areas are to be decomposed into storm regions that are analyzed one at a time. The input data consists of an array of estimated precipitation values for each cell of the array. There are two steps in finding an accurate estimate of the integrated rain rate: segmenting the rain image into independent storm regions and training a neural network to retrieve an integrated rain rate from parameters of each segmented storm.

The training, testing and validation data consisted of 38 orbits of coincident AMSU and NEXRAD rain data over the eastern United States. The eastern United States was chosen because it is an area for which it was possible to obtain dense and accurate NEXRAD data. The NEXRAD data used for this step contained both near and far points, where near points are beyond 30 km but within 110 km of a NEXRAD radar site and far points are beyond 110 from a NEXRAD radar site. NEXRAD precipitation estimates are most accurate for near points, but both near and far points were used due to the limited size of the data set and the need for the detection of a significant number of storms to train, validate and test a neural network.

Table 3.1 contains the dates and times of the 38 orbits that were chosen.

16 Oct 1999, 0030 UTC	2 Apr 2000, 0100 UTC	16 Jun 2000, 0130 UTC
31 Oct 1999, 0130 UTC	4 Apr 2000, 0015 UTC	30 Jun 2000, 0115 UTC
2 Nov 1999, 0045 UTC	8 Apr 2000, 0030 UTC	4 Jul 2000, 0115 UTC
4 Dec 1999, 1445 UTC	12 Apr 2000, 0045 UTC	15 Jul 2000, 0030 UTC
12 Dec 1999, 0100 UTC	12 Apr 2000, 0215 UTC	1 Aug 2000, 0045 UTC
28 Jan 2000, 0200 UTC	20 Apr 2000, 0100 UTC	8 Aug 2000, 0145 UTC
31 Jan 2000, 0045 UTC	30 Apr 2000, 1430 UTC	18 Aug 2000, 0115 UTC
14 Feb 2000, 0045 UTC	14 May 2000, 0030 UTC	23 Aug 2000, 1315 UTC
27 Feb 2000, 0045 UTC	19 May 2000, 0015 UTC	23 Sep 2000, 1315 UTC
11 Mar 2000, 0100 UTC	19 May 2000, 0145 UTC	5 Oct 2000, 0130 UTC
17 Mar 2000, 0015 UTC	20 May 2000, 0130 UTC	6 Oct 2000, 0100 UTC
17 Mar 2000, 0200 UTC	25 May 2000, 0115 UTC	14 Oct 2000, 0130 UTC
19 Mar 2000, 0115 UTC	10 Jun 2000, 0200 UTC	

Table 3.1: Dates and times of 38 AMSU orbits chosen for data set

The orbits of AMSU data that were chosen were processed using the Chen and Staelin [2] precipitation retrieval algorithm and then the part of the orbit over the eastern United States was selected and cropped to the appropriate size. The NEXRAD data was then cropped to the same size and locale as the AMSU data and smoothed to the 15-km resolution of the AMSU data.

Figures 3.1 and 3.2 illustrate the two different types of precipitation images that are used in this project. Figure 3.1 is an AMSU precipitation image that was retrieved using the precipitation retrieval algorithm mentioned above, which was described in detail in Section 1.2.3. Figure 3.2 is a NEXRAD precipitation image. These two images are of the same locale, and contain data from approximately the same time. Note the slight differences in the shape and the intensity of the precipitation detected in each image; the storm-wide precipitation retrieval algorithm is aimed at decreasing the effect of these differences. Figure 3.1 has contour lines at 10, 5, and 1 mm/hour. Figure 3.2 has contour lines at 20, 15, 10, 5 and 1 mm/hour.

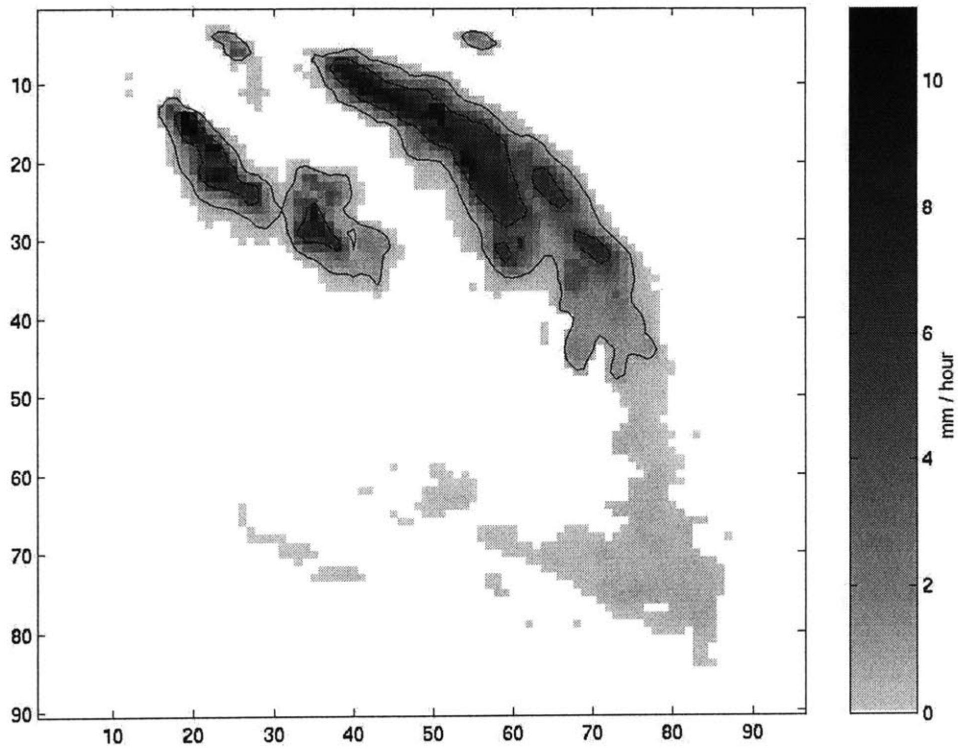

Figure 3.1: Sample precipitation image created by precipitation retrieval algorithm

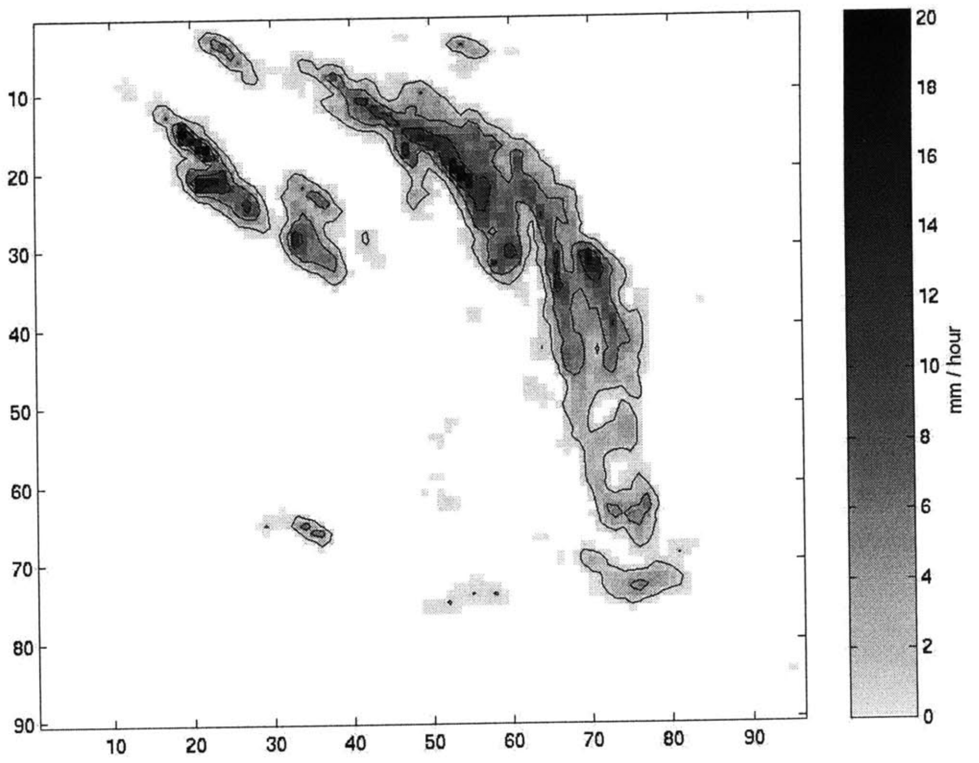

Figure 3.2: NEXRAD precipitation image corresponding to same time and place as Figure 3.1

Section 3.2: Storm Segmentation Techniques

Section 3.2.1: Storm Segmentation Methods that were explored

The process of segmenting a precipitation image into independent storm regions is a complex problem that could be solved in a range of different ways. There are a wide variety of related problems that use clustering and image processing techniques to associate similar features of an image with each other and then use this information to segment (divide up) the image. Precipitation images displayed in gray scale, especially with contour lines, can reveal how pixels are grouped into storms. The storm segmentation algorithm was developed to utilize the same types of information that make it possible for a human to see which storms are independent, and to determine logically which pixels should be associated with each identified storm region. Many different ideas were explored in an effort to find a storm segmentation method that was both effective and efficient.

There are two basic steps to segmenting an image into independent storm regions: (1) determining the locations of the independent storm centers and (2) assigning each of the precipitating pixels of the image to one of the independent storms. Figure 3.3 is a flow chart of the segmentation process.

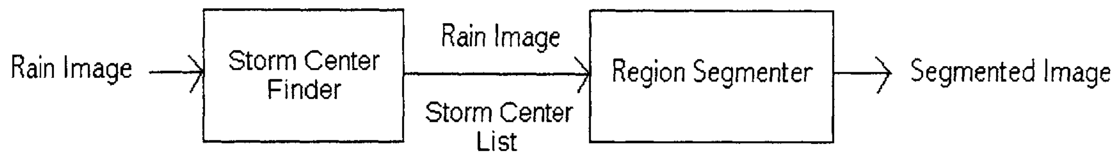

Figure 3.3: Flow chart of segmentation process

In each of the methods that were explored, determining the locations of the independent storm centers was done in much the same way, but the method for assigning pixels to storms varied greatly.

To begin, a brief overview on how the independent storm centers were located is presented. First, the possible storm centers would be determined by finding the regional precipitation maximums and then the regional maximums would be grown outwards, claiming pixels that are associated with that region. When two regions overlap, the comparative levels of the two storm centers would be related, along with a measure of the valley between the two storm centers (the level where they overlap) to determine whether the storm centers are independent, or whether they are indicative of variations within the same storm. If the storm regions were determined to be independent, the pixel location of the overlap between the two regions would be saved for later use, and the algorithm would return to the process of growing the regions. If, however, the regions were found to be dependent, the pixels associated with the smaller regional maximum would be incorporated into the area of the region associated with the larger regional maximum. Once all regional maximums have been compared to each other and the remaining storm centers have been verified to be independent, the next step is to determine where the dividing points between storm regions are located.

One idea that was explored for this problem was to determine the independent storm centers, then use an optimization method to determine where the dividing lines between the storms lie. The motivation behind this idea was that between two independent storm regions there must be an area of low to zero precipitation, and a properly designed optimization method utilizing a random walk to traverse the precipitation image will find the best dividing path between two independent storms. The best division would be defined as the path that truly divides the two storm centers, with one peak on either side of the division, and has the lowest average amount of rain per pixel along the dividing path.

There were many difficulties associated with this method that made it impractical. The first difficulty was in determining and implementing a metric for the optimal path. Ideally, the dividing path should be the shortest path that has pixels with the lowest precipitation values. The metric for the path needs to balance path length and the precipitation values of pixels. The difficulty that this presents in practice is that to find the optimal path, an exhaustive search over the entire precipitation image must be performed. To implement a random walk to find all possible paths is a very computationally complex problem. Therefore implementing the same random walk many, many times to find the dividing paths between multiple storms becomes much too computationally intense to be practical.

In an effort to solve the problem caused by the computational complexity of finding all possible paths that will separate two storms and computing their values to find the optimal dividing path, a greedy optimization was tested. For this case, the path was only allowed to grow to pixels that were at an equal or lower value than the pixel that had just been claimed as part of the path. The difficulty associated with this method is that to select one pixel that is located between the two storm centers and extend it into a line that will divide the two storm centers into separate regions is challenging. If the algorithm is designed so that at each step the path extends to the two lowest pixels, the path may not grow in such a way that it will divide the two regions. For example, take the situation shown below, where each block corresponds to the precipitation value of a pixel in mm/hour, and the grid of the table represents the x- and y directions in space.

17	12	11	11	11	30 (a)
20	15	11	10 (c)	11	16
25 (b)	17	8	9	11	17
22	15	5	6	13	14

The values labeled (a) and (b) are the independent peaks, the value labeled (c) is the point where the claimed pixels overlapped, and the edges of the table are the edges of the precipitation image. In the first step the algorithm will be looking at the pixels

		11	11	11	
		11	10 (c)	11	
		8	9	11	

The pixels that will be claimed have the values 8 and 9. These pixels happen to be next to each other. In the next step the points to be examined will be as follows:

	15	11	**10 (c)**		
	17	**8**	9		
	15	5	6		

		11	**10 (c)**	11	
		8	**9**	11	
		5	6	13	

Now ends of the path will want to claim the pixel that has the value 5, and the dividing path will end up as in the table below with the dividing pixels, which are the pixels that are defined as the dividing line between the two storm regions, being bolded:

17	12	11	11	11	30 (a)
20	15	11	**10 (c)**	11	16
25 (b)	17	**8**	**9**	11	17
22	15	**5**	6	13	14

As can be seen, this path does not divide the two storm centers into separate regions, which illustrates that a greedy algorithm will not always be effective. This highlights another problem with any random walk, whether is greedy or exhaustive: it is very difficult to devise an algorithm that will check to see if the path has actually divided the two storm centers into two separate regions. Also, even if it is possible to determine when the path doesn't actually divide the storm regions the only recourse would be to use the next most optimal path which also may not divide the storm regions.

Finally, the last difficulty that was discovered with these methods was that the precipitation image may be divided into too many regions. For example, take the case of a precipitation image with three storm centers, A, B, and C. Using this algorithm, there will be three dividing paths: one between regions A and B, one between regions A and C, and one between regions B and C. But take the case where the peaks are next to each other in a line from left to right: A, then B, and finally C. The dividing path between A and C will most likely run through the region that should be assigned to peak B.

Because of the difficulties associated with implementing a random walk to divide the storm centers, an entirely different approach was developed and tested. In the process of determining which regional maximums are independent storm centers the algorithm assigns pixels to each regional maximum. However, as the regions grow, many different regional maximums are allowed to all claim the same pixel. This needs to occur to allow a comparison of the valley or division between two regional maximums so it can be determined whether or not they are independent. An important aspect of the segmentation process is that when pixels are being assigned to a storm region, each pixel that is judged to be precipitating can be assigned to one and only one storm region. However, with some adaptations the process of claiming pixels that is used to determine which regional maximums are independent can also be used to assign pixels to a storm region. These adaptations are to prevent a pixel from being claimed by more that one storm region, and to also allow each storm center that is above the current threshold to claim the pixels near it that are also above the threshold. This last step requires that the pixels that reside within one pixel of the current edge of the region be claimed before allowing any storm to claim pixels that are within two pixels of the current edge, etc. This method will be thoroughly discussed in the next section, and the benefits and drawbacks will be discussed in Section 4.1.

Section 3.2.2: Detailed Description of the Storm Segmentation Algorithm

The segmentation of the rain images is performed in two steps: (1) isolating independent storm centers and (2) determining which pixels of the rain image are associated with each independent storm.

The first part of the segmentation process consists of taking a rain image and determining which pixels correspond to independent storm centers. Figure 3.4 illustrates the process of determining the independent storm centers and their locations in a flow chart, and a detailed description of the process follows.

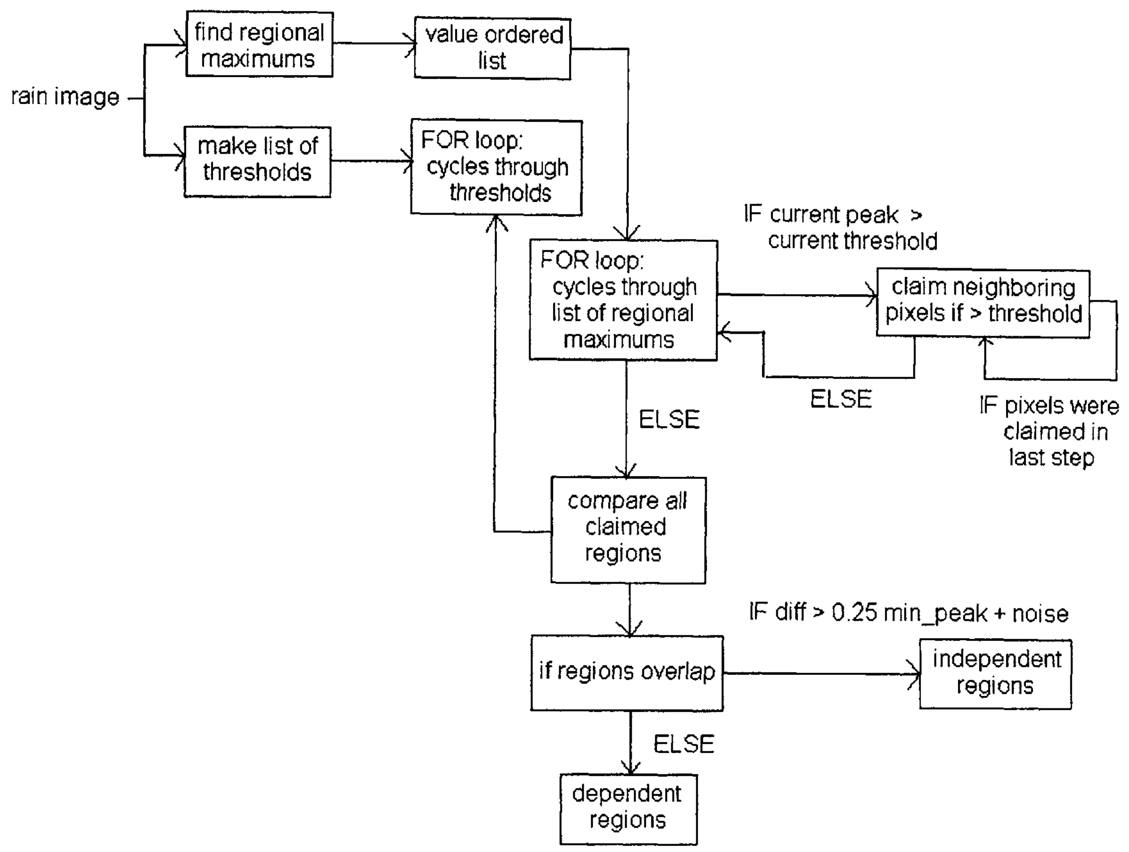
rain image
find regional maximums
value ordered list
make list of thresholds
FOR loop: cycles through thresholds
FOR loop: cycles through list of regional maximums
IF current peak > current threshold
claim neighboring pixels if > threshold
ELSE
IF pixels were claimed in last step
ELSE
compare all claimed regions
IF diff > 0.25 min_peak + noise
if regions overlap
independent regions
ELSE
dependent regions

Figure 3.4: Flow chart describing the process that determines independent storms

The precipitation image shown in Figure 3.5 will be used to illustrate the steps of the segmentation process. Figure 3.5 has contour lines at 15, 10, 5, and 1 mm/hour.

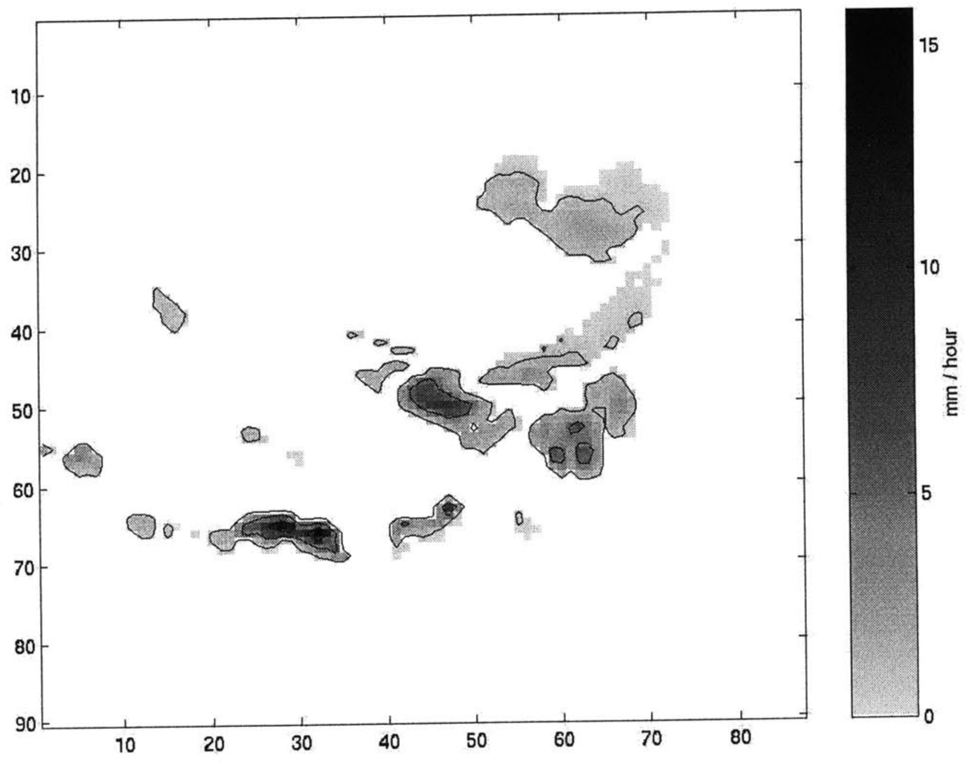

Figure 3.5: Sample precipitation image

The first step of determining the independent storm centers is finding the pixel locations of the regional maximum rain rates using an image processing function. A regional maximum rain rate is defined as a pixel that has a higher rain rate than each of its eight-connected neighboring pixels. In Figure 3.6, the regional maximums are highlighted in black and all other precipitation is gray.

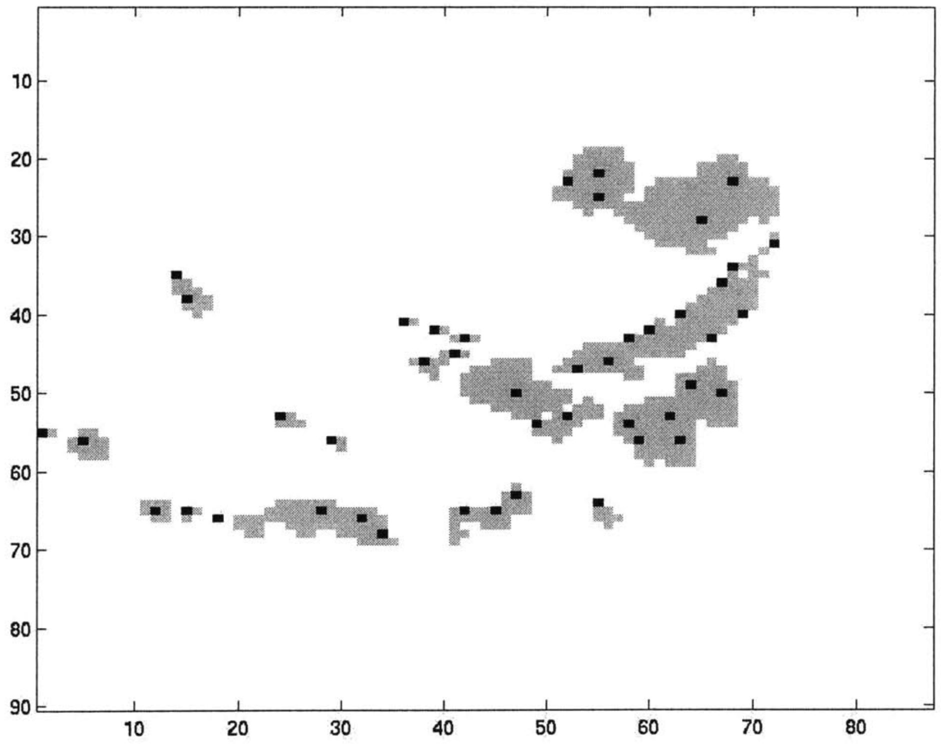

Figure 3.6: Precipitation image from Figure 3.5 with regional maximums highlighted in black

Once the regional maximums are determined, a list of the possible storm centers is compiled by determining which regional maximums are above the noise threshold. The noise threshold was selected to be 0.05 mm/hour because it has been found by comparison with NEXRAD images that areas with rain rates below 0.05 mm/hour are likely to be non-precipitating.

Once a list of all the regional maximums that are possible storm centers has been compiled, the next step is to begin determining which of the regional maximums indicate the peak rate of an independent storm region, and which indicate variations of rain intensity within a storm region. This is done by selectively growing each regional maximum one layer of pixels at a time and comparing the points where the regions begin to overlap. The first step of this process is to create a threshold list. The threshold list is simply a list where the first value is the highest precipitation value in the image and going down to a precipitation value of 0.05 mm/hour in steps of 1 mm/hour. This list is used to determine which regional maximums can be grown at a specific time and which pixels can be claimed by the regional maximums that are allowed to grow. For a regional maximum to be allowed to grow, it must have a precipitation value greater than the current threshold. For a pixel to be claimed by a regional maximum that is allowed to grow, it must be greater than the current threshold, but smaller than the previous threshold. This means that there is a range of precipitation values that can be claimed at any step, where the length of the range is 1 mm/hour. The claiming process is set up in two loops: the outer loop cycles through the threshold list, and the inner loop cycles through the list of regional maximums. Starting from the highest regional maximum, the eight neighboring pixels of each regional maximum can be claimed as thresholds are passed. Once all the pixels that can be claimed at any step (i.e. pixels above the current threshold but below the previous threshold that are neighboring to already claimed pixels) are claimed, the regions that overlap are compared to determine which are independent from each other.

Two overlapping regions are determined to be independent or dependent by comparing the precipitation values of their respective regional maximums to the precipitation value of the point where the two regions overlap. The two regional maximums are compared to determine which is smaller, and then the difference between this smaller precipitation value and the precipitation value of the overlap is compared to a pre-determined threshold, which can be adjusted to obtain a better segmentation. The default threshold is as follows: if minimum peak – overlap value > 0.25 (minimum peak) + 1 mm/hour, the regional maximums are independent. Otherwise, the regional maximums are dependent, and the pixels associated with the smaller regional maximum will be folded into the region associated with the larger regional maximum. This process of regional maximums claiming pixels and being compared for region overlap continues until the end of the threshold list is reached, all pixels considered to be precipitating with a value above 0.05 mm/hour are claimed by some storm, and all the remaining regional maximums have been compared to determine if they overlap.

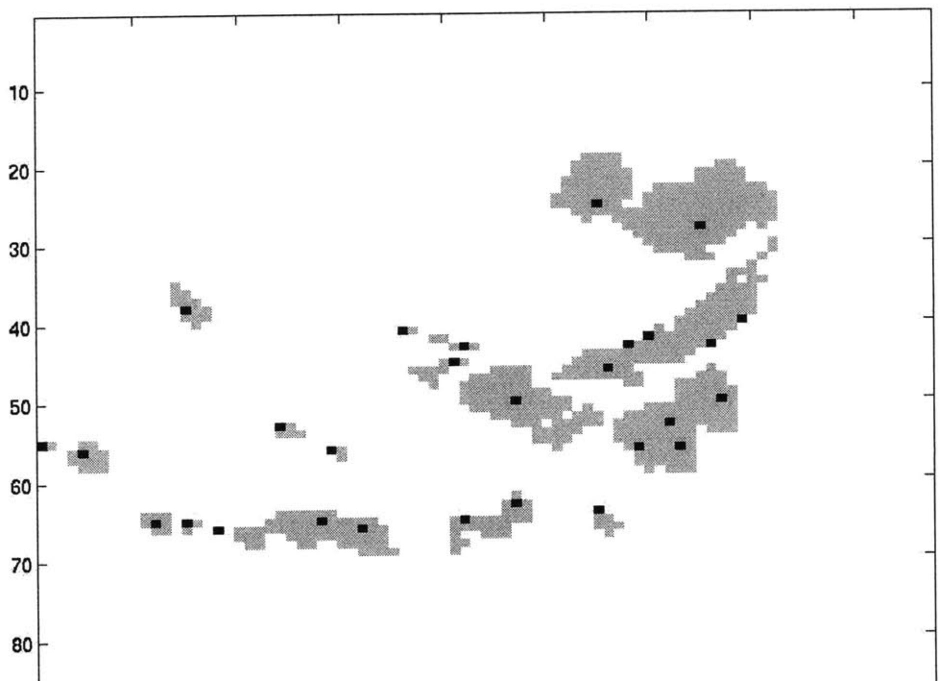

Figure 3. 7: Precipitation image from Figure 3.5 with independent peaks highlighted in black

At this point, a list of the independent storm centers, where each independent storm center was one of the regional maximums that were determined in the beginning, is output by the program. In Figure 3.7, the precipitation image from Figure 3.5 is shown with the independent storm centers highlighted in black and all other precipitation is gray.

The second step of the segmentation algorithm is to determine which pixels of the rain image should correspond to each storm. This process is similar to the process used to determine which regional maximums are independent storm centers, with some important changes. Figure 3.8 illustrates the process of assigning pixels to the regions of the independent storm in a flow chart, and a detailed description of the process follows.

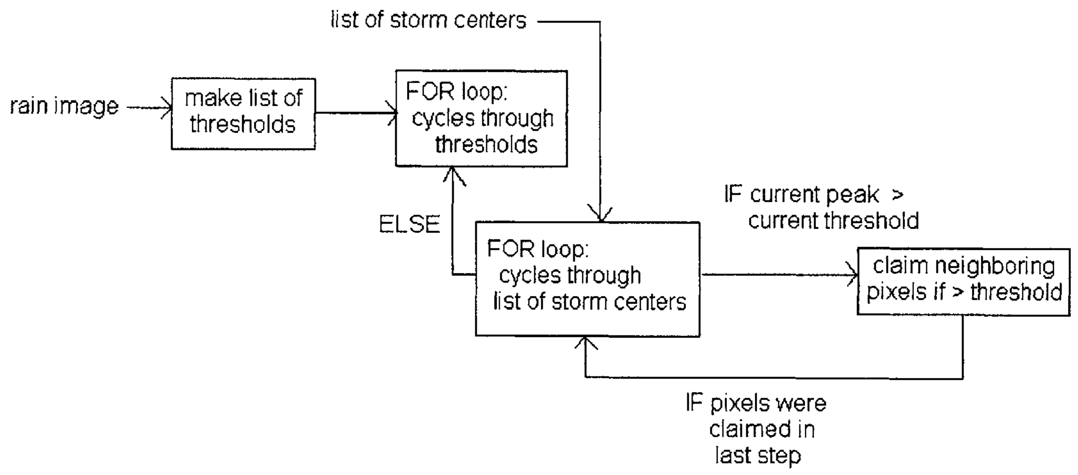
list of storm centers
rain image
make list of thresholds
FOR loop: cycles through thresholds
ELSE
FOR loop: cycles through list of storm centers
IF current peak > current threshold
claim neighboring pixels if > threshold
IF pixels were claimed in last step

Figure 3.8: Flow chart describing the process that assigns pixels to storms

The input to this part of the algorithm is the original precipitation image and the list of independent storm centers determined in the first part of the algorithm. The first step is to create a threshold list as described above. The independent storm centers are then allowed to claim pixels in a similar manner to that described above but with slight differences. An independent storm center can only claim pixels if the precipitation value of the storm center is above the current threshold and a pixel can only be claimed if it has a precipitation value greater than the current threshold, but it is not necessary for it to be below the previous threshold, as discussed above. Another difference between this part of the algorithm and the part of the algorithm that determines independent storm centers is that now each pixel can only be claimed by one storm; there is no overlap allowed between independent storm regions. Since multiple storms are not allowed to claim the same pixels, it is important to have a good method for the order in which pixels can be claimed. This is done by growing each storm center one at a time, where each storm that is above the current threshold is allowed a chance to claim the pixels within its eight-connected neighborhood before allowing any other storm to claim pixels that are further away from itself, such as within the sixteen and twenty-four connected neighborhoods.

Figures 3.9 and 3.10 will illustrate the region growing process. Figure 3.9 is an image of one isolated region from within a larger precipitation image, with contour lines at 60, 30, 10 and 1 mm/hour. Figure 3.10 will show the pixel claiming process that occurred as pixels were claimed by the region in Figure 3.9. The claiming starts with the identified independent rain peak, and claims neighboring pixels as they clear the current threshold.

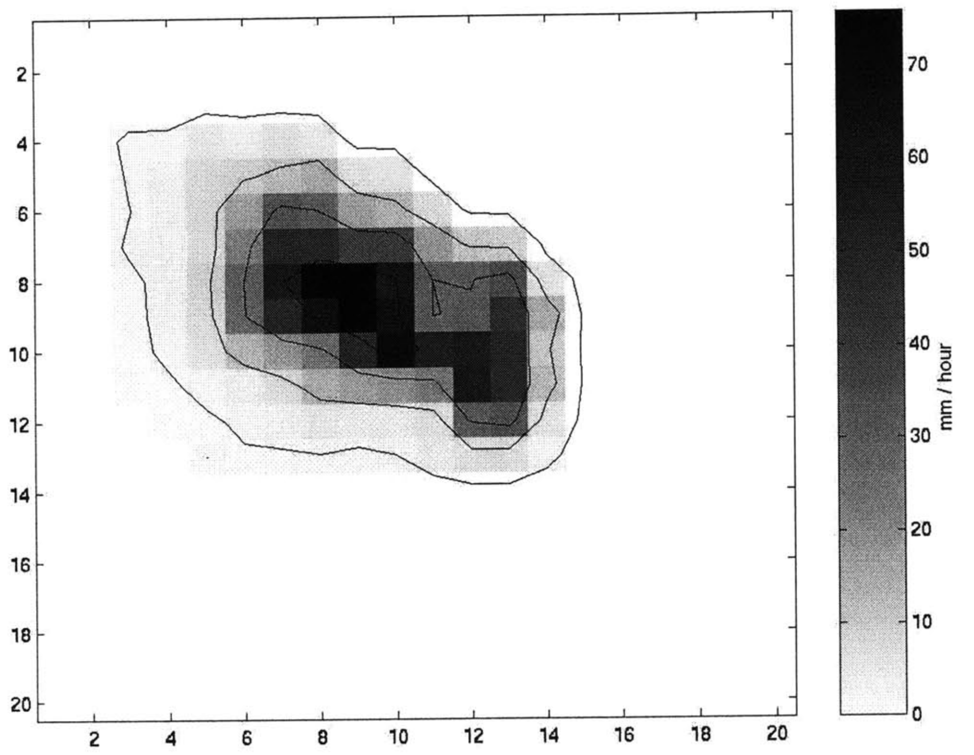

Figure 3.9: Precipitation rates shown within one region

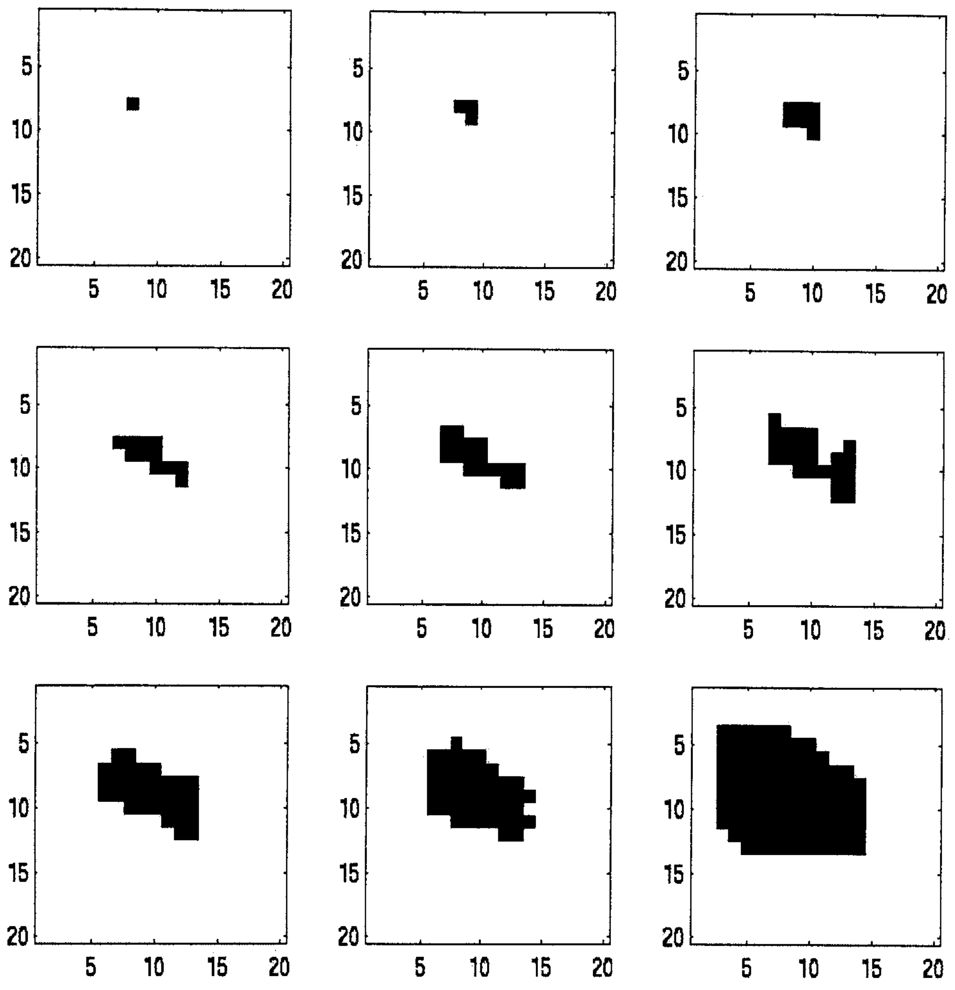

Figure 3.10: Snapshots of region from Figure 3.9 being grown

Another interesting situation occurs when regions aren't isolated by areas of zero precipitation, and instead touch other regions. An example of this situation, where four regions are touching, will be illustrated in Figures 3.11 and 3.12. Figure 3.11 shows the precipitation values of the area in question, with contour lines at 5, 3 and 1 mm/hour, and Figure 3.12 shows the four regions being grown one step at a time.

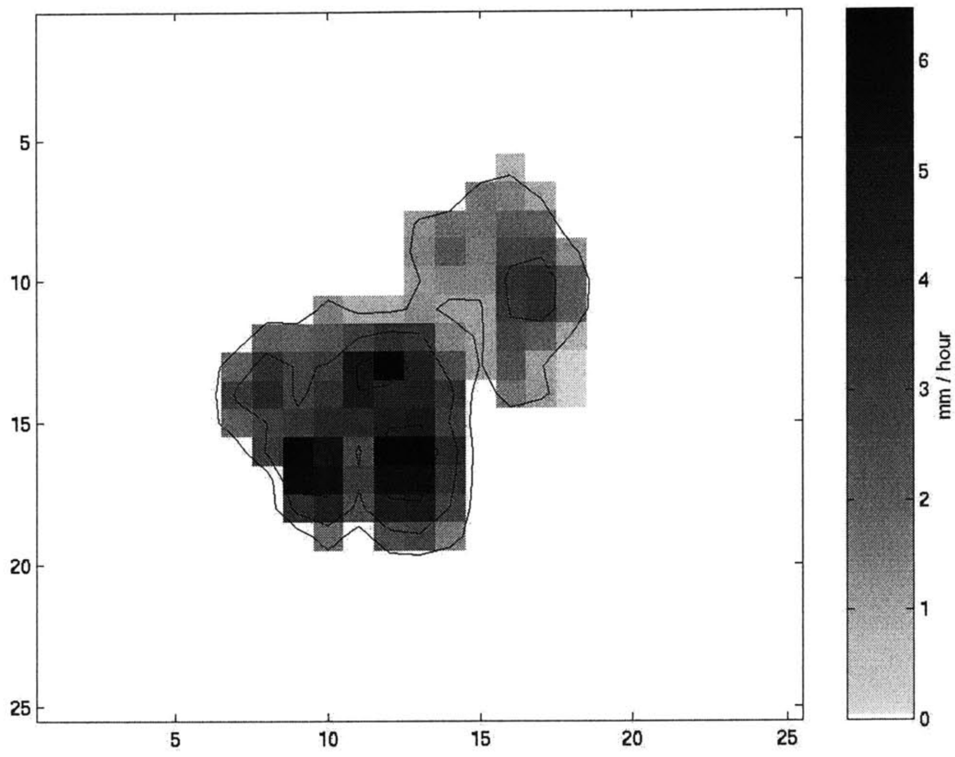

Figure 3.11: Precipitation rates shown within what will be four regions

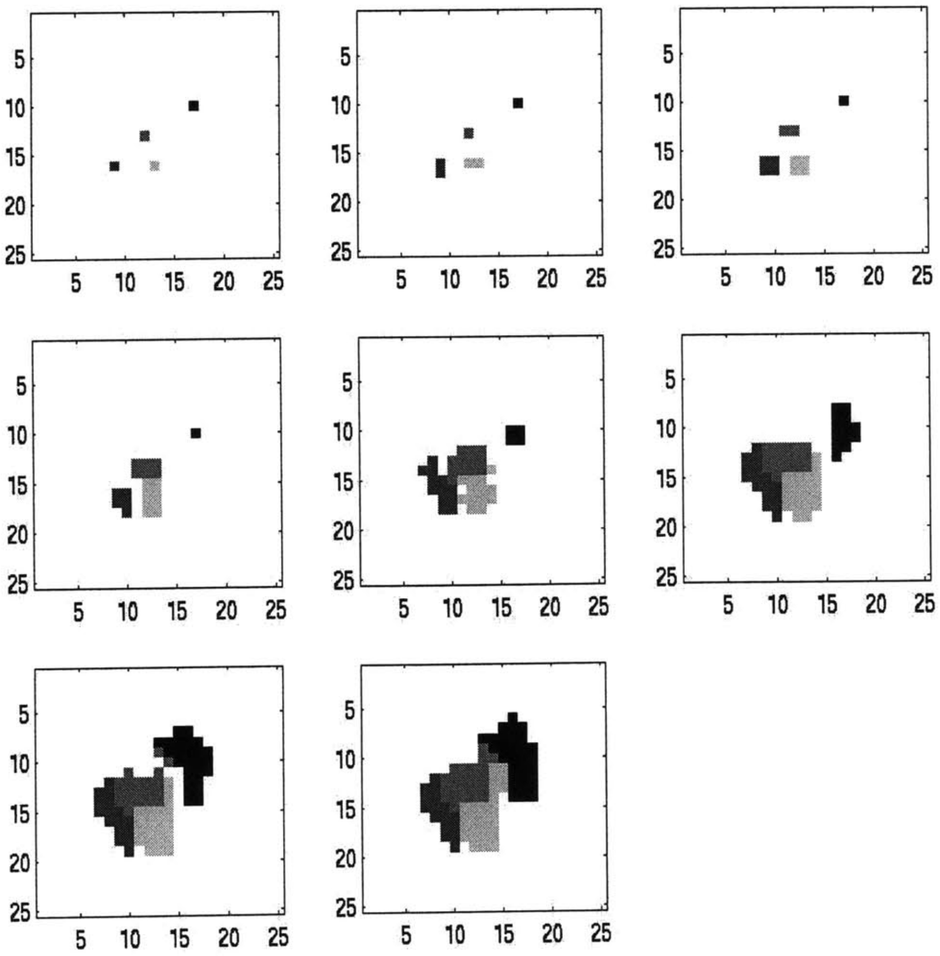
Figure 3.12: Snapshots of the four regions from Figure 3.11 being grown

Once all the pixels with rain rates above 0.05 mm/hour are claimed by one of the storm centers that have been identified to be independent, the regions have been defined. Then the program goes through to check for pixels that are isolated and may not have been assigned to a storm. The resulting output of the algorithm is a list of which pixels of the precipitation image are associated with each independent storm. The segmentation of the original precipitation image shown in Figure 3.5 into storm regions is shown in Figure 3.13.

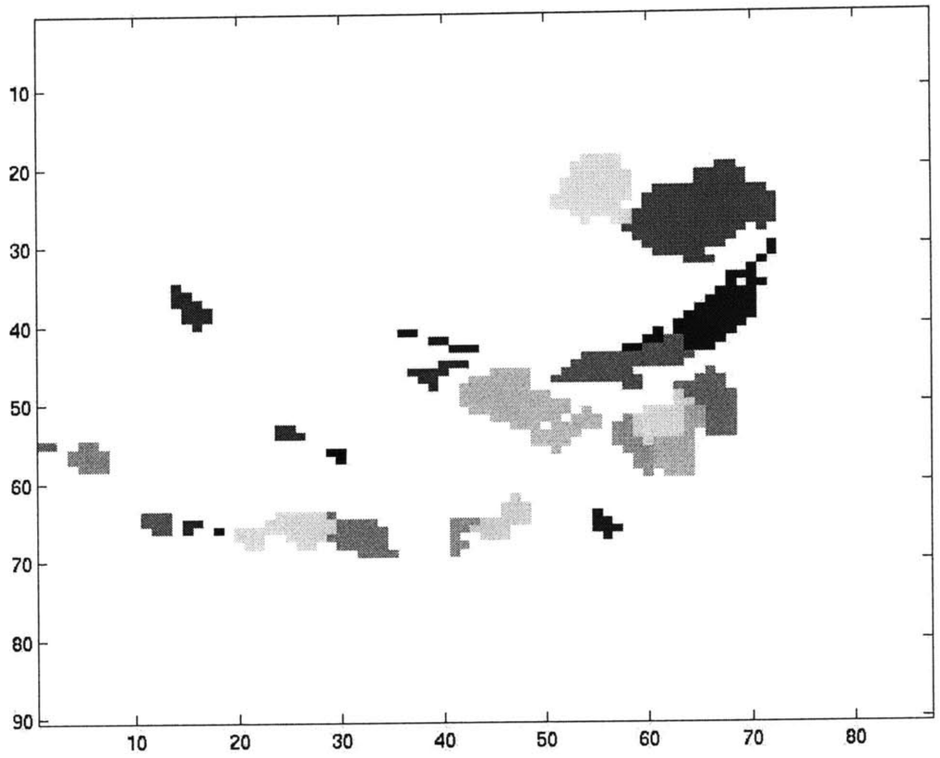

Figure 3.13: Segmentation of precipitation image shown in Figure 3.5

The overall effectiveness of the storm segmentation algorithm will be discussed in Section 4.1.

Section 3.3: Retrieving Integrated Precipitation Rate

Once the data has been segmented into separate storms, the next step to finding an accurate estimate of the integrated precipitation rate is to create, train and validate a neural network to estimate the integrated precipitation rate over each storm region. Over the 38 orbits of AMSU data, 591 storms were identified. These 591 storms were randomly assigned to three data sets for training, validation, and testing. One half of the storms were used for training, and one quarter of the storms were used for each of validation and testing.

The inputs to the neural network are vectors that contain selected parameters of the storm regions. The target values that will be used in training are the logarithm of the NEXRAD estimate of total rain volume within each storm region, in units of mm/hour over each 15-km resolution cell, plus 1 mm/hour. To determine the most effective neural network and set of inputs, the structure of the neural network, the number of inputs, and the combination of inputs were varied, these different networks were trained, and the results were compared.

Eight different parameters of a storm region were selected to be examined as possible inputs to the neural network. These parameters were: the initial estimate of the integrated precipitation rate in units of mm/hour over each 15-km diameter cell, the peak rate over the storm region, the area of the storm region, the average precipitation rate of the storm region, the number of pixels in the storm region that are precipitating at a rate above half the peak rate, the number of pixels in the storm region that are precipitating at a rate above the average rain rate, the number of pixels in the storm region that are precipitating at a rate above a chosen threshold (such as 1 mm/hour), and the ratio of the major and minor axes of an ellipse that has been fitted to the storm area. These parameters were chosen to capture all the features that describe a rain storm and affect the integrated precipitation rate, such as size of the storm region, shape of the storm region, and the intensity of the precipitation within the storm region. A program was written to take as input a rain image and a cell array that lists the pixels associated with each storm region and then output a cell array that contains the parameters of each storm region.

The neural network structure was determined by creating a set of neural networks with different structures, training the set of networks using the same combination of the eight input parameters and targets, and then comparing the results to determine the effectiveness of each network. All the neural networks were feed-forward networks, but they differed by having different numbers of neurons in the input layer, different transfer functions for the neurons, and different numbers of layers. These tests determined that an increase in the complexity of the neural network did not result in a better estimate of the integrated precipitation rate. The neural network that was chosen has five inputs, two neurons in the input layer, and one neuron in the output layer. The neurons in the input layer were chosen to have log sigmoid transfer functions, and the neuron in the output layer was chosen to have a linear transfer function.

Once the best structure for the neural network was determined, it was possible to compare different combinations of inputs to determine the effectiveness of the estimate when the neural network has different combinations of parameters of the storm as the inputs. This was done by creating a set of neural networks with the optimal structure described above, creating different combinations of the eight input parameters, training each network in the same way but with the different sets of inputs, and comparing the results to see which inputs are most effective. The most effective number of inputs was determined to be five out of the eight parameters, and these inputs were found to be the initial estimate of the integrated precipitation rate, the peak rate over the storm region, the area of the storm region, the number of pixels in the storm region that are precipitating at a rate above half the peak rate, and the number of pixels in the storm region that are precipitating at a rate above the average rain rate.

After these tests determined the best structure and set of inputs for the neural network, the next step was to train and validate a network, use the network to retrieve the integrated precipitation rate for the test data set, and compare the results to the integrated precipitation rates found by NEXRAD. The neural networks were trained to minimize the mean-squared-error between the result and the target using Levenberg-Marquardt backpropagation. The resulting layer weights are listed in the two tables that follow. Table 3.2 contains the weights from the inputs to the first layer, and Table 3.3 contains the weights from the output of the neurons in the first layer to the output layer.

Node	estimate of int. rate (mm/hour over area)	peak rate (mm/hour)	area in # pixels	# pixels > ½ peak rate	# pixels > avg. rate	bias
A1	-5388.884	5386.0714	398.3898	4527.2185	163.0929	-1482.9558
A2	0.5727	-0.2928	0.0070	-0.1144	-0.2314	0.0218

Table 3.2: Weights and biases from inputs to first layer of neurons

Node	A1	A2	bias
B1	0.7227	16.4931	-8.2540

Table 3.3: Weights and bias from the first layer to the output layer

After the neural network is used to retrieve a better estimate of the integrated precipitation rate, the logarithm of the estimate in units of mm/hour over a 15-km diameter cell is converted to more reasonable units of m^3/second. The units of m^3/second take into account the rate of precipitation and the area of the storm, for a measure of the total volume of precipitation. The results of the neural network training will be discussed in Section 4.2, and the overall effectiveness of the storm-wide precipitation retrievals will be discussed in Section 4.3.

The final step of the neural network estimation is to use the estimates of total volume of precipitation over each storm to adjust the original 15-km resolution AMSU data. This is done by taking the ratio of the storm-wide precipitation retrieval algorithm estimate of integrated precipitation rate and the original AMSU estimate of integrated precipitation rate. Then each of the pixels associated with the region of a particular storm is multiplied by the ratio associated with that storm. This adjusts the precipitation value of each pixel according to whether the storm-wide precipitation retrieval estimate of integrated precipitation rate is higher or lower than the original AMSU estimate of integrated precipitation rate.

The results of the adjustment of the 15-km resolution AMSU data by using a ratio of storm-wide precipitation retrieval estimates of integrated precipitation rate to AMSU estimates of integrated precipitation rate will be discussed in Section 4.4.

Chapter Four: Results

There are a number of issues to consider when analyzing the results of the storm-wide precipitation retrieval algorithm. The four most relevant issues are: (1) how effective was the storm segmentation, (2) how well does the neural network perform, (3) how well does the neural network storm-wide estimate of integrated precipitation rate agree with the NEXRAD estimate, relative to the agreement between the original AMSU and NEXRAD estimates of integrated precipitation rate, and (4) how well does the adjusted 15-km resolution data agree with the NEXRAD data, relative to the agreement between the original 15-km resolution data and the NEXRAD data. These four questions will be addressed one at a time.

Section 4.1: Storm Segmentation

When studying a precipitation image displayed in a gray scale that corresponds to the level of precipitation, an observer may be able to see a natural segmentation of the pixels in the image. However, it is possible that different observers will see different ways in which an image could be segmented. There is no definitive answer to the question of how to segment a particular precipitation image. As a result, we will explore the effectiveness of the storm segmentation method described in Section 3.2.2 by looking at a number of cases, discussing whether or not the resulting segmentation is acceptable and, if it is unacceptable, discussing how it could be improved.

The first case to be discussed is shown in Figure 4.1. The precipitation image in Figure 4.1 has contour lines at 15, 10, 5 and 1 mm/hour.

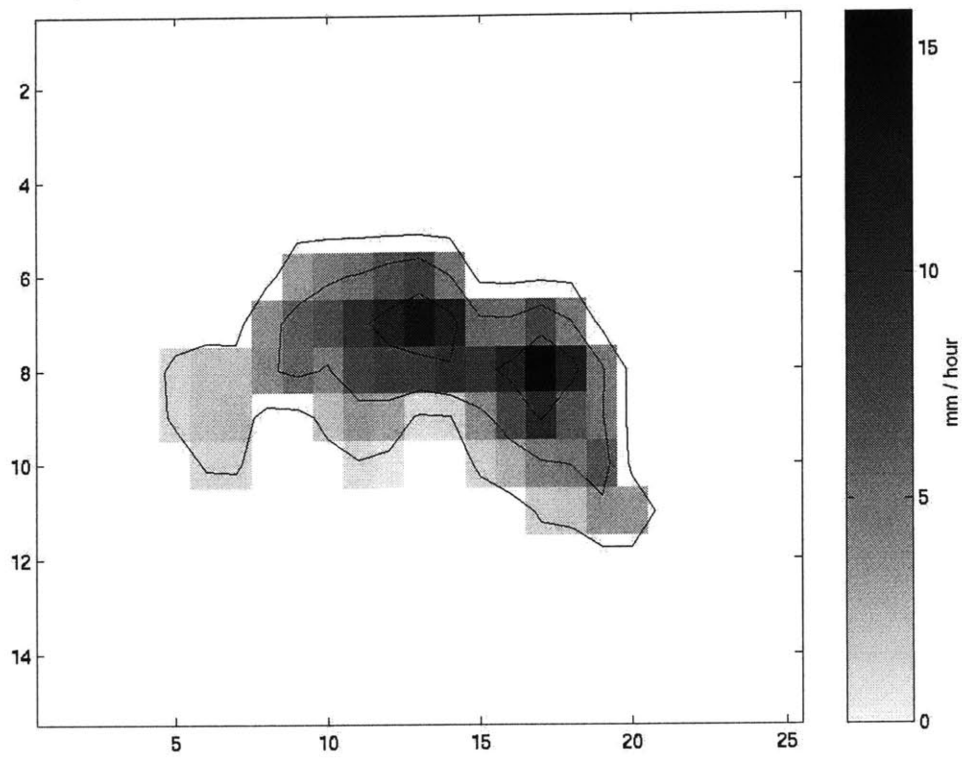

Figure 4.1: Precipitation image

This is a relatively simple precipitation image to segment. There are two points of quite high precipitation relative to the other precipitating pixels in the area, and there are pixels with lower precipitation values in between these points of high precipitation. The two points of high precipitation are found to be storm centers and the algorithm assigned pixels to the two regions as shown in Figure 4.2.

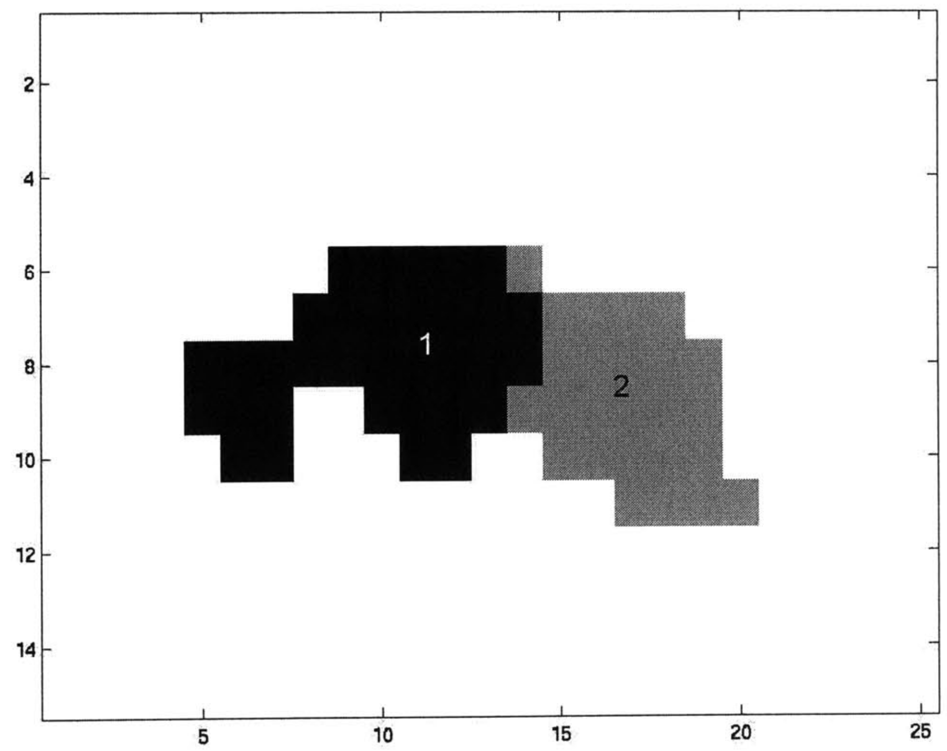

Figure 4.2: Resulting segmentation of precipitation image in Figure 4.1

This is certainly an acceptable segmentation of the precipitation image. The dividing line between the two storm centers follows the lines of pixels with lower precipitation values. The one border pixel from region 2 that is only connected to the rest of region 2 by a corner could have been assigned to region 1, but because it is a single pixel with a precipitation rate of only 5 mm/hour it is not a significant problem.

The next case that will be discussed is shown in Figure 4.3. The precipitation image in Figure 4.3 has contour lines at 15, 10, 5 and 1 mm/hour.

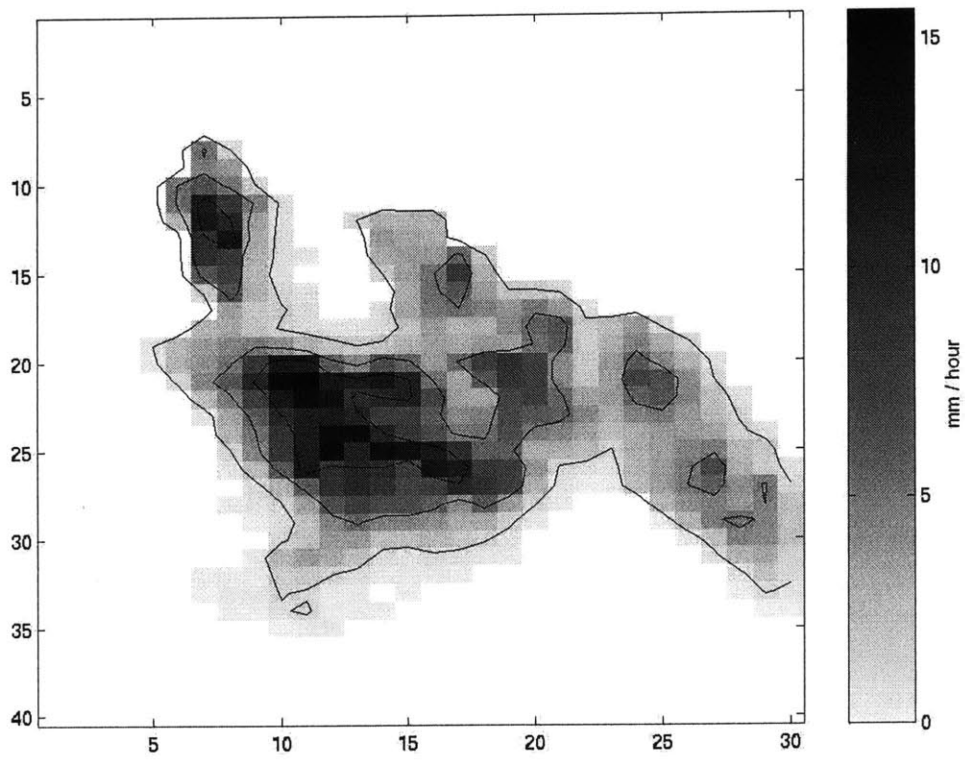

Figure 4.3: Precipitation image

This is a more difficult image to segment because there is more variation in the level of precipitation and there are many more precipitating pixels included in the area. This results in many regional maximums being found to be tested as possible independent storm centers. Six independent storm centers were identified in this image. Pixels were assigned to these six storms as shown in Figure 4.4.

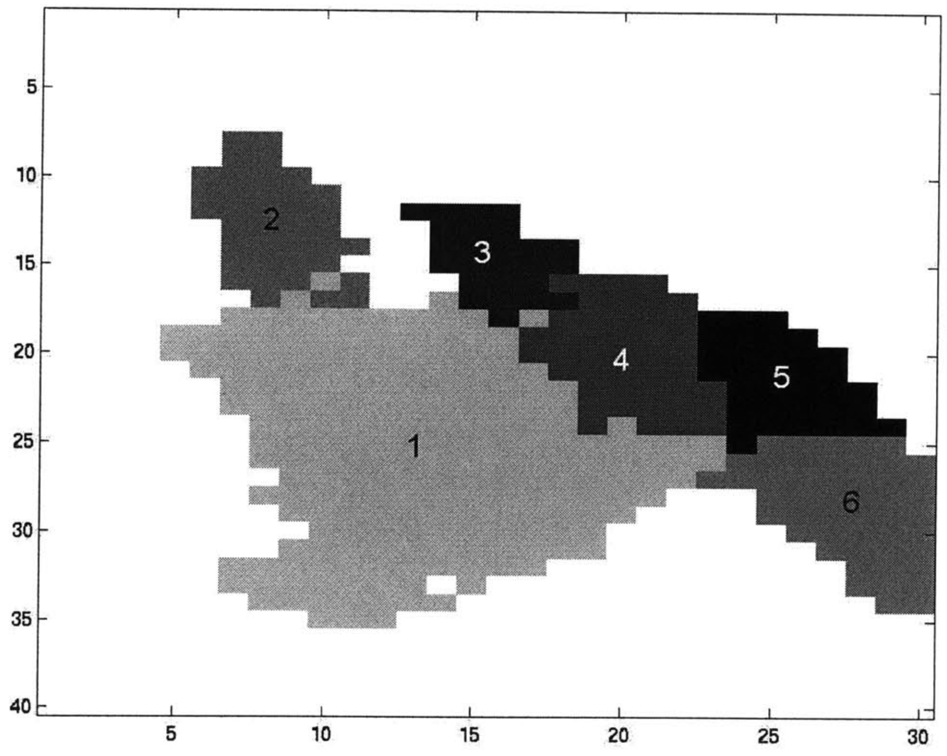

Figure 4.4: Resulting segmentation of precipitation image in Figure 4.3

In this case, the assignment of pixels to storm regions results in an acceptable segmentation of the image. Region 1 contains the pixels associated with the cluster heavily precipitating points. As expected, this is the largest storm region defined within the image. The dividing lines between each of the regions follow the lines of lower precipitation, as desired. Noting the division between the regions 1 and 2, and regions 1 and 3, it would be possible to assign the pixels on the edges between the regions slightly differently by not allowing pixels to be touching the rest of the region they belong to by only a corner. However, this is not a significant problem since it is only a few of pixels with very low precipitation rates of approximately 2 mm/hour.

The next case that will be discussed is shown in Figure 4.5. The precipitation image in Figure 4.5 has contour lines at 10, 5 and 1 mm/hour.

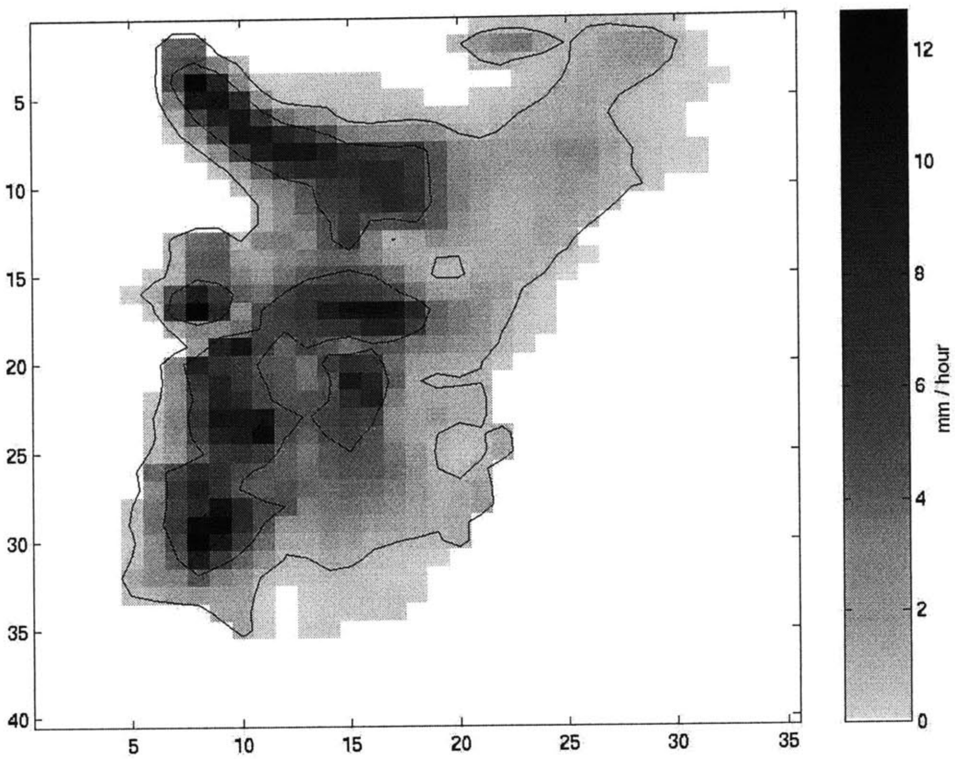

Figure 4.5: Precipitation Image

This is also a difficult image to segment. There is a great deal of variation within the levels of precipitation and there are a number of areas with comparatively heavy precipitation. Eight independent storms were identified in this area, and pixels were assigned to these regions as shown in Figure 4.6.

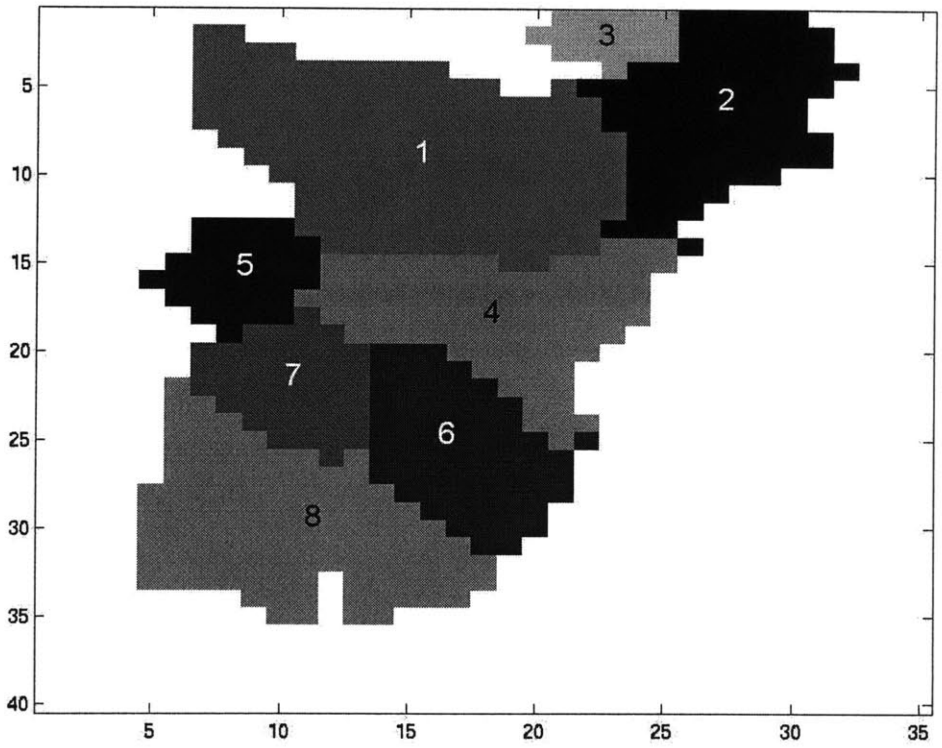

Figure 4.6: Resulting segmentation of precipitation image in Figure 4.5

This is a very good segmentation of the precipitation image. For example, looking at the pixels associated with region 1 it can be seen that the region closely follows the shape of the storm. Comparing each of the regions defined in Figure 4.6 to the same pixels in Figure 4.5, it can be seen that in each case the regions have properly been defined as an independent storm and that the dividing lines follow the areas of lower precipitation between the storm centers.

The final case that will be discussed is shown in Figure 4.7. The precipitation image in Figure 4.7 has contour lines at 20, 15, 10, 5 and 1 mm/hour.

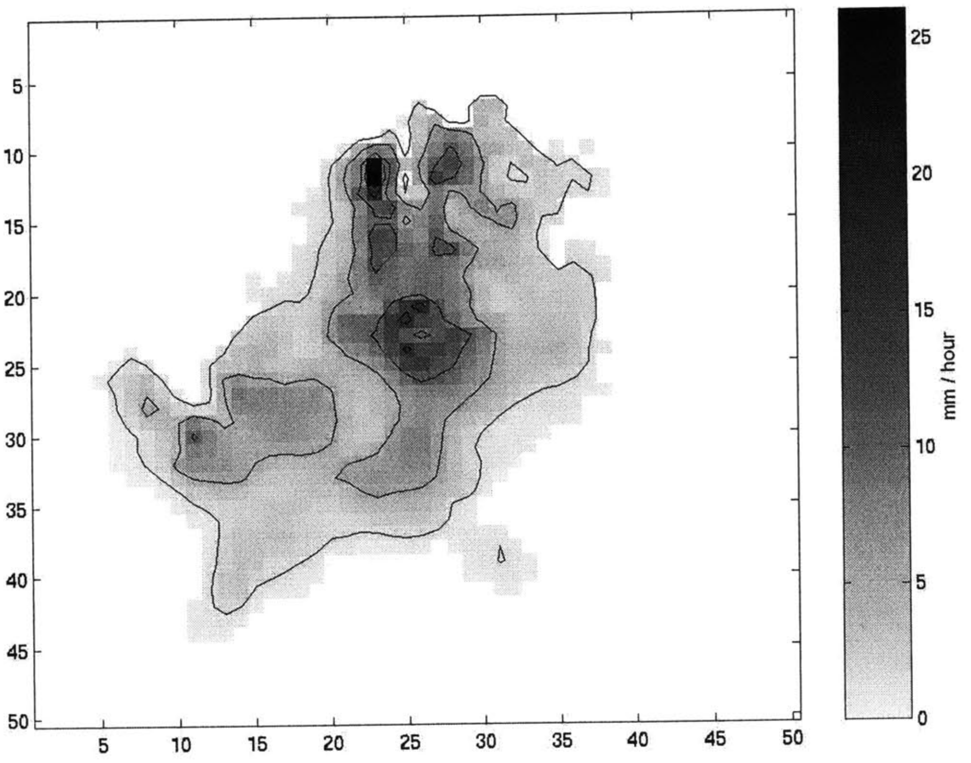

Figure 4.7: Precipitation image

This is a difficult image to segment because of the many variations in intensity of precipitation within the area, and the large number of pixels that have smaller precipitation values. As discussed above, having a great deal of variation results in many regional maximums, which may in turn become many independent storm centers. Many independent storm centers result in many regions having to be defined. Having many pixels at a small precipitation value means that many pixels will be assigned at the same time, which can make it more difficult to have good dividing lines between regions.

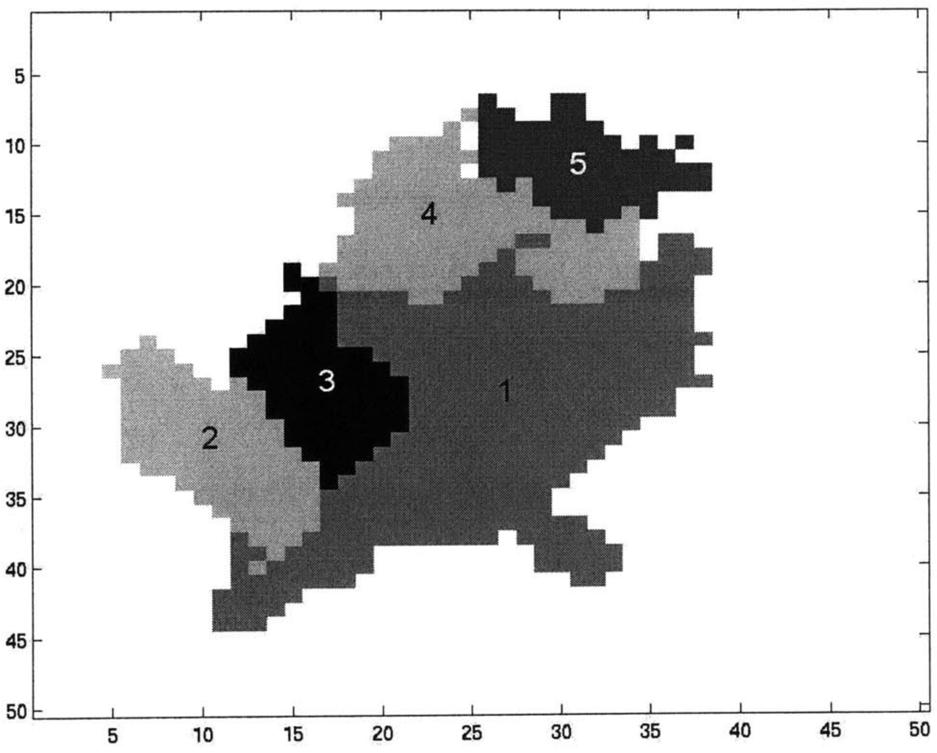

Figure 4.8: Resulting segmentation of precipitation image in Figure 4.7

In this case, the assignment of pixels to storm regions is somewhat unexpected and not particularly desirable. The regions 2, 3, and 5 seem to have been assigned reasonably, but regions 1 and 4 are not well assigned. The region 4 is almost in two parts, with a narrow neck connecting the two parts. The region 1 seems drawn out along the edge of the precipitating area, and seems to reach into areas associated with other regions. In this case the regions do not seem to follow the shapes of the storms very well at all.

However, even though the algorithm does not do very well, it is still close to an acceptable segmentation. The identifiable independent storm centers are in separate regions and the more heavily precipitating pixels near each storm center are assigned to the region of the storm center. The major problem with this segmentation is in the assignment of the pixels which are far from any storm center, and thus less obviously assignable to one region or another. In this case, it would seem to make more sense to assign the pixels in the lower left-hand corner of region 1 to region 2 instead, and assign the pixels in region 1 that make a hook into region 4 to the region 4 instead. These changes would greatly improve the segmentation.

As can be seen from the four cases that were discussed, the algorithm does quite well in some difficult cases, but certainly can have problems in other difficult cases. The two biggest difficulties seem to be in situations when the shapes of the storms aren't particularly clear or when the algorithm is trying to determine which pixels should be associated with each storm along region borders. Luckily these difficulties, while unfortunate, are not catastrophic. When the shapes of the storms are not clear and the borders are hazy, it would seem that any algorithm, or human observer for that matter, would have difficulty assigning pixels to a particular storm. Some thoughts for the further improvement of the storm segmentation algorithm will be discussed in Section 5.2.1.

Section 4.2: Effectiveness of Neural Network

The network was trained to minimize the mean-square-error between the target and the result using Levenberg-Marquardt backpropagation, where the training, testing and validation data consisted of 38 orbits of coincident AMSU and NEXRAD rain data over the eastern United States. The target is the logarithm (base 10) of the NEXRAD estimate of integrated precipitation rate + 1 mm/hour over a 15-km diameter cell, and the result is the neural network estimate of the logarithm (base 10) of the integrated precipitation rate + 1 mm/hour over a 15-km diameter cell. The total mean-square-error between the logarithm of the original estimate of the integrated precipitation rate and the NEXRAD estimate of integrated precipitation rate over the 591 storms that were identified from the 38 orbits, where both estimates are in units of mm/hour over a 15-km diameter cell, is 0.3977. After using a neural network to retrieve a better estimate, the mean-square error between the logarithm of the new estimate and the logarithm of the NEXRAD estimate, also both in units of mm/hour over a 15-km diameter cell, is 0.1906.

Therefore, in the units of mm/hour over a 15-km diameter cell, the neural network had decreased the error between the logarithms by approximately a factor of two. This is certainly a good result, and represents a significant improvement.

Section 4.3: Comparison with NEXRAD and Original Results

The best way to determine the effectiveness of the entire storm-wide precipitation retrieval algorithm is to compare the AMSU and storm-wide precipitation retrieval algorithm estimates of integrated precipitation rate over each storm to the NEXRAD estimates of integrated precipitation rate over each storm. The NEXRAD estimate of integrated precipitation rate is being taken as the true measurement of precipitation to be used to compare the original and final results. Note that the NEXRAD data set used contained both near and far points, where near points are beyond 30 km but within 110 km of a NEXRAD radar site and far points are beyond 110 from a NEXRAD radar site.

As a result the NEXRAD estimates are not as accurate as they would be if the data set contained only near points.

The first step in the comparison is to change the units to m^3/second. The neural network estimates the logarithm of the integrated precipitation rate in units of mm/hour over a 15-km diameter cell, so it is a simple matter to convert the values to m^3/second by reversing the logarithm, multiplying by the area, and making corrections for changing kilometers and millimeters to meters, and changing hours to seconds.

Once the units have been converted, the AMSU and storm-wide precipitation retrieval algorithm estimates of integrated precipitation rate over each storm can be plotted versus the NEXRAD estimates of integrated precipitation rate over each storm. This is an effective method for comparing the different values because the better the estimate, the closer the points will be to the line at a forty-five degree angle that indicates that the estimate and the NEXRAD value are equal. The next two figures compare the AMSU and storm-wide precipitation retrieval algorithm estimates of integrated precipitation rate with the NEXRAD estimates of integrated precipitation rate over each storm. All units are in m^3/second, and both the x- and y-axes have a logarithmic scale. The different shapes of the points on the plot refer to the data set in which each point was included; the three data sets were training data, validation data, and testing data.

Figure 4.9 is a scatter plot of the original AMSU estimate of integrated precipitation rate over each storm versus the NEXRAD estimate of integrated precipitation rate over each storm.

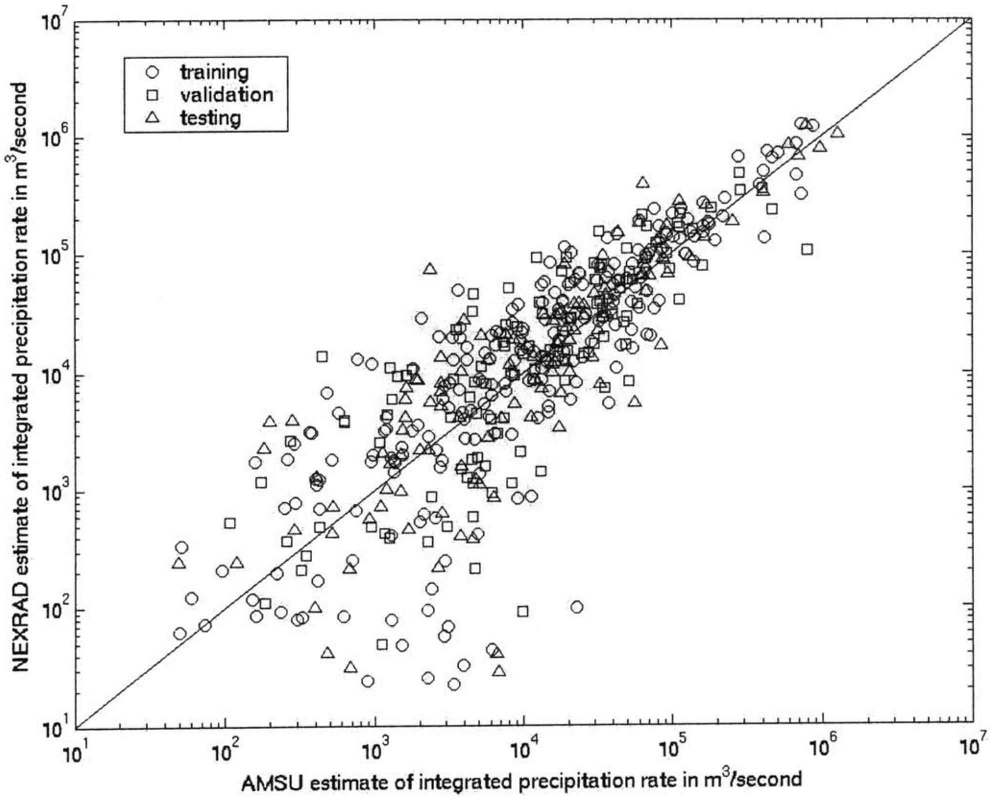

Figure 4.9: loglog plot of estimates of integrated precipitation rate, AMSU vs. NEXRAD

Note that in this figure the general trend of the data follows the line that indicates that the AMSU and NEXRAD values are equal, but that there are many points that are significantly far away from this line.

Figure 4.10 is a scatter plot of the storm-wide precipitation retrieval algorithm estimate of integrated precipitation rate over each storm versus the NEXRAD estimate of integrated precipitation rate over each storm.

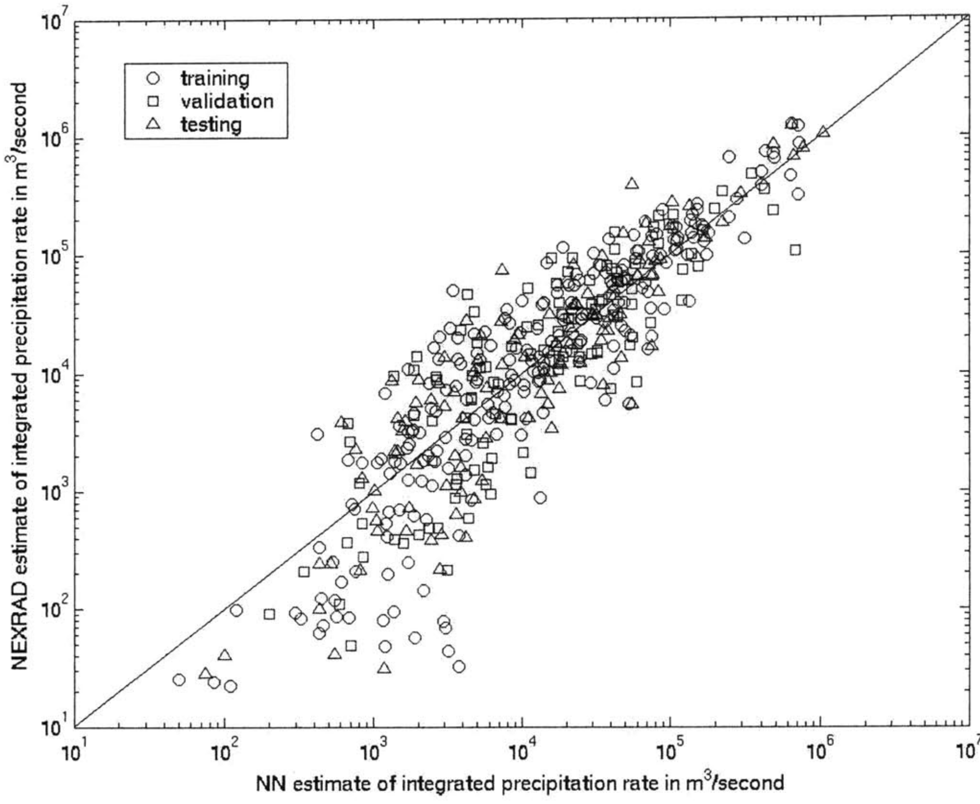

Figure 4.10: loglog plot of estimates of integrated precipitation rate, Neural Network vs. NEXRAD

Note that in this figure the data is more tightly grouped along the line that indicates equality of the storm-wide precipitation retrieval estimate and NEXRAD values. Some of the points are still spread out from the line, but in general the storm-wide precipitation retrieval algorithm increases the accuracy of the estimate, and pulls the points closer to the true value.

The second method for comparing the AMSU and storm-wide precipitation retrieval algorithm estimates of integrated precipitation rate is to find the mean-square-error and root-mean-square discrepancy between the logarithm (base 10) of each of the estimates and the logarithm (base 10) of the NEXRAD estimates of integrated precipitation rate over each storm. This is an effective method of comparison because it makes it possible to quantify the improvement gained by using the storm-wide precipitation retrieval algorithm.

Over the 591 identified storms, the total mean-square-error between the logarithm of the original AMSU estimate of integrated precipitation rate and the logarithm of the NEXRAD estimate, both measurements in units of m^3/second, is 0.33 and the root-mean-square discrepancy is 0.57. The total mean-square-error between the logarithm of the storm-wide precipitation retrieval algorithm estimate of integrated precipitation rate, in units of m^3/second, and the logarithm of the NEXRAD estimate, in units of m^3/second, is 0.19 and the root-mean-square discrepancy is 0.44. The storm-wide precipitation retrieval algorithm improves the accuracy of the estimate of integrated precipitation rate by decreasing the mean-square-error by nearly a factor of two and by decreasing the root-mean-square discrepancy.

A third method for comparing the AMSU and storm-wide precipitation retrieval algorithm estimates of integrated precipitation rate is to find the magnitude of the discrepancy between the logarithm of each estimate and the logarithm of the NEXRAD estimate for each storm; i.e. $|\log (\text{NEXRAD/AMSU})|$. These discrepancies are characterized in two histograms in Figure 4.11.

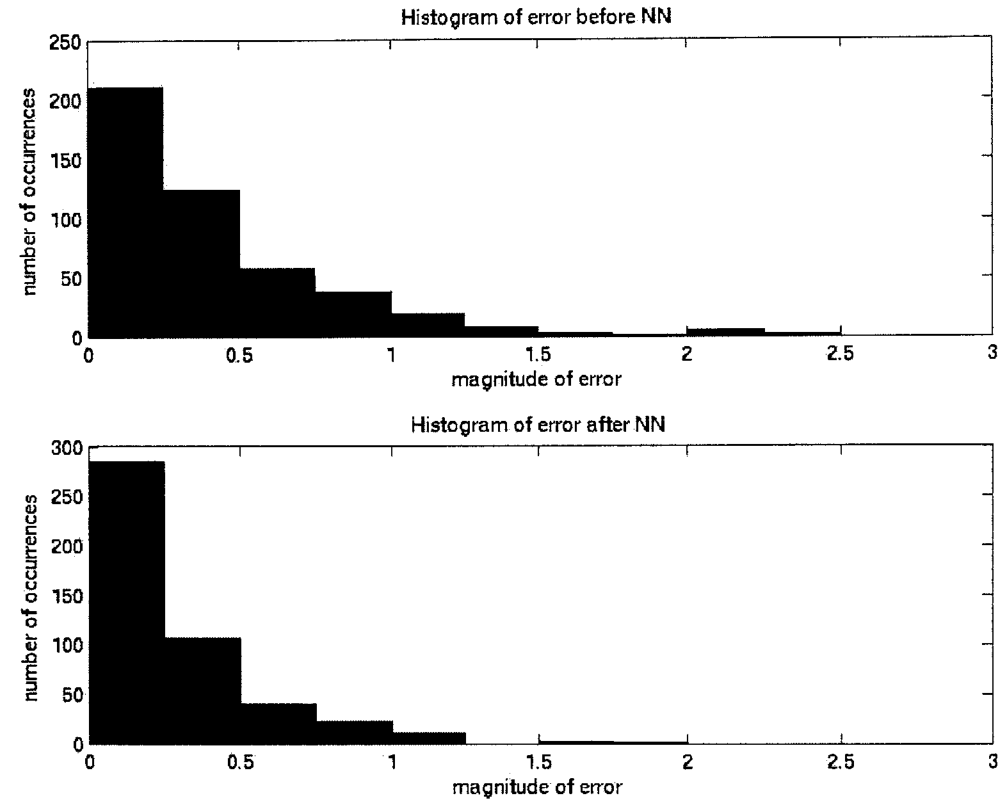

Figure 4.11: Histogram of error between AMSU and Neural Network estimates and NEXRAD, where the discrepancy is characterized by $|\log (\text{NEXRAD/AMSU})|$

The top histogram in Figure 4.11 shows the number of occurrences of each magnitude of error between the logarithm of the AMSU estimate of integrated precipitation rate and the logarithm of NEXRAD estimate of integrated precipitation rate. The bottom histogram in Figure 4.11 shows the number of occurrences of each magnitude of error between the logarithm of the storm-wide precipitation estimate of integrated precipitation rate and the logarithm of NEXRAD estimate of integrated precipitation rate. It can be seen by comparing the two histograms that the storm-wide precipitation retrieval algorithm decreases the occurrences of errors of magnitude greater than 0.25, and increases the number of occurrences of errors of magnitude less than 0.25. It is a good result that the number of larger magnitude errors is decreased, and the number of smaller magnitude errors is increased.

All three of the methods discussed indicate that the storm-wide precipitation retrieval algorithm improves the AMSU estimate of integrated precipitation rate. Possible improvements to the storm-wide precipitation retrieval algorithm and other future work will be discussed in Chapter 5.

Section 4.4: Comparison of 15-km resolution data with NEXRAD

The final issue to consider is how much improvement the storm-wide precipitation retrieval algorithm can make to the 15-km resolution AMSU retrievals. The 15-km resolution retrievals are adjusted by multiplying each pixel within a defined storm region by the ratio of the storm-wide precipitation retrieval algorithm estimate of integrated precipitation rate to the original AMSU estimate of integrated precipitation rate. This adjusts the precipitation value of each pixel according to whether the storm-wide precipitation retrieval estimate of integrated precipitation rate is higher or lower than the original AMSU estimate of integrated precipitation rate.

The effectiveness of the adjustment can be measured by finding the discrepancy between the original AMSU 15-km resolution data and NEXRAD data, and comparing it to the discrepancy between the adjusted AMSU 15-km resolution data and NEXRAD. Only pixels that correspond to a location that is beyond 30 km but within 110 km of a NEXRAD radar site were chosen to be included in the error calculation, as it has been determined that that within this distance range that NEXRAD provides the most accurate estimate of precipitation. Also, only AMSU pixels with an estimated precipitation rate above 0.05 mm/hour were used in the error calculation, as it has been determined that 0.05 mm/hour is the nominal threshold of detection.

The mean-square-discrepancy was calculated by finding the difference between the logarithm of the AMSU or adjusted retrieval plus 0.1 mm/hour and the logarithm of the NEXRAD estimate plus 0.1 mm/hour, and computing the mean-square discrepancy of the error vector. The mean-square-discrepancy over the 12,117 pixels that have been identified as precipitating in the original AMSU 15-km resolution precipitation retrievals is 0.4509. The mean-square-discrepancy in the adjusted 15-km resolution precipitation retrievals is 0.2445. This is a good result, showing that the storm-wide precipitation retrieval algorithm not only significantly improved the estimate of the integrated precipitation rate over each storm, but also the associated adjustment decreases the mean-square-discrepancy over all precipitating 15-km resolution pixels by almost a factor of two.

The root-mean-square discrepancy was calculated by finding the difference between the AMSU or adjusted AMSU precipitation rate and the NEXRAD precipitation rate for each pixel, and computing the root-mean-square discrepancy. The root-mean-square discrepancy over the 12,117 pixels that have been identified as precipitating in the original AMSU 15-km resolution precipitation retrievals is 10.5 mm/hour and the root-mean-square discrepancy in the adjusted 15-km resolution precipitation retrievals is 6.8 mm/hour. This is a reasonable improvement of a factor of nearly one and a half.

A second way to quantify the effectiveness of the adjustment is to group the precipitation by value, and compare the root-mean-square discrepancy between the logarithms of precipitation rates and between precipitation rates. The AMSU 15-km resolution data and the adjusted 15-km resolution data are broken up into nine precipitation ranges based on the NEXRAD estimate of precipitation rate for each pixel in mm/hour: < 0.5, 0.5-1, 1-2, 2-4, 4-8, 8-16, 16-32, 32-64, and > 64. Then the mean-square discrepancy and the root-mean-square discrepancy are found within each range, as listed in Tables 4.1 and 4.2.

Ranges of precipitation rates	RMS in AMSU retrieval (logarithm(NEX/AMSU))	RMS in adjusted retrieval (logarithm(NEX/adjusted))
below 0.5 mm/hour	0.54	0.56
between 0.5 and 1 mm/hour	0.72	0.50
between 1 and 2 mm/hour	0.78	0.53
between 2 and 4 mm/hour	0.72	0.49
between 4 and 8 mm/hour	0.59	0.45
between 8 and 16 mm/hour	0.67	0.48
between 16 and 32 mm/hour	0.58	0.43
between 32 and 64 mm/hour	0.74	0.58
above 64 mm/hour	0.87	0.65

Table 4.1: Comparison of root-mean-square discrepancy of logarithm of AMSU and adjusted retrievals

The adjusted 15-km resolution data has less root-mean-square discrepancy in the logarithm of precipitation rates for nearly every range of precipitation values. In the lowest range of precipitation rates the root-mean-square discrepancy is actually slightly increased, but in every other range the improvement is quite good.

Ranges of precipitation rates	RMS in AMSU retrieval (mm/hour)	RMS in adjusted retrieval (mm/hour)
below 0.5 mm/hour	1.2	0.9
between 0.5 and 1 mm/hour	2.1	1.5
between 1 and 2 mm/hour	2.5	1.8
between 2 and 4 mm/hour	3.0	2.6
between 4 and 8 mm/hour	3.9	3.0
between 8 and 16 mm/hour	7.0	5.9
between 16 and 32 mm/hour	21.4	18.6
between 32 and 64 mm/hour	30.4	25.1
above 64 mm/hour	42.3	37.3

Table 4.2: Comparison of root-mean-square discrepancy (mm/hour) between AMSU and adjusted retrievals

The adjusted 15-km resolution data has less root-mean-square-discrepancy than the original 15-km resolution data in every range of precipitation values. This is a satisfactory result because it indicates that the process of adjusting the pixel values decreases the discrepancy for all levels of precipitation.

These methods show that the storm-wide precipitation retrieval algorithm does an acceptable job of adjusting the 15-km resolution retrievals. The mean-square discrepancy and root-mean-square discrepancies are decreased both overall, and in nearly every range of precipitation values. Further discussion of these results and possible improvements to the adjustment of the 15-km resolution precipitation images appear in Chapter 5.

Chapter Five: Discussion and Conclusions

The goal of this research was to develop a computational method that would improve the accuracy of precipitation estimates by including spatial information in the AMSU precipitation retrieval, which had been solely pixel-based. As discussed in Section 4.4, including spatial information to retrieve a better estimate of the integrated precipitation rate, and using this estimate to adjust the precipitation values of the pixels within each storm improved the accuracy of the 15-km resolution precipitation image as compared to NEXRAD. Now that it has been verified that a basic incorporation of spatial information will improve the estimate of integrated precipitation rate as well as allow the 15-km resolution precipitation images to be adjusted, further work in this area is justified.

Section 5.1: Discussion of Problems

A variety of different challenges were encountered in the effort to develop a storm segmentation algorithm and an effective method of estimating integrated precipitation rates. Before beginning a discussion of possible improvements and future work associated with the storm-wide precipitation retrieval algorithm it is important to briefly discuss what causes problems in the process.

Segmenting a precipitation image into independent storms seems to be a simple problem, but in practice turned out to be quite complicated. There are a number of reasons for this complexity. The resulting segmentation of a precipitation image depends heavily on how a storm region is defined and what features are used to determine which storms are independent. These features may include morphology of minima, definition of thresholds, role of distance, sequence of processing, et cetera. This is one factor that make it difficult to develop a storm segmentation algorithm.

Another difficulty in developing a robust segmentation algorithm that divides a precipitation image into separate storms is that storms come in many shapes and sizes. One storm may consist of a circular or elliptical area with heavy precipitation in the center surrounded by a ring of lower levels of precipitation that tail off to zero precipitation as it reaches the edge of the storm, while another storm may be a large area of lighter precipitation with little definite shape or strong storm center. The different possible appearances of storms make it necessary to have an algorithm that relies on a few basic principles that apply to most, if not all, storms to find and isolate them. The features that were chosen to use to identify independent storms for this project were isolation from other heavily precipitating regions by areas of little to no precipitation and a storm center with a higher precipitation value than the pixels surrounding it.

Another problem that was encountered in the storm-wide precipitation retrieval process involved the use of the improved estimates of integrated precipitation rates to adjust the precipitation value of each pixel within a designated storm. There are three cases for the ratio of precipitation rates: (1) the neural network estimate is greater than the original AMSU estimate so the values of the pixels within the storm will be increased, (2) the neural network estimate is less than the original AMSU estimate so the values of the pixels within the storm will be decreased, or (3) the neural network estimate is equal to the original AMSU estimate so the values of the pixels will be unchanged. In the case where the precipitation values within the storm are increased or decreased, the possibility that some of the pixels in the storm region have overestimated the precipitation value and some of the precipitation values have underestimated the precipitation value but the entire storm region either overestimates or underestimates the integrated precipitation is not taken into account. If this situation occurs, the adjustment may actually increase the pixel by pixel error in that storm.

Possible methods to fix each of these problems will be discussed in Section 5.2.

Section 5.2: Future Work

There are three areas for future work in this research project: (1) possible improvements to the storm segmentation algorithm, (2) improvement of the neural network that estimates integrated precipitation rate, and (3) development of a better method to carry out the adjustment of 15-km resolution pixels.

Section 5.2.1: Future Work on Storm Segmentation Algorithm

As mentioned in Section 4.1, the storm segmentation algorithm is effective, but certainly has limitations. One of these limitations is the way in which pixels are sometimes assigned along the edges between two adjoining independent regions. This problem occurs because pixels can be claimed if they are within the eight-connected neighborhoods, where the eight-connected neighborhood refers to the pixels that form a square around the pixel in question, of pixels that have already been claimed by the region. This means that a pixel can be claimed as part of a region when it is only touching the rest of the region by a corner. A possible solution to this problem is to add an additional piece of code to the storm segmentation algorithm that searches for pixels that are associated with one region but that are surrounded by pixels from another region, and then reassigns them to the region that they are surrounded by, where surrounded by is defined as when the pixel has only pixels from another storm region within its four-connected neighborhood. The four-connected neighborhood refers the pixels that are directly above, below, to the right and to the left of the pixel in question.

Another problem that occasionally occurs in the segmentation is when one region will extend much farther than it seems like it should, possibly encroaching on an area that seems to be more naturally associated with a different storm region. This is a more difficult problem to solve because there are two possible explanations to the problem: either the initial conception of the storm segmentation algorithm is flawed in some way, and further information needs to be considered in the way that the regions are assigned to each storm, or the situations where this seems to occur are actually handled correctly, and it is simply difficult for a human observer to see that the segmentation is correct. More exploration of this problem will be necessary to determine exactly what is occurring. Once this has been determined, it may be possible to find a better solution such as including a distance metric which would consider the strength of the storm center and the pixels distance from the storm center within the process of assigning pixels to storm.

Section 5.2.2: Future Work on Neural Network

The neural networks used in this project utilized some of the simplest measures of the storm region, such as peak rate within the storm and the number of pixels with precipitation rates above average rate within the storm, as inputs. It may be possible to increase the accuracy of the estimate of integrated precipitation rate by including different and possibly more complex measures of storms. One example would be to include the area of the storm as seen by different microwave frequencies or the ratio of areas of the storm as seen by different microwave frequencies [8]. This will give an idea of the state of the storm, whether it is a new storm with increasing precipitation, a storm at its peak precipitation, or a spent storm with decreasing precipitation. The knowledge of what state a storm is in is likely to improve the accuracy of the retrieval.

Another possible improvement to the estimate of integrated precipitation rate would be to train, validate and test the neural network with only NEXRAD data points that are near, which means they are beyond 30 km but within 110 km of a NEXRAD radar site. For this project points beyond 30 km but within 220 km of a NEXRAD radar site were used to train the neural network to estimate integrated precipitation rate. Because NEXRAD data is most accurate beyond 30 km but within 110 km of a radar site, this means that the neural network was not trained on the most accurate data. If a larger data set with only near data points could be compiled it may be possible to significantly improve the accuracy of the estimate of integrated precipitation rate which may in turn significantly increase the accuracy of the adjusted 15-km resolution precipitation estimates.

Section 5.2.3: Future Work on Adjustment Method

As mentioned in Section 5.1, there are certain cases where using a blind adjustment algorithm may actually increase the pixel-by-pixel error of the 15-km resolution estimate. This is a very difficult problem to solve because it is not possible to determine whether doing a blind adjustment will decrease or increase the discrepancy in the 15-km resolution data if there is not ground truth to which the data can be compared. One possible way to explore this problem would be to select a test set and determine how often, and possibly in what conditions, these cases occurs. If it is found to be a significant problem, further exploration of possible solutions can be addressed at that time.

Section 5.3: Conclusions

This project verified that the use of spatial information to segment precipitation images will improve the estimate of integrated precipitation rate as well as allow the 15-km resolution precipitation images to be adjusted so as to decrease the discrepancy between the precipitation estimate and NEXRAD estimates of precipitation rate. This result has justified future work in this area and the possible incorporation of storm-wide precipitation retrieval techniques into all AMSU precipitation retrievals.

References

[1] "NOAA-KLM User's Guide", available at http://www2.ncdc.noaa.gov/docs/klm/index.htm

[2] Chen, Frederick W. and Staelin, David H., "AIRS/AMSU/HSB Precipitation Estimates," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 41, pp. 410-417, Feb. 2003.

[3] Chen, Frederick W., "Global Estimation of Precipitation Using Opaque Microwave Frequency Bands," Cambridge, Massachusetts, Massachusetts Institute of Technology, Ph.D. Thesis, 2004.

[4] Staelin, David H. and Chen, Frederick W., "Precipitation Observations Near 54 and 183 GHz Using the NOAA-15 Satellite," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 38, pp. 2322-2332, Sept. 2000.

[5] Chen, Frederick W., Leckman, Amanda, and Staelin, David H., "Satellite Observations of Polar Precipitation using Aqua," Proc. 7th Conference on Polar Meteorology and Oceanography and Joint Symp. On High-Latitude Climate Variations, Hyannis, MA, May 2003.

[6] Chen, F.W., Leckman, A.M., and Staelin, D.H., "Passive Microwave Signatures of Artic Snowstorms Observed from Satellites," Proc. 2003 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Toulouse, France, July 2003.

[7] Staelin, David H., Personal Communication, 2003.

[8] Leslie, R. Vincent, "Geophysical Parameter Estimation with a Passive Microwave Spectrometer at 54/118/183/425 GHz," Cambridge, Massachusetts, Massachusetts Institute of Technology, Ph.D. Thesis, 2004.

Appendix A: Code

The function peaks_list takes as an input a precipitation image with precipitation rates for each pixel in mm/hour, and outputs a list of the independent storm centers within the precipitation image. The list of independent storm centers identifies the storm centers by precipitation value and pixel location.

```
function peaks_list = peak_finder(rain,noise_thresh,noise,ratio);
% values as arguments ~ noise_thresh, noise, ratio

t0 = clock;

[rows cols] = size(rain);

if exist('noise_thresh') ~= 1
   noise_thresh = 0.05;
end
%values are in mm/hour, peak must be above noise threshold of 0.05 mm/hour to
%be a peak

if exist('noise') ~= 1
   noise = 0.75;
end
%noise value used in dependency calculation

if exist('ratio') ~= 1
   ratio = 0.25;
end
%ratio is the relationship between the height of the valley and the height
%of the smaller of the two peaks that you are comparing, ie when ratio =
%0.25, the valley must be at least 25% of the height of the smaller peak
%ratio can be changed by user to achieve desired segmentation

max_value = ceil(max(max(rain)));
for i = 1:(max_value - 1)
   list(i) = max_value - i;
end

if max_value == 1
   list = [];
end

list = [list 0.05];
%list is used to cycle through the image and assign pixels to areas

rain_peaks = imregionalmax(rain);
```

```
idx = find(rain_peaks);

peak_rates = rain(idx);
peak_rates = double(peak_rates);

tmp_list = sortrows([peak_rates idx],1);
n = length(idx);
for i = 1:n
   peaks_list(i,:) = tmp_list((n+1-i),:);
end
%peaks_list is a list of peak indices and their corresponding
%values in descending order

bad_indices = [];
for i = 1:n
   value = peaks_list(i,1);
   if value > noise_thresh
      %do nothing, retain peak
   else
      bad_indices = [bad_indices;i];
   end
end
peaks_list(bad_indices,:) = [];
%get rid of peaks that are below the noise threshold

num_peaks = size(peaks_list,1);

idx = peaks_list(:,2);

for i = 1:length(idx)
   region_indices{i} = idx(i);
end
%sets up array to for all claimed indices in a region

already_compared = [];
counter = 0;
done = 0;
previous_value = max_value

for g = 1:length(list)
   %outermost loop goes through list, should be no problems here

   current_value = list(g)

   values = peaks_list(:,1);
   valid = find(values >= current_value);
```

```
for h = 1:length(valid)
  indices = region_indices{h};

  for f = 1:length(done)
    temp(f) = (done(f) == h);
    if max(temp)
      tmp = 0;
    else
      tmp = 1;
    end
  end

  while (tmp == 1)
  %this loop takes the indices we have already claimed, claims the
  %eight-connected indices and makes a list of all those indices that are
  %next to the original claimed indices, then checks that list against
  %the original claimed indices to get rid of all the overlap - end up with
  %the indices that are one-pixel out from the original claimed indices

    test_indices = [];
    index = [];

    for i = 1:length(indices)
      index(1) = indices(i) - rows - 1;
      index(2) = indices(i) - rows;
      index(3) = indices(i) - rows + 1;
      index(4) = indices(i) - 1;
      index(5) = indices(i) + 1;
      index(6) = indices(i) + rows - 1;
      index(7) = indices(i) + rows;
      index(8) = indices(i) + rows + 1;

      test_indices = [test_indices;index'];
    end

    used_indices = [];
    temp = [];
    value = rows*cols;
    for i = 1:length(test_indices)
      temp = (test_indices(i) < 1 | test_indices(i) > value);
      if temp
        used_indices = [used_indices,i];
      end
    end
    used_indices = floor(used_indices);
```

```
test_indices(used_indices) = [];
%gets rid of test_indices that are beyond the edges

used_indices = [];
for i = 1:length(test_indices)
   temp = [];
   for j = 1:length(indices)
      temp(j) = (test_indices(i) == indices(j));
   end
   if (max(temp) > 0)
      used_indices = [used_indices;i];
   end
end
used_indices = floor(used_indices);
test_indices(used_indices) = [];

used_indices = [];
for e = 1:(length(test_indices) - 1)
   temp = [];
   for f = (e+1):length(test_indices)
      temp(f) = (test_indices(e) == test_indices(f));
   end
   if max(temp > 0)
      used_indices = [used_indices;e];
   end
end
used_indices = floor(used_indices);
test_indices(used_indices) = [];
% these two loops get rid of test_indices that are already part of indices and
%the ones that appear twice because of overlap

test_values = [];
test_indices = floor(test_indices);
test_values = rain(test_indices);

idx = [];
idx = find(test_values > current_value & test_values < previous_value);

if (isempty(idx))
   tmp = 0;
else
   tmp = 1;
end

indices = [indices;test_indices(idx)];
test_indices = [];
```

```
        end
        region_indices{h} = indices;
    end
    previous_value = current_value;

    %the following loops do comparing stuff
    for j = 1:(length(valid) - 1)
        for k = (j+1):length(valid)
            %these two loops compare all the ridges to each other to
            %determine if any of them overlap

            q = size(already_compared,1);
            for l = 1:q
                %this loop sees if two peaks already been compared
                temp = (already_compared(l,:) == [j k]);
                counter(l) = temp(1) + temp(2);
            end

            if (max(counter) < 2)
                indices_j = region_indices{j};
                indices_j = floor(indices_j);
                region_j = zeros(rows,cols);
                region_j(indices_j) = 1;

                indices_k = region_indices{k};
                indices_k = floor(indices_k);
                region_k = zeros(rows,cols);
                region_k(indices_k) = 1;

                region_overlap = region_j + region_k;
                max_overlap = max(max(region_overlap));

                if (max_overlap > 1)
                    already_compared = [already_compared;[j k]];

                    idx = find(region_overlap == 2);
                    rainy = rain;
                    rainy(idx) = current_value;

                    min_value = current_value;

                    max_peak = peaks_list(j,1);
                    max_peak_idx = peaks_list(j,2);

                    min_peak = peaks_list(k,1);
                    min_peak_idx = peaks_list(k,2);
```

```
          diff = min_peak - min_value;

          if (diff > ratio*min_peak + noise)
            %the two peaks are independent
          else
            %the two peaks are not independent, want to remove
            %smaller peak from consideration
            done = [done;k];
          end
        end
      else
        %already compared, skip
      end
    end
  end
end

%the following loop removes redundancy from done, not a problem
used_indices = [];
for e = 1:(length(done) - 1)
  temp = [];
  for f = (e+1):length(done)
    temp(f) = (done(e) == done(f));
  end
  if max(temp > 0)
    used_indices = [used_indices;e];
  end
end
done(used_indices) = [];

%the following loop finds which initial peaks are peaks
tmp_peaks = [];
tempo = 0;
for i = 1:num_peaks
  for j = 1:length(done)
    tempo(j) = (done(j) == i);
  end

  if (max(tempo) < 1)
    tmp_peaks = [tmp_peaks; i];
  end
end

%the following loop gets index values for peaks
peaks = [];
```

```
for i = 1:length(tmp_peaks)
   m = tmp_peaks(i);
   peaks = [peaks;peaks_list(m,:)];
end

t1 = clock;
total_time = etime(t1,t0)

peaks_list = peaks;
```

The function regions_finder takes as input a precipitation image with precipitation rates for each pixel in mm/hour and a list of the independent storm centers within the precipitation image. The function outputs a cell array containing lists of the pixels associated with each independent storm center.

```
function regions = regions_finder(rain,peaks_list);

t0 = clock;

[rows cols] = size(rain);

max_value = ceil(max(max(rain)));
for i = 1:(max_value - 1)
   list(i) = max_value - i;
end

if max_value == 1
   list = [];
end

list = [list 0.05];

num_peaks = size(peaks_list,1);

idx = peaks_list(:,2);

already_claimed = [];
for i = 1:length(idx)
   region_indices{i} = idx(i);
   already_claimed = [already_claimed;idx(i)];
end
%sets up array for all claimed indices
m = 0;

previous_value = max_value;
for g = 1:length(list)
   current_value = list(g)

   values = peaks_list(:,1);
   valid = find(values >= current_value);

   tmp = 1;
   m = m + 1;
   while (tmp == 1)
      %this loop takes the indices we have already claimed, claims the
```

```
%eight-connected indices and makes a list of all those indices that are
%connected to the original claimed indices, then checks that list against
%the original claimed indices to get rid of all the overlap - end up with
%the indices that are one-pixel out from the original claimed indices

for q = 1:length(valid)

  h = valid(q);
  indices = region_indices{h};

  test_indices = [];
  index = [];

  for i = 1:length(indices)
    index(1) = indices(i) - rows - 1;
    index(2) = indices(i) - rows;
    index(3) = indices(i) - rows + 1;
    index(4) = indices(i) - 1;
    index(5) = indices(i) + 1;
    index(6) = indices(i) + rows - 1;
    index(7) = indices(i) + rows;
    index(8) = indices(i) + rows + 1;

    test_indices = [test_indices;index'];
  end

  used_indices = [];
  for i = 1:length(test_indices)
    temp = [];
    for j = 1:length(indices)
      temp(j) = (test_indices(i) == indices(j));
    end
    if (max(temp) > 0)
      used_indices = [used_indices;i];
    end
  end
  used_indices = floor(used_indices);
  test_indices(used_indices) = [];

  used_indices = [];
  for e = 1:(length(test_indices) - 1)
    temp = [];
    for f = (e+1):length(test_indices)
      temp(f) = (test_indices(e) == test_indices(f));
    end
    if max(temp > 0)
```

```
        used_indices = [used_indices;e];
    end
end
used_indices = floor(used_indices);
test_indices(used_indices) = [];
%these two loops get rid of test_indices that are already part of indices and
%the ones that appear twice because of overlap

used_indices = [];
temp = [];
value = rows*cols;
for i = 1:length(test_indices)
    temp = (test_indices(i) < 1 | test_indices(i) > value);
    if temp
        used_indices = [used_indices,i];
    end
end
used_indices = floor(used_indices);
test_indices(used_indices) = [];
%gets rid of test_indices that are beyond the edges

used_indices = [];
for i = 1:length(test_indices)
    temp = [];
    for j = 1:length(already_claimed)
        temp(j) = (test_indices(i) == already_claimed(j));
    end
    if (max(temp) > 0)
        used_indices = [used_indices;i];
    end
end
used_indices = floor(used_indices);
test_indices(used_indices) = [];
%gets rid of test_indices that have already been claimed by
%another peak

test_values = [];
test_indices = floor(test_indices);
test_values = rain(test_indices);

idx = [];
idx = find(test_values > current_value)

indices = [indices;test_indices(idx)];
already_claimed = [already_claimed;test_indices(idx)];
region_indices{h} = indices;
```

```
            test_indices = [];

            if isempty(idx)
               tempo(q) = 0;
            else
               tempo(q) = 1;
            end
         end
         tmp = max(tempo);
         tempo = [];
      end
      previous_value = current_value;
      step{m} = region_indices;
   end

%look for points > 0.05 that are not claimed by any region...if they are
%isolated, assign them to their own region

t1 = clock;
total_time = etime(t1,t0)

regions = region_indices;
```

The function picture_maker takes as input a precipitation image with precipitation rates for each pixel in mm/hour and a cell array containing lists of the pixels associated with each independent storm center. The function outputs an image that displays the different storm regions within the precipitation image.

```
function picture = picture_maker(rain,regions);

[rows,cols] = size(rain);

picture = zeros(rows,cols);

length = size(regions,2);

for i = 1:length
   idx = regions{i};
   idx = floor(idx);
   picture(idx) = i;
end
```

```
function picture = picture_maker(rain,regions);

[rows,cols] = size(rain);

picture = zeros(rows,cols);

length = size(regions,2);

for i = 1:length
   idx = regions{i};
   idx = floor(idx);
   picture(idx) = i;
end
```

The function feature_finder takes as input a precipitation image with precipitation rates for each pixel in mm/hour and a cell array containing lists of the pixels associated with each independent storm center. The function outputs a vector containing the eight features that were selected to be looked at for each storm region in the precipitation image.

```
function [features] = feature_finder(rain, regions);
% this function takes the segmented regions and returns feature
% vectors for each storm
% integrated_rate is a sum of all the rain rates over the area
% peak_rate is the highest rain rate within the storm
% num_pixels is the number of pixels assigned to the region
% avg_rate is the average rain rate within the storm
% num_half is the number of pixels above half the peak rate
% num_avg is the number of pixels above the average rate
% num_1 is the number of pixels above the rate 1 mm/hr
% ratio is the ratio of the major to minor axes of an ellipse that is fitted to the shape of the
% storm

num_reg = size(regions,2);
[len wid] = size(rain);

for i = 1:num_reg
   idx = regions{i};
   idx = floor(idx);
   num_pixels(i) = size(idx,1);
   values = rain(idx);
   peak_rate(i) = max(values);

   total_value = 0;
   for j = 1:num_pixels(i)
      total_value = total_value + values(j);
   end

   integrated_rate(i) = total_value;
   avg_rate(i) = total_value/num_pixels(i);

   idx = [];
   values = [];
end

features{1} = integrated_rate;
features{2} = peak_rate;
features{3} = num_pixels;
features{4} = avg_rate;
```

```
for i = 1:num_reg
   idx = regions{i};
   idx = floor(idx);
   values = rain(idx);
   peak_rain = peak_rate(i);
   avg_rain = avg_rate(i);

   half_peak = peak_rain/2;

   index = find(values > half_peak);
   num_half(i) = size(index,1);

   index = find(values > avg_rain);
   num_avg(i) = size(index,1);

   index = find(values > 1);
   num_1(i) = size(index,1);
end

features{5} = num_half;
features{6} = num_avg;
features{7} = num_1;

for i = 1:16
   i
   idx = regions{i};
   idx = floor(idx);
   num_pixels(i) = size(idx,1);

   if (num_pixels(i) > 5)
      tmp = zeros(len,wid);
      tmp(idx) = 1;
      [x y] = find(tmp == 1);
      ellipse_t = fit_ellipse(x,y);
      long = ellipse_t.long_axis
      short = ellipse_t.short_axis
      ratio(i) = long/short;
   else
      ratio(i) = 1;
   end
end
ratio(17) = 3;
for i = 18:20
   i
   idx = regions{i};
   idx = floor(idx);
```

```
    num_pixels(i) = size(idx,1);

    if (num_pixels(i) > 5)
       tmp = zeros(len,wid);
       tmp(idx) = 1;
       [x y] = find(tmp == 1);
       ellipse_t = fit_ellipse(x,y);
       long = ellipse_t.long_axis
       short = ellipse_t.short_axis
       ratio(i) = long/short;
    else
       ratio(i) = 1;
    end
end
ratio(21) = 1.5;
for i = 22:num_reg
    i
    idx = regions{i};
    idx = floor(idx);
    num_pixels(i) = size(idx,1);

    if (num_pixels(i) > 5)
       tmp = zeros(len,wid);
       tmp(idx) = 1;
       [x y] = find(tmp == 1);
       ellipse_t = fit_ellipse(x,y);
       long = ellipse_t.long_axis
       short = ellipse_t.short_axis
       ratio(i) = long/short;
    else
       ratio(i) = 1;
    end
end
features{8} = ratio;
```