Short-Term Precipitation Nowcasting for Composite Radar Rainfall Fields

by

Matthew P. Van Horne

B.S., Massachusetts Institute of Technology (2002)

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Civil and Environmental Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2003

Author . . . .
Department of Civil and Environmental Engineering
August 15, 2003

Certified by . . . .
Dara Entekhabi
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by . . . .
Heidi Nepf
Chairman, Departmental Committee on Graduate Students

Short-Term Precipitation Nowcasting for Composite Radar Rainfall Fields

by

Matthew P. Van Horne

Submitted to the Department of Civil and Environmental Engineering
on August 15, 2003, in partial fulfillment of the
requirements for the degree of
Master of Science in Civil and Environmental Engineering

Abstract

Precipitation nowcasting at very short lead times is a difficult and important earth science goal. The implications of nowcasting extend into aviation, flood forecasting and other areas. Using correlation analysis for the generation of velocity vectors to advect a composite radar rainfall field is the method of nowcasting utilized in this work. The MIT Lincoln Laboratory Growth and Decay Storm Tracker (GDST) is a correlation-based nowcasting algorithm that utilizes spatial filtering to eliminate the potentially adverse effects of transient, small-scale rainfall features in the correlation step. The GDST is used in this work to evaluate the benefits of image filtering as compared to a situation where the filtering is absent. The GDST generates a spatially variable velocity field for input rainfall field advection. Forecasts made using this enhancement are compared to forecasts made using a single velocity value for all input pixels in order to determine the benefits of allowing for differential motion within the storm envelope. The results from three storm cases show that image filtering provides improvement in forecast accuracy over an unfiltered case however, to fully determine any benefits from using spatially variable velocities requires more work.

This work also documents the development and testing of a new correlation-based nowcasting algorithm. The Automated Precipitation Extrapolator (APEX) builds on advancements made over the past 40 years to provide highly accurate precipitation nowcasts. Initial testing shows that APEX-generated forecasts are more accurate than persistence forecasts, and are approximately as accurate as forecasts generated by the GDST or with a uniform advection method. Allowing for small errors in forecasted rainfall location, through an extended verification kernel, APEX-generated forecasts are visibly more accurate than GDST forecasts or uniform advection forecasts.

Thesis Supervisor: Dara Entekhabi
Title: Professor of Civil and Environmental Engineering

Acknowledgments

I would like to thank Enrique Vivoni and Dara Entekhabi for their help throughout the duration of this research. I would also like to acknowledge the assistance of Bob Hallowell, Marylin Wolfson and Barbara Forman of Lincoln Laboratory for their help in obtaining and deciphering the original model. Ross Hoffman and Chris Grassotti of Atmospheric and Environmental Research, Inc. also provided significant help along the way. Funding for this thesis was provided by the National Science Foundation grant entitled "An Ensemble Approach to Data Assimilation in the Earth Sciences" (principal investigator: Professor Dennis McLaughlin). Support from this grant is gratefully acknowledged. On a more personal note I would like to thank my longtime girlfriend Sabrina Kinyon for her constant support and love as well as my family for their support for me from day one.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Precipitation forecasting is one of the most difficult earth system problems faced today, and consequently the results are among the most imprecise. The lack of precision is not due to lack of effort as there are numerous models designed specifically to forecast rainfall. Yet the chaotic and transient nature of precipitation continues to foil the majority of these attempts. There is a particular interest in the forecasting of rainfall at very short lead-times, and a significant amount of effort has gone into improving the accuracy of those forecasts over the last 40 years. The short-term forecasting problem is the focus of the work presented here.

The need for accurate short-term forecasts is very real in fields such as aviation safety and flash flood prediction. Precipitation causes a significant number of airplane accidents and greatly contributes to delays. The highly transient nature of precipitation requires that short-term forecasting procedures be available to aid in the final decision making for pilots and flight controllers. Flash floods are almost solely caused by high intensity, short duration rainfall over highly saturated basins. More accurate short-term forecasting will not only provide forecasts of the flood-causing rainfall but can also aid in soil moisture accounting to determine which basins are in danger of flooding, prior to rainfall. These are only two of the potential applications for short-term rainfall forecasts.

1.2 Nowcasting Background

Objective short-term forecasting can take many forms, one of which is nowcasting. Nowcasting is defined as very short-term weather forecasting with forecast lead times ranging from zero to six hours and includes methods such as extrapolation and numerical weather prediction (NWP; Glickman 2000). Both forms of nowcasting mentioned above have beneficial properties for forecasting at different spatial and temporal scales. NWP models are typically run over large spatial scales and at coarse spatial and temporal resolutions as compared to extrapolation forecasting (Browning and Collier 1989).

For many applications, including hydrometeorologic forecasting and aviation, the forecast resolutions provided by NWP models do not meet the user requirements for several reasons. The coarse spatial resolution (30-100 km) of NWP model outputs is often so large that it misses sub-grid scale processes such as small cells of convective activity. The relatively infrequent temporal spacing of forecasts (6-24 hours) can only provide general guidance about the large-scale state of the weather at the forecasted time. The complexity of the algorithms also creates another drawback, as the spin-up and adjustment time for these models is often on the scale of 6 hours (Browning and Collier 1989; Hamill and Nehrkorn 1993; Wilson *et al.* 1998; Pierce *et al.* 2000).

For these and other reasons, extrapolation-based nowcasting models have been developed to provide forecasting for short lead-times at high spatial and temporal resolutions. Rainfall forecasting by extrapolation is a vital portion of the total forecasting process used today. At short lead times, extrapolation algorithms potentially have a higher relative skill than other automated methods of forecasting (Browning 1980). Figure 1-1 (adapted from Zipser 1990) shows a conceptual ranking of several nowcasting techniques in terms of their skill as a function of forecast lead time. The dominance in skill of these methods at extremely short lead times has encouraged significant research into applications of extrapolation for forecasting. The result of this research is a great deal of knowledge about the composition and motion of both small and large rainfall areas. The following work uses this knowledge in the development and testing of two short-term extrapolation algorithms.

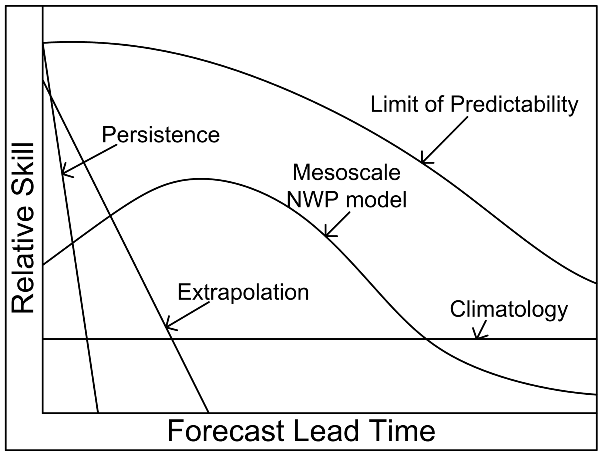

Figure 1-1: Theoretical forecast skill as a function of lead time for several forecasting methods (adapted from Zipser 1990).

Despite improvements, extrapolation-based nowcasting models have their own problems and drawbacks as well. Many extrapolation-based models do not include mechanisms for predicting changes in the structure or intensity of the rainfall field limiting the lead time for effective forecasting. In addition to these drawbacks, many initial choices will also effect the forecast product. One such choice is the feature scale to forecast. Individual cell forecasting can be done by determining cell motion with pattern recognition algorithms for centroid tracking (Einfalt *et al.* 1990; Chen and Kavvas 1992; Dixon and Wiener 1993; Johnson *et al.* 1998; Handwerker 2002) while large scale movement can be predicted by using variations of correlation analysis (Austin and Bellon 1974; Rinehart and Garvey 1978; Browning *et al.* 1982; Tuttle and Foote 1990; Brémaud and Pointin 1993; Bellon and Zawadzki 1994; Li *et al.* 1995; Wolfson *et al.* 1999). Recently a new method for multi-scale forecasting has emerged, variational echo tracking. This method utilizes a minimization process to determine a physical transformation that results in the first field changing into the second field (Grecu and Krajewski 2000; Germann and Zawadzki 2002). This method has potential to account for changes in size and intensity over the forecasting period as well as provide forecasting of multiple scales embedded within each other.

1.3 Outline and Goals

The layout of this work is as follows. An assessment of the improvement provided by including image filtering as a step prior to the correlation analysis in extrapolation-based nowcasting methods is presented in Chapter 2. Also included in Chapter 2 is a history of the use of image filtering in nowcasting, the description of the forecasting model used, a synopsis of the storm event data and evaluation criteria used throughout this work. An in depth description of the algorithm behind the nowcasting method developed for this work is presented in Chapter 3. Results from forecasts generated with this model can be found in Chapter 4, along with comparisons to persistence, uniform velocity advection and the previously developed nowcasting method. Finally, conclusions and recommendations for future development and improvement are found in Chapter 5.

The goals this work intends to achieve are as follows:

1. Goal: Determine the impact of the motion of small-scale features on forecast performance.

2. Goal: Identify the benefits of allowing for differential motion within a storm system in a nowcasting procedure.

3. Goal: Develop and test a MATLAB-based short-term forecasting procedure.

The scientific questions that these goals imply are:

1. Question: What impact does image filtering have on nowcasting accuracy?

2. Question: Do spatially variable velocities improve short-term forecast accuracy?

3. Question: How well does a new algorithm for nowcasting compare with simpler methods and an operational nowcasting algorithm?

Chapter 2

Image Filtering For Short Term Rainfall Forecasting

2.1 Introduction

A brief glance at almost any radar image from a rainfall event will show variations in the rainfall intensity from point to point. Some of these gradients are gentle while others are drastic. For most correlation-based nowcasting algorithms, these gradients are what allow the correlation coefficient to differ between points. Difficulty arises, however, when both dilation and translation occur between the images to be correlated. This changes the relative location of the sharp gradients, with respect to the storm envelope and to each other. In the case of a single-velocity nowcasting algorithm, also referred to as uniform advection, this change in relative location may result in a lower maximum correlation coefficient value than would have occurred in the absence of differential motion. In a spatially-variable velocity vector forecasting method, the effect of this differential motion may be an increase in the number of erroneous matches identified by the correlation coefficient. In either case, the accuracy of the forecast will likely suffer.

It has been well documented (see the following section), that differential motion within a storm envelope exists and is commonly observed in the case of convective weather embedded within a larger storm system. This particular level of rainfall organization can be extremely dangerous as a decrease in speed of the envelope motion can stall the heavy rainfall in a single location, but at the same time it is also extremely predictable. The high degree of predictability follows from the level of organization, since over short lead-times the envelope motion will likely remain relatively constant. Following from that assertion, prediction of the envelope motion can potentially provide very accurate forecasts for lead-times ranging from zero to two hours. One path through which this potential can be reached using a correlation-based forecasting algorithm is by eliminating the possibility of velocity contamination from internal motion within the storm envelope. One method for avoiding the influence of this motion is through image filtering.

This work mainly focuses on the benefits and associated increase in accuracy that result from applying image filtering concepts to radar rainfall fields prior to input into a local area correlation procedure. A brief look into the benefits of using spatially variable vectors as opposed to a single global vector is also included. The remainder of the work is presented according to the following outline. Section 2.2 reviews the major literature and advances in image filtering over the last 45 years, Section 2.3 describes the nowcasting algorithm used to generate the forecasts used within, Section 2.4 describes the three storms used in testing the algorithm and Section 2.5 introduces the criteria used to evaluate the forecasts. The following section, 2.6, presents the evaluations of the forecasts using the previously described criteria and the conclusions from this work can be seen in Section 2.7.

2.2 Filtering History

Image filtering in rainfall forecasting can be referred to by many different names. Some common terms for the practice include spatial smoothing, spatial integration, spatial decomposition, resolution reduction, and several others. Aside from the name, the goal of the process is the same; take a field and remove the small scale features from it. Panofsky and Brier (1958) discuss this topic in terms of one-dimensional time series' but the method is sufficiently flexible to allow multiple dimensions. They note that small scale variations in data may be of secondary importance to the problem and in many cases can actually complicate the problem. Their solution, borrowed from electrical engineering terminology, is to implement a “low-pass” filter and remove the high variability fluctuations from the data and leave only the low frequency components (Panofsky and Brier 1958). This methodology can be applied to forecasting meso-scale frontal precipitation patterns for short periods of time.

The idea of removing small scale features from rainfall maps prior to forecasting is almost as old as correlation-based nowcasting. Wilson (1966) determined that small-scale features in reflectivity or rainfall fields are short-lived and highly perishable, and need not be forecast. Browning *et al.* (1982) were the first to use spatial filtering, implemented as resolution reduction, in an operational objective forecasting scheme. Their approach was to degrade the data from a $128x128$, 5 km horizontal resolution grid to a $32x32$, 20 km horizontal resolution grid prior to centroid matching. This action effectively averaged out the reflectivity echo scales that had predictability times less than one hour. Browning and Collier (1989) generate more credence for filtering by determining that frontal rain bands can persist for several hours and often act as precursors for shorter lived convective weather. The large-scale forcing provided by these storms can then be used as a predictive tool for severe weather that results from smaller convective systems.

The work on spatial filtering was continued by Bellon and Zawadzki (1994) where they determined that averaging scale was proportional to forecast lead-time. They determined an empirical power-law relationship between averaging scale and lead time using root mean squared error and the correlation coefficient as the selection criteria. Zawadzki *et al.* (1994) also found that spatial filtering increases the effective forecast lead time by focusing on forecasting only large-scale features. In a further statement on scale-dependent predictability, Pereira Fo. *et al.* (1999), find that higher rainfall rates are more difficult to forecast since the convective systems that they are often associated with are five times more transient than stratiform rain areas. This result provides justification for the removal of the small-scale features prior to forecasting.

A different approach to filtering emerged in 1999. Prior to that, fields were filtered prior to advection and the smoothed fields were advected for forecast creation. Wolfson *et al.* used filtering in a different capacity. The Growth and Decay Storm Tracker (see Section 2.3) filters the input fields prior to correlation analysis. This allows only the large scale vectors to be created, and removes erroneous matches from small-scale motion within the storm envelope. However, instead of advecting the filtered field with the derived velocities, the original, unfiltered field is used for forecasting (Wolfson *et al.* 1999). This approach has strengths and weaknesses, but has been proven to provide accurate forecasts for aviation uses (Cartwright *et al.* 1999; Hallowell *et al.* 1999; Wolfson *et al.* 1999; Theriault *et al.* 2000).

Filtering can also be applied after the forecast has been issued for verification purposes. Several works have shown that evaluating forecasts made at a high resolution at a lower resolution leads to improved critical success index (CSI), false alarm ratio (FAR), probability of detection (POD) and root mean squared error (RMSE) (Bellon and Austin 1978; Hallowell *et al.* 1999; Pereira Fo. *et al.* 1999; Smith and Austin 2000). This result follows from the fact that large-scale features have longer temporal persistence and are therefore predictable for longer forecast times. In a different quantification, Grecu and Krajewski (2000) found that the correlation predictability time approximately doubles as horizontal resolution is degraded by a factor of eight. Mecklenburg *et al.* (2000) also demonstrated the utility of spatial filtering and showed that it impacts convective situation forecasts more than it impacts stratiform situation forecasts. This is due to the probable elimination of any useful pattern for correlation matching by smoothing a stratiform rain pattern. That work also determined that temporal smoothing does not have significant positive benefits for forecasting.

Very recently two works have emerged that continue to reproduce the results seen above but in different manners. Germann and Zawadzki (2002) found that an increase in scale, the result of a smoothing process, results in an approximately linear increase in feature lifetime. Seed (2003) showed that larger scale features have longer correlation times than smaller features and the removal of smaller features prior to forecasting results in a decrease in RMSE as compared to forecasts made with features at all scales. As expected, the lifetimes of larger scale features were greater than those of smaller scale features, both in an instantaneous comparison and in the mean. Despite the large amount of previous work done in the area of smoothing, there have been few quantitative analysis of the benefits of filtering as a pre-processing step. The following work attempts to fill that void.

2.3 Growth and Decay Storm Tracker

The Growth and Decay Storm Tracker (GDST), developed at Lincoln Laboratory at the Massachusetts Institute of Technology (MIT LL), is a correlation tracker originally created for use in the Federal Aviation Administration (FAA) Integrated Terminal Weather System (ITWS) project (Evans and Ducot 1994). This particular nowcasting method uses multi-scale separation to accurately forecast mesoscale storm events. The ideal events for this method are frontal weather systems embedded with smaller convective cells. The name of the method is deceiving, as there is no direct incorporation of cell growth and decay in the version utilized here. Growth and decay are taken into account insofar as large-scale movement occurs via cell growth on the leading edge of the system and cell decay on the trailing edge of the system. The scale-separation filtering used in this method accounts for this type of growth and decay (Wolfson *et al.* 1999). More advanced versions of the GDST include direct growth and decay trending for explicit characterization of storm growth and decay processes (Dupree *et al.* 2002).

The GDST uses cross-correlation of successive radar images to generate spatially distributed storm motion vectors. Advection is achieved by the application of these vectors to the weather image (Chornoboy *et al.* 1994). The GDST improves on traditional cross correlation analysis by using an elliptical filter to separate the large-scale storm features from smaller-scale cells, thus enabling improved tracking of the storm envelope. This separation removes the small-scale features from the field used in the local correlation analysis and therefore allows the tracking to focus solely on the large-scale motion of the storm envelope. Figure 2-1 shows the effect of a simplified filtering process on a sample storm event from 5 October 1998 using a 60 km square filter. The storm event contains smaller convective cells near the front edge of the storm followed by a trailing area of lighter stratiform rain (Figure 2-1a). From two consecutive large-scale filtered images (Figure 2-1b), a velocity field is derived and applied to the unfiltered field. The average velocity for nonzero precipitation cells in this case is 9.6 m/s directed 17° South of East. The small-scale image (Figure 2-1c) is the result of subtracting the large-scale image (Figure 2-1b) from the full image (Figure 2-1a).

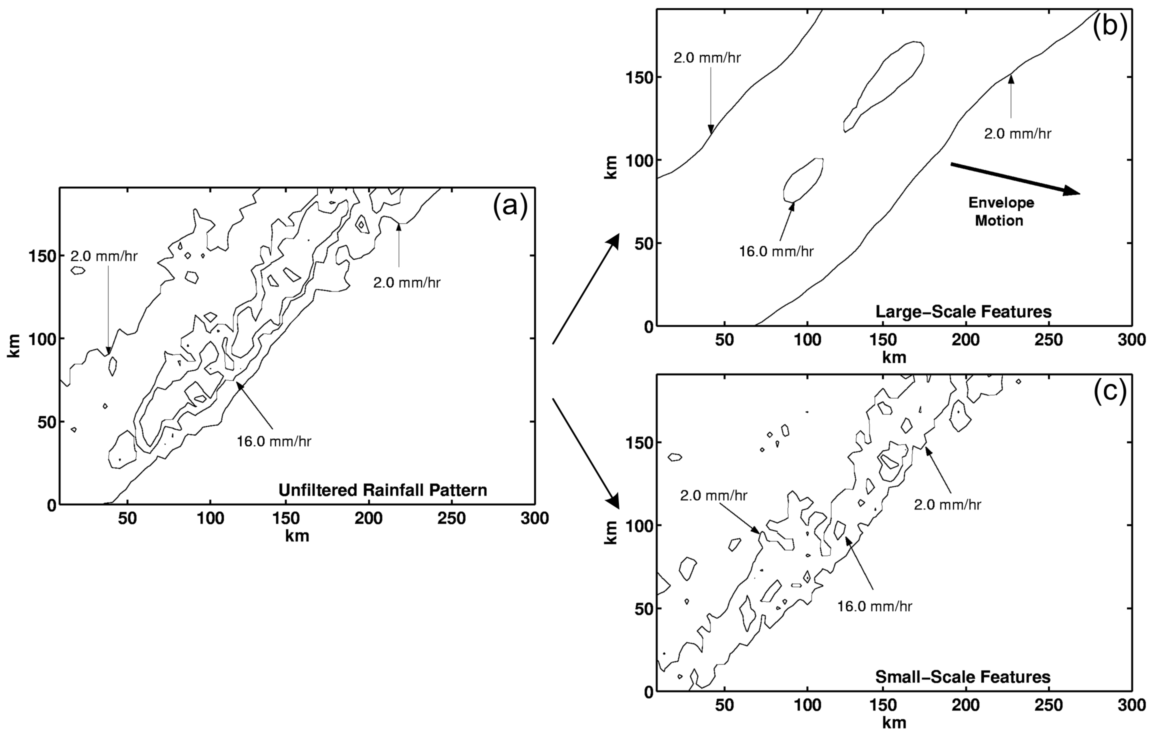

Figure 2-1: The result of applying a square filter (60 km x 60 km) to a line storm from 5 October 1998. The full image (a) is decomposed into its large-scale (b) and small-scale (c) features using an averaging filter. The arrow in (b) corresponds to the velocity derived by the GDST, 9.6 m/s directed 17○ South of East.

MIT LL originally designed the GDST to forecast line storm progression near airport areas to improve flight routing during severe weather (Forman *et al.* 1999). In this study, the GDST is applied to catchment-scale regions for the purpose of general rainfall forecasting. Also the model is tested with radar rainfall data at different spatial and temporal resolutions than previously reported. For example, preliminary testing at the DFW airport used NEXRAD raw reflectivity data at a temporal resolution of 6 minutes and a spatial resolution of 1 km over a 440 km by 440 km area (Theriault *et al.* 2000). The data used here is described in Section 2.4.

In this study we use the GDST method with radar rainfall data derived from the NEXRAD network to produce forecasts for lead-times up to 120 minutes in 15-minute increments. Forecast verification uses radar rainfall images from the forecast valid time. Figure 2-2 illustrates a sample radar rainfall input series (Figure 2-2a and Figure 2-2b are images 15 minutes apart) and output product (Figure 2-2c is the 60 minute forecast from the image in Figure 2-2b) from the GDST model as well as the differences between the forecasted and observed rain rates (Figure 2-2d is the GDST forecast minus the true radar rainfall field). This example shows that the enveloping synoptic movement can be captured and the largest forecast errors are associated with internal small-scale features that, as expected, cannot be predicted well.

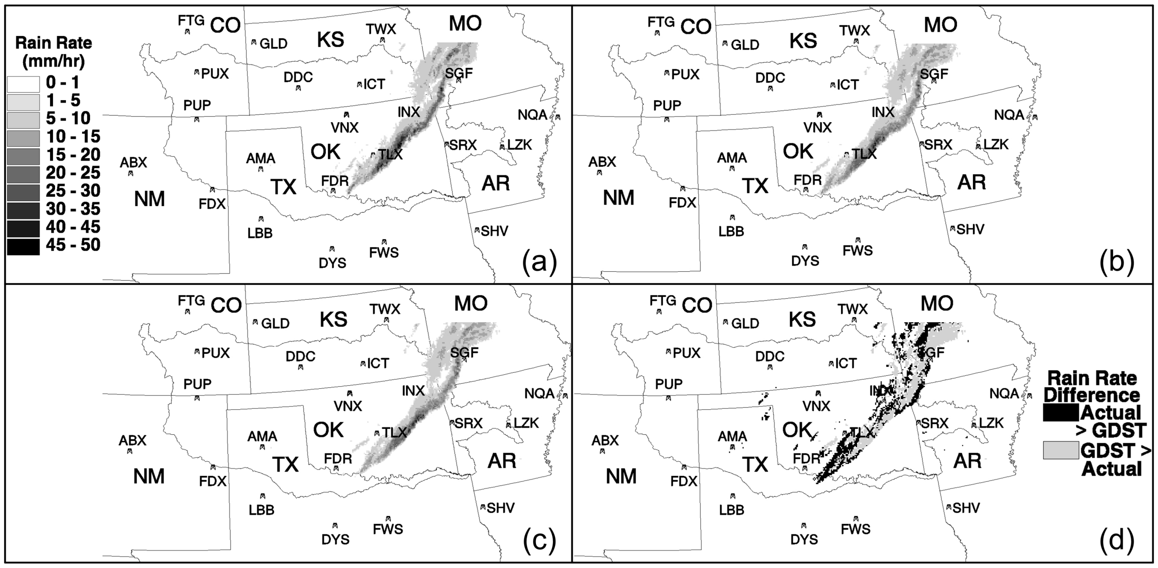

Figure 2-2: (a) Observed image at 1200 UTC on 5 October 1998. The scale in panel (a) also applies to panels (b) and (c). (b) observed image at 1215 UTC of the same day. Images (a) and (b) are cross correlated to generate a velocity field that is used to advect the storm for forecasting. (c) One hour forecast valid at 1315 UTC. (d) Difference between the forecast rainfall field shown in (c) and the actual rainfall at that time. The radar sites providing coverage of the Arkansas-Red River Basin are also shown in all panels.

2.4 Rainfall Data

The analyses contained in the remainder of the work are for three storm cases taken from 1998 and 1999. The characteristics of the storm cases are summarized in Table 2.1. Sample rainfall intensity maps from the time of peak rainfall over the Illinois River Basin can be seen in Figure 2-3. Radar rainfall fields obtained from Weather Services International (WSI) are used as input into the forecasting methods. The horizontal resolution of the data is 4.7625 km while the temporal resolution is 15 minutes. This rainfall accumulation data was chosen due to its regular temporal spacing, which provides a standard for verification of the forecasted fields and nearly continuous availability. A validation of the WSI data with NEXRAD P1, and rain gauge data can be found in a recent paper by Grassotti *et al.* (2003). The area of study is the Arkansas-Red River Basin (ARB) in the Southern Great Plains of the United States. This river basin is over 500 000 km^2 in size while the Illinois River basin, an interior basin used for additional basin-based analysis, is approximately 6 000 km^2 in size. Figure 2-4 shows the ARB with the Illinois River basin expanded.

	Event A	**Event B**	**Event C**
Date	4 Jan 1998	5 Oct 1998	13-14 Apr 1999
Duration	0000-2345 UTC	0000-2345 UTC	1200-1200 UTC
Max Rain Rate (mm/hr)	59.44	90.16	24.26
	Percentile Levels (mm/hr)		
0^{th} (low)	0	0	0
50^{th} (median)	5.08	5.08	5.08
90^{th} (high)	10.16	18.8	13.0
Storm Characteristics	Loose linear organization	Strong linear organization with well defined convective cells	Loose linear organization

Table 2.1: Characteristics of the storm cases used in this study. Rain rates are determined by extrapolating the 15 minute accumulation values to one hour.

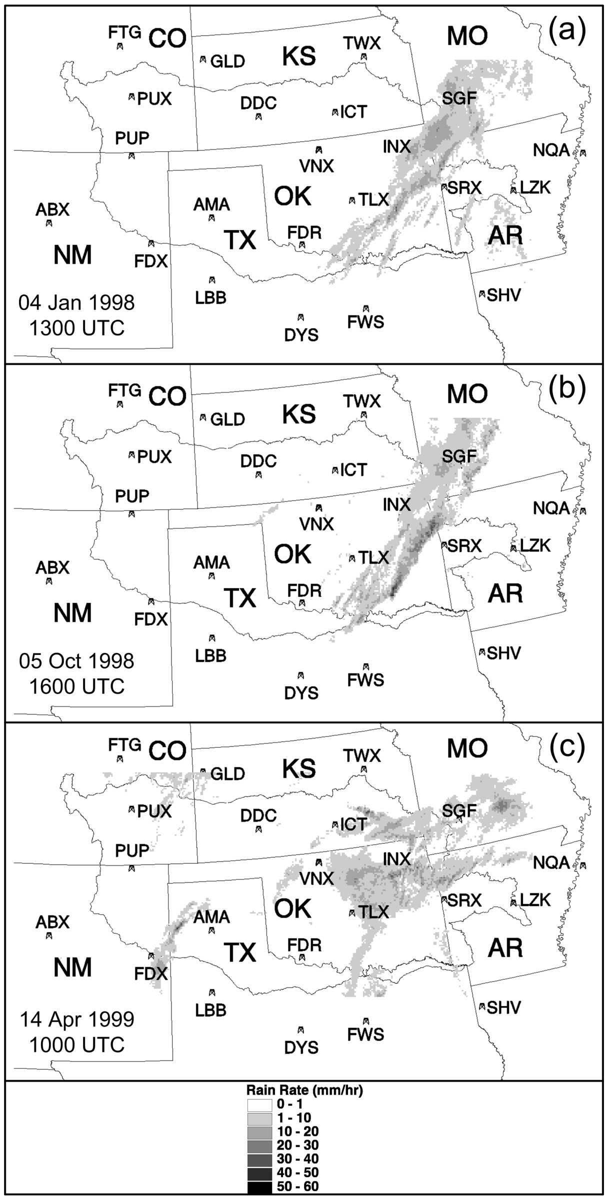

Figure 2-3: Sample radar rainfall images for the January 1998 (a), October 1998 (b) and April 1999 (c) storm events.

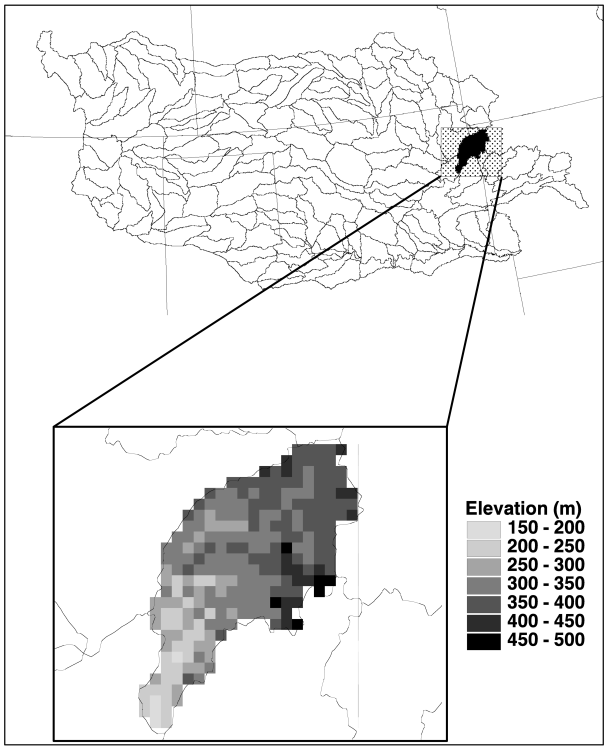

Figure 2-4: The location of the Illinois River Basin within the ARB region.

2.5 Evaluation Criteria

2.5.1 Critical Success Index

The main statistic used for forecast evaluation in this study is the critical success index (CSI; Donaldson 1975). This statistic is based on a 2x2 contingency table (Table 2.2) and evaluates the forecasts on a binary criterion. A "yes" means that the measurement (forecast or observation) exceeds a specified threshold, while a "no" indicates that the threshold was not exceeded. In this work, thresholds are selected based on the storm rainfall rate distributions. The percentile levels used are the 0^{th} (low), 50^{th} (median) and the 90^{th} (high) percentiles. The specific threshold values for each storm can be seen in Table 2.1. The CSI can also be used with an extended search area around the observed pixel in question to verify the forecasted pixel. Verification areas in this study are square grids with side lengths ranging from one to nine pixels. This variation of the CSI was first used by Bellon and Austin (1978) and the side lengths can be representative of the desired spatial forecast precision. Equation 2.1 shows the calculation to determine the CSI from Table 2.2.

		Forecasts	
		YES	NO
Observations	YES	Hit (H)	Miss (M)
	NO	False Alarm (FA)	Null (N)

Table 2.2: General 2x2 contingency table for critical success index computation

$$CSI = \frac{H}{H + M + FA} \tag{2.1}$$

2.5.2 Basin-Based Statistics

While the CSI provides a general overview of the accuracy of the forecast, it does not provide detailed information on the scale of a medium-sized river basin. A trio of basin-based measures provide information on the accuracy of the intensity, extent and distribution of the forecasted rainfall within a catchment. The mean areal precipitation (M), averaged over the river basin in question, demonstrates the ability of the forecasting model to forecast the correct rainfall intensities within a river basin. The fractional coverage (F) is the areal percentage of the basin receiving rainfall. The normalized distance to the basin outlet (D) provides more information than acquired from F by further detailing where the rainfall occurs over the river basin (Smith *et al.* 2002).

M and F are straightforwardly computed from rainfall fields while D, a function of the hydrologic distance to the outlet and a weighting function based on the point rainfall rate, is computed using

$$D = \frac{|A^{-1}| \int_A w\,(x)\, d\,(x)\ dx}{d_{MAX}} \tag{2.2}$$

$$w\,(x) = \frac{R\,(x)}{|A^{-1}| \int_A R(u)\, du} \tag{2.3}$$

where A is the total basin area $[km^2]$, x and u are locational indices, $d\,(x)$ is the distance to the outlet $[km]$, $R(x)$ is the rain rate $[mm/hr]$, and d_{MAX} is the maximum distance from the basin outlet $[km]$ (Smith *et al.* 2002). The normalized distance produces values ranging from 0 to 1 where $D = 0$ indicates that the rain is concentrated at the basin outlet while $D = 1$ indicates that the rain is concentrated at the periphery of the basin. As a point of reference for comparison, for the basin considered in this study, spatially uniform rain would produce $D = 0.58$.

2.5.3 Theoretical Error Analysis

Browning *et al.* (1982) identify four main sources of error in extrapolation based nowcasting applications, two associated with the quality of the input data and two associated with the forecasting process. The forecasting process error sources are temporal changes in the rainfall pattern (growth and decay) and temporal changes in rain feature velocity. Prior statistics describe the accuracy of the forecasts but do not assess the impact of storm changes on forecast accuracy. Understanding the impacts of storm characteristics, such as growth, decay and translation, on CSI scores can aid in the interpretation of forecasts issued by extrapolation methods. Errors in rainfall areal coverage and storm speed between the forecast generation time and the valid time of the forecast are analyzed to determine their impact on CSI scores. The area error (E_A) and the velocity error (E_V) are defined by

$$E_A = (N_O - N_F) * a \tag{2.4}$$

$$E_V = \frac{1}{N_O} \sum_{R_O(x)>0} \sqrt{u_0(x)^2 + v_O(x)^2} - \frac{1}{N_F} \sum_{R_F(x)>0} \sqrt{u_F(x)^2 + v_F(x)^2} \tag{2.5}$$

where the subscripts F and O indicate forecasted and observed values, respectively, N is the number of pixels with a rainfall rate greater than a threshold, a is the area of a single pixel (22.7 km^2), x is a locational index, u and v are the eastbound and northbound velocities $[m/s]$, respectively, and $R(x)$ is the point rainfall rate $[mm/hr]$. Since nowcasting is based on a steady state assumption, positive error values indicate that over the forecast interval the quantity in question (area or velocity) increased while a negative error value indicates a decrease. Table 2.3 provides a qualitative interpretation of the sign of the error values, while the magnitudes of the values are an indication of the strength of the changes.

	E_A	E_V
Positive	Growth	Acceleration
Negative	Decay	Deceleration

Table 2.3: Qualitative interpretations of area and velocity errors

2.6 Results

Several steps were taken to determine the improvement in skill that results from filtering the input images prior to correlation. First, several filter sizes were tested to determine the optimal filter size for each storm event. The optimal filter was chosen using the average CSI score for all lead times over the entire storm event. Next, the scores from the optimal filter were compared to the scores from forecasts made by using a $3x3$ pixel filter. This small filter approximates the unfiltered case since the amount of smoothing done by a filter of that size is negligible. This comparison is done using CSI scores with multiple thresholds. Next, the optimal filter and the zero filter are analyzed using a set of 5 verification kernels at multiple thresholds to determine the effect that spatial accuracy relaxation has on CSI scores. Following that, the optimal filter is then compared to two other forecasting methods: persistence and global advection. Persistence is a true steady state forecasting method. The current weather is assumed to persist for the entire forecasting period with no changes. Global advection is the use of a single velocity vector for forecasting as opposed to the spatially variable velocity vector field utilized by the GDST. As the threshold dependence of the scoring is seen in prior analyses, only the low threshold will be used for this analysis. Finally the 60-minute forecasts are analyzed using the trio of basin-based statistics described in Section 2.5.2 as well as with the error decomposition methodology described in Section 2.5.3.

2.6.1 Filter Size Sensitivity

The optimal filter size for image smoothing is not only lead time dependent as stated by Bellon and Zawadzki (1994) but is also storm type and season dependent (Cartwright *et al.* 1999). For the purposes of this study, the optimal filter size was determined by the averaging the CSI scores for all initial times and for all lead-times. This process returns the filter that is the best overall filter for each storm event. For the three storms (January 1998, October 1998, April 1999) , the optimal filter side lengths were 11 pixels, 41 pixels and 41 pixels respectively. Figure 2-5 shows the filter size dependence on CSI score for all three storm events. The improvement of the optimal filters over the 3x3 filter (hereafter referred to as "no filter") is noticeable even at this level of averaging.

2.6.2 Lead Time and Threshold Comparison

Analyzing the lead time dependence of the CSI scores for the optimal filter case and the no filter case further expands the understanding of the benefit of utilizing image filtering in a correlation forecasting method. Figure 2-6 compares the storm-averaged CSI scores at all lead times between the optimal filter and no filter. The results show that in most cases there is slight improvement at short lead times but the increase in accuracy is more apparent at longer lead times. The October 1998 event shows the most improvement with the use of image filtering which can be expected due to the highly linear storm envelope and well defined convective and stratiform regions. The similarity in the increase in accuracy at all threshold levels indicates that image filtering improves the predictability of all rainfall intensities approximately equally.

2.6.3 Verification Area Analysis

Using an extended scoring kernel in the CSI computation can have a significant impact on the resultant score. The size of the kernel used represents the level of spatial precision desired by the user. A larger kernel indicates that a forecast may

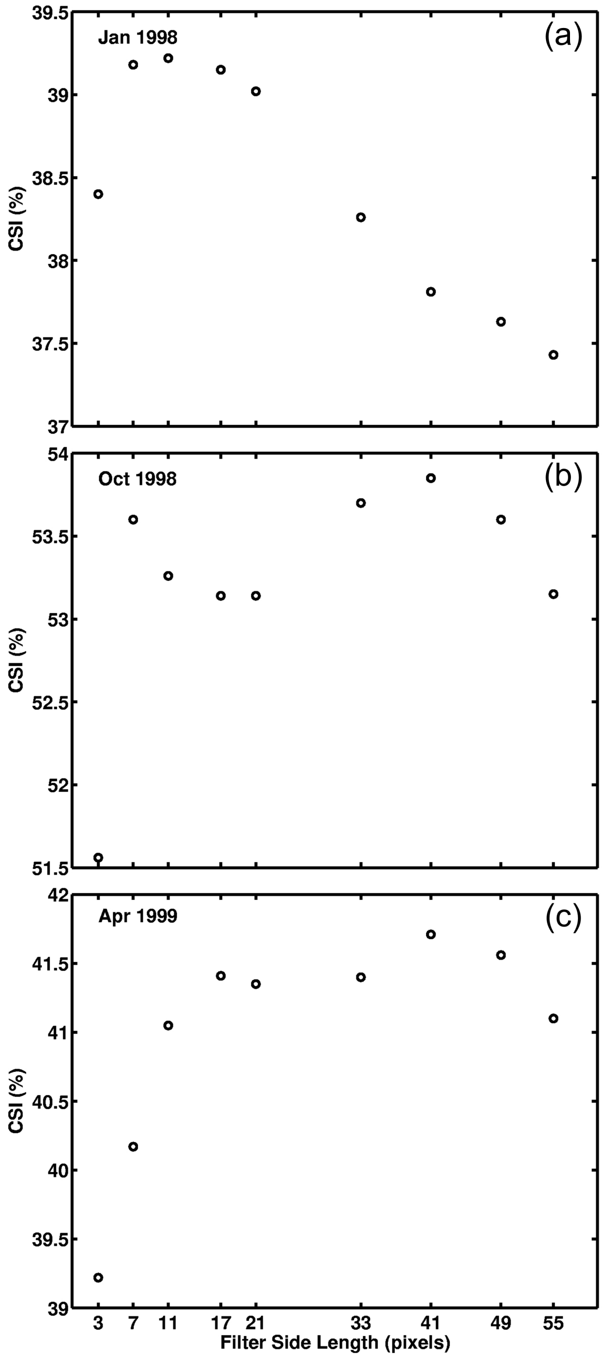

Figure 2-5: Average CSI scores for multiple filter sizes for each storm event.

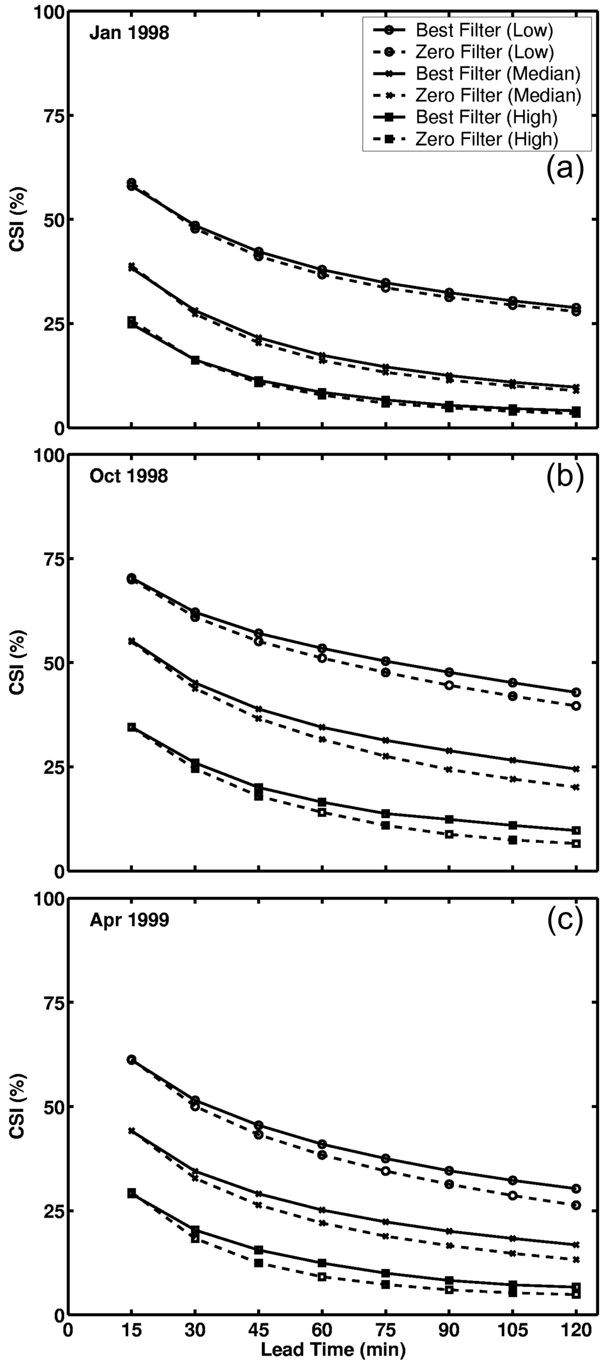

Figure 2-6: CSI scores at several rainfall thresholds versus lead time for the three storm events for forecasts generated by the GDST. Solid lines represent forecasts made using the optimal filter for each storm event while dashed lines represent the use of no filter for each storm event.

still be useful even with some degree of spatial misplacement while a smaller kernel narrows the desired precision. Figure 2-7 shows the optimal filter and the zero filter 60 minute forecasts scored with a range of verification areas at several thresholds. The increased sensitivity of rainfall at higher rainfall thresholds to an increase in verification area shows that when the GDST is forecasting high intensity rainfall, there is a larger chance that its precise location will not be forecasted correctly. The large increase in score with an increase in area shows that, for the particular cases examined here, expanding the kernel to include the immediately neighboring pixels causes the highest intensity rainfall to be forecast approximately as well as the median intensity rainfall. The almost constant improvement seen by the optimal filter over no filter indicates that the improvement in forecast accuracy as the required spatial accuracy is decreased is not dependent on filtering of the images.

2.6.4 Uniform Advection Comparison

Uniform, or global, advection is a strict Lagrangian persistence forecasting method. There is no change in the rainfall pattern over the forecast time since each pixel is moved by the same velocity. To this point there has been no conclusive proof that a distributed velocity field will generate more accurate forecasts than those created using a uniform velocity field (Seed 2003). To determine the benefit of using a distributed velocity field generated by the GDST, forecasts were generated using both methods, and were compared to each other and the forecasts made using a persistence assumption. The results of this comparison can be seen in Figure 2-8. The CSI scores shown here are calculated for the low threshold and it is obvious that applying some form of advection to the input rainfall field greatly increases the accuracy of the forecasts. However, the benefits of applying a distributed velocity field over a uniform velocity field are not obvious at this threshold.

Analysis of the GDST and the uniform advection forecasts at all three thresholds can be seen in Figure 2-9. At higher thresholds, especially for the October storm event, the GDST shows an increased improvement in forecast accuracy over the more simplistic method. However, this improvement is still very small and from this analy-

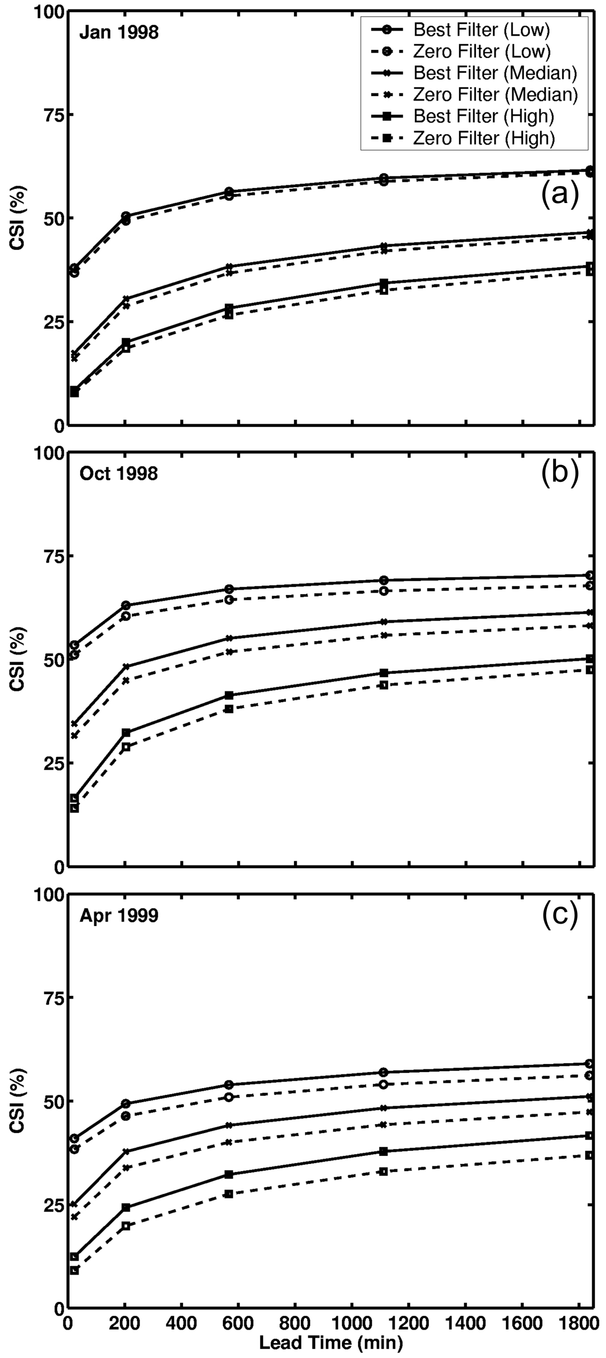

Figure 2-7: CSI scores at several rainfall thresholds versus verification area for GDST optimal filter and no filter 60-minute forecasts for the three storm events.

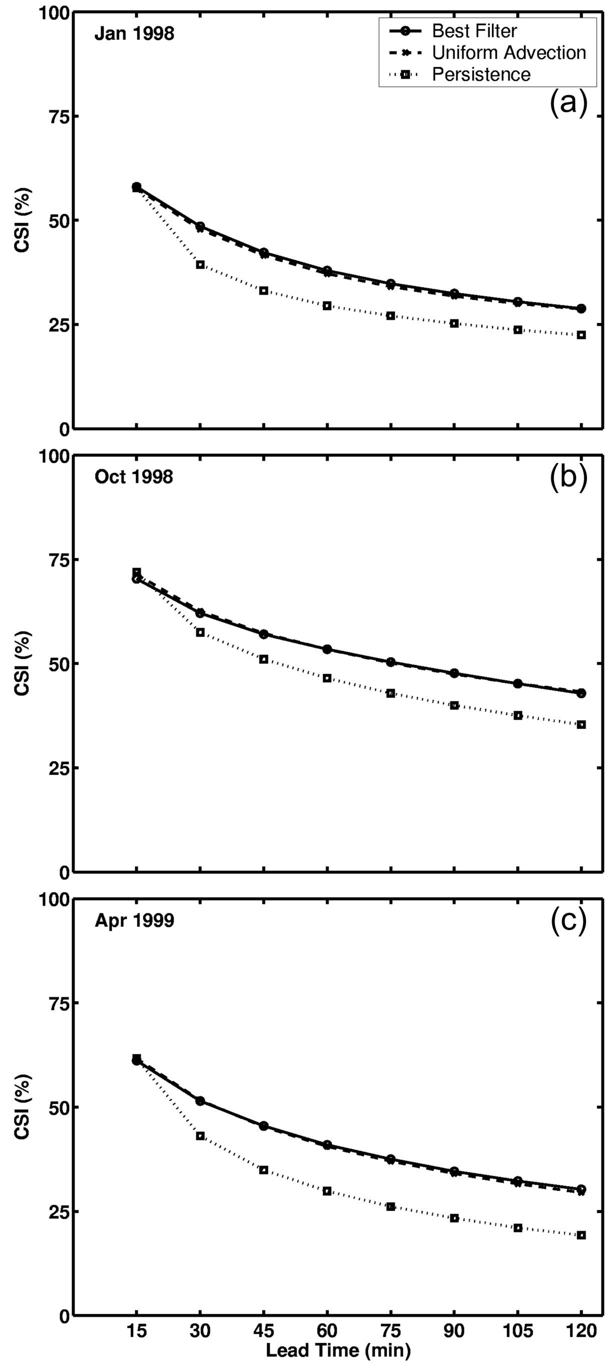

Figure 2-8: Comparison of lead time dependent CSI scores between the GDST (solid line), a global advection algorithm (dashed line) and persistence forecasting (dotted line). Note: GDST (solid) and global advection (dashed) lines almost overlap.

sis it is difficult to conclusively determine whether a spatially distributed velocity field does in fact provide a significant improvement in forecast accuracy. The similarity between the GDST and the uniform advection scoring may be an artifact of the interpolation that is done on the GDST velocity fields. Due to quality control constraints and the use of a block processing algorithm, as opposed to a sliding neighborhood method, the GDST typically uses 30 or less velocity vectors acquired through the correlation analysis. The remainder of the field is interpolated using an inverse distance squared algorithm which would tend to bias the majority of the velocities to be close to the average value of the correlation-generated velocities. This average velocity is likely very close to the global velocity used in the uniform advection procedure and would therefore result in forecasts that have been advected in much the same way as the uniform velocity method.

2.6.5 Basin-Based Analysis

The previous sections have shown the results of large-scale analyses of forecasts generated by the GDST algorithm. To get a better grasp of the smaller scale dynamics of the forecasts, the trio of basin-based statistics presented in Section 2.5.2 is used to show a different kind of accuracy. The Illinois River basin, shown in Figure 2-4, is the basin over which this analysis is done. The first conclusion that can be drawn from the results shown in 2-10 is that the January and October storm events were significant rainfall events over the Illinois River while the April storm did not provide nearly the same amount of water to the basin. A comparison between the observed and the GDST 60-minute forecast values for the mean areal rainfall (M; Figures 2-10a, d, and g) show that for all three storm events, the GDST generally predicted the magnitude and the timing of rainfall peaks with slight lags in time and small overestimations of the magnitude.

The fractional coverage (F) of rainfall over the basin for the GDST 60-minute forecasts and the observations can be seen in Figures 2-10b, e, and h. For the period of highest intensity rainfall, the GDST forecasts the coverage well, again showing a slight time lag behind the observations. The similarity in shape between the observed

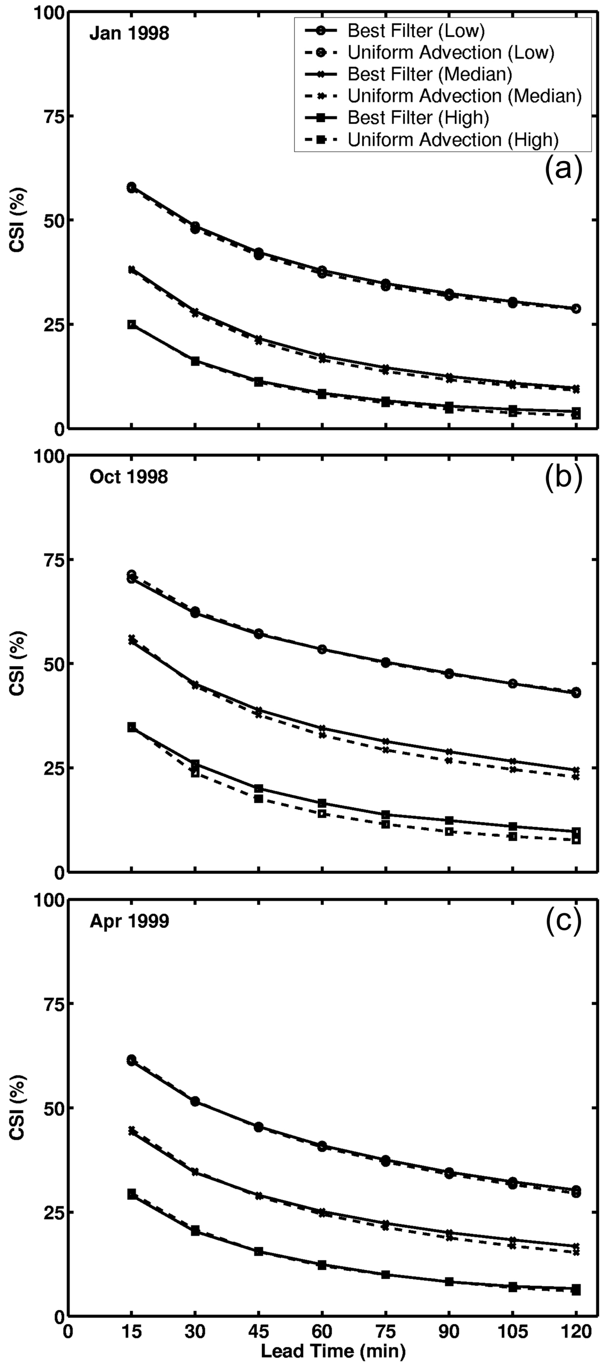

Figure 2-9: Rainfall threshold comparison between the GDST (solid line) and a global advection algorithm (dashed line) for the three storm events.

and the forecast time series' indicate that the GDST is forecasting approximately the correct number of rain-filled pixels for the storms events examined here. The normalized distance to the basin outlet (D) provides more information about the spatial placement of the rainfall than F provides. Seen in Figures 2-10c, f, and i, D displays information about the accuracy with which the correct rainfall intensities are correctly placed within the basin. The horizontal line seen in these Figures at $D = 0.58$ indicates the normalized distance to the basin outlet for uniform rain over the entire basin. Despite the erratic look of the data, for the periods of intense rainfall the GDST tends to forecast the values of D well as compared to the observed values. These statistics may be of use when evaluating the potential skill of a series of forecasts for incorporation into a river basin model.

2.6.6 Theoretical Error Analysis

Introduced in Section 2.5.3, the error decomposition methodology was applied to the 60-minute forecasts for each storm event. Figure 2-11 shows the time series' that resulted from this decomposition after smoothing with a moving average filter to eliminate the transient features and only capture the underlying trends. Figures 2-11a, d, and g show the change in CSI score between consecutive 60-minute forecasts, Figures 2-11b, e, and h show the area error (E_A) between the observations and the forecasts and Figures 2-11c, f, and i show the velocity magnitude error (E_V). When looked at individually, E_A and E_V can show the extent to which the underlying assumption of nowcasting, the steady state assumption, did not apply for the particular storm event. When looked at together and in conjunction with the change in CSI, this error decomposition may function as a diagnostic tool to determine the relative contribution of changes in rainy area and storm speed to the accuracy of the forecast. However, the intricacies of using this methodology in that capacity are complex and are still being developed, so that level of analysis is not attempted in this work. Further research into the utility of this concept may provide a standard by which multiple nowcasting algorithms may be compared.

On a more basic level, E_A and E_V provide useful information about the devel-

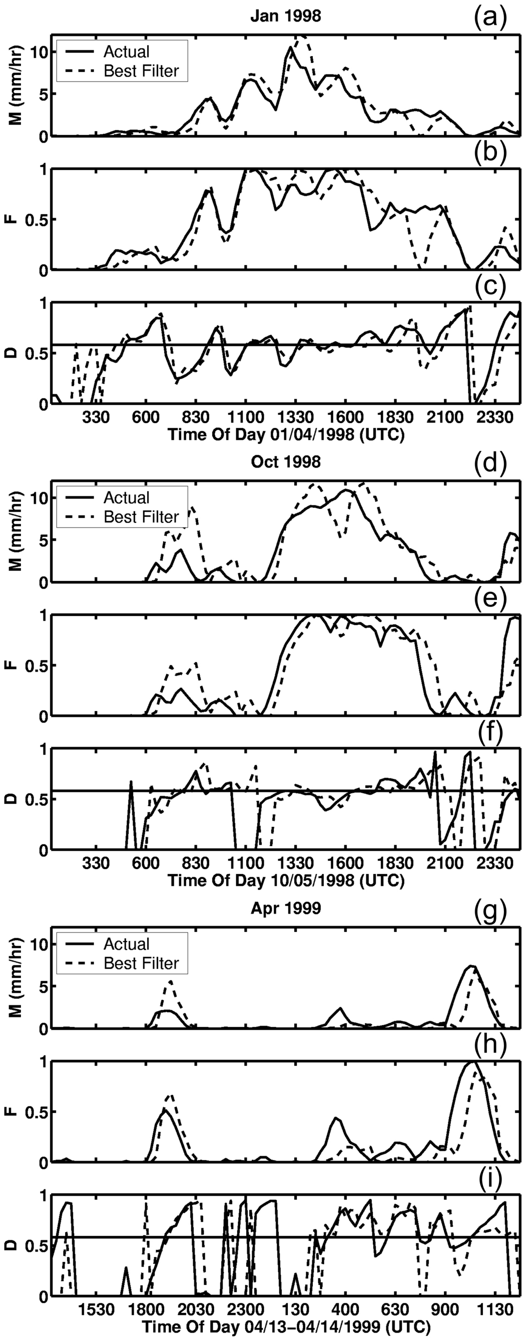

Figure 2-10: Trio of basin based statistics for the January 1998 (panels a, b, and c), the October 1998 (panels d, e, and f) and the April 1999 (panels g, h, and i) storm events comparing the GDST 60-minute forecast (dashed line) with the observed values (solid line). Panels a, d, and g show the mean rainfall rate (M) over the basin during the storm event, panels b, e, and h show the fractional coverage of rainfall (F) over the basin during each event and panels c, f, and i show the normalized distance to the outlet (D) over the duration of the event.

opment and progression of the storm event over its lifetime. For example, the April event consistently shows a relatively small value for E_V (Figure 2-11i) but is in a constant state of growth (Figure 2-11h), which may be one of the causes of its low CSI scores relative to the October event. However, the October event shows small changes in velocity over the course of the event but also has a smaller magnitude of growth and decay, potentially resulting in higher CSI scores. The January event is very erratic in terms of both area and velocity errors creating a very difficult steady state nowcasting situation, which is reflected in the very low CSI scores.

2.7 Conclusions

Using an operational nowcasting algorithm (GDST) the improvement in forecast accuracy as a result of spatial filtering prior to correlation analysis is assessed. In addition, the benefits of using the GDST, a spatially variable velocity vector advection method, as compared to global advection and persistence, methods using more simplistic advection criteria, is also determined. However, as this work only examines a few storm events with a single operational nowcasting algorithm, the applicability of the results to a more general statement on overall forecast accuracy improvement have yet to be determined. Despite this caveat, the results from this work are still encouraging to the development of more accurate nowcasting methods, as new mathematical methods are developed to represent changes in the rainfall fields that are not accounted for within the GDST.

The results from this study show that in comparison to several simplistic forecasting schemes, the enhancements utilized in the GDST provide an increase in forecasting skill for lead times from 15 to 120 minutes. First, the addition of image filtering in the GDST shows improvement in all three storm cases examined in this work, the degree of improvement is likely a function of storm organization and rate of change of speed and area over the forecasting interval. Second, GDST forecasts show significant improvements over persistence forecasting for all three storm cases at lead times greater than 15 minutes. Third, slight benefits are seen by using spatially variable

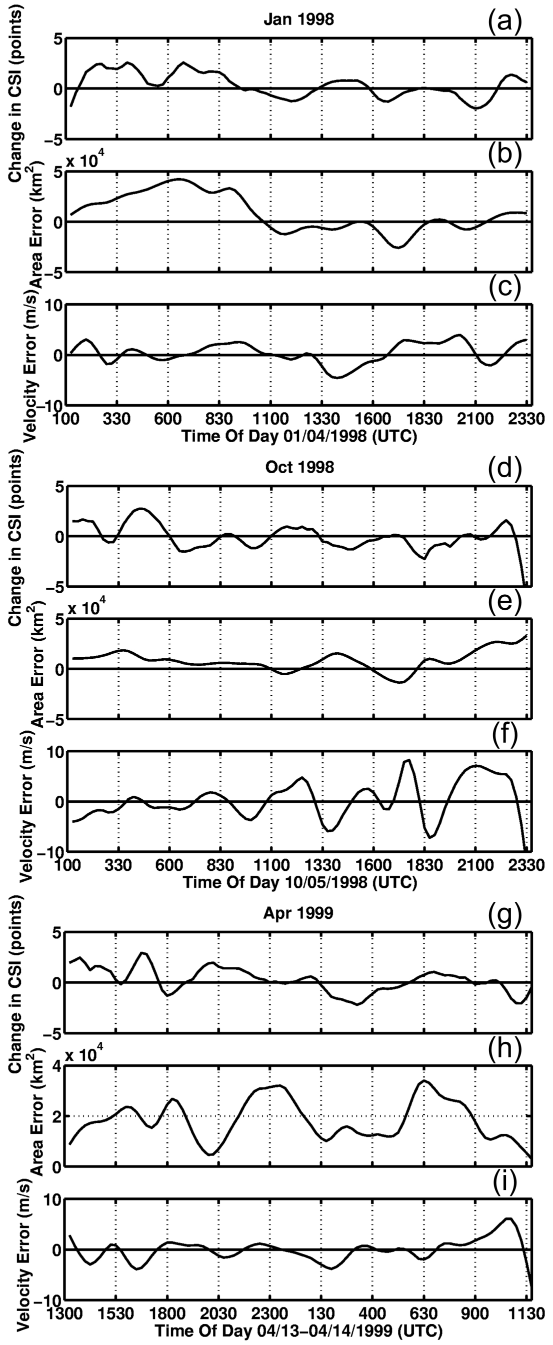

Figure 2-11: Theoretical error analysis for the January 1998 (panels a, b, and c), the October 1998 (panels d, e, and f) and the April 1999 (panels g, h, and i) storm events showing the evaluation for the GDST 60 minute forecasts over the entirety of each event. Each time series has been smoothed using a moving average filter to show only the major trends. Panels a, d, and g show the changes in CSI score between consecutive forecasts, panels b, e, and h show the area errors and panels c, f, and i show the velocity errors.

velocity vectors instead of a single vector for advection of the storm envelope. This aspect of nowcasting would benefit greatly from more research into the benefits of using spatially variable velocities as the implementation of this method has an associated increase in computation time that may or may not be justified by the small increase in accuracy seen here. Fourth, on the scale of a medium-sized river basin, the GDST performs very well for the periods of highest intensity rainfall for the three storm events and less well for periods of significant change. This is to be expected from a steady-state forecasting algorithm. Finally, using the theoretical error analysis included here, the ability of a nowcasting algorithm to account for changes in speed and area can be assessed and information regarding the rate of change of vital storm characteristics can be garnered. More research is necessary to analyze the impact of rainfall nowcasting on fields such as hydrometeorologic forecasting and flood prediction.

Chapter 3

The Automated Precipitation Extrapolator (APEX)

3.1 Introduction

The Automated Precipitation Extrapolator (APEX) nowcasting method was developed at the Massachusetts Institute of Technology to produce short-term nowcasts from composite radar rainfall fields. Section 3.2 details some previous work in correlation-based extrapolation algorithms, Section 3.3 provides a detailed description of the APEX model, and the following Chapter, Chapter 4, provides results of the testing of APEX. A function-by-function description of the algorithm can be found in Appendix A and the actual source code can be seen in Appendix B.

3.2 Correlation Analysis Background

Correlation methods have been used for time series analyses in the meteorological sciences since Panofsky and Brier (1958). Forecasting of radar rainfall patterns via correlation methods was first attempted by Hilst and Russo (1960). They utilized the location of the maximum cross-correlation coefficient between consecutive fields as the displacement of a storm feature. Their methodology was further advanced by Wilson (1966), who applied the value of the correlation coefficient between consecutive radar patterns as it relates to the movement and predictability of radar echoes at different scales. This research soon spawned new innovations in weather forecasting with the realization that correlation analysis can provide more detail than a single "global" vector. Leese *et al.* (1971) used successive satellite images to generate a "distributed" field of vectors based on the cross-correlation of segments of cloud images. This was one of the first examples of using local area correlation analysis to generate a spatially variable velocity field. The use of an optical system for correlation coefficient estimation, provided empirical evidence for two obstacles traditional correlation analysis needed to overcome. The internal motion of individual storm cells within the overall storm envelope, and storm motion being composed of cell growth and decay on the edges of the storm as well as large-scale forcing were these obstacles (Zawadzki 1973).

Following these developments, the correlation method for single radar forecasting was employed in a procedure to correlate whole radar images. This results in the derivation of a single displacement vector best describing the average storm motion in the interval between radar images (Austin and Bellon 1974). This procedure was used in an operational setting in the summers of 1976 and 1977 and showed results that were superior to persistence and two other objective forecasting methods. They also concluded that the major loss of predictability in a steady-state method, such as extrapolation, is from rearrangement of rainfall patterns during the forecast interval (Bellon and Austin 1978). A landmark advancement was made by Rinehart and Garvey (1978), who used the Tracking of Radar Echoes with Correlations (TREC) algorithm to generate a full grid of spatially varying velocities based on correlation analysis of small sub-sections of radar reflectivity images.

The effect that current weather conditions have on weather conditions in the near future cemented the importance of extrapolation methods to overall forecasting ability. The most accurate forecasts for short lead-times (zero to six hours), when considering mesoscale and synoptic scale NWP models and extrapolation techniques, are generated by extrapolation methods (Browning, 1980). This conclusion along with the new TREC algorithm spurred new research into other methods of nowcasting that may aid in the increase in accuracy of the forecasts. The first of such efforts was undertaken in an attempt to extend the effective lead-time of extrapolation forecasts through growth and decay trending. The results showed that the additional computation only negligibly increased the accuracy over simple advection, possibly due to the inability to predict the proper decay rates near the end of the storm lifetime (Tsonis and Austin 1981).

To this point, extrapolation based nowcasting methods had been confined to experiments using data from a single radar, not utilizing the forecasting potential provided by regional radar networks. Many mesoscale storms have lifetimes exceeding their time in the observation area of a single radar. Advancements in data assimilation techniques and an increase in computation time helped to overcome this limitation by allowing for the creation of regional radar mosaics. In 1982, an extrapolation method using the combined data of four weather radars was demonstrated. The forecasting method was a variation on a distributed vector procedure, in that cells were matched individually and assigned different displacement vectors. But as with most centroid tracking algorithms, each rainfall echo was only assigned a single velocity. The results showed that objective extrapolation forecasts using networked weather radar data were superior to subjective human forecasts in most cases (Browning *et al.* 1982). Shortly thereafter, a niche for correlation tracking was distinctly determined in a direct comparison of correlation and centroid tracking methods. The study showed correlation methods performed better on average than did the centroid tracking methods and also were more accurate than persistence forecasts (Ciccione and Pircher 1984). Despite the advantage over centroid tracking, extrapolations main hurdle was still yet to be successfully overcome. This hurdle was "rediscovered" in 1989, validating an observation made more than fifteen years prior by Zawadzki (Zawadzki 1973). This observation was that small-scale rainfall features often travel at velocities different than that of the large scale system that they may be embedded in (Browning and Collier 1989).

The merging of atmospheric physics into extrapolation algorithms has shown potential to improve the accuracy of precipitation nowcasts. Using a correlation tracker (from Leese *et al.* 1971) with a physically based model for growth and decay prediction, one study showed that this combination of methods can produce a forecast with a lower RMSE than advection alone (Seo and Smith 1992). The use of successive GOES satellite images, in conjunction with correlation analysis to generate a regularly spaced vector field provided advancements in quality control and interpolation (Hamill and Nehrkorn 1993). Further development in correlation-based short-term forecasting came in the form of a postprocessing algorithm to modify vectors generated by the TREC extrapolation algorithm. This postprocessing step forces the velocity field to comply with a two-dimensional continuity equation, giving rise to the nomenclature continuity of TREC vectors (COTREC; Li *et al.* 1995).

Due to their high intensities, thunderstorms are often the precipitation type of greatest concern. As of 1998, advances in extrapolation and knowledge of rainfall field characteristics have provided a solid basis for significant advances in the accurate forecasting of convective weather in the near future (Wilson *et al.* 1998). That optimism is shared in a second review paper where the utilization of knowledge of rainfall field statistics can be used to generate more accurate short-term forecast schemes. This conclusion is based on the fact that in the twenty years from 1980 to 2000, significant advancements were made in the description of rainfall fields and knowledge of their statistics, but at the expense of progress in the accuracy of the forecasting methods (Smith and Austin 2000). The TREC/COTREC forecasting procedure was again used to generate nowcasts of convective and stratiform events in mountainous terrain and to assess the predictability of forecasts made by the procedure. The results from this study show that convective cells are predictable using extrapolation algorithms for approximately 40 minutes while stratiform rain areas have potentially longer predictability times since they display a smaller degree of growth and decay (Mecklenburg *et al.* 2000). Very recently Seed (2003) used a variation on the correlation method to explore the effect of decomposition in the spatial and spectral domains on forecast skill. This test which used a global advection velocity for rainfall observed by a single radar, showed that smoothing processes can improve rainfall field RMSE over simple extrapolation techniques (Seed 2003).

3.3 APEX Algorithm Description

3.3.1 Overview

Advances in nowcasting over the last 40 years have provided a strong base for development of new forecasting methods. The APEX algorithm utilizes some of these advancements to produce very short-term radar rainfall nowcasts. The generation of these forecasts is done using correlation analysis combined with spatial filtering to produce a gridded velocity field for the advection of rainfall fields. These goals are achieved via a suite of MATLAB functions designed specifically for this reason. Model development was done extensively using the MATLAB image processing toolbox to decrease computation time, increase the efficiency of execution and reduce source code complexity.

APEX, like other extrapolation forecasting algorithms uses a steady state assumption as its theoretical base. Applied to rainfall forecasting, this assumption implies that rainfall intensities and spatial patterns will remain the same over short forecasting intervals. By applying a static velocity field to this static rainfall field, linear forecasts are created via a process termed "Lagrangian Persistence" since the rainfall field is persisted in a Lagrangian coordinate system (Germann and Zawadzki 2002). This differs from "Eulerian Persistence", in which the rainfall field is persisted in a stationary frame of reference. Eulerian Persistence will be used as a basis for comparison for the Lagrangian Persistence method presented here.

The APEX forecasted rainfall rate at a point (x_F, y_F) can be simplistically described by

$$\Psi_F\left(t_0+\tau, x_F, y_F\right) = \Psi_I\left(t_I, x_I, y_I\right) \tag{3.1}$$

$$x_F = x_I + \tau * u\left(x_I, y_I\right) \tag{3.2}$$

$$y_F = y_I + \tau * v\left(x_I, y_I\right) \tag{3.3}$$

(adapted from Germann and Zawadzki 2002). Where Ψ_F is the forecasted rainfall field, τ is the forecast lead time, Ψ_I is the initial rainfall field, (x_I, y_I) is the location in the initial field where the pixel to be advected is located, and $u(x, y)$ and $v(x, y)$ are the velocity fields for the east-west and north-south directions respectively. The challenge this nowcasting method must overcome is the derivation of the velocity fields and their application to rainfall fields to create short term forecasts.

This algorithm is governed by a set of parameters read into the model from an external text file. These parameters control many aspects of the execution of the code, including the size of the spatial filter, several thresholds and the range in which to search for a maximum correlation coefficient. The parameters and a set of nominal values can be seen in Table 3.1. Algorithm execution requires two input arguments

Parameter Name	**Nominal Value**	**Units**
FILTER_ROWS	41	pixels
FILTER_COLS	41	pixels
MAX_SHIFT	10	pixels
CORR_BOXSIZE	7	pixels
X_RES	4762.5	meters
Y_RES	4762.5	meters
TIME_SPACING	15	minutes
ANGLE_TOL	30	degrees
RAIN_THRESH	0.1	mm
WX_MIN	0.25	fraction
RANGE	11	pixels
SIGMA	1.5	pixels
Z_THRESH	0.55	fraction
SPEED_LIMIT	5	pixels
SEARCH_RADIUS	3	pixels

Table 3.1: Customizable parameters and a set of nominal values based on the spatial and temporal resolution shown. Many parameters are resolution dependent and should be changed accordingly.

and returns a single output argument. The input parameters are the full path file name of a "batch file" that lists the input files to be used in the forecast generation and the full path directory name where the forecast file are to be written to. Currently the single output is an array of Critical Success Index (CSI, see Section 2.5) scores, detailing average forecast accuracy as a function of lead time. The flow of data within the APEX algorithm can be seen in Figure 3-1 and a detailed description of the functions performed by each MATLAB file can be found in Appendix A

3.3.2 Input Field Filtering

Chapter 2 showed the benefits of spatial smoothing using a tested extrapolation-based nowcasting method. The algorithm presented here utilizes results from that work and applies the filtering process to a new nowcasting algorithm. Prior to any analysis on the input rainfall fields, the two-dimensional fields are subjected to spatial smoothing, in the form of a two dimensional low-pass averaging filter. The implementation is via a two dimensional convolution between the input field and the filter. The effect of the filtering is to remove the high frequency components of the rainfall field, usually small scale, high intensity convective cells, which improves the large scale advection of the storm system as seen in Chapter 2. In this algorithm, as in the GDST method, filtering is done prior to correlation analysis. However, the velocities are applied to the unfiltered rainfall field in the advection process.

3.3.3 Global Motion Analysis

The next step in the algorithm is an analysis of the mean storm motion. The second input file (t_0) is correlated with the first input file ($t_0 - t_S$, where t_S is the temporal resolution of the input files) to generate a global displacement (details of correlation based motion analysis can be found in Section 3.3.4). This displacement represents the mean displacement of the entire field during t_S. The returned displacement is later used for three purposes:

1. Biasing local area correlation (LAC) analysis search areas.
2. Eliminating velocities with large deviations from the global displacement in a quality control module (Section 3.3.6)
3. Forecasting with a uniform velocity applied to all points. This means of advection will be referred to as global advection.

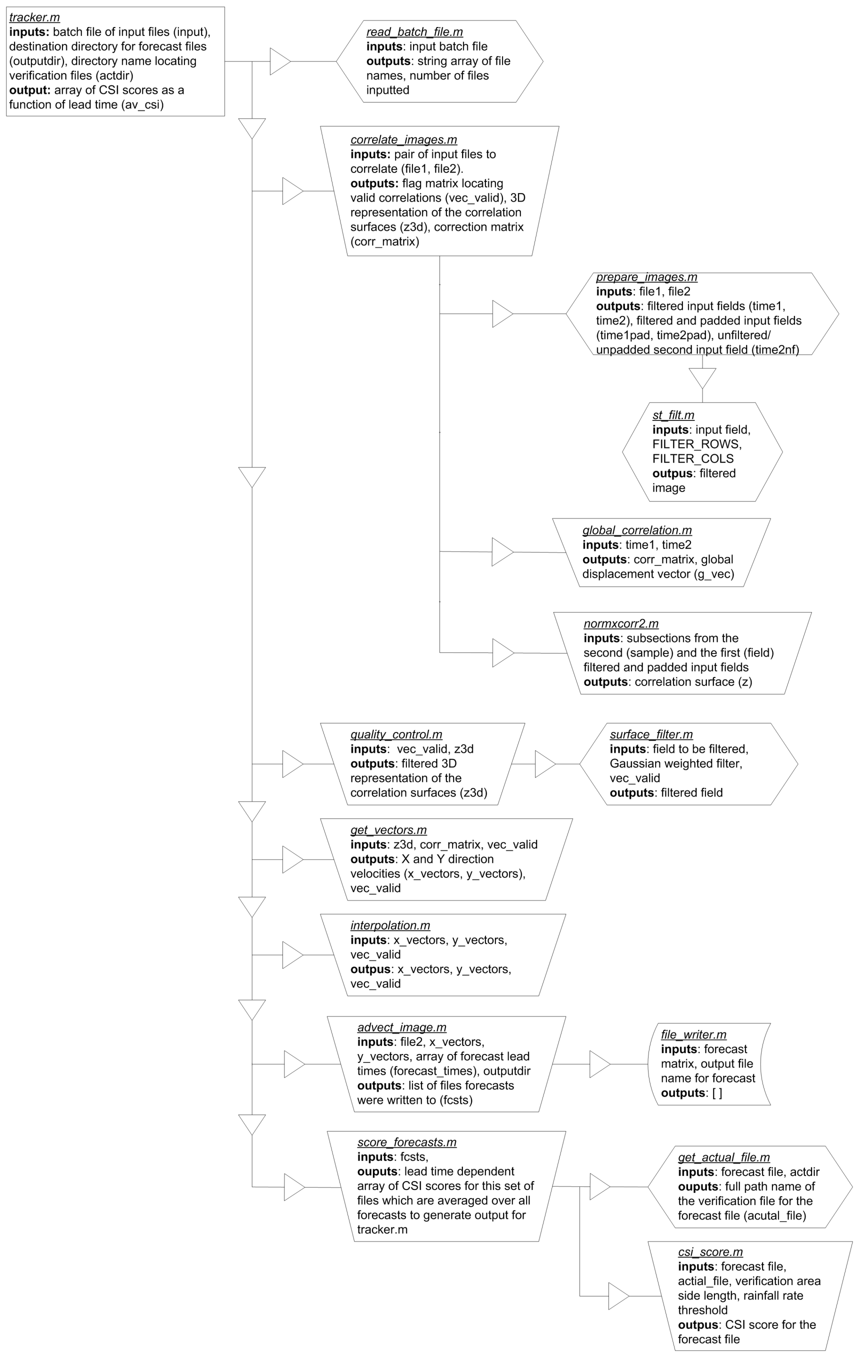
tracker.m
inputs: batch file of input files (input), destination directory for forecast files (outputdir), directory name locating verification files (actdir)
output: array of CSI scores as a function of lead time (av_csi)
read_batch_file.m
inputs: input batch file
outputs: string array of file names, number of files inputted
correlate_images.m
inputs: pair of input files to correlate (file1, file2).
outputs: flag matrix locating valid correlations (vec_valid), 3D representation of the correlation surfaces (z3d), correction matrix (corr_matrix)
prepare_images.m
inputs: file1, file2
outputs: filtered input fields (time1, time2), filtered and padded input fields (time1pad, time2pad), unfiltered/ unpadded second input field (time2nf)
st_filt.m
inputs: input field, FILTER_ROWS, FILTER_COLS
outpus: filtered image
global_correlation.m
inputs: time1, time2
outputs: corr_matrix, global displacement vector (g_vec)
normxcorr2.m
inputs: subsections from the second (sample) and the first (field) filtered and padded input fields
outputs: correlation surface (z)
quality_control.m
inputs: vec_valid, z3d
outputs: filtered 3D representation of the correlation surfaces (z3d)
surface_filter.m
inputs: field to be filtered, Gaussian weighted filter, vec_valid
outputs: filtered field
get_vectors.m
inputs: z3d, corr_matrix, vec_valid
outputs: X and Y direction velocities (x_vectors, y_vectors), vec_valid
interpolation.m
inputs: x_vectors, y_vectors, vec_valid
outpus: x_vectors, y_vectors, vec_valid
advect_image.m
inputs: file2, x_vectors, y_vectors, array of forecast lead times (forecast_times), outputdir
outputs: list of files forecasts were written to (fcsts)
file_writer.m
inputs: forecast matrix, output file name for forecast
outputs: [ ]
score_forecasts.m
inputs: fcsts,
ouputs: lead time dependent array of CSI scores for this set of files which are averaged over all forecasts to generate output for tracker.m
get_actual_file.m
inputs: forecast file, actdir
ouputs: full path name of the verification file for the forecast file (acutal_file)
csi_score.m
inputs: forecast file, actial_file, verification area side length, rainfall rate threshold
outpus: CSI score for the forecast file

Figure 3-1: Flowchart that shows APEX model structure and data transfer pathways.

3.3.4 Correlation Analysis

The most important step in this algorithm is the LAC analysis. This process takes small sub-images of the first and second input fields and produces a field of lag cross correlation coefficients for each pair of sub-images. To conserve computation resources, lag cross correlation surfaces are only computed for points that show rainfall amounts greater than zero in the second image and have more than a certain percentage (*WX_MIN*) of "rainy pixels" in both sub-images. The GDST uses a block processing method for their correlation process, where consecutive sub images are spaced so that there is no overlap. While this conserves computational resources, it requires a large amount of interpolation to generate a full velocity field. In this algorithm, however, a sliding neighborhood processing method is used which generates a correlation surface for sub-images centered on each pixel. The increased computational expense is offset by the benefit of generating velocity values at each point.

Several variations of the correlation coefficient exist (see Giachetti 2000 for a review), and for this model the zero-mean, variance normalized correlation coefficient was chosen. This is the optimal choice for this application since the resultant coefficient field is independent of variations in intensity and in the spatial gradient of the intensity due to normalization by the mean and the variance (Sun 2002). The coefficient ($\gamma(i,j)$) is a function of the lags in x- and y-directions, i and j respectively, and is computed from Equation 3.4.

$$\gamma(i,j) = \frac{\sum_{x,y}\left[S_{t_0-t_S}(x,y) - \overline{S_{t_0-t_S}}\right]\left[S_{t_0}(x-i,y-j) - \overline{S_{t_0}}\right]}{\left\{\sum_{x,y}\left[S_{t_0-t_S}(x,y) - \overline{S_{t_0-t_S}}\right]^2 \sum_{x,y}\left[S_{t_0}(x-i,y-j) - \overline{S_{t_0}}\right]^2\right\}^{0.5}} \tag{3.4}$$

In Equation 3.4 $S_{t_0-t_S}$ refers to the sub-image from the first input file (the search field), S_{t_0} refers to the sub-image from the second file (the sample template), the overbar notation indicates the mean value of the respective sub-image and x and y are horizontal and vertical coordinates within the sub-images. This is a computationally expensive calculation due to the local averaging process. The *normxcorr2* function in

MATLAB's image processing tool box seeks to minimize the computation time using Fourier transforms, fast local-sum algorithms and convolution computations. This frequency domain computation is often faster than the corresponding spatial domain computation.

To ensure that correlation coefficients can be computed for all lags, the sub-image from the first image is padded with zeros. However, this padding causes the resultant correlation surface to have border areas showing large artificial gradients where the padded values were used. These more distant correlation coefficients, and artificially smaller values are eliminated from further analysis, saving computation time in data transfer and further processing. Computational efficiency is also increased by only using a small subsection of the limited correlation surface when searching for the maximum correlation coefficient value. This subsection is found by taking the subsection of the correlation surface surrounding the end point of the global displacement vector. The size of this sub-image is a function of the *SEARCH_RADIUS* parameter. The result of this process is a predisposition for the velocities to be similar to the global velocity.

3.3.5 Correlation Surface Filtering

The individual, globally biased, limited area correlation surfaces found via cross correlation analysis can display many problems. One such problem is a high degree of variability in the relative location of the maxima over small spatial scales. To mitigate this variability, correlation meta surface (CMS) filtering was implemented. A CMS is a collection of individual correlation surfaces within a specified range of a pixel. The filtering process biases each correlation surface (CS) to be similar to neighboring CS, thus eliminating vectors that deviate widely from the local average vector.

This process assumes that over short spatial scales the location of maximum correlation should occur in approximately the same position. However, random matches between the sub-images used as correlation inputs, cause fluctuations in the location of the maxima. This occurs most often in places where there is little variability in the rainfall fields. Distance-weighted filtering of the correlation surfaces has the potential to propagate similar CS shapes to neighboring correlation surfaces. A previous study (Chornoboy *et al.* 1994) used a multi-resolution averaging method for filtering, and a similar method was implemented in this algorithm.

Instead of a computationally intensive multi-resolution averaging process, a simple two dimensional Gaussian filter is applied to the CS points in the same relative position in the CS. For example, all of the values in the top left corner of CS within *RANGE* pixels of the central location are weight averaged to produce a smoothed value for the top left corner in the central CS. Assuming the maximum correlation coefficient (CC) value will be positive, CC values less than zero are not included in the averaging process. The result from this averaging is a set of correlation surfaces that are biased against large spatial variations between neighboring surfaces. Figure 3-2 shows CMS and CS fields before (Figures 3-2a, c and e) and after (Figures 3-2b, d and f) filtering from an actual storm case (5 October 1998). Figures 3-2a and b show the CMS surfaces that influence the filtered value of the center CS, Figures 3-2c and d show the neighboring CS surrounding the central CS and Figures 3-2e and f show the central CS before and after filtering. In Figure 3-2b, the increase in the white area, where the correlation coefficient equals zero, is due to the elimination of all CC values less than *Z_THRESH*. In this case the benefits to the CS can be seen in the increase in range of the correlation values and the decrease in area that has a correlation coefficient above the highest contour.

3.3.6 Quality Control

Even after correlation meta-surface filtering, inferior quality displacement vectors still exist and will have adverse impacts on forecast accuracy. Identification of these invalid vectors is done according to three criteria.

1. Vectors whose directional deviation from the global angle (see Section 3.3.3) is greater than *ANGLE_TOL*.

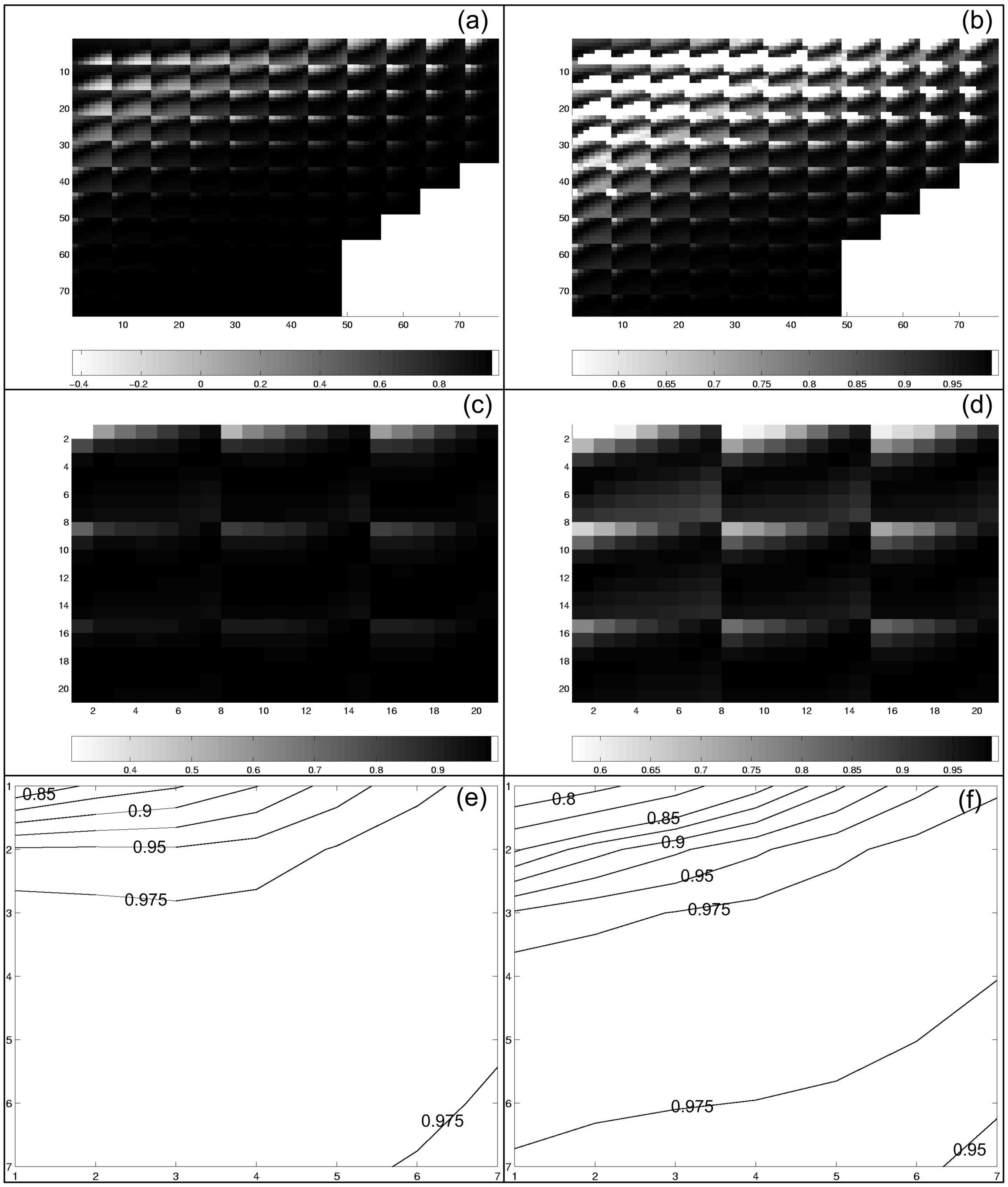

Figure 3-2: Example of correlation surface filtering. Panel (a) shows the correlation meta surface that influences the central correlation surface seen in panel (e), while panel (c) shows the neighboring correlation surfaces of the central correlation surface. Panel (b) shows the result of the correlation surface filtering on the area of influence, while panel (d) shows the filtered neighbors of the central correlation surface and panel (d) shows the filtered central correlation surface.

2. Vectors with magnitudes ($r = \sqrt{u^2 + v^2}$) exceeding *SPEED_LIMIT*.

3. Maximum correlation coefficients less than the *Z_THRESH* parameter.

If any of the above criteria are satisfied, the x- and y-direction displacements are eliminated and flagged to be filled in an interpolation step.

3.3.7 Vector Interpolation

Interpolation is an important step in the vector generation process due to the elimination of vectors in the quality control module. Prior to interpolation, the x- and y-direction pixel displacements are converted into velocities with units of meters per second. This conversion allows for the vectors to be displayed in a manner that has a well defined physical meaning. The interpolation of vectors is confined to eliminated vectors where the second input file had rainfall greater than zero. This decreases computation time and eliminates the implication that clear air velocities can be determined via this algorithm. Each pixel to be interpolated is replaced with the local average of the valid velocity values within a *RANGE* by *RANGE* area around the pixel in question. This also smoothes the final vector field, which aids in eliminating sharp changes in the displacements over short spatial scales.

3.3.8 Advection and Forecast Generation

Once the velocity fields have been generated and processed, they are then applied to the second of the two input fields to create forecasts. This is done with the knowledge that the velocity fields represent the displacement between two consecutive files and the assumption that forecasting is done in multiples of that temporal spacing. First, the initial file is advected over one forecast interval. This is done by taking each input pixel and moving it forward by its displacement value. To eliminate "holes" in the resulting forecast field, each input pixel is advected with its eight neighboring input pixels to the area centered on the forecasted location of the central pixel after advection. This is done for all input pixels by summing all rainfall values "put" in each forecast pixel. The number of values "put" into each forecast pixel is counted and after all input pixel fields have been advected, each final forecast pixel is computed as the average of all the input pixels that were "put" into that location. The same procedure is then done on the x- and y displacement fields so that each forecast pixel also has displacement values advected with it. The next forecast is made using the same procedure but uses the previous forecast as the initial image and the advected displacements as the motion field. Figure 3-3 shows a small sample area being advected using the process described above. The 16 pixel image on the left is the field being forecasted while the three smaller squares on the right are the values after advection into their new position. The arrows depict the displacement vectors used to advect the field forward, and their patterns correspond with the boxes containing the advected values. The shaded areas are the areas where averaging will occur resulting from the three vectors shown.

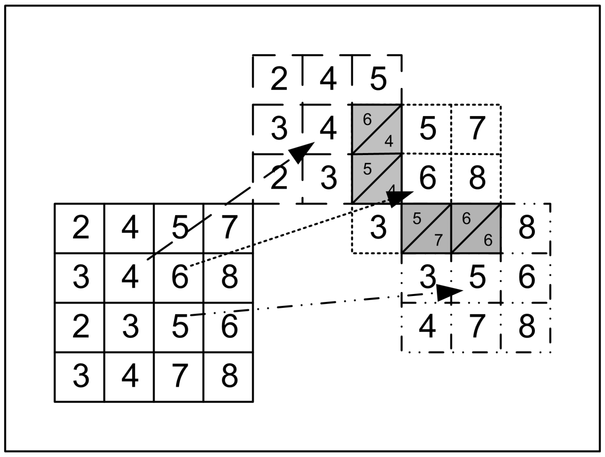

Figure 3-3: Advection scheme used in APEX nowcasting method.

This procedure is based on the assumption that a forecast for twice the original forecast interval is the single interval forecast field advected forward again and a three interval forecast is the result of three consecutive single interval forecasts of the initial field and the subsequent forecasted fields. The benefit of this procedure is that it allows the "holes" in the forecasted field to be filled at each step and ensure that they are not propagated into the next forecast interval. In the absence of uniform advection, “holes” in the forecast image expand with each forecast interval, so by eliminating the holes at each step, these large regions of missing data are not created at long lead times. The method used here for eliminating holes is an intelligent method and maintains the influence of the initial field on the forecast field throughout the forecasting period.

Chapter 4

APEX Results and Comparisons

4.1 APEX Outputs

Like the GDST algorithm, the main result from the APEX method is a series of forecast files. These files are written into a directory specified by the user and are easily incorporated into MATLAB or ArcView programs for further analysis. The first step in comparing the two nowcasting algorithms is to examine the differences between comparable velocity and forecast fields. Figure 4-1 performs this comparison. Figures 4-1a and b show the velocity fields surrounding a point near the leading edge of the storm envelope for the APEX and GDST algorithms, respectively. Figures 4-1c and d show the 60-minute forecast fields valid at 1300 UTC on 5 October 1998 for the APEX and GDST algorithms, respectively. At first glance, the largest difference between the velocity fields is that the APEX-generated field is not full. This is due to the constraint that only points where the rainfall field to be advected is greater than zero are put through the correlation analysis. Also, unlike the GDST, the APEX algorithm does not interpolate to all points in the range and therefore does not imply that clear air velocities can be determined using radar rainfall inputs. The high degree of smoothness seen in the GDST velocity field (Figure 4-1b) is also a consequence of the interpolation process used to fill the field.

The forecast fields show some differences that are indicative of the different degree of smoothness seen in the velocity fields. The APEX velocity field is not very smooth and because of that, the special advection algorithm described in Section 3.3.8 was implemented. A side effect of this advection algorithm is that the exact pattern and shape of the rainfall field is not preserved in forecasting. The GDST however, due to a very smooth velocity field, is able to preserve the pattern and shape of the rainfall field almost exactly over the forecasting interval. A potential drawback to the APEX advection algorithm is that the stratiform areas of the rainfall field appear to widen while the areas of more intense rainfall appear to narrow over the forecasting interval. This result may be due to erroneous velocity vectors produced by the algorithm for areas with small gradients in rainfall, a potential difficulty for correlation matching. The highest intensity rainfall appears to be advected to approximately the same location by both methods.

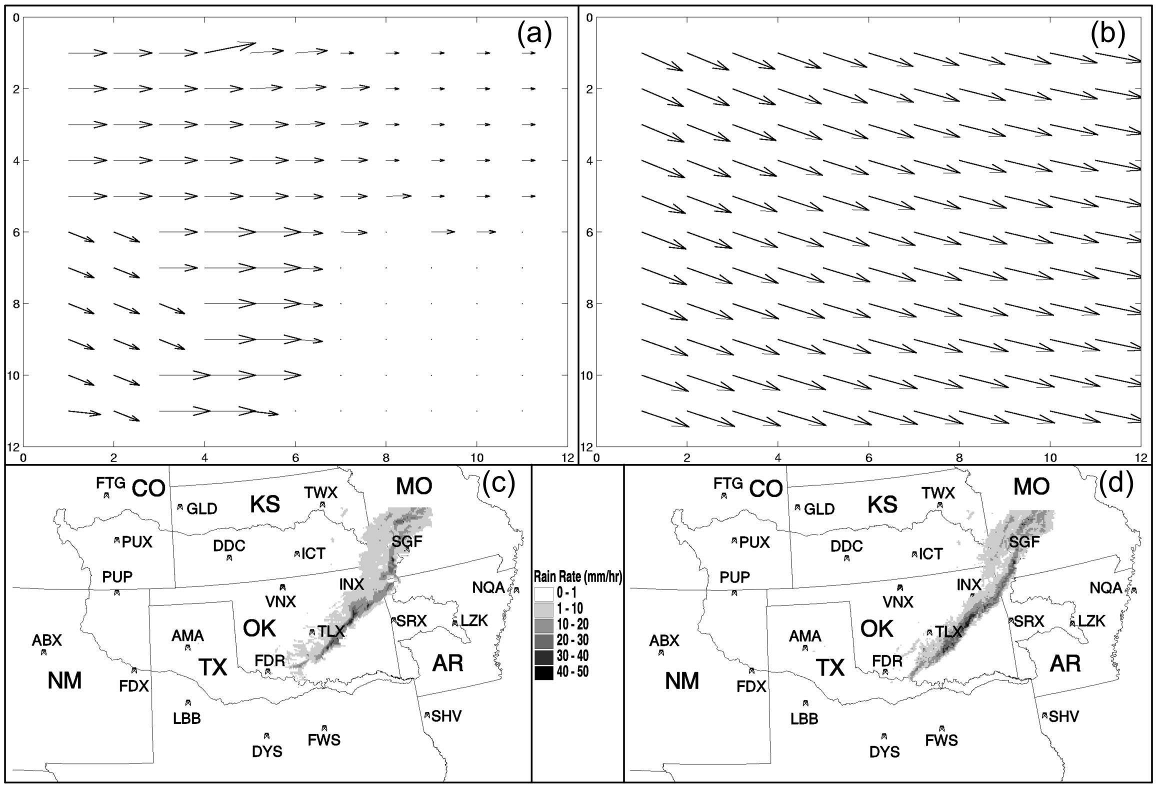

Figure 4-1: Sample outputs from the APEX algorithm (panels (a) and (c)) as compared to to the corresponding outputs from the GDST algorithm (panels (b) and (d)). the velocity vectors shown are a subset of the entire field and are a section located near the leading edge of the storm envelope. The rainfall fields are comparable 60-minute forecasts valid at 1300 UTC.

4.2 Lead Time Comparisons

The inclusion of the APEX algorithm in the comparison seen in Chapter 2.6, shows the improvements and shortcomings of the model. Figure 4-2 shows this comparison for the low rainfall threshold for all storm events as a function of lead time. As seen previously, persistence forecasting scores much lower than any of the advection methods while the GDST and the uniform advection scores are very similar over all lead times. For all three storm events, the APEX method scores higher than any of the other methods at short lead times and slightly lower then the GDST and uniform advection at longer lead times. Part of the explanation for the lower scores at longer lead times may lie in the advection algorithm used by the APEX method. The averaging that occurs in the velocity vectors and in the forecast fields during the advection step may change the resultant fields from the initial conditions in a manner that does not capture the correct evolution of the quantities over the forecast period. Similar comparisons using the median and high thresholds (not shown) show similar differences between the forecasting methods.

4.3 Verification Area Comparisons

When examining the 60-minute forecasts from these four methods further, more detail into the performance of the APEX algorithm can be seen. Figure 4-3 shows the impact that an extended verification kernel has on the CSI scores from the four forecasting methods. For all three storm events, the accuracy of the APEX forecasts improve at a higher rate than do forecasts generated by the GDST. This would tend to indicate that there are many cases where a small error in the displacement was the cause of the lower CSI score. Also seen for all storm events is that at almost all thresholds, the APEX algorithm scores as well or better than any of the other forecasting methods. The magnitude of this difference is storm event and verification area dependent. The results from the 120-minute forecasts, seen in Figure 4-4 show similar results except that the increase in accuracy by the APEX algorithm over the

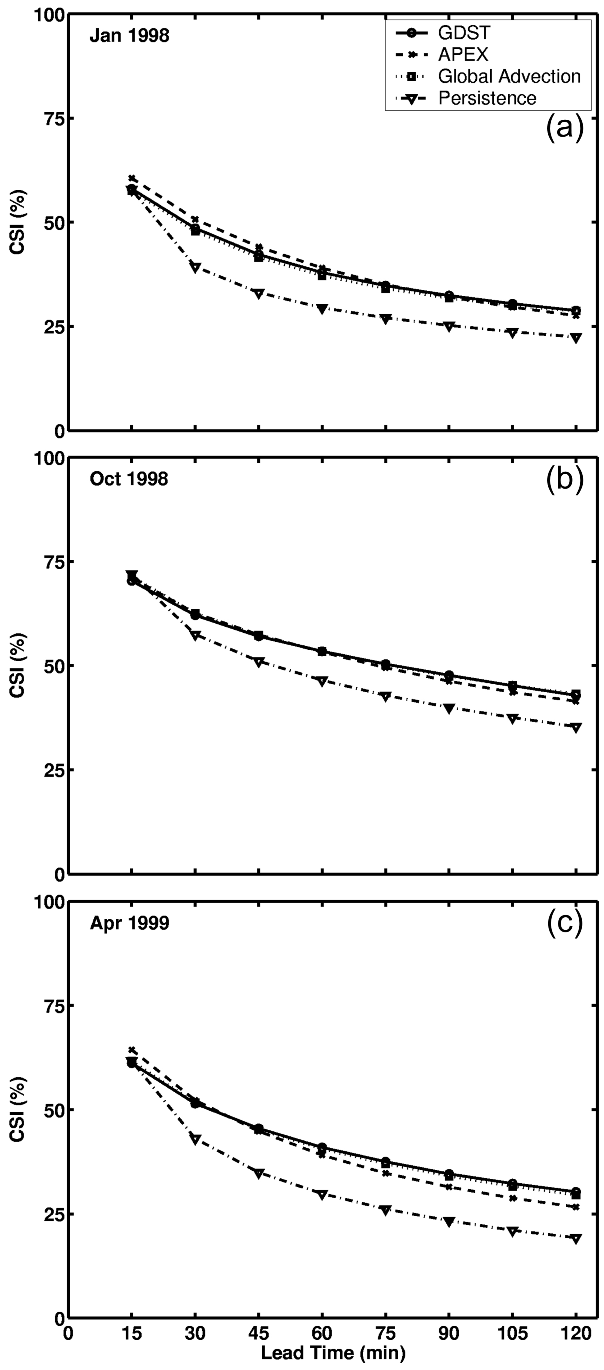

Figure 4-2: Comparison of the GDST (solid line), APEX (dashed line), global advection (dotted line) and persistence (dash-dotted line) CSI scoring as a function of lead time.

GDST algorithm for large verification areas is not as large as seen in the 60-minute forecasts.

4.4 False Alarm Ratio Comparisons

The false alarm ratio (FAR) is a statistic that can be computed from the same contingency table the CSI is derived from. The statistic indicates the percentage of total forecasted pixels that correspond to locations where no rainfall was observed. A FAR of 0 is the optimal score while a FAR of 1 indicates that all of the pixels in the forecast image were in error. In terms of the elements of Table 2.2, the FAR is computed using Equation 4.1.

$$FAR = \frac{FA}{H + FA} \tag{4.1}$$

For the storm cases examined in this study, the FAR analysis results show a different dynamic between the forecasting methods than seen with other statistics. Figure 4-5 shows the values of the FAR as a function of lead time for the different forecasting methods for all three storm events. Unlike the CSI scoring, the APEX algorithm appears to perform more like persistence forecasting when analyzed using the FAR, while the GDST and the uniform advection algorithm continue to perform in a very similar manner. The high FAR scores for APEX may be a consequence of many items. First, an error in the velocity estimates could cause an increase in the FAR if the velocity magnitude or direction do not correspond with the observations. Second, and very likely, the advection algorithm used in APEX causes some degree of smoothing of the forecast field. This smoothing may have the effect of placing rainfall outside of the observed rainfall envelope, thus increasing the number of forecast pixels with a rainfall rate greater than a threshold while the observations do not meet that criterion.

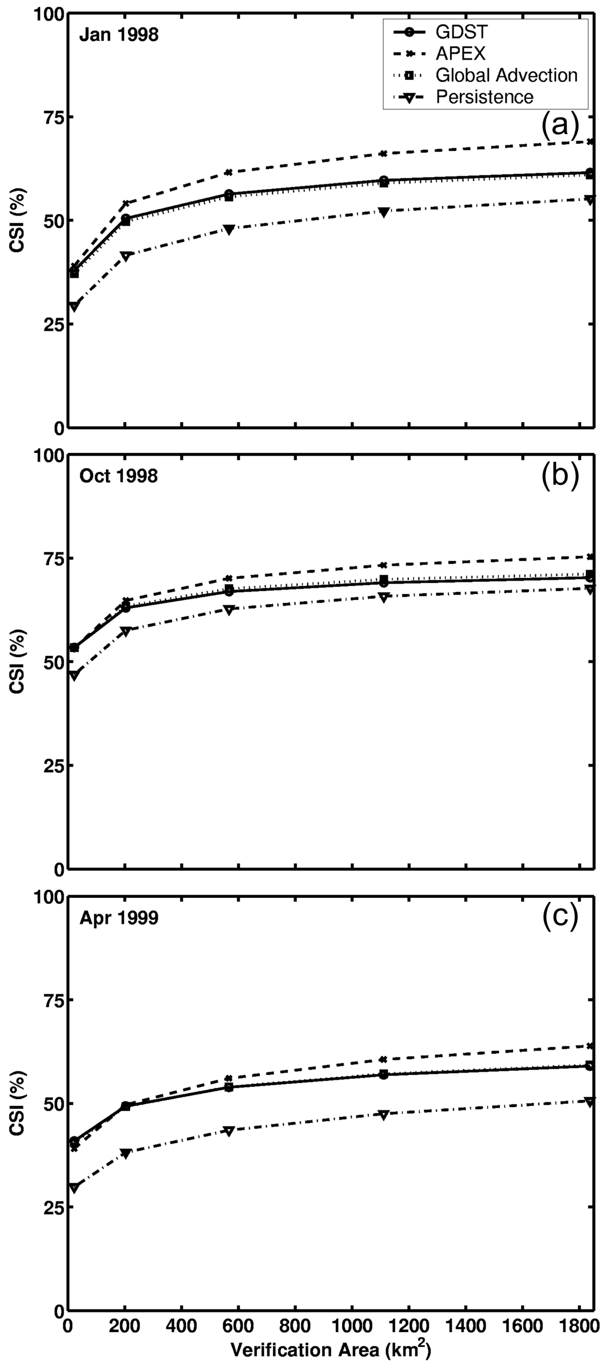

Figure 4-3: Comparison of the GDST (solid line), APEX (dashed line), global advection (dotted line) and persistence (dash-dotted line) CSI scoring as a function of verification area for their respective 60-minute forecasts.

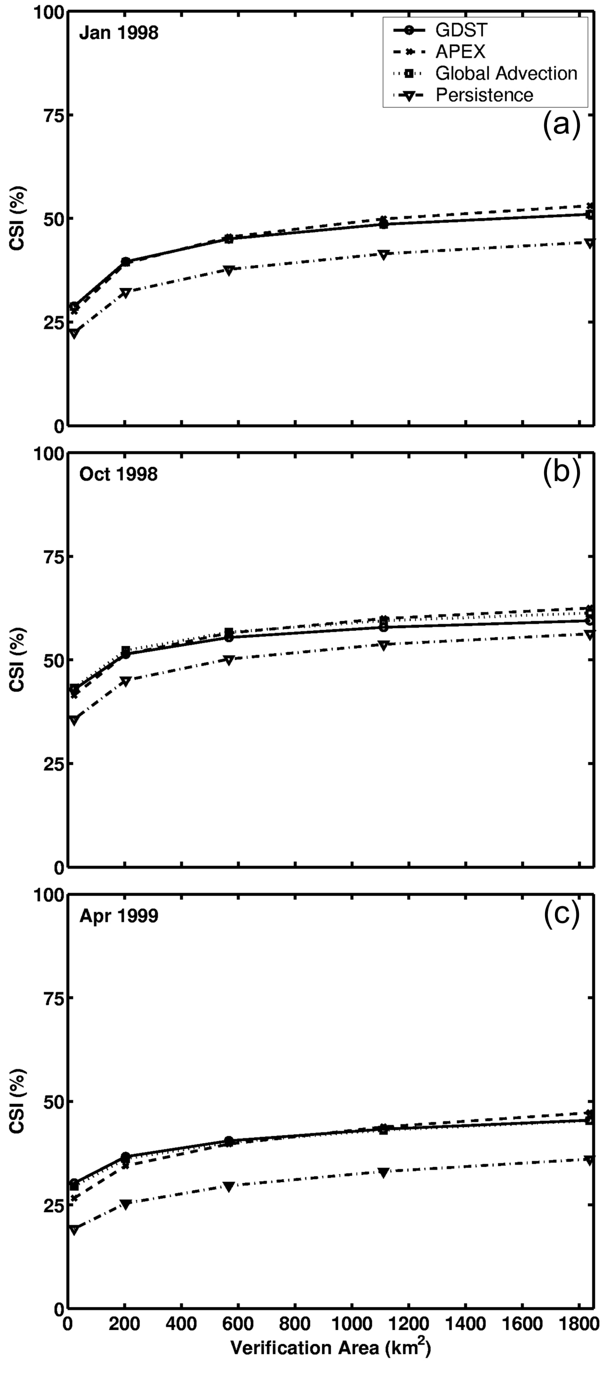

Figure 4-4: Comparison of the GDST (solid line), APEX (dashed line), global advection (dotted line) and persistence (dash-dotted line) CSI scoring as a function of verification area for their respective 120-minute forecasts.

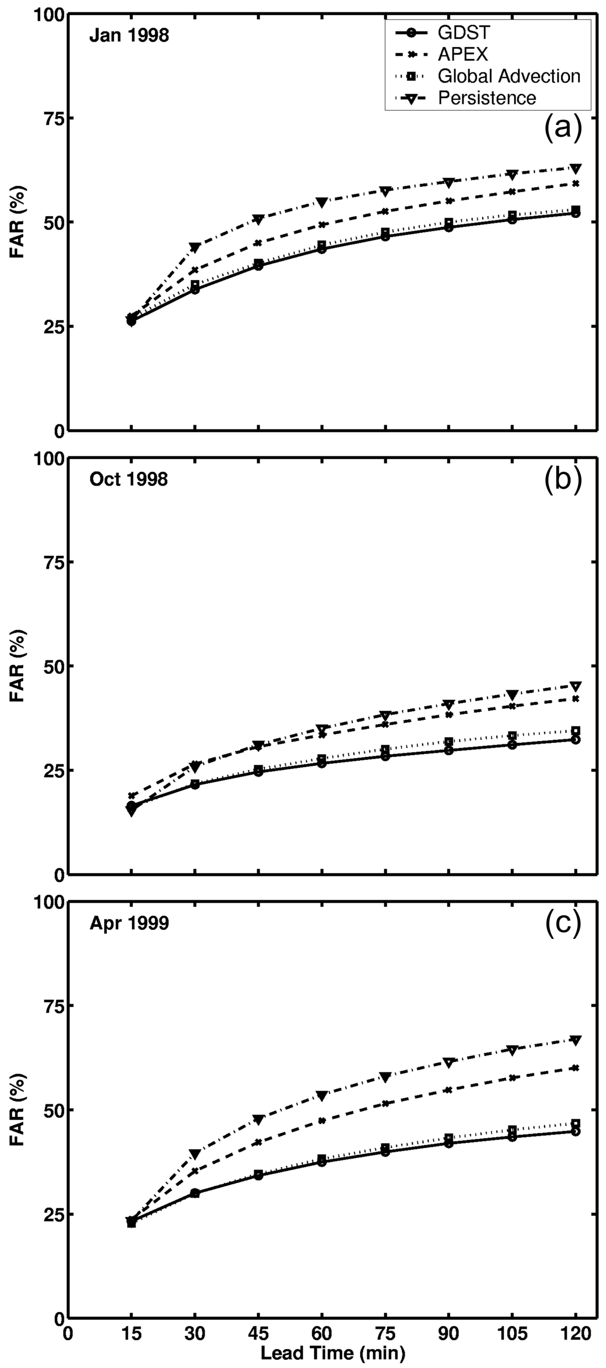

Figure 4-5: Comparison of the GDST (solid line), APEX (dashed line), global advection (dotted line) and persistence (dash-dotted line) FAR scoring as a function of lead time.

Chapter 5

Conclusions

This work addressed several issues that had yet to be addressed in the nowcasting literature. First, in Chapter 2 the forecast improvement as a result of including image filtering to eliminate small scale features prior to correlation was established. That section showed that the erroneous vectors caused by the correlation of images containing highly perishable features degraded the accuracy of the forecasts. The nowcasting method used in this determination was the MIT Lincoln Laboratory Growth and Decay Storm Tracker (GDST), an operational short-term rainfall forecasting algorithm. A relatively simple method for their removal was implemented and the results showed that for lead times from 15 to 120 minutes, forecast accuracy was improved for the three storm cases examined here. Chapter 2 also examined the benefits of deriving spatially variable velocity vectors as opposed to using a single velocity for the entire rainfall field. The gain in accuracy through this enhancement was less clear and more research is needed to ascertain the value of spatially variable velocity fields for nowcasting.

In Chapter 3 a new nowcasting algorithm was introduced and described. Utilizing concepts culled from multiple previous algorithms, the Automated Precipitation Extrapolator (APEX) seeks to combine the benefits from these prior methods to generate accurate very short-term rainfall forecasts using composite radar rainfall fields. Composite radar rainfall fields have the potential to provide a highly accurate input to nowcasting methods. Results from forecasts provided by the APEX algorithm can be seen in Chapter 4. Using a variety of evaluation measures, APEX-generated forecasts show approximately the same degree of accuracy as GDST and uniform advection forecasts show for the storm events analyzed here.

The goals and questions that this work set out to examine and achieve in Chapter 1 are restated here along with a progress evaluation for the goals and answers to the questions. First the goals and the progress.

Goals:

1. Goal: Determine the impact of the motion of small-scale features on forecast performance.

2. Goal: Identify the benefits of allowing for differential motion within a storm system in a nowcasting procedure.

3. Goal: Develop and test a MATLAB-based short-term forecasting procedure.

Progress:

1. **Accomplished through analysis of the benefits of image filtering.**

2. **Partially accomplished through comparison of GDST and uniform advection algorithm. More research is required to fully understand the impacts.**

3. **Accomplished with the development of the APEX algorithm.**

Questions:

1. Question: What impact does image filtering have on nowcasting accuracy?

2. Question: Do spatially variable velocities improve short-term forecast accuracy?

3. Question: How well does a new algorithm for nowcasting compare with simpler methods and an operational nowcasting algorithm?

Answers:

1. Image filtering has a positive impact of several CSI points on short-term nowcasts with the GDST for the three storm cases examined in this work.

2. There was small improvement seen for the GDST over a uniform advection method, but without further examination over more storm events, this question can not be answered conclusively.

3. The APEX algorithm performs approximately as well as the GDST and uniform advection methods and outperforms persistence forecasting. With further improvements and detailed parameter sensitivity testing, APEX may hold the potential to outperform the GDST nowcasting method also.

Bibliography

[1] Austin, G.L. and A. Bellon. The use of digital weather radar records for short-term precipitation forecasting. *Quarterly Journal of the Royal Meteorological Society*, 100:658–664, 1974.

[2] Bellon, A. and G.L. Austin. The evaluation of two years of real-time operation of a short-term precipitation forecasting procedure (SHARP). *Journal of Applied Meteorology*, 17:1778–1787, 1978.

[3] Bellon, A. and I. Zawadzki. Forecasting of hourly accumulations of precipitation by optimal extrapolation of radar maps. *Journal of Hydrology*, 157:211–233, 1994.

[4] Brémaud, P.J. and Y.B. Pointin. Forecasting heavy rainfall from rain cell motion using radar data. *Journal of Hydrology*, 142:373–389, 1993.

[5] Browning, K.A. Local weather forecasting. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, A371:179–211, 1980.

[6] Browning, K.A. and C.G. Collier. Nowcasting of precipitation systems. *Reviews of Geophysics*, 27(3):345–370, 1989.

[7] Browning, K.A., C.G. Collier, P.R. Larke, P. Menmuir, G.A. Monk, and R.G. Owens. On the forecasting of frontal rain using a weather radar network. *Monthly Weather Review*, 110:534–552, 1982.

[8] Cartwright, T.J., M.M. Wolfson, B.E. Forman, R.G. Hallowell, M.P. Moore, and K.E. Theriault. The FAA terminal convective weather forecast product:

scale separation filter optimization. In *Twenty-ninth International Conference on Radar Meteorology*. American Meteorological Society, 1999.

[9] Chen, Zhi-Qiang and M.L. Kavvas. An automated method for representing, tracking and forecasting rain fields of severe storms by Doppler weather radars. *Journal of Hydrology*, 132:179–200, 1992.

[10] Chornoboy, Edward S., Anne M. Matlin, and John P. Morgan. Automated storm tracking for terminal air traffic control. *The Lincoln Laboratory Journal*, 7(2):427–447, 1994.

[11] Ciccione, Monique and Vincent Pircher. Preliminary assessment of very short term forecasting of rain from single radar data. In *Proceedings of Nowcasting II*, pages 241–246. European Space Agency, 1984.

[12] Dixon, Michael and Gerry Wiener. TITAN: Thunderstorm identification, tracking, analysis, and nowcasting - a radar-based methodology. *Journal of Atmospheric and Oceanic Technology*, 10(6):785–797, 1993.

[13] Donaldson, R.J., Jr, R.M Dyer, and M.J. Kraus. An objective evaluator of techniques for predicting severe weather events. In *Preprints: Ninth Conference on Sever Local Storms*, pages 321–326. American Meteorological Society, 1975.

[14] Dupree, W.J., R. Johnson Jr., M.M. Wolfson, K.E. Theriault, B.E. Forman, R.A. Boldi, and C.A. Wilson. Forecasting convective weather using multi-scale detectors and weather classification - enhancements to the MIT Lincoln Laboratory Terminal Convective Weather Forecast. In *Tenth Conference on Aviation, Range and Aerospace Meteorology*. American Meteorological Society, 2002.

[15] Einfalt, Thomas, Thierry Denoeux, and Guy Jacquet. A radar rainfall forecasting method designed for hydrological purposes. *Journal of Hydrology*, 114:229–244, 1990.

[16] Evans, James E. and Elizabeth R. Ducot. The integrated terminal weather system (ITWS). *The Lincoln Laboratory Journal*, 7(2):449–473, 1994.

[17] Forman, B.E., M.M. Wolfson, R.G. Hallowell, and M.P. Moore. Aviation user needs for convective weather forecasts. In *Eighth Conference on Aviation, Range and Aerospace Meteorology*. American Meteorological Society, 1999.

[18] Germann, Urs and Isztar Zawadzki. Scale-dependence of the predictability of precipitation from continental radar images. Part I: Description of the methodology. *Monthly Weather Review*, 130:2859–2873, 2002.

[19] Giachetti, A. Matching techniques to compute image motion. *Image and Vision Computing*, 18:247–260, 2000.

[20] Glickman, Todd S., editor. *Glossary of Meteorology*. American Meteorological Society, second edition, 2000.

[21] Grassotti, Christopher, Ross N. Hoffman, Enrique R. Vivoni, and Dara Entekhabi. Multiple timescale intercomparison of two radar products and rain gauge observations over the Arkansas-Red River basin. *submitted to Weather and Forecasting*, 2003.

[22] Grecu, M. and W.F. Krajewski. A large-sample investigation of statistical procedures for radar-based short-term quantitative precipitation forecasting. *Journal of Hydrology*, 239:69–84, 2000.

[23] Hallowell, R.G., M.M. Wolfson, B.E. Forman, M.P. Moore, B.A. Crowe, T.M. Rotz, D.W. Miller, T.C. Carty, and S.F. McGettigan. The terminal convective weather forecast demonstration at the DFW International Airport. In *Eighth Conference on Aviation, Range and Aerospace Meteorology*. American Meteorological Society, 1999.

[24] Hamill, Thomas M. and Thomas Nehrkorn. A short-term cloud forecast scheme using cross correlations. *Weather and Forecasting*, 8(4):401–411, 1993.

[25] Handwerker, Jan. Cell tracking with TRACE3D - a new algorithm. *Atmospheric Research*, 61:15–34, 2002.

[26] Hilst, G.R. and J.A. Russo Jr. An objective extrapolation technique for semi-conservative fields with an application to radar patterns. Technical Memo 3, The Travelers Weather Research Center, Inc., Hartford, CT, 1960. Contract AF30-635-14459.

[27] Johnson, J.T., Pamela L. MacKeen, Arthur Witt, E. DeWayne Mitchell, Gregory J. Stumpf, Michael D. Eilts, and Kevin W. Thomas. The storm cell identification and tracking algorithm: An enhanced WSR-88D algorithm. *Weather and Forecasting*, 13:263–276, 1998.

[28] Leese, John A., Charles S. Novak, and Bruce B. Clark. An automated technique for obtaining cloud motion from geosynchronous satellite data using cross correlation. *Journal of Applied Meteorology*, 10:118–132, 1971.

[29] Li, L., W. Schmid, and J. Joss. Nowcasting of motion and growth of precipitation with radar over a complex orography. *Journal of Applied Meteorology*, 34:1286–1300, 1995.

[30] Mecklenburg, S., J. Joss, and W. Schmid. Improving the nowcasting of precipitation in an Alpine region with an enhanced radar echo tracking algorithm. *Journal of Hydrology*, 239:46–68, 2000.

[31] Panofsky, Hans A. and Glenn W. Brier. *Some Applications of Statistics to Meteorology.* Pennsylvania State University, first edition, 1958.

[32] Pereira Fo., Augusto J., Kenneth C. Crawford, and David J. Stensrud. Mesoscale precipitation fields. Part II: Hydrometeorologic modeling. *Journal of Applied Meteorology*, 38:102–125, 1999.

[33] Pierce, C.E., P.J. Hardaker, C.G. Collier, and C.M. Haggett. GANDOLF: A system for generating automated nowcasts of convective precipitation. *Meteorological Applications*, 7:341–360, 2000.

[34] Rinehart, R.E. and E.T. Garvey. Three-dimensional storm motion detection by conventional weather radar. *Nature*, 273:287–289, 1978.

[35] Seed, A.W. A dynamic and spatial scaling approach to advection forecasting. *Journal of Applied Meteorology*, 42:381–388, 2003.

[36] Seo, D.-J. and J.A. Smith. Radar based short term rainfall prediction. *Journal of Hydrology*, 131:341–367, 1992.

[37] Smith, James A., Mary Lynn Baeck, Julia E. Morrison, Paula Sturdevant-Rees, Daniel F. Turner-Gillespie, and Paul D. Bates. The regional hydrology of extreme floods in an urbanizing drainage basin. *Journal of Hydrometeorology*, 3:267–282, 2002.

[38] Smith, K.T. and G.L. Austin. Nowcasting precipitation - a proposal for a way forward. *Journal of Hydrology*, 239:34–45, 2000.

[39] Sun, Changming. Fast optical flow using 3D shortest path techniques. *Image and Vision Computing*, 20:981–991, 2002.

[40] Theriault, K.E., M.M. Wolfson, B.E. Forman, R.G. Hallowell, M.P. Moore, and R.J. Johnson Jr. FAA terminal convective weather forecast algorithm assessment. In *Ninth Conference on Aviation, Range and Aerospace Meteorology.* American Meteorological Society, 2000.

[41] Tsonis, A.A. and G.L. Austin. An evaluation of extrapolation techniques for the short-term prediction of rain amounts. *Atmosphere-Ocean*, 19(1):54–65, 1981.

[42] Tuttle, John D. and G. Brant Foote. Determination of the boundary layer airflow from a single Doppler radar. *Journal of Atmospheric and Oceanic Technology*, 7:218–232, 1990.

[43] Wilson, James W., N. Andrew Crook, Cynthia K. Mueller, Juanzhen Sun, and Michael Dixon. Nowcasting thunderstorms: a status report. *Bulletin of the American Meteorological Society*, 79(10):2079–2099, 1998.

[44] Wilson, J.W. Movement and predictability of radar echoes. Final Report 7471-204, The Travelers Weather Research Center, Inc., Hartford, CT, 1966.

[45] Wolfson, M.M., B. E. Forman, R. G. Hallowell, and M. P. Moore. The growth and decay storm tracker. In *Eighth Conference on Aviation, Range and Aerospace Meteorology.* American Meteorological Society, 1999.

[46] Zawadzki, I. Statistical properties of precipitation patterns. *Journal of Applied Meteorology*, 12:459–472, 1973.

[47] Zawadzki, I., J. Morneau, and R. Laprise. Predictability of precipitation patterns: An operational approach. *Journal of Applied Meteorology*, 33:1562–1571, 1994.

[48] Zipser, E. Rainfall predictability: When will extrapolation-based algorithms fail? In *Eighth Conference on Hydrometeorology*, pages 138–142. American Meteorological Society, 1990.

Appendix A

APEX Users Guide

A.1 Parameter File

The parameter file is a text file that allows for flexibility in setting many parameters that govern the execution of the code. The parameter file should be named st.params and located in the same directory as the MATLAB source files. The parameters include thresholds, ranges, resolutions and limits used within the code. Many of the parameters are dependent on the spatial and temporal resolution of the files and should be adjusted accordingly. The following list names all the parameters and describes what they are used for within the code.

- FILTER_ROWS and FILTER_COLS: Sets the vertical and horizontal size of the filter used for scale separation filtering done prior to the local correlation analysis. The values of the filter size parameters should be odd numbers so the filter can be properly centered. Only used as input to the st_filt.m file. *Nominal value: 41 pixels*

- MAX_SHIFT: Component of the *field* parsed from the first input file. The *field* is a square with a side length of $2 * MAX\_SHIFT + 1$ whose center pixel is the pixel that the correlation surface and resulting vector displacement correspond to. The value of this parameter can be either even or odd. Only used in the correlate_images.m file. *Nominal value: 10 pixels*

- CORR_BOXSIZE: Size of the *sample* from the second input file that is correlated against the *field* (defined above) in the local correlation analysis. This number should be smaller than $2 * MAX\_SHIFT + 1$ and an odd number to produce a good correlation analysis and a sub-image that is centered on the pixel in question. Only used in the correlate_images.m file. *Nominal value: 7 pixels*

- X_RES and Y_RES: The horizontal resolution of the input files in meters. Used in the get_vectors.m and advect_image.m files. *Nominal value: 4762.5 meters*

- TIME_SPACING: The temporal resolution of the input files in minutes. Used in the get_vectors.m and advect_image.m files. *Nominal value: 15 minutes*

- ANGLE_TOL: Specifies the maximum deviation, in degrees, between a local vector and the global vector computed in the global correlation analysis. Used in the get_vectors.m file. *Nominal value: 30 degrees*

- RAIN_THRESH: A minimum filtered rainfall value allowed to persist through the code. Used in the prepare_images.m file. *Nominal value: 0.1 mm*

- WX_MIN: A minimum percentage of valid weather in the *sample* and *field* sub-images needed to perform a local correlation between the two sub-images. Used in the correlate_images.m file. *Nominal value: 0.25*

- RANGE: A distance parameter that is used in multiple instances to define areas for interpolation or filtering. Used as a side length in the correlation meta surface (CMS) filtering process and as a side length of an area of influence for vector interpolation. Occurs in the quality_control.m, surface_filter.m and interpolation.m files. *Nominal value: 11 pixels*

- SIGMA: Used in the generation of the two dimensional Gaussian filter employed for CMS filtering. In conjunction with the RANGE parameter, SIGMA serves as an input to the *fspecial* function found in the image processing tool box. Used in the quality_control.m file. *Nominal value: 1.5*

- Z_THRESH: Sets a minimum correlation coefficient value for a valid maximum. Used in the quality_control.m file. *Nominal value: 0.55*

- SPEED_LIMIT: Sets a maximum displacement for local vectors. Vectors with magnitudes over this value are eliminated and interpolated later. Used in the get_vectors.m file. *Nominal value: 5 pixels*

- SEARCH_RADIUS: Defines an area around the endpoint of the global vector in which to search for the maximum correlation coefficient. Is a component of the correction values, that turn displacements from the global vector endpoint into displacements from the center pixel. Used in the correlate_images.m and global_correlation.m files. *Nominal value: 3 pixels*

- INTERP_RADIUS: A component of the area used in the interpolation of the forecast files. Side lengths of the influential area are $2 * INTERP\_RADIUS + 1$ pixels long. Used in the interp_fcst.m file. *Nominal value: 2 pixels*

- INTERP_PCT: Sets the minimum percentage of non-zero weather that must be in the influential area for forecast interpolation. Used in the interp_fcst.m file. *Nominal value: 0.50*

A.2 Source File Description

The MATLAB code that performs the short-term, extrapolation-based forecasting procedure is composed of 18 *.m* files. The source code is found in Appendix B.

tracker.m

This file is the main file for the execution of the Storm Tracker algorithm. From this file, other files are called to provide support and outputs to pass on to other functions. This files takes a batch file of rainfall field text files as inputs, produces zero to two hour forecasts, in fifteen minute increments, and returns a vector of critical success index scores for the various lead times averaged over all the forecasted files. Actions taken within this file include reading of the parameter file and writing the averaged CSI scores into a file. The functions called directly from this function are read_batch_file, correlate_images, quality_control, get_vectors, interpolation, continuity, advect_image, and score_forecasts.

read_batch_file.m

This file takes the name of a batch file of full path file names as input and returns a MATLAB array of strings of the filenames and the number of files in the batch file.

correlate_images.m

This file performs the local area correlation analysis between two input files. Inputs to this file are strings containing the pair of file names to perform the correlation on and the four outputs are a binary field with values of one indicating that a valid correlation was performed, a three dimensional data structure containing the global vector biased, limited search area correlation surfaces for the valid correlations, a two element vector containing the correction values for turning a location in the global vector biased, limited search area correlation surface into a location relative to the pixel the surface is associated with and a binary field of ones indicating all the locations where correlation was performed for use in the vector interpolation file. The functions called directly from this file are prepare_images and global_correlation. Other acts performed by this file include a check to ensure that the the first of the two images is not a field of zeros, the calling of the *normxcorr2* function from the image processing tool box to perform the correlation between the sub-images of the input fields, and the clipping of the full correlation surface to create an output variable that is efficient to pass between functions.

prepare_images.m

This file pre-processes the input images prior to the correlation analysis step. The inputs are the strings containing the filenames to be processed and the outputs are the filtered fields, the filtered and padded fields, and the unfiltered second field. This function calls the st_filt function to perform the filtering on the two fields. Other processes of this function are the thresholding of the filtered fields, a check to ensure that the fields are the same size, and the padding of the fields by using the *padarray* function from the image processing tool box.

st_filt.m

This file filters a two dimensional field using an averaging filter. The inputs are the field to be filtered, the number of rows in the filter and the number of columns in the filter, while the output is the filtered field. The filtering is done using the *conv2* function from the image processing tool box.

global_correlation.m

This file performs a correlation analysis between the whole first and second images. Inputs to the file are the strings containing the first and second file names and the outputs are the correction array mentioned in the correlate_images.m file, and the Cartesian components of the global vector. The *normxcorr2* function provides the correlation surface of which the location of the maximum correlation coefficient is found. The displacement of this location from the center of the correlation surface is the global vector and from that vector the global angle and magnitude can be found and in conjunction with the SEARCH_RADIUS parameter the correction array is computed. No other functions are called from this file.

quality_control.m

This file performs the correlation meta surface filtering. Inputs to the file are the three dimensional correlation surface data structure and the binary field of valid correlation locations and the output is the filtered three dimensional correlation surface data structure. Using the *fspecial* function from the image processing tool box, a two dimensional Gaussian weighted filter is created than applied to the correlation surfaces. This file calls the surface_filter function to do the filtering on the surfaces. The surfaces that are filtered are created by taking all the pixels from the same relative positions in the correlation surfaces and smoothing them using the filter specified above. Correlation values less than Z_THRESH are eliminated prior to their output.

surface_filter.m

This file does the actual filtering of the correlation meta surfaces. The inputs are the field to be filtered, the filter weights, and the binary field of valid correlation locations while the single output is the filtered surface. To only focus on the correlation coefficient values that will yield a maximum, negative coefficients are eliminated prior to the filtering operation, and are not included in the weighted average computation. No other files are called in the execution of this file.

get_vectors.m

This file converts the globally biased, limited search area correlation surfaces into a field of displacement vectors for the pair of input files. Inputs to this file are the filtered three dimensional correlation surface data structure, the correction array, the binary field of valid correlation locations, and the binary interpolation field and outputs horizontal and vertical vector fields in meters per second as well as adjusted versions of the two input binary fields. Each valid correlation surface is analyzed for its maximum value and its location is converted into a displacement vector. Before conversion into a velocity, each vector is checked against the ANGLE_TOL and SPEED_LIMIT parameters to eliminate largely erroneous vectors. No other files are called from within this file.

interpolation.m

This file interpolates holes in the vector field caused by the elimination of vectors due to failing one of the quality control criteria. Inputs to this file are the horizontal and vertical vector fields and the binary interpolation field dictating which locations to fill with interpolation, and the outputs are the interpolated vector fields. This file only interpolates using valid vectors, which are composed solely of locations where the second image file had rainfall rates greater than zero. The remainder of the locations are forced to be not-a-number (NaN) and are not included in the calculation of the interpolated value. No other files are called within this file.

advect_image.m

This file applies the derived vector fields to the second input field and produces a series of forecast files. The inputs to this file are the file to be forecasted from, the horizontal and vertical velocity fields and a string of forecast times and outputs an array of strings that are the names of the files that the forecasts were written to. Displacements for each pixel are determined by multiplying the vector fields by the forecast lead time to generate a displacement appropriate for that lead time. The pixels from the input file are then advected to that new point using a "put" operation. Other functions called from this file are interp_fcst and file_writer.

interp_fcst.m

This file fills holes in the forecast field that are the result of the "put" forecasting operation. The raw forecast field is the input to this file and the filled forecast field is the output. By using an inverse distance squared interpolation algorithm, missing points in the forecast field whose areas of influence include more than INTERP_PCT percentage of pixels within the $2 * INTERP\_RADIUS + 1$ square surrounding the missing point are filled with the weighted average of the surrounding pixels. This file does not call any other functions.

file_writer.m

This file takes a two dimensional field and writes it into a text file on the hard disk. Inputs to this file are the field to be written and the full path name of the file to contain the field, and there are no outputs. This file does not call any other files.

score_forecasts.m

This file take a set of forecasts from a single input file and scores them using a critical success index comparison with the observed field from the valid time of the forecast. The input to this file is an array of strings that are the filenames of the forecast files that have been written to the hard disk and the output is an array of CSI scores for the difference lead times in the file. This file calls the get_actual_file and csi_score functions to find the observed file for comparison and to determine the CSI score between the two files.

get_actual_file.m

This file takes a file written by the file_writer function, which has its initial time as the first section of the file name and the lead time as the second section of the file name, and finds the observed file that is valid at the time that the forecast is valid. The input to this file is the string containing the forecast file name and the output is the string containing the full path location of the observed file needed for scoring. This file determines the change in minute, hour, and day by adding the lead time to the initial time but does not compute the change in month or year, however this would be relatively simple to incorporate. This file does not call any other files. The directory name where the observed files are stored should be changed prior to use with different storm events so the proper comparison can be made.

csi_score.m

This file takes a forecast file and its corresponding observed file and computes the CSI score between the fields. The inputs to this file are the strings containing the names of the forecast and observed files as well as the verification area size and the rainfall rate threshold. The output from this file is the CSI score, as a decimal, indicating how well the forecast scored. Aside from square windows, called for by entering a single number in the verification area size input, a “cross” or a three by five rectangle can be used by entering “cross” or “rect” respectively for the argument.

The extended verification area scoring is done via a smoothing process on the observed field prior to comparison. No other functions are called by this file.

Appendix B

MATLAB Source Code

B.1 tracker.m

```
function [av_csi] = tracker(input, outputdir, actdir)
% Main function for MATLAB based, short term extrapolation rainfall
% forecasting.  Takes a batch file of input files and generates a series of
% forecast files for each pair of initial files. Also requires directory
% names for the forecast files to be written to as well as the location of
% the verification files for comparison.  Calls correlate_images,
% quality_control, get_vectors, interpolation,  continuity and advect_image
% to do velocity vector generation, quality  control, interpolation,
% conversion, and image advection.  Files are written into the directory
% specified by outputdir.  Please use full directory/file names for all
% inputs.

% Initialize timer and close all open figure windows.
start = clock;
close all

% Declare Global Variables: Defined in file st.params
global FILTER_ROWS FILTER_COLS MAX_SHIFT CORR_BOXSIZE
global X_RES Y_RES TIME_SPACING ANGLE_TOL RAIN_THRESH
global WX_MIN RANGE SIGMA Z_THRESH SPEED_LIMIT SEARCH_RADIUS

% These global variable are defined within the following function calls
global TOTAL_ROWS TOTAL_COLS HALFSIZE THETA_GLOBAL
global MARGIN ANGLE_TOL_RAD

% Read parameters from external parameter file
[params, values] = textread('st.params', '%s %n', 'delimiter',...
    '\t','commentstyle','matlab');
for j=1:length(values)
    assign = sprintf('%s%s%f%s', char(params(j)), '=',values(j),';');
    eval(assign);
end

% Define variables that are dependent on global variables
HALFSIZE = floor(CORR_BOXSIZE/2);
ANGLE_TOL_RAD = ANGLE_TOL*pi/180;

% Desired forecast times (min)
forecast_times = [0 15 30 45 60 75 90 105 120];

% Get list of filenames to be used in forecast
[file_names,file_count] =  read_batch_file(input);

% Initialize total forecast counter.
n=0;

% Determine number of input file pairs to be analyzed
TOTAL_PAIRS = file_count-1;

% Start generating forecasts with second file
for h=2:file_count
    % Display progress
    PAIR_NUMBER = (h-1);
    tag = sprintf('%s%d%s%d', 'This is pair ', PAIR_NUMBER,...
        ' out of ', TOTAL_PAIRS);
    disp(tag)

    % Define first and second files to be used in correlation analysis
    file1 = file_names(h-1, :);
    file2 = file_names(h, :);

    % Perform correlation analysis on pair of images
    disp('Starting Correlation Analysis')
    [vec_valid, z3d, corr_matrix] = correlate_images(file1, file2);

    % Display correlation meta surface (CMS) surrounding selected pixel (optional).
    create_meta_surface(z3d,100,225);

    % vec_valid will equal -9999 when the first of the two images has no
    % weather in it.  Forecasts will not be generated and the csi scores
    % will not effect the average.
    if vec_valid ~= -9999
```

```
        %Increment total forecast counter
        n = n+1;

        % Filter the CMS to bias maximum displacement towards local average
        % location
        disp('Starting Quality Control (Meta-Surface Filtering)')
        [z3d] = quality_control(vec_valid, z3d);

        % Display filtered CMS around selected pixel (optional)
        create_meta_surface(z3d,90,225);

        % Convert vectors and remove errant vectors
        disp('Converting vectors')
        [x_vectors, y_vectors, vec_valid] = get_vectors(z3d,...
            corr_matrix, vec_valid);

        % Interpolate the removed vectors
        disp('Interpolating Missing Vectors')
        [x_vectors,y_vectors, vec_valid] = interpolation(x_vectors,...
            y_vectors,vec_valid);

        % Advect the image and write forecasts to files
        disp('Generating Forecasts')
        fcsts = advect_image(file2, x_vectors, y_vectors, forecast_times, outputdir);

        % Evaluate forecast accuracy with CSI scoring
        csi(h-1,:) = score_forecasts(fcsts, 1, 0, actdir);
    else
        disp('Empty input field: forecasting process skipped')
    end
end

% Plot average forecast accuracy against lead time
av_csi = sum(csi,1)./n;
figure
plot(forecast_times,av_csi.*100,'-*')
title('Average CSI v. Lead Time')
xlabel('Lead Time (min)')
ylabel('CSI (%)')

% Store forecast accuracy
csi_outfile = sprintf('%s%s', outputdir, '/average_scores.csi');
fid = fopen(csi_outfile,'w');
for i=1:size(av_csi,2)
```

```
    fprintf(fid, '%d   %f\n',forecast_times(i),av_csi(1,i));
end
fclose(fid);

% Determine and display elapsed time
elapsed_time = etime(clock,start)
```

B.2 global_tracker.m

```
function [av_csi] = global_tracker(input, outputdir, actdir)
% Main function for MATLAB based, short term extrapolation rainfall
% forecasting with a single vector derived from correlation analysis
% between whole images.  Takes a batch file of input files and generates a
% series of forecast files for each of the pairs of files.  Also requires
% destination directory for the forecast files as well as the location of
% the files for verification.  Calls  global_correlation, and advect_image
% to do velocity vector generation and image advection.  Files are written
% into the directory specified by outputdir.  Please use full
% directory/file names for all input files.

% Initialize timer and close all open figure windows
start = clock;
close all

% Declare Global Variables: Defined in file st.params
global FILTER_ROWS FILTER_COLS MAX_SHIFT CORR_BOXSIZE
global X_RES Y_RES TIME_SPACING ANGLE_TOL RAIN_THRESH
global WX_MIN RANGE SIGMA Z_THRESH SPEED_LIMIT SEARCH_RADIUS

% These global variable are defined within the following function calls
global TOTAL_ROWS TOTAL_COLS HALFSIZE THETA_GLOBAL
global MARGIN ANGLE_TOL_RAD

% Read parameters from external parameter file
[params, values] = textread('st.params', '%s %n', 'delimiter',...
    '\t','commentstyle','matlab');
for j=1:length(values)
    assign = sprintf('%s%s%f%s', char(params(j)), '=',values(j),';');
    eval(assign);
end

% Define variables that are dependent on global variables
HALFSIZE = floor(CORR_BOXSIZE/2);
ANGLE_TOL_RAD = ANGLE_TOL*pi/180;
```

```
del_t = TIME_SPACING*60;

% Desired forecast times (min)
forecast_times = [0 15 30 45 60 75 90 105 120];

% Get list of filenames to be used in forecast
[file_names,file_count] = read_batch_file(input);

% Initialize total forecast counter
n=0;

% Determine number of input pairs to be analyzed
TOTAL_PAIRS = file_count-1;

% Start generating forecasts with second file
for h=2:file_count
    % Display progress
    PAIR_NUMBER = (h-1);
    tag = sprintf('%s%d%s%d', 'This is pair ', PAIR_NUMBER,...
        ' out of ', TOTAL_PAIRS);
    disp(tag)

    % Define first and second files to be used in correlation analysis
    file1 = file_names(h-1, :);
    file2 = file_names(h, :);

    % Prepare images for correlation analysis
    [time1,time2,time1_pad,time2_pad,t2nf] = prepare_images(file1,file2);

    if sum(time1(:))~=0
        % Increment total forecast counter
        n=n+1;

        % Perform global correlation analysis
        [corr_matrix,g_vec] = global_correlation(time1,time2);

        % Clear unneeded variables
        clear time1 time2 time1pad time2pad

        % Turn pixel displacements into velocities in meters per second
        x_vectors = (g_vec(2)*X_RES/del_t);
        y_vectors = (g_vec(1)*X_RES/del_t);

        % Advect input file and score and write forecasts to files
        disp('Generating Forecasts')
        csi(h−1,:) = advect_global(file2, x_vectors, y_vectors,...
            forecast_times, outputdir, actdir);
    else
        disp('Empty input field: forecasting process skipped')
    end
end

% Plot average forecast accuracy against lead time
av_csi = sum(csi,1)./n;
figure
plot(forecast_times,av_csi.*100,'-*')
title('Average CSI v. Lead Time')
xlabel('Lead Time (min)')
ylabel('CSI (%)')

% Store forecast accuracy
outfile = sprintf('%s%s',outputdir, '/aprglobal.csi');
fid = fopen(outfile,'w');
for i=1:size(av_csi,2)
    fprintf(fid, '%d   %f\n',forecast_times(i),av_csi(1,i));
end
fclose(fid);

% Determine and display elapsed time.
elapsed_time = etime(clock,start)
```

B.3 read_batch_file.m

```
function [filenames, filecount] = read_batch_file(batch_file_name)
% Function to obtain contents of a batch file of filenames. Takes the
% batch filename as an input and returns a list of filenames from within
% the batch file and a count of the number of filenames found.

% Open batch file
batch_fid = fopen(batch_file_name, 'r');

% Initialize file counter
filecount = 0;

% Get number of lines (files) in batch file
while fgetl(batch_fid) ~= −1
    filecount = filecount + 1;
end
```

```
% Reset file position indicator
frewind(batch_fid);

% Get filenames from file
for g=1:filecount
    filenames(g,:) = fgetl(batch_fid);
end
```

B.4 correlate_images.m

```
function [vec_valid, z3d, corr_matrix] = correlate_images(file1, file2)
% Function that performs local area correlation between 'file1' and
% 'file2'.  Takes as inputs the full-path filenames and returns:
% 'vec_valid', a matrix of locations flagging which resulted in valid
% correlations, which are invalid and which need to be interpolated; 'z3d',
% a 3D representation of the global-vector biased, limited search area,
% correlation surface at each pixel; and 'corr_matrix', a row and column
% correction value to be applied to the raw locations of maximum
% correlation to determine displacement from center pixel.

% Declare Global Variables
global FILTER_ROWS FILTER_COLS MAX_SHIFT CORR_BOXSIZE WX_MIN
global RANGE SEARCH_RADIUS TOTAL_ROWS TOTAL_COLS HALFSIZE

% Prepare input files for global and local correlation analyses
[time1,time2,time1_pad,time2_pad,t2nf] = prepare_images(file1,file2);

% Check to ensure there is weather in the first image
if (sum(time1(:)) == 0) | (time1 == -9999)
    vec_valid = -9999;
    z3d = 0;
    corr_matrix = 0;
    disp('Error encountered in files: forecasting process skipped')
    return
end

% Perform Global Correlation analysis
[corr_matrix,g_vec] = global_correlation(time1,time2);

% Clear unneeded variables
clear time1 time2

% Initialize variables
counter = 0;
```

```
vec_valid = zeros(TOTAL_ROWS,TOTAL_COLS);
z3d = zeros(TOTAL_ROWS,TOTAL_COLS, (2*SEARCH_RADIUS+1)^2);

% Loop over all rows and columns in input images
for row=1:TOTAL_ROWS
    for col=1:TOTAL_COLS
        if t2nf(row,col) > 0
            % Designate all pixels who provide acceptable correlations as a
            % pixel to be interpolated later.  If there is no problem with
            % the correlation, the flag is set to 1 which indicates the
            % point as a "good" vector.
            vec_valid(row,col) = 2;

            % Select region from second image to locate in the first image
            sample = time2_pad(row:row+2*HALFSIZE, col:col+2*HALFSIZE);

            % Select restricted search area from first image
            field = time1_pad(row:row+2*MAX_SHIFT, col:col+2*MAX_SHIFT);

            % Compute the percentage of the sub areas that "have weather"
            per_weather_field = sum(field(:)>0)/prod(size(field));
            per_weather_sample = sum(sample(:)>0)/prod(size(sample));

            % Only do correlations for pixels whose sub areas have weather
            % coverage exceeding the set threshold and have weather in the
            % second image to be advected.
            if (per_weather_field > WX_MIN)&(per_weather_sample > WX_MIN)
                % Increment valid correlation counter
                counter = counter +1;

                % Adds noise to uniform sample pattern, as required by
                % normxcorr2
                if std(sample(:)) == 0
                    noise = (exp(0.25*randn(size(sample))));
                    sample = sample .* noise;
                end

                % Correlate the two sub images (image processing toolbox
                % function)
                z = normxcorr2(sample, field);

                % Create global vector biased and limited search area
                if counter==1
                    % Find the coordinates of the center of the
                    % full correlation matrix
```

```
          center = ceil(size(z)./2);

          % Define the center of a limited region of the full
          % correlation surface within which a maximum will be
          % identified.  This center is offset from the full center
          % by the global displacement center.
          new_center = center−g_vec;

          % Determine the beginning and ending rows and columns
          % of the re-centered limited search area.
          starts = new_center − SEARCH_RADIUS;
          ends = new_center + SEARCH_RADIUS;
        end

        % Check to ensure the edges of the limited search area are
        % within the unpadded correlation area (eliminates effects
            % of zero padded edges)
        if (ends(1) <= size(z,1)−2∗HALFSIZE)&...
            (starts(1) >= 2∗HALFSIZE)&...
            (ends(2) <= size(z,2)−2∗HALFSIZE)&...
            (starts(2) >= 2∗HALFSIZE)

          % Define the limited search area
          z_t = z(starts(1):ends(1),starts(2):ends(2));

          % Ensure that the limited search area is of proper size
          if size(z_t)==[2∗SEARCH_RADIUS+1,...
                          2∗SEARCH_RADIUS+1]
            % Define this pixel as a valid correlation location
            vec_valid(row,col) = 1;

            % Assign limited search area to output variable
            z3d(row,col,:) = z_t(:);
          end
        end
      end
    end
  end
end
```

B.5 prepare_images.m

```
function [time1,time2,time1_pad,time2_pad,time2_nf]=prepare_images(file1,file2)
% Function that takes in the two files to be correlated
% and prepares them for the correlation procedure.  Outputs
% from this function are the two filtered numerical fields,
% the two numerical fields filtered and padded with zeros,
% and the second field in its original form.

% Declare global variables
global FILTER_ROWS FILTER_COLS RAIN_THRESH TOTAL_ROWS
global TOTAL_COLS HALFSIZE MAX_SHIFT

% Load input files
time1 = load(file1);
time2_nf = load(file2);

% Filter the images prior to correlation
time1 = st_filt(time1, FILTER_ROWS, FILTER_COLS);
time2 = st_filt(time2_nf, FILTER_ROWS, FILTER_COLS);

% Eliminate the non-exceeding rainfall values.
time1(time1 < RAIN_THRESH) = 0;
time2(time2 < RAIN_THRESH) = 0;

% Check to ensure that the input files are of the same size
if (size(time1) ~= size(time2))
    disp('Files not the same size')
    return
end

% Define global variables dealing with the size of the data
[TOTAL_ROWS,TOTAL_COLS] = size(time1);

% Pad fields with zeros so that the correct sizes of sub-regions
% can be made without errors due to exceeding size of the matrix.
time2_pad = padarray(time2,[HALFSIZE HALFSIZE],0,'both');
time1_pad = padarray(time1,[MAX_SHIFT MAX_SHIFT],0,'both');
```

B.6 st_filt.m

```
function    filtered_image = st_filt(z, filter_rows, filter_cols)
% Filters a 2-D image using an averaging filter.  Takes image to be
% filtered and size of filter as inputs and returns filtered image.

% Define filter
weights = ones(filter_rows, filter_cols)/(filter_rows * filter_cols);
```

```
% Initialize output
filtered_image = z;

% Perform convolution filtering between input and filter.  Returns a matrix
% with the same dimensions as the input.
filtered_image = conv2(z, weights, 'same');
```

B.7 global_correlation.m

```
function [corr_matrix,g_vec] = global_correlation(time1,time2)
% Function to determine global displacement between two images.  Takes
% image pair as input and returns the correction matrix, which is the
% coordinate transform required to turn a pixel location in global
% displacement biased, limited search area into a displacement from the
% center pixel, and the global vector.

% Declare global variables
global THETA_GLOBAL SEARCH_RADIUS

% Correlate the second image with the first
z = normxcorr2(time2, time1);

% Find the location of the maximum correlation value.
[m_g, imax] = max(z(:));

% Turn the vector index into a matrix subscript location
[dy_g, dx_g] = ind2sub(size(z),imax);

% Find the coordinates of the center of the correlation matrix
center = ceil(size(z)./2);

% Convert from the location within the correlation vector to the actual
% displacement vector
g_vec(1) = -(dy_g-center(1));
g_vec(2) = -(dx_g-center(2));

% Find the "global angle" between 0 and 2*pi
THETA_GLOBAL = atan2(g_vec(1), g_vec(2));
if THETA_GLOBAL < 0
    THETA_GLOBAL = THETA_GLOBAL + 2*pi;
end

% Determine coordinate transform vector
corr_matrix = -(g_vec+SEARCH_RADIUS+1);
```

B.8 quality_control.m

```
function [z3d_out] = quality_control(vec_valid, z3d)
% Perform filtering on the correlation meta surface (CMS) to bias the
% location of the maximum correlation coefficient toward the local average.
% Takes the vec_valid flag matrix and the 3D representation of the
% correlation surfaces as inputs and returns the filtered 3D correlation
% surface representation.

% Declare global variables
global RANGE SIGMA Z_THRESH

% Determine the number of pixels in each individual correlation surface
% (CS)
c = size(z3d,3);

% Generate Gaussian weighted filter using an image processing toolbox
% function.
H = fspecial('gaussian', RANGE, SIGMA);

% Loop over total number of elements in the correlation surfaces
for g=1:c
    % Extract same relative pixel from each surface for filtering
    temp = z3d(:,:,g);

    % Filter the field with the Gaussian filter
    output = surface_filter(temp, H, vec_valid);

    % Eliminate correlation surface values below Z_THRESH
    output(output<Z_THRESH) = 0;

    % Assign filtered field to output variable
    z3d_out(:,:,g) = output;
end
```

B.9 surface_filter.m

```
function fsurf = surface_filter(matrix, H, valid)
% Function to smooth a 2D field using a specified filter not including the
% NaN values.  Takes the field to be filtered, the filter, and a field of
% locations to generate filtered values at as inputs and returns the
% filtered field.

% Remove extraneous screen output
```

```
warning off 'MATLAB:divideByZero'

% Declare global variables
global RANGE TOTAL_ROWS TOTAL_COLS

% Define filter radius
halfsize = floor(RANGE/2);

% Initialize output variable
fsurf = zeros(TOTAL_ROWS,TOTAL_COLS);

% Turn negative and zero correlation coefficients into NaNs
matrix( matrix<=0 ) = NaN;

% Loop over all interior points in field
for r= halfsize+1 : TOTAL_ROWS−halfsize
    for c= halfsize+1 : TOTAL_COLS−halfsize
        % Only filter points where valid correlations were made
        if valid(r,c)== 1
            % Define area of influence
            sub_matrix = matrix(r−halfsize:r+halfsize,...
                c−halfsize:c+halfsize);

            % Multiply by filter
            z = H.*sub_matrix;

            % Compute mean of positive values within filter weighted
            % influential area
            numer = nansum(z(:));
            t = z>0;
            w = H.*t;
            denom = sum(w(:));

            % Do not return infinite mean values
            if denom==0
                fsurf(r,c) = 0;
            else
                fsurf(r,c) = numer/denom;
            end
        end
    end
end
```

B.10 get_vectors.m

```
function [u,v, vec_valid] = get_vectors(z3d, corr_matrix,vec_valid)
% Function to convert correlation surfaces to vectors in meters per second.
% Also incorporates several quality control tests.  Takes the all the
% correlation surfaces, the correction matrix, the valid vector field and
% the interpolation field as inputs and returns the x- and y-direction
% velocities, and the adjusted valid vector field.

% Declare global variables.
global X_RES Y_RES TIME_SPACING SPEED_LIMIT TOTAL_ROWS
global TOTAL_COLS ANGLE_TOL_RAD THETA_GLOBAL

% Define other variable that are function of the global variables
del_t = TIME_SPACING*60; %seconds

% Initialize variables
x_shift = NaN*ones(TOTAL_ROWS,TOTAL_COLS);
y_shift = NaN*ones(TOTAL_ROWS,TOTAL_COLS);
theta_loc = NaN*ones(TOTAL_ROWS,TOTAL_COLS);

% Determine number of pixels in each CS
c = size(z3d,3);

% Loop over all rows and columns in the second image
for row=1:TOTAL_ROWS
    for col=1:TOTAL_COLS

        % Only generate velocities for valid correlations
        if vec_valid(row,col) == 1

            % Find the maximum correlation values and the maximum row index
            % over all the rows for each pixel.
            [m(row,col), imax] = max(z3d(row,col,:));

            % Retrieve the maximum column index for its associated row
            [y,x] = ind2sub([sqrt(c) sqrt(c)], imax);

            % Transform the returned maximum correlation location into a
            % displacement from the center of the full correlation surface
            x_shift(row, col) = −(x + corr_matrix(1,2));
            y_shift(row, col) = −(y + corr_matrix(1,1));

            % Get local vector angle between 0 and 2*pi
            theta_loc(row,col) = atan2(y_shift(row,col),x_shift(row,col));
            if theta_loc(row,col) < 0
                theta_loc(row,col) = theta_loc(row,col) + 2*pi;
            end

            % Get local magnitude
            r_local(row,col) = sqrt(x_shift(row,col)^2 + y_shift(row,col)^2);

            % Compute deviation from global angle
            angle_diff = abs(theta_loc(row,col)−THETA_GLOBAL);
            angle_diff = min([angle_diff abs(angle_diff−2*pi) (angle_diff+2*pi)]);

            % Remove vectors that fail tests on angle deviation, local
            % magnitude, and maximum correlation coefficient value and
            % specify them to be interpolated (vec_valid = 2)
            if (angle_diff > ANGLE_TOL_RAD)  |...
                    (r_local(row,col) > SPEED_LIMIT)|...
                    (m(row,col) == 0) | (isnan(m(row,col)))
                vec_valid(row,col) = 2;
                x_shift(row,col) = NaN;
                y_shift(row,col) = NaN;
                m(row,col) =NaN;
                theta_loc(row,col) = NaN;
                r_local(row,col) = NaN;
            end
        end % End if statement
    end % End Column loop
end % End Row loop

% Convert from displacements to velocities in meters per second
u = x_shift .* (X_RES/del_t);
v = y_shift .* (Y_RES/del_t);
```

B.11 interpolation.m

```
function [xo,yo,v] = interpolation(x,y,v)
% Function to interpolate missing vectors values with the average of the
% surrounding magnitudes.  Takes the vectors fields and the valid
% field as inputs and returns full vector fields and modified valid field..

% Declare global variables
global TOTAL_ROWS TOTAL_COLS RANGE

% Define radius of influence
halfbox = floor(RANGE/2);
```

```
% Pad vector fields with NaN values.
x = [NaN*ones(halfbox,TOTAL_COLS+2*halfbox);...
        NaN*ones(TOTAL_ROWS,halfbox) x...
        NaN*ones(TOTAL_ROWS,halfbox);...
        NaN*ones(halfbox,TOTAL_COLS+2*halfbox)];
y = [NaN*ones(halfbox,TOTAL_COLS+2*halfbox);...
        NaN*ones(TOTAL_ROWS,halfbox) y...
        NaN*ones(TOTAL_ROWS,halfbox);...
        NaN*ones(halfbox,TOTAL_COLS+2*halfbox)];

% Loop over all rows and columns in image
for m=1:TOTAL_ROWS
    for n=1:TOTAL_COLS
        % Only interpolate vectors identified by the interpolation field
        if v(m,n) == 2
            % Define local areas
            loc_area_x = x(m:m+2*halfbox,n:n+2*halfbox);
            loc_area_y = y(m:m+2*halfbox,n:n+2*halfbox);

            %Compute local means
            loc_mean_x = nanmean(loc_area_x(:));
            loc_mean_y = nanmean(loc_area_y(:));

            % Assign local means to center pixels
            x(m+halfbox,n+halfbox) = loc_mean_x;
            y(m+halfbox,n+halfbox) = loc_mean_y;

            % Change the flag on the vector to represent a valid vector
            v(m,n) = 1;
        end
    end
end

% Trim fields to original size
xo = x(halfbox+1:end-halfbox,halfbox+1:end-halfbox);
yo = y(halfbox+1:end-halfbox,halfbox+1:end-halfbox);
```

B.12 advect_image.m

```
function fcst_file=advect_image(initial_file,x_vec,y_vec,forecast_times,outputdir)
% Function that performs the actual forecasting step.  Takes an original
% file, x- and y-direction vectors, a string of forecast times, and the
% directory for the outputted files as inputs and returns a list of files
% that the forecasted fields were written to.  Calls file_writer.

% Declare global variables
global X_RES Y_RES TOTAL_ROWS TOTAL_COLS TIME_SPACING

% Eliminate extraneous output to the screen
warning off 'MATLAB:divideByZero'

% Load the initial file
initial_file_mat = load(initial_file);

% Determine number of forecasts to be made
d = size(forecast_times, 2);

% Convert forecast times (in min) to seconds
forecast_times_sec = forecast_times .* 60;

% Remove NaN values from vector fields
x_vec(isnan(x_vec)) = 0;
y_vec(isnan(y_vec)) = 0;

% Define the field to be advected
start = initial_file_mat;

% Compute incremental velocity field.
xv = x_vec .* (TIME_SPACING.*60./X_RES);
yv = y_vec .* (TIME_SPACING.*60./X_RES);

% Loop over all forecast times greater than zero
for t=1:d
    if forecast_times(1,t) ~= 0

        % Initialize forecast, velocity and averaging fields
        forecast = zeros(TOTAL_ROWS,TOTAL_COLS);
        xv_new = zeros(TOTAL_ROWS,TOTAL_COLS);
        yv_new = zeros(TOTAL_ROWS,TOTAL_COLS);
        check = zeros(TOTAL_ROWS,TOTAL_COLS);

        % Loop over all internal rows and columns in initial image
        for row = 2:TOTAL_ROWS-1
            for col = 2:TOTAL_COLS-1
                % Only advect pixels that have weather
                if start(row,col) > 0

                    % Determine the index in the forecast for each point in
            % the initial file based on the initial index and the
            % displacement
            row_out = round(row + yv(row, col));
            col_out = round(col + xv(row, col));

            % Do not put weather outside of forecast area
            if (row_out > 1) & (col_out > 1) &...
                (row_out<TOTAL_ROWS) & (col_out <TOTAL_COLS)
                % Translate each pixel and its neighborhood to the
                % corresponding locations in the forecast or
                % advected velocity fields.
                forecast(row_out−1:row_out+1, col_out−1:col_out+1) = ...
                    forecast(row_out−1:row_out+1,...
                    col_out−1:col_out+1)+...
                    start(row−1:row+1, col−1:col+1);
                xv_new(row_out−1:row_out+1, col_out−1:col_out+1) = ...
                    xv_new(row_out−1:row_out+1,...
                    col_out−1:col_out+1) +...
                    xv(row−1:row+1,col−1:col+1);
                yv_new(row_out−1:row_out+1, col_out−1:col_out+1) = ...
                    yv_new(row_out−1:row_out+1,...
                    col_out−1:col_out+1) +...
                    yv(row−1:row+1,col−1:col+1);

                % Increment the counter for number of values placed
                % in each pixel
                check(row_out−1:row_out+1, col_out−1:col_out+1) =...
                    check(row_out−1:row_out+1, col_out−1:col_out+1)...
                    + ones(3,3);
            end
        end
    end
end

% Average advected values
xv = xv_new./check;
yv = yv_new./check;
forecast = forecast./check;

% Remove any cells that had zero pixels advected there.
xv(isinf(xv)) = NaN;
yv(isinf(yv)) = NaN;
forecast(isinf(forecast)) = NaN;

% Get forecast file name
```

```
        fcst_file(t,:) = sprintf('%s%s%s%03d%s', outputdir,...
            '/f',initial_file(end−19:end−7), forecast_times(1,t), '.txt');

        % Write forecast file
        file_writer(forecast, fcst_file(t,:));

        % Change the file to be advected in the next forecasting step
        start = forecast;
    else
        % Do not do the advection for the "zero time" forecast
        forecast = initial_file_mat;

        % Determine file name and write zero time forecast to the file
        fcst_file(t,:) = sprintf('%s%s%s%03d%s', outputdir,...
            '/f',initial_file(end−19:end−7), forecast_times(1,t), '.txt');
        file_writer(forecast, fcst_file(t,:));
    end
end
```

B.13 advect_global.m

```
function csi=advect_global(initial_file,x_vec,y_vec,forecast_times,outputdir,actdir)
% Function that performs the actual forecasting step.  Takes an original
% file, x- and y-direction vectors, a string of forecast times, the
% directory to write the forecasts to and the directory where the
% verification files can be found as inputs and returns an array of CSI
% scores for the forecasts.  Calls file_writer and csi_score.

% Declare global variables
global TOTAL_ROWS TOTAL_COLS X_RES Y_RES

% Load the initial file
initial_file_mat = load(initial_file);

% Determine number of forecasts to be made
d = size(forecast_times, 2);

% Convert forecast times (in min) to seconds
forecast_times_sec = forecast_times .* 60;

% Loop over all positive forecast times
for t=1:d
    if forecast_times(1,t) ~= 0
        % Compute displacements in pixels for each forecast time
        x_vec_appl = x_vec .* (forecast_times_sec(1, t)./X_RES);
        y_vec_appl = y_vec .* (forecast_times_sec(1, t)./Y_RES);

        % Initialize forecast field
        forecast = NaN.*ones(TOTAL_ROWS,TOTAL_COLS);

        % Loop over all rows and columns in initial image
        for row = 1:TOTAL_ROWS
            for col = 1:TOTAL_COLS
                % Only advect pixels that have weather
                if initial_file_mat(row,col) > 0

                    % Determine the index in the forecast for each point in
                    % the initial file based on the initial index and the
                    % displacement
                    row_out = round(row + y_vec_appl);
                    col_out = round(col + x_vec_appl);

                    % Do not put weather outside of forecast area
                    if (row_out > 0) & (col_out > 0) &...
                            (row_out<=TOTAL_ROWS)&...
                            (col_out <=TOTAL_COLS)
                        % Translate each pixel to its corresponding point
                        % in the forecast image.
                        forecast(row_out, col_out) = initial_file_mat(row, col);
                    end
                end
            end
        end

        % Get forecast file name
        fcst_file(t,:) = sprintf('%s%s%s%03d%s', outputdir,'/f',...
            initial_file(end-19:end-7), forecast_times(1,t), '.txt');

        % Write forecast field to file
        file_writer(forecast, fcst_file(t,:));
    else
        forecast = initial_file_mat;
        % Get forecast file name
        fcst_file(t,:) = sprintf('%s%s%s%03d%s', outputdir,'/f',...
            initial_file(end-19:end-7), forecast_times(1,t), '.txt');

        % Write forecast field to file
        file_writer(forecast, fcst_file(t,:));
    end
    % Get actual file name to compare forecast to
    actual = get_actual_file(fcst_file(t,:), actdir);

    % Score forecast with CSI
    csi(1,t) = csi_score(fcst_file(t,:), actual, 1,0);
end
```

B.14 file_writer.m

```
function file_writer(matrix, outputfile)
% Function that takes in a matrix and writes it to a file.  Takes matrix
% and output file name as input and has no outputs.

% Open output file
out_fid = fopen(outputfile, 'w');

% Get size of input field
[rows, cols] = size(matrix);

% Loop over all rows and columns.
for r=1:rows
    for c=1:cols
        % Write each pixel to output file
        fprintf(out_fid, ' %6.3f ', matrix(r,c));
    end
    % Move to new row
    fprintf(out_fid, '\n');
end

% Close output file
fclose(out_fid);
```

B.15 score_forecasts.m

```
function csi = score_forecasts(file_names,va,th,actdir)
% Function to take forecast files and determine their CSI score as compared
% to the actual file at that time.  Takes a list of forecast files, a
% verification area side length, a rainfall threshold and the directory
% where the verification files are located as inputs and returns a vector
% of csi scores.

% Determine the number of forecasts to be scored.
file_count = size(file_names,1);
```

```
for h=1:file_count
    % Determine the actual file corresponding to the forecast file
    actual_file(h,:) = get_actual_file(file_names(h,:), actdir);

    % Score the forecast file
    csi(h,1) = csi_score(file_names(h,:),actual_file(h,:),va,th);
end
```

B.16 get_actual_file.m

```
function [actual_weather] = get_actual_file(text_input, actdir)
% Takes a forecast file produced by the MATLAB Storm Tracker and determines
% the actual file that corresponds to the forecast. Takes as input the
% forecast file and the directory name where the file should be located and
% returns the full path name of the actual file. Does not test if increase
% in time goes into another month or year(yet)

% Extract the initial year from the forecast file
year_str = text_input(end−15:end−12);
year = str2num(year_str);

% Extract the initial month from the forecast file
month_str = text_input(end−19:end−18);
month = str2num(month_str);

% Extract the initial day from the forecast file
day_str = text_input(end−17:end−16);
day = str2num(day_str);

% Extract the initial hour from the forecast file
hour_str = text_input(end−11:end−10);
hour = str2num(hour_str);

% Extract the initial minute from the forecast file
min_str = text_input(end−9:end−8);
min = str2num(min_str);

% Extract the forecast lead time from the forecast file
t_f = text_input(end−6:end−4);
t_f = str2num(t_f);

% Increment the initial minute by the forecast time
min_out = min+t_f;
```

```
% Determine the house increment
hour_inc = floor(min_out/60);

% Determine the output minute
if min_out >=60
    min_out = mod(min_out,60);
end

% Increment the hour by the minute excess
hour_out = hour + hour_inc;

% Determine the day increment
day_inc = floor(hour_out/24);

% Determine the output hour
if hour_out>=24
    hour_out = mod(hour_out,24);
end

% Determine the output day
day_out = day+day_inc;

% Write the full path name for the actual file into a string
actual_weather = sprintf('%s%s%02d%02d%s%02d%02d%s',...
    actdir, '/w', month,day_out,year_str,hour_out,min_out,'.001.txt');
```

B.17 csi_score.m

```
function    [csi] = csi_score(forecast, actual, window_size, thresh)
% Function to compare a forecast file with the actual file at that time.
% Takes as inputs the forecast file, the actual file, the size of the
% verification area and the rainfall rate threshold and returns the CSI
% score for that file pair.  The window_size input can be either a single
% integer (for a square verification area), the string 'cross' (for a cross
% verification area), or the string 'rect' (for a 3 row by 5 column
% rectangle verification area).

% Check to ensure that the window exists
if window_size < 1
    disp('window_size must be greater than or equal to 1')
    csi_area = NaN;
    return
end
```

```
% Load files
fcst = load(forecast);
act = load(actual);

% Determine size of files, assumes they are the same size
[n,m] = size(act);

% Eliminate NaN locations in the forecast and actual file
fcst(isnan(fcst)) = 0;
act = act > thresh;

% Define verification area weighting functions
if sum(size(window_size)) == 2
    weights = ones(window_size) / window_size^2;
elseif strcmp(window_size, 'cross')
    weights = [0 0.2 0 ; 0.2 0.2 0.2; 0 0.2 0];
elseif strcmp(window_size, 'rect')
    weights = (1/15).*ones(3,5);
end

% Initialize convolution variable
sz = act;

% If the verification area is larger than 1x1, smooth the binary actual
% field by the verification area
if size(weights)~=[1 1]
    sz = conv2(act, weights, 'same');
end

% Initialize counters
correct_no = 0;
hit = 0;
miss = 0;
false_alarm = 0;

% Loop over all rows and columns in the images
for x=1:m
    for y=1:n
        % If there was actually rain within a "window_size" area around the
        % pixel in question, sz will be greater than zero, and if fcst is
        % also greater than that threshold, increment the hit counter
        if sz(y,x) > 0 & fcst(y,x) > thresh
            hit = hit + 1;
            % If both the actual and the forecast are less than the
            % threshold, increment the correct_no counter
        elseif act(y,x) == 0 & fcst(y,x) <= thresh
            correct_no = correct_no +1;
            % if the actual is less than the threshold and the forecast is
            % greater than the threshold, increment the false alarm counter
        elseif act(y,x) == 0 & fcst(y,x) > thresh
            false_alarm = false_alarm + 1;
            % If the actual exceeds the threshold but the forecast does
            % not, increment the miss counter
        else
            %act(y,x) == 1 & fcst(y,x) <= thresh
            miss = miss + 1;
        end
    end
end

% The CSI is the ratio of the hits to the total of the hits, misses and
% false alarms
csi = hit/(hit+miss+false_alarm);
```