UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

AUTOMATIC RECOGNITION OF SUPERCELLS THUNDERSTORMS IN NUMERICAL WEATHER DATA

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

THIBAULT LUCIDARME
Norman, Oklahoma
2014

AUTOMATIC RECOGNITION OF SUPERCELLS THUNDERSTORMS IN NUMERICAL WEATHER DATA

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Acknowledgments

I would like to thank Professor Amy McGovern, my supervisor at the University of Oklahoma as well as the chair of my master's thesis committee, for accepting me as her student, and for helping me all along during my time in the United States. I would like to thank all my current and former committee members, namely Dr. Rodger Brown, for his helpful counseling and insights concerning meteorological subjects, Dr Dean Hougen, Dr. Valliappa Lakshmanan and Dr. Adam Clark for helping me and orienting me when I started this research project.

I also thank Brittany Benson Dahl for her patience and for providing all the simulation data that I have used.

I truly thank Andrew MacKenzie, David John Gagne II, Greg Blumberg, who have helped me so much, be it for research matters or for everyday life problems. I am also grateful to my awesome roommates Tim Supinie and Heather Stout, as well as my unforgettable friends Diana Leigh Goeller, Cathy Bolene Gagne, Jeremy Pfeifer and Mikael Perrin for their unconditional friendship and for their support during the hard times. They all have greatly contributed to making my stay as profitable and as enjoyable as possible.

I would also like to thank all the members of the I.D.E.A. lab who have contributed to giving me a wonderful research experience.

Lastly, I would like to thank my amazing parents Olivier and Patricia Lucidarme who have supported me during all this time and to whom I owe everything.

Table of Contents

List Of Figures

Abstract

The Tornado Alley is a geographic region in central United States that is home to the largest number of violent tornadoes in the world. The cloud structures that produce the most violent tornadoes are known as supercells. Being able to predict the formation and the evolution of supercell thunderstorms can improve the forecasting of such severe weather events, thereby preventing human and resources loss. Currently, most of the detection of weather events such as supercells are done using radar data. Radar data only provides limited information and it is presently impossible to possess a full set of fundamental variables in real time. However, in the near future, it will likely be possible to use data assimilation to infer those fundamental variables. Updraft helicity is one such non-radar related measure. Updraft helicity is a measure of the amount of an updraft rotation. The goal of this study is to automatically detect at which timestep a simulated thunderstorm evolves into a supercell using exclusively fundamental variables such as updraft helicity and vertical velocity. By solving this problem we are able to identify supercells among nonsupercell thunderstorms using Support Vector Machine as a classifier.

Keywords: supercell identification, vertical helicity, watershed algorithm, image registration, Support Vector Machine

Chapter 1

Introduction

Oklahoma is positioned in the central area of the United States of America. This region of the world is the meeting point of multiple great flows of air (Figure 1.1) that are the cause of multiple hazardous weather phenomena, such as tornadoes, hail, flash floods and strong winds, which result in significant loss of life and property [Pielke and Carbone, 2002]. Tornadoes are most common in this area because it is a region where warm, moist air from the Gulf of Mexico meets cold, dry air from Canada, creating the strongest and longest lasting type of thunderstorm—a highly organized storm called a *supercell thunderstorm*—that is responsible for causing the most of the significant damage.

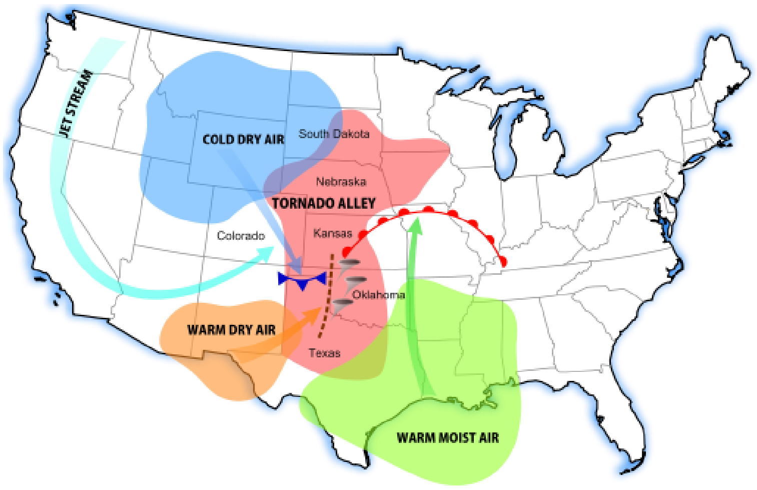


Figure 1.1: Origin of the tornado alley. [Wikipedia, 2013]

In this environment, weather forecasting plays a crucial role. Since supercell thunderstorms have a unique radar configuration, forecasters have traditionally depended on Doppler weather radar observations of supercell thunderstorms to issue short–term severe storm warnings [Moller et al., 1994]. The severe thunderstorm and tornado warnings heavily rely on the presence and detection of the reflectivity hook echo and Bounded Weak Echo Regions (BWER) as well as a Doppler velocity signature of strong rotation (mesocyclone).

Figure 1.2 sketches the horizontal cross-sections of the radar reflectivity (also called radar echo) structure of a supercell thunderstorm at heights indicated along the left side. The reflectivity (indicated by dBZ values) becomes stronger with increase in precipitation particle size. The origin of the AB–CD axes is the location of storm's strong updraft. The updraft is rising so fast that cloud droplets in it do not have time grow to radar–detectable raindrops until they are halfway up through the storm—resulting in a BWER at low and midaltitudes and a reflectivity maximum at the top of the storm. Once radar–detectable particles form, they are associated with the strongest reflectivity values in the upper portion of the storm [Chisholm and Renick, 1972; Lemon and Doswell III, 1979].

Research Question

Non-RADAR related supercell identification More recently, it has been proposed that a numerical model—that has been initialized using current environmental conditions—be used to predict supercell thunderstorms and tornadoes; when perfected, the concept is that the time and location of numerically–modeled tornadoes and other hazardous weather conditions could be used to issue warnings before the storms become severe. This approach is called *Warn–on–Forecast* [Stensrud and et al., 2009]. Using the Warn–on–Forecast approach, the identification of a supercell would be based on its definition as a storm with a strong rotating updraft i.e. the presence of *updraft helicity* [1] which is defined as the correlation (colocation) of an updraft [2] and a counter–clockwise rotation about a vertical axis: *vertical vorticity* [3] . For a more detailed definition of the vertical vorticity, see Section 2. Typically only one value is used for updraft helicity, which is computed as the vertical summation of horizontally–averaged vertical helicity within the lower portion of the storm [Kain and et al, 2008; Naylor and et al, 2003]. The question arises whether important information is lost by treating UH as a single volumetric value. For this study, we investigate whether using the full field of vertical helicity region instead of a single maximum provides valuable information that can be used to characterize a storm as a supercell storm, or not.

[1]Updraft Helicity abbrev: UH

[2]Updraft abbrev: W

[3]Vorticity abbrev: ζ

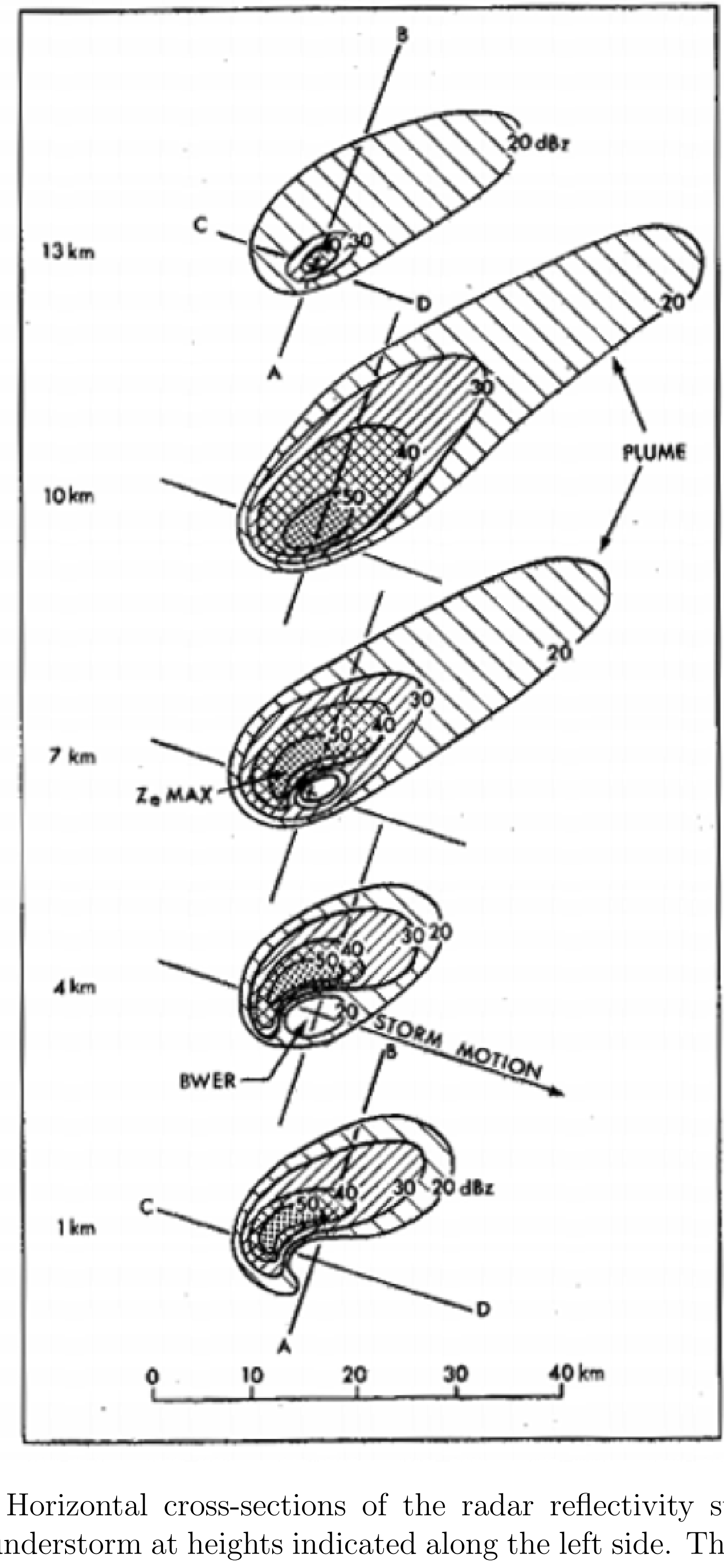


Figure 1.2: Horizontal cross-sections of the radar reflectivity structure of a supercell thunderstorm at heights indicated along the left side. The intersection of the two axes indicate the location of a counter–clockwise rotating updraft; the low–altitude hook–shaped echo on the right rear side of the storm arises from precipitation being drawn around the rotating updraft; The strong updraft produces a BWER at lower heights because cloud droplets have not had time to grow to radar–detectable raindrops. [Chisholm and Renick, 1972]

Chapter 2

Dataset Description

Almost all studies mentioned in Section 1 focus on radar data. Radars work by emitting radio waves that eventually scatter and reflect on various materials. The higher the electrical conductivity of the material encountered by the emitted waves, the more reflection will occur and the brighter the object will appear on the radar. *Reflectivity* refers to the fraction of incident electromagnetic power that is reflected at an interface. In meteorology, most of the work related to supercell detection is based on reflectivity. It is currently impossible to give real time full weather observations, such as wind, temperature and pressure profiles in details, because instruments used to measure those fields remain suitable for local measures but are inappropriate to follow fast-paced massive 3D phenomena such as supercells or tornadoes. Data analysts are currently limited to the reflectivity observed with a radar.

Using fundamental variables can bring a lot to the studies of the formation and evolution of tornado-leading supercells. In the future, it is expected that we will be able to perform data assimilation in real time. When the technology comes, we hope that the results of this study will be applied in order to provide a real time segmentation of supercells based on the real definition and description of their physical features.

The simulations output files that contain the 1D variable values in the 3D space field. Those variables are extracted and treated as a collection of 3D grayscale images on which we will devise and apply various data mining techniques.

The main information we need that is not accessible in real data is the three-dimensional wind field; which forms the core data from which we will infer important value fields such as vorticity and helicity.

Velocity The *wind velocity* is a 3 dimensional vector whose components are conventionally called (u,v,w):

- **u:** the East-West wind component
- **v:** the North-South wind component
- **w:** the vertical wind component

Vorticity *Vorticity* is the field that describes the local spin of a fluid. Mathematically, vorticity is defined as the curl of the velocity:

$$\overrightarrow{\zeta} = \overrightarrow{curl}(\overrightarrow{velocity})$$

In practice, we use the curl formula in Cartesian coordinates:

$$\overrightarrow{\zeta} \quad = \quad \nabla \times \overrightarrow{velocity}$$

Helicity In meteorology, helicity refers to the product of velocity and vorticity, where the vorticity vector points in the same direction as the velocity vector. We will focus on the vertical helicity (H), which is defined as:

$$H = \overrightarrow{velocity_v}.\overrightarrow{vorticity_v}$$

where $velocity_v$ and $vorticity_v$ are respectively the vertical velocity and vertical vorticity.

As mentioned in section 1, in meteorology, updraft helicity is a scalar measure of the vertical helical flow in a fluid. Instead of being used as a field, updraft helicity is integrated down to a single scalar and only then it is used to determine the existence or not of a supercell [Kain and et al, 2008]. This study differs as it uses the full 3D field of vertical helicity to characterize a tornadic storm instead of a single scalar value.

Supercells have the capability to deviate from the mean of the horizontal wind throughout the depth of the storm. If they track to the right or left of the mean wind, they are said to be *right-movers* or *left-movers*, respectively. Supercells can sometimes develop two separate updrafts with opposing rotations, which splits the storm into two supercells: one left-mover and one right-mover. It was decided that only right movers would be considered in this study because they generally produce the strongest supercells. For this reason, both the updraft and the vertical vorticity are thresholded to keep only positive values. This forces the helicity to be positive and removes the possible anti-mesocyclonic left-movers. It is noted that the content of this study could also apply to anti-mesocyclonic updrafts (left-movers with negative helicity); in case a thunderstorm splits into two supercells with opposite rotating updrafts, it is possible to apply the pipeline described in Figure 3.1 to both updrafts separately. We restict ourselves to the mesocyclonic updrafts in this study.

Our data set consists of five simulated storms that all evolve into supercells. The storms are fully simulated Geary model thunderstorms spanning 3 hours, from birth (0 second) to total dissipation (10800 seconds). The horizontal resolution is 100 meters, the vertical resolution is a dilated Arakawa staggered

C-grid and the temporal resolution is 30 seconds. Before 1200 seconds and after 9000 seconds, the storms' simulations diverge from reality as they suffer from boundary condition noise and discrepancies. Therefore we restrict each of the five storms to their development between 1200 seconds and 9000 after formation. Moreover, in order to decrease the volume of data to process while keeping significant events in the supercells lifespan, we subsample the u,v and w fields to 1 timestep every 300 seconds ($1/10^{th}$ of the temporal resolution). Each timestep was labeled by Greg Blumberg, a meteorology graduate student from the University of Oklahoma. Timesteps were labeled with a boolean: true if the storm depicted at least one right mover supercell, false otherwise. Those values will be used as truth values during the classification process.

Chapter 3

Algorithm Formulation

As mentioned in Section 2, all of the algorithms we will devise aim at identifying, segmenting and classifying regions of interest in space-time vertical helicity.

The flow chart in Figure 3.1 represents the overall structure of the algorithm devised.

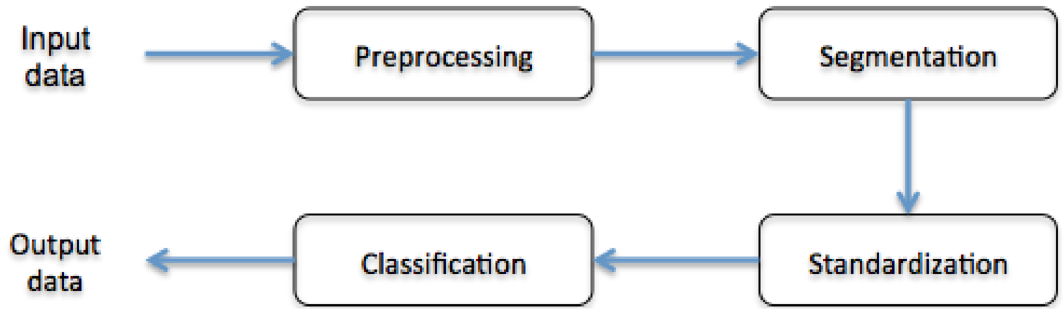


Figure 3.1: Algorithm flow chart.

The pipeline can be separated into the following distinct steps:

1. *Preprocessing:* A lot of segmentation algorithms are gradient based methods. This step aims to reduce the minor variations in the data's gradient in order to improve the segmentation.

2. *Segmentation:* Using an enhanced watershed algorithm on the preprocessed image of an helicity field, we segment the main rotating updraft of the supercell.

3. *Standardization:* We register the binary masks obtained from the segmentation in order to have a standardized set of segmented supercell features.

4. *Classification:* We run a Support Vector Machine (SVM) of the standardized data set to iteratively create two clusters: a supercell cluster and a nonsupercell cluster.

Using this pipeline we are able to significantly differentiate, for a given time, between supercell thunderstorms and nonsupercell thunderstorms. Using this technique it is possible to mark the time of formation of a supercell within a thunderstorm.

3.1 Preprocessing: Anisotropic Diffusion

Gradient based algorithms are often extremely affected by insignificant variations in gradient. The definition of significance is usually addressed as the magnitude of the gradient vector. If a gradient magnitude has a higher value, it is considered being globally more significant than a lower magnitude gradient. Since most regions of interest are characterized by higher gradients (or lack thereof) the aim of Anisotropic Diffusion is to smooth out the smallest and the least significant gradient regions all the while conserving the most significant gradients without smoothing them.

3.1.1 Principle of Anisotropic Diffusion

The Anisotropic Diffusion algorithm, also known as the Perona-Malik algorithm [Perona and Malik, 1990], is a variation of the isotropic diffusion algorithm best known as the Gaussian blurring algorithm. The Gaussian blurring effect smooths equally all gradients according to the size of the convolution mask. Therefore a lot of unwanted gradient variations are removed, but so are a lot of interesting gradient based features. Usually, the size of the Gaussian filter is designed to retain those features roughly the size of the region of interest. Unfortunately in meteorological data, each region can widely vary in size and each datatset may have a different resolution which results in multiple features in physical space that are represented over a great spectrum of sizes in pixel space. Those variations are such that no fixed Gaussian filter size may be appropriate from one image to another.

The idea proposed by Perona and Malik [Perona and Malik, 1990] is to use a partial derivative equation PDE-based denoising method. It applies Fick's second law of thermal diffusion, also formulated as the heat equation, to the pixels' intensity values of an image. The heat equation describes the distribution of heat (or variation in temperature) in a given region over time.

$$\frac{\partial h}{\partial t} - \alpha \nabla^2 h = 0$$

where:

$h(x, y, z, t)$ represents the 3 dimensional helicity field, in any coordinate system, that varies over time t;

∇^2 is the laplacian operator;

α is a diffusion coefficient called thermal diffusivity (m^2/s)

In thermodynamics the equation is used to model the diffusion of a temperature gradient within an isolated system. The analogy made by Perona and Malik consists of treating a pixel's intensity value as a temperature and diffuse it according the the value of the gradient. In the heat equation the term α represents how easy it for the field u to diffuse in its environment. The higher the value, the faster heat diffuses. The more diffusion occurs, the more homogeneous the system will locally be when reaching stability. The goal is to obtain an almost-piecewise constant approximation of the image. The idea of Anisotropic Diffusion is to have a spatially dependent diffusion coefficient that is function of image position instead of being a constant scalar. This results in an inhomogeneous and nonlinear diffusion filter

$$\frac{\partial h}{\partial t} - \alpha(x, y, z, t)\nabla^2 h = 0$$

Where the gradient is low, the corresponding regions of the image smooth out completely and become homogeneous whereas where the gradient is high almost no diffusion should occur at all and the edges are preserved. In their original papers Perona and Malik propose two functions as the diffusion coefficient.

- A Gaussian based coefficient favors high-contrast edges over low-contrast ones.
$$\alpha(x, y, z, t) = \exp\left(-\frac{\nabla h(x, y, z, t)^2}{K}\right)$$

- A sigmoid based coefficient favors large regions over smaller ones.

$$\alpha(x, y, z, t) = \frac{1}{1 + \left(\frac{\nabla h(x, y, z, t)^2}{K}\right)}$$

where:

h is the 3D helicity image given at a time t

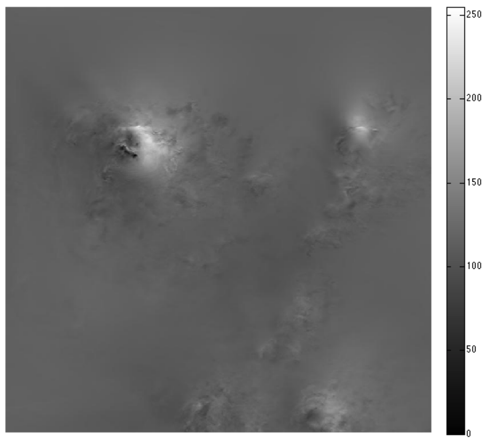
∇ is the gradient operator

The constant K is fixed by hand at some fixed value.

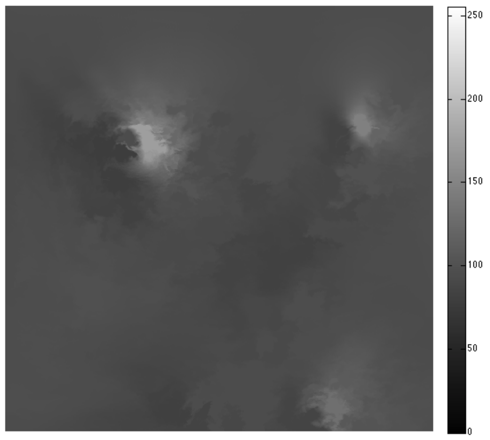
In our case, the region width will already be chosen using the enhanced version of the watershed (c.f. 3.2). Choosing the Gaussian version of the diffusion coefficient ensures that both criteria, contrast and region width, are taken into account in the overall data flow.

Because of the heterogeneous nature of the meteorological features' size, there may be small regions in pixel space that are blurred when they should not be. This blurring may remove discontinuities in intensity that can significantly modify the output of the segmentation process. Even if the result is more satisfactory than using a Gaussian filter, this problem prevents us from devising algorithms that are robust to resolution changes. In order to solve this issue, we decided to apply the filter in physical space instead of applying it in pixel space, using a convolution mask with a given physical size. This way, no matter the resolution, a characteristic minimal length of the feature is all that is needed. Having to switch back and forth between physical space and pixel space is a quick operation, so no additional computation is needed.

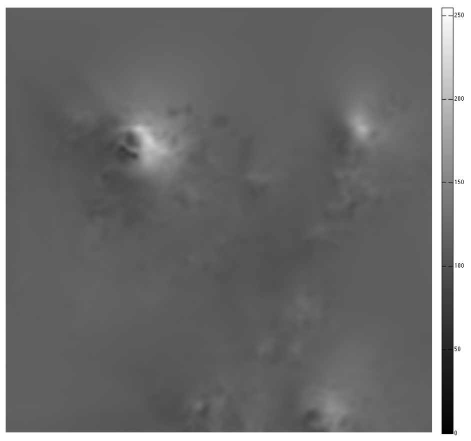
The problem of the size of the convolution mask proves to be a minor problem. By creating a Gaussian or sigmoid gradient-based diffusion coefficient, we ensure that there is a limitation to the diffusion in pixels space. The algorithm converges before reaching an equilibrium where the entire image is a single homogeneous value. The process is similar to quantization: a region of low gradient will theoretically diffuse until homogeneous and the diffusion will stop at the higher gradient. Therefore, a small enough filter size, to ensure that gradient do not overlap, and a high number of iteration, to make sure that the diffusion is complete, will ensure a satisfying output (Fig. 3.2).



(a) Horizontal slice of vertical helicity



(b) Perona-Malik diffused image; the diffusion parameter depends on the gradient.



(c) Gaussian blurred image; convolution kernel size is 11 pixels and standard deviation is 5 pixels.

Figure 3.2: Comparison of Perona-Malik diffusion and Gaussian blurring applied on weather data

3.2 Segmentation: Enhanced Watershed

Computer analysis of image objects starts with deciding which pixels belong to each object. This is called image segmentation, the process of separating objects from the background, as well as from each other. One such algorithm that is often used in spatial data mining is called the Watershed Algorithm. Following the results obtained by Lakshmanan [Lakshmanan et al., 2009], this algorithm was selected as the segmentation algorithm.

3.2.1 Principle of the Watershed Algorithm

A watershed is a basin-like landform defined by highpoints and ridgelines that descend into valleys. The Watershed Algorithms are based on a topological analogy between the grayscale gradients and a height map. The higher the pixel value, the higher the corresponding height of the point in 3D space. There are multiple variations of the Watershed Algorithms. The most common ones are the *Watershed by flooding* and the *Watershed by distance.*

In this study, we are going to use Watershed by flooding. This choice is justified by the fact that the Enhanced Watershed Algorithm discussed in section 3.2.2 is more efficient with an horizontal saliency criterion which easier to determine on a Watershed by flooding Algorithm. On the other hand the Watershed by topographic distance is more appropriate for a vertical sliency criterion.

Watershed by flooding Also referred to as the bottom-to-top algorithm, the idea has been introduced in 1979 by S. Beucher and C. Lantuejoul [Beucher and Lantuéj, 1979]. It consists of placing a water source in each regional minimum (cf. Figure 3.3a) to flood the relief from the sources in a fashion similar to a region growing algorithm. Where the multiple growth from the multiple seeds meet (cf. Figure 3.3b), we consider a “barrier” to be built . The set of barriers when no additional growth is possible is the result of the Watershed Algorithm by flooding (cf. Figure 3.3d).

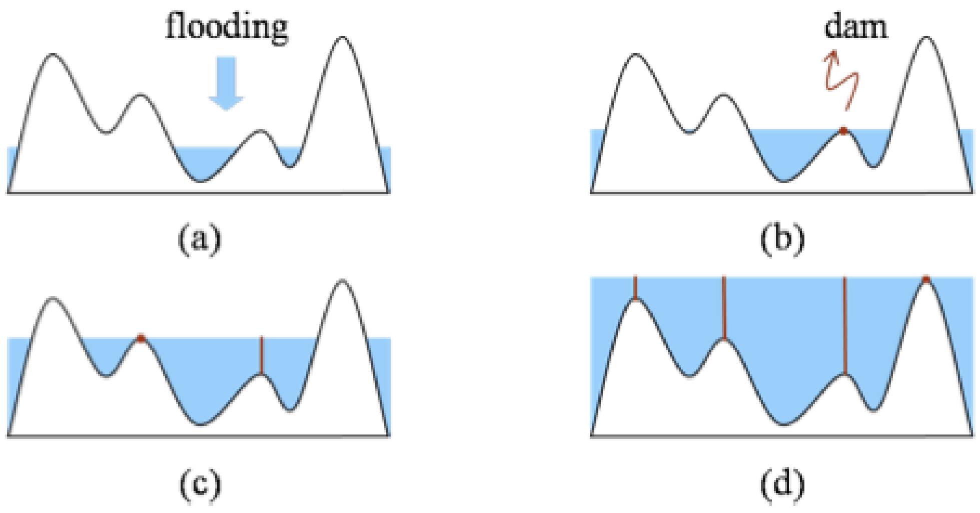


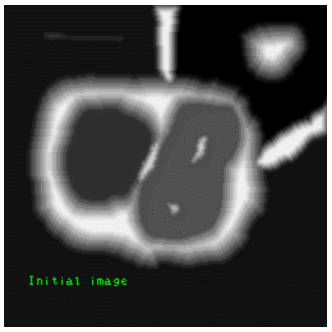
Figure 3.3: Concept of Watershed by flooding. The data is viewed as a height map within which catchment basins are progressively flooded. When they merge, a segmentation barrier is created. [Lakshmanan, 2013]

The algorithm is as follow:

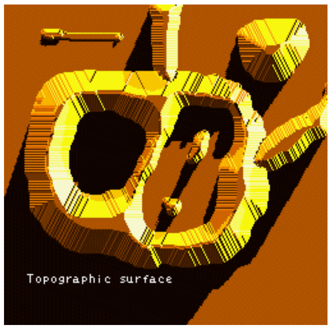
1. First the levelsets of the pixels are computed. The flooding starts with the pixel with the minimum value. A label is assigned to it.
2. The neighboring pixels of each marked area are inserted into a priority queue with a priority level corresponding to the gray level of the pixel (the position of its value in the level set).
3. The pixel with the highest priority level is extracted from the priority queue. If the neighbors of the extracted pixel that have already been labeled all have the same label, then the pixel is labeled with their label. All non-marked neighbors that are not yet in the priority queue are put into the priority queue.
4. Redo the previous step until the priority queue is empty.
5. The non-labeled pixels are the watershed boundaries.

An example using test data is presented in Figure 3.4. The result of the Watershed when applied to an horizontal slice of actual weather data (vertical helicity) is shown in Figure 3.5.

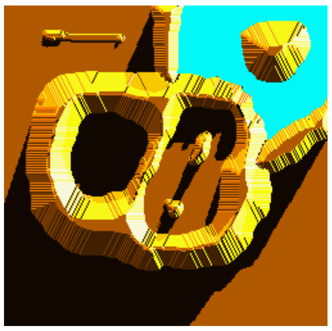
The result is not satisfying for two reasons: First, too many regions are segmented, which results in too many very small region that are neither relevant nor conclusive to the study. Secondly, the regions that are correctly located and that do engulf regions of interest are too big and do not segment features of the supercell accurately enough. Both of these problems are addressed by the following enhanced version of the Watershed Algorithm.



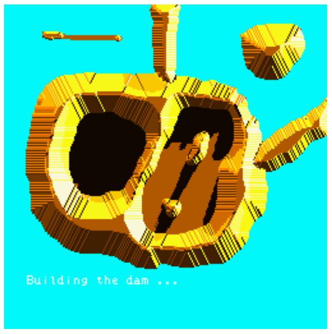
(a) Original grayscale image



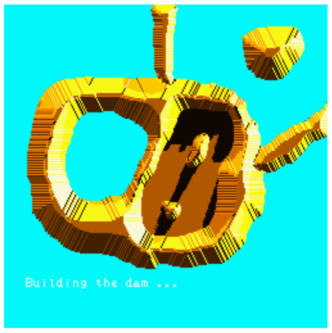
(b) Topographic analogy of the grayscale image



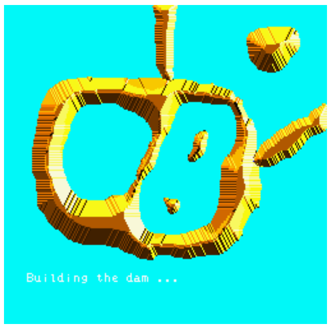
(c) region growing from the base of the levelset



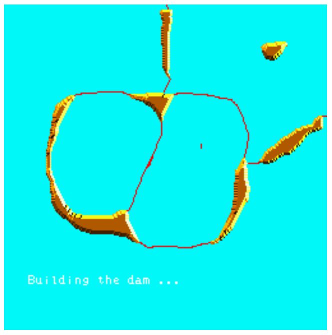
(d) region growing along of the levelset



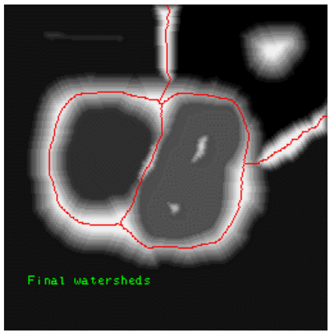
(e) region growing along of the levelset



(f) region growing along of the levelset



(g) Where the labels meet they form barriers.



(h) Final segmentation. The barriers are in red.

Figure 3.4: Progression of a 2D Watershed Algorithm [Beucher, 2010]

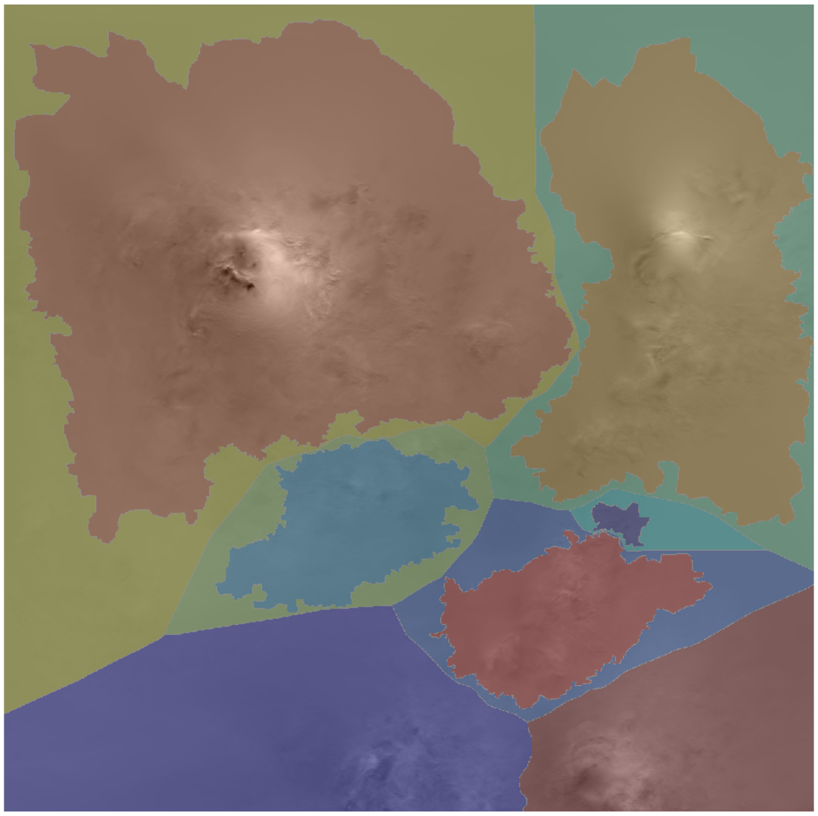


Figure 3.5: Segmentation the vertical helicity field using a standard Watershed Algorithm. Each segmented cell has been colored. Theoretically, the only region we are interested in in the big red cell in the top left corner. The other cells are the result of over-segmentation.

3.2.2 The Enhanced Watershed Algorithm

The Watershed Algorithm suffers from a severe drawback: over-segmentation. It produces a distinct labelled region for each local minimum, which, in practice, is too many regions. To reduce the effects of this phenomenon, several methods exist. The most notable ones are establishing a sliency requirement and using markers.

- *Establishing a saliency requirement.* This method imposes a criterion that has to be met before two growing regions can meet and create a ridge.
  - The most common salicency criterion is a vertical criterion called minimum watershed depth. The watershed depth is the difference in height between the watershed minimum and the lowest boundary point. In other words, it is the maximum depth of water a region could hold without flowing into any of its neighbors. Thus, a watershed segmentation algorithm can sequentially combine watersheds whose depths fall below the minimum until all of the watersheds are of sufficient depth. However it appears that this criterion is insufficient for meteorological data.
  - An horizontal saliency criterion has been introduced by Lakshmanan [Lakshmanan et al., 2009]. The observation made was that vertical saliency checks perform poorly on weather data because there is a lack of conceptual coherence between the depth criterion of the watershed and the definition of a storm cell. According to the work of Augustine and Howard (1988), a storm entity is defined as salient if it meets a certain size threshold. Therefore, having a watershed that

is based on a size criterion makes more sense thereby providing better results. The full algorithm is described in Lakshmanan's article [Lakshmanan et al., 2009].

The depth and size measurement can be combined using different approaches to provide new saliency measurements, but this study will only use the size measure as a region merging criterion.

- *Using markers*: Markers are seeds from which the flooding will start. The algorithm is similar to a regular flooding algorithm but the growth can only sprout from the seeds. The consequence of this method is that in the final segmentation, there can be only as many segmented regions as there are markers. Indeed, two regions cannot merge because if they cross path, a boundary will be formed. Additionally, there cannot be more labels than numbers of markers since no new label can be created and grow if there isn't a markers to start it. Hence, when applying a Watershed Algorithm, it a good idea to preprocess the data to select rough locations for the Regions of Interest (ROI) that can serve as markers as outlined below.

  1. A set of seeds are chosen, usually local extrema. Each is given a different label.
  2. The neighboring pixels of each marked area are inserted into the seed queue. They will constitute the seeds for the next iterations. The regions are grown from the seeds independently of the level set value of the pixels. If a neighboring pixel is already labeled with a label other than that of the seed, it forms a barrier.
  3. Redo the previous step until the queue is empty.

For ease of implementation purposes, the watershed-by-flooding algorithm will be used. This algorithm suffers from the fact that the image has to be quantized to define levelsets and therefore loses a lot of precision, depending on the level of quantization. The watershed-by-topographic-distance works with gradient descent and can be numerically solved more precisely. The watershed-by-flooding algorithm is easier to use both with the size criterion and the region markers, whereas the watershed-by-topographic-distance algorithm is easier to use with the depth criterion because all that is needed is to measure the progress of the gradient descent.

Both saliency criterion and markers have been tested. For the marker's algorithm, the following testing procedure was used:

1. Preprocessing of the image using anisotropic diffusion.

2. From these preprocessed images we want to extract the foreground and the background. The foreground contain all the clouds and therefore will contain the peaks in velocity magnitude, notably vertical velocity, and helicity.

3. Using automatic thresholding algorithms such as Otsu's variance based method [Nobuyuki, 1979] or an entropy based method to select a region of interest to serve as a foreground marker. The main updraft is obtained by thresholding the vertical velocity and vertical vorticity.

4. To create the background markers, the idea is to detect zones that are distant enough from all the local maxima detected by the foreground markers and homogeneously spread over the image so that the basin growths of the watershed do not oversegment in any particular direction or zone of

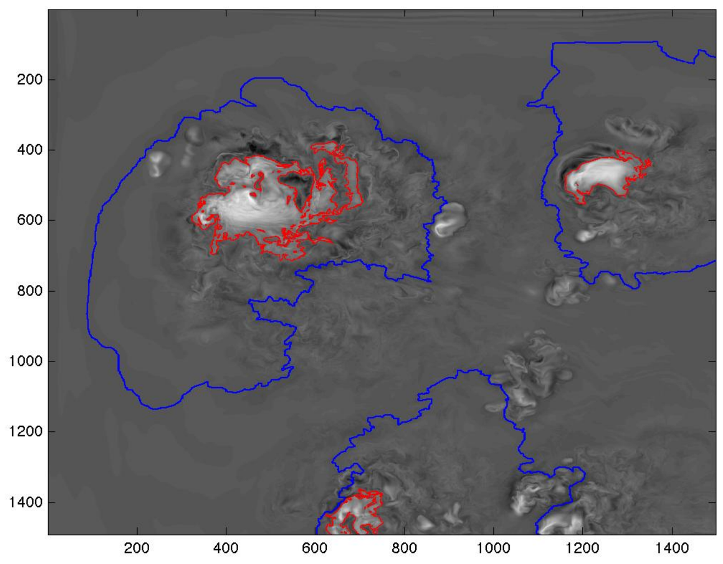
the image. The idea we have been trying is to start from the foreground markers binary mask and apply a Watershed Algorithm on the binary mask growing from the foreground markers. This is equivalent to finding Voronoi cells centered on the markers. The boundaries found by this first Watershed Algorithm are distant from all the foreground markers. Hence they are considered to belong to what can reasonably be considered the background.

5. Finally a Watershed Algorithm is once again used to grow from the markers.

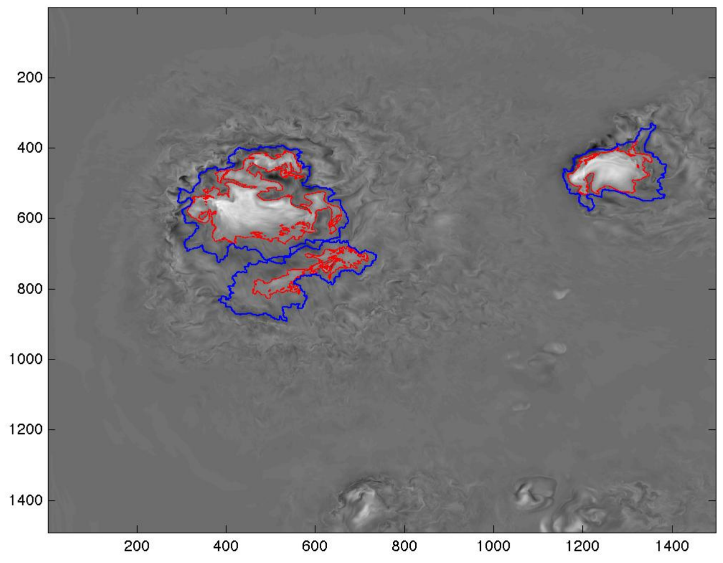
Figure 3.6 shows the comparative segmentation obtained by applying both a saliency based watershed and a marker based watershed on the simulated weather data.

Computationally the size saliency criterion algorithm is faster because it does not require any automatic thresholding or additional runs of the Watershed Algorithm in order to respectively find the foreground and background markers. Also the result provided by the size saliency criterion empirically proves to have a more accurate result: The contours of the segmentation mask of the markers based algorithm are more loose and do not delimitate the supercell as precisely. This can be due to the great variability of the marker based watershed's output according to the initial markers' size. In theory, the size and shape of a marker do not influence the result of a segmentation. However this is only true as long as the marker is entirely contained within the expected boundary of the final segmentation mask.

Therefore, this study uses the segmented results obtained by flooding a size-governed watershed. The output of the watershed is a labelled 3D image of the



(a) Vertical helicity at timestep 4200



(b) Vertical helicity at timestep 5100

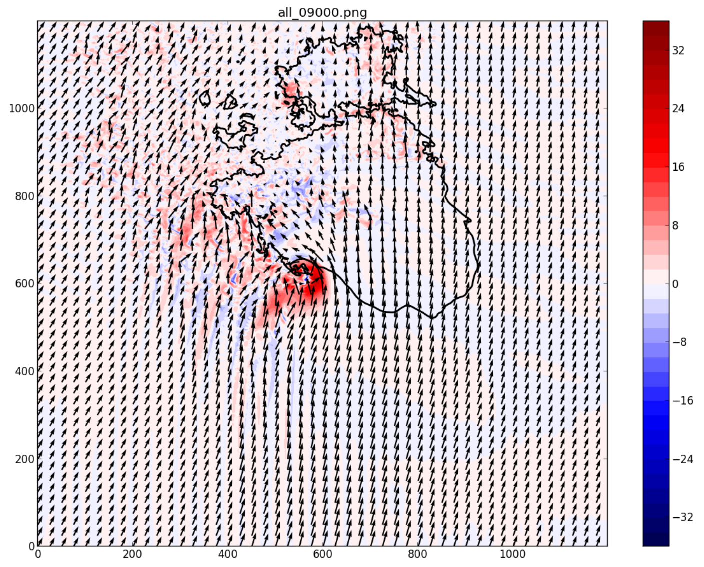
Figure 3.6: Comparison of marker based watershed and saliency based watershed. The Watershed Algorithm was run on the 3D helicity field. The 6km horizontal slice is displayed. The marker based watershed produces the blue cells and the horizontal saliency based watershed produces the red cells. The bottom cells are remnants of boundary conditions numerical artifacts. Only the main updraft (on the left of a. and b.) is kept (See Section 3.2.4).

same dimension of the original input image. Each pixel belonging to the same identified object has the same distinct label value. Pixels that do not belong to any segmented object/region have a 0 value. The output of this Watershed Algorithm is then binarized (zero thresholded) and sent to a hysteresis filter.

3.2.3 Reduction of the dimensionality of the problem: Projection

One of the original ideas that this study explored was to use Statistical Shape Analysis on the output of the Watershed Algorithm in order to classify the thunderstorms according to their shape. The principle is to accumulate and average multiple images in order to create a mean vertical helicity shape both for supercell thunderstorms and nonsupercell thunderstorms, similar to a k-means algorithm. Although this idea could still be relevant when applied to radar data (reflectivity), it was deemed inappropriate for helicity fields. The segmented updrafts of right mover supercells proved to be too small and their shape held too much variation to provide any relevant mean shape.

It was decided that, in order to improve the performance of the overall pipeline, the binary mask provided by the supercell would be collapsed by summation along its vertical axis, between the 2km and 12km heights. The result is a 2D image with values bounded between 0 and the height (in pixel) of the original 3D field (Figure 3.7). All non zero values denote places somewhere along the vertical profile of the helicity field. The higher the value, the stronger the rotating updraft is at this position.



(a) Slice of raw, unprocessed data.

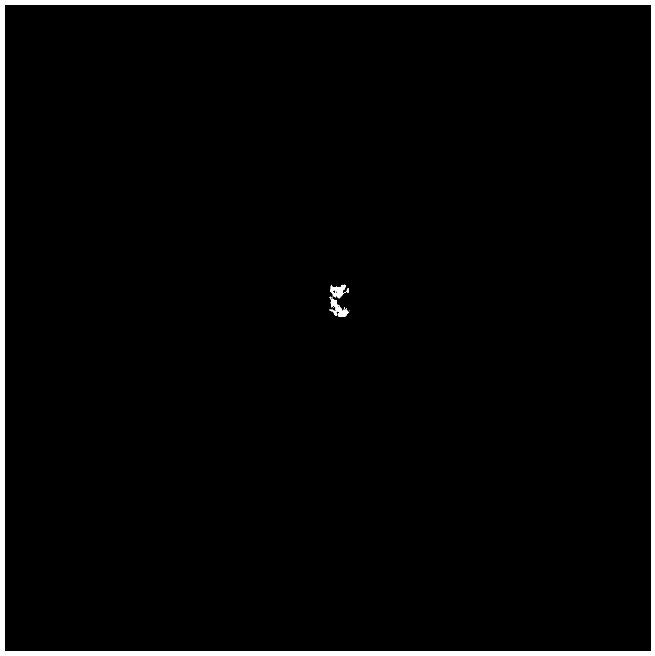

(b) Vertical helicity 2D projected field

Figure 3.7: for Figure a., the color scheme represents the magnitude of w, the vertical velocity field. The arrows represent the horizontal projection of the (u,v) and the contour lines represent the shape of the storm's reflectivity. This representation is more practical to represent the areas of colocation of updrafts and vorticity.

3.2.4 Post-Segmentation processing: Hysteresis

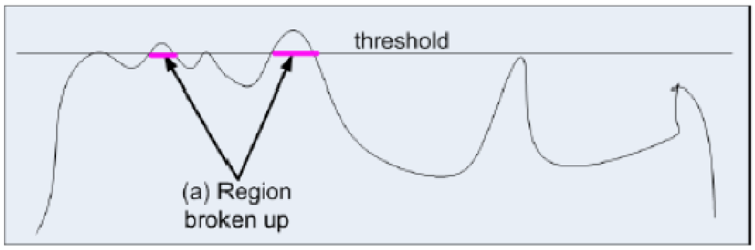
Despite the good performances, the resulting image is plagued by the presence of small regions that have been segmented because of the presence of local maxima. Supercells are characteristically defined by strong rotating updrafts that can reach the upper layers of the troposphere. Therefore we are interested in only keeping the regions that contain the global maximum value i.e. the regions where the updraft is the strongest. We use an hysteresis algorithm to perform this task.

Thresholding is subject to either noise proliferation or object erosion. Even when adaptive thresholding is used, optimal thresholds only guarantee a theoretical and mathematical optimum separation as opposed to a practical separation. Thresholds assume that the complete object is characterized by values exceeding the threshold value. But in practical applications, objects' pixel values may vary to a point where they can be under the optimal threshold. If the threshold is high enough, as in Figure 3.8a, the object we want to segment may be truncated on its edge. On the other hand, if the threshold is too low, as in Figure 3.8b, the segmented object may be complete, but too many noisy pixels can appear.

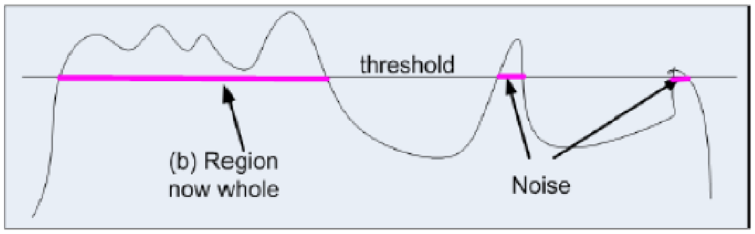
The hysteresis algorithm is a region growing algorithm that addresses this problem by adding criteria. The hysteresis algorithm is characterized by two thresholds: a lower threshold and an upper threshold. The assumption is that the higher a pixel value, the more important is the pixel and the more likely it belongs to an object that we want to segment. Each pixel of the region must have a value greater than the lower threshold but must also be connected to at least one pixel that has a value above the upper threshold, as in Figure 3.8c.

This guarantees that objects can be segmented with enough buffer that they are not truncated, but isolated semi-high valued pixels are not selected.

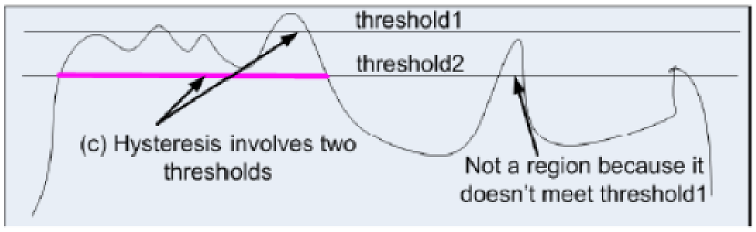
This method has the advantage of restricting the number of segmented objects to those that have at least a certain degree of importance. In our case we want to select the region of space that contains the strongest rotating updraft.



(a) Too high thresholds do not get object entirely



(b) Too low thresholds allow for too much noise



(c) Hysteresis only allows to grow objects down to the lower threshold if the criterion of the upper threshold is met.

Figure 3.8: Concept of hysteresis thresholding. [Lakshmanan, 2013]

For vertical helicity data, we choose to keep only helicity values greater than $15m^2s^{-2}$ [Bunkers et al., 2006; Markowski and Richardson, 2010]. Consequently the lower threshold is set to 15 and the higher threshold is set to the maximum value of the helicity field. The resulting data is an image that contains only one region: the location of the strongest rotating updraft of the supercell thunderstorm.

3.3 Standardization: Registration

As a reminder, this study focuses of identifying supercells from nonsupercellular storms. The watershed extracted a contour from the fundamental velocity variables' data. The output of the watershed algorithm enables us to have an approximate overall shape of the vertical helicity, at a given moment, in the form of a binary mask from which we can extract the contour. From this data, the goal is to classify the storms according to their shape. However, every storm differs, be it by their internal structure or by geographical orientation. While structure differences can be used during the classification process and should not be normalized, positioning and orientation can significantly decrease the measure of similarity used in classification.

The goal of the standardization process is to remove as much of translation and rotation component of the supercell so as to reduce the variability between the different storms solely to their size (scale) and structure (shape).

3.3.1 Principle

Image registration is the process of overlaying two or more images of the same scene taken at different times, from different viewpoints, and/or by different sensors. It geometrically aligns two images, the reference image, also called fixed image, and the target image, also called moving image. The aim of the registration process is to find the optimal transform that will map the moving image to the fixed image.

The process is iterative and follows the flow chart in Figure 3.9. The fixed image and the moving image are first compared using a metric and are then sent to an optimizer that finds a first set of parameters that are sufficient to describe

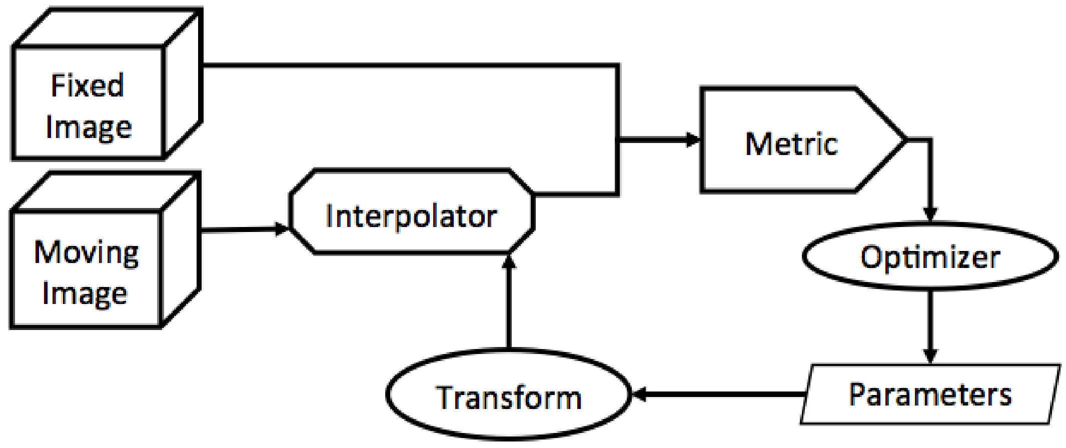


Figure 3.9: Registration flow chart.

the transform. The set of parameters depends on the type of transform that is chosen to go from the moving image to the fixed image. The transform is then applied to the moving image. The transformed image is compared once again to the fixed image to judge the validity of the transform. While the validity is deemed insufficient and end conditions specified by the user are not met, the optimization process iterates.

The quality of the registration process greatly depends on the choice of the multiple processes that compose it:

- *The Transform:* The type of transform that is going to map the moving image to the fixed image has to be chosen by the user before the registration process: The transform can be rigid, in which case it is composed of a translation, or a rotation, or a combination of both. Or the transform can be deformative, in which case the transform also includes an anisotropic scaling. In this study, both types of transforms have been tried, and rigid transformations empirically proves to give better results.

- *The Metric:* The metric is the function that admits the two images as inputs and returns a scalar that quantifies how similar the two images are. The similitude between the two images is defined by how well their shape, volume and intensity values correspond pixel per pixel. Multiple shape metrics have been tried.

  In the following metric examples X is the fixed image, Y is the moving image, x is a pixel of X and y is a pixel of Y.

  - The *Hausdorf* distance measures how far two subsets of a metric space are from each other.

    $$d_{\mathrm{H}}(X,Y) = \max\{\, \sup_{x\in X} \inf_{y\in Y} d(x,y),\ \sup_{y\in Y} \inf_{x\in X} d(x,y)\,\},$$

    where,
    $d_{\mathrm{H}}(X,Y)$ is the Hausdorff distance between X and Y

  - The *Mutual Information* of two random variables is a measure of the variables' mutual dependence.

    $$I(X,Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)\log\left(\frac{p(x,y)}{p(x)\,p(y)}\right),$$

    where,
    $I(X,Y)$ is the mutual information between X and Y
    $p(x)$ and p(y) are the marginal probability distribution functions of X and Y respectively
    $p(x,y)$ is the joint probability distribution function of X and Y

– The *Mean Squared Error* (MSE) measures the average of the squares of the "errors." The error is the amount by which the estimated value differs from the reference truth value.

$$\mathrm{MSE}(X, Y) = \frac{1}{n} \sum_{(x,y)\in(X,Y)} (x - y)^2.$$

where,

$\mathrm{MSE}(X, Y)$ is the Mean Squared Error between X and Y

n is the number of pixels in X and Y

After trial and error, we settle for MSE as it consistently provides better empirical results. Mutual Information gives more inconsistent results and takes a higher number of iteration for the overall system to converge to an acceptable solution, if convergence there is. The Hausdorf depends greatly on the shape of the ROI measured. Because of the fuzzy nature of the weather data as well as the size of the segmented blobs of vertical helicity, this measures is too inconsistent as well.

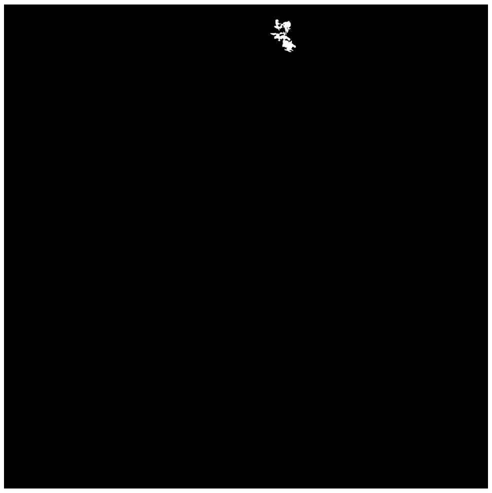
- *The Optimizer:* Optimizers represent the mathematical procedure by which a minimization problem is solved. Examples of optimizers that have been considered for this study include the Gradient Descent, the Regular Gradient Descent, the Conjugate Gradient, the simplex algorithm (Amoeba), LBFGS (Broyden, Fletcher, Goldfarb and Shannon minimization) and evolutionary algorithms. Due to time constraints a simple Conjugate Gradient Algorithm has been implemented.

- *The Interpolator:* When the moving image is transformed, it is important to interpolate its transformed pixel values into a grid that can be compared to the fixed image. Linear interpolation is used.

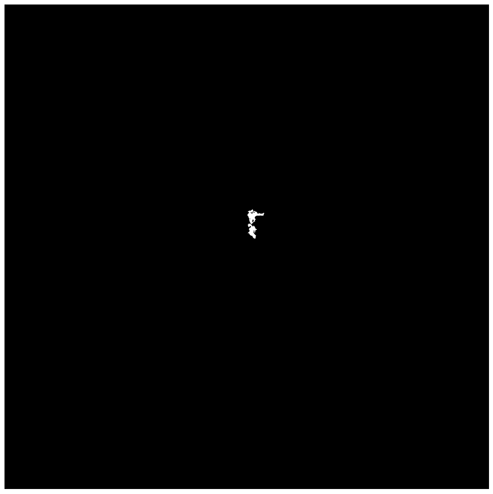
The iteration loop ends when a user specified condition is met. That condition can be either a number of iteration or a metric value that is satisfying enough, i.e. the moving image to which the transform has been applied is close enough to being the fixed image.

3.3.2 Results

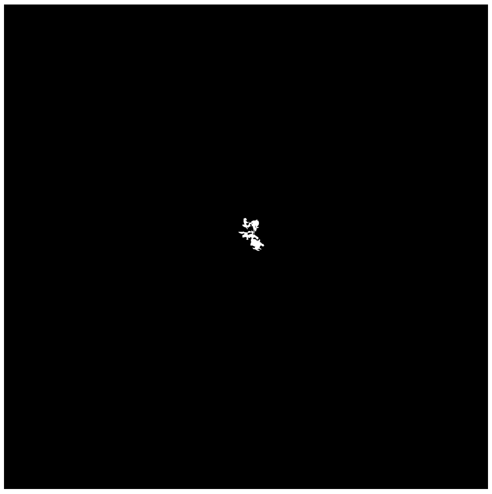
The supercells themselves are mostly centered: the 1200x1200x51 window that contains the simulation adjusts for thunderstorm motion of $6ms^{-1}$ to the north and $8ms^{-1}$ to the east. Consequently all of the updrafts are mostly centered. The registration process eliminates most of the remaining displacements of the updraft. The fixed image that is served as a reference is the image of the timestep that predates the moving image (Figure 3.10b). Consequently the first image is arbitrarily centered and all timestep that follow are registered on its predecessor. In the best case scenario, when using a rigid transform, this procedure is equivalent to registering all images on the first. The registration process is very dependant on the parameters mentioned in Section 3.3.1. Therefore, much tweaking is needed and, due to time constraints the result is only approximate.



(a) Original moving image; timestep 9300



(b) Fixed image; timestep 9000



(c) Final moving image

Figure 3.10: Registration of a timestep; timestep 9300

3.4 Classification: Support Vector Machines

The idea behind the classification method is to label each timestep of a storm with a boolean. If the image of the collapsed storm produced by the projection of the watershed mask contains a right-mover supercell, the boolean label for this timestep is set to true. Otherwise, the boolean label for the timestep is set to false. The objective is to learn a recurring pattern that is crucial in the classification. Multiple data mining and machine learning methods exist to classify data using this truth value boolean. Consequently, the classification process can be executed using multiple methods. In this study we decided to focus on a machine learning technique (SVM) and two numerical optimization techniques (LDA and QDA). The overall pipeline is conserved, only the classification algorithm differs.

The most appropriate way to classify weather data, for which we know the truth, into two classes is to use supervised learning algorithms. Supervised learning is the machine learning task of inferring a function from labeled training data. Such methods include Support Vector Machines, decision trees, k-nearest neighbor algorithm, and Neural Networks (Multilayer perceptron). The danger of supervised learning methods is that they are prone to overfitting. Overfitting is the natural behavior of supervised methods to learn exactly the data it is given. This limits the generalization of the classifier to any set of data. In order to solve this problem, the overall data is separated into different working sets: the training set, which is used to train the classifier, the validation set, which is used to evaluate the convergence of the classifier, and the testing set, which used to actually classify the new data.

3.4.1 Principle of SVMs

In this study, we decided to classify our data using Support Vector Machines (SVM) because they avoid some weaknesses that can be found in other supervised learning method. Because they are mathematically optimal, they can be resistant to overfitting because additional data only affect the results if they are support vectors. Support vectors are the data points that lie closest to the decision surface. They are the most difficult to classify. They have direct bearing on the optimum location of the decision surface. In practice this depends on the careful choice of parameters [Cawley and Talbot, 2007, 2010].

Support Vector Machines are kernel based methods. Kernel methods map the data into higher dimensional spaces in the hope that in this higher-dimensional space the data could become more easily separated. This mapping function does not need to be computed because of a tool called the kernel trick. The kernel trick is a mathematical tool which replaces the dot product by a kernel function which permits us to project the data in the higher-dimensional spaces.Going from low dimensional space to high dimensional space ensures that there exist an hyperspace in which all the data can be separated by an hyperplane.

If the training data are linearly separable, we can select two hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. The region bounded by them is called "the margin". These hyperplanes can be described by the equations.

$$\mathbf{w} \cdot \mathbf{x} - b = 1$$

and

$$\mathbf{w} \cdot \mathbf{x} - b = -1.$$

In our case, we want to find the hyperplane that divides the timesteps labeled as true (contain a right-mover supercell) from those labeled as false (do not contain right-mover supercells). Any hyperplane can be written as the set of points $\mathbf{x}$ satisfying

$$\mathbf{w} \cdot \mathbf{x} - b = 0$$

where $\mathbf{w}$ is the normal vector to the hyperplane. The objective function to minimize is a constraint problem:

$$\arg \min_{(\mathbf{w},b)} \frac{1}{2} \|\mathbf{w}\|^2$$

subject to the following constraints

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1$$

where

x_i is an observation. In our case, x_i is the 2D collapsed image output from the watershed.

y_i is the truth value associated with the observation x_i. In our case it is 1 when the timestep features a right-mover supercell thunderstorm, 0 otherwise.

The parameter $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$

However, it is not always possible to obtain a clean cut separation between the two classes. In 1995, Vapnik suggested a modified maximum margin idea that allows for mislabeled examples [Vapnik and Cortes, 1995]. If there exists no hyperplane that can exactly split the examples, the Soft Margin method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split.

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 - \xi_i \quad 1 \leq i \leq n.$$

where ξ_i is a measure of the degree of misclassification of the observation x_i.

The objective function to minimize takes the form of :

$$\arg\min_{\mathbf{w},\xi,b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \right\}$$

Such quadratic problems are solved using the dual formulation of the problem, in which case the parameter C is used to bound the lagrangian coefficients. C is called the box constraint. Increasing the size of the box constraint can decrease the number of support vectors (improved classification), but also might increase training time.

Choosing the most appropriate kernel depends on the problem and tuning the parameters is a difficult task [Howley and Madden, 2007]. We chose a kernel that is easily understandable and that is known for fitting most general purpose SVMs, the Gaussian Radial Basis Function (RBF) kernels:

$$k(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right)$$

where

k is an odd sized convolution window

x and y are the local coordinates within the kernel

σ is the standard deviation of the RBF. Adjustable parameters are the standard deviation σ, and the box constraint C.

We use cross validation to find the best value for those two parameters.

3.4.2 Cross validation

Cross validation is the process of varying the determination of the training set and the testing set in order to limit the specificity of the kernel to the data. The two main cross validation techniques:

- The holdout method is the simplest kind of cross validation. The data set is separated into two sets, called the training set and the testing set. The model is fit using the training set only. Then the model is used to predict the outcome values for the data in the testing set (it has never seen these output values before). The errors it makes are accumulated to give the mean absolute test set error, which is used to evaluate the model. Its evaluation can have a high variance. The evaluation may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made.

- K-fold cross validation is one way to improve over the holdout method. The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set k-1 times.

> The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation.

The 5-fold cross validation method is particularly adapted to our case: we possess 5 storms in total. Each storm is individually taken as testing set and is pitted against the other 4 storms that form the training set. The algorithm is run 5 times: once for each storm. For each run, the training set is divided in 2 equal and uniformily distributed parts. The first half of the training set is run multiple times, each time with a different (box constraint, standard deviation) pair. The pair that provided the best classification is kept, and reused to train a new SVM with the second half of the training set. This procedure ensures the best possible choice of parameters (C, σ) and avoids possible overfitting of the SVM.

The results are presented in the following section.

Chapter 4

Evaluation and Case Study

4.1 Discussion of the parameters

The cross validation that gives us the best values for the box constraint and the standard deviation is presented in Figure 4.1.

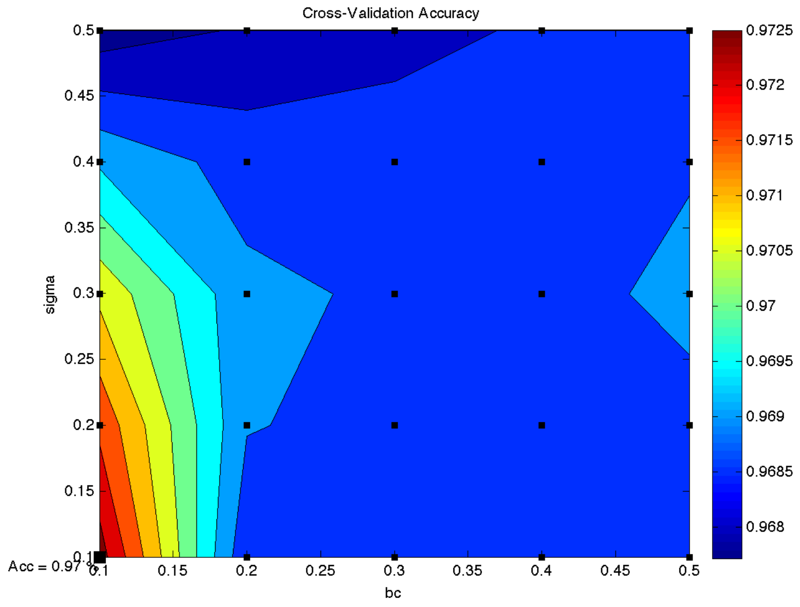
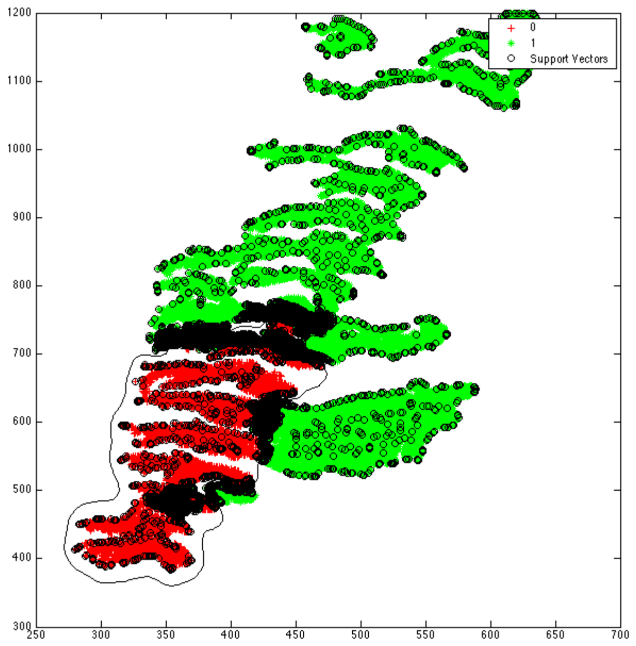


Figure 4.1: The color component represents the accuracy of the SVM trained with the corresponding box constraint (bc) and sigma. The actual pair used to derive this graph are represented by the black markers.

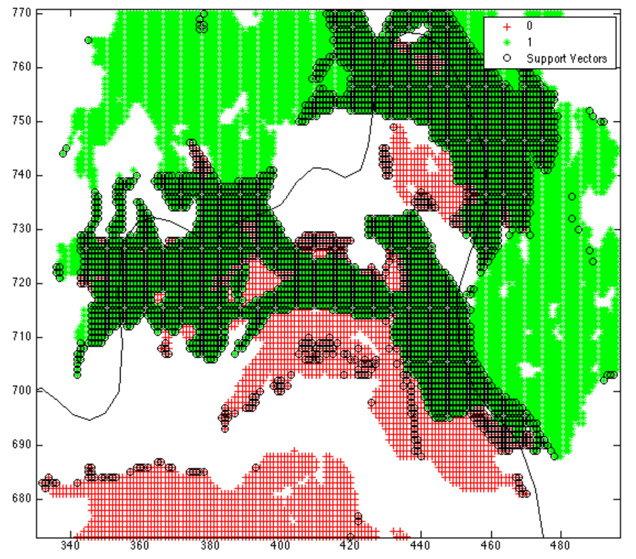
It appears that smaller values for both the box constraint and the standard deviation provide better results. For values that lower than (0.08,0.08) the systems loses stability and does not converge everytime. For values lower (0.05, 0.05), the system almost nerver converges. To keep the system stable, we keep the value (0.1, 0.1), which assures convergence.

The fact that small values of box convergence work better is explained by the fact that the more bounded the lagrangian multipliers of the minimization problem are, the less distinct is the soft margin that separates the two classes. More support vectors are needed to accurately converge (Figure 4.2). If the margin had not been a soft margin, i.e. if a margin that effectively separated the data had existed, higher values for the box constraint would have been better, as they would have decreased the number of support vectors, thereby limiting the risks of overfitting and improving the generalization of the kernel.

The occurrence of small values of standard deviation is explained by the similar reasoning: When the data is projected back from the higher dimensionality, the mapping function based on the Gaussian kernel of the SVM is applied to the support vectors. The Gaussian kernel computed with a support vector is an exponentially decreasing function whose maximum value is at the support vector and which decreases in all directions around the center. The SVM classifier with the Gaussian kernel is simply a weighted linear combination of the kernel function computed between a data point and each of the support vectors. Therefore, the lower the standard deviation, the lower the radius of influence of each support vector, which allows for a more flexible, less "locally linear" separation. Theoretically a combination of low standard deviation and low box constraint imply a separation whose local smoothness rarely goes beyond class $\mathbb{C}^0$. In our case, more flexible separation curves give better results.



(a) Overall SVM projection



(b) Zoom on a separation margin

Figure 4.2: (b) is a zoom on the margin between the two classes featured in (a) In this region most of the data points are support vectors.

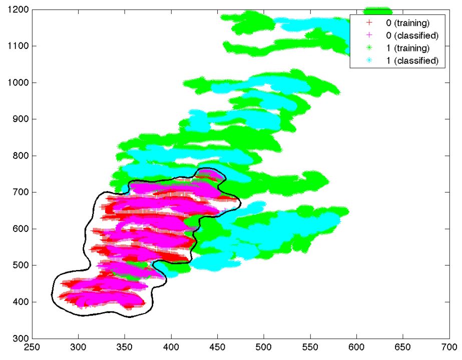
4.2 Case studies

To evaluate the performance of the final classification performed by the algorithm (Figure 4.3), we first check the counting matrix indicates the number of correctly classified and number of misclassified pixels for each storm. Table 4.4 represents the counting matrix for the SVM trained on storms 1 to 4 and tested on storm 5.

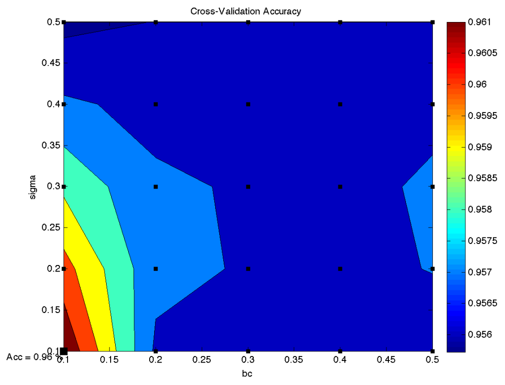
For this SVM most of the misclassified pixels are pixels that do not necessarily depict a supercell formation (truth label = 0) but have been considered as such. It is interesting to see where those misclassified pixels come from.

From Figure 4.5 we conclude that timesteps 3300 and 3600 are likely misclassified meaning that a lot of pixels from the helicity structure segmented during those timestep have been misclassified by the SVM. How much of those pixels have been misclassified exactly is the answer that is provided by Figure 4.6. To each timestep is associated the percentage of correctly classified pixels.

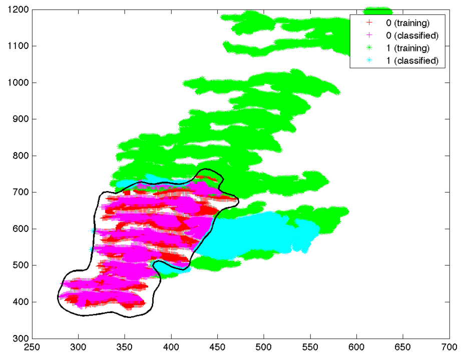
The closer to 1 the value of the timestep per timestep accuracy is, the more reliable the classifier is for the given timestep. Depending on how accurate the user wishes his prediction to be, the threshold that decides whether an accuracy is acceptable can be set to various values. Figure 4.6 represents the timestep per timestep accuracy for various thresholds. If we consider that having a minimum of 50% of the timestep classification correct, then the classification for the entire storm is more likely to be correct. If we raise our expectations for correctness to 80%, then several timesteps are considered statistically wrong.



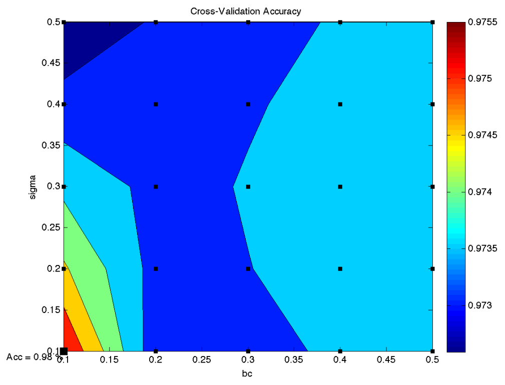
(a) SVM tested on storm 1



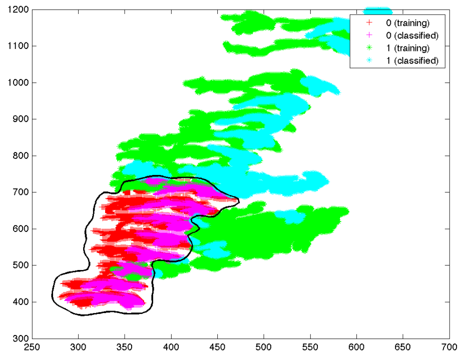
(b) Parameter crossvalidation



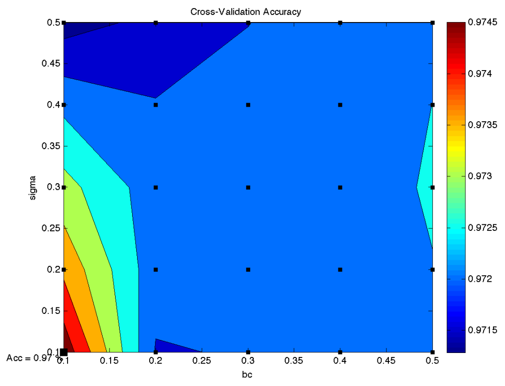
(c) SVM tested on storm 2



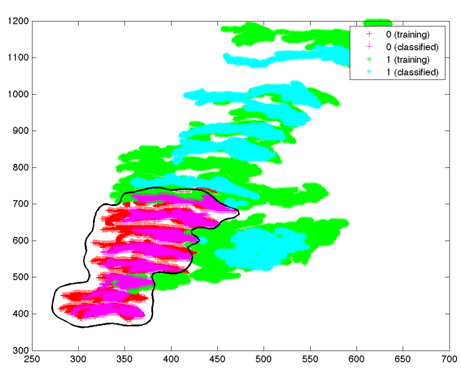
(d) Parameter crossvalidation



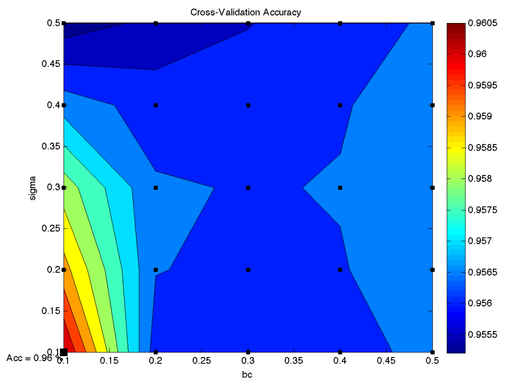
(e) SVM tested on storm 3



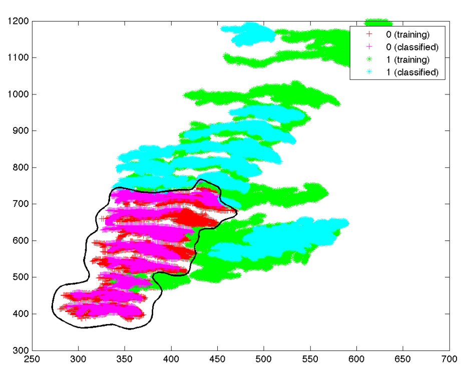
(f) Parameter crossvalidation



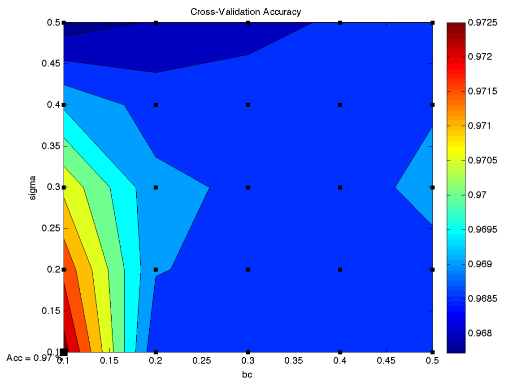
(g) SVM tested on storm 4



(h) Parameter crossvalidation



(i) SVM tested on storm 5



(j) Parameter crossvalidation

Figure 4.3: Final converged separation obtained for each storm when all the other storms are used to train the SVM. The red entries are the thunderstorm pixels of the training set. The green entries are the supercell pixels of the training set. The pink entries are the testing set's pixels classified as thunderstorm pixels. The teal entries are the testing set's pixels classified as supercell pixels.

		Classified as		Accuracy
		0	1	
Truth value	0	9589	485	0.983
	1	64	21532	

Figure 4.4: Counting Matrix for the SVM trained on storms 1 to 4 and tested on storm 5 (Figure 4.3i)

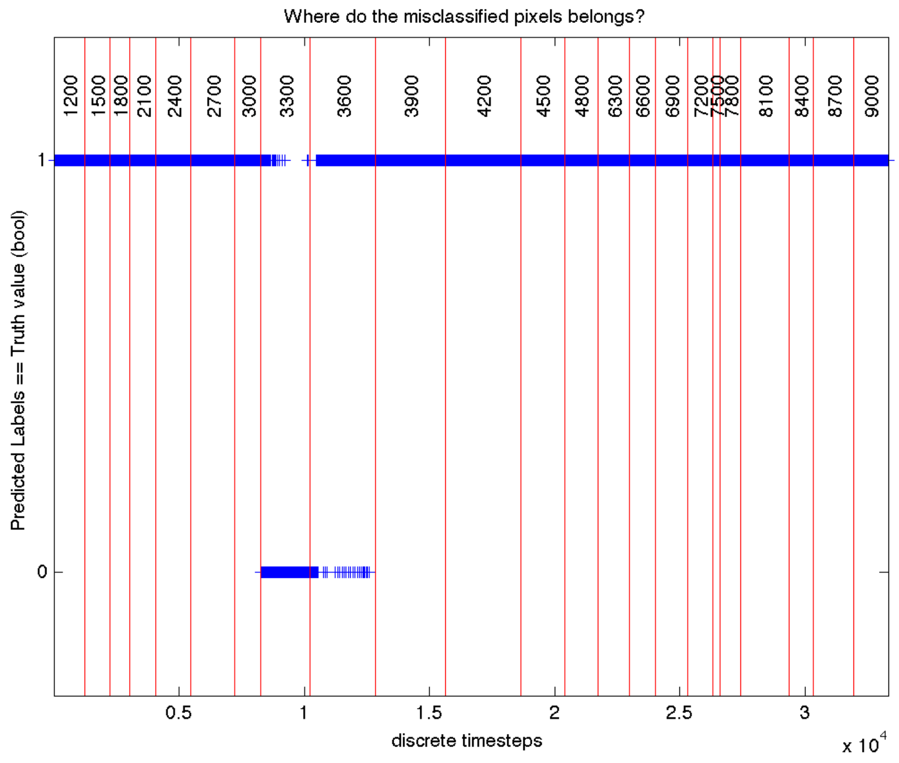


Figure 4.5: Timestep per timestep classification of the pixels. Each pixel selected by the segmentation process is labeled with the values learned from the SVM. A value of 1 indicates that the pixel has been correctly classified. A value of 0 indicates a misclassified pixel. The ordering of the pixels within a timestep is arbitrary and has no influence

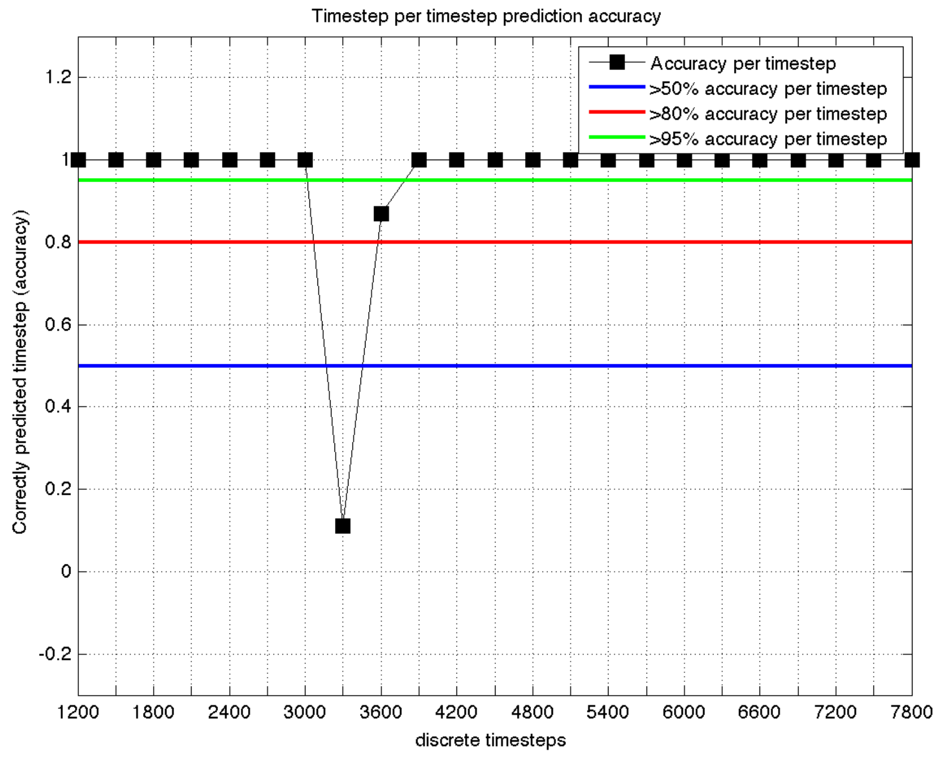


Figure 4.6: Accuracy of the classification per timestep

The same phenomenon occurs if 95% of each timestep is needed to consider a classification valid (timestep 3600).

Figure 4.7 shows how the overall choice of this accuracy threshold impacts the detection of the timestep where a thunderstorm evolves into a supercell.

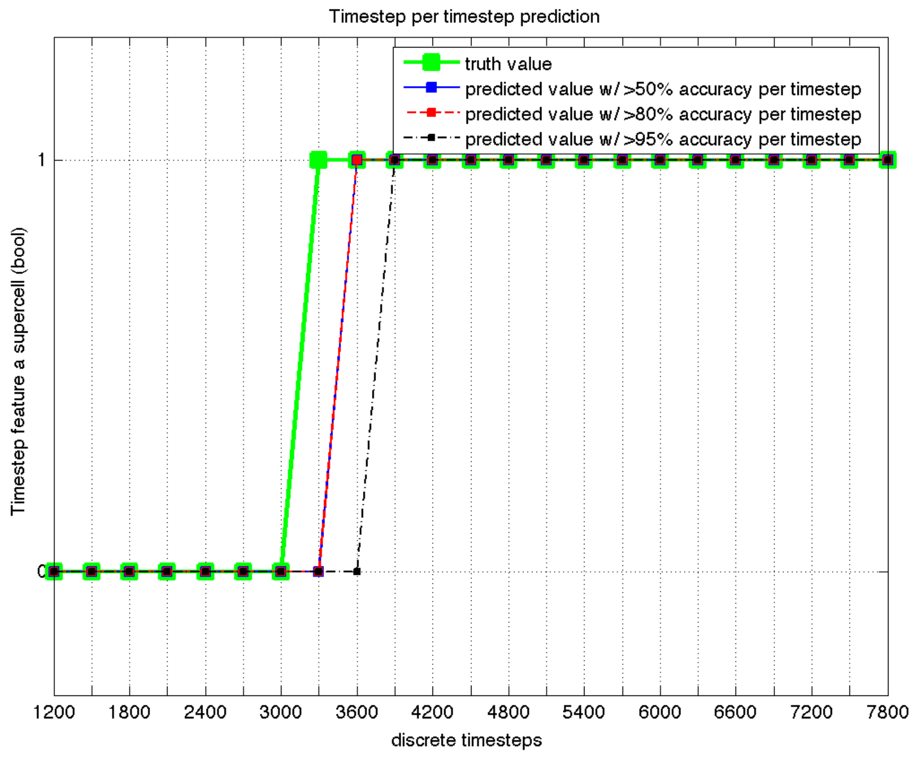
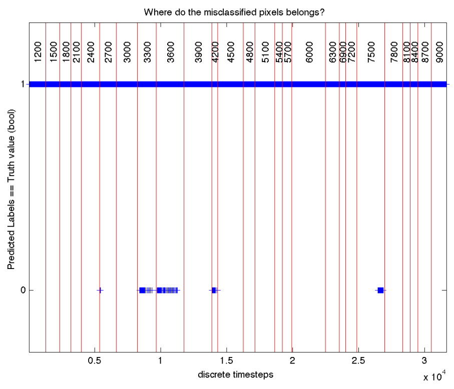


Figure 4.7: Prediction on the evolution of a storm. The green line represents truth value boolean that classifies each timestep of a storm. In this case thunderstorm 1 is not considered to be a supercell before it reaches timestep 3300. The 50% and 80% classifications predict a very close approximation of the outcome. The several misclassified timesteps from Figure 4.6 for the 95% classification decrease the overall accuracy of the timestep classification for higher and more demanding thresholds.

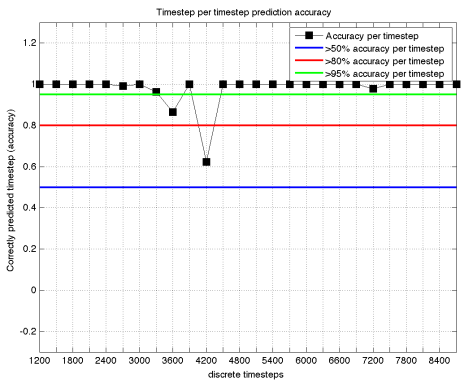
The results for the SVMs trained and tested on the different storms are presented in Figures 4.8, 4.9, 4.10 and 4.11.

		Classified as		Accuracy
		0	1	
Truth value	0	9573	509	0.982
	1	76	21508	

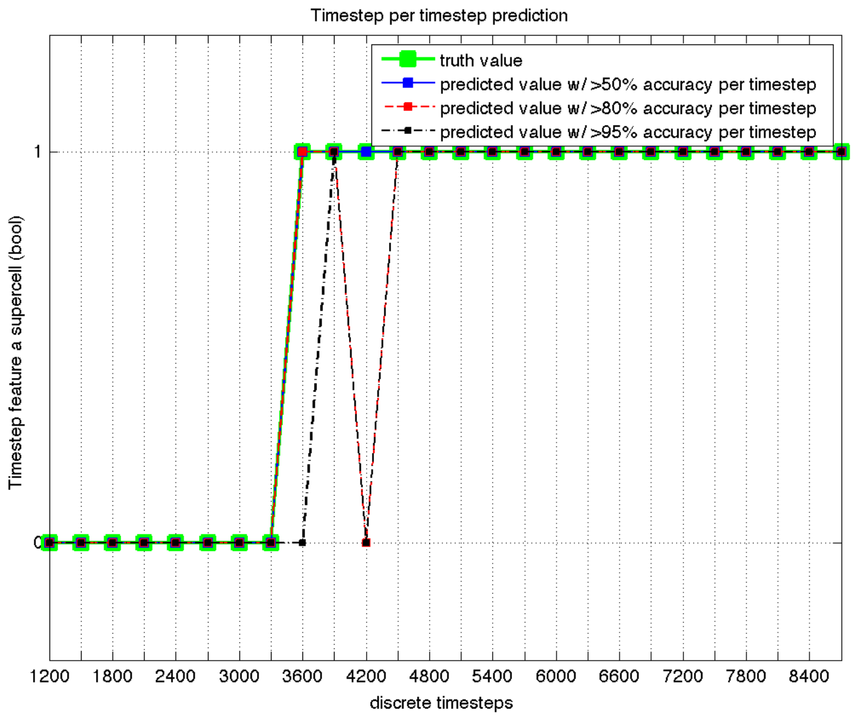
(a) Counting Matrix



(b) Timestep per timestep classification of the pixels



(c) Accuracy of the classification per timestep

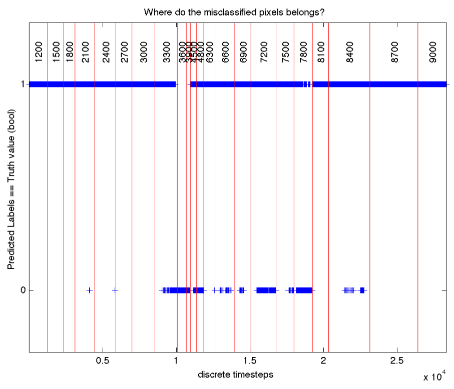


(d) Prediction on the evolution of the storm

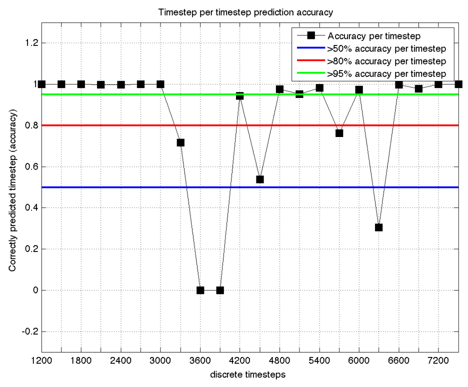
Figure 4.8: Results obtained for the SVM trained on storms 2-5 and tested on storm 1

		Classified as		Accuracy
		0	1	
Truth value	0	9666	2611	0.894
	1	399	15663	

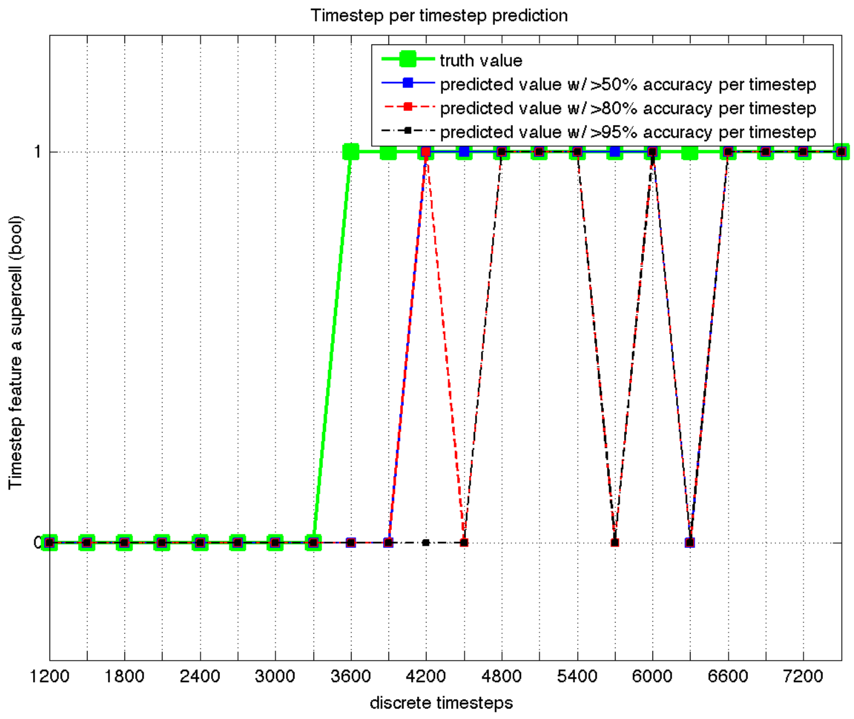
(a) Counting Matrix



(b) Timestep per timestep classification of the pixels



(c) Accuracy of the classification per timestep

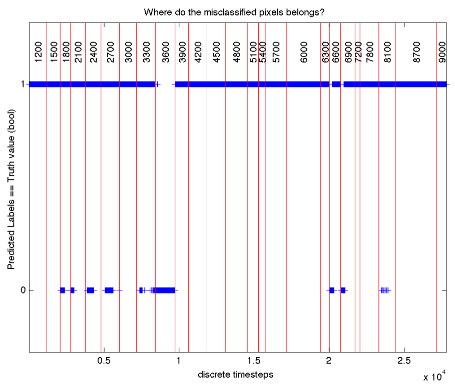


(d) Prediction on the evolution of the storm

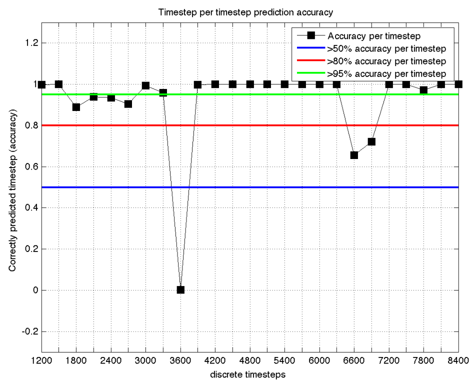
Figure 4.9: Results obtained for the SVM trained on storms 1,3-5 and tested on storm 2

		Classified as		Accuracy
		0	1	
Truth value	0	8076	4547	0.921
	1	1645	17536	

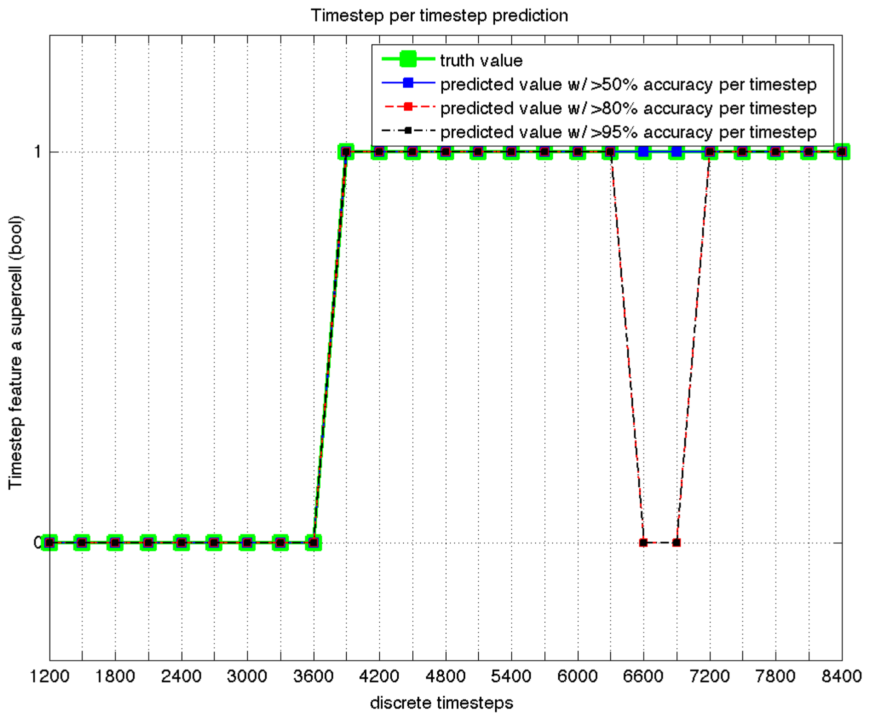
(a) Counting Matrix



(b) Timestep per timestep classification of the pixels



(c) Accuracy of the classification per timestep

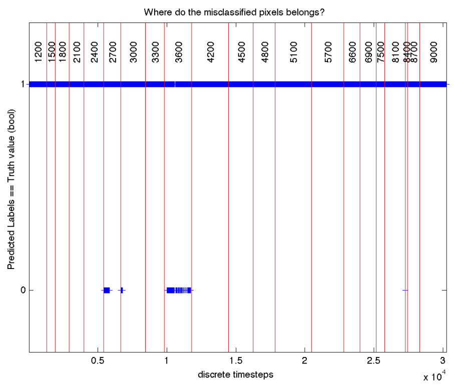


(d) Prediction on the evolution of the storm

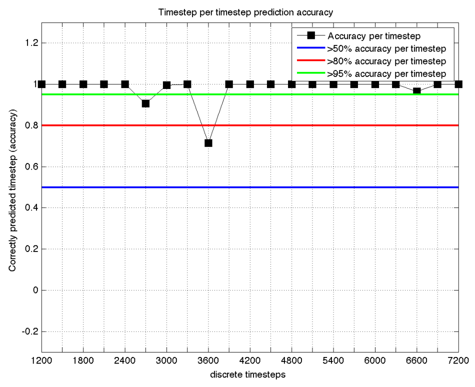
Figure 4.10: Results obtained for the SVM trained on storms 1,2,4,5 and tested on storm 3

		Classified as		Accuracy
		0	1	
Truth value	0	9685	553	0.978
	1	129	19881	

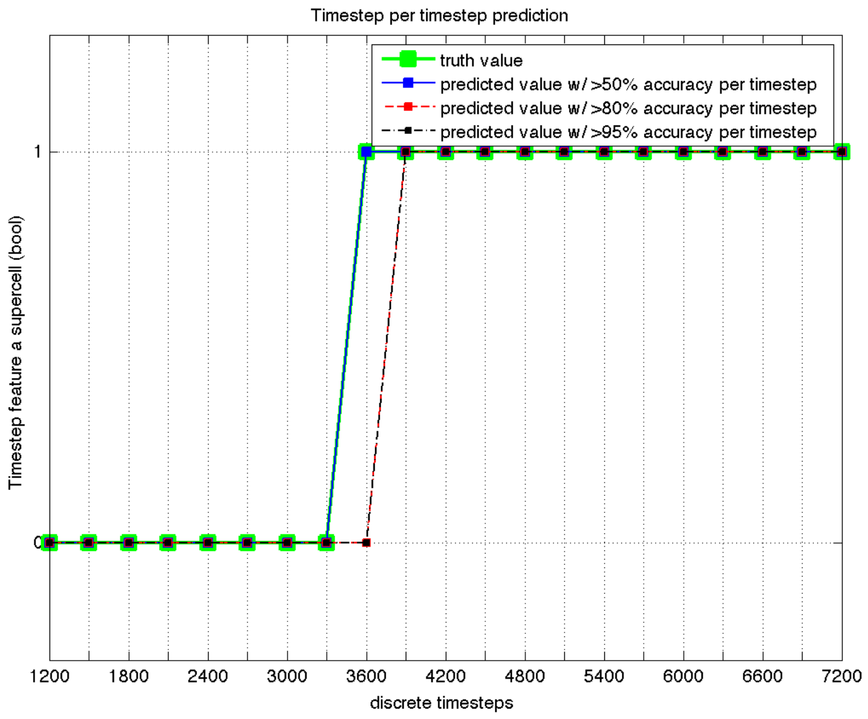
(a) Counting Matrix



(b) Timestep per timestep classification of the pixels



(c) Accuracy of the classification per timestep



(d) Prediction on the evolution of the storm

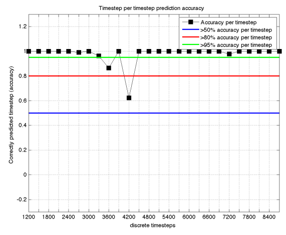
Figure 4.11: Results obtained for the SVM trained on storms 1-3,5 and tested on storm 4

We can compare how well our SVM classification performs to two other methods: Linear and Quadratic Discriminant Analysis (LDA and QDA). The results are presented in Figure 4.12. The quality of a ”timestep per timestep accuracy” curve depends on the number, width and depth of each downward peak. The deeper, the more misclassified are the pixels in the timestep. The wider, the more consecutive timesteps are likely to be misclassified. Therefore optimal results need few, small and thin curve peaks. LDA performs the worst as its peaks a generally the widest and deepest. QDA has less peaks but they are comparatively deeper than the SVMs. Therefore QDA has less misclassified timesteps than SVM, but those that are misclassified are by a larger margin than SVM. On the other hand, SVM has more peaks but the majority are small and do not affect the results when the timestep per timestep accuracy is set to a threshold around 70%-80%.

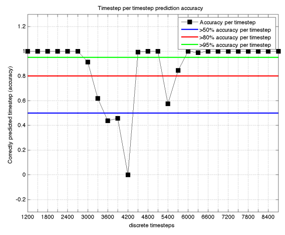
4.3 Study of the misclassifications

Some misclassified timesteps can make sense when the indecisiveness of the classifier is focused around the time of shift, when the thunderstorms becomes a supercell. This is the case seen in Figure 4.7 where timesteps 3600 and 3900 are considered thunderstorms instead of supercells. However, some misclassified timesteps are more unexpected: storm 3, timesteps 6600-6900 (Figure 4.10d)). To understand this phenomenon, let us focus on the original data associated with these faulty timesteps and compare them with other timesteps that have been correctly classified: Figure 4.13.

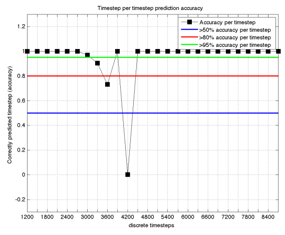
The reason for those failures could come from multiple possible sources:



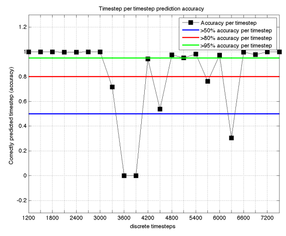
(a) SVM storm 1



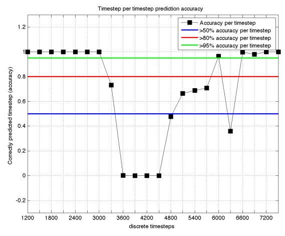
(b) LDA storm 1



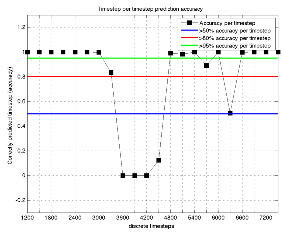
(c) QDA storm 1



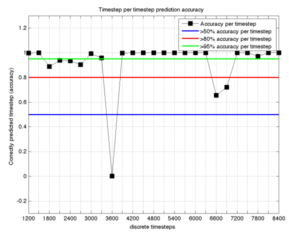
(d) SVM storm 2



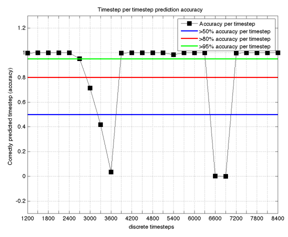
(e) LDA storm 2



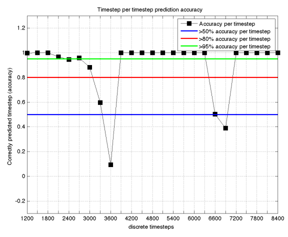
(f) QDA storm 2



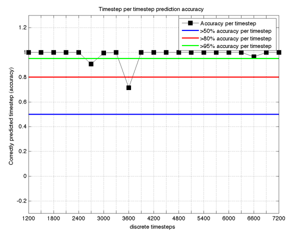
(g) SVM storm 3



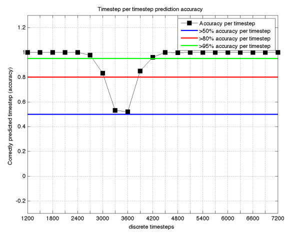
(h) LDA storm 3



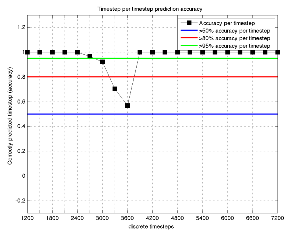
(i) QDA storm 3



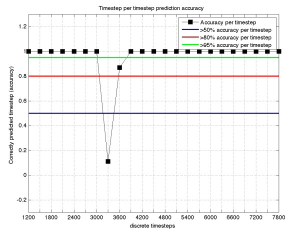
(j) SVM storm 4



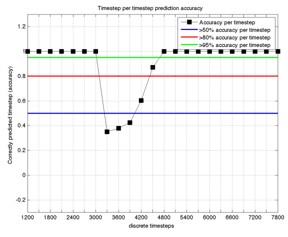
(k) LDA storm 4



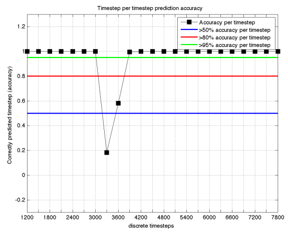
(l) QDA storm 4



(m) SVM storm 5

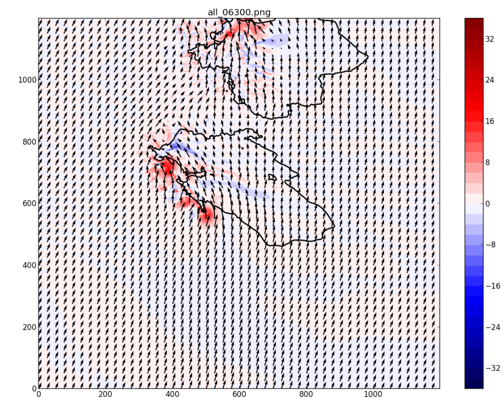


(n) LDA storm 5

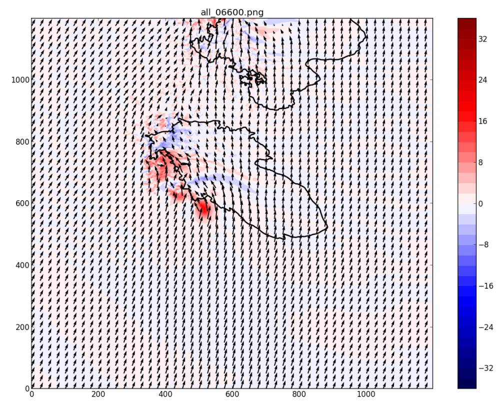


(o) QDA storm 5

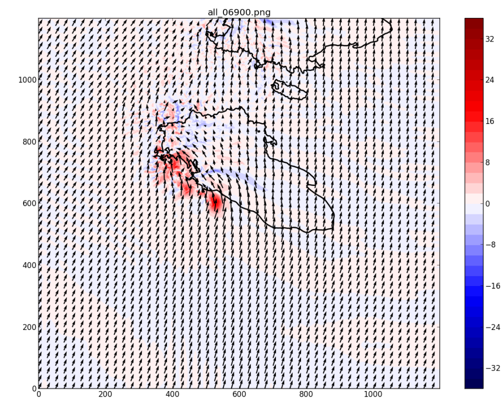
Figure 4.12: Comparison with other methods



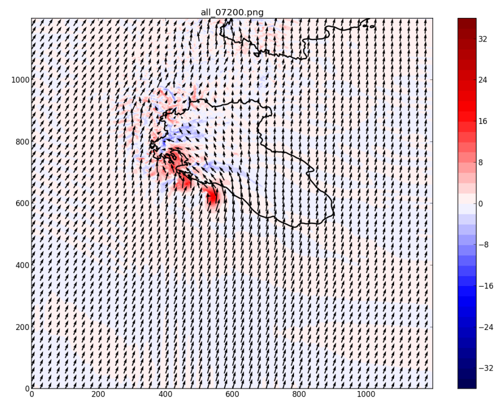
(a) Storm 3, timestep 6300



(b) Storm 3, timestep 6600



(c) Storm 3, timestep 6900



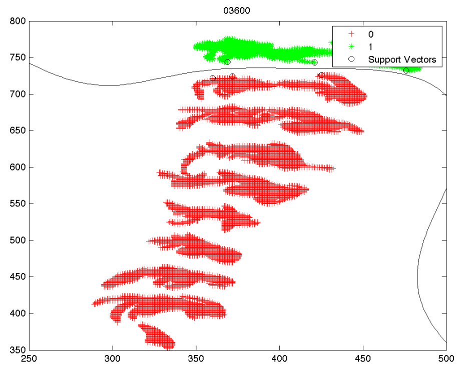
(d) Storm 3, timestep 7200

Figure 4.13: Comparison of correctly and incorrectly classified timesteps in real space data. The color scheme represents the magnitude of w, the vertical velocity field. The arrows represent the horizontal projection of the (u,v) fields and the contour lines represent the shape of the storm's reflectivity. Figures 4.13b and 4.13c have been misclassified whereas the previous and following timesteps have been correctly classified (4.13c and 4.13d)

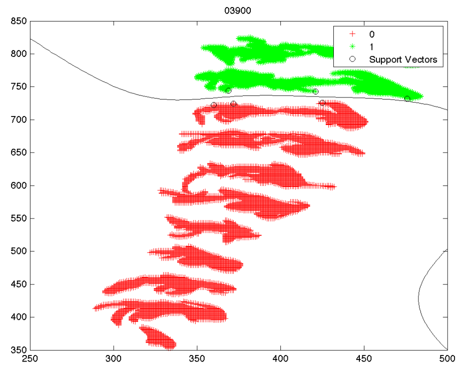
First, the watershed algorithm performs a local threshold on the gradient of the image. Since this input image is smoothed by the anisotropic filtering, some pixel values can on the borderline of being segmented in or out of the final mask. Those very same pixels are on the periphery of the segmented regions (since the watershed grows regions centered on the local maxima). Assuming the SVM's mapping function approximately conserves the relative position of the pixels when they are projected back and forth from the higher dimensions, those borderline pixels are more likely to be the support vectors for the SVM classifier. This slight variation can affect the percentage of pixels correctly classified per timestep and result in timesteps classifications.

Secondly, in the current stage of the project, the image registration process is the most unstable process in the pipeline because it is extremely dependent on a number of parameters. The results we have achieved are satisfying enough and greatly improve the performance of the classifier. However the end result is far from being optimal and could be the reason behind the shape of the SVM's projected data. Figure 4.14 is an example that follows the disposition of several timesteps when displayed on the SVM's projected plane. Although some timesteps are surprisingly placed (Figures 4.14c, 4.14f) for unknown reasons, we can see that each added timestep in the classification forms a new oblong shaped blob. Defining and tweaking more elaborate metrics, interpolators, transform types and optimizer parameters could potentially result in the merge of all those blobs into only 2 or 3 groupings of pixels, which would ease the minimization problem and potentially improve the accuracy of the SVM.

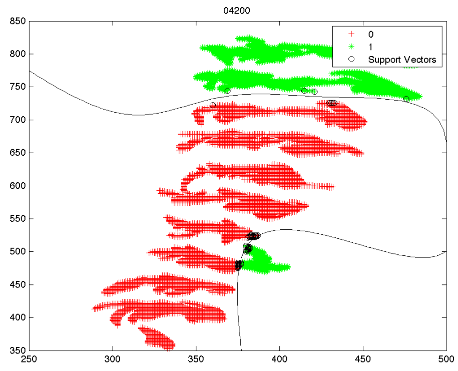
Additionally, knowing where each timestep is situated within the SVM's projection plane could prove useful to determine in advance the eventual time



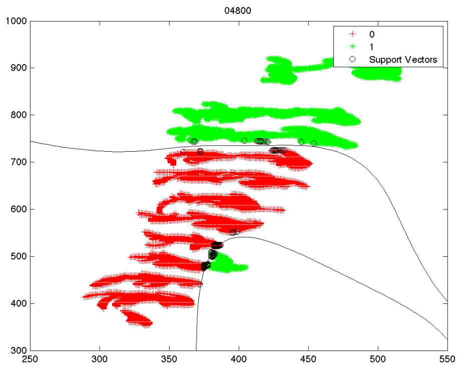
(a) timestep 3600



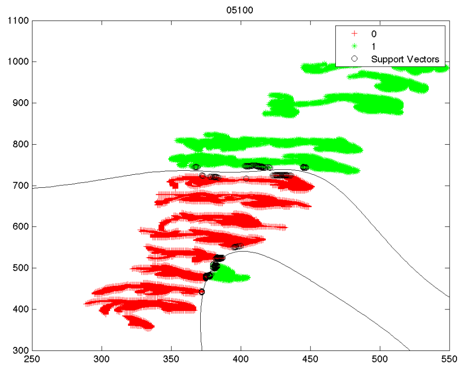
(b) timestep 3900



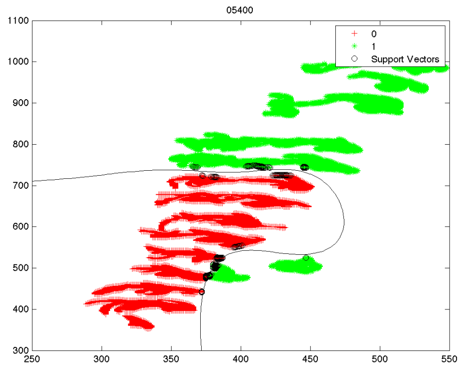
(c) timestep 4200



(d) timestep 4800



(e) timestep 5100



(f) timestep 5400

Figure 4.14: Progression of the storm when projected back from the higher dimensions using the SVM's mapping function. Each new blob represents a new timestep added to the training set of the SVM. The 0 and 1 labels are the truth values used to train the SVM. As more timesteps are added, the system converges to Figure 4.2a.

of formation of a supercell. By studying the evolution of the position of each timestep on the plane, and with a precomputed separation boundary between the supercell pixels and nonsupercell pixels, it might be possible to extrapolate the next positions of the timesteps blobs and possibly predict their label ahead of time. This is currently beyond the scope for this study.

Chapter 5

Conclusions and Future Expansion

In this study, we have devised a method to segment and classify vertical helicity fields in order to determine whether a thunderstorm at a given time features a supercell. The data was first smoothed, using a Perona-Malik anisotropic diffusion algorithm, then used as input to an enhanced 3D watershed algorithm that satisfies a basin width saliency criterion. The resulting segmented mask is registered to prevent the following classification process from depending too much on the location of the segmented features. Once registered, those features are then projected onto a horizontal 2D plane and fed to a Support Vector Machine whose parameters have been chosen by cross validation. The results are empirically satisfying, as most of the time, the classification of the timestep during which a thunderstorm evolves into a supercell is given accurately within a window of time spanning 5 to 10 minutes in the worst case scenario. It may happen that some timesteps at a later stage of a thunderstorm's life get misclassified but those misclassification could result from the vagaries of the initial thunderstorm's structure.

In order to improve future results, additional work can be performed on each of the various algorithms used in this study. Indeed, most of the misclassifications that happen after the shift from thunderstorm to supercell can be attributed to values of the gradient of the smoothed input image being on the borderline of being segmented in or out. Those very same pixels can become indecisive support vectors for the SVM classifier. This slight variation can affect the percentage of pixels correctly classified per timestep and result in timesteps classifications. In addition, image registration is extremely dependent on a number of parameters. The results we have achieved are satisfying enough and greatly improve the performance of the classifier. However defining and tweaking more elaborate metrics, interpolators, transform types and optimizer parameters could potentially result in the classification of two classes that are more accurately separated. This would also decrease the number of misclassified pixels per timestep and therefore, the number of misclassified timesteps overall. Finally additional study of the SVM's mapping function could prove useful not only to detect when a supercell forms but also to be able to predict in advance when the change is going to occur. Indeed, in the current state of the classifier, each timestep can be individually identified on the SVM. If a final kernel can be precomputed, calculating the percentage of pixels in the timestep that are support vectors can be an indication of how close to the separating timestep the current thunderstorm can be. This affirmation is purely theoretical at this stage of the study.

The method devised in this study satisfyingly serves its purpose and can be used to empirically identify a supercell whether from within a thunderstorm lifetime or as an independent weather phenomenon.

Reference List

Beucher, S., 2010: The watershed transformation page.
URL `http://cmm.ensmp.fr/~beucher/wtshed.html`

Beucher, S. and C. Lantuéj, 1979: workshop on image processing, real-time edge and motion detection. *Centre de Geostatistique et de Morphologie Mathematiques*.
URL `http://cmm.ensmp.fr/~beucher/publi/watershed.pdf`

Bunkers, M. J., M. R. Hjelmfelt, and P. L. Smith, 2006: An observational examination of long-lived supercells. part i: Characteristics, evolution, and demise. *Weather and Forecasting*, **21**, 673–688.

Cawley, G. C. and N. L. C. Talbot, 2007: Preventing over-fitting in model selection via bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, **8**, 841–861.

— 2010: Over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, **11**, 2079–2107.

Chisholm, A. J. and J. H. Renick, 1972: The kinematics of multicell and supercel alberta hailstorms. *Research Council of Alberta, Hail Studies Report*, **72-2**, 24–31.

Gonzalez, R. C. and P. Wintz, 1987: *Digital Image Processing*. Addison-Wesley.

Howley, T. and M. Madden, 2007: The evolution of a kernel-based distance metric fork-nn regression. *Proceedings of AICS-2007: 18th Irish Conference on Artificial Intelligence and Cognitive Science*.

Kain, J. S. and et al, 2008: Some practical considerations regarding horizontal resolution in the first generation of operational convection-allowing nwp. *Weather and Forecasting*, **23**, 931–951.

Lakshmanan, V., 2013: .
URL `http://www.cimms.ou.edu/~lakshman/autospatialgrids/outline/syllabus.htm`

Lakshmanan, V., K. Hondl, and R. Rabin, 2009: An efficient, general-purpose technique to identify storm cells in geospatial images. *Journal Atmospheric and Oceanic Technology*, **26**, 523–537.

Lemon, L. R. and C. A. Doswell III, 1979: Severe thunderstorm evolution and mesocyclone structure as related to tonadogenesis. *Monthly Weather Review*, **107**, 1184–1197.

Markowski, P. and Y. Richardson, 2010: *Mesoscale Meteorology in Midlatitudes*. Wiley-Blackwell.

Mei, C. and M. Dauphin, 2003: .
URL `http://rsbweb.nih.gov/ij/plugins/mixture-modeling.html`

Moller, A. R., C. A. Doswell III, and et al, 1994: The operational recognition of supercell thunderstorm environments and storm structures. *Weather and Forecasting*, **9**, 327–347.

Naylor, J. and et al, 2003: Comparison of objective supercell identification techniques using an idealized cloud model. *Montly Weather Review*, **140**, 2090–2102.

Nobuyuki, O., 1979: A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*.

Pal, N. R. and S. K. Pal, 1991: Image model, poisson distribution and object extraction. *J. Pattern Recognition Artific. Intell.*, **5 (3)**, 459–483.

Perona, P. and J. Malik, 1990: Space-scale and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**.

Pielke, R. and R. Carbone, 2002: Weather impacts, forecasts, and policy. *Bulletin of the American Meteorological Society*, **83**, 393–403.

Stensrud, D. J. and et al., 2009: Convective–scale warn–on–forecast system: A vision for 2020. *Bulletin of the American Meteorological Society*, **90**, 1487–1499.

Vapnik, V. and C. Cortes, 1995: Support-vector networks. *Machine Learning*, **20**, 273.

Wikipedia, 2013: Tornado alley.

Appendices

Appendix A

Other Notable Algorithms Used

Adaptive thresholding

The adaptive thresholding methods we are going to delve into are all histogram based. The procedure for each method is similar: First, the histogram of the grayscale data is calculated, and then operations are performed on the histogram. Those operations range from variance calculation to entropy calculations to gaussian fitting the histogram given certain parameters. The output in all cases is a scalar value which is the threshold value, used to binarize the image.

Histogram based methods

The underlying assumption is that the histogram is more or less bimodal and that each distribution represents either one of two classes: $\mathcal{C}_1$ and $\mathcal{C}_2$. The main idea of histogram based methods is to find the valley in the histogram that is the most representative of the difference between the two distribution $\mathcal{C}_1$ and $\mathcal{C}_2$. $\mathcal{C}_1$ and $\mathcal{C}_2$ are more commonly referred to as the foreground and the background.

The idea behind the thresholding is to find the regions of the data that have the highest values. The underlying assumption is that the hook echo is responsible for the highest values in helicity and the main updraft of the supercell is responsible for peaks in both helicity and vertical velocity. Hence, by appropriately selecting the highest values of the corresponding data, we will be able do locate the centers of those two regions of interest.

The main drawback to histogram based methods is that the results are very dependent on the number of bins that are chosen when creating the histogram.

For histograms computed from N elements, there are multiple approaches to choosing the number of bins:

1. The number of sample points: $\sqrt{N}$
2. The range of the data: $1 + \log_2 (max - min)$
3. The standard deviation of the data and the number of sample points: $\frac{3\sigma}{N^{1/3}}$

The first and third propositions generally do not work very well on spatial grids because N is too large: in the first proposition, N is used in the numerator which results in an overestimate of the number of bins. When there are too many bins in an histogram, most of them tend to contain only few pixels and are therefore not significant. The third proposition uses N in the denominator and thereby underestimates the number of bins. With N being very high it is frequent to have only a couple of bins, if more than one at all. The following methods assume that the various histograms have been calculated using the range of the data as number of bins.

Variance maximization thresholding

Presented by Otsu Nobuyuki in 1979 [Nobuyuki, 1979], the idea is to find a threshold that separates the histogram in two classes/bins, one above the threshold and one under, such that the intra-class variance is minimized. This problem is equivalent to finding a threshold that maximizes the interclass variance.

For a given 3D image, considering its histograms of *nbins* classes, we calculate the Cumulative Distribution Function (CDF) by calculating the cumulative histogram of that image and normalizing it by the sum of all pixels of the image.

For each value of pixel t, we name $\omega(t)$ the CDF value of t and $\mu_0(t)$ the class mean computed from the bins of the histogram situated to the left of the threshold t.

$$\mu_0(t) = \sum_{x<t} xp(x)$$

Similarly we name $\mu_1(t)$ the class mean computed from the bins of the histogram situated to the right of the threshold t.

$$\begin{aligned} \mu_1(t) &= \sum_{x<t} (x_{max} - x)(1 - p(x)) \\ &= \sum_{x>t} xp(x) \end{aligned}$$

Following Otsu's paper, the intra-class variance σ_W, mentioned as the within-class variance, that is to be minimized, is given by the formula:

$$\sigma_W = \omega\mu_0^2 + (1 - \omega)\mu_1^2$$

The inter-class variance σ_B, mentioned as the between-class variance, that is to be maximized, is given by:

$$\begin{aligned} \sigma_B &= \omega(\mu_0 - \mu_T)^2 + (1 - \omega)(\mu_1 - \mu_T)^2 \\ &= \omega(1 - \omega)(\mu_0 - \mu_1)^2 \end{aligned}$$

To maximize σ_B we loop through all the bins and the best threshold proposed by Otsu's method is the value of the maximum of the bin for which σ_B is the biggest.

Entropy minimization thresholding

In information theory, entropy is a measure of the uncertainty in a random variable. In this context, the term usually refers to the Shannon entropy, which quantifies the expected value of the information contained in a message. Shannon's entropy is defined as follow:

$$H(X) = E[-\ln P(X)] = -\sum_i P(x_i) \log_b (P(x_i))$$

where $H(X)$ is the entropy of a random variable $X = \{x_0 ... x_N\}$, E is the expected value operator and $P(x_i)$ is the probability mass function, or the probability that X is exactly equal to a certain value x_i.

The objective of this method is similar to Otsu's method, but uses class entropy instead of class variance.

Let us define:

- The probability of occurrence of a grayscale pixel x, in the k^{th} bin, under the threshold bin t

$$p_k^0(x) \quad = \quad \frac{p(x)}{\sum_{ix<t} p(x)}, k \leqslant t$$

- The probability of occurrence of a grayscale pixel x, in the k^{th} bin, over the threshold bin t

$$p_k^1(x) = \frac{p(x)}{\sum_{x>t} p(x)}, k > t$$

The gray-level histogram is often modeled as a mixture of normal distributions [Gonzalez and Wintz, 1987]. N. R. Pal and S. K. Pal have established that image histograms are more appropriately modeled by a mixture of poisson distributions [Pal and Pal, 1991]. In Pal and Pal's work, modeling of the gray-level histogram by a mixture of poisson distributions has been derived based on the theory of formation of image. Consequently we are going to use a Poisson model distribution to model the image information distribution (foreground distribution q^0 and background distribution q^1).

- The probability of a grayscale pixel x, in the k^{th} bin, under the threshold bin t to be in the foreground.

$$q_k^0(x) = \frac{\lambda_0(x)^k e^{-\lambda_0(x)}}{k!}, k \leqslant t$$

- The probability of a grayscale pixel x, in the k^{th} bin, over the threshold bin t to be in the foreground. It is also the probability of a grayscale pixel x, in the k^{th} bin, under the threshold bin t to be in the background.

$$q_k^1(x) = \frac{\lambda_1(x)^k e^{-\lambda_1(x)}}{k!}, k > t$$

- with the parameters

$$\begin{aligned}\lambda_0(x) &= \frac{\sum_{x<t} xp(x)}{\sum_{x<t} p(x)}\\ \lambda_1(x) &= \frac{\sum_{x>t} xp(x)}{\sum_{x>t} p(x)}\end{aligned}$$

The total cross entropy for a threshold t is calculated by:

$$H(t) = \sum_{k\leqslant t}(p_k^0 - q_k^0)\log_2(\frac{p_k^0}{q_k^0}) + \sum_{k>t}(p_k^1 - q_k^1)\log_2(\frac{p_k^1}{q_k^1})$$

That entropy is calculated for every value of t and the resulting threshold is the value of the maximum of the bin for which H is the smallest.

Mixture Model thresholding

The main idea of a Mixture model is to fit a function of the form

$$f(x|\lambda) = \sum_{i=0}^{N}\omega_i g(x|\delta_i)$$

where

- N is given, either user-chosen or automatically determined
- $\lambda = \{(\omega_0, \delta_0), ..., (\omega_N, \delta_N)\}$
- δ varies on the choice of the model g

The most commonly used Mixture model used is the Gaussian Mixture Model (GMM), where

- $\delta = \{(\mu_0, \sigma_0), ..., (\mu_N, \sigma_N)\}$

- $g(x|\mu,\sigma) = \mathcal{N}(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$
- μ is the mean of the distribution g
- σ is the standard deviation of the distribution g

For this thresholding, we assume the histogram is bimodal (foreground/background) and we apply a GMM algorithm with N=2 to it. Each of the two gaussians will be fit as best as possible to one of the histogram's class. Where the two curves meet represent the pixel value for which there is uncertainty. This particular pixel value is the threshold.

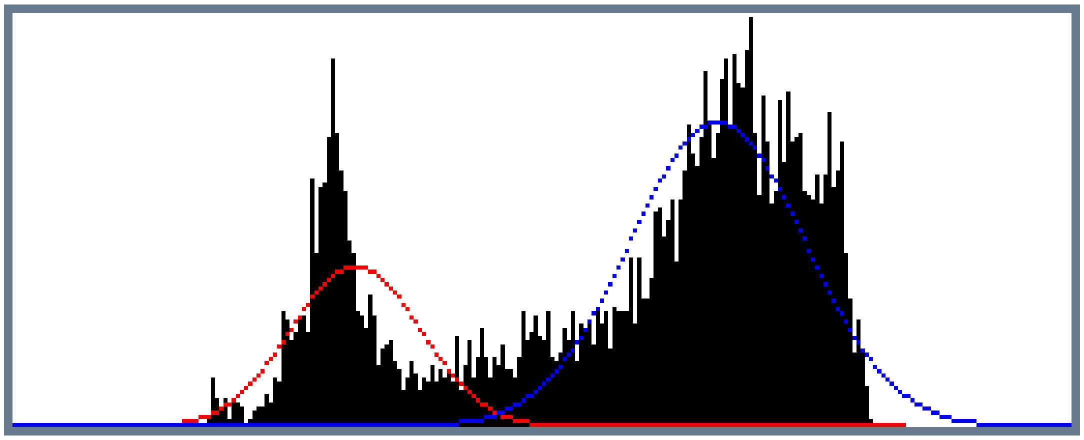


Figure A.1: Histogram Mixture Model [Mei and Dauphin, 2003]

To find the best fit for the mixture model, we use an Expectation Maximization (EM) algorithm:

Expectation Maximization This algorithm is an iterative process of two steps: the Expectation step (E-step) and the maximization step (M-step).

- **M-step**: The likelihood function that serves as evaluation function of the algorithm is maximized given the values of the parameters ω, μ and σ.

$$\theta_{i,j} = \theta_i(x_j) = \frac{\omega_i \mathcal{N}(x_j|\mu_i, \sigma_i)}{P(x_j)}$$

  with $P(x)$ given by the histogram

- **E-step**: the parameters of the gaussian ω, μ and σ are estimated assuming the new likelihood.

$$\begin{aligned} \mu_i &= \frac{\sum_j x_j \theta_{i,j}}{\sum_j \theta_{i,j}} \\ \sigma_i &= \sqrt{\frac{\sum_j \theta_{i,j}(\mu_i - x_j)^2}{\sum_j \theta_{i,j}}} \\ \omega_i &= \frac{\sum_j \theta_{i,j}}{N} \end{aligned}$$

When the algorithm converges, we obtain the parameter set λ_1, λ_2. The maximum uncertainty as to wether pixels belong to the background or not, is for the pixel value x_t which is at the intersection of the two gaussian curves and is therefore solution of

$$\omega_1 g(x_t|\delta_1) = \omega_2 g(x_t|\delta_2)$$

This is a quadratic function in x_t, therefore if x_t exists, it is either unique or has two theoretical values. If only one value is possible (it is unique or the other is lower than the image's minimum value or greater than the image's maximum value), then that value is the threshold. If no solution is found then the fitting function is inappropriate because, by the principle of the GMM-EM algorithm, the curves where chosen so that there would be a solution.

Discriminant Analysis Classifiers

Linear Discriminant Analysis (LDA)

Consider a set of observations $\vec{x}$ (also called features, attributes, variables or measurements) for each sample of an object or event with known class "y".

LDA approaches the problem by assuming that the conditional PDFs $p(\vec{x}|y = 0)$ and $p(\vec{x}|y = 1)$ are both normally distributed with mean and covariance parameters $(\vec{\mu}_0, \Sigma_{y=0})$ and $(\vec{\mu}_1, \Sigma_{y=1})$, respectively. Under this assumption, the Bayes optimal solution is to predict points as being from the second class if the log of the likelihood ratios is below some threshold T, so that;

$$(\vec{x} - \vec{\mu}_0)^T \Sigma_{y=0}^{-1} (\vec{x} - \vec{\mu}_0) + \ln|\Sigma_{y=0}| - (\vec{x} - \vec{\mu}_1)^T \Sigma_{y=1}^{-1} (\vec{x} - \vec{\mu}_1) - \ln|\Sigma_{y=1}| \ < \ T$$

Without any further assumptions, the resulting classifier is referred to as quadratic classifier, or quadratic discriminant classifier. LDA also makes the simplifying assumption that the class covariances are identical, so $\Sigma_{y=0} = \Sigma_{y=1} = \Sigma$) and that the covariances have full rank. In this case, several terms cancel and the above decision criterion becomes a threshold on the dot product.

$\vec{w} \cdot \vec{x} > c$

for some threshold constant "c", where

$\vec{w} \propto \Sigma^{-1}(\vec{\mu}_1 - \vec{\mu}_0)$

This means that the criterion of an input $\vec{x}$ being in a class "y" is purely a function of this linear combination of the known observations.

It is often useful to see this conclusion in geometrical terms: the criterion of an input $\vec{x}$ being in a class "y" is purely a function of projection of multidimensional-space point $\vec{x}$ onto vector $\vec{w}$ (thus, we only consider its direction). In other words, the observation belongs to "y" if corresponding $\vec{x}$ is located on a certain side of a hyperplane perpendicular to $\vec{w}$. The location of the plane is defined by the threshold c.

Quadratic Discriminant Analysis (QDA)

Statistical classification considers a set of vectors of observations x of an object or event, each of which has a known type y. This set is referred to as the training set. The problem is then to determine for a given new observation vector, what the best class should be. For a quadratic classifier, the correct solution is assumed to be quadratic in the measurements, so y will be decided based on

$$x^T Ax + b^T x + c$$

In the special case where each observation consists of two measurements, this means that the surfaces separating the classes will be conic sections (i.e. either a line, a circle or ellipse, a parabola or a hyperbola). In this sense we can state that a quadratic model is a generalization of the linear model, and its use is justified by the desire to extend the classifier's ability to represent more complex separating surfaces.

Quadratic discriminant analysis (QDA) is closely related to linear discriminant analysis (LDA), where it is assumed that the measurements from each class are normally distributed. Unlike LDA however, in QDA there is no assumption that the covariance of each of the classes is identical. When the normality assumption is true, the best possible test for the hypothesis that a given measurement is from a given class is a likelihood ratio test: Suppose there are only two groups, $y \in \{0, 1\}$, and the means of each class are defined to be $\mu_{y=0}, \mu_{y=1}$ and the covariances are defined as $\Sigma_{y=0}, \Sigma_{y=1}$. Then the likelihood ratio will be given by

$$Likelihoodratio = \frac{\sqrt{2\pi|\Sigma_{y=1}|}^{-1} \exp\left(-\frac{1}{2}(x-\mu_{y=1})^T \Sigma_{y=1}^{-1}(x-\mu_{y=1})\right)}{\sqrt{2\pi|\Sigma_{y=0}|}^{-1} \exp\left(-\frac{1}{2}(x-\mu_{y=0})^T \Sigma_{y=0}^{-1}(x-\mu_{y=0})\right)} < t$$

Appendix B

Tools

The following tools have been used during the course of this study. Their description is the official description taken from their respective official websites

MATLAB

"MATrix LABoratory refers to a programming language and a development environment. It is used for calculus matters. Developed by the company *The MathWorks*, MATLAB enables array and matrix manipulation, allows to plot single and multiple dimensional curves and data, allows to implement algorithms and can interface with others languages, notably C and C++. MATLAB is used in numerous domains including engineering, science, economy, as much in an industrial context as in a research context." *URL:* `http://www.mathworks.com/products/matlab/`

NetCDF

"NetCDF (network Common Data Form) is a set of interfaces for array-oriented data access and a freely distributed collection of data access libraries for C, Fortran, C++, Java, and other languages. The netCDF libraries support a machine-independent format for representing scientific data. Together, the interfaces, libraries, and format support the creation, access, and sharing of scientific data.

Advantages of NetCDF data includes, but is not restricted to:

- Self-Describing. A netCDF file includes information about the data it contains.
- Portable. A netCDF file can be accessed by computers with different ways of storing integers, characters, and floating-point numbers.
- Scalable. A small subset of a large dataset may be accessed efficiently.
- Appendable. Data may be appended to a properly structured netCDF file without copying the dataset or redefining its structure.
- Archivable. Access to all earlier forms of netCDF data will be supported by current and future versions of the software."

URL: `http://www.unidata.ucar.edu/software/netcdf/`

HDF5

"HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and is extensible, allowing applications to evolve in their use of HDF5. The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing data in the HDF5 format. HDF5 aspires to be a versatile data model that can represent very complex data objects and a wide variety of metadata, completely portable file format with no limit on the number or size of data objects in the collection." *URL:* `http://www.hdfgroup.org/HDF5/`

CImg

"The CImg Library is an open-source C++ toolkit for image processing. It mainly consists in a single header file providing a set of C++ classes and functions that can be used in your own sources, to load/save, manage/process and display generic images. The CImg Library is under CeCILL licensing. This is an open-source license which gives you rights to access, use, modify and redistribute the source code, under certains conditions." *URL:* `http://cimg.sourceforge.net`

ITK

"The Insight Segmentation and Registration Toolkit (ITK) is an open-source, cross-platform system that provides developers with an extensive suite of software tools for image analysis. Developed through extreme programming methodologies, ITK employs leading-edge algorithms for registering and segmenting multidimensional data." *URL:* `http://www.itk.org`

NICS

"The National Institute for Computational Sciences (NICS) at the University of Tennessee, Knoxville is one of the leading high performance computing centers for excellence in the United States." *URL:* `https://www.nics.tennessee.edu`