An Improved Storm Cell Identification and Tracking (SCIT) Algorithm based on DBSCAN Clustering and JPDA Tracking Methods

Jenny Matthews and John Trostel
Georgia Tech Research Institute, Severe Storm Research Center, Atlanta, GA

Abstract

Accurate storm cell identification and tracking is a major challenge in radar and severe weather operations, especially in the event of large scale storm systems that constantly change. A new method is presented which obtains higher accuracy of identification and differentiation of storm cells over the current Storm Cell and Identification Tracking (SCIT) algorithm. Sources of identification and tracking errors include the splitting and merging of existing storm cells, in addition to storm cells that may develop or dissipate over time.

The new cell identification method utilizes a density-based unsupervised clustering algorithm which requires no a priori knowledge of the number of existing cells and is not sensitive to reflectivity "dropouts". Furthermore, storm cells are identified and stored according to the entire area of the storm cell, which is contrary to the current method of maintaining just a centroid point. This information becomes very useful in applications that require association of storm cells with other meteorological phenomena such as tornadoes and lightning.

In addition to improved storm cell identification, a superior tracking and association algorithm is presented. As previously mentioned, storm cell areas are determined and tracked rather than centroid locations. A scheme of joint probabilistic data association (JPDA) problems is formed to associate storm cells. A traditional combinatorial optimization algorithm, the Hungarian Method, is performed on a particle representation of storm cells. This, in turn, produces a cost matrix which reflects the overall probability of assignment between two storms. Lastly, two iterations of an abbreviated Hungarian Algorithm, capable of making assignments that reflect splitting and merging cells, produce the final storm cell associations. Overall, storm cells are identified and tracked with a much higher degree of fidelity than the currently implemented SCIT algorithm.

1.0 Introduction

Numerous attempts have been made to properly identify and track storm cells over the years[1, 3, 4, 6, 7]. This problem is difficult and somewhat ill defined. The vague definition of a storm cell leaves much to be questioned. A storm cell can be defined as an air mass that contains up and down drafts in convective loops, moves and reacts as a single entity, and/or functions as the smallest unit of a storm producing system. However, storm cells often collide, move along the same front, or split. Due to the poor definition of storm cells, the novel tracking approach proposed in this paper does not look for individual storm cells that can be associated one-to-one temporally between time scans. It assumes that any given area of reflectivity identified at a point in time will most likely continue to exist at the next point in time. These smaller areas within storm cells are then temporally associated. The matches between these smaller areas are then

exploited to make the best conjecture as to which storm cells are temporally related. For the purpose of this paper, a storm cell shall be defined to have a unique reflectivity core above a given threshold that can be differentiated from other reflectivity cores near it, similar to the definition of a storm cell in radar meteorology.

1.0 Previous Work

Many similar algorithms have been proposed over the years to address the challenge of identifying and tracking storm cells. Among these include TITAN [1], ETITAN [3], and Johnson's SCIT algorithm [4]. Similar to the proposed algorithm, these methods locate regions of contiguous high reflectivity values and cluster them together to form storm cells. SCIT, TITAN, and ETITAN, attempt to decrease the dimensionality of the problem by first finding contiguous reflectivity values above a specific threshold along one-dimensional segments, grouping those segments to form two-dimensional components, and then grouping these two-dimensional components to identify storm cells. Storm cells are then stored and identified by a centroid position [4] or an ellipse [1,3] with a shape, size, and orientation similar to that of the identified storm.

Numerous attempts at tracking storm cells have been made in the past couple decades as well. These algorithms include matching storm cell's centroid position with a simple least squares estimate of the storm cell's expected position [4]. They also include matching storms based on a storm's overlapping are with its expected coverage area [9]. Another approach has been to treat the tracking and association problem as a constrained global optimization problem based on both storm cell position and size [1, 3].

This paper discusses an algorithm to identify and track storm cells and their morphology over time. It employs a sophisticated method of identifying and storing storm cells by location, size, and exact shape. It fashions an advanced examination of storm cell organization and exploits this information in an innovative tracking algorithm.

2.0 The Identification Algorithm

The proposed algorithm processes volumetric reflectivity data from the NEXRAD weather radar with the purpose of identifying and tracking storm cells. NEXRAD base reflectivity data is preprocessed using the WDSS-II quality control neural network to reduce spurious echoes from anomalous propagation and non-meteorological targets such as insects, birds, and ground clutter [13]. Prior to tracking, storm cell identification is performed on the quality controlled reflectivity in two key phases. The first phase entails identification of two-dimensional components or clusters at each elevation slice of a volume scan. The second phase involves vertical association of the two-dimensional components to construct a storm cell. The steps of first identifying two-dimensional components and then vertically associating these components are similar to the identification approach of the SCIT algorithm of Johnson [4] and the TITAN algorithm of Dixon and Weiner [1]. This approach is usually favored for its reduction in dimensionality, thus increasing computational efficiency. The algorithm described

here, however, employs different methods in order to identify two-dimensional components and associate them.

2.1 Two-Dimensional Component Identification Using DBSCAN

Identification of two-dimensional storm components is performed using a type of Density Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [2]. The DBSCAN algorithm is a basic density based clustering algorithm based on the following definitions:

1. Suppose there is a set *P* of objects *p* that we desire to divide into clusters, defined as

$$P = \{p \in \mathbb{R}^N\}$$

2. Let the *ε-neighborhood* of an object *p*, denoted by $N_\varepsilon(p)$, be defined as

$$N_\varepsilon(p) = \{q \in N_\varepsilon(p) | dist(p,q) \leq \varepsilon\}.$$

   That is, it includes all points *q* within a radius ε of object *p*.

3. A *core object* is an object *p* with respect to the parameters ε and *MinPts* if

$$|N_\varepsilon(p)| \geq MinPts$$

   That is, to be defined as a core object, an object *p*, must contain at least *MinPts* objects within its *ε*-neighborhood.

4. An object *q* is *directly density reachable* from *p* if

   a. $q \in N_\varepsilon(p)$
   b. $|N_\varepsilon(p)| \geq MinPts$ (*p* is a core object)

5. An object *q* is *density reachable* from *p* if there is a chain of directly density reachable objects connecting *p* and *q*.

6. An object *q* is *density connected* to a point *p* with respect to the parameters *ε* and *MinPts* if there is a point *o* such that both *p* and *q* are density reachable from *o*.

7. A *cluster* with respect to the parameters *ε* and *MinPts* is a non-empty subset *C* satisfying the following conditions

   a. If $p \in C$ and *q* is density reachable from *p* with respect to the parameters *ε* and *MinPts*, then $q \in C$.
   b. If $p, q \in C$, then *p* is density connected to *q* with respect to the parameters *ε* and *MinPts*.

8. *Noise* is defined as an object that does not belong to any cluster.

Some difficulties with the DBSCAN method include computational complexity, sensitivity to user input, and the inability to find clusters of varying densities. These effects are a problem in many applications that require processing of random sets of data; however, these effects are minimized when operating on a regular grid of data. Multiple grid-based DBSCAN methods such as GriDBSCAN [8] and GRIDBSCAN [12] have been suggested to overcome the computational complexity issue. The regular polar grid of the NEXRAD data prevents this from being an issue. Moreover, the definition of the ε-neighborhood may be defined in terms of grid spacing instead of the standard Euclidean distance metric to further simplify the computational complexity of DBSCAN. The problems of sensitivity to user input and inability to find clusters of varying densities is addressed in this work by using various reflectivity thresholds to create two-dimensional components, analogous to the SCIT algorithm of Johnson [4].

For the purpose of this algorithm, the ε-neighborhood of an object is defined in terms of the radar's polar grid. It extends one range cell and one azimuth cell on either side of the object. As illustrated in Fig. 1, a grid point has four other grid points in its ε-neighborhood. Each grid point may or may not be an object. Given a specific reflectivity threshold, any grid point that has a reflectivity above that threshold is considered to be an object. Any grid point that has a reflectivity below the given threshold is not considered to be an existing object. As seen in Fig. 1, a grid point may have as many as four objects in its ε-neighborhood. The numerical value *MinPts* shall be set to three. Thus, in order to be a core object, a grid point must be an existing object and have at least three neighboring objects. In other words, the grid point of interest must be equal to or above the given threshold and at least three of the four neighboring grid points must meet or exceed the given reflectivity threshold.

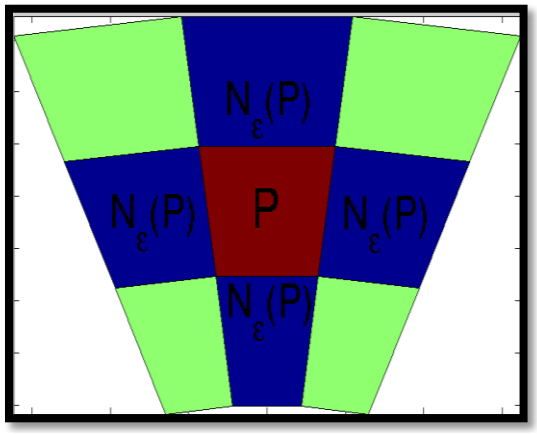


Fig. 1. The valid ε-neighborhood of a point in the DBSCAN algorithm as applied is the neighboring grid points in azimuth and range of the data point of interest.

There are several aspects of DBSCAN that make it an advantageous method in two-dimensional component creation over other proposed methods. Unlike partitioning methods such as K-means, density-based clustering algorithms require no *a priori* knowledge of the number of clusters to be formed. These other partitioning methods also always produce convex shapes, whereas density-based methods work better with arbitrary shapes [8]. Another dilemma with K-means is that the initial choice of cluster centroids is random and can affect the ending

result. While a method involving texture based image segmentation and hierarchal K-means has been suggested to remedy these problems [6], the hierarchal methods still require a termination criterion which can be difficult to determine. These algorithms can also be very computationally expensive [8].

As previously mentioned, other methods of creating two-dimensional components have been implemented in SCIT [4], TITAN [1], and ETITAN [3]. The SCIT method requires processing data on a radial by radial basis, creating one-dimensional storm segments composed of contiguous grid points with a reflectivity value above a given threshold. Storm segments in adjacent radials that overlap in range are then combined to form two-dimensional components. TITAN and ETITAN perform a similar procedure in Cartesian coordinates instead of polar coordinates. This process is repeated for one (TITAN), two (ETITAN) or up to seven (SCIT) reflectivity thresholds. The one-dimensional segment creation creates sensitivity to variation in the reflectivity fields. If one point along the radial drops below the threshold amidst the reflectivity field of a storm, the segment would be prematurely terminated. To remedy this, a set of *dropout* parameters are defined for the SCIT and TITAN algorithms. In creating a segment of contiguous reflectivity points above a given threshold, if a point is encountered whose value is below the desired threshold, but is still greater than a secondary *dropout threshold*, the segment may continue to be extended, as illustrated in Fig. 2. Another parameter, *the dropout count*, defines the maximum number of contiguous dropout points allowed before a segment is terminated. These parameters are subjective and can be altered at the users' discretion. Furthermore, a minimum segment length threshold is defined. Any segment that is shorter than this threshold is not saved. The proposed DBSCAN method finds contiguous points across both dimensions eliminating the need for both the dropout threshold and count parameters. If a dropout in the reflectivity field of a storm does occur, the DBSCAN method will simply "work around" the hole.

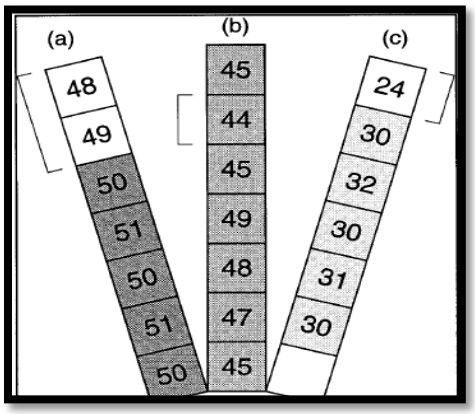


Fig. 2. For the SCIT algorithm, with a dropout reflectivity of 5 dBz below the search threshold and a maximum dropout count of two allowed, the shaded areas represent the saved portions of the following storms segments (braces indicate dropouts or termination values): (a) 50-dBz storm segment followed by two dropouts, thus terminating the segment (b) 45-dBz storm segment with one dropout, (c) 30-dBz storm segment followed by a reflectivity value less than the dropout value (termination value).

Note: Reprinted from "The Storm Cell Identification and Tracking Algorthm: An Enhanced WSR-8D Algorithm" by J.T. Johnson, 1998, Weather and Forecasting Vol. 13, pp. 263-276.

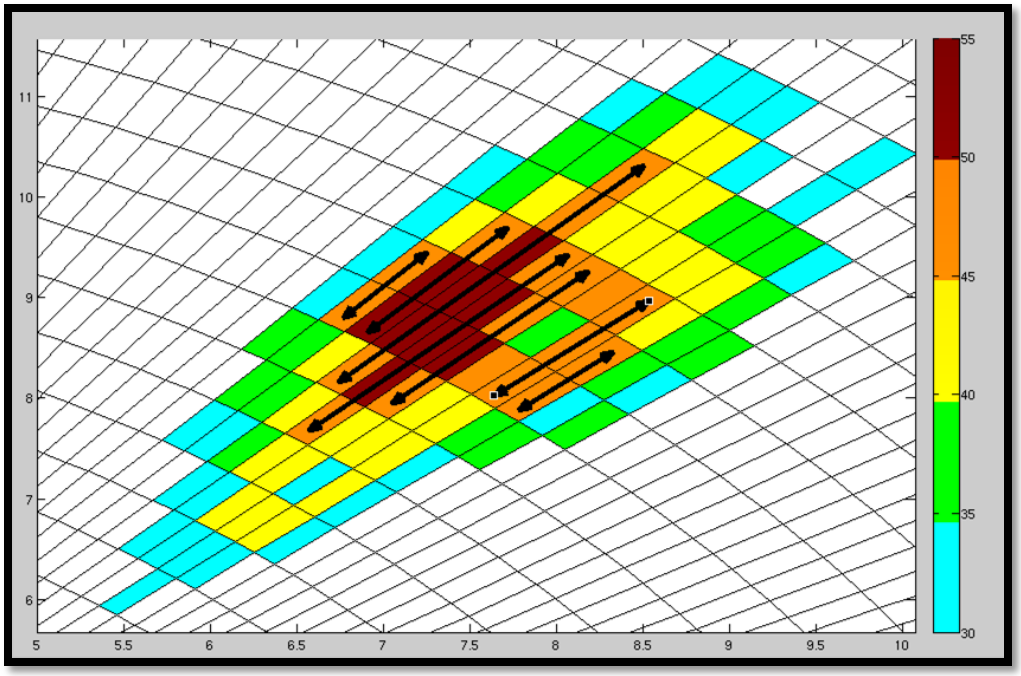


Fig. 3. A reflectivity intensity map shown above displays segments created for a 45 dBz threshold. This scenario shows no segment created for a single radial in the storm. The result is identification of two separate storms when there may only be one storm.

Fig. 3 shows an example where SCIT and TITAN storm segment creation is unsuccessful at identifying a storm component properly due to the one-dimensional search method. In the figure, there is a single radial where a one-dimensional segment is not created due to a low reflectivity point in the center of the radial. The proposed DBSCAN method would identify the above-threshold points as non-core points. As a result two different components would still be created. It should be noted, however, that both components would share these non-core points and would be overlapping. Furthermore, a different method of vertical association is applied than in the original SCIT algorithm that will allow for the possibility of these components to be part of one storm cell or two separate storm cells. In comparison, the original SCIT algorithm does not have this type of flexibility in the vertical association of storm cells. In this case, misidentification of the number of actual components at any given elevation would have a detrimental effect on the identification process.

Like the SCIT algorithm, the proposed method creates two-dimensional components for as many as seven different reflectivity thresholds (30, 35, 40, 45, 50, 55, 60 dBz). As seen in Fig. 4, any higher reflectivity component would be an interior subset of a lower reflectivity threshold component. This means, for example, that a 60 dBz threshold would be located in a 55 dBz component of a larger or equal size, and the 55 dBz component would be located inside a 50 dBz component of larger or equal size, and so on. In the SCIT algorithm, the inner most reflectivity component is saved while all lower reflectivity components that contain the highest reflectivity core component are discarded, as seen in Fig. 4b. Unlike the SCIT algorithm, the largest and lowest reflectivity core component is saved while all lower reflectivity components containing multiple core components are discarded, as in Fig. 4a. As a result, the inner most storm cells are saved while being represented by the largest possible area.

Another advantage of the proposed method is that it stores the two-dimensional components as entire polygonal regions whereas in the SCIT algorithm, the locations of two-dimensional components are saved simply as weighted centroid points. The additional information of the entire polygonal area of the storm will be used for more accurate vertical association in the second phase of the storm cell identification process. Moreover, since storms are most frequently shaped as something other than a perfect circle, having an accurate representation of the storm area is important for applications that require associating storms with meteorological phenomena such as lightning and tornadoes. Current methods associate phenomena based on the Euclidean distance from the weighted centroids of the storms. This can be problematic with closely spaced storms or storms that are oddly shaped, resulting in inaccurate association of meteorological phenomena and storm cells. Defining each storm by its actual shape means that meteorological phenomena will be more accurately associated.

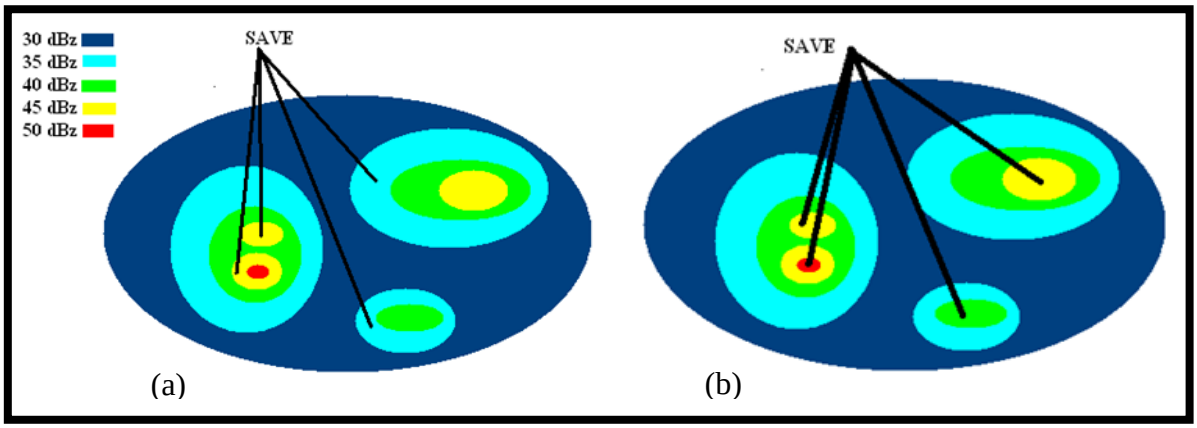


Fig. 4. (a) The proposed algorithm saves the lowest reflectivity region that identifies the same core objects. (b) The SCIT algorithm saves a centroid representing the smallest and highest reflectivity core object in the reflectivity field of a storm.

2.2 Vertical Association of Two-Dimensional Components

Once two-dimensional components have been formed via DBSCAN, they are vertically associated based on overlapping area, $A_{O,}$ as seen in Fig. 5. Components in adjacent elevation angles are vertically associated when their polygonal regions overlap from a plan view. Unlike the SCIT algorithm, multiple components at each elevation slice may belong to a single storm cell as in Fig 5. Also, each component can belong to multiple storm cells. This is a more realistic interpretation of the varying shapes of storm cells. This method of vertical association also provides more accurate association for oddly shaped storms, specifically non-circular, vertically tilted, and closely spaced storms.

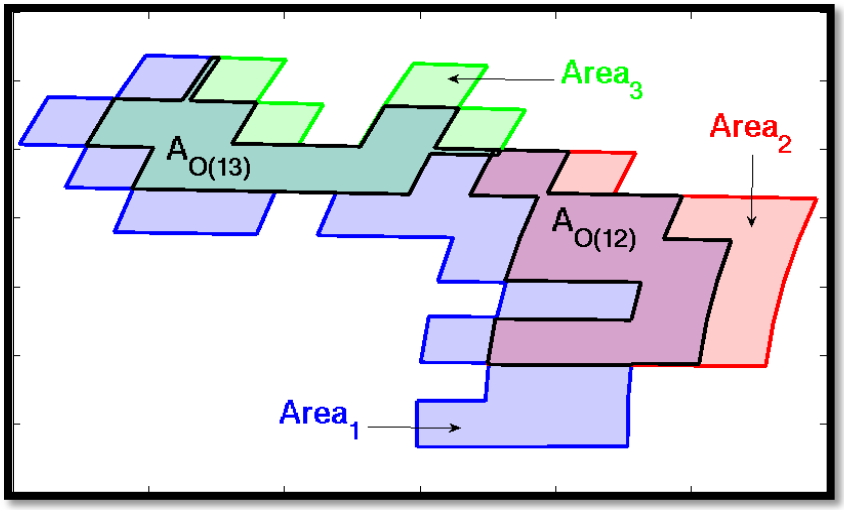


Fig. 5. A component $Area_1$ (blue) at the lowest elevation angle overlaps with two components, $Area_2$ (red) and $Area_3$ (green), at the next highest elevation slice. Therefore, $Area_1$ is associated with both higher elevation components.

The vertical association process is performed starting with the highest elevation angle, associating each component at elevation *e* with one component at elevation *e-1*. Note that while components at the higher elevation may only be associated with one component at the next lower elevation, the opposite is not true. In other words, components at elevation *e-1* may be associated with multiple components at elevation *e*. This method of vertical association is adapted due to the nature of storm cell reflectivity fields. In the past, other algorithms such as SCIT and TITAN have performed vertical association from the lowest elevation to the highest elevation. However, as can be seen in Fig. 6, taken from actual NEXRAD data, the two-dimensional components which distinguish closely spaced storm cells tend to occur at higher elevations, while the reflectivity fields at lower elevations tend to blend together.

The vertical association process is performed in a three step process as follows:

1. For each elevation slice, starting with the highest, $e = N_E$, each component at *e* is associated with the component at *e-1* that overlaps by the largest area. For example, as illustrated with the components in Fig. 7, cells would be created as follows:

    a. Cell 1: A8→A7→A6→A5→A4→A3→A2→A1
    b. Cell 2: D7→B6→C5→D4→A3→A2→A1
    c. Cell 3: C6→D5→E4→C3→A2→A1
    d. Cell 4: E5→F4→D3→B2→A1

2. Suppose a component at an elevation slice e has been associated with a component at *e-1* that already belongs to one or more existing cells. If and only if the component was not associated with a component at elevation *e+1*, it is set aside temporarily.

    a. Component B7 is associated with A6, but A6 already belongs to Cell 1.
    b. Component C7 is associated with A6, but A6 already belongs to Cell 1.
    c. Component E7 is associated with B6, but B6 already belongs to Cell 2.
    d. Component B5 is associated with A4, but A4 already belongs to Cell 1.

e. Component B4 is associated with A3, but A3 already belongs to Cells 1 and 2.
f. Component C4 is associated with A3, but A3 already belongs to Cells 1 and 2.
g. Component B3 is associated with A2, but A2 already belongs to Cells 1, 2, and 3.

3. The components that are set aside are treated as follows. If the component can be added to one and only one cell, it is added. If it corresponds to more than one cell, it is discarded. This reflects the fact that these multiple correspondences for a single cell are simply an artifact of multiple closely spaced cells and not a distinguishing component of any single cell. This yields the flowing results in the given scenario:

   a. Component B7 is added to Cell 1.
   b. Component C7 is added to Cell 1.
   c. Component E7 is added to Cell 2.
   d. Component B5 is added to Cell 1.
   e. Component B4 is discarded since it could be associated with both Cells 1 and 2.
   f. Component C4 is discarded since it could be associated with both Cells 1 and 2.
   g. Component B3 is discarded since it could be associated with Cells 1, 2, and 3.

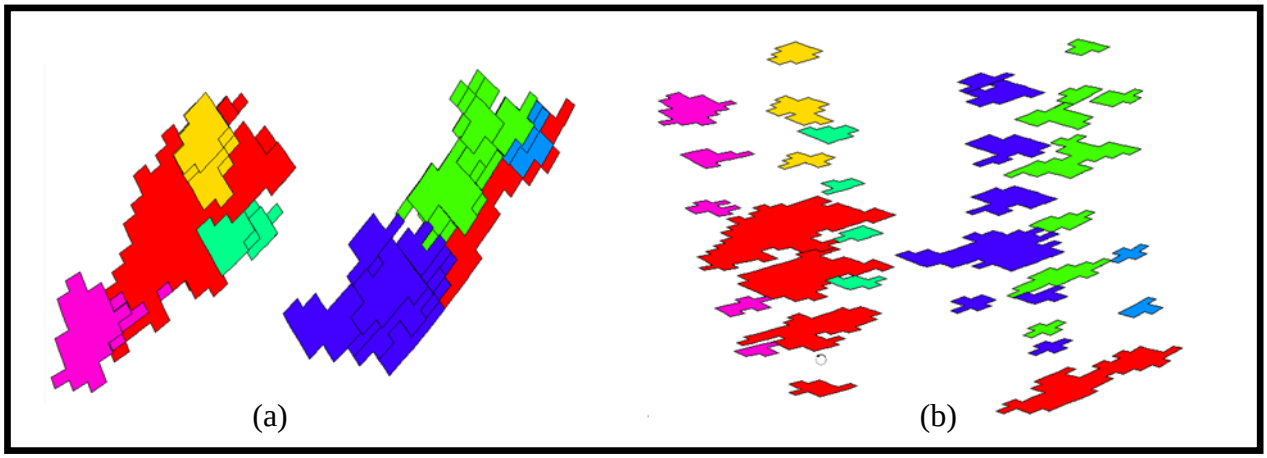


Fig. 6. (a) Plan view of closely spaced storm cells whose reflectivity fields have blended/merged at the lower elevation fields. (b) Example of the same closely spaced storm cells from a side view. In these figures, each component portrayed as the color red has been associated to multiple storm cells. Each higher component that has been identified uniquely with a single storm cell has been assigned a different color per that storm cell.

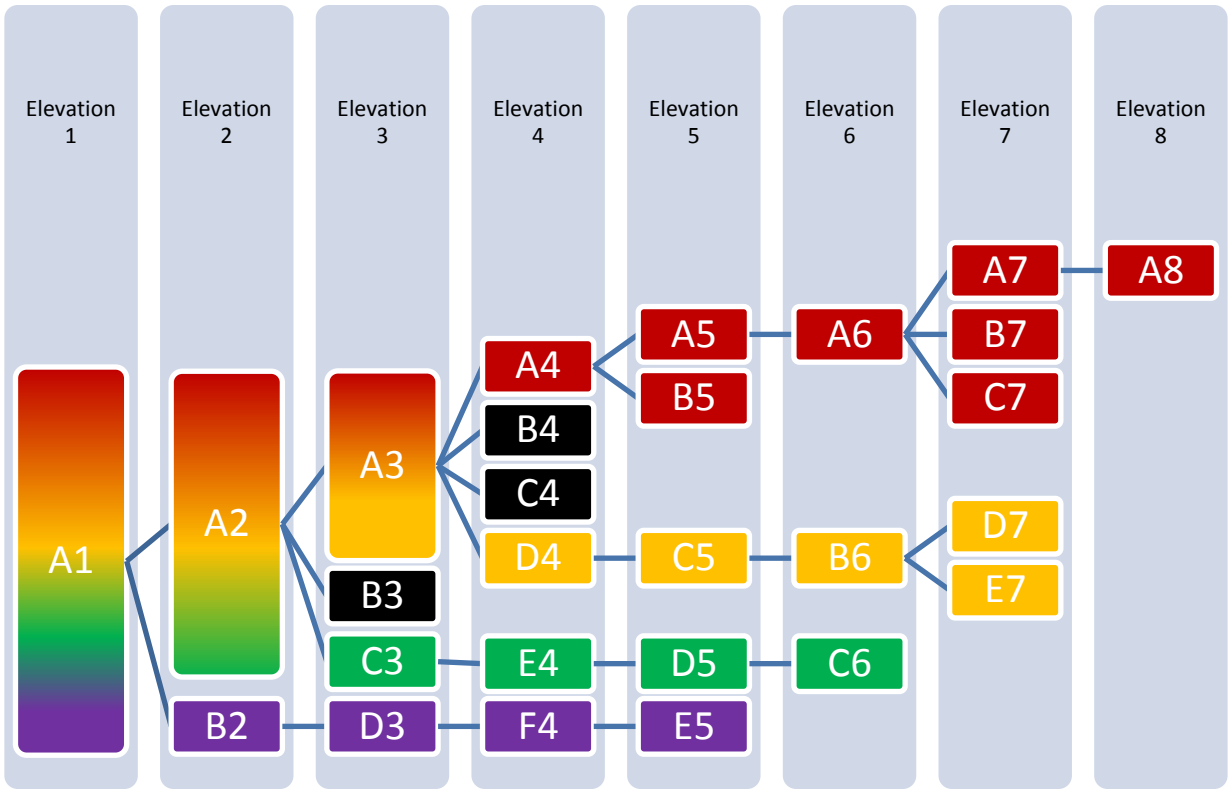


Fig. 7. Example of vertical association of two-dimensional components to define three-dimensional storm cells. The black components are discarded. The red components belong strictly to Cell1. The yellow components belong strictly to Cell 2. The green components belong strictly to Cell 3. The purple components belong strictly to Cell 4. The multicolored components belong to multiple cells.

Once vertical association is complete, storm cell locations are represented by polygonal regions which are a plan view union of all the two-dimensional component polygons that form the three dimensional storm cells *and are unique to those storm cells*. This means, in Fig. 7, Cell 1 would be represented by a union of components A4, A5, B5, A6, A7, B7, C7, and A8; Cell 2 would be represented by components D4, C5, B6, D7, and E7; Cell3 would be represented by components C3, E4, D5, and C6; and Cell 4 would be represented by components B2, D3, F4, and E5. These polygons are then dilated to cover the area of the non-unique components associated with them. Once the polygonal regions of the storm cells have been defined, a uniform particle grid representation is created for each storm, as illustrated in Fig. 8, for future use in the tracking algorithm. The uniform grid has spacing Δ_p for all storm cells in both x and y directions, making the number of particles in each cell a function of area.

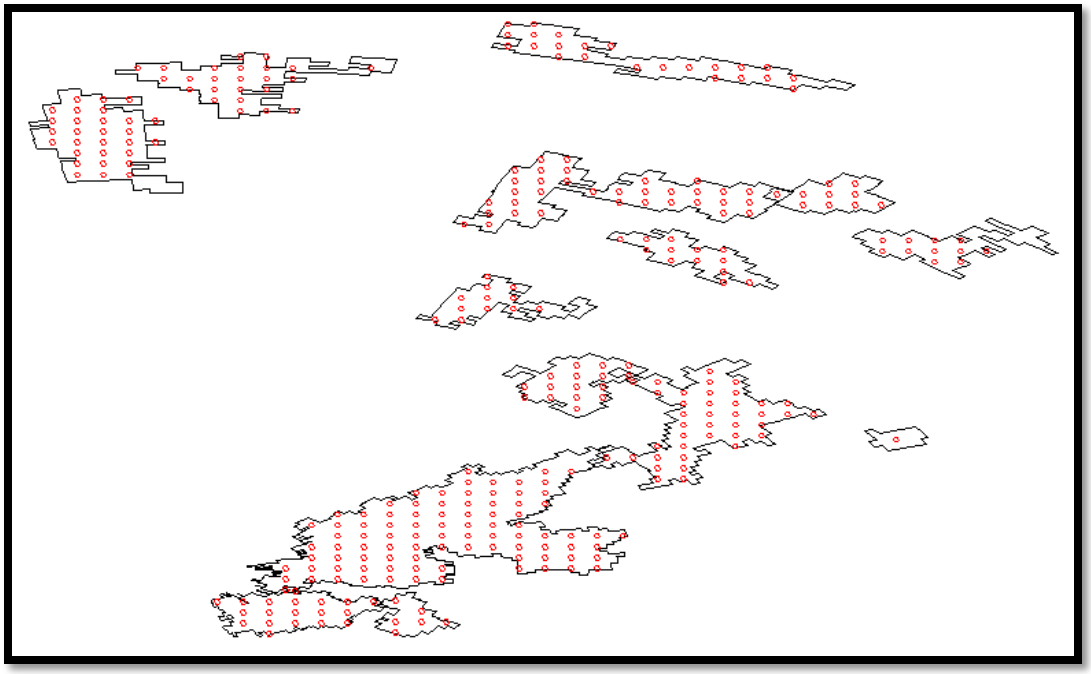


Fig. 8. Example of particle representation of storms to be used in tracking algorithm.

3.0 The Tracking and Association Algorithm

The proposed tracking and association algorithm forms and solves a series of joint probabilistic data association (JPDA) problems to temporally associate storm cells, while allowing for the possibility of splits and mergers. The classic JPDA formulization would create a matrix of costs for associating each pair of storm cells at consecutive time steps based on the difference in location and/or other characteristics of those storm cells. It would assume a one-to-one assignment between storm cells at two consecutive time steps. The one-to-one assignment of typical JDPA methods are far from optimal due not only to splits and mergers that may be occurring, but also due to the rapid evolutionary nature of storm cells and relatively large time periods between complete scans.

3.1 Initialization of Tracking and Association Algorithm

The new algorithm is initialized by entering one or more velocity estimates for different locations in the radar area. These estimates are entered as:

$$\bar{V}(l) = \left[V_x(l)\ V_y(l)\right] at\ \bar{P}(l) = \left[P_x(l)\ P_y(l)\right] for\ l = 1,2,\ldots|\bar{P}(l)| \qquad (1)$$

where $\bar{P}(l)$ is the location of the l^{th} velocity estimate (in Cartesian coordinates, with the radar at the origin) and $\bar{V}(l)$ is the l^{th} velocity estimate. $|\bar{P}(l)|$ is the cardinality of the set of input estimates. Velocity estimates for all storm cells within the convex hull of $\bar{P}(l)$ for $l = 1,2,\ldots L$ are then calculated via a linear two-dimensional interpolation of these given point estimates.

Any storm cells not within the convex hull of these point locations are assigned the same velocity as the nearest point at which an estimated velocity was given. The source for these estimates is currently a user estimate based on a visualization of storm cells plotted on the same map for times *t* and *t+1*, as seen in Fig. 9.

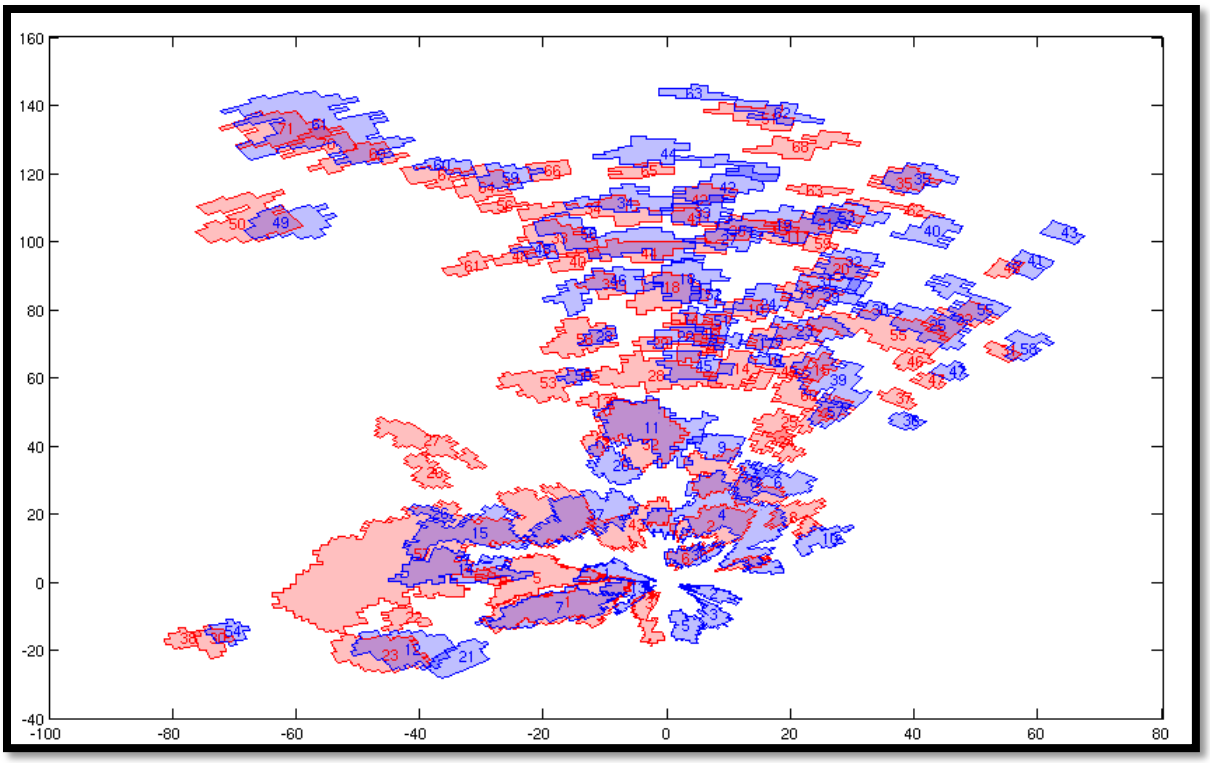


Fig. 9. Example of the display for a user to estimate velocities at multiple locations based on the perceived movement of storm cells between volume scans. Storm cells detected at time t are displayed in red and storm cells detected at time t+1 are displayed in blue. The map coordinates are in kilometers relative to the radar location.

3.2 Velocity Estimation for a Simple Particle

For any time $t>t_0$, assume an estimate for the velocity of each existing storm cell is given. These velocities are then assigned to each particle of the storm cell, that is all particles in a given storm cell are assumed to have equal velocities:

$$V_t^P(i,j) = V_t^S(i) \quad for \quad i = 1,2,\dots|S_t|,\ j = 1,2,\dots|P_t(i)| \qquad (2)$$

where $V_t^P(i,j)$ is the assumed velocity of the jth particle of the i^h storm cell at time *t*, $V_t^S(i)$ is the assumed velocity of the i^{th} storm at time *t*, $|S_t|$ is the cardinality of the set of storm cells at time *t*, and $|P_t(i)|$ is the cardinality of the set of particles in the i^{th} storm at time *t*. For example, if storm cell A is estimated to have a velocity of 30 km/hr and consists of 100 particles, all of those 100 particles are each estimated to have a velocity of 30 km/hr. Column vectors for the velocities of all particles at time *t* are then defined as:

$$\bar{V}_t^P = [V_t^P(1,1), V_t^P(1,2), \dots V_t^P(1,|P_t(1)|), \dots V_t^P(|S_t|,1), V_t^P(|S_t|,2), \dots V_t^P(|S_t|,|P_t(|S_t|)|)]' = [V_t^P(1), V_t^P(2), \dots V_t^P(|P_t|)]' \quad (3)$$

$V_{t,x}^P$ and $V_{t,y}^P$ are the respective *x* and *y* components of the of the particle velocities, V_t^P. Note, for standard notation in the remainder of this paper, particle indices will no longer sub-indexed by the storm cell they belong to.

A set of velocity matrices for time *t*+1 is then computed based on the location of particles at time *t* and *t*+1. The velocity matrices are defined as follows:

$$\begin{aligned} \bar{V}_{t,x}(i,j) &= \frac{x_{t+1}(j) - x_t(i)}{\Delta t} \\ \bar{V}_{t,y}(i,j) &= \frac{y_{t+1}(j) - y_t(i)}{\Delta t} \end{aligned} \quad for\ i = 1,2,\dots|P_t|, \quad j = 1,2,\dots|P_{t+1}| \quad (4)$$

Here, $x_t(i)$ and $y_t(i)$ are the *x* and *y* locations of the i^{th} particle $p_t(i)$ at time *t* and Δt is the time elapsed between time *t* and time *t+1*. Each entry in this velocity matrix is the velocity that the i^{th} particle $p_t(i)$ at time *t* must have in order to be at the location of the j^{th} particle $p_{t+1}(j)$ at time *t*+1.

3.3 Computation of Path Coherence Cost for a Simple Particle

Naturally, if particle $p_t(i)$ at time *t* is truly a match to particle $p_{t+1}(j)$ at time *t*+1, velocities $V_t(i)$ and $V_{t+1}(i,j)$ should be similar in magnitude and direction; i.e., the velocities or paths at those times should be coherent. A path coherence cost, C_{PC}, is computed for each pair (i,j) of particles at time *t* and particles at time *t*+1. The path coherence cost is a measure of the consistence in speed and direction of movement of the particle between positions at time *t* and *t*+1. The cost to associate particle $p_t(i)$ at time *t* with particle $p_{t+1}(j)$ at time *t*+1 is:

$$C_{PC}(i,j) = w_D T_D(i,j) + w_S T_S(i,j) \quad (5)$$

$T_D(i,j)$ is a penalty term for changes in direction of movement and $T_S(i,j)$ is a penalty term for changes in speed. The penalty terms are:

$$T_D(i,j) = 1 - \frac{V_t(i) \cdot V_{t+1}(i,j)}{\|V_t(i)\| \|V_{t+1}(i,j)\|} \quad (6)$$

$$T_S(i,j) = \begin{cases} 1 - \frac{2\sqrt{\Delta_{MAX}\,(\Delta_{MAX} - |V_t(i) - V_{t+1}(i,j)|)}}{2\Delta_{MAX} - |V_t(i) - V_{t+1}(i,j)|}, & |V_t - V_{t+1}| \le \Delta_{\text{MAX}} \\ 1, & |V_t - V_{t+1}| > \Delta_{\text{MAX}} \end{cases} \quad (7)$$

The terms w_D and w_S are weights on the individual costs where:

$$w_D + w_S = 1 \quad (8)$$

Here, Δ_{MAX} is the change in speed between times *t* and *t+1* that yields the maximum cost for change in speed. This path coherence function is a variation of the original coherence function defined by Sethi and Jain [11]. The speed penalty term has been modified to yield equal penalty for equal increases and decreases in speed and to define an allowable absolute maximum change in speed before reaching the maximum penalty value $T_S(i,j)$ of 1.

Path coherence is a better measure of matching objects moving over time since it takes into account not only the expected position of the object at time *t+1* but also the expected velocity. An object's motion cannot change instantaneously due to inertia. This is the concept behind path coherence. Equal error in predicted position is not the same as equal error in velocity. A path coherence cost matrix formed for each pair of particles at time *t* and *t+1*. Particle matches that correspond to highly coherent paths have a cost value near 0 and particle matches that do not have highly coherent paths have a cost value near $2w_D + w_s$. Any particle match that corresponds to a change in direction greater than 90° ($T_D(i,j) \geq 1$) or a change in speed greater than Δ_{MAX} ($T_S(i,j) = 1$) are set to have an infinite cost, eliminating the possibility of assignment.

3.4 Overall Path Coherence Cost

Once a complete path coherence matrix has been created, a combinatorial optimization algorithm is performed on the matrix to minimize the overall path coherence cost in particle assignment:

$$COST_{PC} = \sum_{j=1}^{|P_t|} \sum_{k=1}^{|P_{t+1}|} A(i,j) C_{Pc}(i,j) \qquad (9)$$

The matrix A is an assignment matrix containing at most a single one in each row and each column. These entries represent associations made between particles at times *t* and *t*+1. All other entries are zero, meaning no association is made.

The optimization algorithm employed to perform the assignment operation is the Hungarian Method [5]. This method assumes a one-to-one assignment problem, making the maximum number of possible assignments while assigning no more than one particle at time *t* to one particle at time *t*+1 and vice versa. In general, this means that if there are $|P_t|$ particles at time *t* and $|P_{t+1}|$ particles at time *t*+1, then the number of assignments that will be made is equal to $\min(|P_t|, |P_{t+1}|)$. This may not be the case if one or more particles cannot be assigned to any another particle. This may occur if all costs associated with a particle have been set to be infinite as is often the case for particles belonging to new storm cells at time *t*+1 or dissipating storm cells at time *t*. The basics of the Hungarian algorithm are as follows.

1. Assume there are N_t storms at time *t* and N_{t+1} storms at time *t*+1 that need to be temporally correlated. There is a different cost associated with each temporal assignment. Only one storm at time t may be assigned to one storm at time *t*+1. Create a matrix $N_t \times N_{t+1}$ of the associated costs. If $N_t \neq N_{t+1}$, the maximum number of possible assignments, $\min(N_t, N_{t+1})$, will be made.

2. For all rows in the cost matrix, subtract the row minimum value from all entries in that row.

3. For all columns in the cost matrix, subtract the column minimum value from all entries in that column.

4. Draw lines across rows and columns in such a way that all zeros are covered and that the minimum number of lines have been used. If the number of lines just drawn is $\min(N_t, N_{t+1})$, the algorithm terminates and the zero entries result in the optimal one-to-one assignment solution. If the number of lines is greater than $\min(N_t, N_{t+1})$, proceed to step 5.

5. Find the smallest entry which is not covered by the lines and subtract it from each entry not covered by the lines. Also add it to each entry which is covered by a vertical and a horizontal line. Return to step 4.

Once the particle assignment has been performed by a linear assignment optimization algorithm such as the Hungarian or Jacobi Auction algorithm, two probability matrices are formed, P_A and P_B. These matrices reflect the probability that two storm cells are a correct match between times *t* and *t+1*. They both have dimensions $|S_t| \times |S_{t+1}|$ where, as before, $|S_t|$ is the number of identified storms at time *t*. The probabilities that the i^{th} storm $s_t(i)$ at time *t* and the j^{th} storm $s_{t+1}(j)$ at time *t+1* are a match are:

$$P_A(i,j) = \frac{NumPts\ (i,j)}{Max\ (|P_t(i)|,|P_{t+1}(j)|)} \qquad (10)$$

$$P_B(i,j) = \frac{NumPts\ (i,j)}{Min\ (|P_t(i)|,|P_{t+1}(j)|)} \qquad (11)$$

Here, $NumPts(i,j)$ is the number of particles that were matched between storms $s_t(i)$ and $s_{t+1}\ (j)$ at in the optimization algorithm.

Both matrices are formed because P_A will favor large area storms, or storms with more particles, and P_B will favor small area storm matches, or storms with fewer particles. For example, suppose a storm at time t composed of 100 points splits in two at time t+1 where one of the new cells is composed of 80 particles and the other is composed of 20 particles. Suppose that all 100 particles are associated properly. In this case, P_A would have values of 80% and 20% respectively. On the other hand, P_B would have values of 100% in both cases. P_A is more discriminative. However P_B detects matches that occur from splits and mergers. In general, if P_A does not indicate a high likelihood of a match, then P_B may. Nevertheless, P_B must have an extremely high value to be considered valid match. This concept is especially important in the event of a split or merger event in order to be sure all the proper associations are made.

Two cost matrices are formed from the probability matrices P_A and P_B:

$$C_A = 1 - P_A \qquad (12)$$

$$C_B = 1 - P_B \qquad (13)$$

From these probability matrices, an initial set of storm assignments are made by performing only the first set of steps of the Hungarian algorithm once. Performing only steps 1-3 will find optimal assignment solutions for each row by means of step 2. Using step3, it will also find any secondary suboptimal solutions for each column that was not assigned a solution in step 2. Note that this solution does not require a one-to-one assignment. This allows for multiple associations in the event of splits. Performing steps 1-3 with steps 2 and 3 reversed does the same except finding the optimal solution for each column and the secondary set of suboptimal solutions for each row. This allows for multiple associations in the event of mergers. Therefore, steps 1-3, in order, are first performed on cost matrices C_A and C_B. Then, steps 1-3, with steps 2 and 3 reversed, are performed on cost matrices C_A and C_B. This creates a myriad of assignments, most of which are correct.

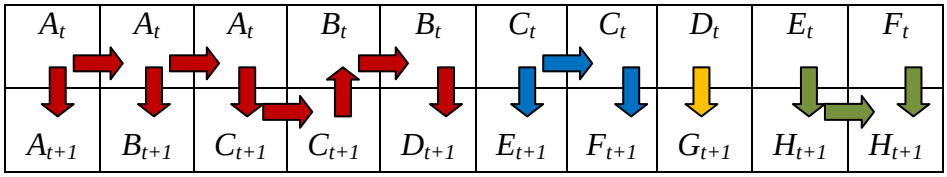
Lastly, storm cells are divided into subsets based on their previous assignment. For example, suppose the following assignments have been made:

A_t	A_t	A_t	B_t	B_t	C_t	C_t	D_t	E_t	F_t
A_{t+1}	B_{t+1}	C_{t+1}	C_{t+1}	D_{t+1}	E_{t+1}	F_{t+1}	G_{t+1}	H_{t+1}	H_{t+1}

The following subsets will be formed:

$$A_t, B_t \leftrightarrow A_{t+1}, B_{t+1}, C_{t+1}, D_{t+1}$$
$$C_t \leftrightarrow E_{t+1}, F_{t+1}$$
$$D_t \leftrightarrow G_{t+1}$$
$$E_t, F_t \leftrightarrow A_{t+1}, H_{t+1}$$

These subsets are formed by following the chain of all combinations of assignments for each storm cell:



The same particle assignment optimization as performed before on the entire set of storms is then performed on each storm subset. Two sets of probabilities are computed as before in Eq. 12 and 12. In order for an assignment to be valid:

$$\max(P_A, P_B) > .5 \qquad (14)$$

If this condition is not met for all assignments in the subset, then those that do not meet this criterion are discarded. The remaining assignments in the subset are maintained.

3.5 Velocity Estimation for a Storm Cell

Lastly, velocity must be estimated. For each subset that is maintained, the velocity of all storms in that subset are estimated as:

$$V = \frac{\left(\frac{\sum_{i=1}^{N_{t+1}} A_{t+1}(i) C_{t+1}(i)}{\sum_{i=1}^{N_{t+1}} A_{t+1}} - \frac{\sum_{j=1}^{N_t} A_t(j) C_t(j)}{\sum_{j=1}^{N_t} A_t} \right)}{\Delta t} \qquad (15)$$

Here, N_t and N_{t+1} are the number of storms at times *t* and *t+1*, respectively, in the current subset. A_t and A_{t+1} are the storm areas. C_t and C_{t+1} are the weighted centroids of the storms and Δt is the time between volume scans. If there are no splits or mergers occurring, the subset consists of a single storm at time *t* and a single storm at time *t+1*, and Eq. 12 reduces to the movement between the two weighted centroids:

$$V = \frac{C_j(t+1) - C_k(t)}{\Delta t} \qquad (16)$$

These velocities are then used to compute path coherence for the next time step and the tracking and association process is repeated.

4.0 Future Work

The proposed identification process applies the DBSCAN method in a two-dimensional space to form two-dimensional components. Vertical association methods required in the final identification of storm cells follows the standard technique of storm cell identification that has been applied by previous algorithms. Applying the DBSCAN algorithm in three-dimensions, while decreasing computational efficiency, will eliminate the need for many of the parameters currently required as input to the identification algorithm. The only parameters required will be those input to the DBSCAN algorithm and not those for vertical association of components. This will decrease the number of numeric parameters required to be input by the user and decrease sensitivity to user input. The algorithm results will be more consistent as they will not depend on the storm cells identified to have a certain shape or structure. It will not matter if a storm cell's reflectivity field merges with another on the highest level or the lowest level; the storm cell cores will still be distinguished. Furthermore, they will be defined by the largest volume possible just as the current method represents each component by the largest area possible.

Another implementation under consideration is Enhanced DBSCAN [10]. Enhanced DBSCAN is a variant of the traditional DBSCAN algorithm. It employs an additional parameter that allows for identification of clusters of varying densities. In this application, varying densities may correspond to employing a search for storm cells of different reflectivity strengths, eliminating the need to perform the identification algorithm for multiple different thresholds.

Initial efforts are being made to use this and future algorithms as a benchmark for design of a new highly computationally efficient computer. Such a design would make the

computational complexity of a three-dimensional DBSCAN algorithm very practical and advantageous, running the algorithm in faster than real time.

The tracking algorithm should also benefit from a highly computationally efficient computer. As previously mentioned, the tracking algorithm is based on a series of optimization problems that minimize path coherence cost between particles. In theory, decreasing the spacing between the storm cell particles, thus increasing the number of storm cell particles would increase the association accuracy of the algorithm. This comes at a high computational cost though. As the number of particles N increases, the order of complexity increases by $O(N)^3$.

Additionally, recall the tracking algorithm is currently initialized by a set of user input velocities based on a visual representation of identified storms. Velocity data from the radar or other independent sources could be used to initialize the algorithm, making it independent of a user. Velocity data may also be used in a control theory state space representation to constantly correct and re-estimate storm cell velocities.

With a highly accurate storm cell identification and tracking algorithm, much more accurate data on storm cells may be collected to calculate all types of storm cell properties. Over time, these values can be used to create a database of storm cells, their characteristics, and the natural phenomena they produced. This kind of database would be very advantageous in identifying storm cell properties that produce severe phenomena such as heavy rain, high winds, tornadoes, and hail. Specifically, collection of a very large database would be very valuable in employing various data mining techniques to make more accurate predictions with respect to severe weather phenomena.

References

[1] Dixon, Michael, and Gerry Weiner, 1993: TITAN: Thunderstorm Identification, Tracking, Analysis, and Nowcasting—A Radar-based Methodology. *Journal of Atmospheric and Oceanic Technology Vol. 10 No. 6,* pp. 785-797.

[2] Ester, Martin, Hans-Peter Kriegal, Jörg Sander, and Xiaowei Xu: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, Portalnd, WA, 1996, pp. 226-231.

[3] Han, Lei, Shengxue Fu, Lifeng Zhao, Yongguang Zheng, Hongqing Wang, and Yinjing Lin, 2009: 3D Convective Storm Identification, Tracking, and Forecasting—An Enhanced TITAN Algorithm. *Journal of Atmospheric and Oceanic Technology, Vol. 26, No. 4*, pp. 719-732.

[4] Johnson, J.T., Pamela L. MacKeen, Arthur Witt, E. DeWayne Mitchell, Gregory J. Stumpf, Michael D. Eilts, and Kevin W. Thomas, 1998: The Storm Cell Identification and Tracking Algorithm: An Enhanced WSR-88D Algorithm. *Weather and Forecasting Vol. 13*, pp. 263-276.

[5] Kuhn, Harold K., 1955: The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly Vol. 2*, pp. 83-97.

[6] Lakshmanan, V. , R. Rabin, and V. DeBrunner, 2003: Multiscale Storm Identification and Forecast. *Elsevier Atmospheric Research 67-68*, 367-380.

[7] Lakshmanan, V., K. Hindl and R. Rabin, 2008: An Efficient, General-Purpose Technique for Identifying Storm Cells in GeoSpatial Images.

[8] Mahran, Shaaban, and Khaled Mahar, 2008: Using Grid for Accelerating Desnity-Based Clustering. *8th IEEE International Conference on Computer and Information Technology*, pp. 35-40.

[9] Morel, C., F. Orain, and S. Senesi, 1997: Automated detection and characterization of MCS using the meteosat infrared channel. *Proc. Meteor. Satellite Data Users Conf.*, Brussels, Belgium, EUMETSAT, 213–220.

[10] Ram, Anant, Ashish Sharma, Anand S. Jalal, Raghuraj Singh, and Ankur Agrawal, 2009: An Enhanced Density Based Spatial Clustering of Apllications with Noise. *IEEE 2009 International Adanced Computing Conference*, pp. 1475-1478.

[11] Sethi, Ishwar K., and Ramesh Jain, 1987: Finding Trajectories of Feature Points in Monocular Image Sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.9 No. 1*, pp. 56-73.

[12] Uncu, Ozge, William A. Gruver, Dilip B. Kotak, Dorian Sabaz, ZafeerAlibhai, and Colin Ng, 2006: GRIDDBSCAN: GRId Density-Based Spatial Clustering of Applications with Noise. *2006 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2976-2981.

[13] University of Oklahoma and NOAA/National Severe Storms Laboratory, 2000-2006. *Warning Decision Support System—Integrated Information.*