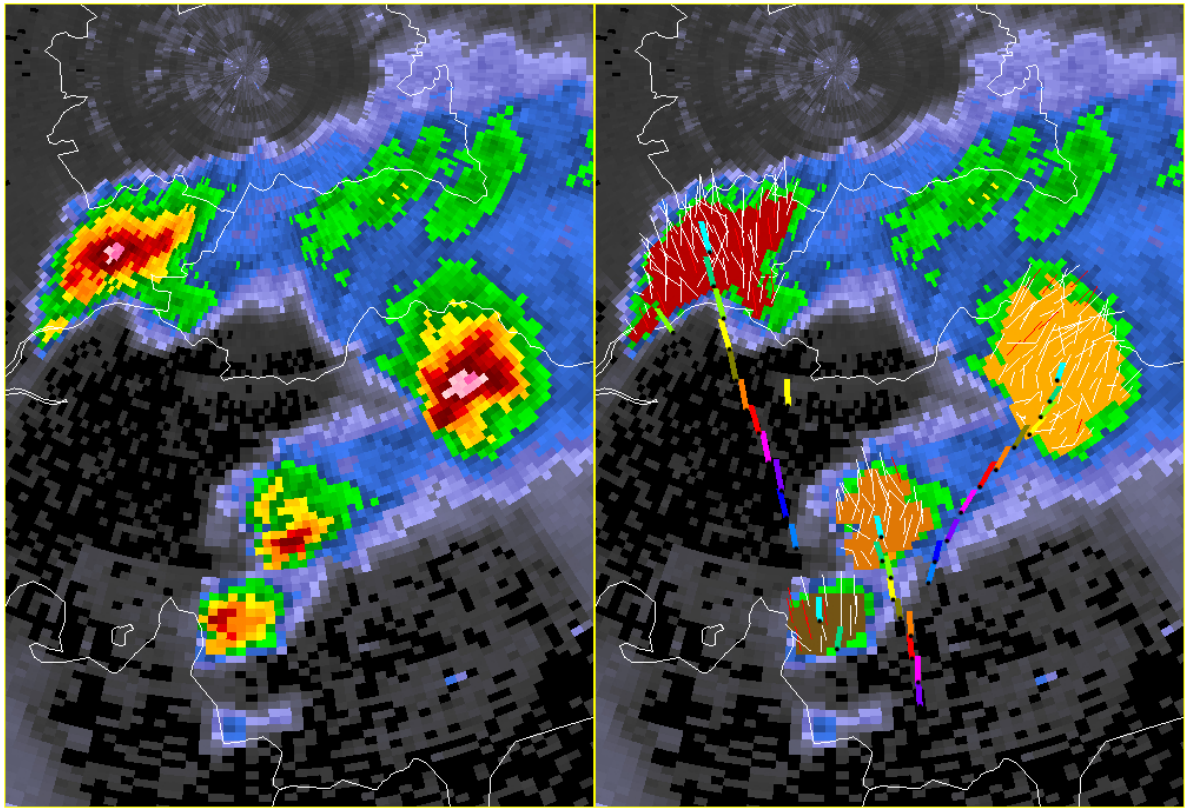
Creation and evaluation of a storm-tracking algorithm

Bachelor thesis

Bram van't Veen

Physics & Astronomy



Supervisors:
Sander Tijm
KNMI
and
dr. Aarnout van Delden
Utrecht University

June 14, 2017

Abstract

Thunderstorms can lead to flash floods, and can be accompanied by large hail, severe wind gusts and tornadoes, possibly having a high social impact. Accurate and detailed warnings for these storms are therefore desired. Because operational meteorologists, which monitor the weather and generate forecasts, are usually busy in situations with thunderstorms, they will benefit from algorithms that can automatically generate storm forecasts. These algorithms could determine the intensity of storms and their velocity, enabling it to produce a forecast. Such a forecast could provide a quick overview of storm intensities and velocities to operational meteorologists, or if accurate enough, might even be used to warn the public. The focus of this project is the creation and evaluation of a storm-tracking algorithm, which automatically produces storm motion estimates. Such a storm-tracking algorithm usually receives input from weather radars, which provide information about the precipitation intensity. The coupling (association) of precipitation patterns at consecutive times enables these algorithms to determine the motion. Several of these storm-tracking algorithms have been created in the past, but most of them suffer from storm interactions, e.g. the splitting of one storm into two, or the merging of two storms into one. This is because these algorithms perform one-to-one association of storms at consecutive times, which is not possible in the case of storm splits and mergers. Separate treatment of these processes is therefore required in these algorithms. The proposed new algorithm, named CBVELOCITY, uses a method that automatically treats storm interactions, in the same way as the usual translation of storms is treated. The key concept is the division of storm cells into subcells, which are subsequently associated between consecutive times, instead of the storm cells themselves. The velocity of the storm cells is then determined as the average of the velocities of the subcells in the storm. The subcell concept is based on the approach used in the storm-tracking algorithm of Matthews and Trostel (2010), but is optimized for determining velocity estimates instead of following thunderstorms through time. An evaluation of the performance of CBVELOCITY shows promising results compared to SCIT, the only existing algorithm to which it has been compared. It is further shown that the performance depends on the storm speed, with decreasing performance for decreasing storm speed.

1 Introduction

Automatic determination of the motion of thunderstorms is useful for severe weather nowcasting, i.e. the process of making short-term forecasts for thunderstorms. An algorithm that is able to do this is called a storm-tracking algorithm, and it could provide a quick overview of thunderstorm motions to operational meteorologists, or might be used for automatic generation of warnings for the public. With the latter one must however be very careful, because not only errors in the output of the storm-tracking algorithm hinder an accurate forecast, true changes in storm motions do so too.

A storm-tracking algorithm is usually based on data collected by weather radars, specifically the reflectivity. Weather radars emit radiation, which will be partly back-scattered when objects like thunderstorms are present in the path of the radar beam. The reflectivity is a measure of the amount of radiation that is back-scattered towards and collected by the radar. In conventional form it is labeled by Z, and has units of $\mathrm{mm}^6/\mathrm{m}^3$. A more convenient quantity is however often the logarithmic reflectivity, defined as $dBZ = 10 \log Z$. When writing reflectivity in this thesis, this is the quantity to which is referred. Z will be referred to as linear reflectivity.

Thunderstorms often appear as regions of high reflectivity. A weather radar repeats its scanning process every x minutes (often around 5), and a storm-tracking algorithm should couple(associate) areas of high reflectivity that are detected at different times, because this enables it do determine the motion of these areas.

The tracking of thunderstorms is generally most easy when the storms are isolated, because this makes it unambiguous to associate storms that are detected at consecutive times. In reality, thunderstorms can also appear in clusters however, and this can substantially hamper the association process. Because a wrong association leads to a wrong estimate for the storm motion, this also hampers the process of obtaining correct storm motion estimates. In addition to the possibility of the presence of multiple storms in close proximity to each other, it is also possible that one storm splits into two, or that two storms merge into one (storm interactions). In these cases a one-to-one association is not possible, or would at least lead to incorrect storm motion estimates. There are still other processes to take into account however, which are the development of new storms and the decay of existing ones.

It is the treatment of closely spaced storms, and storm splits and mergers for which a better method is sought. As described in the next section, most existing algorithms try to associate storms one-to-one (as a whole), with in some cases a separate treatment of storm splits and mergers. DBSCAN SCIT (Matthews and Trostel 2010) is an exception, and employs a general method for handling storm interactions, which does not require a separate treatment of storm splits and mergers. DBSCAN SCIT is however aimed at tracking (following) storms across time, and not at determining their motion. In this thesis, the focus lies specifically at determining storm motions, for which a modification of DBSCAN SCIT is deemed to be more appropriate. This new algorithm, named CBVELOCITY, is the subject of this thesis, together with an evaluation of its performance.

The thesis is structured as follows: Section 2 contains a description of existing storm-tracking algorithms, and section 3 introduces the new algorithm, CBVELOCITY. Section 4 gives a complete overview of its design. Section 5 describes the method by which its performance is evaluated, and in section 6 the evaluation is performed. Also included in section 6 is a discussion of these results, and its effect on possible applications of CBVELOCITY. Section 7 finally contains the conclusions.

Before starting with the description of existing algorithms however, it must be noted that storm-tracking algorithms are part of a larger class of algorithms, which will now be described briefly.

Storm-tracking algorithms are part of a larger class of algorithms, which determine the motion of precipitation based on the reflectivity at consecutive times. This larger class can be divided into three major groups (Wang et al. 2013).

1. There are algorithms that track the entire precipitation field by examining correlation between reflectivity patterns at different times. This technique is most appropriate to track large-scale precipitation patterns.

2. There are artificial neural network algorithms (ANN), in which the algorithm is learned to describe the evolution of precipitation patterns by training it with particular datasets (French et al. 1992). Such an approach can however not be expected to perform well for cases that differ substantially from the cases for which it has been trained.

3. Finally, there are the storm-tracking algorithms, which is a group of algorithms that is able to identify and track individual storm cells. This group has the advantage of being capable to deliver cell-based information, because it identifies each storm cell separately. This makes it possible to determine properties like the hail potential for individual storms.

Before continuing, it is finally mentioned that in the remainder of this thesis the terms (thunder)storms, storm cells and cells are used interchangeably. They all refer to the same phenomenon.

2 Previous work

2.1 TITAN

Development of cell-based tracking algorithms started in the late 70s. The first algorithm that received major attention was TITAN (Dixon and Wiener 1993): Thunderstorm Identification, Tracking, Analysis and Nowcasting. This algorithm uses a single reflectivity threshold to identify individual storms (in 3D), and tracks them by finding the optimum association of storms using the Hungarian algorithm. This is a combinatorial optimization algorithm, that finds the association of storms that minimized a particular cost function. Splits and mergers are treated seperately. Storm motions are then determined by applying a linear regression model to the history of centroid positions of the storm, where the number of historic positions is user-defined. Old centroid positions are exponentially weighted less. Finally, the growth and decay of storms is taken into account by assuming a linear trend for the storm area and volume.

It must be noted that in subsequent years this algorithm has been updated, and has grown into a relative large system designed for thunderstorm nowcasting: `http://www.rap.ucar.edu/projects/titan/home/whatis_titan.php`. The updated tracking algorithm (updated TITAN) uses two thresholds for storm identification, to better distinguish closely spaced storms. The first reflectivity threshold is the primary threshold, and the second threshold is used to divide storms indentified by the first threshold into parts. The storms identified by this second threshold are finally grown until they meet the boundary of the storm identified by the first threshold. Another difference is that the updated algorithm first tries to match storms at subsequent volumes using an overlapping technique, whereafter the remaining storms are matched using the Hungarian algorithm. A match by overlap occurs when storms overlap by at least a user-defined fraction.

2.2 SCIT

In another attempt to distinguish closely spaced storms, Johnson et al. (1998) developed SCIT: the Storm Cell Identification and Tracking algorithm. This algorithm uses 7 different reflectivity thresholds to identify storm components, and saves for each detected storm the smallest storm component that contains it (i.e. with the highest reflectivity). Storm components are subsequently vertically associated to obtain a 3D representation of a storm. Tracking is performed by comparing forecast positions of cell centroids at time 1 with observed positions of identified cells at time 2. Here and in the remaining part of the thesis, time 2 refers to the time for which velocities estimates and tracking information must be obtained, while time 1 refers to the time before time 2, for which this information is already available. The cell at time 2 that is closest to the forecast position of an existing cell is taken as the continuation of that track, except when a maximum distance is exceeded. Storm motions are determined by applying a linear least squares fit to the storm's current centroid position and up to 10 previous positions.

2.3 TRACE3D

Handwerker (2002) developed TRACE3D, which uses one reflectivity threshold to identify ROIP's (regions of intense precipitation). Subsequently, it determines the maximum reflectivity value in each ROIP, after which inside each ROIP cells are identified as regions for which the reflectivity at the edge differs by not more than a predefined value from this maximum reflectivity. As in SCIT, tracking is performed by comparing forecast centroid positions with observed positions, but here routines are included to identify storm splittings and mergers.

2.4 ETITAN

Another improvement to TITAN was given by Han et al. (2009), the algorithm called ETITAN. ETITAN uses multiple thresholds for storm identification, typically 2-6, in contrast to the fixed single(dual)-threshold approach of (updated) TITAN. Other changes are the use of two mathematical morphology operations for storm identification, the inclusion of a dynamic constraint in the application of the Hungarian algorithm, and the use of the correlation method TREC (Rinehart and Garvey 1978) to determine storm motions.

The morphology operations are erosion and dilation, where erosion is applied to identified cells to remove small segments that could connect closely spaced cells, and would lead to detect them as one cell. The dilation method is applied at cells identified by the second or greater threshold, and is used to grow them until they meet the boundary of a cell identified by the lower threshold. The dilation method preserves the shape of the cell as identified by the higher threshold, in contrast to the growing method used by the updated TITAN

algorithm. The dynamic constraint added in the application of the Hungarian algorithm relates the maximum allowed velocity to the storm area, where the velocity is the velocity required to go from a cell at time 1 to another cell at time 2. Storms that span a greater area are allowed a larger velocity, to account for erratic behaviour of the position of the storm centroid due to changes in the covered area. Finally, ETITAN uses the storm motions calculated by TREC instead of those calculated by applying linear regression to cell centroid positions, to decrease errors due to erratic movement of the cell centroid.

2.5 DBSCAN SCIT

A substantially different approach to tracking storms was adopted by Matthews and Trostel (2010). They determine storm areas instead of centroid positions, and subsequently divide these into particles, which are then associated to particles at another time by means of combinatorial optimization. First, 2D storm components are identified by applying the DBSCAN algorithm (Martin et al. 1996). For this reason the algorithm is here termed DBSCAN SCIT, although the authors did not provide a name for it.

As in SCIT, the 2D storm components are identified for 7 different reflectivity thresholds. In contrast to SCIT however, DBSCAN SCIT saves the largest storm component that uniquely encompasses a particular storm. Vertical association subsequently takes places by finding overlapping (in a plan view) 2D storm components. The storm cells now identified are divided into particles on a uniform grid. These particles are subsequently associated using the Hungarian algorithm, where the cost function is composed of penalty terms for changes in particle speed and direction. What next follows is a complex method to determine storm associations and velocities.

First, probability matrices P_A and P_B are formed, where the elements $P_A(P_B)_{ij}$ give the ratio of the number of matched particles between storm i and j, and the number of particles of the largest(smallest) of those 2 storms. Then cost matrices $C_A = 1 - P_A$ and $C_B = 1 - P_B$ are formed, and storm associations are generated by applying the Hungarian algorithm with some modifications to these matrices, resulting in a myriad of associations. Then, for each subset of storms the particle optimization procedure is repeated, and all resulting associations with $\max(P_A, P_B) > 0.5$ are approved. No different routines for handling splits and mergers are required in this approach. Finally, for each storm subset centroid positions are determined, both for the set of storms at time 1 and the set of storms at time 2. Such a centroid position is the average of the centroid positions of the different storms, weighted by the area of the storms. The velocity of the storm subset is then estimated as the velocity required to go from the centroid position at time 1, to that at time 2.

2.6 AALTO

The previous algorithms were all designed for data from a single radar, or for data from multiple radars that has been combined into a composite. AALTO (Limpert et al. 2015) can ingest data from multiple radars, without the need to combine it into a single composite. In addition to that, it examines future object positions to associate objects, which only works when operating on archived data. The first part of the tracking process consists of searching for new objects to continue existing tracks. This is done by finding all objects that are within a particular distance from the forecast position of an existing object, and lie within a particular search angle interval. If there is only one such object, then it is regarded as the continuation of the existing track. If there is more than one, then a search tree is built that includes all possible candidate tracks, with tracks consisting of the object positions at the 'current' and future times. The first-level branch (object at the 'current' time) of the track with the lowest mean position error, defined as the distance between the forecast position and the actual position along the track, is chosen as the continuation of the existing track. With increasing level, branches in the search tree are exponentially weighted less, and when dropping below a user-defined threshold, they are no longer added to the search tree. Forecast positions of objects are determined from the motion history over the previous 30 minutes.

3 Proposement of new algorithm; CBVELOCITY

A new storm-tracking algorithm has been developed, which specifically aims at providing storm motion estimates for storms during nowcasting. It does not aim at determining the complete track history of thunderstorms, implying that it is not suited for climatological purposes in which properties like the life time of thunderstorms are examined. Because of the focus on nowcasting, it further holds that approaches designed specifically for archived data (as present in AALTO) do not work.

The new algorithm is named CBVELOCITY, standing for cell-based velocity. To overcome problems with closely spaced storms that arise when associating storms between different times, CBVELOCITY adopts the particle (but from now on referred to as subcell) representation from Matthews and Trostel (2010), and also

associates these subcells using combinatorial optimization. There are however some important differences with their algorithm, which are the following:

1. CBVELOCITY determines the complete area spanned by the storm cells, or at least the part with a reflectivity above a user-defined threshold. DBSCAN SCIT also determines storm areas, but these do not have to be the complete areas spanned by a storm. The 2D storm components saved in DBSCAN SCIT are the largest ones that uniquely encompass a particular storm, but no enlarging operation is applied that associates every surrounding area of reflectivity (contiguous to the storm component) that is above a particular threshold to the storm component. In updated TITAN and ETITAN (the dilation method) such an enlarging method is applied however, where the method used by updated TITAN resembles the one used by CBVELOCITY the most.

   Having determined the complete area spanned by a storm has the advantage of making it easy to associate properties like the vertically integrated liquid or maximum expected hail size to an identified storm cell, which provide information about its intensity.

2. Penalty terms in the cost function are not only dependent on the velocity, but also on the reflectivity. It implies that subcell associations that lead to a small change in reflectivity are preferred over those that are associated with a large change. This appears more natural.

3. The complex method by which the final association of storms takes place is replaced by a much simpler one. The approach of associating storms as a whole is dropped completely, and is replaced by simply determining storm motions as a weighted average of the velocities of the subcells that constitute the storm.

The approach of associating subcells instead of storm cells has the advantage of being substantially less sensitive to erratic movement of cell centroids. Such an erratic movement can be caused by the growth of new cells on the flank of an existing cell, or more generally by cell mergers and cell splits.

Most previous algorithms mitigate the error in the storm motion estimate that results from erratic movement by also taking previous centroid positions into account (up to 30 minutes back, or 10 volumes $\sim$ 50-60 minutes for SCIT). This has however the disadvantage of decreasing the ability of the algorithm to handle true changes in storm motion. With the default settings, CBVELOCITY uses only the last (time 2) and one previous velocity (time 1), enabling it to detect these changes in storm motion soon.

CBVELOCITY will only associate those subcells whose change in velocity required for going from their position at time 1, to the position of another subcell at time 2, is smaller than a predefined value. The result is that only the most likely associations are made.

4 Algorithm design

A storm-tracking algorithm consists of a storm cell identification part and a tracking part, and these parts are treated in the following two subsections. The description includes multiple parameter names, and these are all listed in the appendix, where also the function and default values of these parameters are described.

Frequent use is made of the term (radar) bin in the text, which refers to a grid cell of the polar grid on which the reflectivity data is available. It is also used to indicate elements in the matrices in which the data is stored, where the bin with indices (i, j) refers to the i^{th} row and j^{th} column in this matrix. This row and column correspond to the i^{th} azimuthal sector and j^{th} radial sector in the polar grid on which the data is available, as sketched in figure 1.

4.1 Cell identification

Most weather radars scan at different elevation angles (angles with the vertical), to obtain information at multiple heights above the ground. For an estimation of the precipitation intensity at the ground, it is most appropriate to use reflectivity data from the lowest scan. When however attempting to detect newly formed storms, that only have a precipitation core above the ground, it is more appropriate to use data from higher scans.

CBVELOCITY takes into account the reflectivity data from all scans, because it uses the maximum reflectivity in a vertical column. It is available on a 2D polar grid, in which each radar bin contains the maximum reflectivity in the vertical direction, stored in the matrix Z_{max}. Z_{max} includes only contributions from reflectivity data that was obtained above a height of 1.5 km. This is done to reduce the amount of clutter near the radar. These clutter echoes are peaks in reflectivity that are not associated with precipitation, but are caused

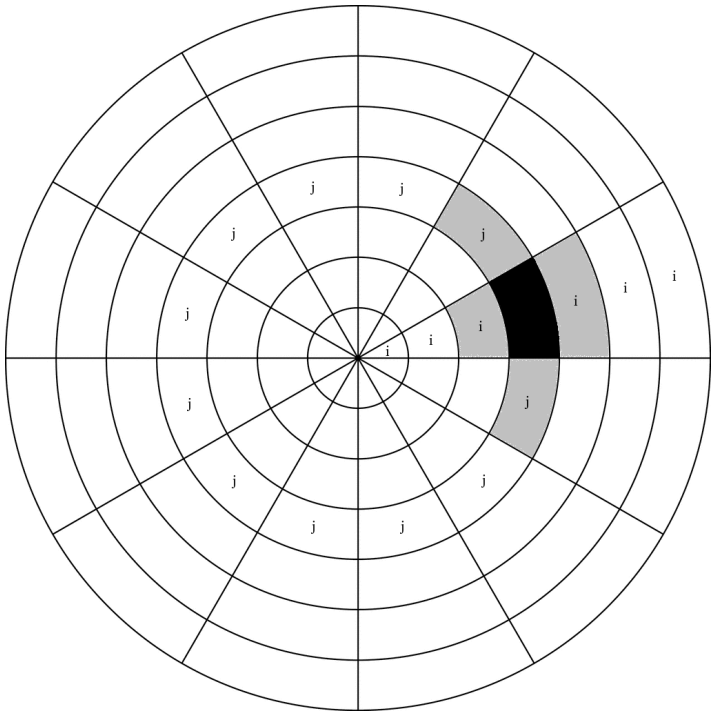


Figure 1: Polar grid at which reflectivity data is available. Labeled are the i^{th} azimuthal sector and the j^{th} radial sector, which correspond to the i^{th} row and j^{th} column of the matrix that stores the data that is located at the grid. Also marked with grey are the 4 neighbouring radar bins of the selected black bin.

by reflections on e.g. tall buildings. The height at which the radar scans (at which the center of the radar beam is located) is calculated assuming the $\frac{4}{3}$ earth radius model (Dovak and Zrnić 1993, p. 21).

In addition to the presence of clutter, another problem to take care of is the presence of so-called bright band contamination, which arises in situations with melting precipitation. This bright band is a band of higher reflectivities at the height of the melting level, which does not arise from a higher precipitation rate. It is caused by a combination of the increase in radar reflectivity when ice crystals melt (rain reflects stronger), and the still relatively low fall speeds of precipitation particles (ice crystals have a lower fall speed). This relatively low fall speed gives a higher number density of precipitation particles, which in combination with the stronger reflection leads to the maximum in reflectivity (AMS glossary of meteorology, 'bright band').

Because of the higher reflectivity, the presence of the bright band leads to an overestimation of the precipitation intensity. The use of Z_{max} implies that the bright band is prevalent in areas with stratiform precipitation, i.e. areas with low to moderate precipitation rates. This kind of precipitation occurs often when a weather front passes, or surrounds the stronger precipitation cores in thunderstorm complexes. A solution for the bright band contamination is not included in the present form of the algorithm. An example of the resulting product is shown in figure 2, in which the bright band signature is visible in the form of (parts of) bands of reflectivities above 40 dBZ, embedded in larger areas of lower reflectivity.

The next part is the identification of storm cells in Z_{max}, which is done using multiple reflectivity thresholds. These thresholds range from MIN_DBZ_THRESHOLD to MAX_DBZ_THRESHOLD, and increase in steps of STEP_DBZ_THRESHOLD. Thresholds are denoted by T^n, with n the index of the threshold, increasing with increasing threshold value. In the following it is assumed that the reflectivity data is given at a polar grid, but modifying it for use at a Cartesian grid is relatively straightforward. This part consists of the following steps:

1. For each reflectivity threshold, bins are sought that satisfy the condition $Z_{max} \geq$ threshold. These bins are subsequently only retained when at least 2 neighbouring bins also satisfy this condition, and are then called core objects. A neighbouring bin is one of the four radar bins that is located below, above, to the left or to the right of the radar bin (as seen from the radar view point) that is currently checked (figure 1).

   The condition that at least 2 neighbouring bins should also have a reflectivity above the threshold is included to remove the amount of clutter, which at heights above the ground can be caused by reflection on air planes. Other sources of clutter in Z_{max} can be caused by the downward bending of the radar beam and subsequent reflection on objects at the ground, which occurs in situations with a temperature

inversion. This is called anomalous propagation, in which the 4/3 earth radius model overestimates the height of the center of the radar beam (AMS glossary of meteorology, 'anomalous propagation')

2. For each threshold the core objects are grouped into clusters, in which each core object has at least one other core object of the same cluster in its 4 neighbouring radar bins. The combination of step 1 and this clustering method is a slightly modified version of the DBSCAN method used by Matthews and Trostel (2010). Clustering is performed by the following algorithm, executed for each threshold independently:

Clustering algorithm

Clusters defined here, and all core objects belonging to it, are given a unique cluster ID, stored in a matrix C^n with the same shape as Z_{max}. C^n is the matrix with clusters that corresponds to the threshold T^n.

Starting with the radar bin $(i_0, 1)$, where i_0 denotes an arbitrary row (azimuth) in Z_{max}, and 1 refers to the first column (radius) in that row, the algorithm checks for each bin (i, j) whether it contains a core object. If true, then it checks whether there is also a core object in the bins $(i-1, j)$ and $(i, j-1)$ (that have been treated before). If false, then $C^n_{i,j}$ is assigned a unique cluster ID, given by $\max(C^n) + 1$. If true, there are two possibilities:
The core objects in the bins $(i-1, j)$ and $(i, j-1)$ can either belong to two different clusters, or to only one. In the latter case $C^n_{i,j}$ is assigned the single cluster ID in bins $C^n_{i-1,j}$ and $C^n_{i,j-1}$. In the former case it is assigned the cluster ID $\min(C^n_{i-1,j}, C^n_{i,j-1})$. In this case, elements in C^n that are equal to $\max(C^n_{i-1,j}, C^n_{i,j-1})$ are also replaced by $C^n_{i,j}$, because it appears that they belong to the same cluster.
The above process of assigning cluster ID's is repeated for each row (azimuth) and column (radius) in Z_{max}, where the loop over columns is the inner loop.

In the case of a polar grid, care must be taking with the handling of the first and last row, because of the periodic boundary conditions. This is done by iterating through the first row twice, giving two sets of cluster ID's. Next, a map $I : k \to l$ is created, where k denotes the new ID and l the list of old ID's that correspond to k. Then for each $k = k_i$, all ID's in C^n equal to k_i are replaced by the first element in $l_i = I(k_i)$, $l_i(1)$. ID's in C^n that are equal to other elements in l_i are also replaced by $l_i(1)$.
Finally, if l_i contains more than one element, then for each $k \neq k_i$ it is checked whether $I(k)$ contains elements that are contained in l_i. If so, these elements in $I(k)$ are also replaced by $l_i(1)$ (otherwise $I(k)$ would contain ID's that do not exist anymore). The combination of these steps assures that a group of contiguous bins is assigned to only one cluster.

3. The final step consists of combining the information gathered for the different dBZ thresholds, leading to a matrix C that contains the identified cells. As for the C^n, this matrix has the same shape as Z_{max} and contains cell ID's, where all bins that belong to the same cell get assigned the same cell ID.

   The approach in this step is based on the fact that the clusters identified for a lower dBZ threshold are always greater than or equal to the corresponding clusters for a higher dBZ threshold. It is executed by the following algorithms:

Cluster combining algorithm

For each C^n, define I^n as the list of cluster ID's contained in C^n. Also, for each cluster ID $I^n(i)$ in I^n, define $C^n\{i\}$ and $C\{i\}$ as the set of bins in C^n and C that is occupied by $I^n(i)$.

At the start, set $C = C^m$, with m the index of MAX_DBZ_THRESHOLD. Also set $I = 1$, where I is a cell ID. Then, for each cluster ID $I^{m-1}(i)$, check whether $C^{m-1}\{i\}$ encompasses more than one cluster in C. If not, then set $C\{i\} = I$; $I = I + 1$. If it does, then it must be checked if these clusters should be combined into one cell, or if they should be regarded as different cells. In order to do this, first determine for each of the k encompassed clusters in C its maximum reflectivity. These maximum reflectivities are stored in increasing order in a list M, such that $M(j)$ contains the maximum reflectivity of the encompassed cluster with the j^{th} smallest maximum reflectivity.
If $M(k-1) \leq T^{m-1}$ + MAX_DBZDIFF_COMBINING_CLUSTERS, then set $C\{i\} = I$; $I = I + 1$. This implies that when the second largest maximum reflectivity is less than the threshold, it is assumed that all encompassed clusters belong to one cell.

If $M(k-1) > T^{m-1}$ + MAX_DBZDIFF_COMBINING_CLUSTERS, then it is assumed that

the encompassed clusters belong to 2 or more cells. In this case, for each encompassed cluster $j \in C$ it is checked whether $M(j) \leq T^{m-1}$ + MAX_DBZDIFF_COMBINING_CLUSTERS. If true, then apply the *Assign bins to nearest clusters* algorithm to this cluster, which distributes its encompassed area over the other encompassed clusters. The cluster which receives the largest amount of bins is the one to which cluster j is assigned. Cluster j is then removed from the list of encompassed clusters. If false, then nothing occurs at this point (then it is one of the clusters that is identified as a different cell, to which other clusters can get assigned). This process is repeated for all encompassed clusters.

The result is now that the clusters in C encompassed by $C^{m-1}\{i\}$ are distributed over all encompassed clusters $j \in C$ that satisfy $M(j) > T^{m-1}$ + MAX_DBZDIFF_COMBINING_CLUSTERS, and which are recognized as different cells. This does not yet include the extra area spanned by the cluster in C^{m-1} however, and this is handled by application of again the *Assign bins to nearest cluster* algorithm. This time it distributes the complete area spanned by $C^{m-1}\{i\}$ over the encompassed cells in C.

The above set of actions is repeated for each dBZ threshold T^i, $i = m-1, ..., 1$, and matrix C gets updated during each iteration. The last step is the exclusion of cells for which the maximum reflectivity is less than Min_MAXIMUM_DBZ, which is included to remove very weak cells and areas of stratiform precipation (if the reflectivity in the bright band is not too strong).
Finally, for the sake of simplicity cell ID's are changed in such a way that the lowest is equal to 1, and they also increase in steps of 1.

Assign bins to nearest cluster algorithm

This algorithm distributes a set of radar bins B over a set of clusters C, with the goal to assign each bin in B to the nearest cluster in C. The matrix that contains the enlarged clusters, as emerging during the execution of the algorithm, will be denoted by A. At the start, all bins in A occupied by a cluster in C are filled with the corresponding cluster ID, and the remaining bins are filled with zeroes. During the execution of the algorithm, all bins in A that belong to B get filled with the ID of the cluster to which they get assigned.
Expansion of the clusters is realized by checking for all unassigned bins whether there are neighbouring bins that do belong to a cluster, and if so, the bins are assigned to this cluster. This expansion is repeated until all bins in B are assigned. It is attempted to let this expansion take place as uniform as possible in all directions. The polar nature of the grid does provide some difficulties however, because of the nonuniform radar bin dimensions. The method given below attempts to deal with this in a reasonable way. The text frequently refers to the left, right, bottom or top neighbour, which should be interpreted as being valid from the radar view point (figure 1).

The first step is to check for neighbours in the azimuthal direction, where an equal amount of checks is performed for left and right neighbours. This is done by checking for each $A(i,j) \in B$ that is not yet assigned to a cluster, whether $A(i \pm k, j)$, $k \in \{1, ..., \text{ceiling}(n_a/2)\}$ contains a cluster ID. Here $n_a = \max(2, \frac{\Delta r}{r\Delta\phi})$, with Δr the radial dimension of a radar bin in km, and $\Delta\phi$ the radar beamwidth in radians. It gives the number of checks in the azimuthal direction that is required to span a distance equal to Δr, or if $\Delta r < r\Delta\phi$, it is set to 2. The last part ensures that at least one check is performed for both left and right neighbours. When taking $A(i \pm k, j)$, care must be taken with the first and last row in A, because of the periodicity in the azimuthal direction. If a cluster ID is present in $A(i \pm k, j)$, then set $A(i,j) = A(i \pm k, j)$. If not, then continue.
The second step is to check for neighbours in the radial direction, where now an equal amount of checks is performed for bottom and top neighbours. This is here done by checking for each $A(i,j) \in B$ that is not yet assigned to a cluster, whether $A(i, j \pm k)$, $k \in \{1, ..., \text{ceiling}(n_r/2)\}$ contains a cluster ID, with $n_r = \max(2, \frac{r\Delta\phi}{\Delta r})$. Analogous to n_a, n_r gives the number of checks in the radial direction that is required to span a distance equal to $r\Delta\phi$, or if $r\Delta\phi < \Delta r$, it is set to 2, such that at least one check is performed for both bottom and top neighbours. If a cluster ID is present in $A(i, j \pm k)$, then set $A(i,j) = A(i, j \pm k)$. If not, then continue.
Thirdly, repeat step 1, but now first check for right instead of left neighbours.
Fourthly, repeat step 2, but now first check for top instead of bottom neighbours.
These last steps are included to reduce the preference of clusters for expanding in a particular direction.

Finally, cells that span an area of less than CELLS_MINIMUM_AREA, or that occupy less than CELLS_MINIMUM_NUMBEROFBINS radar bins are removed, to reduce the chance that clutter is identified as a cell, and to remove very small cells. The result of applying the complete cell identification

algorithm is shown in figure 3.

4.2 Cell tracking

As noted in section 3, CBVELOCITY uses combinatorial optimization to assign subcells, instead of the cells themselves. The first step is therefore to divide the cells into subcells, which is done by the following algorithm. An example of the resulting subcells is shown in figure 4.

Divide cells into subcells algorithm

As for the cell matrix C, a matrix S is created that will contain subcell ID's in all bins occupied by subcells. As was defined in the description of the *Cluster combining* algorithm, $C\{i\}$, $S\{i\}$ denotes the set of bins in C, S that is occupied by a cell, subcell with ID i. For each cell with cluster ID i in C, do the following:

First determine the values x_{min}, x_{max}, y_{min} and y_{max}, which are the minimum and maximum x and y coordinates of all bins in $C\{i\}$. Then create a rectangular grid with vertices at positions (x_{min}, y_{min}), (x_{min}, y_{max}), (x_{max}, y_{min}) and (x_{max}, y_{max}), and with grid spacing determined by DESIRED_1D_SPAN_SUBCELLS. The true grid spacing can be different in the x and y directions, and is also determined by $x(y)_{span} = x(y)_{max} - x(y)_{min}$. Because an integer number of subcells is required in both directions, given by $n_{x(y)} = \text{ceil}(x(y)_{span}/\text{DESIRED\_1D\_SPAN\_SUBCELLS})$, the grid spacings in both directions are given by $\Delta x(y) = x(y)_{span}/n_{x(y)}$.
The desired positions of the subcell centers are now available in Cartesian coordinates, and the next task is to find for each subcell the radar bin in S in which it is located. These radar bins get assigned a subcell ID, increasing in steps of 1.
The final task for each cell with ID i is to distribute all bins in $C\{i\}$ over the subcells, which is done using the *Assign bins to nearest cluster* algorithm.

Finally, as was for simplicity also done for C, subcell ID's in S are changed in such a way that the lowest is equal to 1, and they also increase in steps of 1.

In what now follows, the tracking aspect of CBVELOCITY is described. The algorithm is initialized by assigning an initial velocity to all cells at the initialization time, if there are cells present. This initial velocity estimate is currently a single velocity, but this could be modified to using different estimates at different locations. The initial velocity estimate is given by VELOCITY_ESTIMATE_NEWCELLS.
After all the above algorithms have been applied for 2 consecutive times t_1 and t_2, two sets of cells and subcells are obtained, with matrices denoted by C_1, C_2, S_1 and S_2. The maximum reflectivities (Z_{max} are also available for these times and stored in matrices, which are for brevity denoted by Z_1 and Z_2. For time t_1 there are also cell and subcell velocities known, for t_2 these have to be determined. The cell and subcell velocities at t_1 are stored in matrices V_1^c and V_1^s, where $V_1^{c(s)}(i)$ contains the velocity of the (sub)cell with ID i.
When a cell was detected for the first time at t_1, then the velocities of all subcells inside it are set equal to that cell velocity. At later times this does not occur, to retain nonuniformities in the motion of the subcells inside a cell, which is e.g. useful for an early detection of storm splitting.
The goal is now first to construct a cost matrix T, in which each element $T_{i,j}$ contains the cost associated with the association of a subcell i at t_1, with a subcell j at t_2. The cost is composed of two terms, stored in matrices TV and TR. The first is related to the change in velocity that is required to go from subcell i to j, and the second is related to the difference in the average reflectivity of subcells i and j. The cost matrix is then given by $T = \text{WEIGHT\_V} \cdot TV + \text{WEIGHT\_R} \cdot TR$.

4.2.1 Creation of cost matrix

In order to determine the first term in the cost matrix, it is necessary to determine the centroid positions of the subcells. For a subcell k these are calculated as

$$r_{1(2)}(k) = \overline{R(C_1(2)\{k\})}, \tag{1}$$

where the overbar operator gives an unweighted average. The R operator gives the position of a radar bin centroid, in polar coordinates given by $(r + \Delta r/2, \phi + \Delta\phi/2)$, when the radar bin is located between the radial lines with azimuths given by ϕ and $\phi + \Delta\phi$, and the circles with radii r and $r + \Delta r$.
Next, a velocity matrix V is constructed, that contains for each pair of subcells i and j the velocity required to go from i to j. Its elements are given by

$$V_{i,j} = \frac{r_2(j) - r_1(i)}{\Delta t}, \quad \Delta t = t_2 - t_1. \tag{2}$$

The cost matrix term TV can now be constructed, and has elements given by

$$TV_{i,j} = \frac{\|V_1^s(i) - V_{i,j}\|}{MAX\_DV}, \tag{3}$$

i.e. they are equal to the ratio of the norm of the change in velocity associated with the association of subcell i with j, and a maximum allowed norm of this velocity difference.

For the calculation of the second term in the cost matrix, TR, first the (unweighted) average reflectivity per subcell with ID k is calculated as

$$\bar{Z}_{1(2))}(k) = \overline{Z_{1(2)}\{k\}}. \tag{4}$$

The cost matrix term TR is then calculated as

$$TR_{i,j} = \frac{|\bar{Z}_2(j) - \bar{Z}_1(i)|}{MAX\_DR}, \tag{5}$$

where MAX_DR denotes the maximum allowed change in reflectivity in a time step Δt.

The next step in the creation of the cost matrix is to ensure that associations with $TR > 1$ or $TV > 1$ are not made, because for these associations the maximum allowed velocity change or reflectivity change is exceeded. This can be done by setting the cost for these associations to infinity, or at least a value high enough (which must be the same for each unallowed association), to ensure that these associations are not made until after the maximum number of allowed associations is made. It is then possible to remove these unallowed associations afterwards.
The final step consists of ensuring that subcell j at t_2 can only get associated with subcell i at time 1, if the velocity of subcell j differs by not more than MAX_DV_SUBCELL_CELL from the velocity of the cell to which subcell i belongs (at time 1). This step is included to prevent subcell velocities from deviating too much from the velocity of the cell to which they belong, as this increases the possibility of false associations when multiple cells are close together. It is prevented in the same way as above, by assigning a high enough cost to these associations.

4.2.2 Subcell association and estimation of subcell velocities

The associations of subcells is realized by the Jonker-Volgenant algorithm (Jonker R. and Volgenant A. 1987), using the Python implementation provided by Kazmar T. [8], specifically the LAPJV implementation. This algorithm determines the minimum cost association, and due to the fact that unallowed assignments have a very high cost, this minimum cost association will maximize the number of allowed assignments as mentioned above.
The result of these associations is that the associated subcells at t_2 can be assigned a velocity.

$$V_2^s(j) = V(i,j), \tag{6}$$

where i is the row index (subcell ID at t_1) of the association for column j (subcell ID at t_2).

Some averaging of the velocities over time is desired however, to smooth out irregularities. This is accomplished by defining another matrix with subcell velocities, V_{12}^s, which has the same dimensions as V_2^s. It includes for each subcell j the unweighted average over the most recent SUBCELLVELOCITIES_AVERAGING_NTIMES velocities for the chain of associated subcells in which j is the last one. If the chain is shorter than SUBCELLVELOCITIES_AVERAGING_NTIMES, then only the available velocities are used in the calculation of the average. The corresponding matrix with cell velocities is $V_{1,2}^c$, which will in the end contain the cell velocities that are given as output by CBVELOCITY.

Not all subcells at t_2 will get associated with a subcell at t_1 however, and for these subcells the following algorithm is used to estimate their velocity. It is written for V_2^s, but is executed in the same way for $V_{1,2}^s$, with replacement of $V_2^{s(c)}$ by $V_{1,2}^{s(c)}$.

Estimating velocity unassigned subcells algorithm
For each unassociated subcell j, it is first checked whether there are other subcells in the same cell that got assigned a velocity. If there are at least
n =UNASSIGNEDSUBCELLS_NREQUIRED_NEIGHBOURING_SUBCELLVELOCITIES of them, then the velocity for subcell j is estimated to be equal to the (unweighted) average of the velocities of the n

nearest subcells.
If there are $k < n$ of them, then the subcell velocity is estimated as

$$V_2^s(j) = \frac{\sum_{i=0}^{k} V_2^s(i) + (n-k)V_m^c}{n}, \tag{7}$$

where V_m^c is an average velocity over neighbouring cells. It includes a contribution from VELOCITY_ESTIMATE_NEWCELLS if there are not enough cells that lie within a range of UNASSIGNEDSUBCELLS_MAXSEARCHRANGE_CELLVELOCITIES from subcell j. V_m^c is calculated as

$$V_m^c = \frac{\sum_{i=0}^{l} V_{t_i}^c(i) + (m-l)V_{new}}{m}, \tag{8}$$

where m =UNASSIGNEDSUBCELLS_NREQUIRED_NEIGHBOURING_CELLVELOCITIES is the number of cells over which the average velocity should be determined, and $l <= m$ is the number of cells that satisfies the range condition, with a maximum of m. If there are more than m cells that satisfy the range condition, then only the m cells nearest to subcell j are used. If $l < m$, then V_{new} =VELOCITY_ESTIMATE_NEWCELLS in included in the calculation, and is given a weight of $m-l$.

t_i is the time for which the cell velocity is valid. Because cell velocities at t_2 are not yet available, the cell velocities for $t_1 - \delta t <= t <= t_1$ are used in the calculation of V_m^c, where the range of times is determined by δt =TIMERANGE_NEIGHBOURING_CELLS. One reason for taking into account cell velocities from before time 1, is that it allows CBVELOCITY to better track cells that where lost during a short time because of severe attenuation of the radar beam by precipitation, leading to an underestimation of the reflectivity. For $\delta t > 0$, it is possible that one cell contributes more than once in the calculation of the average (but with likely different velocities, because they are estimated for different times).

For determining which cells are nearest, the centroid positions of the cells must be determined. These are calculated as in eq. 1, with the difference that the unweighted average is replaced by an average weighted by linear reflectivity (given by $Z^l = 10^{Z/10}$). This is done to reduce the contribution of regions of relatively low reflectivity (just above MIN_DBZ_THRESHOLD) to the placement of the cell centroid.
If there are more than m cells that satisfy the range condition, then before determining which cells are nearest, the cell centroid positions are extrapolated in time according to their estimated velocity at time t_i.

An example of the resulting subcell velocities is shown in figure 5. Shown is a storm that is about to split, as is hinted at by the distribution of subcell velocities.

4.2.3 Estimating cell velocities

After application of this algorithm, a velocity estimate is available for each subcell at t_2, in both matrices V_2^s and $V_{1,2}^s$. It is therefore now possible to calculate a velocity estimate for each cell at t_2. In order to do this, first the subcell velocities are distributed over a matrix U, in such a way that $U\{j\} = V_2^s(j)$, where $U\{j\}$ denotes the set of bins in U that is occupied by the subcell with ID j. The velocity for a cell at t_2 is then calculated as

$$V_2^c(j) = \frac{\sum_{i=0}^{n} Z_2^l(i)U(i)}{\sum_{i=0}^{n} Z_2^l(i)}, \tag{9}$$

where Z_2^l denotes the matrix with linear reflectivities at t_2, and the sum is performed over all radar bins (n in total) that are spanned by the cell with ID j. The description is again valid for V_2^c, but the same proceeding is followed for calculating the velocities in $V_{1,2}^c$.

$V_{1,2}^c$ now contains the cell velocities that are given as output by CBVELOCITY, while $V_2^{s(c)}$ becomes $V_1^{s(c)}$ in the next iteration of the tracking algorithm. It is not $V_{1,2}^{s(c)}$ that becomes $V_1^{s(c)}$ in the next iteration, because this would cause old velocities to get an increasingly large influence at the determination of new cell and subcell velocities.
Examples of the velocities given as output by the algorithm are shown in figures 6, 7 and 8.

5 A method for evaluating the performance

The performance of CBVELOCITY is evaluated by comparing forecast cell centroid positions to observed ones. However, CBVELOCITY does not directly associate cells, only subcells, and subcells from one cell can be associated to subcells from multiple other cells. A method to determine cell associations is therefore required, and the method used is the following:

5.1 Determining cell associations

For a cell with ID i at time t_1 it is checked with which cells at time t_2 its subcells get associated, i.e. to which cells the subcells at t_2 belong with which its own subcells get associated. If at least 90 % of the subcells that get associated by the Jonker-Volgenant algorithm is associated with subcells from only one cell, then this cell is regarded as a potential continuation of cell i. The ID of this cell at t_2 is denoted by j.
Next it is checked with which cells at t_1 the subcells of cell j get associated. If now at least 90 % of the subcells that gets associated by the Jonker-Volgenant algorithm is from cell i, then cell j is still regarded as a potential continuation of cell i. The reason for using the two-way condition is that it prevents that very small cells get associated to very large cells, because for these small cells one of the one-way conditions is easily satisfied. Such an associations would lead to large jumps in cell centroid positions, with accompanying large errors in forecast positions, which are not caused by incorrect velocity estimates by the algorithm.

Thus far however, only the subcells that get associated by the Jonker-Volgenant algorithm have been taken into account, which do not have to constitute the whole cell. An extra condition included in the method is therefore that at least 50 % of the subcells from cell i and j must be associated by the Jonker-Volgenant algorithm.

5.2 Determining performance

Using the above method it is possible to associate cells, and these can be tracked into time for as long as all conditions remain satisfied. It is therefore possible to compare forecast cell centroid positions (weighted by linear reflectivity) to observed ones, for forecast lead times dependent on the time interval over which a cell can be followed. This has been done for 42 sequences with convection near or over the Netherlands, during the period of 2008-2016. These sequences spanned a total time of 587 hours.

The radars from which data was used were located in De Bilt and Den Helder, for which characteristics of the scanning scheme are given in table 1. In the calculation of Z_{max}, the bottom scan was excluded, primarily because of the amount of clutter that its data contained.

A total of 18507 cells satisfied the conditions in the above method for at least one pair of consecutive times. For time spans of 5, 10, 15, 30, 45 and 60 minutes, the number of cells that was followed for this amount of time is given in the second column of table 2. The third column of this table gives the number of cell chains available for each time span, where a cell chain is defined to be a sequence of detections of the same cell at consecutive times, over a time range given by the time span. The number of cell chains that spans a time of $5m$ minutes (and not longer), $N^s(m)$, is therefore given by

$$N^s(m) = \sum_{i=m}^{n} (i - m + 1) N^c(i), \tag{10}$$

where $N^c(i)$ denotes the number of cells that was followed for a time span of $5i$ minutes (and not longer).

For each cell chain of length m, the forecast position of the cell centroid was compared with the observed one, valid $5m$ minutes later. The forecast position was calculated by means of linear extrapolation, using the storm motion vector determined by CBVELOCITY at the start of the cell chain. For each pair of forecast and observed centroid positions, the forecast error was calculated, which is defined as the norm of the difference in position ($||\Delta \boldsymbol{r}||$). Also calculated was the angle (θ) between the observed translation vector (during the time span of the cell chain) and the forecast translation vector, which points in the direction of the storm motion vector.
The distributions of $||\Delta \boldsymbol{r}||$ and θ are shown in figures 9a and 9b, for forecast lead times of 5, 10, 15, 30, 45 and 60 minutes. The densities are normalized in such a way that they integrate up to 1. The average values of $||\Delta \boldsymbol{r}||$ and θ are shown in the fourth and fif5h column of table 2, for the same lead times.
Figure 9c shows the distribution of the storm speeds as determined by CBVELOCITY, which for a cell chain of length m is calculated as

$$||\boldsymbol{V}|| = \frac{\sum_{i=1}^{m+1} \boldsymbol{V}(i)}{m+1}, \tag{11}$$

i.e. as an average over all storm motion estimates available for the cell during the time span of the cell chain. $\boldsymbol{V}(i)$ denotes the storm motion determined by CBVELOCITY for time t_i, and the average is over $m+1$ velocities instead of m, because a cell chain that spans $5m$ minutes contains $m+1$ cell detections.

Other interesting aspects of the performance to consider are the distribution of θ as a function of $||\Delta\boldsymbol{r}||$, and the distributions of θ and $||\Delta\boldsymbol{r}||$ as a function of $||\boldsymbol{V}||$. It might e.g. be the case that the performance of CBVELOCITY depends on the storm speed.
These distributions are shown in the scatter plots of figure 10. In these plots, the color represents the normalized densities. They have been determined by dividing the graph up into bins with equal dimensions, in which the number of points is counted, and normalized such that it integrates up to 1. Also shown in these plots is the mean value of the parameter on the y axis as a function of the parameter on the x axis. This mean is represented by the black line.

Table 1: Specifications of the scanning scheme under which the radars in De Bilt and Den Helder were operated in the period of 2008-2016.

Parameters	**Values**
Scan elevations (°)	0.3, 0.4, 0.8, 1.1, 2 , 3, 4.5, 6, 8, 10, 12, 15, 20, 25
Radial resolutions (km)	1 for lowest 6 scans, 0.5 for higher scans
Radial slant ranges (km)	320, 240, 240, 240, 240, 170, 170, 150, 150, 120, 120, 120, 120, 120
Volume completion time	5 minutes

Table 2: Second and third column: The number of cells and number of cell chains as a function of the time span over which a cell/cell chain was followed. Fourth and fifth column: The average forecast error and average deviation in direction as a function of the forecast lead time (represented by the time span).

Time span (min)	**# of cells**	**# of cell chains**	**Average $||\Delta\boldsymbol{r}||$ (km)**	**Average θ (°)**
5	9565	41902	1.25	17.4
10	3913	23407	2.10	15.7
15	1978	14467	2.92	14.8
30	411	4388	5.28	13.3
45	118	1529	7.58	12.7
60	40	591	10.14	12.8

Table 3: The number of cells and average forecast error as a function of the lead time for SCIT (Johnson et al. 1998, table 6).

Lead time (min)	**# of cells**	**Average $||\Delta\boldsymbol{r}||$ (km)**
5	898	2.0
15	498	5.0
30	227	9.9
45	109	15.2
60	55	22.8

6 Evaluation of performance and discussion

For thunderstorm nowcasting, two aspects are very important; the direction of motion, determining which areas get hit by the storm, and the time of arrival along the path of the thunderstorm. It is therefore important to evaluate the performance of CBVELOCITY on these aspects.

The performance on the direction of motion is determined by the deviation in the direction (θ), whereas the performance on the time of arrival is determined by the error in the estimated storm speed. This deviation in storm speed has however not been determined directly, only the forecast error ($||\Delta\boldsymbol{r}||$). It might however be assumed that for a given value of $||\Delta\boldsymbol{r}||$, the estimated storm motion vector is distributed uniformly over a circle with radius $||\Delta\boldsymbol{r}||$, centered around the tip of the 'true' storm motion vector. This assumption appears reasonable, as no systematic biases in the estimated storm motion vector are known. Under this assumption, the distribution of the error in the storm speed follows from the distribution of $||\Delta\boldsymbol{r}||$.

In order to evaluate the performance of CBVELOCITY on these aspects, first a comparison with the algorithms mentioned in section 2, where possible. Secondly, the performance of CBVELOCITY as a function of the storm speed is evaluated, showing interesting differences in accuracy for low versus high storm speeds. This leads to ideas about the usefulness of CBVELOCITY for thunderstorm nowcasting as a function of the storm speed. Finally, the processes that appear to contribute most to forecast errors are discussed.

6.1 Comparison with previous algorithms

As indicated in table 2, the average forecast error increases from 1.25 km for a forecast lead time of 5 minutes, to 10.14 km for a lead time of 60 minutes. It is now attempted to compare these results to those of the algorithms treated in section 2. However, for only one of these algorithms were forecast errors determined, which is SCIT. This prevents a comparison with the other algorithms, as no attempt has been done to reproduce those algorithms, and applying them to the same set of convective situations. Also, SCIT has been tested for different convective situations, further complicating a comparison. Further, the methods used for determining the positions of storm cell centroids are not equal. Finally, (Johnson et al. 1998) examine the performance of SCIT for all storms, whereas the method used here excludes storms for which a one-to-one association of storms appears less reasonable, i.e. most cases with storm interactions. The performance of SCIT would likely improve when excluding these cases too, as these appear to be the type of storms that cause the largest forecast error. Nevertheless, these cases with storm interactions must also be treated by the algorithm, and it might be expected that the method used by CBVELOCITY treats them better, because it does not perform associations of complete storm cells.

Taking the preceding considerations into account, a comparison of the average forecast error for SCIT and CBVELOCITY is performed. For CBVELOCITY these are shown in column 4 of table 2, and for SCIT they are shown in table 3 (Johnson et al. 1998, table 6). It is clear that the forecast errors for SCIT are significantly higher that those for CBVELOCITY, for all lead times. Keeping the preceding considerations into mind, it does seem that CBVELOCITY produces on average better storm motion estimates. The lack of a fair comparison prevents making robuster conclusions however.

6.2 Performance as a function of storm speed

An evaluation of the performance of CBVELOCITY as a function of the storm speed is based on the results in figure 10. Figures 10C) and 10D) show that the average θ increases substantially with decreasing storm speed, as might logically be expected. Comparison of 10C) and 10D) further shows that the average θ is smaller for a lead time of 15 minutes than for 5 minutes. This is believed to be the case because the situations in which slow-moving storms develop are often situations with little wind shear. Such a small amount of wind shear leads to short-lived storms (AMS glossary of meteorology, 'ordinary cell'), or in the case of new development on the outflow of existing storms, often to storms that are in close proximity to each other. In such a case of new development, the storms will likely not satisfy the conditions for the cell association method.

Figures 10E) and 10F) show that the forecast error is barely dependent on the storm speed, for both a lead time of 5 minutes and 15 minutes. Under the assumption made about the distribution of the deviation in storm speed, it follows that the error in the forecast storm speed is also barely dependent on the storm speed. The effect of an error in storm speed at the time of arrival for a thunderstorm is however dependent on the relative size of this error, which is the ratio of the error in storm speed and the storm speed itself. If the error in the forecast storm speed is barely dependent on the storm speed, this implies that the relative error grows with decreasing storm speed, as will the error in time of arrival do.

Both the average error in the direction of motion and the error in the time of arrival increase therefore with decreasing storm speed. This information can be combined with the idea mentioned above, i.e. that slow-moving storms often occur in situations with little wind shear, leading to shorter-lived storms. The shorter the life time of storms, the more important the growth and decay of storms become, which are not treated by CBVELOCITY. This also decreases the nowcasting ability of CBVELOCITY in many situations with slow-moving storms. These three insights lead to the conclusion that CBVELOCITY should not be used for nowcasting in situations with slow-moving storms. The speed threshold used for defining when a storm moves 'too slow' should thereby depend on the desired accuracy, and could be based on figures 10C) and 10D). An example of the output for a case with slow-moving storms is shown in figure 8, where the estimated storm motions are clearly erratic.

Finally, the three processes that are responsible for the decrease in performance (on average) with decreasing storm speed do not appear to be specific to CBVELOCITY, although the growth and decay of thunderstorms is taken into account by some algorithms, e.g. TITAN. This decrease in performance (regarding the direction of motion and time of arrival) with decreasing storm speed might therefore be more generally valid for storm-tracking algorithms.

6.3 Processes responsible for forecast errors

Several processes can be responsible for forecast errors, and the most important ones are listed here:

1. Storms will evolve during the time between consecutive radar scans, implying that the shape of a storm and its number of subcells can change, hampering the association of subcells. In addition to this, the number of storms can change due to the development of new ones and the decay of old ones. While subcell associations are only allowed when they lead to a change in velocity and reflectivity that is less than MAX_DV and MAX_DR, this still allows for incorrect associations. An important source for these incorrect associations is the development of a new cell close to an existing one, which can lead to the association of some subcells from the existing storm with subcells from the new one, giving an incorrect velocity estimate for the new one. The number of occurrences of this problem could be reduced by decreasing MAX_DV, but this also reduces the ability of CBVELOCITY to detect true changes in storm motion, or to correct for wrong initial velocity estimates for new cells.

2. The motion of thunderstorms can deviate from linearity, which could become important mostly for greater forecast lead times. A possible source of this, somewhat related to point 1, is the development of a new thunderstorm updraft on the flank of an existing cell, which occurs so close to the existing cell that CBVELOCITY identifies it as one storm. This is called propagation, and influences the position of the storm centroid.

3. There is a 'discretization error', which arises because the reflectivity is not available as a continuous field, but on a polar grid. This implies that only certain steps in position are allowed for subcells, corresponding to certain velocities.

4. Attenuation of the reflectivity due to the presence of areas with a high precipitation intensity between the radar and a storm of interest is not directly responsible for errors in the estimated storm motion. It could however have as a result that CBVELOCITY misses the storm, because its reflectivity does not exceed the minimum reflectivity thresholds.

7 Conclusion

A new storm-tracking algorithm has been created and named CBVELOCITY, which stands for cell-based velocity. It has been created with as goal to improve the treatment of thunderstorm interactions compared to existing algorithms, without the use of separate methods for the treatment of storm splittings and mergers. In CBVELOCITY, the translation of thunderstorms and storm interactions are all treated in a uniform way, by means of the subcell approach.

Storms are divided into subcells, which are associated by means of combinatorial optimization using the Jonker-Volgenant algorithm, whereafter storm motions are determined as the average over the subcell velocities. Because subcells of one storm can be associated to subcells of multiple other storms, storm interactions are automatically treated. The subcell approach is based on the particle representation by Matthews and Trostel (2010), but is optimized for estimating storm motions, instead of the tracking (following) of storms through time.

The performance of CBVELOCITY has been evaluated by comparing forecast centroid positions to observed ones. This has been done for 42 sequences with convection near or over the Netherlands, giving a total of 18507 cells that were followed for at least one time interval. It was found that the average forecast error, which is the norm of the deviation in position, increases from 1.25 km for a forecast lead time of 5 minutes, to 10.14 km for a lead time of 60 minutes. This has been compared with the results for SCIT (Johnson et al. 1998), which is the only algorithm for which the same forecast error was determined. For SCIT the forecast errors ranged from 2.0 km for a lead time of 5 minutes to 22.8 km for a lead time of 60 minutes. Although these results are encouraging, differences in the storm sample and sampling technique preclude a robust conclusion about the performance of CBVELOCITY compared to SCIT.

Also evaluated was the performance of CBVELOCITY as a function of the storm speed. It was found that the error in the direction of motion and the error in the time of arrival of a storm both increase with decreasing storm speed. Combined with the usually shorter life times for slow-moving storms, because short-lived storms often occur in areas with little wind shear, this leads to the conclusion that the performance of CBVELOCITY decreases substantially with decreasing storm speed. The processes responsible for this are not specific to CBVELOCITY however, and other storm-tracking algorithms will suffer from them too.

There remain possibilities to improve the storm motion estimates, and the following subsection lists work that might be performed in an attempt to improve the results, or to make CBVELOCITY more suitable for certain tasks.

7.1 Possible improvements

1. CBVELOCITY produces currently only storm motion estimates, and lacks a method by which the storms are translated by the storm motion vector, as to produce forecast positions. If it would be used for automatic generation of forecasts for future storm positions, such a method is required. This could be done by means of linear extrapolation of the current storm positions, using the estimated storm motion vectors.

2. In its current form, CBVELOCITY suffers from bright band contamination, that leads to the detection of large areas of stratiform precipitation. This is because the reflectivity often exceeds 40 dBZ in the bright band, which is above the default value of MIN_MAXIMUM_DBZ. Depending on the applications, it might be the case that tracking of stratiform precipitation is not desired. If this is the case, then the tracking of stratiform precipitation is a waist of computation time, and a search for a method to correct for the bright band contamination is desired. If less stratiform precipitation gets detected, then it is possible to decrease DESIRED_1D_SPAN_SUBCELLS, leading to smaller subcells. This will reduce the discretization error, and might therefore improve the accuracy.

3. No systematic optimization of parameter values has yet been performed. The current default values are based on the results of a visual inspection of the output, and not an examination of the forecast error, which was used to evaluate the performance. Performing such a systematic optimization could improve the performance, although large improvements appear unlikely based on experience with changing parameter values.

Appendices

A Description and default values for the algorithm parameters

The default values given in this appendix are values that have been found to work good for the C-band radars operated by the KNMI in De Bilt and Den Helder (The Netherlands). This has been done by visual inspection of the output generated by CBVELOCITY. A systematic approach for finding the optimum parameter values has not yet been used. The parameter values might require changes when applying the algorithm to other radars.

1. MIN_DBZ_THRESHOLD:
   The minimum dBZ threshold for which clusters are identified by the *Clustering* algorithm.
   Default: 35 dBZ.

2. MAX_DBZ_THRESHOLD:
   As above, but the maximum dBZ threshold for which clusters are identified.
   Default: 60 dBZ.

3. STEP_DBZ_THRESHOLD:
   Starting at MIN_DBZ_THRESHOLD and continuing until MAX_DBZ_THRESHOLD, clusters are identified for thresholds that differ by steps of STEP_DBZ_THRESHOLD.
   Default: 1 dBZ.

4. MAX_DBZDIFF_COMBINING_CLUSTERS:
   Is explained in the *Clusters combining* algorithm, and is used when a cluster for a threshold T^{n-1} encompasses more than one cluster as identified for the next higher threshold T^n. It determines the number of cells that is generated, as a smaller (larger) value makes it more (less) likely that the algorithm will break a cluster up into different cells.
   Default: Depends on the distance to the radar according to MAX_DBZDIFF_COMBINING_CLUSTERS= $m(r) = \frac{\alpha_0}{1+\frac{r}{r_1}(\frac{\alpha_0}{\alpha_1}-1)}$, with $\alpha_0 = m(r=0)$, $\alpha_1 = m(r=r_1)$, and r_1 is the radius at which m should equal α_1. This formula can be rewritten as $m(r) = \frac{a}{b+r}$, where $a = \frac{\alpha_0 r_1}{\alpha_0/\alpha_1 - 1}$ and $b = \frac{a}{\alpha_0}$.

5. MIN_MAXIMUM_DBZ:
   Cells for which the maximum reflectivity is less than this value are discarded.
   Default: 40 dBZ.

6. CELLS_MINIMUM_AREA:
   Cells that are identified by the cell identification algorithm, but which span an area less than this, are discarded.
   Default: 3 km^2.

7. CELLS_MINIMUM_NUMBEROFBINS:
   As CELLS_MINIMUM_AREA, but now the minimum number of radar bins that must be occupied by a cell.
   Default: 2.

8. DESIRED_1D_SPAN_SUBCELLS:
   In the division of cells into subcells it is attempted to obtain subcells whose shape is reasonable close to square, despite the use of a polar grid. This parameter represents the desired length of the sides of the square. It influences the amount of subcells generated by the *Divide cells into subcells* algorithm, which in turn influences the computation time.
   Default: 2 km.

9. VELOCITY_ESTIMATE_NEWCELLS:
   Velocity estimate used to initialize the tracking algorithm, and also used by the *Estimating velocity unassigned subcells* algorithm, when there are not enough neighbouring subcells and cells to estimate the velocity of an unassigned subcell from.

10. WEIGHT_V:
    Weight of TV in the calculation of the cost matrix T.
    Default: 0.2.

11. WEIGHT_R:
    As WEIGHT_V, but for TR.
    Default: 0.8. This value is set much higher than that of WEIGHT_R, based on the idea that it is more important to associate subcells in such a way that their reflectivities differ by not too much, than it is to associate them in such a way that the new velocity does not differ by much from the previous velocity.

12. MAX_DV:
    Maximum allowed norm of the change in velocity of a subcell between subsequent times. Associations of subcells for which this value is exceeded, are not allowed.
    Default: Dependent on the radar bin dimensions (and therefore on the distance to the radar) according to MAX_DV= $v(r) = \sqrt{v_0^2 + v_1(r)^2}$, where $v_1(r) = \frac{1}{\Delta t^2}\max\{(r\Delta\phi)^2 + \Delta r^2, (1000 \cdot \text{DESIRED\_1D\_SPAN\_SUBCELLS})^2\}$. Here, $v_0 = v(r = 0)$, with default value $v_0 = 10$ m/s. $v_1(r)$ gives approximately the velocity required to go from one subcell to another subcell, located one subcell position below/above and one to the left/right of it, assuming square subcells. Inclusion of this term ensures that it is possible for a subcell to make such a step. The use of the square root in the relation for $v(r)$ gives a smooth transition between domination of $v(r)$ by v_0, and domination by $v_1(r)$.

13. MAX_DR:
    Maximum allowed difference in subcell averaged reflectivity between subsequent volumes.
    Default: 20 dBZ.

14. MAX_DV_SUBCELL_CELL:
    An association of subcell i at time 1 and subcell j at time 2 is only allowed when the difference between the new subcell velocity (following from the association) does not differ by more than this value from the velocity of the cell to which subcell i belongs (at time 1).
    Default: Equal to MAX_DV.

15. SUBCELLVELOCITIES_AVERAGING_NTIMES:
    In determining $V_{1,2}^{c(s)}$ in the tracking algorithm, velocities of subcells that have been associated by the Jonker-Volgenant algorithmm are calculated as the average over the last SUBCELLVELOCITIES_AVERAGING_NTIMES velocities for that subcell, including the new subcell velocity in V_2^s.
    Default: 2.

16. UNASSIGNEDSUBCELLS_NREQUIRED_NEIGHBOURING_SUBCELLVELOCITIES:
    Used in the *Estimating velocity unassigned subcells* algorithm. When estimating the velocity of an unassociated subcell, this is the amount of nearest subcell velocities in the same cell over which the average is calculated. If there are not enough subcells, then the algorithm looks for neighbouring cells, or takes VELOCITY_ESTIMATE_NEWCELLS into account.
    Default: 10.

17. UNASSIGNEDSUBCELLS_MAXSEARCHRANGE_CELLVELOCITIES:
    If there are not enough subcells as described above, then this is the range in which for neighbouring cells is searched.
    Default: 100 km.

18. UNASSIGNEDSUBCELLS_NREQUIRED_NEIGHBOURING_CELLVELOCITIES:
    The number of neighbouring cells that is required to average over. Can include cells at previous times, as described in the *Estimating velocity unassigned subcells* algorithm.
    Default: 10.

19. TIMERANGE_NEIGHBOURING_CELLS:
    The time range over which for neighbouring cells is searched, as described in the *Estimating velocity unassigned subcells* algorithm.
    Default: 15 minutes.

References

[1] Dixon M. and Wiener G., 1993: TITAN: Thunderstorm identification, tracking, analysis and nowcasting - A radar-based methodology. *J. Atmos. Oceanic Technol.*, **10**(6), 785-797.

[2] Doviak, R., and D. Zrnić, 1993: Doppler Radar and Weather Observations. 2nd ed. Academic Press, 562 pp.

[3] French, M.N., Krajewski, W.F. and Cuykendall, R.R., 1992: Rainfall forecasting in space and time using a neural network. *J. Hydrol.*, 137: 1-31.

[4] 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. *J. Atmos. Oceanic Technol.*, **26**(4), pp. 719-732.

[5] Handwerker, J., 2002: Cell tracking with TRACE3 - A new algorithm. *Atmos. Res.*, **61**, 15-34.

[6] Johnson J.T., MacKeen P.L., Witt A., Mitchell E.D., Stumpf G.J., Eilts M.D. and Thomas K.W., 1998. The storm cell identification and tracking algorithm: an enhanced WSR-88D algorithm. *Weather Forecast.* **13**: 263-276.

[7] Jonker R. and Volgenant A., 1987: A shortest augmenting path algorithm for dense and spare linear assignment problems, *Computing*, **38**, 325-340.

[8] Kazmar T., 2012-2017: python-lapjv - LAPJV/LAPMOD implementation. Version from May 10, 2017. `https://github.com/gatagat/lapjv`.

[9] Limpert G., Houston G., and Lock N., 2015: The advanced algorithm for tracking objects (AALTO), *Meteorol. Appl.*, **22**, 694704.

[10] Martin, E., Kriegal H.P., Sander, J., 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*, Portland, WA, 1996, pp. 226-231.

[11] Mathews, J., Trostel, J.: An Improved Storm Cell Identification and Tracking (SCIT) Algorithm based DBSCAN Cluster and JPDA Tracking Methods, *Proceedings of the International Lightning Detection Conference*, Orlando, FL, USA, 19-20 April 2010.

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

[13] Wang, G. L., W. Wong, L. P. Liu, and H. Y. Wang, 2013: Application of multi-scale tracking radar echoes scheme in quantitative precipitation nowcasting. *Adv. Atmos. Sci.*, **30**(2), 448-460, doi: 10.1007/s00376-012-2026-7.

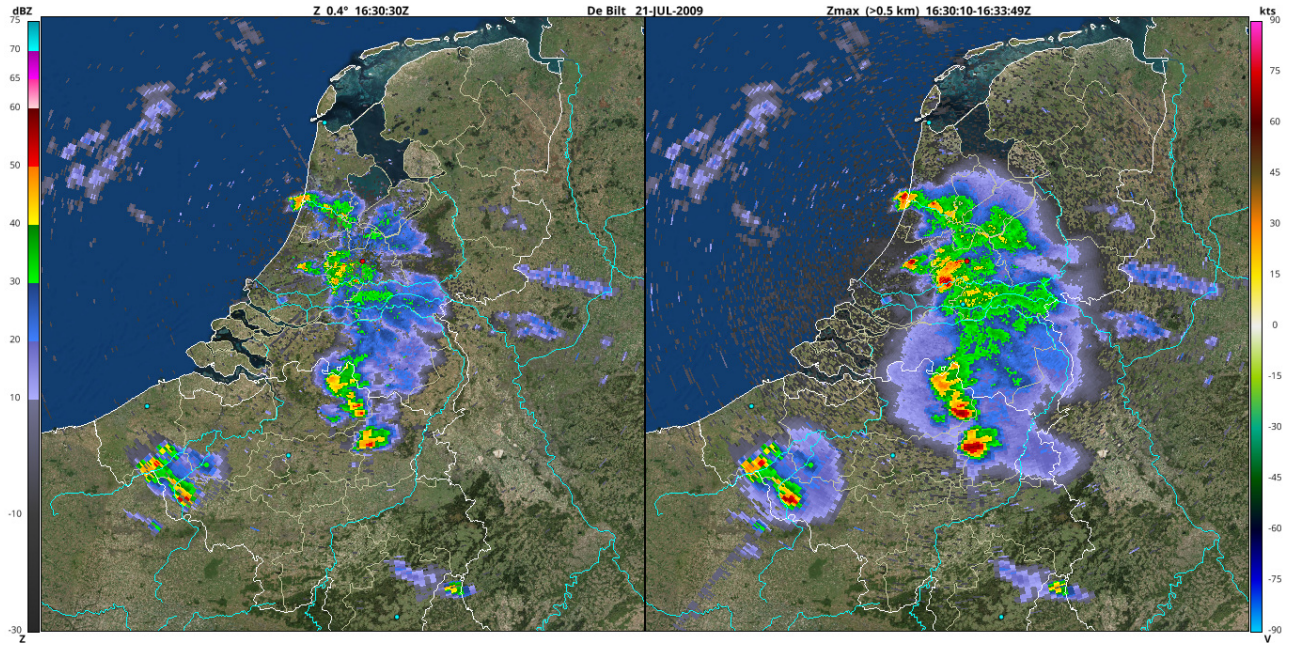


Figure 2: Z_{max} for a case with semi-discrete storms embedded in an area of stratiform precipitation, with the bright band signatures visible near the radar. The 0.4-degree scan is put to the left of it as a reference. The relevant legend is the one on the left.

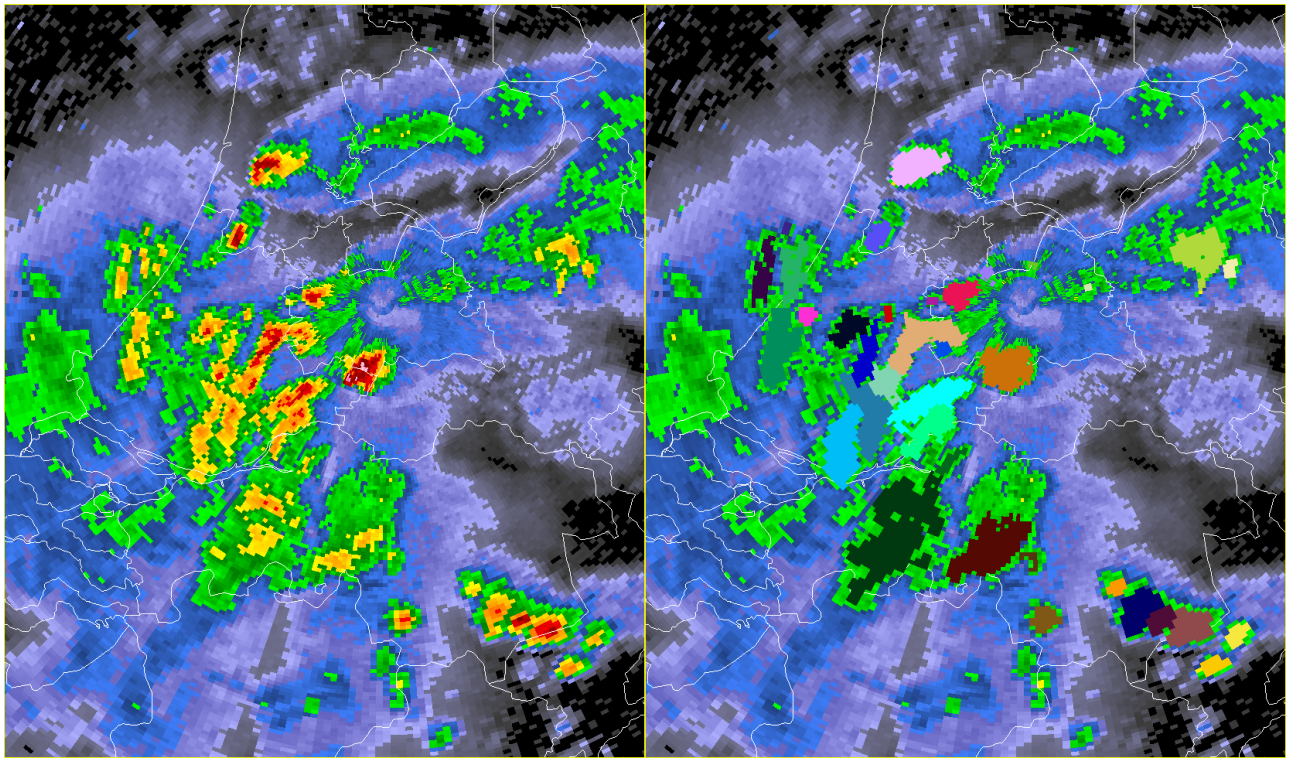


Figure 3: Example of identified cells, where each cell is given a different color. The left panel shows Z_{max} as a reference. The color scale is equal to that in figure 2.

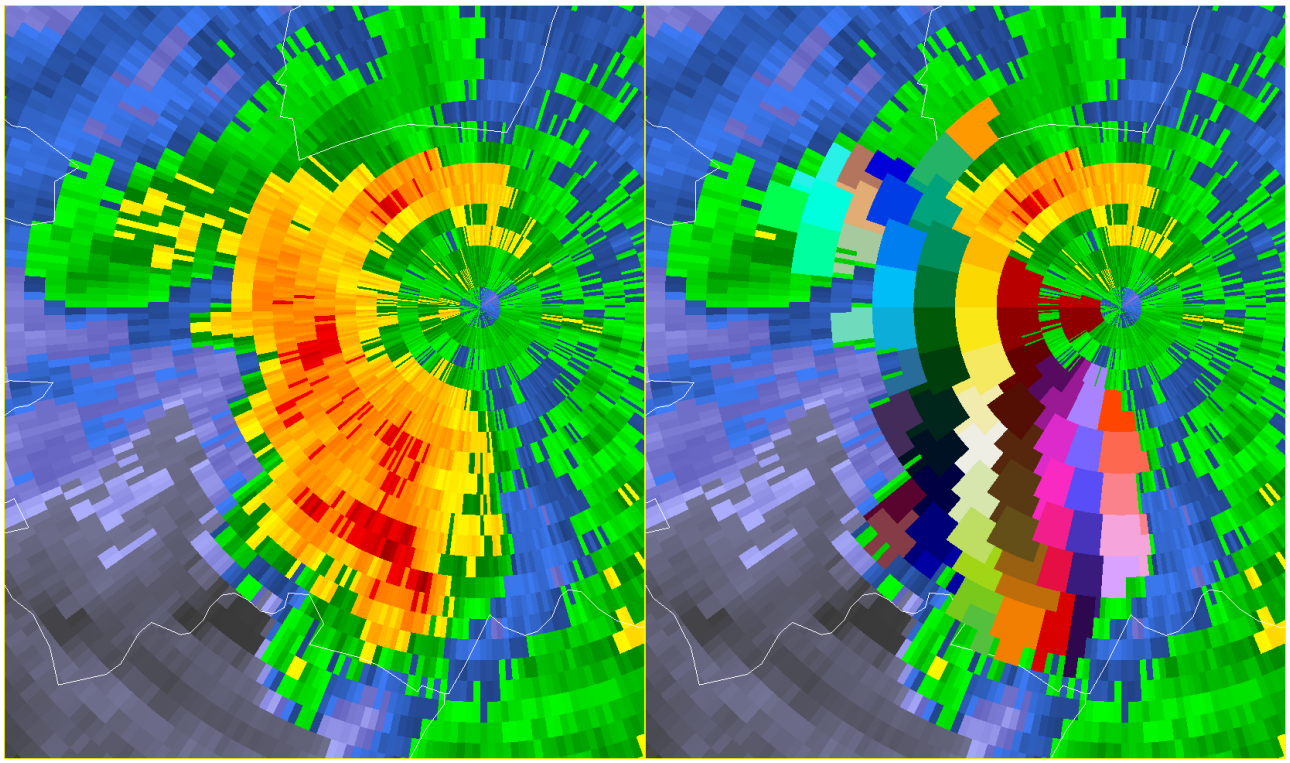


Figure 4: Identified subcells, marked with different colors, in a cell that is located near the radar.

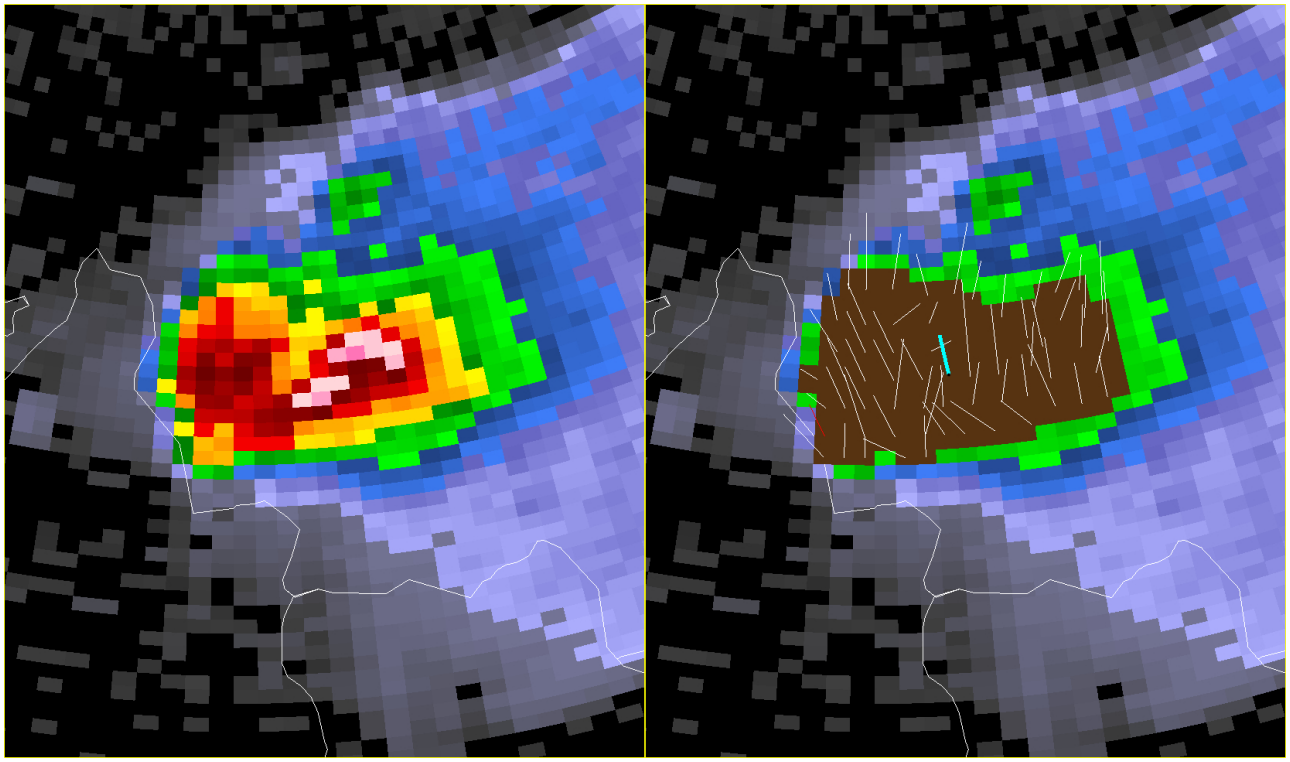


Figure 5: Subcell velocities (V_2^s) in a storm that is about to split. White lines represent velocities assigned by the Jonker-Volgenant algorithm. Red lines represent velocities assigned by the *Estimating velocity unassigned subcells* algorithm. The cyan thick line represents the estimated cell velocity ($V_{1,2}^c$).

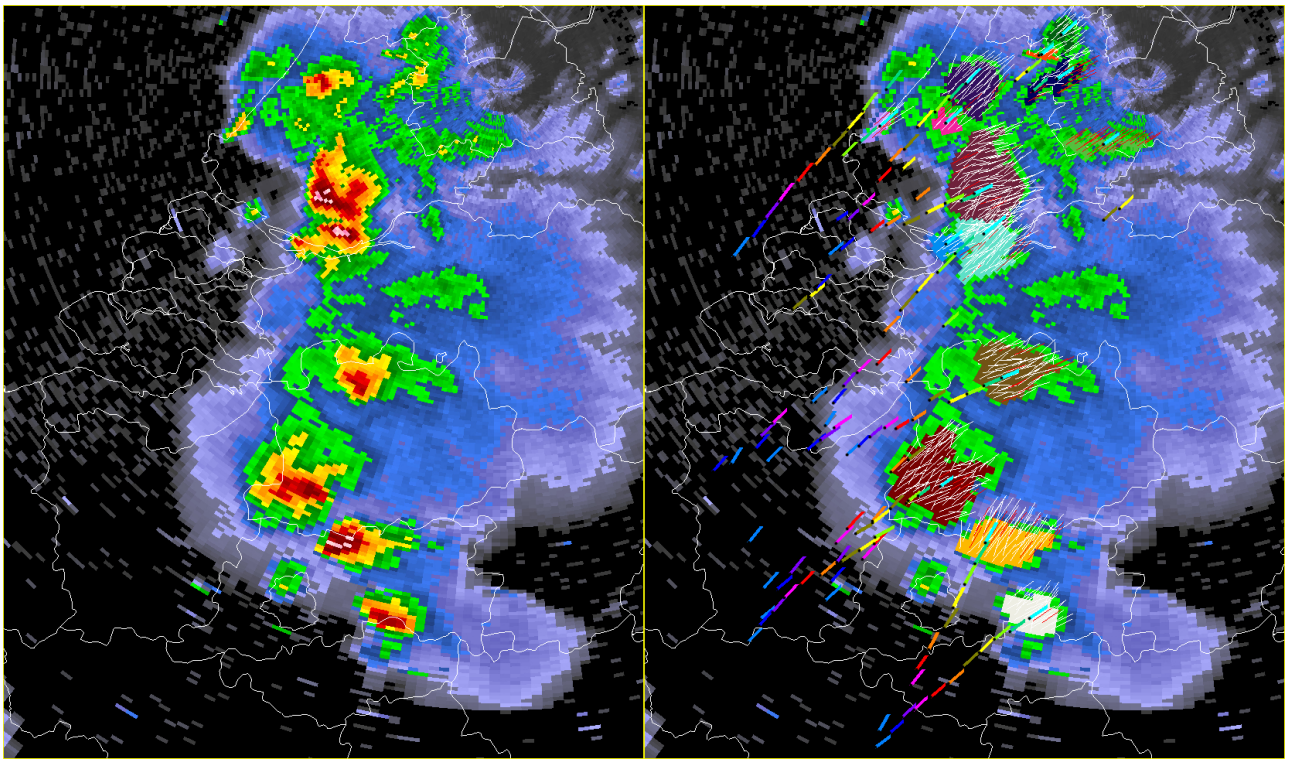


Figure 6: Estimated cell velocities for an event with several cell splits and supercells. The meaning of the thin white and red lines, and the blue thick line, is as in figure 5. The other thick lines represent cell velocities at 10 earlier times, 5 minutes apart. The southeasternmost cell was detected at all previous 10 times, and therefore shows which color corresponds to which previous time.

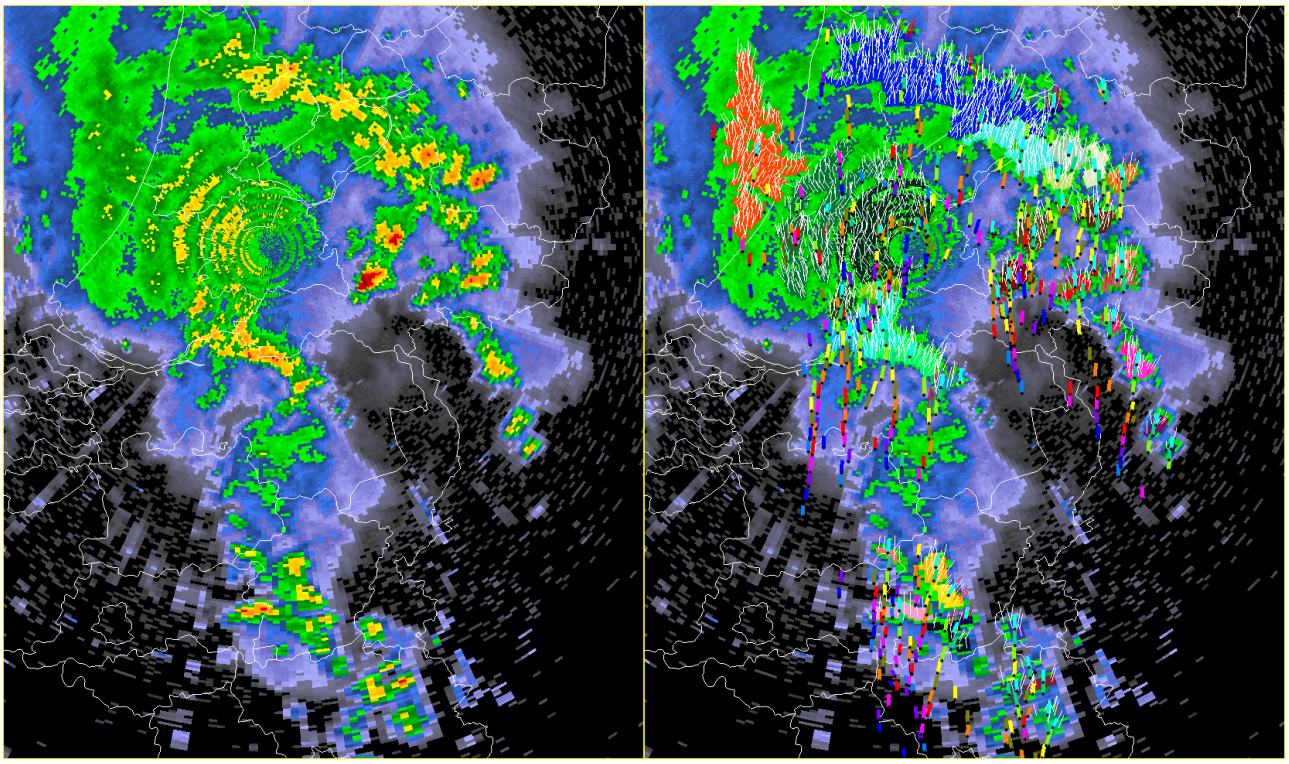


Figure 7: As figure 6, but for a situation in which a squall line passed over the Netherlands, with a few other lines of cells behind it.

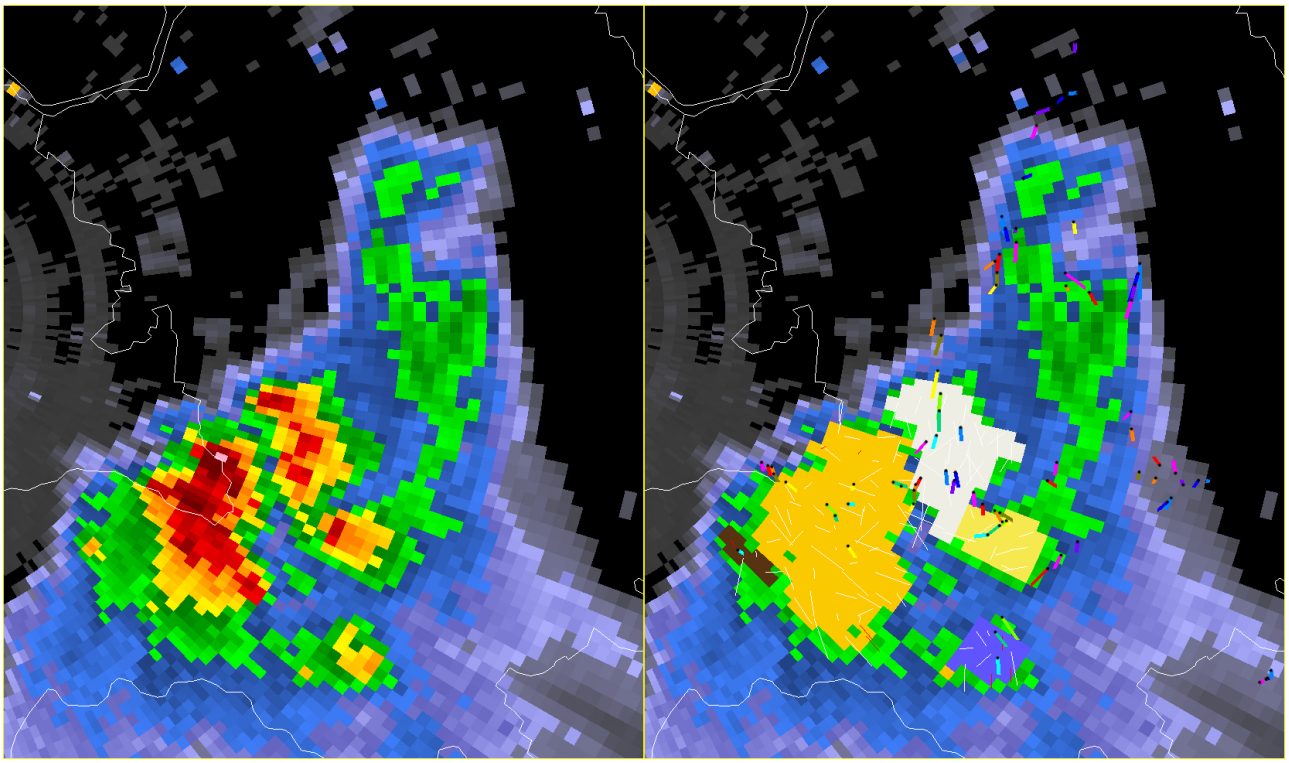


Figure 8: As figure 6, but for a situation with slow-moving thunderstorms.

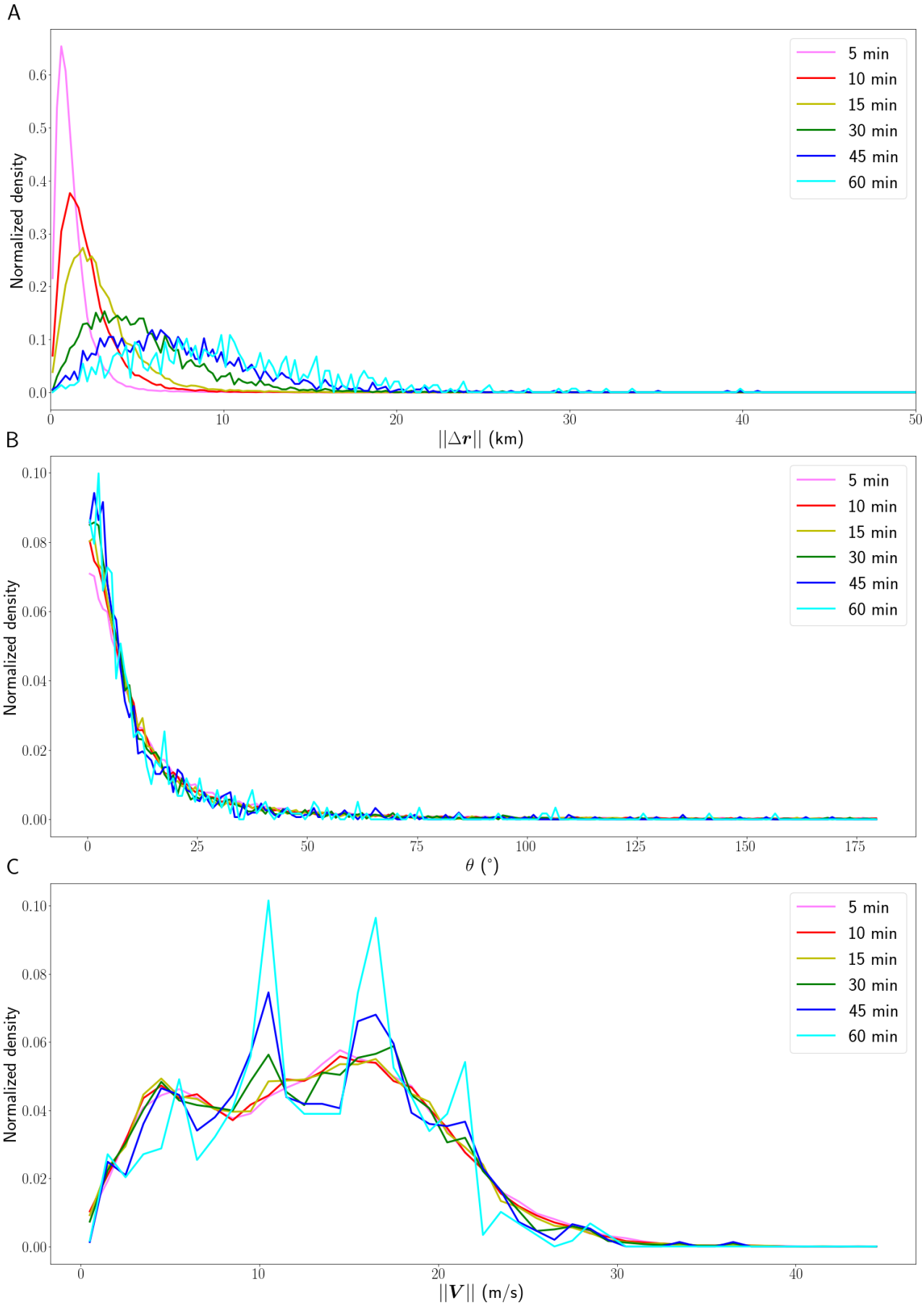


Figure 9: The distributions of A) the forecast error, and B) the deviation in direction for forecast lead times of 5, 15, 30, 45 and 60 minutes. In C), the distribution of the storm speed is shown as a function of the time spanned by a cell chain.

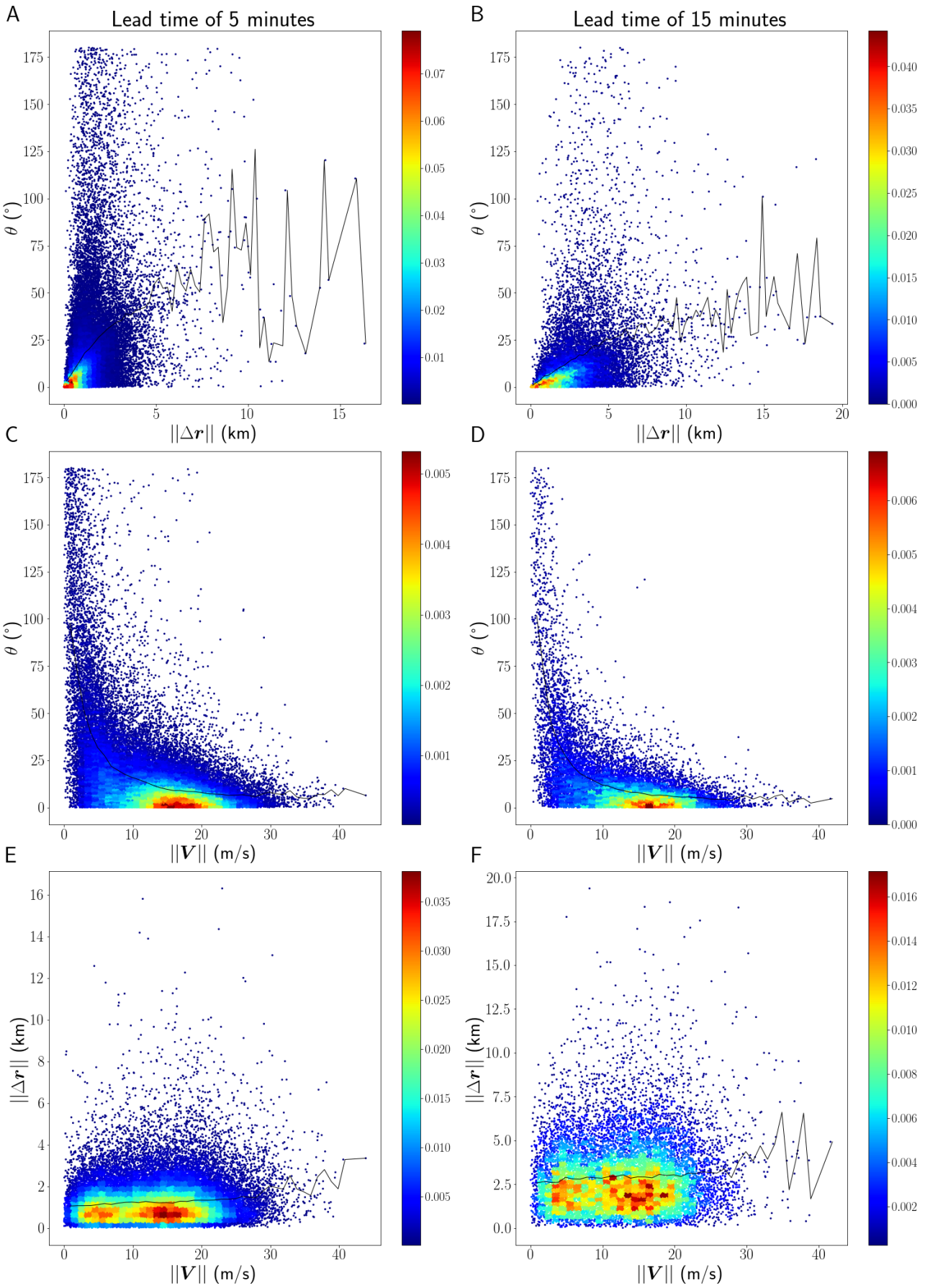


Figure 10: Scatter plots of A), B), the deviation in direction vs. the forecast error, C), D), the deviation in direction vs. the storm speed, E), F), the forecast error vs. the storm speed. A, C and D are for a lead time of 5 minutes, and B, D and F for a lead time of 15 minutes. The color represents the normalized density of points. The black line gives the mean of the parameter on the y axis as a function of the parameter on the x axis.