Method for Object-Based Diagnostic Evaluation

Randy Bullock
Barbara Brown
Tressa Fowler

NCAR Technical Notes
NCAR/TN-532+STR

National Center for
Atmospheric Research
P. O. Box 3000
Boulder, Colorado
80307-3000
www.ucar.edu

NCAR TECHNICAL NOTES

http://library.ucar.edu/research/publish-technote

The Technical Notes series provides an outlet for a variety of NCAR Manuscripts that contribute in specialized ways to the body of scientific knowledge but that are not yet at a point of a formal journal, monograph or book publication. Reports in this series are issued by the NCAR scientific divisions, serviced by OpenSky and operated through the NCAR Library. Designation symbols for the series include:

EDD – Engineering, Design, or Development Reports
Equipment descriptions, test results, instrumentation, and operating and maintenance manuals.

IA – Instructional Aids
Instruction manuals, bibliographies, film supplements, and other research or instructional aids.

PPR – Program Progress Reports
Field program reports, interim and working reports, survey reports, and plans for experiments.

PROC – Proceedings
Documentation or symposia, colloquia, conferences, workshops, and lectures. (Distribution maybe limited to attendees).

STR – Scientific and Technical Reports
Data compilations, theoretical and numerical investigations, and experimental results.

The National Center for Atmospheric Research (NCAR) is operated by the nonprofit University Corporation for Atmospheric Research (UCAR) under the sponsorship of the National Science Foundation. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

National Center for Atmospheric Research
P. O. Box 3000
Boulder, Colorado 80307

NCAR/TN-532+STR

NCAR TECHNICAL NOTE

2016-11

Method for Object-Based Diagnostic Evaluation

Randy Bullock

Joint Numerical Testbed

National Center for Atmospheric Research Boulder CO

Barbara Brown

Joint Numerical Testbed

National Center for Atmospheric Research Boulder CO

Tressa Fowler

Joint Numerical Testbed

National Center for Atmospheric Research Boulder CO

Joint Numerical Testbed

Research Applications Laboratory

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

P. O. Box 3000

BOULDER, COLORADO 80307-3000

ISSN Print Edition 2153-2397

ISSN Electronic Edition 2153-2400

Contents

List of Figures

List of Tables

Acknowledgements

The original MODE research group consisted of Dave Ahijevych, Dave Albo, Barb Brown, Randy Bullock, and Chris Davis. The authors would like to thank Dave Albo for teaching our group the basics of fuzzy logic.

Thanks are also due to Michelle Harrold for helping with collecting data and choosing cases for illustration.

John Halley Gotway took the original research version of MODE and incorporated it into MET.

Finally, our sincere thanks to the people who took time from their schedules to read draft versions of this document and give comments, helpful suggestions, and to correct errors. We especially thank Tara Jensen for doing the internal review of this document. Any mistakes which remain are the first author's responsibility.

This document was produced by TEX

1

Introduction

Dare to be naïve.

– R. Buckminster Fuller

1.1 Overview

This technical note discusses the Method for Object-Based Diagnostic Evaluation (MODE), which is part of the Model Evaluation Tools (MET) verification software package. MODE is an implementation of one approach to object-based verification (we'll discuss what this means shortly). Working with objects is more intuitive for most forecast users and providers than working with traditional verification statistics such as Probability of Detection (POD) or False Alarm Ratio (FAR). It allows statements about (for example) size, displacement and intensity errors that directly correspond to familiar properties of the forecast.

This first section gives an introduction to object-based forecast verification, which is a particular subtype of the general framework of spatial verification (see Brown *et al.*, 2011), and is the technique used by MODE. Section 2 details the algorithms MODE uses to resolve a gridded data field into objects. Human beings find this task very easy to do, but the method MODE uses is fairly simple, and reproduces what a human being would do in most cases. Section 3 examines object attributes. MODE calculates a great many attributes, such as location, size, intensity—we will only cover the most important ones. Besides being of interest in their own right, attributes are used by MODE to perform matching and merging of objects. The fuzzy logic engine that MODE uses to do this is examined in Section 4, while Section 5 shows how the engine uses object attributes to actually do the matching and merging. Section 6 looks at the effect of varying the MODE parameter settings and gives some guidelines on how to choose them. Section 7 shows how the MODE output files can be used. Section 8 looks at several possible ways that the general MODE methodology could be extended to other forecast verification situations. Finally, Section 9 gives a short history of MET and MODE and some concluding remarks.

For the reader's convenience, beginning on page 74 there is a list of mathematical symbols used in this document and a glossary beginning on page 76. The examples in this document will use the version of

MODE in the MET 5.1 release. The reader will find many additional descriptions and examples of MODE in the MET User's Guide.

We will assume that the reader has a basic familiarity with MODE: how to run it and how to interpret its output. This document will explain in detail how those results are produced. The reader who lacks this background should instead begin with the MODE chapter in the MET User's Guide.

1.2 Motivation: Beyond POD and FAR

To get a feel for the ideas behind object-based verification, consider Figure 1. In part (a) of the figure, we see a forecast and observation field on a small grid. The forecast field is red, and the observed field is blue. For the purposes of this example, the reader can think of the fields as precipitation. We ask the question: "Is this a good forecast?"

Traditional scores like POD and FAR, comparing forecasts and observations on a gridpoint-by-gridpoint basis, would answer "No." The calculated POD is zero, and the FAR is one. Other much-used verification scores, shown in the table in Figure 1, give a similar verdict. Yet there is a sense in which the forecast predicted many aspects of the storm correctly. The size, shape and orientation of the storm are correct. The location of the storm is not quite correct—there is a small displacement error to the right. Similarly, in part (b) of the figure, the size, shape and orientation of the storm are all forecast correctly, but the location error is larger. The forecast in part (c) has the size and shape correct, but has both position and orientation errors. In part (d), neither the size, shape, orientation or position of the storm is forecast correctly. Yet as the table in the figure shows, many widely-used verification measures assign the *same* scores to the forecasts in parts (a)–(d). These measures assign a positive degree of skill only to the forecast shown in part (e), which many people (*e.g.*, forecast users and developers) would consider the worst forecast of the lot.

This example illustrates the shortcomings of single-number scores such as POD and FAR for verification purposes. They simply don't give enough meaningful information, and in many cases they fail to distinguish between forecasts that have correctly predicted all but a few features of the overall field and those that have not.

It would be much more useful both to the model developer and to the end user of the forecast to have quantitative summaries of various spatial errors and biases in the forecast. Considerations such as these have given rise in recent years to the development of an alternative approach to verifying forecasts called *spatial verification*.

1.3 Spatial verification

The term *spatial verification* refers to any of a number of forecast verification strategies that attempt to make use of spatial information to help assess forecast quality. As shown in Figure 2, spatial verification methods fall into two broad categories: *filtering methods* and *displacement methods*.

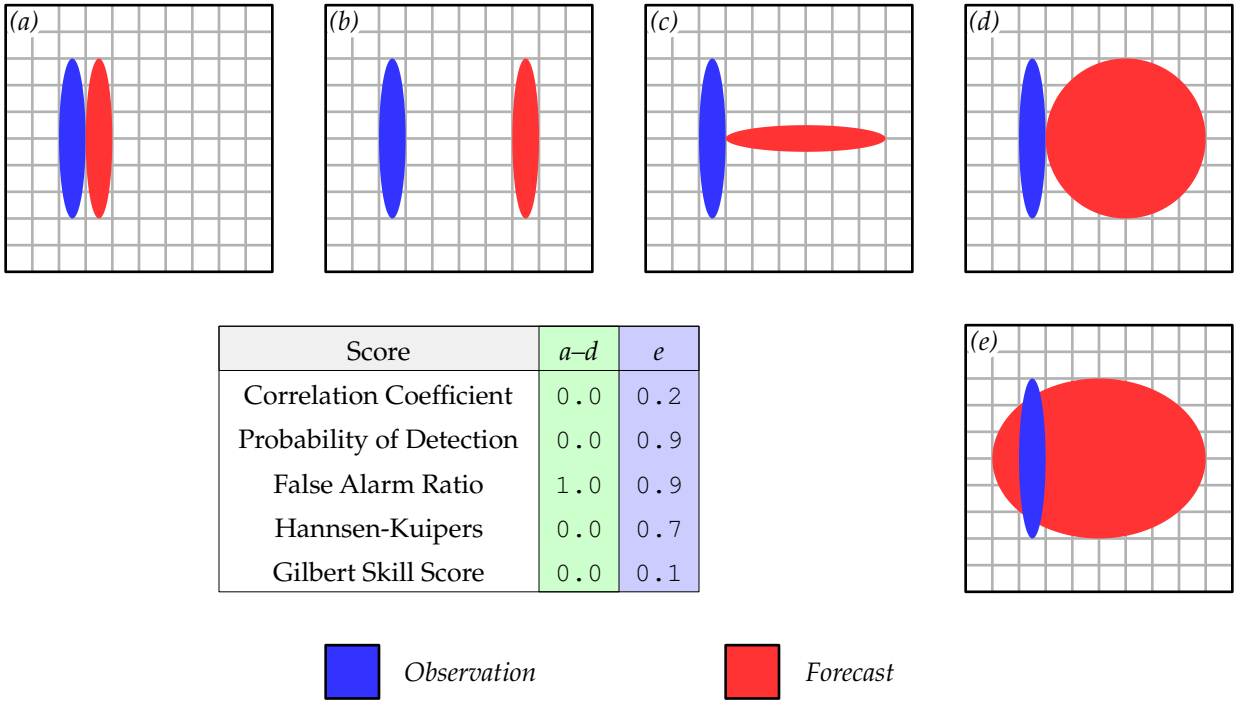


Figure 1 Motivation for object-based verification

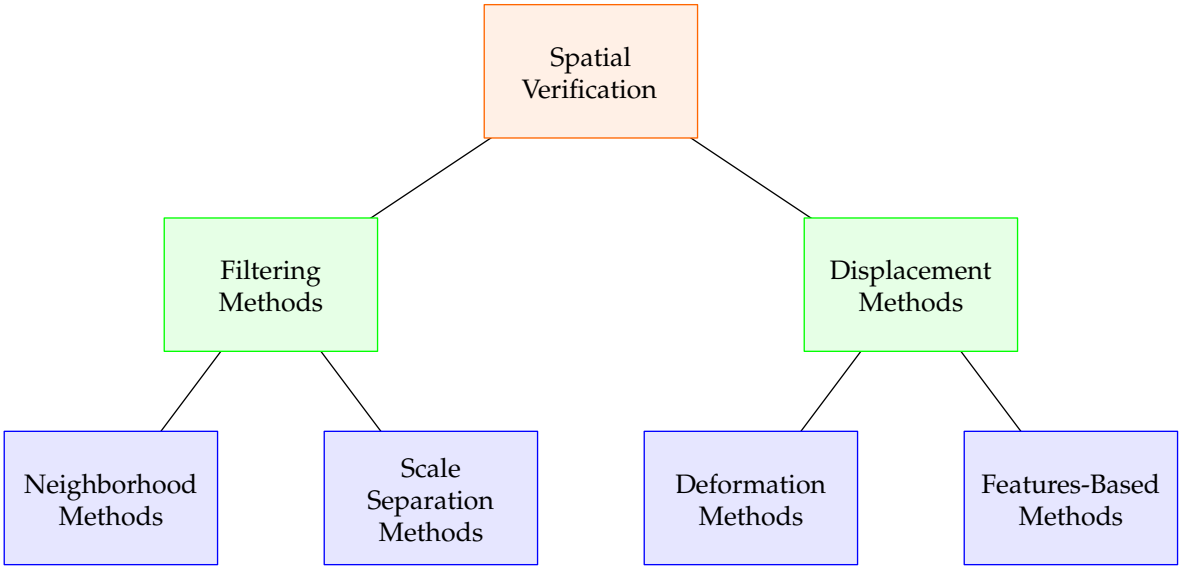


Figure 2 Main Categories of Spatial Verification

Filtering methods include neighborhood methods and scale separation methods. Neighborhood verification is an attempt to determine forecast skill at different spatial resolutions (*e.g.*, Ebert 2009). Instead of looking at each grid square individually, we combine the values in a box of some size (a *neighborhood*) centered at each grid point. One example of neighborhood methods is the Fractions Skill Score (Roberts and Lean 2008).

Displacement methods include field deformation such as field morphing, image warping and image metrics (*e.g.*, Gilleland *et al.* 2010b, Keil and Craig, 2007), and features-based approaches such as the contiguous rain area (CRA) method (Ebert and McBride 2001). Features-based methods try to determine how well the forecast captures spatial patterns. MODE is a features-based method. Overviews of the various methods are provided in Brown *et al.* (2011), Gilleland *et al.* (2009) and Gilleland *et al.* (2010a).

1.4 What are Objects?

Let's start out with what the word *objects* means in the context of forecast verification. Objects are spatial *regions of interest*. They are spatial features in the raw forecast or observation data that are interesting for whatever verification analysis is being performed. Probably the prototypical object is a rain area in a precipitation field. In fact, MODE is most commonly used to evaluate precipitation forecasts. A cloud is another obvious candidate for an object.

Figure 3 shows an example using precipitation fields. Part (a) of the figure shows a raw forecast field, while part (b) shows a raw observed field. The precise numerical values in the field are not important—just think of warm colors (orange, red) as being high values, and cool colors (green, blue) as low values. Parts (c) and (d) of the figure show one possible way of resolving the corresponding raw fields into objects. Notice that the objects roughly correspond to the areas of high precipitation that the eye picks out naturally. Once objects are in hand, they can be grouped together, as shown by the blue rubber-band shapes. This aspect of MODE will be discussed more thoroughly in the section on matching and merging (p. 51).

Fundamentally, MODE is used to analyze the spatial structure of forecast and observed gridded data fields, and to compare them to see how well the forecast captures the structure of the observations. Whatever spatial structure you're looking for in the data, the basic building blocks of that structure will be the *objects* for you.

1.5 Examples

Two examples of applications of MODE are presented in the following subsections. The first relates to a basic application of MODE to high-resolution precipitation forecasts in the Colorado Front Range. The second example demonstrates a more advanced application of MODE results to diagnose forecast biases.

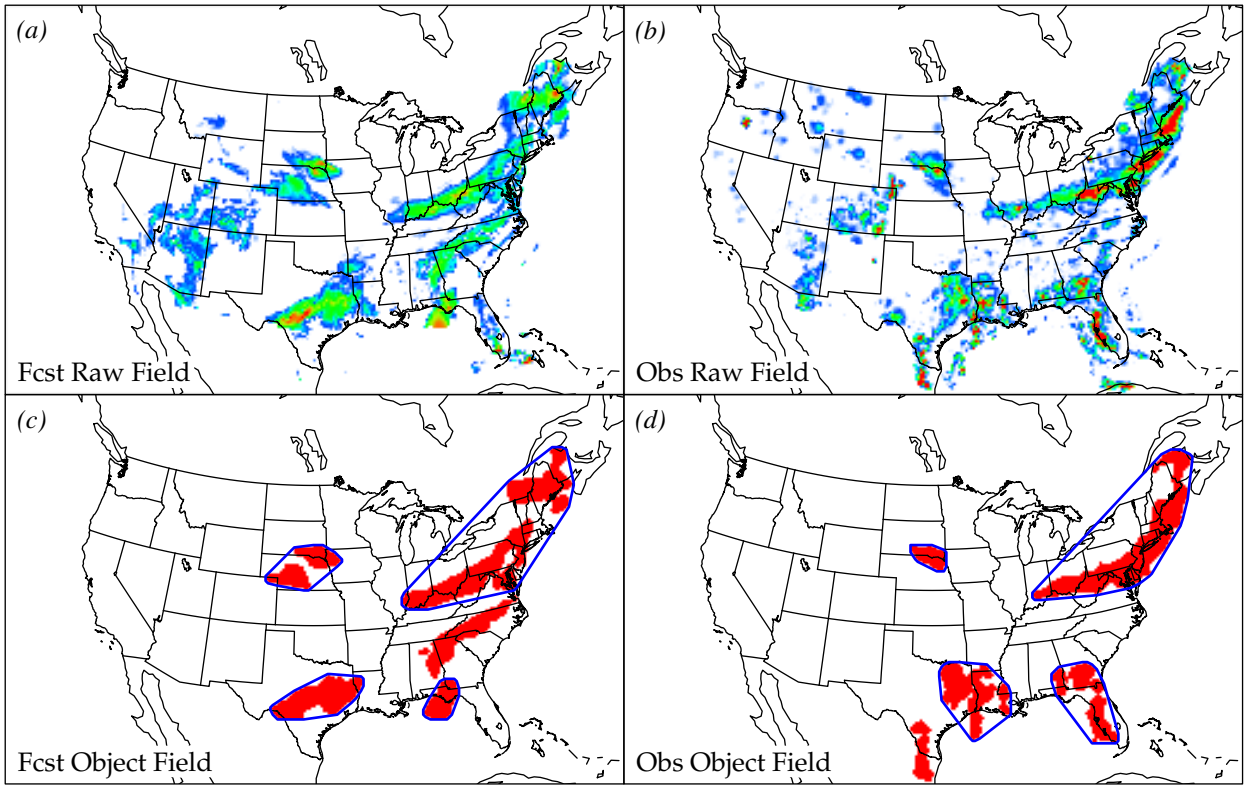


Figure 3 What are objects?

1.5.1 High-resolution precipitation example

The first example of an application of MODE is shown in Figure 4. Parts (a) and (b) in this figure show gridded forecast and observed precipitation values for a convective storm case in the plains to the east of the Rocky Mountains, primarily in Colorado and Kansas. The forecasts in this case are based on a 4-km version of the Weather Research and Forecasting (WRF) model. The observation field is based on a combined analysis of radar and rain gauge measurements. The forecast was a 6-h prediction of 1-hourly accumulated precipitation in millimeters.

The MODE objects identified for this case are represented by the colored areas in parts (c) and (d) of Figure 4. The MODE object resolution process is explained in Section 2. Merged individual objects within the same field are identified as clusters in by the "rubber band" (convex hull outline) surrounding them and by having the same color (*e.g.*, the red objects in the observed field). The simple object numbers are shown in parts (e) and (f).

Unmatched objects (*i.e.*, false alarms in the forecast field and misses in the observed field) are shown as blue. In this example, there is only one small false alarm; however, several small blue object areas in part (d)

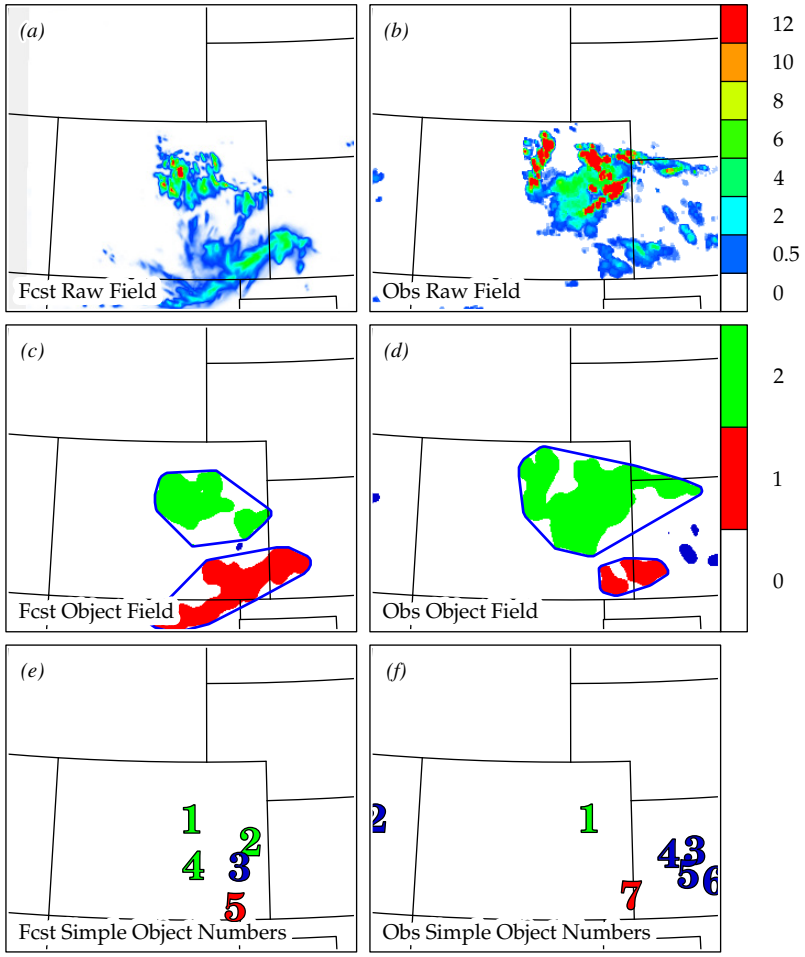


Figure 4 Example MODE application to a gridded precipitation forecast

indicate that there were a number of missed areas. The process of matching observed and forecast objects is based on comparisons of certain attributes of the objects in both fields (described in Section 3) and applying a fuzzy logic algorithm (considered in Sections 4 and 5). Measuring the attributes (which include things like object area, intensity, and location) is described in Section 3.

Table 1 shows a summary of the attributes associated with the objects in Figure 4, and their comparison between the forecast and observed fields. The results shown in this table are described and explained in later sections of this document.

1.5.2 Appalachia example

An illustrative example of the uses to which the notion of objects can be put comes from the early days of MODE development, when the basic ideas behind this approach to object-based verification were still being sorted out. For this investigation the continental United States was divided up into about a dozen

Attribute	Cluster 1	Cluster 2	Attribute	Cluster 1	Cluster 2
Centroid Distance	31.4	20.9	Forecast 50%-ile Intensity	1.12	1.13
Axis Angle Difference	12.8	19.9	Observed 50%-ile Intensity	1.00	2.40
Forecast Area	3802	3187	50%-ile Intensity Difference	0.12	-1.27
Observed Area	1208	7168	50%-ile Intensity Ratio	1.12	0.47
Area Difference	2594	-3981	Forecast 90%-ile Intensity	2.68	4.61
Area Ratio	3.2	0.44	Observed 90%-ile Intensity	2.10	15.20
Intersection Area	1080	2436	90%-ile Intensity Difference	0.58	-10.59
Union Area	3930	7919	90%-ile Intensity Ratio	1.27	0.30
Symmetric Difference	2850	5483	Total Interest	0.92	0.95

Table 1 Attribute comparisons for MODE example

regions, and for each region the vector differences between centroids of forecast precipitation objects and corresponding matched observed objects were examined, subject to the condition that the observed object was inside the given region. The matched forecast object was not required to be inside the region.

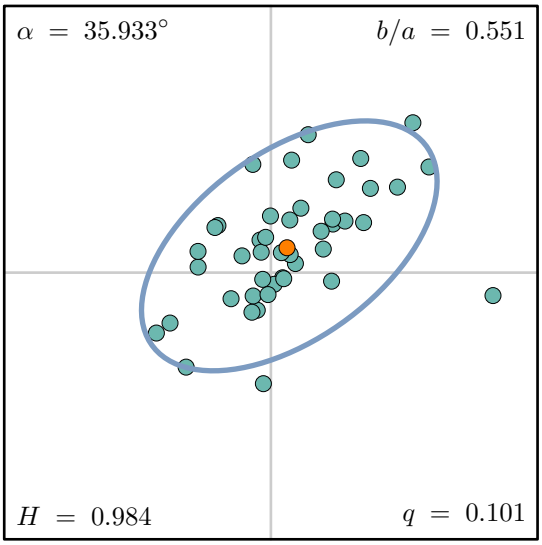


Figure 5 Appalachia example

One of the regions covered the Appalachian mountain range. A plot of the vector differences for that region is shown in Figure 5. Notice the obvious directional signal in the vectors. This result was interpreted

as indicating that when there was an error in the forecast storm location, there was a strong tendency for the storms to be displaced parallel to the mountain range, rather than in some other direction. Thus there was a *directional* bias to the forecast displacements, evidently due to influences from local topography.

From this point, one could go on to stratify the verification by other object attributes. Does the direction bias hold for all storms, or is the bias greater for more intense storms than for weaker ones? Does the size of the storm matter? Does it matter whether the storm is over water or over land? Is the bias the same in different parts of the grid?

These are just a few good examples of the wide variety of questions that can be explored when an object-based approach to forecast verification is used. Such information simply cannot be extracted from summary statistics like POD and FAR.

1.6 Summary

MODE is a verification and model evaluation methodology that has been developed to provide diagnostic information about forecast performance. In addition to a direct comparison of matched objects, MODE can be used to compare climatological distributions of selected object attributes. The methodology has been widely applied to weather forecast verification, and is increasingly being utilized in climate applications.

2

Resolving Objects

2.1 Overview

Object-based verification begins with resolving a gridded data field into *objects*. As mentioned earlier, objects are just spatial regions of interest. How does MODE resolve the raw data field into these "regions of interest"?

Figure 6 How do we resolve objects?

Human beings find it easy to examine a plot of a forecast or observed data field (*e.g.*, precipitation) and resolve the field into objects (both simple and composite). The original design goal for this part of MODE was in fact to emulate what a human being would do. However, programming a computer to do this task turns out to be fairly difficult. While the MODE developers initially looked at some fairly exotic methods, they ultimately decided on a convolution-thresholding approach. This breaks down into several steps (see Figure 7). These operations (convolution, thresholding, restoring the raw data and splitting) are discussed in this section.

2.2 Grids

Before we say too much about objects, let's look at how grids are thought of in MODE. Meteorological grids have a great deal of associated metadata, such as which map projection is used, the grid resolution, the latitude and longitude of the anchor point (usually the lower-left corner), *etc.*, but MODE only cares about the number of grid points in the x and y directions, which we'll denote by N_x and N_y respectively.

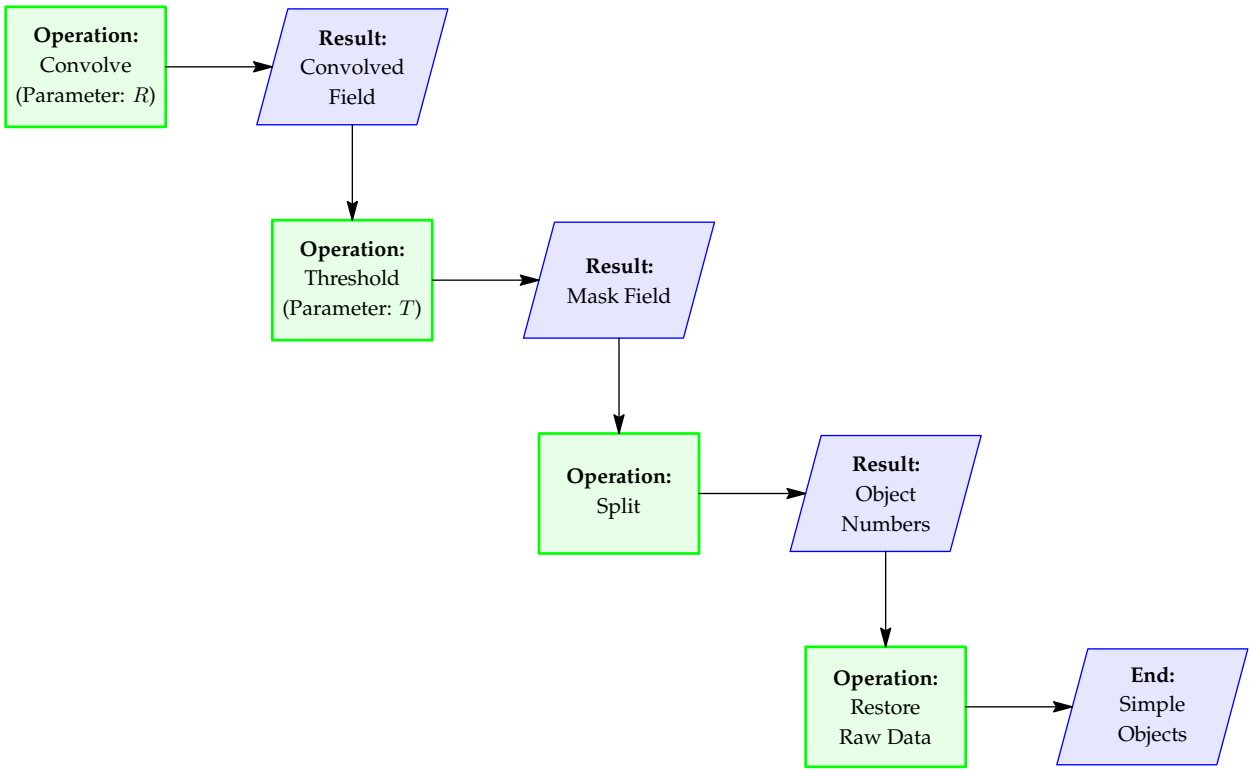


Figure 7 Flowchart for resolving objects

Thus, as far as MODE is concerned, a grid G is just the set of $N_x N_y$ points

$$G = \{ \ (x, y) \ \ | \ \ 0 \leq x < N_x, \quad 0 \leq y < N_y \quad \text{with} \quad x, y \in \mathbb{Z} \ \}$$

where $\mathbb{Z}$ is the set of integers: $\mathbb{Z} = \{0, \pm 1, \pm 2, \ldots\}$.

A data field F on a grid G is then a real-valued function on G; in other words, a rule that associates a real number to each point in G. Occasionally it will be useful to consider the domain of a gridded data field F to be extended to all of $\mathbb{Z}^2$ by defining it to be zero outside of the grid.

2.3 Convolution

Convolution is an operation on two functions, which returns another function. It has many uses in physics, statistics, engineering, mathematics, and computer vision. For example, in statistics, weighted averages can be written as convolutions (more on this later). Also, the probability density function (pdf) of a sum of two random variables is the convolution of the two individual pdf's. In computer vision, many digital image filters can be implemented as convolutions.

In MODE the convolution operation is used as a general way to implement the idea of a weighted average. The two input fields to the convolution operation are the raw data field and another function,

called the convolution filter. The convolution process is governed by the filter's one tunable parameter R, referred to as the *radius*, or the *radius of influence*. If we let x and y denote (integer) coordinates on a grid G and let $f(x, y)$ denote the raw data field on G, then the convolved field $C(x, y)$ is given (in the discrete case) by

$$C(x,y) = \sum_{(x',y')\in G} f(x',y')\,\phi(x-x',y-y') \tag{2.1}$$

where the convolution filter function ϕ is given by

$$\phi(x,y) = \begin{cases} H & \text{if } \sqrt{x^2+y^2} \le R \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

In the continuous case, where x and y can take on real (and not just integer) values, the sum is replaced by an integral in Eq (2.1).

The height H of the filter is chosen so that (in xy coordinates) the area under the graph of ϕ is one:

$$\int_G \phi = \pi R^2 H = 1. \tag{2.3}$$

Convolution will be denoted by $*$, so we can rewrite Eq. (2.1) as $C = f * \phi$. An easy calculation shows that the convolution operation is commutative (*i.e.*, $f * \phi = \phi * f$) so that one will sometimes see convolution defined as

$$C(x,y) = \sum_{(x',y')\in G} \phi(x',y')\,f(x-x',y-y') \tag{2.4}$$

rather than as in Eq. (2.1).

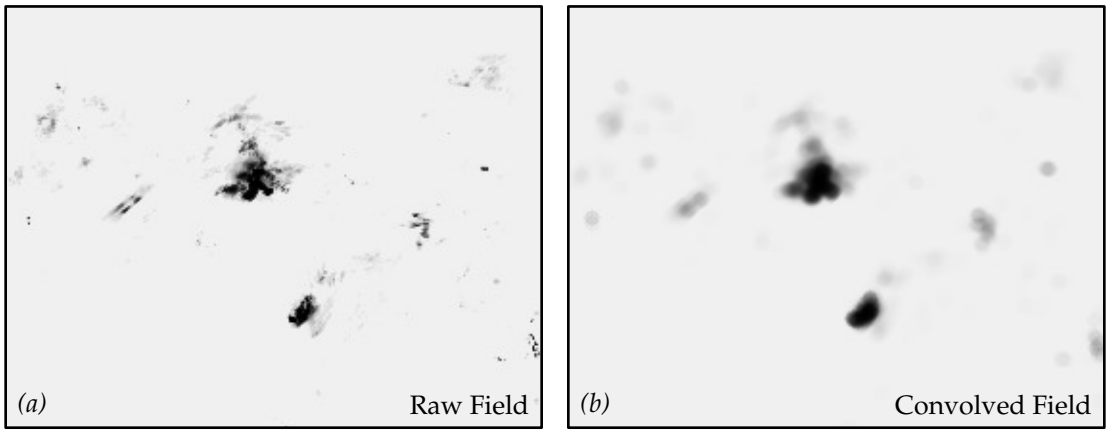


Figure 8 Comparison of raw field with convolved field

The result of this process can be seen in Figure 8. Part (a) of that figure shows a raw data field on a grid. The result of the convolution operation is shown in part (b). Note that the end result is very much

like applying a low-pass filter—features with high spatial frequencies are attenuated. As mentioned above, the simple circular convolution filter ϕ that MODE uses has only one tunable parameter—the convolution radius R. Later in this document (Section 6.4, page 60) we'll discuss why we don't use other, more general, filters in MODE.

2.3.1 Interpretation

Probably the best way to think about convolution is as a moving average. In this section we'll use the continuous (*i.e.*, integrals instead of sums) definition of convolution because it's a little easier to work with. Also, we'll just work in one dimension rather than two in order to simplify the argument.

Recall from elementary calculus that the average value of a function g on an interval $[a, b]$ is given by

$$\text{Average of } g = \frac{1}{b-a}\int_a^b g(x)\,dx.$$

In 1 dimension, we define our convolution filter function ϕ as

$$\phi(x) = \begin{cases} H & \text{if } |x| \le R \\ 0 & \text{otherwise} \end{cases}$$

(compare Eq. (2.2)) where the constant H is chosen so that $\int_{\mathbb{R}} \phi = 1$, in other words, $H = 1/(2R)$ (compare Eq. (2.3)). For a suitably well-behaved function f, the convolution $C = f * \phi$ is given by

$$C(x) = \int_{-\infty}^{\infty} f(u)\phi(x-u)\,du$$

How can we interpret the function C? We can restrict the range of integration to where the filter function ϕ is nonzero:

$$\begin{aligned} \phi(x-u) \neq 0 \quad &\text{iff} \quad |x-u| \le R \\ &\text{iff} \quad |u-x| \le R \\ &\text{iff} \quad -R \le u-x \le R \\ &\text{iff} \quad x-R \le u \le x+R \end{aligned}$$

and for this range of u values we have $\phi(x-u) = H = 1/(2R)$. Hence

$$\begin{aligned} C(x) &= \int_{x-R}^{x+R} f(u)\,H\,du \\ &= \frac{1}{2R}\int_{x-R}^{x+R} f(u)\,du \\ &= \text{average of } f \text{ on } [x-R, x+R]. \end{aligned}$$

Thus the effect of convolving a function f with our specially chosen filter function ϕ is to calculate the average of f in a fixed-size neighborhood (radius R) around each point.

Similar reasoning shows that if the filter function ϕ is not constant in this neighborhood, the result is still an average, but now it's a *weighted* average, with the values of ϕ determining the weights.

2.3.2 Properties

Convolution has several noteworthy properties that are useful in calculations and for gaining further insight into the operation.

We've already mentioned the commutativity of the convolution operation ($f * g = g * f$), and its interpretation as a moving average. Another interesting property is that if f and g are real-valued functions on $\mathbb{R}^n$, then

$$\int_{\mathbb{R}^n} (f * g)(x)\, dx = \left(\int_{\mathbb{R}^n} f(x)\, dx \right) \left(\int_{\mathbb{R}^n} g(x)\, dx \right)$$

so that if f represents a raw data field and if $\int \phi = 1$, then $\int (f * \phi) = \int f$, and hence the total field amount on a grid is preserved (or nearly so, because some will drop off the edge of the grid when smoothed). This is the reason we require $\int \phi = 1$ (*cf.* Eq (2.3)).

2.3.3 Fourier Approach

As remarked earlier, convolution is basically a smoothing operation. High spatial frequencies are greatly attenuated, lower frequencies less so. The reader may wonder whether a frequency domain approach (such as the Fourier Transform) would be more suited to this task. In fact, a Fourier approach is really no more general than what MODE uses. This can be seen as follows. One of the general properties of the Fourier transform is that it maps convolutions to products. Thus, if $\mathbf{F}$ denotes the Fourier transform, then

$$\mathbf{F}(f * \phi) = \mathbf{F}(f)\, \mathbf{F}(\phi) \tag{2.5}$$

Suppose $\phi = \mathbf{F}^{-1}(\hat{\phi})$ for some frequency-domain function $\hat{\phi}$. (For example, $\hat{\phi}$ could be a step function representing a band-pass filter.) Then we would have

$$\mathbf{F}(f * \phi) = \mathbf{F}(f)\, \mathbf{F}(\phi) = \hat{f}\, \hat{\phi} \tag{2.6}$$

where $\hat{f} = \mathbf{F}(f)$. Taking inverse transforms gives

$$f * \phi = \mathbf{F}^{-1} \left(\hat{f}\, \hat{\phi} \right) \tag{2.7}$$

This says that the net effect of (1) taking the Fourier transform of the raw data f, (2) applying a filter $\hat{\phi}$ (such as a band-pass filter) in the frequency domain, and then (3) taking the inverse transform, can be achieved by convolving the original field with the inverse Fourier transform of the frequency domain filter $\hat{\phi}$. The two approaches are completely equivalent mathematically. Which one is done is therefore a matter of choice, perhaps guided by considerations of computational efficiency.

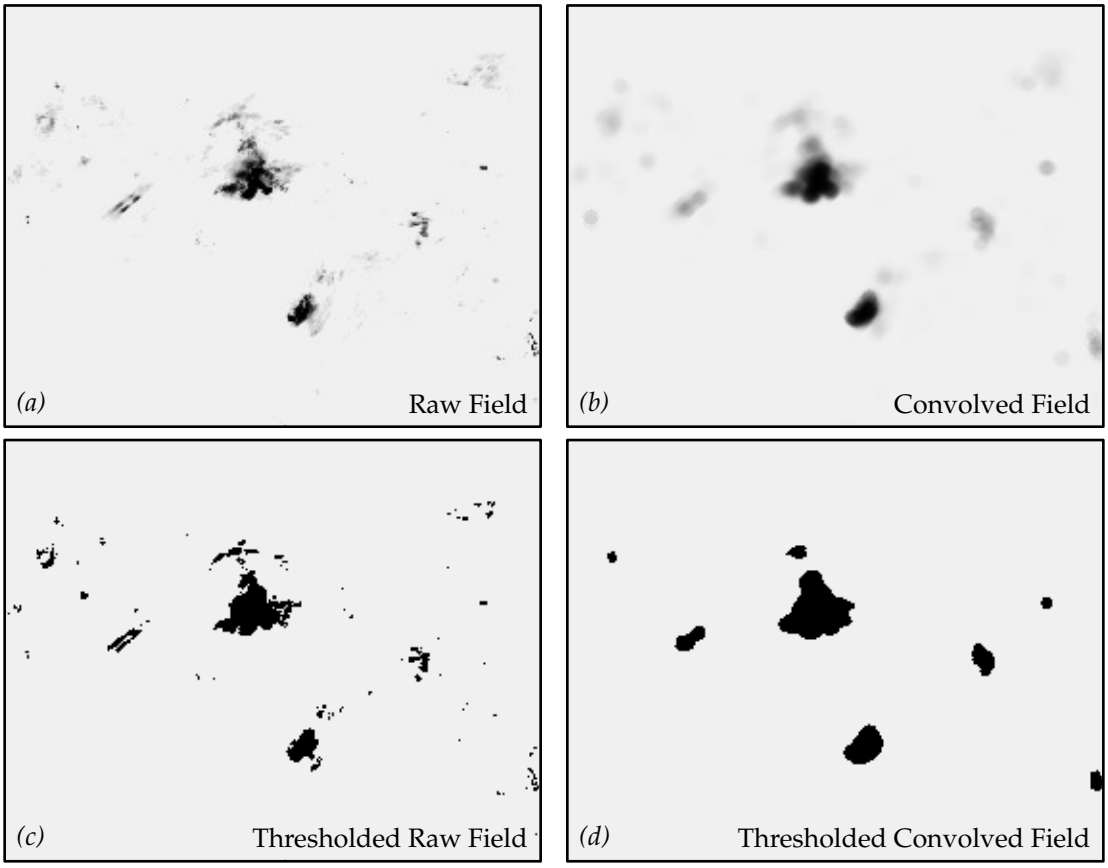


Figure 9 Thresholding the raw field and the convolved field

2.4 Thresholding

Once the convolved field C has been calculated, it is thresholded to produce a mask field M:

$$M(x,y) = \begin{cases} 1 & \text{if } C(x,y) \geq T \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

Here, T is the threshold. The object field consists of those grid squares where $M = 1$, and the individual objects will be the connected components of that field. All of the purely geometrical properties of the objects are derivable from M. The only things missing are the data values inside the objects.

The reader may wonder whether simply thresholding the raw data would be enough to resolve the field into objects. If this were so, we could do away with the convolution step altogether. Unfortunately thresholding the raw data does not usually result in a small number of representative objects in the field, as can be seen in Figure 9, which is an expanded version of Figure 8 (page 22). In parts (a) and (b) we see the raw and convolved fields taken from Figure 8. Parts (c) and (d) show the results of thresholding the

raw and convolved fields. The thresholded raw field contains 78 objects, most of them quite small. Compare this with the result of thresholding the convolved field, which shows 8 objects, most of them fairly large. The thresholded convolved field gives us a collection of objects that corresponds more closely to the way a human being would draw the objects in the raw field. The thresholded raw data gives too many objects, most of which are very small, and separates regions that should be a single object into many parts. This is typical behavior for thresholded raw fields and shows the necessity of processing the raw data with some sort of intermediate step (such as convolution) before thresholding is performed.

Once the convolved field C has been thresholded, and the mask field M is in hand, the convolved field in essentially thrown away. MODE has no further use for it. It plays no role either in the remainder of the object resolution process, or in the calculation of object attributes, or in the fuzzy-logic-based matching and merging algorithms.

2.5 Splitting

After thresholding, the mask field M assigns a zero or one to each grid point, according to Eq. 2.8. Those grid points (x, y) such that $M(x, y) = 1$ will be used to form the simple objects. The simple objects will be the connected parts of the grid where $M(x, y) = 1$. Figure 10 illustrates this idea. On the left side of the figure we have a small grid with those points where $M(x, y) = 1$ marked in green. Individual grid *points*, of course, have no area, so in order to associate a region of the grid to each object, we consider the object to be made up of those grid *squares* that have the mask points as their lower left corner. These squares are shown in blue. We see that there the blue squares form 3 connected pieces. Thus there are 3 simple objects in this field.

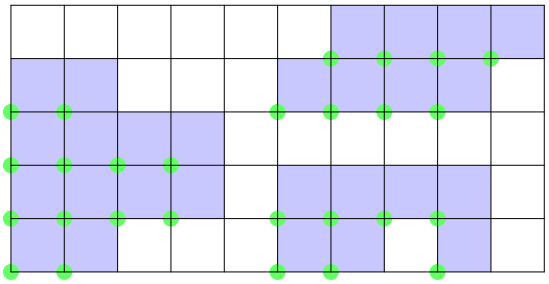
Once the connected pieces (*i.e.*, the simple objects) have been found, they are numbered consecutively. This is show in the right side of Figure 10, where each grid square is assigned a number corresponding to the simple object of which it is a part.

This process of determining the connected pieces of the mask field on the grid, and assigning numbers to the objects is called *splitting*. Once splitting has been done, the simple objects and their object numbers are known. At this point, simple object areas could be obtained by counting the grid squares comprising each simple object. For example, in Figure 10, object #1 has area 8, object #2 has area 12, and object #3 has area 7.

2.6 Restoring the Data

The final step is to restore the original data to object interiors. All original data outside the objects is zeroed out. The object field B is defined by

$$B(x, y) = \begin{cases} f(x, y) & \text{if } M(x, y) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$



0	0	0	0	0	0	1	1	1	1
2	2	0	0	0	1	1	1	1	0
2	2	2	2	0	0	0	0	0	0
2	2	2	2	0	3	3	3	3	0
2	2	0	0	0	3	3	0	3	0

Figure 10 Numbering simple objects

or, more simply, $B = fM$. This restoration of the raw data enables the calculation of intensity-based attributes for objects. Thus we can speak, for example, of the median intensity value for an object or the 90^{th} percentile intensity value. It also would allow the calculation of other intensity-based object attributes, such as an intensity-weighted centroid or axis.

2.7 Example

The convolution radius used for the example in Section 1.5.1 (page 16) was 6 grid squares and the threshold applied was 0.5 mm. Comparing the raw field to the convolved objects indicates that these choices resulted in reasonable definitions of the objects, but not at as fine a scale that would be required to only identify the central "core" areas of the storm systems. On the other hand, the objects are not very smooth so likely can provide a reasonable assessment of the storm systems. Furthermore, in order to identify the central "core" areas of the storm system, a smaller convolution radius or a higher threshold could be used.

2.8 Summary

In MODE the identification of objects is based on a convolution and thresholding process. The two parameters that must be selected by the user are (i) the convolution radius and (ii) the convolution threshold. The characteristics of the objects defined are very dependent on how these parameters are selected, and can range from many small objects to a few very large smooth objects. Different combinations of these parameters may be desired for different applications.

3

Attributes

3.1 Overview

Attributes are quantities that measure spatial features or properties of objects that may be of interest. MODE, of course, doesn't "see" objects the way humans do—it works instead with the object attributes, and all of its computations are ultimately based on measured or calculated values for these attributes.

MODE uses a great many attributes—we'll only look at some of the most commonly used ones in this paper.

3.2 Single Attributes and Pair Attributes

The attributes that MODE computes fall into two categories. The first kind of attribute is for single objects. Individual objects resolved from either the forecast field or the observation field will have attributes that can be calculated without reference to any other object. One example of a single-object attribute is object area. We'll talk more about area attributes later, but for now just think of the area as a count of the number of grid squares inside the object.

The other type of attribute is for object *pairs*. Taking one object from the forecast field and one from the observed field, one can calculate attributes that involve both objects together. An example of this type of attribute is intersection area: the area that is inside both objects simultaneously.

Many pair attributes are derived by taking either differences or ratios of single-object attributes. Thus, for example, you could look at the ratio of the forecast object area to the observed object area. One advantage of taking ratios is that the resulting value is dimensionless and hence independent of the scale of the grid.

3.3 Moments

Moments are used in the calculation of several object attributes. In this section we discuss the calculation of moments for an object defined either as a collection of grid squares, or as a region bounded by a simple closed polyline. Both cases are important for MODE, since the forecast and observed objects are collections of grid squares, while several attributes involve the creation of closed polylines that derive from the original

object in some way. Only the calculation of moments differs for the two cases—once the moments are in hand, calculation of attributes (such as centroids and axes) proceeds in the same way for both situations.

Note: Readers with a background in statistics may be familiar with the concept of moments as used in describing, *e.g.*, probability density functions. However, the use of moments in MODE has no statistical overtones.

3.3.1 Shapes Composed of Unions of Grid Squares

First, we'll examine moments for shapes that are defined as collections of grid squares. Suppose we have a two-dimensional grid $G = \left\{ (x, y) \in \mathbb{Z}^2 \mid 0 \leq x < N_x,\ 0 \leq y < N_y \right\}$ and a region of interest (*i.e.*, object) $\Omega \subset G$. (The reader who is unfamiliar with some of this notation should consult the *List of Symbols* on page 74.) We can define a real-valued function χ on G by

$$\chi(x, y) = \begin{cases} 1 & \text{if } (x, y) \in \Omega \\ 0 & \text{if } (x, y) \notin \Omega \end{cases} \tag{3.1}$$

(*Note:* Statisticians would call χ the *indicator function* of Ω, while mathematicians would call χ the *characteristic function* of Ω.) We can then define moments of Ω by summing certain functions over the grid.

Consider the area of an object. This will be expressed simply as a count of the number of grid squares occupied by the object and can be thought of as a kind of "zero-th moment" of the object.

$$A = \sum_{(x,y)\in\Omega} 1 \tag{3.2}$$

$$= \sum_{x,y} \chi(x, y) \tag{3.3}$$

There are two first moments, one in x and one in y, denoted respectively by S_x and S_y, defined by

$$S_x = \sum_{x,y} x\,\chi(x, y) \qquad \text{and} \qquad S_y = \sum_{x,y} y\,\chi(x, y) \tag{3.4}$$

There are four 2nd moments (although only three are in general distinct) defined by

$$S_{xx} = \sum_{x,y} x^2\,\chi(x, y) \qquad S_{xy} = S_{yx} = \sum_{x,y} xy\,\chi(x, y) \qquad S_{yy} = \sum_{x,y} y^2\,\chi(x, y) \tag{3.5}$$

While 3rd order moments will be needed when we discuss curvature, we will not examine in detail moments higher than the 2nd in this section.

3.3.2 Shapes Bounded by Closed Polylines

Now we'll examine the case where we're calculating moments for the region inside a closed polyline. Figure 11 shows an example of a closed polyline consisting of straight line segments connecting some vertices. The shape consists of the shaded area inside the polyline.

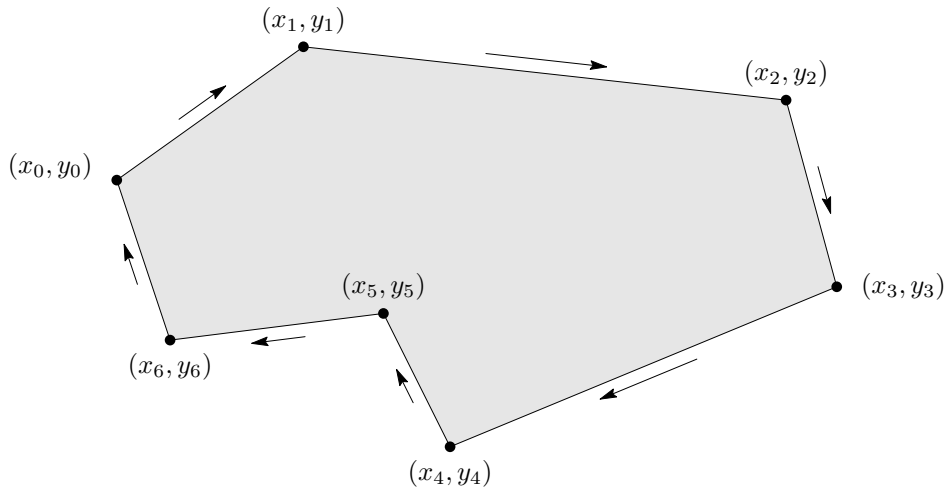


Figure 11 Region inside a closed polyline

Surprisingly, the situation here (at least from a computational point of view) is really no more complicated than it was for objects on a grid. Deriving the requisite formulas is more involved, but the actual computations really only involve looping around the vertices of the polyline and performing some simple arithmetic on the x and y coordinates of the vertices.

Here we'll use integrals instead of sums to calculate moments. Also, we'll use a theorem from the calculus of several variables to help us do those integrals. Recall Green's Theorem in the Plane:

$$\int_C (L\,dx + M\,dy) = \iint_D \left(\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y}\right) dx \wedge dy \tag{3.6}$$

for any sufficiently well-behaved functions $L(x, y)$ and $M(x, y)$. (*Note:* Those readers who are unfamiliar with the wedge product "$\wedge$" symbol in this or in later formulas can either ignore it, or simply consider it a device for keeping track of orientation.)

For later convenience, we'll rewrite Eq. (3.6) slightly as

$$\int_C \alpha = \iint_D \beta \tag{3.7}$$

where

$$\alpha = L\,dx + M\,dy \quad \text{and} \quad \beta = \left(\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y}\right) dx \wedge dy \tag{3.8}$$

This theorem relates the integral of β over a two-dimensional region D to the integral of α over the boundary C of the region. The boundary C has to be piecewise smooth for the theorem to hold, but this will certainly be the case for the closed polylines we're interested in.

Let's start, as before, by looking at area. The area form on the plane $\mathbb{R}^2$ is $\beta = dx \wedge dy$. We can use Green's Theorem to turn the integral of β over Ω into an integral around the enclosing polyline of another

form α chosen so as to make the theorem work — in other words, α and β must be related to each other in the manner specified by (3.8). As mentioned before, we need to know that Green's Theorem holds for regions whose boundaries are only piecewise smooth, but fortunately it does. (Note: the technical term for such a region is a two-dimensional *manifold with corners*. See Lee (2013) in the references for more information.)

There are many choices for α. We will use $\alpha = (x\,dy - y\,dx)/2$. It is now necessary to integrate this form around the closed polyline bounding our object. We do this by integrating α over each of the edges of the polyline and then summing the results. Let's say the polyline has n vertices (x_i, y_i) with $i \in \mathbb{Z}/n\mathbb{Z}$ (as shown in Figure 11, where $n = 7$). If we consider edge $\#i$ as being the straight line segment running from (x_i, y_i) to (x_{i+1}, y_{i+1}), we can then parametrize this segment as follows:

$$\begin{aligned} x(t) &= x_i + t\,(x_{i+1} - x_i) \\ y(t) &= y_i + t\,(y_{i+1} - y_i) \end{aligned} \tag{3.9}$$

for $0 \le t \le 1$. Using this parametrization we can pull α back to a form involving t, which (after some cancellation) simplifies down to $(1/2)\,(x_i\,y_{i+1} - y_i\,x_{i+1})\,dt$. Finally, we integrate this over the range $0 \le t \le 1$ to get $(1/2)\,(x_i\,y_{i+1} - y_i\,x_{i+1})$. The area of the object is now obtained by summing over the edges:

$$A = \frac{1}{2} \sum_i (x_i\,y_{i+1} - y_i\,x_{i+1}) \tag{3.10}$$

A few points to note here: first, as has been mentioned, our integral over the bounding polyline is an *oriented* integral. If the direction (clockwise *vs.* counter-clockwise) in which the polyline is traversed is changed, the integral will change sign. Therefore to get the true geometrical area we should take absolute values after the summation. Second, the summation index i takes values in $\mathbb{Z}/n\mathbb{Z}$ rather than $\mathbb{Z}$. Thus if $i = n - 1$, then $i + 1$ is to be interpreted as 0, not n.

We can do similar things for higher moments. We will simply present the results here, rather than plodding through each derivation separately. In Table 2, we indicate the form β which would be integrated over the object to define the moment, the form α, chosen to make Green's Theorem work, integrated around the bounding polyline, and the i^{th} term in the resulting summation. Looking at this table, the reader can easily see why we are not considering moments higher than the 2nd in this section ... there would hardly be room to write them down.

3.3.3 Transformation of Moments

Now that we know how to calculate moments for both classes of objects—those that are given on a grid and those given as polyline boundaries, it's worthwhile to consider moments more generally. Up until now, we have been working with moments *about the origin*. What about moments about another point? (Ultimately, we will want to work with moments about the centroid, for doing this will simplify some of the equations drastically.) And what if our object moves or rotates? Do we have to calculate the moments all over again from scratch? It would be nice if there were some way to calculate new moments from old. This

β	α	Summation Term
$dx \wedge dy$	$\frac{x\,dy - y\,dx}{2}$	$\frac{x_i\,y_{i+1} - y_i\,x_{i+1}}{2}$
$x\,dx \wedge dy$	$\frac{x^2\,dy}{2}$	$\frac{\left(x_i^2 + x_i\,x_{i+1} + x_{i+1}^2\right)\,(y_{i+1} - y_i)}{6}$
$y\,dx \wedge dy$	$-\frac{y^2\,dx}{2}$	$\frac{(\,x_i - x_{i+1}\,)\,\left(y_i^2 + y_i\,y_{i+1} + y_{i+1}^2\right)}{6}$
$x^2\,dx \wedge dy$	$\frac{x^3\,dy}{3}$	$\frac{\left(x_i^3 + x_i^2\,x_{i+1} + x_i\,x_{i+1}^2 + x_{i+1}^3\right)\,(y_{i+1} - y_i)}{12}$
$y^2\,dx \wedge dy$	$-\frac{y^3\,dx}{3}$	$\frac{(\,x_i - x_{i+1}\,)\,\left(y_i^3 + y_i^2\,y_{i+1} + y_i\,y_{i+1}^2 + y_{i+1}^3\right)}{12}$
$xy\,dx \wedge dy$	$\frac{xy\,(\,x\,dy - y\,dx\,)}{4}$	$\frac{(x_i\,y_{i+1} - y_i\,x_{i+1})\,(\,2\,x_i\,y_i + x_{i+1}\,y_i + x_i\,y_{i+1} + 2\,x_{i+1}\,y_{i+1}\,)}{24}$

Table 2 Differential forms used for calculating object moments

can be done provided we restrict ourselves a simple enough class of coordinate transformations. In our case, it will be enough to consider coordinate transformations that are compositions of translations and rotations.

Suppose we have a coordinate transformation $(x, y) \mapsto (u, v)$ defined by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix} \tag{3.11}$$

where θ, b_x and b_y are given constants. Translations and rotations are special cases of this general form. How are the u, v moments related to the x, y moments? Let ϕ denote the transformation, so that $(u, v) = \phi(x, y)$.

We'll denote the 2×2 matrix above by $\mathbf{M}$, or by $\mathbf{M}(\theta)$ if we want to emphasize its dependence on the angle θ. Note the the determinant of $\mathbf{M}$ is 1, independent of θ, which implies that $du \wedge dv = dx \wedge dy$.

As an example of the use of this transformation, let A' be the (signed) area of the image object $\phi(\Omega)$. In other words, $\phi(\Omega)$ is the result of applying the rotation and translation described by Eq. (3.11) to the object Ω. Then from the general formula for change of variables in multiple integrals we have

$$A' = \int_{\phi(\Omega)} du \wedge dv = \int_{\Omega} \phi^* \left(du \wedge dv\right) = \int_{\Omega} dx \wedge dy = A \tag{3.12}$$

(Note that the Jacobian of the transformation given by Eq. (3.11) is just the determinant of $\mathbf{M}$, which is 1.) The notation ϕ^* denotes the operation of substituting in for u and v their definitions in terms of x and y.

Thus $A' = A$, and this confirms our intuition that moving an object around or rotating it shouldn't change its area.

What about first moments? We have $\phi^* (u\, du \wedge dv) = (x \cos\theta + y \sin\theta + b_x)\ dx \wedge dy$, and so integrating as above we get

$$S_u = S_x\ \cos\theta + S_y\ \sin\theta + b_x\, A \tag{3.13}$$

Similarly,

$$S_v = -S_x\ \sin\theta + S_y\ \cos\theta + b_y\, A \tag{3.14}$$

Second moments are messier. We have $\phi^* (u^2\, du \wedge dv) = (x \cos\theta + y \sin\theta + b_x)^2\ dx \wedge dy$, from which

$$S_{uu} = S_{xx} \cos^2\theta + S_{yy} \sin^2\theta + b_x^2\, A + 2 S_{xy} \sin\theta \cos\theta + 2\, b_x\, (S_x \cos\theta + S_y \sin\theta) \tag{3.15}$$

Similarly,

$$S_{vv} = S_{xx} \sin^2\theta + S_{yy} \cos^2\theta + b_y^2\, A - 2 S_{xy} \sin\theta \cos\theta + 2\, b_y\, (-S_x \sin\theta + S_y \cos\theta) \tag{3.16}$$

and

$$\begin{aligned} S_{uv} = {} & (S_{yy} - S_{xx}) \sin\theta \cos\theta + \left(\cos^2\theta - \sin^2\theta\right)\ S_{xy} \\ & + b_x\, (-S_x \sin\theta + S_y \cos\theta) + b_y\, (S_x \cos\theta + S_y \sin\theta) + b_x\, b_y\, A \end{aligned} \tag{3.17}$$

Second-order moments will suffice for the calculation of object centroids and axes. For object curvature, 3rd order moments will be needed. However, we won't need to know how those moments transform under rotation, so we'll stop here.

3.4 Fitted Attributes

Some attributes, like area, are obtained directly from the objects, while others have to be "fitted" to the object in some way. This process could involve least squares, or some other goodness-of-fit criterion. In this subsection we examine such fitted attributes.

3.4.1 Centroid

It is possible to move the object without rotating it so that the transformed first moments are zero: $S_u = S_v = 0$. When this condition holds, the origin of the coordinate system is at the *centroid* of the object. When the rotation angle θ is zero, the transformation equations for first moments (Eqs. (3.13) and (3.14)) become

$$S_u = S_x + b_x\, A \tag{3.18}$$

$$S_v = S_y + b_y\, A \tag{3.19}$$

To find the coordinates $(\overline{x}, \overline{y})$ of the centroid, we translate by $(b_x, b_y) = (-\overline{x}, -\overline{y})$ to bring the centroid to the origin. The condition that S_u and S_v both be zero now gives

$$(\overline{x}, \overline{y}) = (S_x/A, S_y/A) \tag{3.20}$$

Note that we do not take absolute values of S_x, S_y or A in this equation. The centroid is illustrated in Figure 12(a). It represents the geometric center of an object. If an object has an axis of symmetry, the centroid will lie on that axis. If the object has two axes of symmetry, the centroid will be at the point where the two axes intersect.

We should also mention an easy error that many people fall into when calculating the centroid of a region inside a closed polyline. Consider the rectangle in Figure 13. By symmetry, the centroid is the center of the rectangle, indicated by the blue dot in the figure. Since this is the same point that would be obtained by taking the coordinates of the four corners and averaging them, it's easy to suppose that this is a good general method for calculating centroids: just take the coordinates of vertices of the enclosing polyline and average them. The failure of this approach can be seen by considering the closed polyline obtained from the red dots in Figure 13. The polyline encloses the rectangle, but the end result of averaging the coordinates of all these red dots is shown by the green dot, which clearly gives an incorrect position for the centroid.

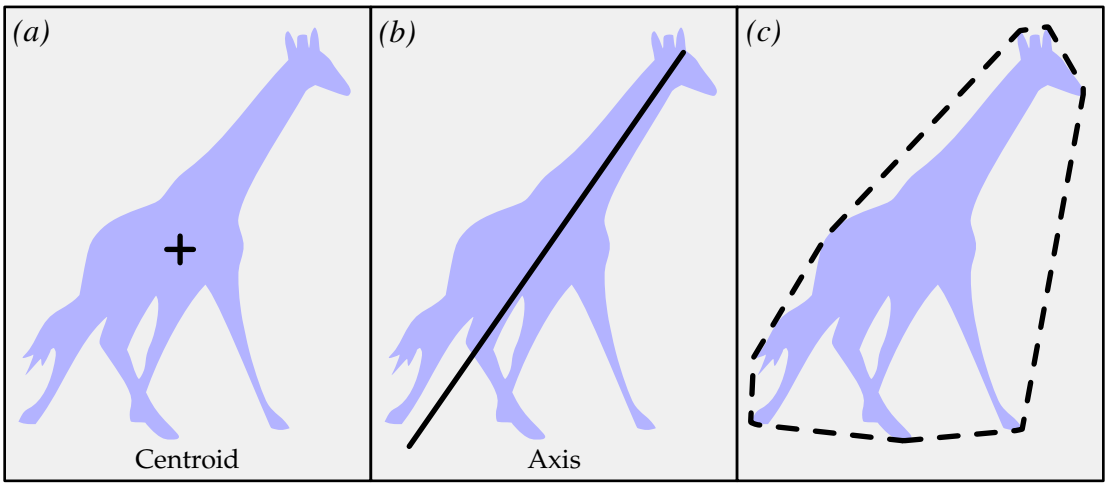


Figure 12 Centroid and Axis

3.4.2 Axis

Now let's consider the axis. The axis can be very useful in assigning an overall spatial orientation to an object. Spatial orientation differences for forecast and observe objects, besides being of interest in their own right, are used as one input to the fuzzy logic matching and merging engine.

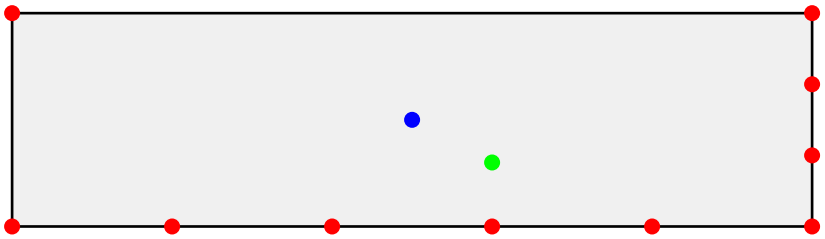


Figure 13 How not to calculate the centroid of an object

We will suppose that the origin of our coordinate system is at the object's centroid. This means that $S_x = S_y = 0$. We'll also confine ourselves to rotations of the coordinate system, so that $b_x = b_y = 0$. Then the transformation laws for 2^{nd} moments become

$$S_{uu} = \cos^2\theta\, S_{xx} + \sin^2\theta\, S_{yy} + 2\,\cos\theta\,\sin\theta\, S_{xy} \tag{3.21}$$

$$S_{vv} = \sin^2\theta\, S_{xx} + \cos^2\theta\, S_{yy} - 2\,\cos\theta\,\sin\theta\, S_{xy} \tag{3.22}$$

$$S_{uv} = \cos\theta\,\sin\theta\,(S_{yy} - S_{xx}) + \left(\cos^2\theta - \sin^2\theta\right)\, S_{xy} \tag{3.23}$$

The urge to jump right in with trigonometric identities and simplify this is strong, but we will resist for the moment, for a better simplification is available. Let

$$\sigma_0 = S_{uu} + S_{vv}, \qquad \sigma_1 = (S_{uu} - S_{vv})/2, \qquad \sigma_2 = S_{uv} \tag{3.24}$$

and, similarly,

$$\rho_0 = S_{xx} + S_{yy}, \qquad \rho_1 = (S_{xx} - S_{yy})/2, \qquad \rho_2 = S_{xy} \tag{3.25}$$

Rewriting the above transformation laws in terms of the σ's and ρ's (and *now* using trigonometric identities) we get a great improvement:

$$\sigma_0 = \rho_0 \tag{3.26}$$

and

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} = \begin{bmatrix} \cos 2\theta & \sin 2\theta \\ -\sin 2\theta & \cos 2\theta \end{bmatrix} \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix} \tag{3.27}$$

We'll define the object axis by requiring that S_{uu} (considered as a function of θ) be maximized. Taking the above expression for S_{uu} and differentiating, we have, as the reader can verify, $S'_{uu} = 2\,\sigma_2$, and $S''_{uu} = -4\,\sigma_1$. By elementary calculus, for S_{uu} to be a maximum we must have $S'_{uu} = 0$ and $S''_{uu} < 0$. We therefore determine the axis angle θ by two conditions—first, that σ_2 be zero, and second, that σ_1 be positive. The above matrix equation is easy to invert:

$$\begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix} = \begin{bmatrix} \cos 2\theta & -\sin 2\theta \\ \sin 2\theta & \cos 2\theta \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} \tag{3.28}$$

and upon setting $\sigma_2 = 0$ we have that $\rho_1 = \sigma_1 \cos 2\theta$ and $\rho_2 = \sigma_1 \sin 2\theta$. It's tempting to simply conclude that $\tan 2\theta = \rho_2/\rho_1$, but what about the quadrant of θ? And what if ρ_1 is zero? Since $\sigma_1 > 0$, we can write

$$\theta = \frac{1}{2} \arg(\rho_1 + \rho_2 \, i) \tag{3.29}$$

Many programming languages have a 2-argument arctangent function, usually called `atan2`. The expression `atan2(y,x)` resolves the quadrant of the point (x, y) correctly (though the fact that the order of x and y is reversed in the argument list is a continual nuisance). Thus we finally have

$$\theta = \texttt{0.5*atan2}(\rho_2, \rho_1) \tag{3.30}$$

and this is our axis angle. For an illustration, see Figure 12(b), which shows an axis angle of about 55°.

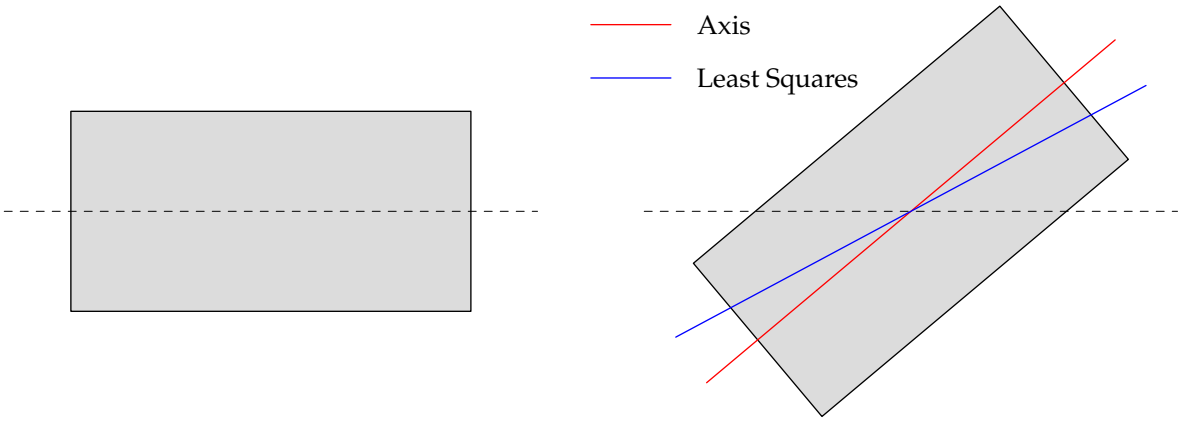


Figure 14 Axis *vs.* Least Squares

The reader may wonder why MODE doesn't simply use a least-squares approach to fitting a line to an object. After all, least-squares linear fits are in very common use—why not just use that approach rather than this (admittedly complex) axis calculation? The answer is that the usual approach to least-squares line fitting is not *rotation invariant*.

What rotation invariance means can be understood by looking at Figure 14. Here we have two objects, indicated by the gray rectangles, that are identical in every respect except that the object on the right has been rotated from the horizontal (*i.e.*, the x-direction, indicated by the dashed line) through an angle of 40°. For the object on the left, both the MODE approach and the least-squares approach to fitting a line to the object give the same result: the fitted line is horizontal. No surprises here.

For the object on the right however, the two approaches give different results. The MODE axis line (shown in red) makes an angle of 40° with the horizontal, while the least-squares line (shown in blue) makes an angle of only about 28°. In effect, the MODE axis line has followed the rotation of the object exactly, while the least-squares line has not. This is what rotation invariance means. The MODE axis line is rotation invariant; the least-squares line is not.

Rotation invariance is clearly a desirable property in an axis line (or indeed any object attribute). If our line-fitting algorithm has the property of rotation invariance, then we can be sure that the fitted line does not depend on how the object happens to be oriented with respect to the local grid directions. The grid lines are, after all, purely imaginary. They shouldn't have any effect on attributes for real-world objects.

3.4.3 Length and Width

To define the length and width of an object, we enclose the object in a rectangle, and set the length and width of the object to be the length and width of the fitted rectangle. Unfortunately, finding the "best" (in whatever sense — smallest area, smallest perimeter, *etc.*) rectangle that encloses a given shape is a nontrivial problem in general, involving a fair amount of computation. We therefore satisfy ourselves with finding a "good," though probably not optimal, rectangle.

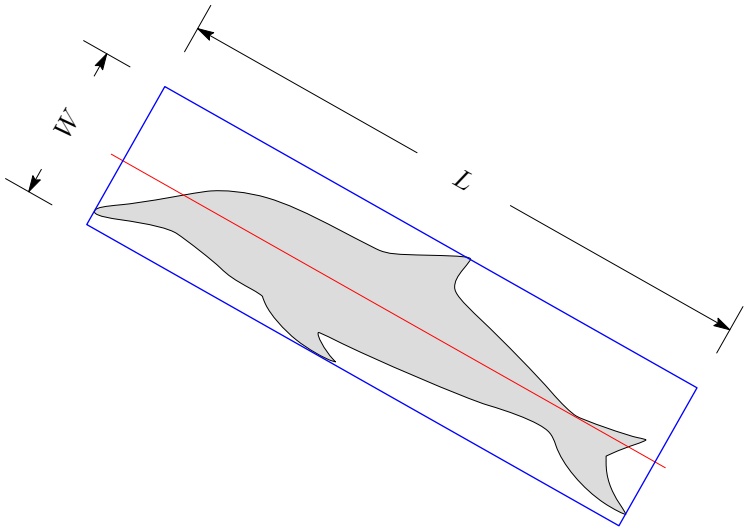


Figure 15 Length and width

We require the rectangle to be the smallest one enclosing the object subject to the condition that the long sides of the rectangle are parallel to the object's axis. This approach is illustrated in Figure 15, where we show a typical object in gray. The object's axis is indicated by the red line, and the enclosing rectangle is outlined in blue. The length L and width W of the rectangle are indicated. Note that the center of the rectangle will not in general coincide with the centroid of the object.

3.4.4 Curvature & Center of Curvature

What do we mean by "curvature" in this context? To get a handle on this, we'll use an analogy based on the slope of a line.

The slope m of a line in the xy-plane is given by the ratio of the rise to the run, $m = \Delta y/\Delta x$, as shown in Figure 16(a). You might say that the geometric "template" for the concept of slope is a straight line. If we have a curve $y = f(x)$ instead of a line, as in Figure 16(b), we can still define a slope for the curve at a point by looking at the line that best approximates (or best *fits*) the curve at that point. The slope of that line (which will now depend on x) is the derivative $f'(x)$. If we have neither a line nor a curve, but just a collection of points in the xy-plane, as in Figure 16(c), we can use (for example) least squares to fit a line to the points, and define the slope of the cloud of points to be the slope of the least squares line.

The situation for curvature is analogous, with the difference that while the geometric template for slope is a line, the geometric template for curvature is a circle. See Figure 16(d). The curvature κ of a circle is defined to be the reciprocal of its radius. The smaller the radius, the greater the curvature. More generally, for a plane curve, we can define the curvature at some point by finding the circle of best fit to the curve at that point (what in geometry is called the *osculating circle*, see Figure 16(e)), and defining the curvature at that point to be the curvature of the best-fit circle. As in the case of slopes, there's a formula involving derivatives for calculating the curvature of a plane curve at any point.

What if we don't have a circle or a curve, but just a collection of data points in the xy-plane, as in Figure 16(f)? After all, that's essentially what an object on a grid really is. It turns out that, as in the case of slopes, the least squares approach provides a way to do this. To get the curvature of an object, MODE does a least squares fit of a circle to the object (considered as a collection of grid points) and defines the curvature of the object to be the curvature of the fitted circle. The *center of curvature* of the object is then the center of the fitted circle.

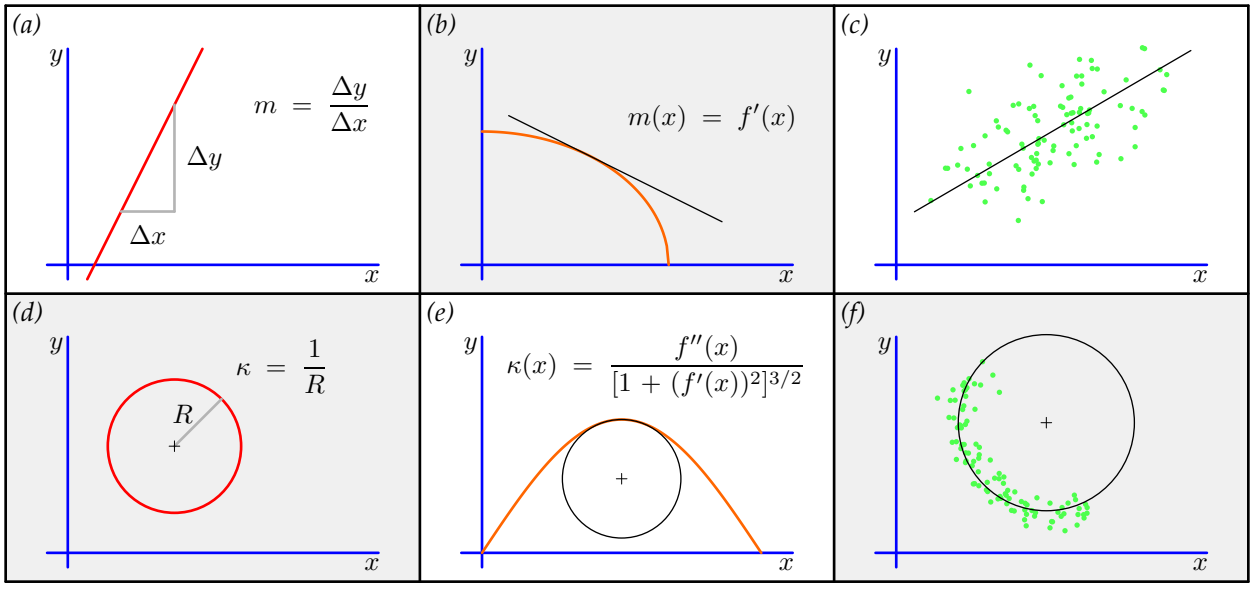


Figure 16 Slope and Curvature

So let's look at how to do a least squares fit of a circle to a set of points. Given a finite set of points in $\mathbb{R}^2$, say $\{\,(x_i, y_i)\,|\,0 \le i < N\,\}$, we want to find the circle that "best" (in a least squares sense) fits the points. Define

$$\overline{x} = \frac{1}{N}\sum_i x_i \qquad \text{and} \qquad \overline{y} = \frac{1}{N}\sum_i y_i \tag{3.31}$$

and let $u_i = x_i - \overline{x}, \quad v_i = y_i - \overline{y}$ for $0 \le i < N$. We solve the problem first in (u, v) coordinates, and then transform back to (x, y).

Let the circle have center (u_c, v_c) and radius R. We want to minimize $S = \sum_i [\,g(u_i, v_i)\,]^2$, where $g(u, v) = (u - u_c)^2 + (v - v_c)^2 - \alpha$, and where $\alpha = R^2$. To do that, we differentiate $S(\alpha, u_c, v_c)$.

$$\frac{\partial S}{\partial \alpha} = 2\sum_i g(u_i, v_i)\,\frac{\partial g}{\partial \alpha}(u_i, v_i) \tag{3.32}$$

$$= -2\sum_i g(u_i, v_i) \tag{3.33}$$

Thus $\partial S/\partial\alpha = 0$ iff

$$\sum_i g(u_i, v_i) = 0 \tag{3.34}$$

Continuing, we have

$$\frac{\partial S}{\partial u_c} = 2\sum_i g(u_i, v_i)\,\frac{\partial g}{\partial u_c}(u_i, v_i) \tag{3.35}$$

$$= 2\sum_i g(u_i, v_i)\,2(u_i - u_c)(-1) \tag{3.36}$$

$$= -4\sum_i (u_i - u_c)\,g(u_i, v_i) \tag{3.37}$$

$$= -4\sum_i u_i\,g(u_i, v_i) + 4\,u_c\,\underbrace{\sum_i g(u_i, v_i)}_{=\,0 \text{ by } (3.34)} \tag{3.38}$$

Thus, in the presence of Eq. (3.34), $\partial S/\partial u_c = 0$ holds iff

$$\sum_i u_i\,g(u_i, v_i) = 0 \tag{3.39}$$

Similarly, requiring $\partial S/\partial v_c = 0$ gives

$$\sum_i v_i\,g(u_i, v_i) = 0 \tag{3.40}$$

Expanding Eq. (3.39) gives

$$\sum_i u_i\left[\,u_i^2 - 2\,u_i\,u_c + u_c^2 + v_i^2 - 2\,v_i\,v_c + v_c^2 - \alpha\,\right] = 0 \tag{3.41}$$

Defining $S_u = \sum_i u_i$, $S_{uu} = \sum_i u_i^2$, *etc.*, we can rewrite this as

$$S_{uuu} - 2\,u_c\,S_{uu} + u_c^2\,S_u + S_{uvv} - 2\,v_c\,S_{uv} + v_c^2\,S_u - \alpha\,S_u = 0 \tag{3.42}$$

Since $S_u = 0$, this simplifies to

$$u_c\,S_{uu} + v_c\,S_{uv} = \frac{1}{2}\,(S_{uuu} + S_{uvv}) \tag{3.43}$$

In a similar fashion, expanding (3.40) and using $S_v = 0$ gives

$$u_c\,S_{uv} + v_c\,S_{vv} = \frac{1}{2}\,(S_{vvv} + S_{vuu}) \tag{3.44}$$

Solving Eq. (3.43) and Eq. (3.44) simultaneously gives (u_c, v_c). Then the center (x_c, y_c) of the circle in the original coordinate system is $(x_c, y_c) = (u_c + \overline{x}, v_c + \overline{y})$.

To find the radius R, expand Eq. (3.34):

$$\sum_i \left[\, u_i^2 - 2\,u_i\,u_c + u_c^2 + v_i^2 - 2\,v_i\,v_c + v_c^2 - \alpha \,\right] = 0 \tag{3.45}$$

Using $S_u = S_v = 0$ again, we get

$$N\,\left(\, u_c^2 + v_c^2 - \alpha \,\right) + S_{uu} + S_{vv} = 0 \tag{3.46}$$

Thus

$$\alpha = u_c^2 + v_c^2 + \frac{S_{uu} + S_{vv}}{N} \tag{3.47}$$

and, of course, $R = \sqrt{\alpha}$.

Example : Let's take a few points from the parabola $y = x^2$ and fit a circle to them. Table 3 shows the (x, y) points used, as well as the corresponding (u, v) values.

i	x_i	y_i	u_i	v_i
0	0.00	0.00	-1.50	-3.25
1	0.50	0.25	-1.00	-3.00
2	1.00	1.00	-0.50	-2.25
3	1.50	2.25	0.00	-1.00
4	2.00	4.00	0.50	0.75
5	2.50	6.25	1.00	3.00
6	3.00	9.00	1.50	5.75

Table 3 Points used in circle-fit example

Here we have $N = 7$, $\overline{x} = 1.5$, and $\overline{y} = 3.25$. Also, $S_{uu} = 7$, $S_{uv} = 21$, $S_{vv} = 68.25$, $S_{uuu} = 0$, $S_{vvv} = 143.812$, $S_{uvv} = 31.5$, $S_{vuu} = 5.25$. Thus, using Eq. (3.43) and Eq. (3.44), we have the following 2×2 linear system for (u_c, v_c):

$$\begin{bmatrix} 7 & 21 \\ 21 & 68.25 \end{bmatrix} \begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} 15.75 \\ 74.531 \end{bmatrix} \tag{3.48}$$

Solving this system gives $(u_c, v_c) = (-13.339, 5.196)$, and thus $(x_c, y_c) = (u_c + \overline{x}, v_c + \overline{y}) = (-11.839, 8.446)$. Substituting these values into Eq. (3.47) gives $\alpha = 215.689$, and hence $R = \sqrt{\alpha} = 14.686$. A plot of this example is shown in Figure 17.

Using this fitted circle, we can then say that the curvature of this set of points is

$$\kappa = 1/R = 1/14.686 = 6.809 \times 10^{-2}$$

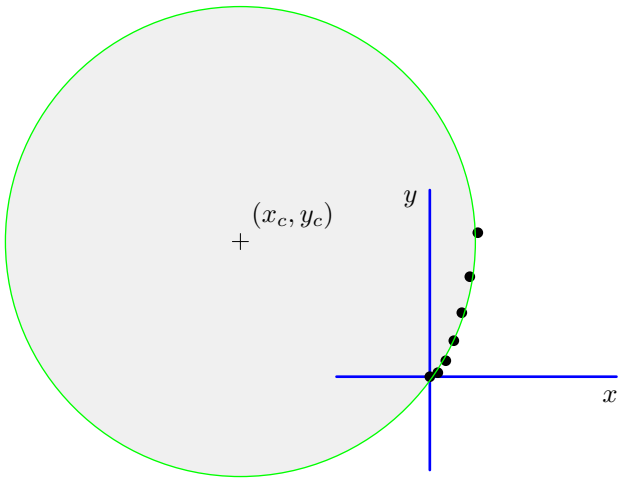


Figure 17 Result of circle-fit example

Curvature, like any other object attributes, can be used to restrict the verification analyses to those meteorological features that are of interest. For example, one possible way to restrict verification to front lines would be to examine only those objects with a high aspect ratio and a low curvature.

3.4.5 Convex Hull

What is a convex region? Unlike many mathematical definitions which are complicated and abstract, the definition of a convex region is very simple and intuitive. A region is convex if, whenever you select two points from the region, the straight line segment joining the points lies entirely inside the region. For example, the elliptical region on the left side of Figure 18 is convex. Choose any two points you like inside the ellipse, and the straight line segment joining them will lie entirely inside the ellipse. The star-shaped

region in the middle of Figure 18 is *not* convex. A straight line joining the indicated points A and B will not lie completely inside the star.

The *convex hull* of a region is the smallest convex region that contains the given region. A good way to mentally picture the convex hull of a region is to imagine a rubber band wrapped around the region. The area inside the rubber band is the convex hull. For example, the convex hull of the star on the right in Figure 18 is the region inside the pentagon outlined in green.

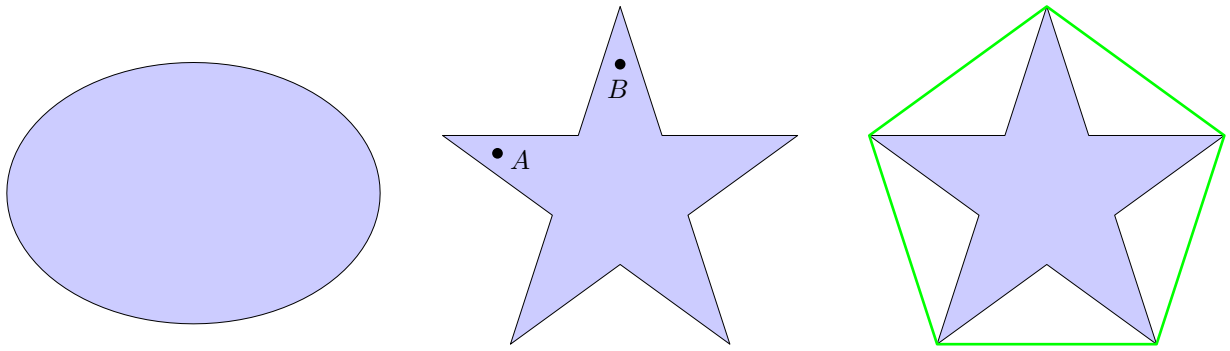


Figure 18 Illustration of convexity

MODE uses convex hulls for several things. One is the calculation of a single-object attribute called *complexity* (discussed later). Another is the pair-object attribute *convex hull distance*, the distance between the convex hulls of the two objects.

3.5 Area Attributes

Area measures are directly related to object size and (for pairs) amount of overlap. In MODE, the area of an object is simply a count of the number of grid squares the object covers. MODE has several area measures for pairs of objects which also reflect a count of grid squares. For example, given two objects that may or may not overlap, the *union area* is the area that is inside one of the objects or the other (or both). *Intersection area* is the area inside both objects simultaneously. If the two objects do not overlap, then this area is zero. The *symmetric difference* is the area that's inside at least one of the objects, but not inside both. Symmetric difference can be a good indicator of forecast quality, since in order for the symmetric difference to be small, the two objects must nearly coincide. This means the forecast must have correctly predicted the location, size and orientation of the object.

Figure 19 illustrates these different area measures. In the top of the figure, a forecast object is outlined in blue, and an observed object is outlined in green. In the bottom of the figure, different area measures are indicated by gray shading.

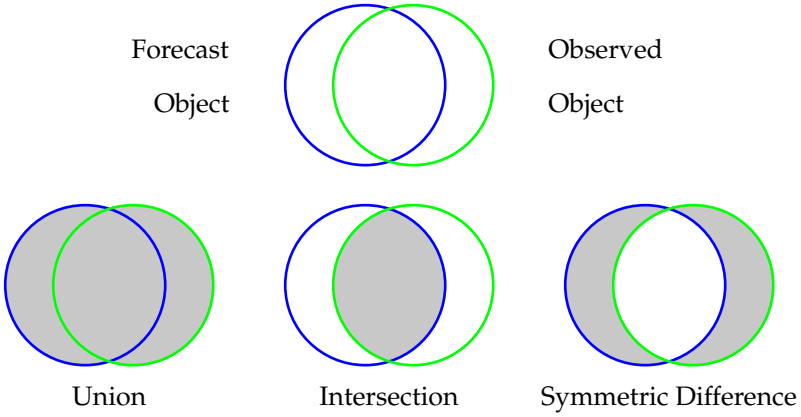


Figure 19 Area attributes

3.5.1 Complexity

Object *complexity* attempts to measure how simple the internal structure of an object is—does the object have holes, does it have appendages; in short, is the object complicated or not? MODE measures this by looking at the difference between the area of the object and the area of its convex hull. More precisely,

$$\text{Complexity} = \frac{A_{\text{hull}} - A_{\text{object}}}{A_{\text{hull}}} \tag{3.49}$$

We have $0 \leq \text{complexity} < 1$, and the complexity is zero if and only if the object is convex. An illustration of this attribute is given in Figure 20, which shows various objects in blue, along with their convex hulls (indicated by a green outline). The complexity values calculated for the objects in parts (a), (b), (c) and (d) of the figure are, respectively, 0.2, 0.26, 0.58 and 0.79.

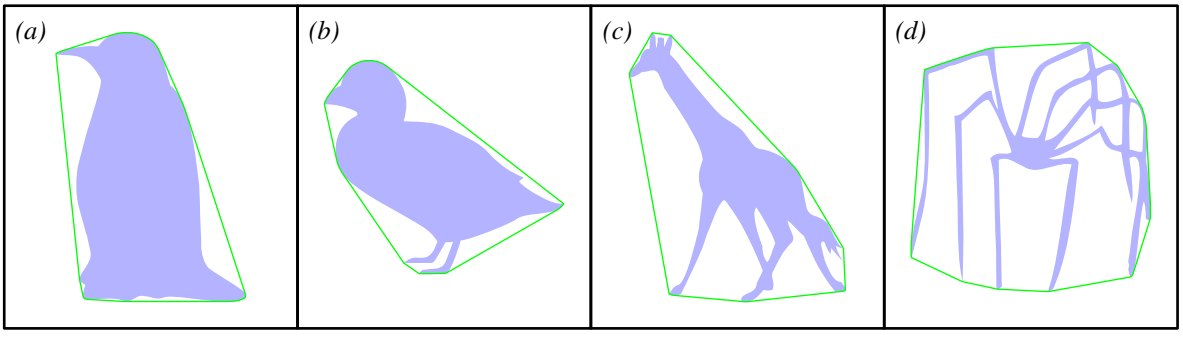


Figure 20 Object Complexity

3.6 Distance Attributes

Distance attributes are pair attributes that give straight line distances measured with respect to the grid that the data are on. Of course, the phrase "straight line" should be taken with a grain of salt, since every grid is based on a map projection, and a path on the Earth that looks straight in one projection will not look straight if viewed in another projection. Still, in most cases there should be a rough correspondence between distances in grid units and true distances on the Earth.

Centroid distance is the distance (in grid units, as explained above) between the centroids of two objects. If one object has centroid grid coordinates (x_1, y_1) and the other has centroid coordinates (x_2, y_2), then the centroid distance is computed via the usual Cartesian formula:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.50}$$

Note that if the two objects are small or not too complicated, then centroid distance may give valuable information regarding their degree of separation. If the objects are large or complex, then centroid distance may not have much meaning.

Convex hull distance is the shortest distance between a point on one object's convex hull and a point on the other object's convex hull. This distance will be zero of the two hulls intersect. Note that it's possible for the convex hulls of two objects to overlap even if the objects themselves don't.

Boundary distance is similar to convex hull distance, except that the object boundaries are used instead of the convex hulls. This distance will be zero if the two objects touch.

3.7 Intensity Attributes

So far most, if not all, of the object attributes discussed here have been purely geometrical. They could just as well have been calculated from the mask field, rather than the object field. However, the raw data values (also called "intensity values") inside an object can be used to create a class of intensity-based attributes that are at least as important for judging forecast quality as the geometrical ones.

In particular, *intensity percentiles* can be used summarize the raw data values inside the objects. (Note that in the case or precipitation data, MODE uses only the nonzero data values to calculate these percentiles.) Currently, MODE calculates the 10^{th}, 25^{th}, 50^{th}, 75^{th} and 90^{th} intensity percentiles, as well as one user-specified percentile. In addition, MODE writes out the sum of the intensity values inside each object.

3.8 Ratio Attributes

Ratio attributes are pair attributes that involve ratios of an attribute for a forecast object and the same attribute for an observed object. Because they use ratios of values of the *same* attribute, they are dimensionless, and hence independent of the resolution of the grid.

Area ratio is defined using the area A_f of the forecast object and the area A_o of the observed object. Specifically,

$$\text{Area ratio} = \frac{\min(A_f, A_o)}{\max(A_f, A_o)}$$

Similarly, *complexity ratio* is given by

$$\frac{\min(C_f, C_o)}{\max(C_f, C_o)}$$

where C_f and C_o are the complexity values of the forecast and observed objects.

Intersection over area: this poorly-named attribute should really be called "intersection over minimum" because it's the ratio of the intersection area of the two objects to the smaller of the two object areas.

Finally, *percentile intensity ratio* is given by

$$\frac{\min(P_f, P_o)}{\max(P_f, P_o)}$$

where P_f is the user-defined percentile intensity value inside the forecast object, and P_o is the user-defined percentile value inside the observed object.

3.9 Example

Table 1, page 18 shows results for some single and pair attributes for the two clusters shown in Figure 4, page 17. For example, the centroid distances (a pair object attribute) for the two pairs of clusters are 31.4 and 20.9 grid units; and the forecast areas (a single object attribute) are 3802 and 3187 grid squares. Note that many pair attributes can be derived from the single attributes. For example, the differences in 0.50th intensity values for the forecast and observed objects can be directly computed from the individual median intensity values.

Results in Figure 4 indicate that the forecast area for cluster # 1 was somewhat too large, whereas the forecast area for cluster # 2 was small compared to the observed area. In addition, the forecast precipitation intensity values were too small for cluster # 2 compared to the intensity values for the observed cluster.

3.10 Summary

This section has provided the mathematical background and underpinnings for the way MODE computes the values of a wide variety of attributes of objects. These attributes are important for both the object matching and merging process (discussed in subsequent sections) and for use in evaluating the ability of a forecast or a set of forecasts to predict observed object characteristics. Many object attributes are calculated by MODE, the majority of them being purely geometric in nature. Some of them are used internally by the fuzzy engine, while others are provided for the user's benefit. Future releases or extensions of MODE may include additional attributes.

4

Fuzzy Logic

4.1 Overview

Object attributes are interesting in themselves, and statistical summaries and climatologies of such attributes are useful for characterizing meteorological phenomena, and also give a method of doing long-term comparisons of forecasts with observations. The main use of attributes in MODE however, is to enable the process of associating certain forecast and observed objects with each other. This process is done using fuzzy logic. We refer the fuzzy logic part of MODE as the *fuzzy engine.*

Only pair attributes are used by the fuzzy engine, so in this section all uses of the term *attributes* refer to pair attributes, not single attributes.

4.2 Interest Maps

MODE needs to be told what values of a particular object attribute are interesting. The relationship between an attribute and its interest value is a function (called an *interest map*) that takes the attribute value as an argument and returns a number called the *interest value.*

Interest values are scalars in the range of 0 to 1. Zero represents no interest, while one represents the highest possible interest value. (Note: generally interest maps used in fuzzy logic take values either in the range $[0, 1]$ or $[-1, 1]$. All MODE interest maps take values in $[0, 1]$.)

Figure 21 shows graphs for some hypothetical interest maps. In Figure 21(a) we have the default interest map that MODE uses for centroid distance. Small values of this attribute are of high interest, while for larger values the interest drops off, eventually becoming zero. This interest map expresses the idea that having forecast and observation objects that are close together is more significant for matching than objects that are far apart.

Not all interest maps have this profile. In Figure 21(b) we see an interest map with essentially the opposite features. This interest map is for intersection area—the area inside both the forecast and observed object. For this map, high values of the attribute are interesting while low values are not. This is just the opposite of the interest map for centroid distance.

Still another profile is shown in Figure 21(c), demonstrating that interest maps need not be monotone. This hypothetical interest map is for the ratio of the area of the forecast object to that of the observed object. In this case, values close to one are of high interest, while the interest values fall off as the ratio moves away from one in *either* direction.

As a final example, we note that the interest map for an attribute may be a function of more than one variable. Figure 21(d) shows an interest map that gives high interest only if the given attribute (x-axis) takes small values and *simultaneously* another attribute (y-axis) takes high values. So far, MODE has not needed such multivariable interest maps, but the general fuzzy logic framework allows for them, should they ever be needed.

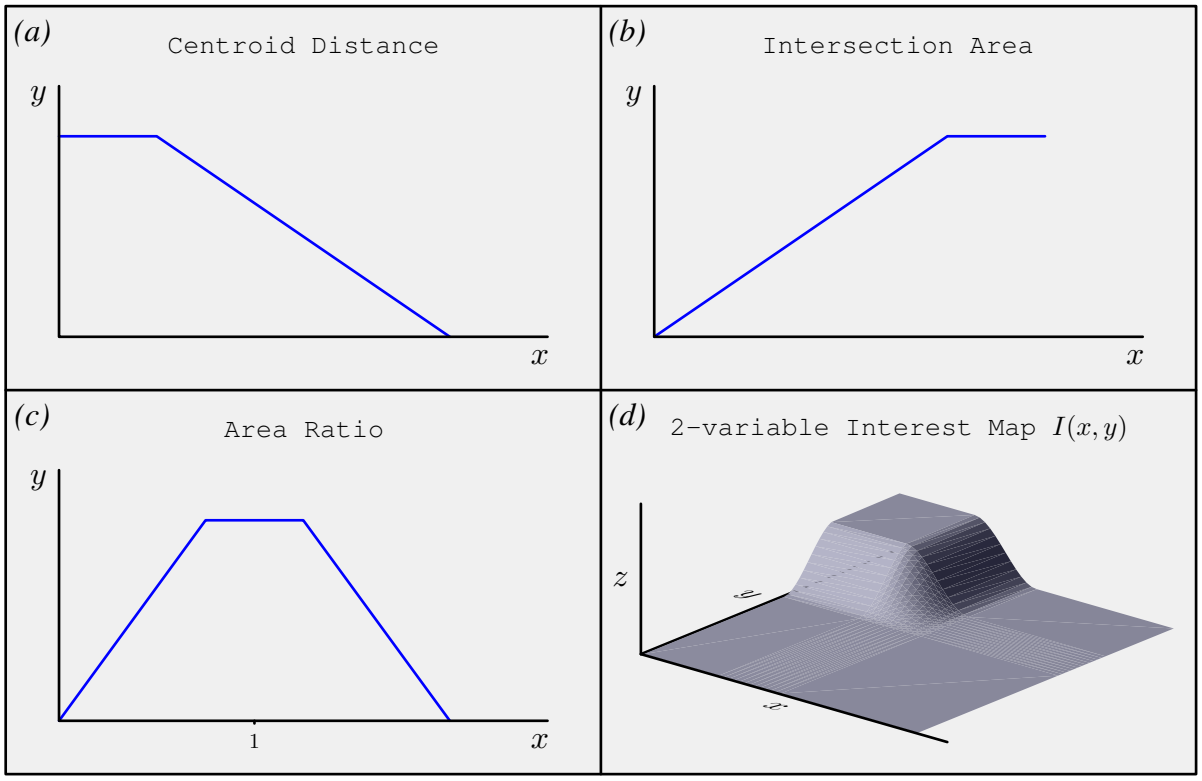


Figure 21 Example interest maps

4.3 Confidence Maps

In addition to assigning an interest value to each attribute via its interest map, we may also express a level of confidence in the computed value of an attribute. Confidence maps take values in the range $[0, 1]$, with zero expressing no confidence and one expressing complete confidence.

As an example, consider a wind direction sensor, such as a wind vane. If the wind speed drops below a certain value, the wind will not be able to move the detector, and so the wind direction values reported

by the sensor become meaningless. Below that critical wind speed, we simply don't believe the reported values of wind direction, and for wind speeds above but close to the critical value, we may be in some degree skeptical of the reported wind direction. This example illustrates the fact that the level of confidence we have in one quantity (or attribute), in this case wind direction, may well depend on the value of some other quantity (*e.g.*, wind speed). While the general fuzzy logic framework allows for all attributes to have associated confidence maps, currently MODE only uses two confidence maps: one for axis angle and another for centroid distance.

There are cases where the axis of an object is not well defined. For example, if an object is nearly square or nearly circular, an axis angle can certainly be calculated, but a very small change in the shape of the object can greatly affect the direction of the axis. The way MODE detects this situation is by looking at the value of another attribute: the aspect ratio. To define the aspect ratio of an object, we draw a rectangle, aligned with the object axis, around the object and shrink the rectangle as much as possible while still requiring it to enclose the object. The ratio of the lengths of the sides of the rectangle is the object's aspect ratio. For values of this ratio that are close to one, the confidence is small, and the confidence grows as the ratio departs from one in either direction. Explicitly, the confidence map for axis angle difference is given by the following equation:

$$C_{\text{axis}}(r) = \left[\frac{(r-1)^2}{r^2+1}\right]^{\beta} \tag{4.1}$$

Here, r is the aspect ratio. This equation was obtained by requiring four properties: $C_{\text{axis}}(1) = 0$, $C_{\text{axis}}(0) = 1$, $\lim_{r\to\infty} C_{\text{axis}}(r) = 1$ and $C_{\text{axis}}(1/r) = C_{\text{axis}}(r)$. This last condition is imposed so that it doesn't matter whether aspect ratio is defined as width $\div$ height, or *vice versa*. The exponent β was originally a tunable parameter. Currently MODE uses $\beta = 0.3$. Figure 22 shows a plot of $C_{\text{axis}}(r)$ versus r for several different values of β.

The reader may wonder why we don't simply use a piecewise linear function for C_{axis}, like we do for the interest maps? The answer is that piecewise linear functions are only useful if the domain of the function is bounded, or at least if the function has compact support, neither of which is true for C_{axis}, since the aspect ratio r can take on any positive real value.

4.4 Weights

Interest maps give information on which values of a particular attribute are more interesting than other values of the same attribute. *Weights* give information on which attributes are more interesting or important than other attributes. Unlike interest maps or confidence maps, weights are simple scalars, not functions. Assigning a large weight to a particular attribute tells MODE that this attribute is more important than others for the purposes of matching and merging. Say, for example, for a particular verification situation, overlap area is the major indicator of a good match between forecast and observation. In that case, the user would give overlap area a large weight relative to the weights assigned to the other attributes.

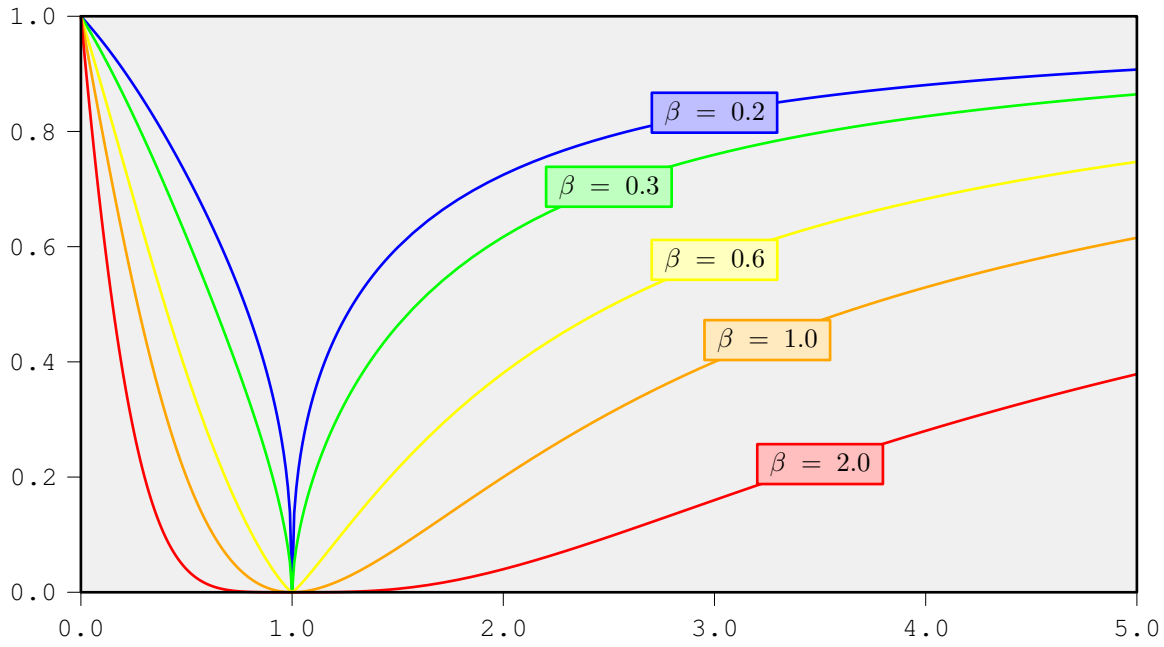


Figure 22 Plot of object axis confidence map $C_{\text{axis}}(r)$

In MODE, all weights are nonnegative. Assigning a weight of zero to a particular attribute essentially "turns off" that attribute, and values for that attribute will have no effect on matching or merging.

4.5 Total Interest

Once the attribute values and the associated weights, interest maps and confidence maps are in hand, these quantities are combined into a single scalar value, the *total interest*. If we denote the value of the i^{th} attribute by α_i, and let $\alpha = (\alpha_1, \ldots, \alpha_n)$ denote the n-tuple of all attributes, then the total interest T is given by

$$T(\alpha) = \frac{\sum_i w_i \, C_i(\alpha) \, I_i(\alpha)}{\sum_i w_i \, C_i(\alpha)} \tag{4.2}$$

In this equation, w_i is the weight, C_i is the confidence map, and I_i is the interest map associated to α_i.

In order to gain a little insight into this equation, let's rearrange it a little. Let the attribute values be fixed, and define $S = \sum_i w_i C_i(\alpha)$, and $\mu_i = w_i C_i(\alpha)/S$. Then, suppressing the dependence on α, we have $T = \sum_i \mu_i I_i$. From this we see that the total interest is simply a weighted average of the individual interest values I_i. The coefficients in this weighted average add up to one: $\sum_i \mu_i = 1$.

4.6 Example

For the example in Figure 4, page 17, total interest values were computed for all pairs of MODE simple objects (identified by numbers in parts (e) and (f) of that figure). For this collection of objects, the largest interest values are shown in Table 4.

Pair Number	Forecast Object Number	Observed Object Number	Total Interest Value
1	1	1	0.93
2	5	7	0.92
3	2	1	0.88
4	4	1	0.86
5	5	1	0.67
6	2	7	0.62

Table 4 Total interest values for MODE example

The interest values in Table 4 indicate that the best match between simple objects is between forecast object # 1 and observed object # 1, and that good matches are also achieved for pairs # 2, 3, and 4. However, the total interest values decrease quite a bit for other pairs. Although the total interest threshold used by MODE to identify matches can be adjusted by the user, a typical threshold (applied here) is 0.70. Thus, forecast simple object # 1 is found to match observed simple object # 1; forecast simple object # 5 is found to match observed simple object # 7; forecast simple object # 2 is found to match observed simple object # 1; and forecast simple object # 4 is found to match observed simple object # 1. Note that observed simple object # 1 is found to be a good match for three different forecast simple objects (# 1, 3, and 4).

4.7 Summary

This section has shown how fuzzy logic can be applied to identify similarities between forecast and observed objects. Fuzzy logic is a valuable tool for this application because it is able to (*i*) make use of human knowledge and experience to replicate human judgements of similarities between objects; and (*ii*) be adapted as the types of objects that are of interest vary. For example, in some situations matching the size of areas may be most important, whereas in others matching the object locations may be of most relevance. In both cases, the desired outcome could be achieved by adjusting the weights or revising interest maps.

5

Matching & Merging

5.1 Overview

Individual objects are often of less interest to users than collections of objects that are related to each other in some way. When objects are associated with other objects, additional information becomes available. MODE associates objects via *matching* and *merging*, using the fuzzy engine with user-selected criteria.

The most common example is the association of an observed object with a forecast object. This is called *matching*. With a matched forecast and observation pair, a user can determine how well the forecast and observed objects match and in exactly what ways a forecast could improve. For example, users can determine if the forecast object is smaller than the associated (*i.e.*, matched) observed object and whether it is displaced spatially. It can also be of interest to cluster together or *merge* objects in a single field. For example, if several thunderstorm objects are near each other, they may represent a single phenomenon such as a squall line. Thus, it makes sense to treat these individual storms as a single cluster object.

The goal of the fuzzy logic algorithm is to mimic the human ability to identify objects that go together, while removing subjectivity. Forecast and observed objects that have been matched are referred to as *pairs*, while objects that have been merged are referred to as *clusters*.

MODE uses the object attributes along with interest and confidence maps as inputs to a fuzzy logic engine that performs matching and merging. *Merging* refers to associating objects in the same field, while *matching* refers to associating objects in different fields. Several methods are available for both matching and merging. We'll discuss these approaches in detail later in this section.

Though conceptually distinct, these two operations need not be independent at the implementation level, since some types of matching will make use of merging algorithms, and *vice versa.* For example, one simple way to do merging in, say, the forecast field, given that matching algorithms are implemented, is to make a copy of the forecast field and then perform matching in the two duplicate forecast fields. Conversely, information from the matching routines can feed back to the merging routines as well. If, for example, two different forecast objects are matched to the same observed object, that might be interpreted as saying that the two forecast objects should be merged.

In practice, this interconnectedness of the matching and merging processes forces an iterative approach, where multiple passes are performed over the set of objects in both fields. The iteration continues until there are no further changes in the collection of cluster objects in either field.

5.2 Simple Objects

The MODE fuzzy engine makes a distinction between simple objects and cluster (or composite) objects. Simple objects are just the connected subsets of the grid where the mask field has value one. For example, in the left half of Figure 10 on page 27, we have a mask field (indicated by the blue squares) that is in 3 connected pieces; therefore there are 3 simple objects in that field. In contrast to the cluster objects discussed in the next section, the simple objects in both the forecast and observed fields are known before the calculation of object attributes begins. The interest map and confidence map values calculated from the simple object attributes are the input values to the fuzzy engine.

5.3 Cluster Objects

Cluster objects are the result of the merging process. Conceptually, we think of a cluster object as one object that just happens to be in several pieces. As mentioned previously, both matching and merging can be iterative processes, involving several passes over the collection of objects. For example, if one merging pass determines that forecast objects #1 and #2 should be merged, and also that forecast objects #2 and #3 should be merged, then the next pass will merge all three objects into one cluster object. Thus cluster objects from one pass can themselves be merged into larger clusters on the next pass. On the other hand, if a simple object is not merged with any others, then that object is a cluster consisting of one object.

Attributes are calculated for cluster objects, and are written out in MODE's `ascii` output file, but interest maps are not applied, nor are these values used in any way by the fuzzy engine.

5.4 Graphs

The matching and merging algorithms can best be explained by using some ideas from graph theory. A *graph* is a collection of objects (called *nodes*), together with a set of *edges*, connecting pairs of nodes. As an illustration, Figure 23 shows a graph with 10 nodes, labelled with integers running from 0 to 9, along with some edges, indicated by straight line segments. There are edges connecting nodes 1 and 2, for example, and also connecting nodes 5 and 7.

Related to the idea of an edge is the concept of a *path*. A path connecting two nodes is a sequence of edges, where the terminal node of each edge is the initial node of the next edge. Referring to Figure 23 again, we see that while there is no edge connecting nodes 0 and 1, there is a path connecting them. The path starts at node 0, and then travels along edges first to node 4, then to node 2 and then to node 1.

An example of graphs applied to object matching and merging is illustrated in Figure 24. In part (a) of the figure we see a hypothetical collection of simple objects in the forecast field. The objects are given

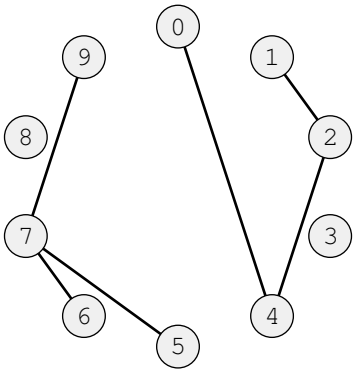


Figure 23 Basic graph

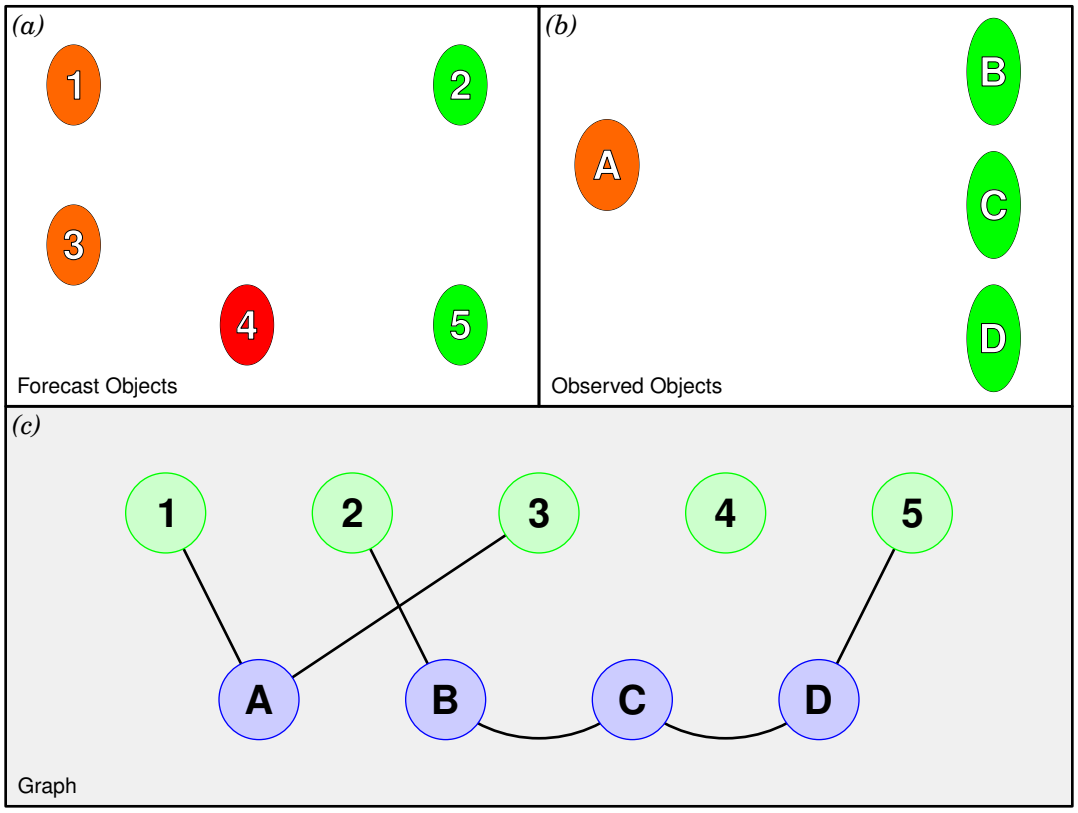


Figure 24 Match & Merge example graph

consecutive numbers as identifiers. In part (b) we see a collection of simple objects in the observed field. Here, the objects are labelled by letters to help distinguish them from forecast objects. Part (c) of the figure shows a graph with nodes labelled by forecast and observed identifiers. Forecast nodes are colored green and

observed nodes are colored blue.

For the sake of this simple example we'll just proceed intuitively. Given this object layout in the two fields, most people would probably say that object 2 should be matched to object B, object 5 should be matched to object D, Objects 1 and 3 should be matched to object A, and objects B, C and D should be merged into a cluster. These matches and merges are reflected in the graph edges shown in part (c) of the figure. Note that object 4 has no observed object matched to it, and no forecast object merged with it. We can consider it to be a false alarm.

These matches and merges can immediately be read off from the graph: objects in the same field (forecast or observed) are merged if there is a path connecting them that consists of nodes from the same field. Objects from different fields are matched if there is an edge connecting their nodes.

Up until now, most people would agree (or at least not strongly disagree) with the matches and merges we've specified in this example. Suppose now we ask the question "Should objects 1 and 3 be merged?" The answer is not as clear in this case and therefore people may well answer differently. Strictly speaking, there is no path connecting them that consists entirely of forecast nodes, so some people might say "No." On the other hand, if we relax the condition that the nodes in the path be all from the same field, then there *is* a path connecting objects 1 and 3 in the graph: start at 1, go to A and then to 3. So, under this relaxed merging condition, the answer would be "Yes," they should be merged.

This "relaxed" definition of merging is called "additional merging" in the following sections. The user may specify in the configuration file whether this additional merging is to be allowed in either field. In the end it depends on the type of verification analysis being done.

In our example, if additional merging is allowed in the forecast field, objects 1 and 3 will be merged into a cluster, which will be matched to object A. Object A, since it is merged with no other observed simple object, will become a cluster consisting of a single object. Also, if additional merging is allowed in the forecast field, then objects 2 and 5 will be merged even though they are quite widely separated in space, since there is a path in the graph connecting them, namely $2 \rightarrow \mathrm{B} \rightarrow \mathrm{C} \rightarrow \mathrm{D} \rightarrow 5$.

Internally, you can think of MODE as keeping track of an object graph, with nodes for each simple object in each of the two fields. Initially, it makes each simple object in each field into a cluster object consisting of that simple object alone. (It's allowed for a cluster to consist of only one object.) As it makes multiple passes over the collection of objects, it adds edges to the graph according to which matching and merging options have been specified by the user. Eventually, when MODE sees that further passes are adding no more edges to the graph, the iteration exits, and the final matches and merges are read off from the graph, with merges interpreted according to whether the relaxed or strict (*i.e.*, additional merging allowed or not) rule is in effect for that field.

To summarize, merging decisions are made as follows. **Question:** Are two given objects from the same field part of a cluster? **Answer:** If "additional merging" is not allowed, then there must be a path in the graph connecting the two objects that consists entirely of nodes from the same field. If "additional merging"

is allowed, then any path at all in the graph connecting the two objects will do.

Once the clusters in each field are determined, matching decisions are made as follows: **Question:** Are two given cluster objects from the different fields to be matched? **Answer:** If and only if there is an edge connecting one of the simple objects in one cluster to one of the simple objects in the other cluster.

As a final step, MODE as currently implemented does not allow unmatched clusters, so clusters that remain unmatched to anything after the matching and merging process is finished are broken back up into their constituent simple objects.

5.5 Merging Methods

MODE supports several different types of merging: double threshold merging and fuzzy engine merging. The two types can be turned on or off independently, resulting in four possibilities. If both are turned off, the merging step is simply skipped for that field. If both are turned on, then double threshold merging is performed first, followed by fuzzy engine merging. Let's examine the the various types of merging in more detail.

Double threshold merging is conceptually the simpler of the two methods. In addition to the convolution threshold used to resolve the given field into objects, the user can specify a second, lower, threshold for use in object merging. An example of this is shown in Figure 25. In that figure we see three objects that are separate at the higher convolution threshold (indicated by the darker color), but which bleed together at the lower threshold (lighter color). MODE can use this information to determine that the three original objects should be merged into a cluster.

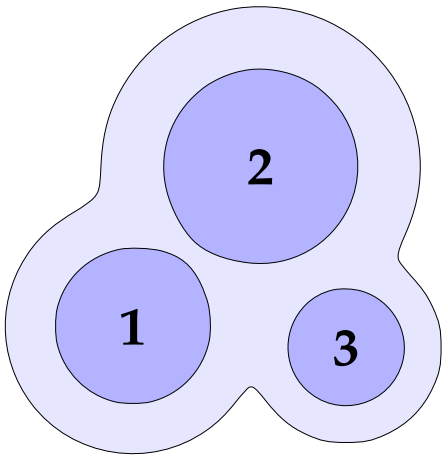


Figure 25 Double threshold merging

In the fuzzy engine merging method another, independent, instance of the fuzzy engine is created. The field (forecast or observed) is duplicated and given to the new engine as both a forecast and observed

field. The new engine then performs matching and merging according to the parameters set in a separate configuration file, called the *merge configuration file.* The new engine by default inherits all the interest maps and thresholds from the original engine. If desired, these can be overridden by using the `config_merge` command-line option.

5.6 Matching Methods

We now briefly discuss the matching methods that are available in MODE.

No matching With this option, no matching is performed. This is useful if all the user cares about is the object attributes and doesn't want to slow MODE down by running any of the matching algorithms.

Match & Merge Forecast & Observed. With this option, "additional merging" (as explained earlier) is permitted in both fields.

Match & Merge Forecast Only. With this option, additional merging is allowed only in the forecast field. This option can be sometimes useful when comparing multiple forecasts to the same observation field.

Match with No Additional Merging. With this option, no additional merging is allowed in either field. Thus, when this option is chosen, each object will match at most one object in the other field.

5.7 Example

As shown in Table 4, page 50, observed simple object 1 is matched to forecast simple objects 1, 2, and 4, and observed simple object 7 is matched to forecast simple object 5. Thus, for this example (with simple matching and merging applied), two clusters are formed. Cluster 1 consists of forecast objects 1, 2, and 4 and observed simple object 1; cluster 2 includes forecast simple object 5 and observed simple object 7. The total interest values for these two clusters (as shown in Table 1, page 18, are 0.92 and 0.95, respectively. Note that forecast simple object 3 is not matched to an observed simple object and thus is a false alarm. In addition, observed simple objects 2, 3, 4, 5, and 6 are also unmatched and thus are misses. The total areas of false alarms and misses (not shown here) can also be computed as a measure of these kinds of failures by a forecast.

5.8 Summary

This section has focused on the methods that MODE uses to define matches and merges between objects. A variety of combinations of parameters can be applied, with some more appropriate than others in identifying meaningful clusters of objects, depending on the kind of forecast situation of interest. Thus, the approach offers users a great deal of flexibility. Examples of the impacts of varying the MODE parameter settings are shown in the next section.

6

Tuning the Parameters

6.1 Overview

We now consider how to set values for the basic MODE configuration file parameters. We'll experiment with changing several of the MODE parameters to get a sense of their effect on object resolution and on the matching and merging processes. For many users, this is the most difficult part of using MODE — the program has many tunable parameters, with little in the way of orthogonality amongst them. It can be challenging to find reasonable values for these that give the sort of objects, matches and merges that are relevant for the task at hand. This is made worse by the fact that many users possess very little geometric intuition for things like convex hulls, symmetric differences and convolution filters (to name a few).

We will use a single example data set throughout this section, trying to adjust the MODE configuration file parameters to get results similar to what a human being would produce by examining the forecast and observed fields visually.

In the next two sections, we'll look at how changing the values of convolution radius and threshold affect the way objects are resolved in the raw data field. We'll use the precipitation fields shown in Figure 26 for our examples. Throughout this section, the units for both raw precipitation data and for thresholds is millimeters.

MODE has many tunable parameters—too many to cover in a short section like this. Therefore we'll content ourselves with looking only at the convolution radius and threshold.

6.2 Radius

As explained in an earlier section, the convolution radius R is effectively a radius of influence. To get the convolved data field value at some point $P(x, y)$ in the grid, essentially all of the raw data in a circle of radius R and centered at P is averaged. Thus the overall effect is to smooth the precipitation data. High spatial frequencies are attenuated; rain areas are "flattened" and spread out.

This is illustrated in Figure 27. We show the result of the convolution-thresholding step for the observed field in Figure 26b with the threshold fixed at 2, and for several values of the convolution radius R. Notice

that for small values of R the objects retain much of the spatial detail of the original field. Thresholding with a small convolution radius is not much different than simply thresholding the raw field. As the radius R increases, the resulting objects become more smooth-featured, and there are fewer of them. At the largest radius, only objects corresponding to the largest and most intense rain areas are present.

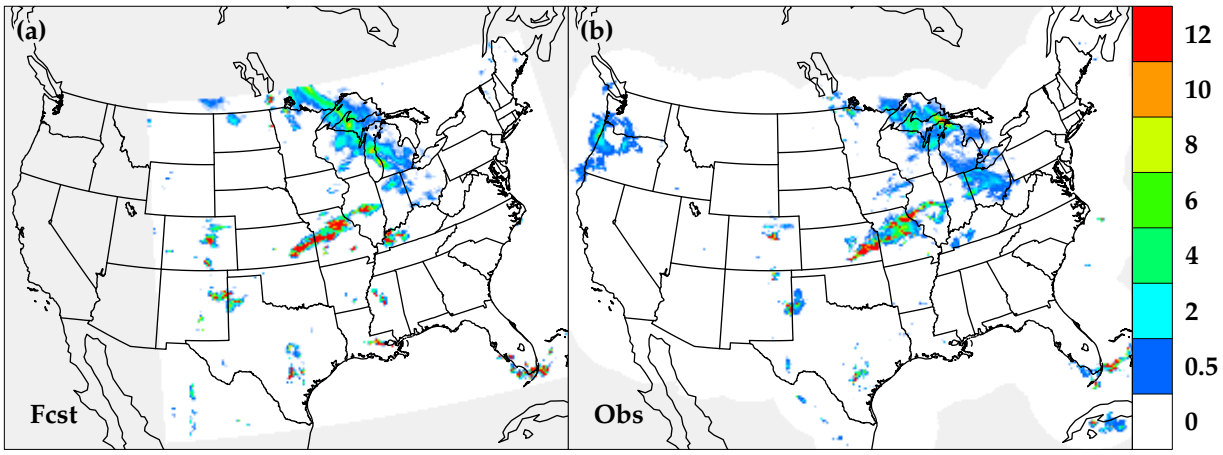


Figure 26 Raw data fields for radius and threshold tuning example.

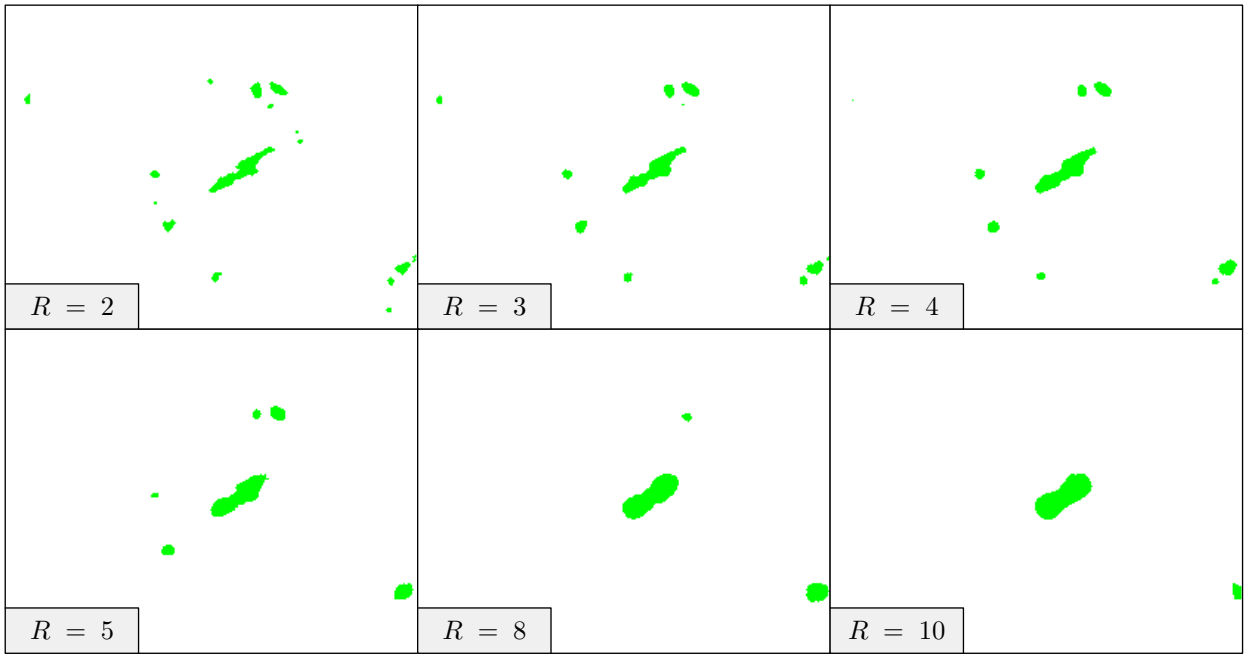


Figure 27 Effect of varying convolution radius

6.3 Threshold

Besides the convolution radius, the threshold is the other MODE parameter that determines how object are resolved in the raw data field. In order to set this parameter intelligently, a good first step is to look at your data. Percentiles for the nonzero data values taken from the two data fields shown in Figure 26 are given in Table 5. Note that these values are percentiles for the *raw* data, not the convolved data. While the threshold is actually applied to the convolved field, examining the distribution of raw data values often provides valuable guidance for choosing the threshold.

Percentile	5	10	20	30	40	50	60	70	80	90	95
Obs	0.04	0.08	0.20	0.29	0.42	0.50	0.65	1.00	1.57	3.83	7.97
Fcst	0.10	0.10	0.20	0.30	0.50	0.90	1.30	2.10	3.60	7.60	18.20

Table 5 Percentiles for test-case data

In Figure 28, we show the effect of choosing several thresholds after the raw data was convolved with a radius of 2. The objects become smaller and fewer in number as the threshold is increased, which should be intuitively clear.

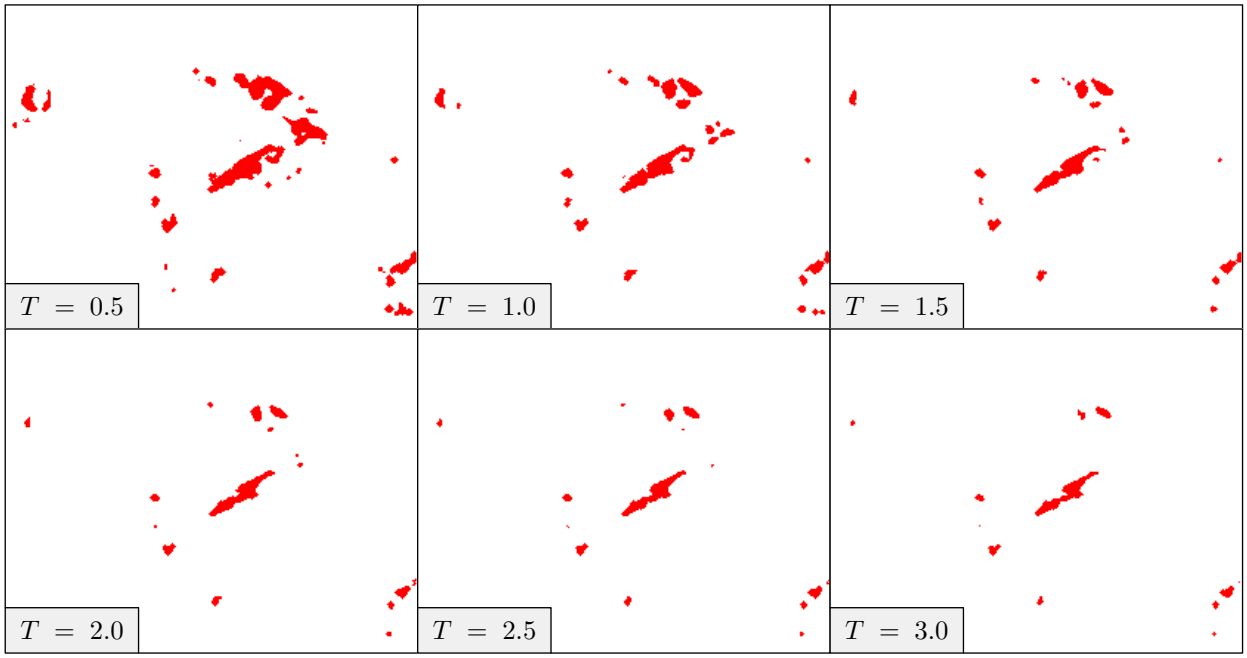


Figure 28 Effect of varying object threshold

These simple experiments with convolution radius and threshold demonstrate some general principles we can use to help guide the choice of values for these parameters. The user should have some idea of what sort of spatial features she's interested in. (Recall our original definition of objects as *regions of interest.*) Figure 29 gives some idea of the characteristics of objects typically produced by low *vs.* high values of radius and threshold. Low values of threshold give fairly large objects, while high thresholds produce smaller objects. High values of convolution radius tend to produce objects with smooth outlines, while low values of radius give objects with more detailed outlines. Finally, low values of radius and threshold together produce many objects, and the number of objects decreases as either parameter is increased.

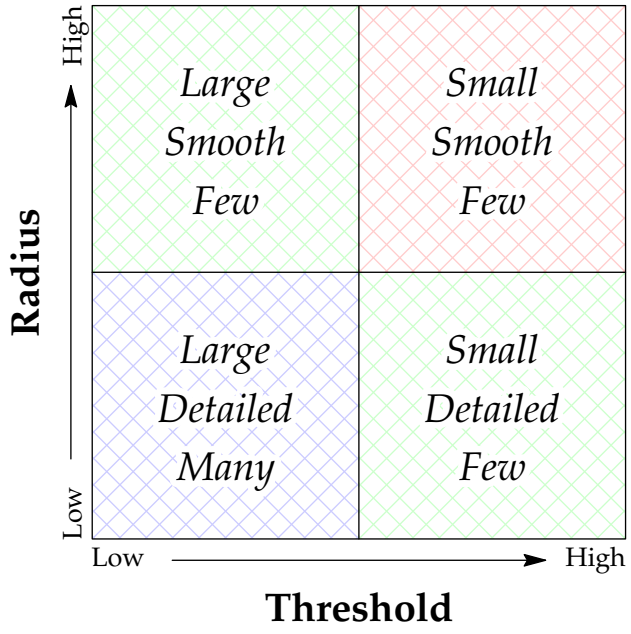


Figure 29 Varying the convolution radius and threshold

6.4 Using Other Filters

Initially, when the ideas behind MODE were being researched and experimented with, the question of which convolution filter to use came up. There were many possible choices, and we initially tried quite a few different ones before settling on the simple cylindrical filter described earlier in this document, including conical filters and gaussian filters, to name a few. Interestingly, it turned out not to matter very much which filter was used—the resulting objects were always very similar. Of course there were some differences depending on which filter was used, but the differences were much smaller than expected. The large-scale features of the objects were not much affected. So we went for what seemed the simplest option: the cylindrical filter.

The fact that the resulting objects don't depend much on what filter is used is reassuring. It says that

the objects are not simply an artifact of what filter we use—they have an intrinsic meaning. They are not created by our chosen algorithm for resolving them.

7

Using MODE Output

7.1 Overview

A typical MODE run produces four output files: a PostScript graphics file (filename suffix "`.ps`"), an `ascii` object attributes file (suffix "`obj.txt`"), an `ascii` contingency table counts file (suffix "`cts.txt`") and a `netCDF` data file (suffix "`.nc`"). These output files together contain all the information produced by MODE during the run. They can be used to perform specialized or exploratory analyses that are not provided by MODE. [Note that MODE is frequently updated in new releases of MET, so users should review the *MET Users' Guide* for up-to-date information about MODE output.]

7.2 NetCDF File

The MODE `netCDF` output file contains information that would be too cumbersome to put into an `ascii` text file. Also, some things, such as convex hull boundaries, have different numbers of data points for different objects, and hence do not really lend themselves to being written in tabular form.

Note: The configuration file used to run MODE enables the user to determine which information is written out to the `netCDF` file. Therefore some of the things described in this section may not be present in the user's `netCDF` file, if the chosen configuration options are switched off for that particular output. What follows is a description of some (not all!) of the `netCDF` file contents when the default configuration options are used.

The raw data values for both the forecast and observed fields are written to the file. This can be useful for accessing the raw data fields if the original input data files are in a difficult-to-read format, such as `GRIB`. Simple and cluster object numbers at each grid point are also included. Together this information can be used to generate plots (maps) of individual cases. Figure 3 on page 16 was produced in this way.

The total numbers of simple forecast and simple observed objects are included. The total number of cluster objects is written out as well, though in the case of cluster object numbers there is no distinction between forecast and observed. Convex hull outlines for clusters are also included. This information was used in drawing the hull outlines in parts (c) and (d) of Figure 3.

Convolution threshold and radius values specified in the configuration file for both forecast and observed fields are written to this file. Finally, specifications for the spacing, orientation and map projection used in the data grid are given, although (as of this writing) the information is in a non-standard form. In future releases of MODE (and MET generally) this information will be in CF-compliant form.

7.3 Object Attributes File

The `ascii` attributes file is probably the most-used MODE output file. All object attribute information calculated by MODE is written out to this file, even attributes not used by the fuzzy engine. Simple and cluster attributes as well as single and pair attributes are written out. Simple object attributes such as area and axis angle are calculated for cluster objects and written out. This file also contains timestamp information (*e.g.*, valid, lead) for both forecast and observed data files. The file is in a tabular format with (currently) over 50 columns. One cautionary note to the anyone writing code to parse these files: the column widths are likely to vary from one MODE run to the next. MODE simply makes each column as wide as it needs to be in each file. This means that simple formatted read statements (such as are commonly used in `FORTRAN`) will not work.

These attribute data files are very suitable for loading into a database. Once this has been done, database queries based on object size, intensity, or indeed any object attribute can be made. One could, for example, examine the range of values of some object attributes subject to restrictions on the values of one or more other attributes. Filtering by time, date, or season would also be possible. It would also be possible to restrict the query to matched or unmatched objects.

7.4 Contingency Table File

The contingency table statistics file contains information on object hits, misses and false alarms interpreted on a gridpoint-by-gridpoint basis. Essentially, all information on object matches and merges is thrown away, and only whether or not each gridpoint is part of an object is counted. The information is presented as traditional 2×2 contingency table entries: hits, misses, false alarms and correct negatives. In addition to the bare counts, some commonly used contingency table statistics are calculated and given for the user's convenience.

These values from multiple MODE runs can be loaded into a database for further analyses such as aggregation, filtering or establishing a climatology for objects.

7.5 PostScript File

The PostScript graphics file provides visual information about the MODE run. Basic plots of the raw data fields are shown, in addition to plots of both simple and cluster objects, showing the objects themselves and their assigned object numbers. Matches and merges are indicated. Total interest values for simple object

pairs are given so the user can see which pairs made the total interest cutoff threshold and which pairs did not.

For the user's convenience, key parameters from the configuration file are also shown, as well as summaries of certain object attributes from the `ascii` output files.

These plots are useful as a visual check that MODE is capturing those regions of interest in the raw data fields that are pertinent to the verification. They can also be used to examine in detail any unusual or exceptional cases, or cases where the matching and merging did not go as expected.

The plots are deliberately kept simple and functional, and while they can be very useful, they are not really intended to be used for producing graphics for presentations, papers, posters or web sites. For such purposes, the user should create her own graphics.

7.5.1 Why PostScript?

Why does MODE (and more generally MET) use its own graphics library rather than some other pre-existing graphics engine, such as MATLAB, IDL, or NCAR Graphics? Mostly to avoid saddling the user with the necessity of having any particular such library installed locally. If MODE used NCAR Graphics, for example, then *all* users of MODE all over the world would have to have NCAR graphics installed on their system in order to use MODE's graphical output. The same holds for systems like MATLAB, with the added burden that the software is not free.

Then again, why use PostScript as an output graphics format, rather than some other format, such as PDF, or a raster format like PNG? PostScript is a very sophisticated vector graphics language (though it can handle raster graphics as well), and so is well suited to producing high-quality plots. In addition, PostScript viewers are freely available and, on Linux and other Unix-like systems, are already installed. Finally, many if not most printers can interpret PostScript directly, something which is not true of raster formats, or even other vector formats such as PDF or SVG.

For those who prefer the PDF document format, a conversion utility is available. The `ps2pdf` utility, a shell wrapper that uses `ghostscript` to do the conversion, is freely available. The command

```
ps2pdf -dPDFSETTINGS=/prepress in.ps [out.pdf]
```

will produce a high-quality conversion of a PostScript file to PDF. If the "`-dPDFSETTINGS=/prepress`" option is left out, the conversion will be of low quality.

7.6 Tips for Making Your Own Plots

In this section we discuss a few tips for making maps (similar to parts (c) and (d) of Figure 3, page 16) that show objects and some of their associated information. We'll also mention a few things to be careful about when drawing your own maps. MODE users are encouraged to make their own illustrations for papers, posters, *etc.*, rather than doing copy-and-paste from the MODE PostScript file.

When making plots of simple or cluster objects, you can use the object number as an index into a color table. For cluster objects, this will correctly indicate which clusters in the forecast and observed fields have been matched. If you want to show the object number, place the number over the object's centroid.

Care needs to be used when drawing convex hulls. If the map projection used in your plot doesn't match the projection used for the data grid, you may get an incorrect picture of the convex hull. Often, people will take the array of points that define the convex hull and unthinkingly connect the points with straight line segments on their map. The problem with this approach is illustrated in Figure 30. Here we see two maps, the one on the left using a Stereographic projection, and the one on the right using a Mercator projection. Each map shows a boundary outline (such as a convex hull) connecting the same three points using straight line segments on each map. The boundary is shown in blue. In terms of true area on the Earth, the boundary on the right encloses nearly half again as much area as that on the left. Many locations, such as Chicago's O'Hare International Airport (marked in red), are inside the boundary on one map but outside it on the other. There are other discrepancies, too. For example, one map indicates that most of Nebraska is inside the boundary, while the other says that *none* of it is.

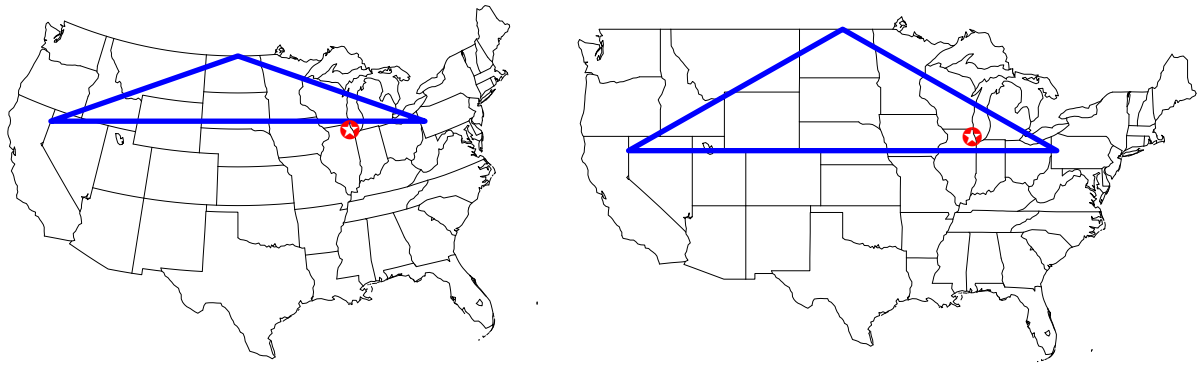


Figure 30 How not to draw a boundary on a map

Because of this, a simpleminded approach to drawing boundaries such as the convex hull will not work. If your map uses a different projection from that used by the raw data, then you will need to transform points on the boundary (*not* just the vertices, but the edges too) from raw grid coordinates to coordinates in your plot's map projection.

Drawing an axis line for an object is something that comes up often in making plots. Two pieces of information are needed: a point on the line, and a slope. Both of these pieces of information can be obtained from the `ascii` attributes file. The axis line for an object always passes through the centroid of the object,

so that gives us our point. The slope is $\tan\theta$, where θ is the axis angle.

Be aware that the object axis angle is measured relative to the local horizontal (*i.e.*, x-coordinate) grid line. Translating this into an angle with respect to the local cardinal directions is a nontrivial computation. Because of this, it's probably best to make sure that the map projection used in your plot exactly matches the one used by the grid.

8

Future Extensions

8.1 Overview

In this section we examine a few ways in which the general MODE methodology could be adapted or extended to other forecast verification situations.

8.2 Three Spatial Dimensions

One obvious extension of the MODE methodology is the use of three spatial dimensions. MODE was originally developed using fields like precipitation and cloud cover as prototypes. Such fields are inherently two dimensional. However, other fields such as relative humidity are also of interest for verification purposes and are naturally three dimensional.

The move to three spatial dimensions is straighforward. The convolution/threshold process for object resolution generalizes easily. Similarly for object attributes, although some, such as convex hull, become much more difficult to calculate in higher dimensions. Finally, once the attributes have been calculated and the interest maps and weights have been defined, the fuzzy logic engine doesn't know or care that the inputs to the engine are from objects that are not two dimensional.

The only real limiting factor in this approach is that while three dimensional forecast fields are quite common, corresponding 3D observation fields are less so. However, it seems likely that in the future this will change, and so 3D Spatial MODE may someday become a reality.

8.3 Vector Fields

One forecast verification situation that arises from time to time is verifying vector fields, for example a 2-dimensional wind field (see Figure 31). MODE's object-based techniques give a new way to approach the problem of verifying wind fields.

While there are a few techniques for verifying gridded vector fields, the state of the art is much less developed than that for scalar fields such as precipitation. Most such techniques involve comparing the forecast and observed vectors at each grid point, calculating some scalar measure of the extent to which they

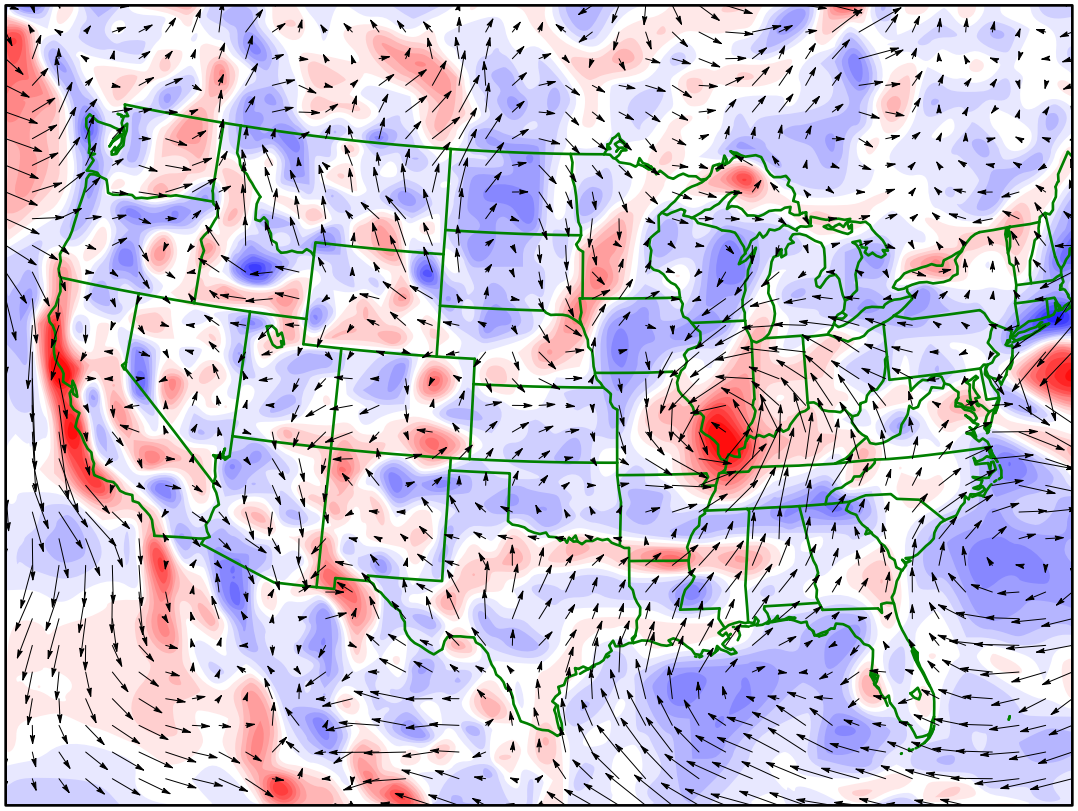


Figure 31 Wind vectors and vorticity

disagree, and giving summary statistics of that measure over the whole grid. Unfortunately, these summary measures suffer from the same disadvantages as other kinds of summary statistics like POD and FAR (see the discussion in the Introduction). They convey no *specific* information on the relative strengths or weaknesses of the forecast.

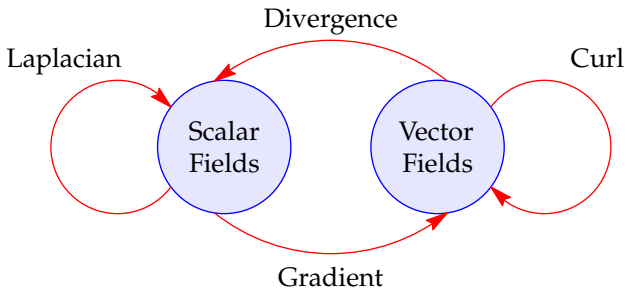


Figure 32 Divergence and Curl

How do we use MODE with vector fields? MODE requires, as input, gridded scalar fields. Given gridded vector fields then, we must convert them into scalar fields. There are several ways to do this, for example replacing the vector at each grid point by its length. In the case of wind vectors, this means calculating the wind speed at each point. While this is possible, and has been done in the past, to do object-based verification we'd prefer to have some method of creating a scalar field which preserves at least some of the spatial structure of the original vector field. To do this, we make use of two differential operators from vector analysis: *divergence* and *curl*. As shown in Figure 32, there are several operations that map back and forth between vector fields and scalar fields. Divergence maps a vector field directly into a scalar field. Curl, in general, maps a vector field to another vector field, but since we're dealing with 2-dimensional vectors, the curl has only a vertical component, so we'll regard it as a scalar. (Note: in the case of wind, meteorologists call this the *vorticity* of the wind field.)

If the wind vector is $\mathbf{V}(x, y) = [u(x, y), v(x, y)]$ where x and y are grid coordinates, then for our purposes we can define the divergence and curl of $\mathbf{V}$ as

$$\operatorname{div} \mathbf{V} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \qquad \text{and} \qquad \operatorname{curl} \mathbf{V} = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{8.1}$$

(Note: these expressions change slightly in the case of a non-conformal grid, but we'll ignore that additional complexity here.) Figure 31 shows a contour plot of the curl of a wind field. Positive values are red and negative values are blue. The curl field gives information on the vortexes in the wind field. Divergence gives information on the presence of sources and sinks. In each case, the spatial features of the calculated divergence or curl field correspond exactly to spatial features of the wind field.

Recall the Helmholtz Theorem from vector analysis, which says that (under some fairly mild hypotheses, almost always satisfied in practice) the original vector field can be recovered from its divergence and curl. Thus no information is lost when replacing a wind field with its divergence and curl.

8.4 Signed Objects

Unlike a precipitation field, for which negative values have no meaning, both divergence and curl can take on any real value. Thus we are confronted with the idea of *signed objects*, *i.e.*, objects that can be either positive or negative.

In this case, resolving the field into objects proceeds as before, with the exception that thresholds become more complicated—a single threshold may not be enough. Recall how the mask field M is obtained from the convolved field C (Eq. 2.8, p. 25, reproduced here):

$$M(x, y) = \begin{cases} 1 & \text{if } C(x, y) \geq T \\ 0 & \text{otherwise.} \end{cases} \tag{8.2}$$

If $T > 0$, then any negative portions of the convolved field C will be erased with this scheme.

Matching and merging become more difficult as well. To see this, suppose there are two objects, one a forecast object and the other an observed object, that are similar enough in their attributes that MODE

would ordinarily match them, except that one is positive and the other is negative. Should they still be matched? Should it instead be considered a miss or a false alarm (perhaps depending on which one is positive and which one negative)? Is it *both* a miss and a false alarm at the same time?

Many issues need to be resolved before signed objects can be used in MODE. Situations can arise (like the one just described) where it's just not clear how to proceed.

8.5 Global MODE

Global models are becoming more widely used, and soon they may be the rule, while regional models become the exception. Currently, however, MODE will not work with a global model. It will not understand, for example, that an object which straddles the international date line is really just one object, not two.

There are other difficulties as well. For example, the convolution radius is expressed in grid units. This is translatable into a distance only if all grid points are at the same distance from their several nearest neighbors. In a regional grid this is often approximately true, but for a global grid this assumption fails utterly. All data points on the top row of a global Plate Carée grid will correspond to the North Pole, and thus the true distance between such grid points will be zero! The rough correspondence usually assumed between grid distance and Earth distance collapses. No fixed value for the convolution radius will do. If the convolution radius could be given in kilometers that would help help somewhat, but even then a circular region on the Earth with a fixed radius will enclose a variable number of grid points depending on where the center of the region is located.

The distance problem due to unequal spacing of grid points becomes even worse for areas. In MODE, areas are expressed in grid units—basically a count of the number of grid squares an object covers. For global grids, you could have two object with greatly different areas in terms of grid squares, but roughly the same true area on the Earth.

Also there would be trouble with axis angles. This angle is currently referred to the grid coordinate lines, but for, *e.g.*, a Plate Carée grid, the grid y-coordinate lines converge to the poles, so this would fail for an object near either of the poles. Referring axis angles to the cardinal directions (north, east, *etc.*) won't help, because the cardinal directions are badly behaved near the poles too. And what if the object actually straddled one of the poles? What should the reference direction for axis angles be in that case?

These are just a few of the issues that would arise if a global version of MODE should ever be built. None of these difficulties is insurmountable, in fact most of them are quite straightforward problems in the analytic geometry of the sphere, but they would require a foundational shift in many of the MODE algorithms, essentially amounting to a complete rewrite of those parts of the code that resolve objects and calculate their attributes.

8.6 Climate Applications

MODE has recently seen increasing use in climate investigations. Drought areas, atmospheric rivers, and other climate phenomena lend themselves quite readily to being interpreted as objects. In addition, available historical datasets can be used to examine these objects over years or even decades, producing climatologies of object attributes such as size, intensity, location, orientation, and so on. Trends in these attributes can be isolated, and used to compare object evolution over one time period with evolution over another period.

8.7 Summary

Many extensions of MODE are possible — this section has examined only a few. Each has possible advantages for particular types of verification. Should future funding allow, some of these extensions may be implemented. In particular, the initial beta release of MODE Time Domain was completed in MET version 5.2. For more information, please refer to the MET User's Guide.

9

Conclusion

9.1 MET

MODE was originally not intended to be a general-release software package. At first, the original development group intended simply to publish the basic ideas (convolution-thresholding object resolution, geometrical and intensity-based object attributes, fuzzy-logic matching and merging) and let interested users implement their own versions. (That's one of the reasons why convolution filtering is used in MODE rather than some more sophisticated method—convolution is fairly easy to code up, even for non-professional programmers.) However, we found that while the convolution-thresholding method for object resolution was easy to implement, the fuzzy-logic matching and merging algorithms were really too complex to expect the typical user to implement them in their full generality. Other parts of MODE, such as the parsing of configuration files, turned out to be non-trivial as well.

Furthermore, at about that time, it was becoming clear that the modelling community needed some standard software, implementing some well-founded and generally agreed-upon methods for forecast verification. Until that time, everybody had their own favorite methods for performing model verification, using their own independently written code (in any one of a dozen or so programming languages), making comparisons of verification statistics produced at different facilities difficult. Some degree of standardization was clearly needed.

Thus the Model Evaluation Tools (MET) project was begun. The original vision for MET was to provide high quality, freely available code for performing a variety of model verification tasks. It was also intended that new verification methods should, after they've proved their worth, be incorporated into MET, thus making the new methods available to the larger verification community. An implementation of MODE was included in the original release of MET and has seen wide and increasing use since then.

Some other spatial verification methods have been incorporated into MET (*e.g.*, the `wavelet_stat` tool), and others are slated for possible future implementation, but so far MODE is the only object-based verification method in MET.

9.2 Growth of MODE

While many approaches to object-based verification were being developed and researched in the early days of MODE, few of them have survived the test of time. Spatial verification methods in general are still being widely researched, and new techniques are continually being explored, but MODE remains one of the dominant techniques in the field of object-based verification.

MODE continues to grow and adapt to new forecast verification situations. MODE's general object-based approach seems to be broad enough to be adapted to many other situations, such as vector fields, time variability and object tracking. Furthermore, MODE has been used on other fields besides precipitation, such as cloud cover, vorticity, relative humidity, and even temperature.

List of Symbols

$\in$, $\notin$ Set membership. If A is a set, we write $x \in A$ to indicate that x is an *element* (or a *member*) of A. We also write $x \notin A$ to indicate that x is not an element of A. For example, if $A = \{1, 3, 5\}$ then $1 \in A$, and $6 \notin A$.

$\subset$ Set containment. If A and B are sets, and if every element of A is also an element of B, then we say that A is a *subset* of B, and we write $A \subset B$. We could express the same thing by saying that B is a *superset* of A, or that B *contains* A, and writing $B \supset A$.

$*$ Convolution.

$\varnothing$ The empty set.

$\wedge$ Wedge product. Used in the exterior calculus.

$\times$ Cartesian product of two sets. $A \times B = \{\, (a, b) \,|\, a \in A,\, b \in B \,\}$

$[a, b]$ The closed interval with endpoints a and b. In other words, the set of real numbers x satisfying $a \leq x \leq b$.

α_i The i^{th} attribute of an object or pair of objects..

$\mathbb{C}$ The set of complex numbers.

C_i Confidence map associated to attribute α_i.

G Grid. In MODE, and in MET generally, grids are represented internally as subsets of $\mathbb{Z} \times \mathbb{Z}$. More specifically, a grid is the set $\{(x, y) \mid x, y \in \mathbb{Z},\, 0 \leq x < N_x,\, 0 \leq y < N_y\}$, where N_x and N_y are the number of grid points in the x and y directions, respectively.

I_i Interest map associated to attribute α_i.

$S_x, S_y \ldots$ Moments.

$\mathbb{N}$ The set of positive integers, sometimes called the "natural numbers": $\mathbb{N} = \{1, 2, 3, \ldots\}$

ϕ Convolution filter function.

$\mathbb{R}$ The set of real numbers.

w_i Weight associated to attribute α_i.

$\mathbb{Z}$ The set of integers: $\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$

$\mathbb{Z}/n\mathbb{Z}$ The set of integers modulo n: $\mathbb{Z}/n\mathbb{Z} = \{0, 1, 2, 3, \ldots, n-1\}$. This is the set of all possible remainders of integers after division by n. Addition and multiplication are defined in the usual way, with the proviso that multiples of n are thrown away. For example, in $\mathbb{Z}/5\mathbb{Z}$ we have $2 + 4 = 1$ and $2 \times 4 = 3$.

Glossary

Attribute Any feature or characteristic of an object that is of interest. Attributes can be purely geometric, like area, centroid and axis angle. Other attributes are intensity-based, such as 90^{th} percentile intensity values inside an object.

Axis A line associated to an object giving information on the spatial orientation of the object. MODE outputs the axis angle, which gives the inclination of the axis line to the x grid direction. Since the axis line always passes through the centroid of the given object, this angle (along with the location of the centroid) suffices to determine the axis line.

Centroid The centroid of an object is the geometric center of the object. In most cases, this point can be thought of as the "midpoint" of the object, although for objects with complex shapes or having holes, the centroid may actually lie outside the object.

Cluster Object A collection of objects in one field that have been merged together. Often consists of more than one simple object, but it's possible for a cluster to contain a single simple object.

Confidence Map A function associated to an attribute that tells MODE how much to trust the calculated or measured value of that attribute. The degree of confidence in the value of one attribute may well depend on the values of other attributes.

Configuration File This is an `ascii` file that MODE reads in at the beginning of each run to set values for all of the user-tunable parameters.

Convex A region in the xy-plane is convex if whenever two points are chosen in the region, the straight line segment joining the points lies entirely inside the region.

Convex Hull The convex hull of an object is the smallest convex set containing the given object.

Convolution A two-dimensional smoothing operation used on input forecast and observation fields to reduce the effect of noise on subsequent steps of object identification. "Smoother" is a generic name for any of a variety of techniques that can be used to either remove undesired high-spatial-frequency data points (*e.g.*, locations of rapid variation, such as noise contamination), or to resample dependent variable values to other independent variable locations using the values of the data at nearby points.

Convolved Field The field obtained by applying a convolution filter to the raw field.

Curvature The curvature of a curve at a point measures the extent to which the curve deviates from a straight line near the point. This can be obtained by determining the circle that best fits the curve near the point (called the *osculating circle*). The curvature is defined to be the reciprocal of the radius of the fitted circle.

Field A scalar dataset on a grid. Examples are temperature, precipitation, humidity.

Fuzzy Engine The part of MODE that uses fuzzy logic to perform matching and merging.

Fuzzy Logic Algorithm Fuzzy Logic is system of logic in which a statement can be true (1), false (0), or any of the values in between. A fuzzy logic *algorithm* uses a set of weighted functions to produce a value between 0 and 1. In MODE, the functions and weights are chosen by the user. The "interest" value produced by the algorithm typically compares how well two items correspond to each other in terms of the attributes used in the functions or interest maps. For example, two objects may correspond very well (*i.e.*, have a total interest value close to 1) if their boundary distance is 0.

Grid In MODE (and more generally in MET) atmospheric grids are represented internally as the composition of a basic map projection and an affine transformation of the xy plane. Often it is convenient to pretend that grids have real-valued x and y coordinates, so that we can use x and y as a coordinate system on that region of the Earth covered by the grid. However, since we only have data values at integer values of x and y, for many purposes we regard grids merely as subsets of $\mathbb{Z} \times \mathbb{Z}$.

iff Abbreviation for the phrase "if and only if." Widely used in mathematics.

Interest Map A function, associated with an attribute, that tells MODE which values of the attribute are interesting (or not) for purposes of matching and merging.

Mask Field The field obtained by thresholding the convolved field. Takes the value 1 where the convolved field meets or exceeds the threshold, and 0 elsewhere.

Matching Associating objects in the forecast field with objects in the observed field.

Merging The process of associating objects in a single field into clusters.

MET Model Evaluation Tools, a free verification software package that includes MODE and many other tools.

Moment The sum of some scalar quantity over an object. For example, the first x moment S_x is the sum of values of the x coordinate over the object. Sometimes the origin of the coordinate system is translated to the centroid of the object before the moment is calculated, in which case the moment is called a *central* moment.

Object Generally, any spatial feature in the forecast or observed fields that is of interest.

Object Field The field obtained by restoring the raw data to the grid points where the mask field takes the value 1.

Pair Attribute An attribute defined on a pair of objects, one object being from the forecast field and the other from the observed field. Examples are intersection area, distance between centroids, and the angle between axis lines.

Raw Field The original data field on a grid.

Simple Object An object that has not yet participated in the matching and merging process. Alternatively, one of the connected pieces of the collection of grid points where the mask field takes the value 1.

Single Attribute An attribute of a simple or cluster object in one field. Examples are centroid, area and axis angle. MODE writes out attributes for both simple and cluster objects in its `ascii` output file, but only attributes for simple objects play a role in the matching and merging process.

Spatial Verification Umbrella term for any forecast verification technique that extracts spatial information from forecast and observed fields and uses that information to make statements about forecast quality.

Splitting The process by which the connectedness of each (grid)point to other points is determined. Points that are connected form simple objects. Points that are not connected are contained within different simple objects. See Figure 10 on page 27 for an illustration. Essentially, splitting is the process of distinguishing objects from one another and from the surrounding empty grid space.

Total Interest A weighted average of interest and confidence maps for a given pair of objects. This value is thresholded to determine the matches and merges.

Weight A scalar value associated to a certain attribute that tells MODE how to gauge the relative importance of that attribute (as compared to others) in the matching and merging process.

References

Books

Folland, Gerald: *Real Analysis: Modern Techniques and their Application*

Lee, John M.: *Introduction to Smooth Manifolds, Second Edition.* Springer-Verlag, 2013. ISBN 1441999817

O'Neill, Barett: *Elementary Differential Geometry, First Edition.* Academic Press, 1966.

Conference Papers

Fowler, Tressa L., and Randy Bullock, 2010: *Examination of Spatial Wind Features Associated with Ramping Events using MODE* First Conference on Weather, Climate, and the New Energy Economy. American Meteorological Society, Atlanta, January 16–20, 2010.

Further Reading

Brown, B. G., R. Bullock, J. Halley Gotway, D. Ahijevych, C. Davis, E. Gilleland, and L. Holland, 2007: *Application of the MODE object-based verification tool for the evaluation of model precipitation fields.* AMS 22$^{\text{nd}}$ Conference on Weather Analysis and Forecasting and 18$^{\text{th}}$ Conference on Numerical Weather Prediction, 25–29 June, Park City, Utah, American Meteorological Society (Boston). Available at http://ams.confex.com/ams/pdfpapers/124856.pdf.

Bullock, R and T.L. Fowler, 2009: *Verifying vectors.* 19$^{\text{th}}$ Conference on Numerical Weather Prediction, American Meteorological Society (Boston).

Casati, B., G. Ross, and D. B. Stephenson, 2004: *A new intensity-scale approach for the verification of spatial precipitation forecasts.* Meteorolgical Applications, **11**, 141–154.

Clark, A. J., J. S. Kain, T. L. Jensen, R. Bullock, M. Xue and F. Kong, 2012: *Diagnosing sensitivities to microphysics parameterizations in convection-allowing models using object-based time-domain diagnostics.* 26$^{\text{th}}$ Conference on Severe Local Storms. 05–08 November, Nashville TN, American Meteorological Society (Boston).

Clark, A. J., C. Karstens, R. Bullock, and T. L. Jensen, 2015: *Applying MODE time-domain for diagnosis and visualization of simulated supercells.* Fourth Conference on Transition of Research to Operations, 6–10 January, Phoenix AZ, American Meteorological Society (Boston).

Clark, W .L., H. Yuan, T. L. Jensen, G. Wick, E. I. Tollerud, R. G. Bullock and E. Sukovich, 2011: *Evaluation of GFS water vapor forecast errors during the 2009–2010 West Coast cool season using the MET/MODE object analyses package.* 25th Conference on Hydrology. January 22–27, Seattle, WA. American Meteorological Society (Boston).

Clark, W. L., G. A. Wick, E. I. Tollerud, T. L. Jensen, J. H. Gotway, E. Sukovich and H. Yuan, 2012: *Assessing and Comparing Real-Time Northeast Pacific Atmospheric River IWV and IVT Intensities Provided by Model (GFS) and Satellite (SSM/I, SSMIS) Using DTC's MET/MODE Object Attributes.* 21st Conference on Probability and Statistics in the Atmospheric Sciences, 22–27 January, New Orleans LA, American Meteorological Society (Boston).

Davis, C. A., B. G. Brown and R. G. Bullock, 2006a: *Object-based verification of precipitation forecasts, Part I: Methodology and application to mesoscale rain areas.* Monthly Weather Review, **134**, 1772–1784.

Davis, C. A., B. G. Brown, and R. G. Bullock, 2006b: *Object-based verification of precipitation forecasts, Part II: Application to convective rain systems.* Monthly Weather Review, **134**, 1785–1795.

Ebert, E. E., and J. L. McBride, 2000: *Verification of precipitation in weather systems: Determination of systematic errors.* Journal of Hydrology, **239**, 179–202.

Gilleland, E., D. A. Ahijevych, B. G. Brown, and E. E. Ebert, 2010: *Verifying Forecasts Spatially.* Bull. Amer. Meteor. Soc., **91**, 1365–1373. DOI: http://dx.doi.org/10.1175/2010BAMS2819.1

Gilleland, E., D. A. Ahijevych, B. G. Brown, B. Casati and E. E. Ebert, 2009: *Intercomparison of Spatial Forecast Verification Methods.* Weather and Forecasting, **24**.
DOI: http://dx.doi.org/10.1175/2009WAF2222269.1

Harrold, M., T. L. Jensen, B. G. Brown, S. J. Weiss, P. T. Marsh, M. Xue, F. Kong, A. Clark, K. W. Thomas, J. S. Kain, M. C. Coniglio, and R. Schneider, 2010: *Spatial verification of convective systems during the Hazardous Weather Testbed 2010 Spring Experiment.* 25th Conference on Severe Local Storms. 11–14 October, Denver CO. American Meteorological Society (Boston).

Jensen, T. L., T. L. Fowler, I. Jankov, J. H. Gotway, R. Bullock, E. Tollerud, and B. G. Brown, 2015: *Understanding SREF Ensemble Behavior Using the Method for Object-Based Diagnostic Evaluation (MODE).* 22nd Conference on Probability and Statistics in the Atmospheric Sciences, 6–10 January, Phoenix AZ. American Meteorological Society (Boston).

Jensen, T. L., E. I. Tollerud, B. G. Brown, J. H. Gotway, T. L. Fowler, R. Bullock, P. Oldenburg, and I. Jankov, 2012: *Evaluation of Ensemble-Based Precipitation Forecasts from an Object-Oriented*

Perspective. 21st Conference on Probability and Statistics in the Atmospheric Sciences. 22–27 January, New Orleans LA. American Meteorological Society (Boston).

Jensen, T. L., M. Harrold, B. G. Brown, S. J. Weiss, P. T. Marsh, M. Xue, F. Kong, A. J. Clark, K. W. Thomas, J. S. Kain, R. S. Schneider, D. R. Novak, F. E. Barthold, J. J. Levit and M. C. Coniglio, 2010: *An Overview of the Objective Evaluation Performed During the Hazardous Weather Testbed (HWT) 2010 Spring Experiment.* 25th Conference on Severe Local Storms, 11–14 October, Denver CO. American Meteorological Society (Boston).

Jensen, T. L., B. Brown, M. Coniglio, J. S. Kain, S. J. Weiss and L. Nance, 2010: *Evaluation of Experimental Forecasts from the 2009 NOAA Hazardous Weather Testbed Spring Experiment Using Both Traditional and Spatial Methods.* 20th Conference on Probability and Statistics in the Atmospheric Sciences. 16–21 January, Atlanta GA. American Meteorological Society (Boston).

Mittermaier, M. and R. Bullock, 2013: *Using MODE to explore the spatial and temporal characteristics of cloud cover forecasts from high-resolution NWP models.* Meteorol. Appl., **20**, 187–196.

Papadimitriou, Christos H., and Steiglitz, Kenneth: *Combinatorial Optimization.* Dover Publications, 1998. ISBN 0-486-40258-4

Roberts, N. M., and H. W. Lean, 2008: *Scale-selective verification of rainfall accumulations from high-resolution forecasts of convective events.* Monthly Weather Review, **136**, 78–97.

Williamson, S. Gill: *Combinatorics for Computer Science.* Dover Publications, 2002. ISBN 0-486-42076-0

Index