Automated Analysis of Spatial Grids

16

Valliappa Lakshmanan

16.1 Introduction

Environmental data are often spatial in nature. In this chapter, we will examine image processing techniques which play a key role in artificial applications operating on spatial data. These AI applications often seek to extract information from the spatial data and use that information to aid decision makers.

Consider for example, land cover data. Since different locations have different types and amounts of forestry, land cover information has to be explicitly tied to geographic location. Such spatial data may be collected either through in-situ (in place) measurements or by remote sensing over large areas. An in-situ measurement of land cover, for example, would involve visiting, observing and cataloging the type of land cover at a particular location. A remotely sensed measurement of land cover might be carried out from a satellite. The remotely-sensed measurement would cover a much larger area, but would be indirect (i.e., the land coverage would have to be inferred from the satellite channels) and would be gridded (i.e., one would get only one land cover value for one *pixel* of the satellite image). Users of land-cover data often wish to use the data to recover higher-level information such as determining what fraction of a particular country is wooded – AI applications help provide such an answer, building on well understood image processing methods.

In this chapter, we will consider spatial data that are on grids, or that can be placed in grids. Spatial grids are digital in nature and arranged in rows and columns of approximately equal resolution in space. Depending on the application, there may be a time sequence of gridded data (as with weather imagery) or the temporal nature may be irrelevant (as is often the case with hazard maps).

In this chapter, we will use the standard matrix notation because it is the one most commonly used in image processing. The first dimension is the row number and the second number is the column number. Thus, (0,0) is the top-left corner and (0,1) is the first row, second column. One potentially confusing effect of the standard matrix notation is that the first dimension increases southwards. If the images are in a cylindrical equidistant projection, then latitude decreases in the first dimension and longitude increases in the second dimension.

16.2 Gridding of Point Observations

In-situ measurements may be placed on spatial grids to enable easier interpretation and analysis by automated applications. Spatial interpolation (see Fig. 16.1) is used to place point observations onto a spatial grid. If the point observations are very close together, so that the average distance between the observations is smaller than, or similar in magnitude to, the resolution of the grid resolution, a technique known as kriging may be used. If, as is more common, the observations are far apart, spatial interpolation relies on balancing the competing concerns of a smooth field and a field that matches the point observations exactly at the locations that the in-situ measurements were carried out. A Cressman analysis favors the creation of smooth fields;

Valliappa Lakshmanan (✉)
University of Oklahoma & National Severe Storms Laboratory,
120 David L. Boren Blvd., Norman, OK 73072-7327, USA
Email: Valliappa.Lakshmanan@noaa.gov

S. E. Haupt et al. (eds.), *Artificial Intelligence Methods in the Environmental Sciences*

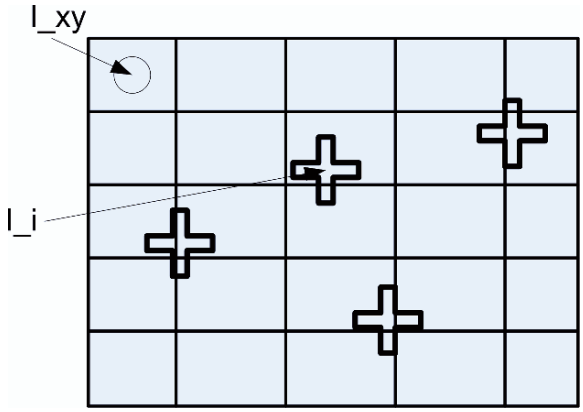

Fig. 16.1 Spatial interpolation is required to take *in situ* observations (shown by the pluses) and place them on to grids

a Barnes analysis is less smooth but attempts to match the point observations better.

Regardless of the method chosen, care should be taken that the pixel resolution is reasonable. If the chosen pixel resolution is too fine, sharp gradients in the underlying data will be smoothed away by the interpolation. If the chosen pixel resolution is too coarse, multiple observations will end up being averaged to obtain a single pixel value, resulting in degraded data quality. As a rule of thumb, it is wise to choose as the pixel resolution a large fraction (typically half) of the mean distance between the original observations.

Cressman (1959) introduced a technique of objective analysis, of interpolating observation data onto spatial grids. Consider Fig. 16.1. In Cressman analysis, the value at a pixel (x,y) is given by:

$$I_{xy} = \frac{\sum_i I_i \frac{R^2-(x-x_i)^2-(y-y_i)^2}{R^2+(x-x_i)^2+(y-y_i)^2}}{\sum_i \frac{R^2-(x-x_i)^2-(y-y_i)^2}{R^2+(x-x_i)^2+(y-y_i)^2}}$$

The impact of an observation at a pixel falls with its distance away from the pixel, so that closer observations have a much impact than observations far away. The parameter, R, determines the scaling or "the radius of influence". The larger the value of R, the more the effect of far-away points is.

Barnes analysis (Koch et al. 1983) improves on the Cressman analysis in two ways. Rather than using a polynomial weighting function, Barnes analysis uses exponential weighting functions. As our discussion on convolution filters later in this chapter will show, Gaussian functions have the nice property of providing the best possible trade-off between noise-reduction and spatial fidelity. In a Barnes scheme, the weights for interpolation are given by: e^{-r^2/σ^2} where r is the distance between the grid point and the observation.

One problem with interpolation techniques is that after gridding, even grid points at the same location as the observations have different values from the observations. This is because of the impact of farther-away points, and is often desirable in case the observation in question is faulty. If a better match to the observation point is desired, Barnes analysis allows for successive corrections. The difference between the observed values and the interpolated values at each of the observation points is interpolated onto the spatial grid and subtracted (with a fractional weight) from the result of the previous iteration. This is carried out until either a maximum number of iterations is reached or until the magnitude of the difference field is small enough. It should be kept in mind that a n-pass Barnes analysis is very sensitive to incorrect data at any of the observation points and can lead to non-smooth grids. However, it can also capture sharp boundaries much better than a 1-pass filter.

Kriging is an interpolation technique that assumes that the co-occurrence of data values can be used to gain a better interpolation. The co-occurrence is estimated by constructing a variogram – a function of correlation between pixel values against the distance between the pairs of pixels. The variogram is then used to make predictions at unobserved locations. See Oliver and Webster 1990 for more details.

16.3 Extraction of Information from Spatial Grids

It is often desired to extract information from spatial grids. For example, it might be desirable to extract from an image of land cover data, all tree-covered areas where drought conditions prevail. Or it might be desired to identify, from a weather satellite's visible channel, where a storm front is. This involves processing the image using automated applications looking for features that make it likely that an area is tree covered or undergoing drought or that the pixels of the image correspond to a storm front.

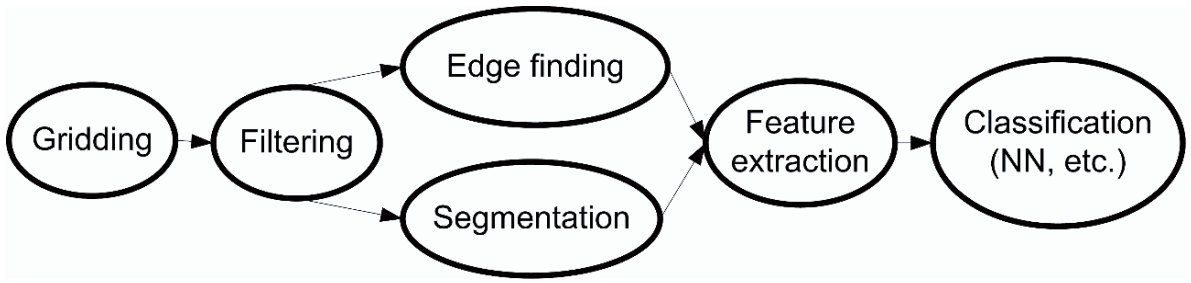

Fig. 16.2 The workflow of a typical AI application operating on spatial grids

There is a considerable body of literature and techniques for such automated analysis of images, finding predetermined objects and patterns and acting on the analysis. Image processing applies the mathematics of signal processing to two dimensions. Thus, the concept of filtering an image, reducing noise, accentuating features to make them easier to find, etc. have been the subject of much research in the electrical engineering and computer science communities. Pattern recognition follows the same approach to images but rather than simply filter images, pattern recognition approaches yield objects as results. Thus, pattern recognition adds to the image processing arsenal tools for segmentation (finding distinct areas in an image) and morphological operations (processing data on shape). Data mining is a larger field of study, of extracting information from all types of data, including data in a relational database. Knowledge discovery is a relatively new sub-discipline that in the context of spatial grids often refers to the extraction of relationships between objects that have been identified in the grids.

Image processing and pattern recognition are specific forms of data mining, concerned with processing and identifying objects in images. Although image processing and pattern recognition have been the subject of decades of research and development (and movies and TV crime shows!), these techniques work only in highly controlled environments. The presence of noise, entities that have a variety of shapes and sizes and incomplete or faulty data, pose significant problems for AI techniques. The most successful implementations of AI techniques are in such environments as factory floors, where unexpected objects are unlikely, all parts are within carefully selected parameters and incomplete data can be engineered away. AI applications in environmental science are some of the most challenging, because in many cases, it the expected size of objects is unknown, and significant artifacts pollute to the imagery presented to these techniques. The rest of this chapter presents the most mature sub-disciplines of these fields and highlights a few applications of these techniques in the automated analysis of real-time weather images.

16.3.1 Work Flow of a Typical AI Application

A typical AI application that operates on spatial imagery to find the presence or absence of some entity within the image is organized as shown in Fig. 16.2.

Preprocessing typically involves taking the spatial data and remapping it so that the grid resolution is locally uniform. If the input data are not in gridded form preprocessing may even involve placing the data in a locally uniform grid.

Uniform spatial grids are filtered to remove artifacts, noise and features that the system designer deems unnecessary or potentially disrupting to the rest of the AI process. Depending on the type of information being mined for, the gridded and filtered data are subject to either edge finding or segmentation. Edge finding finds strong gradients in the spatial data, connects them up and creates cartoons which are then subject to feature extraction and/or pattern recognition. Segmentation finds contiguous data and combines them into objects that are then subject to feature extraction and/or pattern recognition.

Features are properties computed from either the edges or the objects. For example, the eccentricity of the shape and the size and texture of the object are commonly used features. These features are then presented to a classifier such as a neural network, support vector machine or a genetic algorithm. The

output of the classifier typically indicates the presence or absence of some feature that is the goal of this AI application.

16.3.2 Markov Random Process

A basic assumption behind most image processing operations is that the image pixels can have any value, but that the value of a pixel is interwoven with the value of its immediate neighbors. This intuitive idea is formalized by assuming that the pixels of an image are generated through first-order Markov random processes. Two pixels are correlated if, and only if, they are adjacent to each other. For example, consider three adjacent pixels:

A	B	C

The pixels A and B are correlated as are the pixels B and C. The first-order Markov assumption means that the correlation between A and C is captured solely by the correlation between A–B and B–C.

This neighborhood assumption will be seen clearly in the filtering and segmentation operations. These image processing operations do not work well on fields where the pixel values of neighboring pixels are uncorrelated. Visually, such images are highly speckled, like that of a television set with no signal. Most environmental data, however, are relatively smooth spatially. Such data can be processed with image processing operations.

Gridding using spatial interpolation methods such as the Cressman and Barnes operations leads to extremely smooth fields because pixel values are obtained through interpolation of the point observations. Ideally, the grid resolution is chosen to be no more than half the maximum of the distances between every point observation and its nearest neighbor. At higher resolutions, interpolation artifacts can become evident and the neighborhood size for pattern recognition has to be made larger – a first order Markov process won't fit the data anymore. The artifacts and large neighborhood sizes can make pattern recognition harder to perform. It is recommended, therefore, that pattern recognition be performed on data gridded at a reasonable resolution.

When changing the geographical projection of a spatial grid, it is possible that, in the new projection, a couple of pixels may derive their value from the same pixel in the original image. Such repetition of pixels can also lead to spatial artifacts. It is recommended, therefore, that image processing and pattern recognition be performed on data as close as possible to the native format of the data.

Amongst image processing operations, certain operations operate only on the neighborhood of a pixel. Other operations act on the image globally or are greedy – they process as many pixels as do satisfy some predetermined constraints. Global or greedy operations are unsafe on grids where the size of a pixel varies dramatically over the image. Consider, for example, a spatial grid covering the Northern Hemisphere projected in a Mercator projection. The area covered by a pixel at the northern extremity of the image is much smaller than that of a pixel near the equator. This affects the validity of global or greedy operations since the pixels at different areas of the image are quite different. The same problem affects the processing of radar data in their original polar (actually a flattened cone, in 3D) format. Pixels closer to the radar are smaller than pixels farther away. In such situations, it is preferable to process data in a format or projection where the area of a pixel is constant. Examples of such area-preserving projections include the Albers conic and Lambert azimuthal equal-area projections. Naturally, this recommendation is at odds with the previous recommendation of performing image processing in a native format. It might be necessary to perform the operations in both a native format and in an area-preserving projection to discover what works best for a given application. In the case of radar data, the equal-area projection would be to map the radar data onto a Cartesian grid, probably one tangent to the earth's surface at the location of the radar. Repeated pixels then become problematic at long ranges. It may be necessary to try the operation both in the native polar format and in the Cartesian grid projection to discover what works best.

16.4 Convolution Filters

Intuitively, one way to reduce the noise in an image is to replace the value of a pixel with some sort of neighborhood average. Surprisingly, this very simple concept leads to very powerful local operations on images. Instead of simply using neighborhood average, more complex mathematical and statistical operations may

be performed on the set of values in the neighborhood of a pixel. For example, the median of the neighborhood values is one of the best speckle filters available. Sorting the neighborhood values and computing a weighted average of the sorted set is an excellent way to identify edges in an image. Changing the shape of the neighborhood and the weight assigned to neighborhood pixels provides the ability to identify different types of objects in images. This operation, often replacing a pixel with a weighted sum of its neighbors, is termed convolution.

Mathematically, replacing a pixel by a local average can be written as:

$$I_{xy} = \frac{1}{(2k+1)\cdot(2k+1)} \sum_{i=x-k}^{i=x+k} \sum_{j=y-k}^{j=y+k} I_{ij}$$

where I represents the image, x,y represents the pixel at row number x and column number y and k is the half-size of the neighborhood. The above equation means that for every pixel xy, we need to look in a two-dimensional neighborhood, up all the values and divide by the number of pixels. For example, if k were to be 2, we would be computing a 5×5 average around the pixel and dividing by 25. Figure 16.3 shows the effect of such local averaging operation on an infrared satellite image. Note that the jaggedness of the edges has been considerably reduced. Note also that the image is considerably smoother. Such an operation, which reduces high spatial variations in the image, is termed a smoothing operation.

16.4.1 Gaussian Filters

One need not provide the same weight to all the neighbors of a pixel. Often, a higher weight is assigned to pixels that are closer in to the center value. The above equation may be generalized as follows:

$$I_{xy} = \sum_{i=x-k}^{i=x+k} \sum_{j=y-k}^{j=y+k} W_{ij} I_{ij}$$

where the weight W essentially determines the type of operation that is performed. Technically, this operation is cross correlation. However, if one is using symmetric matrices (as we almost inevitably will be), cross correlation and convolution are the same thing. In the case of a 3×3 local average, the weight matrix is given by:

$$W = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The local average weight matrix, often called the boxcar kernel, is a poor choice for smoothing because it can result in large spatial relocations of maxima. A better convolution kernel for smoothing is one where the weight matrix has large values in the center and smaller values around the edges. That way, values far away from the center pixel have a lesser effect on the final value than do values closer in. A Gaussian convolution kernel offers the best trade-off between spatial smoothness and the moderate variability within the image that are characteristic of local maxima. Because the features do not smudge as much, one can smooth a lot more effectively with a Gaussian kernel. In the Gaussian kernel, the values of the weight matrix are computed using this equation:

$$W_{xy} = \frac{1}{\sigma^2} e^{\frac{(x)^2+(y)^2}{-2\sigma x^2}}$$

where x and y range from −k to k. Since the Gaussian has infinite range, one needs to be careful to choose an appropriate value of k – approximately thrice the value of sigma typically works well. Note that the above matrix equation does not add up to one within the neighborhood. Therefore make sure to divide by the total weight. The higher the value of sigma, the more smoothing happens. A comparison of smoothing with the Gaussian kernel and a nearly equivalent boxcar kernel is shown in Fig. 16.4. Note that the Gaussian provides a smoother image with less smudging of maxima (the smudging of maxima is evident in the figure (top) in the greenness of the thin vertical line at the bottom left of image).

16.4.2 Matched Filters

By changing the weights in the convolution kernel, it is possible to extract a wide variety of features. For example, thin vertical lines may be extracted using the weight matrix shown below:

$$W = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

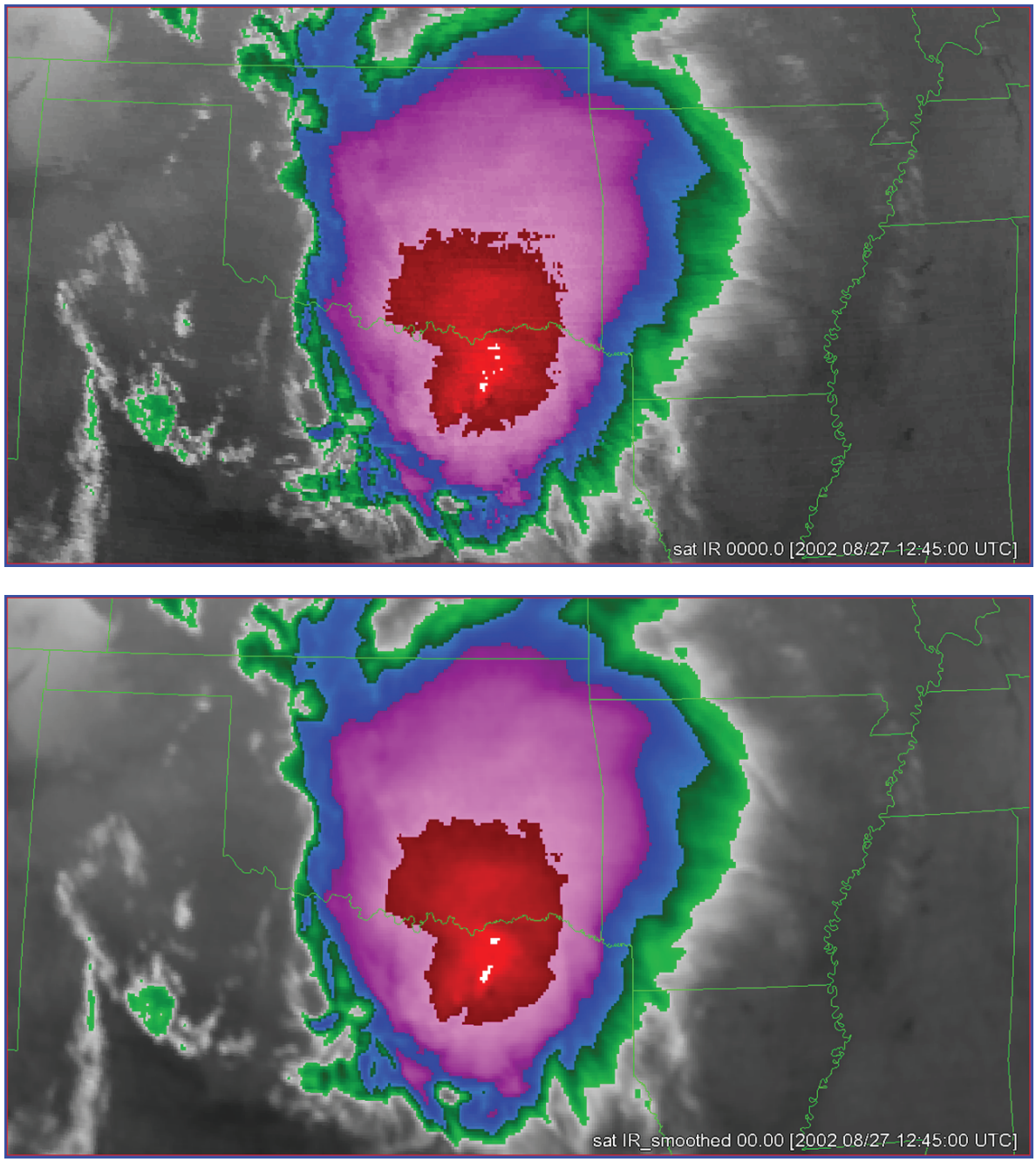

Fig. 16.3 Top: an infrared satellite image. Bottom: the same image with a 5×5 local average applied to it

Note that the above weight matrix when applied to an image results in high values, wherever there are high values to the left of low values. In areas where there is very little change, the positives and negatives canceled each other out, resulting in very small and pixel values in the in the result. The result of this operation when applied to an infrared image is shown in Fig. 16.5.

As a general principle, a convolution kernel may be used to extract features that look like it. In other words, convolution may be thought of as a matched filter operation. To avoid simply getting high values in

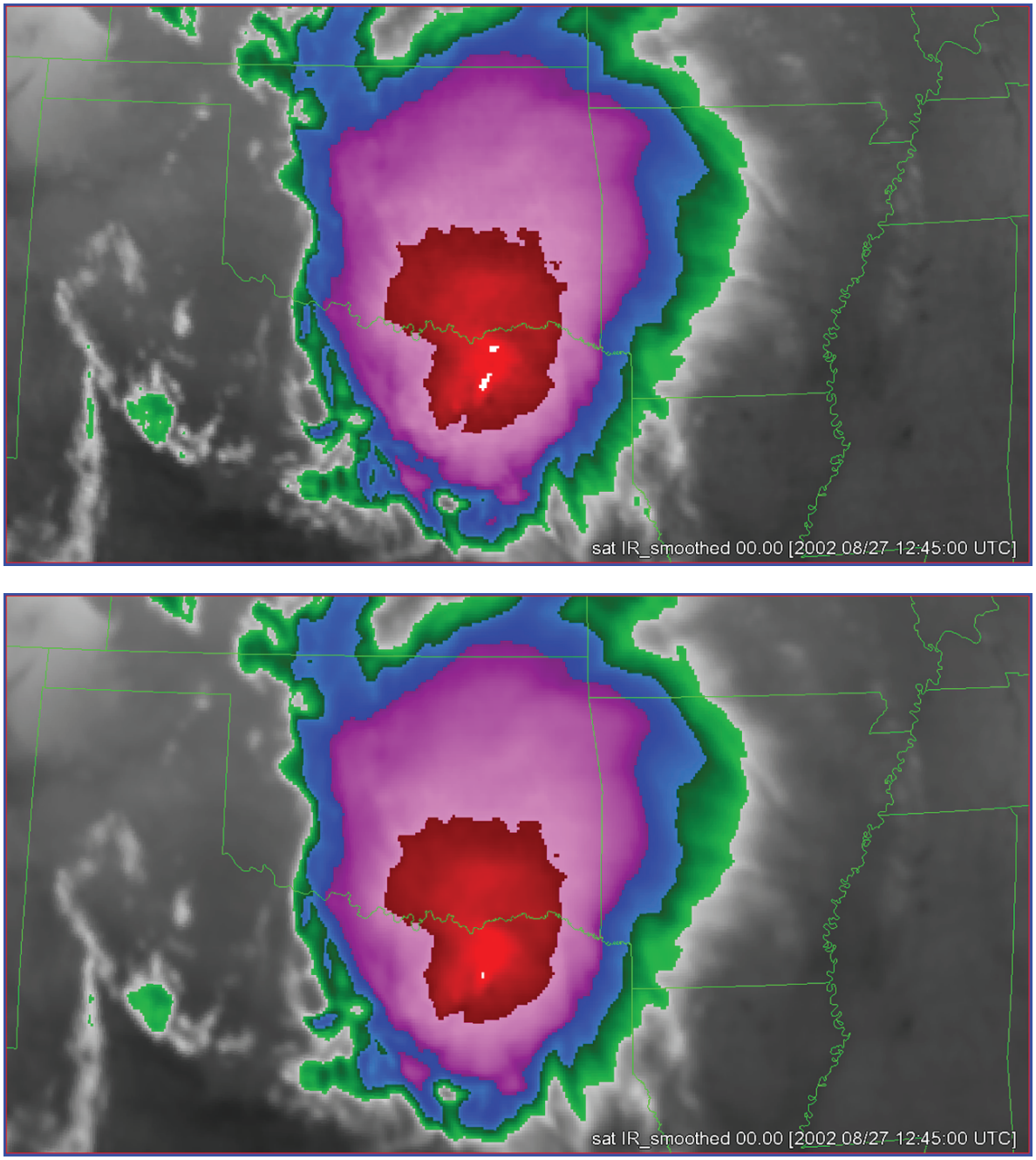

Fig. 16.4 Top: The infrared image of Fig. 16.3a smoothed with a box kernel with a half-size of 3. Bottom: The same infrared image smoothed with a Gaussian kernel with a sigma of 3 (and half-size of 9)

the result wherever there are high values in the input image, one needs to normalize the kernel result at a pixel by the smoothed value at that pixel. The kernel used in Fig. 16.5 returns large magnitudes for thin vertical lines, because if one were to think off it as a topographic map, the weight matrix is appears like a ridge. Of course, as Fig. 16.5 shows, the matched filter is not perfect – the operation returns large values for anything where the values of the left are higher than the values on the right, not just thin lines.

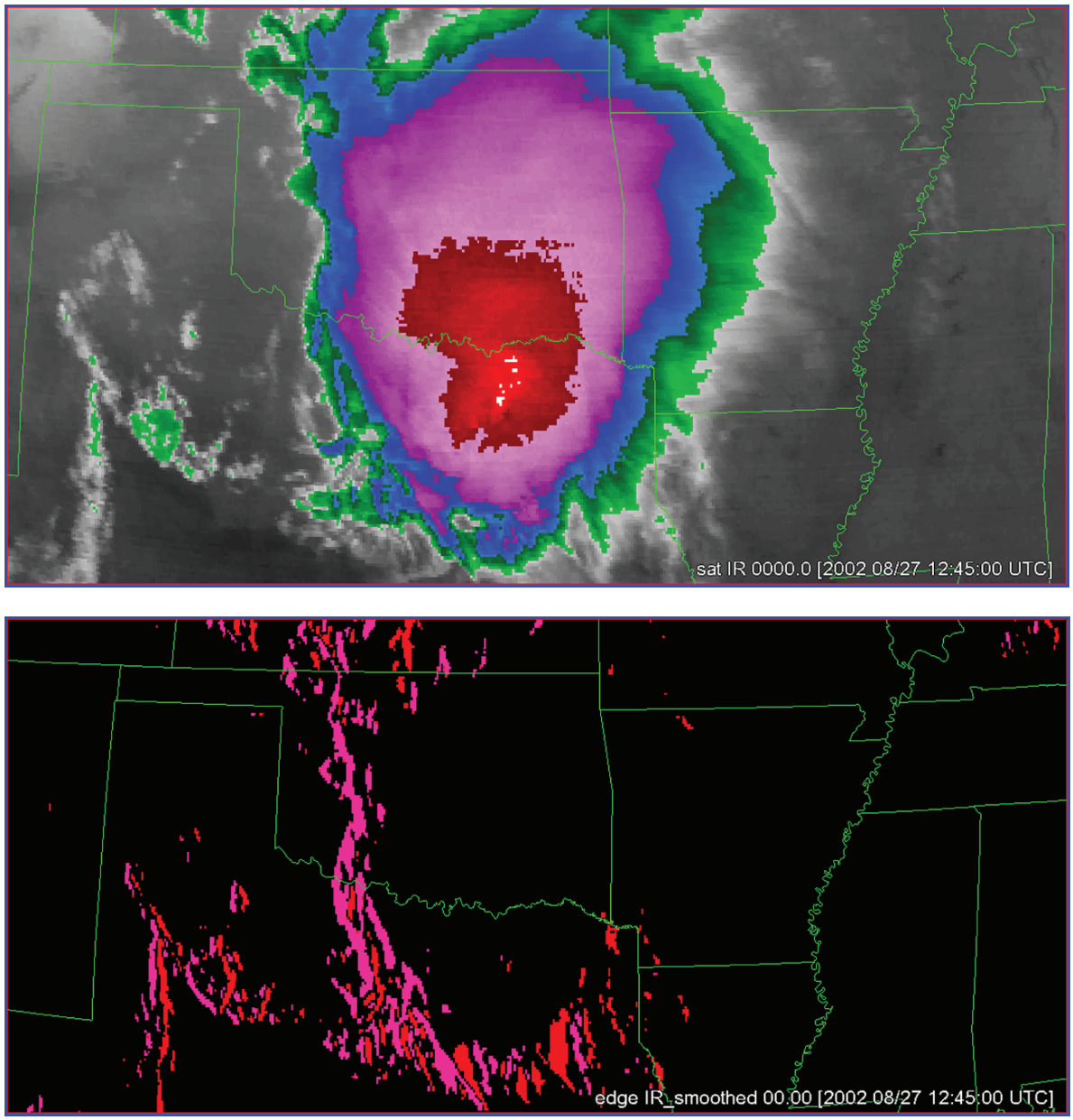

Fig. 16.5 Bottom: the effect of applying a vertical-line detection kernel to the infrared image on the top

16.4.3 Filter Banks

The matched filter idea may be used to find objects in an image. Unfortunately, though, convolution filters are both scale and orientation dependent. Thus, if one needs to find ridges that are five pixels thick, the convolution kernel needs to be five pixels thick. If the boundary of interest is not vertical, but horizontal, the matched filter needs to have its ones and zeros oriented horizontally. This makes matched filters extremely hard to use to find objects whose scale and orientation are not known in advance.

Most commonly, matched filters to find objects are used as part of a filter bank, as shown in Fig. 16.6. Several filters of different sizes and orientations are used to filter an image. The final result at a pixel is determined

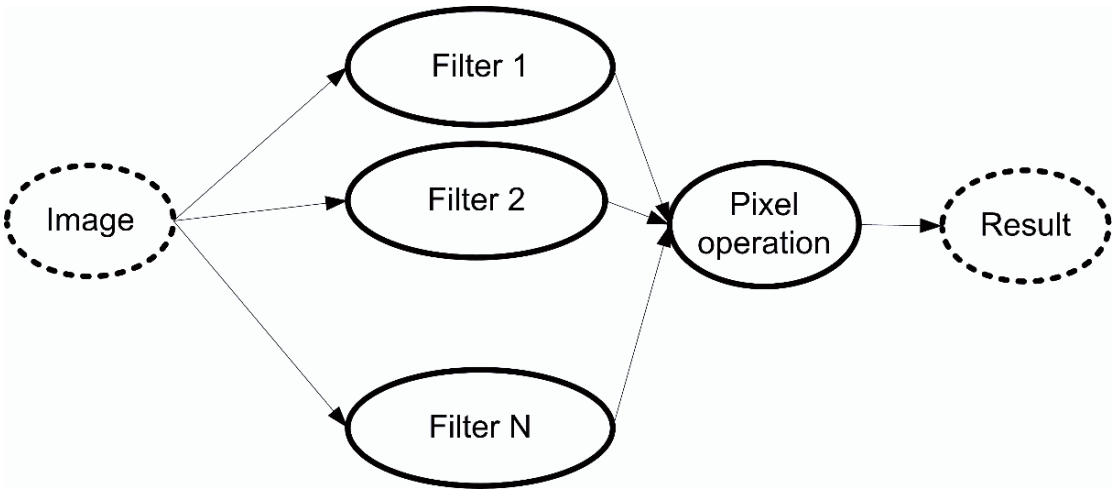

Fig. 16.6 A matched filter is typically used as part of a filter bank, with filters of different scales and operations. The best response from the individual filters is chosen at each pixel to provide the final result

by combining the result of the different convolution filters at that pixel. The combination method may be to take the average of the individual filter responses or to choose the best response, for example the maximum.

A general purpose smoothing convolution filter that can be used to match regions of different sizes and orientations is shown below:

$$W_{xy} = \frac{1}{{\sigma_x}^2 + \sigma_y^2} e^{\frac{(x\cos\theta)^2 + (y\sin\theta)^2}{-({\sigma_x}^2 + \sigma_y^2)}}$$

This is a Gaussian, where the two sigmas determine the vertical and horizontal scales while theta represents the orientation off the object that could be matched.

A number of boxcar convolution filters at different orientations (but not scales) are used in a filter bank to identify storm fronts by Wolfson et al. 1999. They then choose the maximum filter response to be the final output of the smoothing operation. Doing so achieves the nice effect of smoothing along the storm front.

There are several drawbacks to performing multi-scale or orientation analysis using a matched filter bank. Because repeat convolutions have to be performed on the original image, filtering can take a very long time. Also, because the results of the filters are not related, it may not be possible to perform higher level operations based on just filter banks. Several simplifications are possible in order to improve the efficiency and usability of filter banks. Firstly, as Section 16.3.5 illustrates, convolution itself may be sped up to by taking advantage of Fourier transforms and separable filters. If the resulting features are related, it is possible to first identified the objects, and then combine them outside the filter bank. If the resulting images themselves are related, it is possible to use wavelets to perform higher level operations on the related set of images.

16.4.4 Missing Data and Image Boundaries

Often, parts of the domain will not have been measured. The sensor may have had equipment problems. The point of view of the sensor may have been such that some part of the domain is out of range. The beam may have been blocked. Yet, numerical operations like image processing operators can not deal properly with such missing data. If one needs to compute a local average around the center pixel:

5	2	X
4	3	6
4	2	X

and "X" denotes a pixel that was not measured, how can a local average be computed? There are two broad approaches: (a) compute the average only on the non-missing pixels, in which case, the answer would be $26/6 = 4.3$ or (b) assume that a pixel when missing

is a "typical" value, say 5, in which case, the answer would be $36/8 = 4.5$.

Computing the value using only non-missing data usually yields better, more representative results with fewer artifacts. However, the second approach of filling unmeasured pixels with a default value is conducive to several optimizations, in particular of performing operations in a transform domain. The problem, of course, is of correctly choosing the default or "typical" value.

The same problem occurs in a different guise when the operations lead to the edge of the image. The simplest approach is to add an imaginary row or column, assuming that the entire row/column is missing and use one of the above two approaches. The exception is in the case of radar data where the radials "wrap" back around, so that the boundary condition is only when the data go out of radar range.

16.4.5 Speeding Up Convolution Operations

One problem with using Gaussian filters is that the scale of the filters tends to get quite large. This translates directly to filter size. So, the larger the filter, the more time it takes to compute a local average. If Gaussian filters are used in a filter bank, this loss of performance adds up to become a critical bottleneck. There are two ways to speed up a matched filter bank made up of Gaussian filters.

Convolution can be performed in Fourier transform space. The original image and awaiting matrix are both transformed into Fourier space. In Fourier transform space, convolution is merely pixel to pixel multiplication. Thus, convolution even with very large weighting matrixes takes only as long as the time it takes to compute the two Fourier transforms – the larger the kernel the more dramatic the speed up (Lakshmanan 2000). There is one drawback, however, to performing convolution in Fourier transform space. In order to compute a Fourier transform, it is necessary for the entire image to consist of valid values. If certain pixels within the grid went unsensed, Fourier transform methods cannot be used directly. Instead, in the preprocessing step, missing pixels have to be set to some default value. Traditionally, this default value is either zero or the mean of the entire image.

The following steps will perform convolution in Fourier transform space:

1. Pad weighting kernel to nearest power of 2 or other small prime
2. Compute Fourier Transform of weighting kernel
3. Pad image to nearest power of 2 or other small prime
4. Preprocess the image and fill all missing pixels with a default value, say zero
5. Compute Fourier Transform of image
6. Multiply the two transforms pixel-by-pixel
7. Compute the inverse transform of image

The reason for padding the image for computing the Fourier transform is that fast methods (called Fast Fourier Transforms, FFTs) exist to compute Fourier transforms on digital data.

Fourier transform methods cannot be used on images, such as radar data, that commonly have missing values. A second alternative exists for speeding up convolution operations. This is to formulate the weighting matrix as a separable function. For example, the Gaussian kernel without any orientation can be written in separable form as:

$$W_{xy} = \frac{1}{\sigma^2} e^{\frac{(x)^2+(y)^2}{-2\sigma_x^2}} = \frac{1}{\sigma^2} e^{\frac{(x)^2}{-2\sigma_x^2}} e^{\frac{(y)^2}{-2\sigma_x^2}}$$

Separable convolution filters can be implemented in a fast manner by first processing the image row by row, and then processing it column by column. If the weighting kernel were 25×25, convolution using the inseparable form would require 625 operations at every pixel. On the other hand, the separable form can get away with just 50. Therefore, the separable form of convolution can lead to significant speedups, although not as dramatic as the Fourier transform methods. For an approximation to oriented Gaussian filters in a separable form please see Lakshmanan 2004.

16.4.6 Wavelets

Wavelets are a multiresolution technique. They provide a way by which images may be broken up into sub-images such that one of the sub-images is a smaller but faithful representation of the larger image. The smooth images in Fig. 16.7 (S0, S1, S2, etc.) are decomposed to yield three detailed images (D0, D1, D3, etc.). That sub-image can itself be broken up

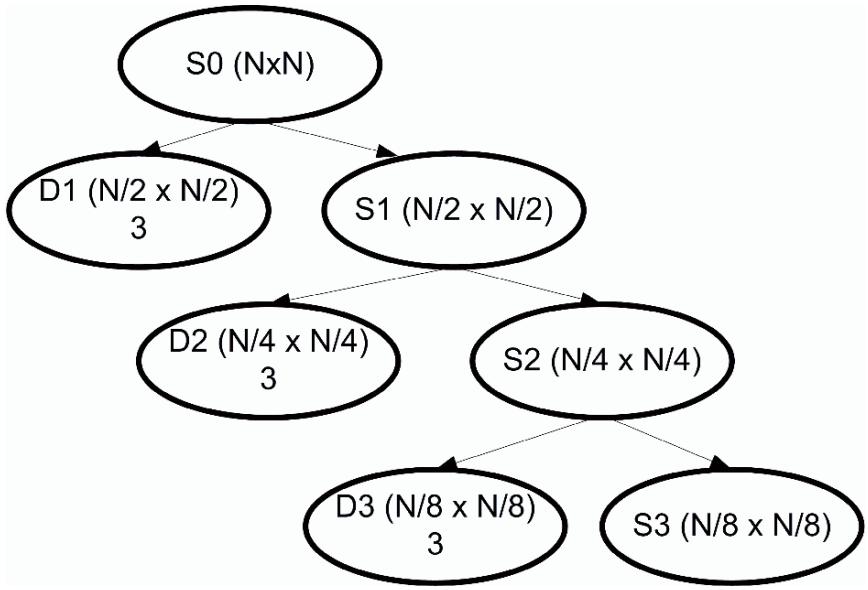

Fig. 16.7 Using wavelet analysis, images are decomposed into detailed images (left) and a smaller smoothed image (right)

into the more sub images. At each stage, the four sub images can be combined to yield the larger image that was decomposed to yield the sub images.

In order to decompose images such that the smoothed images have the above relationship, only specific convolution weight matrices may be used. The most commonly used convolution filters in wavelet analysis are the Haar function and the Daubechies p-functions. If all that is required is to be able to process images at multiple scales, wavelets are overkill. Simply filtering the image using Gaussian filters at different sigmas may suffice.

16.4.7 Edge Finding

The line finding filter described in Section 9.3.2 is not a robust way to find edges. Just as using a boxcar kernel to smooth images leads to abrupt transitions, using a box-like line finding filter results in smudging of the lines that are detected. To identify lines in an image, use the Canny edge-finding filter (Canny 1986). The Canny filter involves using the weighting matrix shown below:

$$W_{xy} = \left(\frac{1}{2} - \frac{x^2 + y^2}{\sigma^2}\right) e^{-\frac{x^2+y^2}{\sigma^2}}$$

After applying the above convolution filter (known as the Laplacian of a Gaussian because it is obtained by differentiating the Gaussian equation twice), look for zero crossings in the convolution result. Connecting up the zero crossings provides the location of the edges in the image.

16.4.8 Texture Operations

Convolution involves computing the weighted average of the pixel in the neighborhood of a central pixel. Other operations, however, can be performed once the values of the pixel's neighbors have been extracted. In fact, the boxcar kernel may be thought off as a statistical operation – the mean – on the neighboring values. Taken together, these statistical operators are called texture operators.

Texture provides a good way to distinguish different objects based on something other than just their data values. Even if two objects have the same mean value, the distribution of values within the objects may be different. Texture operators attempt to capture this difference in the distribution of data values. Commonly used texture operators include those based on second and third order statistics. Variance and standard deviation capture the range of values – high values of these are associated with "noisy" regions. Homogeneity is a third order statistic that captures how smooth the data values are in the neighborhood of the pixel. Another very useful texture operator is the entropy, which is

Shannon's measure of information content. It is computed by taking all the data values in the neighborhood and forming a histogram. If p is the fraction of pixels in each bin of the histogram, the entropy is given by:

$$\sum_i p_i \log(p_i)$$

The entropy is low in regions where all the pixels fall in the same bin (because the logarithm of one is zero and the probability in the all other bins is zero as well). The entropy is highest in regions where there is a large diversity of pixel values. Typically, very high entropies are associated with noise while very low entropies are associated with instrument artifacts. Of course, the actual thresholds have to be found by experiment.

16.4.9 Morphological Operations

In addition to convolution filters and texture operators, the neighborhood values may be sorted and operators

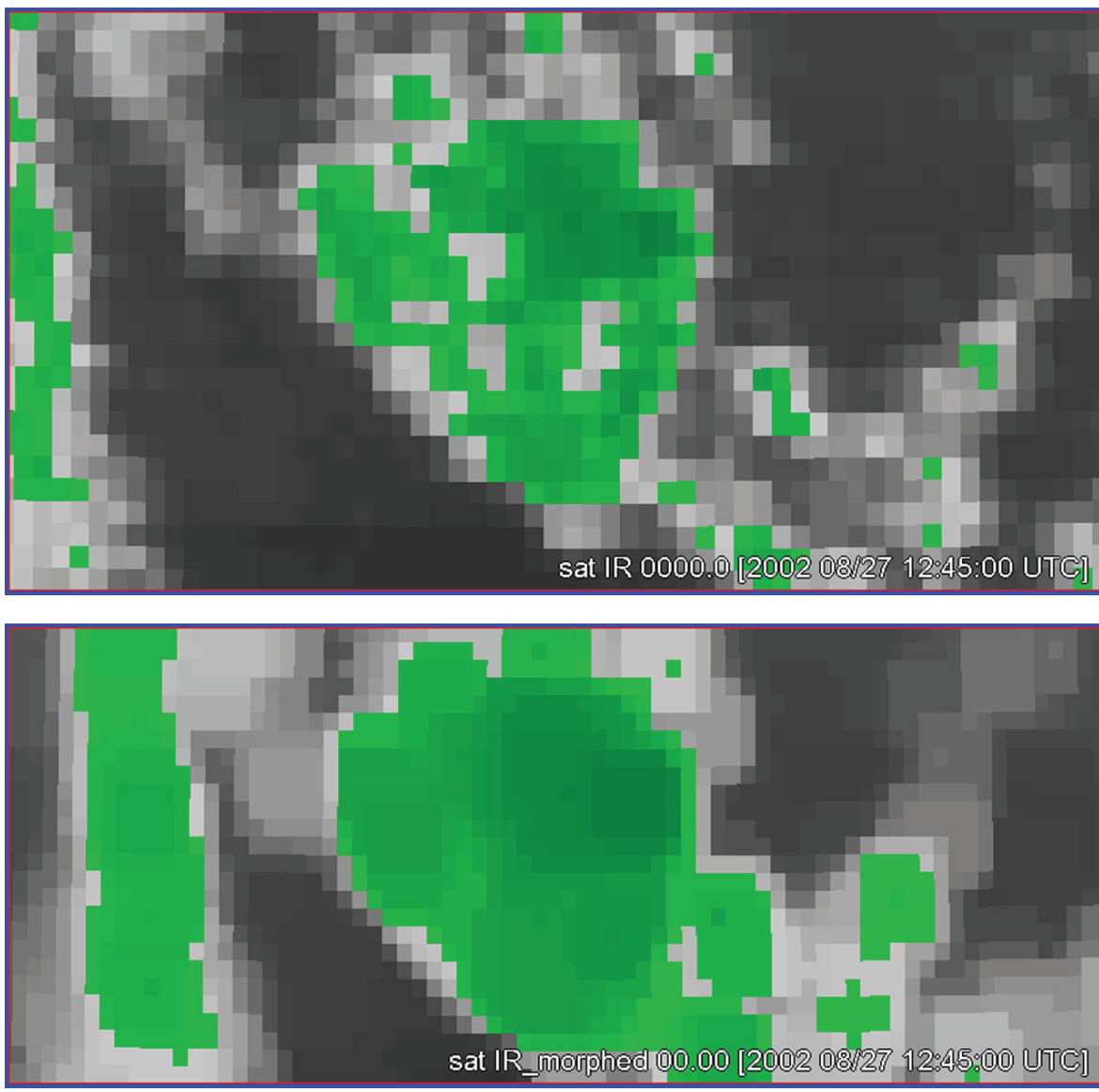

Fig. 16.8 Spatial dilation and erosion operations may be performed by replacing a pixel with the maximum or minimum of the values of its neighbors

Fig. 16.8 (*Continued.*)

based on the sorted values may be applied to the images. The median value off the neighborhood values is an excellent smoothing filter. The median filter works especially well, and should be chosen in preference over Gaussian filters, when the noise has speckle characteristics.

Taking the minimum value of the pixel values in a neighborhood has the effect of eroding the image spatially, so that regions become smaller and small regions get removed. Similarly, taking the maximum has the effect of dilating the image spatially. The minimum and maximum are affected detrimentally, if there is noise present in the images. Taking something like the fifth or 95th percentile may help tradeoff the noise characteristics of the image. Alternately, taking the second-lowest or the second-highest value in the neighborhood also helps to reduce the impact of noise. Spatial erosion and dilation where the second lowest and highest values are chosen is shown in Fig. 16.8.

16.4.10 Filter Banks and Neural Networks

So far, we have looked at the number of operations that can be performed on the neighborhood values of the pixel. These include convolution filters (boxcar, Gaussian, matched filters, orientation detectors), texture operators (variance, homogeneity, entropy) and morphological operators (erosion, dilation). Several of these filters may be applied in to an image, either in parallel or one after the other, to an image as part of a filter bank as shown in Fig. 16.9.

For every pixel in the original image, we will obtain a vector of filter results. A neural network or some other classifier can take all of these input and classifying each vector into two or more categories. In other words, each pixel provides a pattern to the neural network. Since a $1{,}000 \times 1{,}000$ image will provide one million patterns, one may have to be selective about whether all the pixels in an image get presented to the neural network for training and/or classification. Because many of the pixels in an image are highly correlated, one must be careful to not assume that the patterns from a single image are independent training samples to the neural network. A good rule of thumb is to treat all the patterns from a single image as simply one data case when deciding whether one has enough of a training or validation set.

As an example of an AI application that proceeds from images to filters through a neural network towards the classification result, consider the radar reflectivity quantity control system described in (Lakshmanan 2006). In that application, radar reflectivity, velocity and the spectrum width polar data are

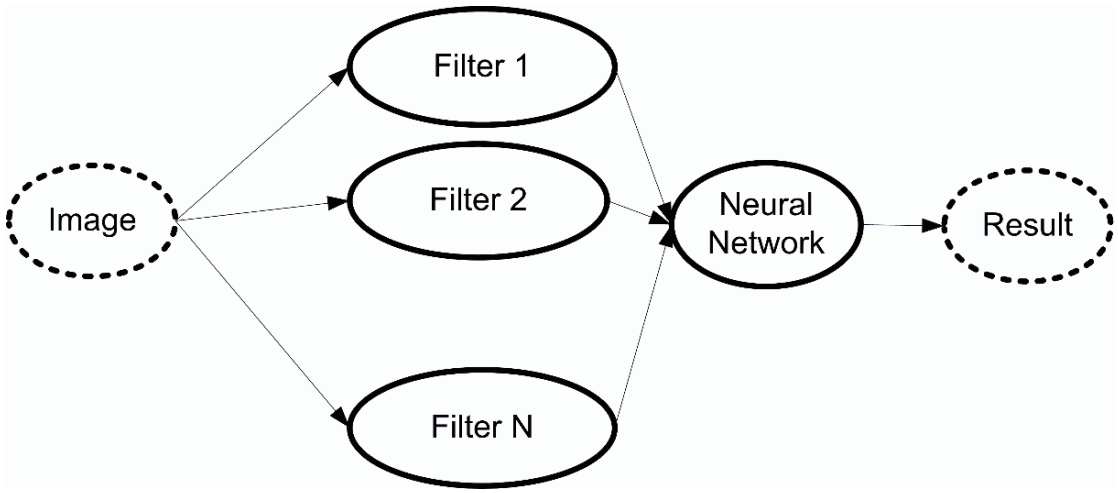

Fig. 16.9 The output of a filter bank may be used to provide patterns to a neural network for classifying an image pixel-by-pixel

converted to a uniform resolution and indexed polar grid. A variety of texture operators are applied to the three polar grids at each elevation. The results of the filters at each pixel form the patterns that are presented to a classification neural network. The output of the neural network is thresholded at 0.5 to determine whether the pixel in question corresponds to good data or to non-meteorological artifacts. Not all the pixels are presented to the neural network. Instead, a preclassification step classifies those pixels that can be done quite easily based on a rule engine. The pixels that are presented to the neural network comprise the hard to classify pixels. If the pixels are classified solely by the neural network and the rules engine, the resulting field will have different classifications even for adjacent pixels that are part of the same storm or artifact. In order that regions off the radar image all get classified the same, it is necessary to average the results of the classification over objects found in the images rather than treat the pixels as being independent. How to do this is the focus of the next section.

16.5 Segmentation: Images to Objects

So far, we have looked at ways of processing images. Convolution filters, texture operators, edge detection and morphological operators all have, as their output, images. In most applications, however, what is desired is to be able to identify objects and to classify or track these objects as entities. Classifying pixels makes no sense, because pixels are ultimately just an artifact of the remote sensing instrument. In this section, we will look at how to identify objects from grids.

16.5.1 Hysterisis, to Convert Digital Data to Binary Images

The process of identifying self-similar groups of pixels and combining them into objects is called segmentation. Segmentation algorithms work only on binary data. Therefore, the pixel values have to be converted to zeros and ones before segmentation can begin. Using just one threshold to convert the data into binary typically results in lots of very tiny objects (if the threshold is too high), a few very large objects (if the threshold is too low) or in objects that have lots of holes (if a moderate threshold is chosen).

To avoid the problems associated using only one threshold, employ hysteresis to convert pixel values into zeros and ones. Hysteresis involves picking two thresholds, say t1 and t2. Pixel values below the first threshold are always set to zero and pixel values above the second threshold are always set to one. Pixels with values between the two thresholds are set to one only if they are contiguous to a pixel that has already been set to one – this condition is easily applied during region growing, discussed in the section. The use of this technique mitigates, but does not completely prevent, the

problems associated to with a single threshold – the choice of the two thresholds t1 and t2 has to be made through experiment.

16.5.2 Region Growing

The segmentation process is done through region growing. In the region growing algorithm, contiguous pixels are assigned the same label such that all the pixels that have the same label together form a region. The steps of a region growing algorithm are listed below:

1. Initialize an image called the label image. All of its pixels are set to zero.
2. Initialize current label to zero.
3. Walk through the image pixel by pixel and check if the pixel value of the image is greater than t2 (see Section 16.4.1 on hysteresis) and if the label at this pixel is zero. If both conditions are true:
   (a) Increment current label.
   (b) Set label at this pixel to be the current label.
   (c) Check all eight neighbors of this pixel. If a neighbor's pixel value is greater than t1, repeat steps b and c at the neighboring pixel.

At the end of the above steps, the label image has a region number assigned to every pixel. Pixels where the label image has a value of zero do not belong to any region. Two pixels with the same label are part of the same region. Therefore, region properties may be computed from the original image's pixel values by maintaining a list of statistics (one for each region), walking through the pixels of the original image and updating the statistic for the corresponding region (read out from the label image) that the pixel belongs to. This way, a vector of properties or statistical features is obtained for each region. These features may be presented to a neural network or other classifier to classify the identified regions into two or more categories.

16.5.3 Vector and Hierarchical Segmentation

Although with hysteresis, one uses two thresholds to reduce the incidence of disconnected regions, the determination is made ultimately on a single pixel value. The incidence of disconnected regions could be reduced even further if the values of neighboring pixels could be considered when creating the regions. In other words, it would be better if a filter bank could be applied to an image, and the results from all the filters in the filter bank could be used to determine which region a pixel belongs to. Such a segmentation technique is called a vector segmentation technique because it operates on a vector of inputs, not just one value at every pixel.

Another drawback of using the hysteresis-based region growing approach is that it is not possible to obtain hierarchical regions with overlapping thresholds. In real-world AI applications, it helps to be able to identify regions and place them into larger regions. Regions, sorted by size, can be used for different tasks and to evaluate different types of constraints. Such a segmentation technique is termed a hierarchical approach.

The classical hierarchical segmentation technique is the watershed algorithm of (Vincent and Soille 1991). The technique consists of first sorting the pixel values within the image in ascending order of magnitude and then slowly raising a "flood" level. Connected regions, identified through region growing, form a hierarchy with the flood level determining how high up in a hierarchy a region exists before it is subsumed into a larger region. The saliency of a region, the maximum "depth" of a region, provides an indication of how important it is. Unfortunately, watershed segmentation works very poorly in images with statistical noise. Watershed segmentation may work quite well on model fields and other smooth datasets, so it is worth trying before attempting more complex techniques. Statistical noise is a given in most remotely sensed images, so a better hierarchical technique, ideally one that works on vector data, is required.

Lakshmanan 2003 describes a technique where an explicit trade off is made between the self-similarity of a region and the idea that a region should be compact and consist of contiguous pixels. A pixel is similar to a region if the Euclidean distance between the result of a filter bank applied to the image at that pixel is close to the mean of the filter bank results of all the pixels that are already part of the image. This is essentially the K-Means clustering approach, of arbitrarily choosing K regions and then updating the regions based on which

pixels get added or removed from them. When the set of pixels in a region does not change, or if the magnitude of those changes is quite small, the clustering is stopped. This vector segmentation approach is made hierarchical by using steadily relaxing the allowed inter-cluster distance before regions are merged.

16.6 Processing Image Sequences

So far, we have considered processing images (or spatial grids) individually. In many applications, the temporal change of spatial data is important. For example, one might wish to study the change in forest cover over a decade or the movement of a hurricane over ocean. In motion picture processing, individual images are termed "frames" and a set of frames arranged in temporal order forms a "sequence". There has been quite bit of research into techniques for processing image sequences, mainly for applications such as the compression of video and for automated security monitoring using video cameras.

16.6.1 Detecting Change

The most common requirement in processing image sequences is to simply determine whether a change has occurred. The simplest way of identifying changes is to compute a pixel-by-pixel difference between selected frames of a sequence. Pixels where the absolute maximum of the change is high indicate locations where a change has occurred.

The differencing technique works well for sequences where objects suddenly appear or disappear. For example, in a forestry application, it is possible to use differencing to monitor whether areas of a forest have been cleared.

The differencing technique works poorly in applications where the objects are moving from one location to another. This is because the magnitude of the difference field will be high only in areas where the object has completely moved to or away. Where there is an overlap, the magnitude of the difference will be small. Thus, a single movement might appear to be two separate areas of change. This limits the utility of differencing techniques in applications such as storm tracking.

16.6.2 Tracking by Object Association

One way to mitigate the problems associated with pixel-by-pixel differencing is to compute differences on a region-by-region basis. In other words, segment the frames of a sequence, associate objects between frames and then tabulate changes in aggregate statistics (such as size, mean value, etc.) of the regions.

This is easier said than done. Several hard problems arise in the technique outlined above. Segmentation is a notoriously noisy operation – the change of magnitude in a few pixels can drastically change the objects that are identified. Since successive frames in a sequence are slightly different, the results of segmentation on these frames may result in dramatically different region identification. This makes associating regions problematic. Even if the segmentation problems are resolved satisfactorily so that slightly different images yield only slightly different regions, the region association problem is not insignificant. In storm tracking, for example, individual storm cells may split or merge – this needs to be accounted for in the object association.

If the objects are large enough and move slow enough for them to overlap significantly, the association problem is not difficult. A minimum overlap in terms of spatial correlation may suffice for associating objects. This is the strategy commonly employed in tracking mesoscale convective systems. See for example Carvalho and Jones 2001.

When there are numerous small features identified, it can be unclear as to what the optimal assignment ought to be. A principled way of associating objects would be to minimize a global measure of fitness, such as the total Euclidean distance between the vectors of properties between associated objects. This is the approach followed by Dixon 1993, where a linear programming approach is employed to minimize a least squares metric. A less principled approach, but one that works quite well, is to extrapolate the movement of regions and then simply choose the region in the next frame closest to the anticipated location. This is the method employed by Johnson et al. 1998.

16.6.3 Estimating Movement Using Optical Flow

Object association techniques tend to identify movement only at the scale of the objects. If the objects are ephemeral or difficult to identify, the quality of the movement estimated from object tracking methods suffers.

A better way of estimating movement may be to ignore object identification altogether. Instead, the image is considered a fluid field and the movement at a pixel is assumed to be that movement which when removed from the current field would result in the smallest magnitude difference field to the previous frame. To compute this, a neighborhood of pixels around a central pixel is moved around in the previous frame until the point at which the difference is minimal is identified. This movement is the movement at that pixel. This process is repeated for each of the pixels in the current frame. Such a technique usually minimizes the sum of the squares of the differences in pixel values, and is therefore termed spatial cross-correlation. Tuttle and Gall (1999), for example, take rectangular subgrids of radar data, move it around the previous frame and use the minimal movement to estimate movement of tropical cyclones.

Wolfson et al. 1999 improved the basic cross-correlation technique by smoothing the images with a filter bank consisting of elliptical convolution filters (to smooth along the storm front), added smoothing criteria so that adjacent pixels do not have wildly different motion estimates and incorporated a global motion vector to eliminate outliers at the expense of being unable to track circular movements such as hurricanes. The Lagrangian technique of Turner et al. 2004 explicitly formulates this as a multi-variable optimization problem but avoids the global motion vector criterion to retain the ability to track continent-scale flows in multiple directions.

Object association techniques yield more accurate wind speeds for small-scale storms, but the movement of larger-scale features is better predicted by optical flow methods. One drawback of optical flow methods is an inability to obtain long-term statistics about the trends of object properties, because no objects are identified in these techniques. Lakshmanan et al. 2003 introduced a hybrid technique that estimates movement of objects identified through hierarchical segmentation using optical flow techniques. Because the optical flow movement is estimated, not on rectangular subgrids of the image, but on templates the size and shape of the identified clusters, the clusters can be displaced through multiple historical frames to yield object statistics.

16.6.4 Trends of Geographic Information

So far, we have looked at temporal changes in objects, such as storm systems, that move. How about obtaining the temporal properties from a stationary viewpoint, such as the total precipitation that falls within the boundaries of a city?

The simplistic approach is to map the geographic data onto the same grid as the remotely sensed data and then perform the accumulation operation on the pixels that correspond to the city. However, gridding geographic information system (GIS) data is not always practical. Ideally, it would be possible to leave GIS data in its original vector form and compute spatial properties that fall within the vector.

To decide whether a pixel falls within a polygon, simply map the vertices of the polygon to the gridded

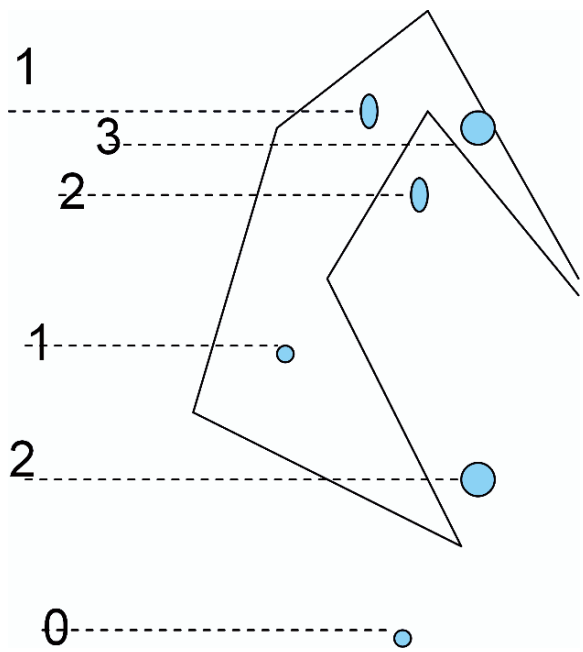

Fig. 16.10 Count the number of times an edge of a polygon is encountered to determine whether a pixel in question falls within the polygon

reference i.e. convert the polygon vertices to (x,y) locations. Assuming, for simplicity that the polygon is completely within the grid, walk row-wise from the grid boundary to the pixel of interest, counting the number of times that the sides of the polygon are crossed (this can be done by finding the intersection point of the line corresponding to the row and the line corresponding to each of the edges of the polygon – see any geometry book for details on the formulae to do this). If the total number of crossings is odd both when walking row-wise (see Fig. 16.10) and when walking column-wise, the pixel falls inside the polygon and its pixel value contributes to polygon statistics.

16.7 Summary

In this chapter, we have looked at how image processing plays a key role in artificial applications that operate on spatial data. We discussed techniques to convert in situ observations into spatial grids. We also examined ways to reduce noise, find edges and identify objects from spatial grids. Finally, we looked at purely image-driven AI applications such as tracking and the extraction of spatial properties within vector boundaries.

References

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *8*(6), 679–698.

Carvalho, L., & Jones, C. (2001). A satellite method to identify structural properties of mesoscale convective systems based on the Maximum Spatial Correlation Tracking Technique (MASCOTTE). *Journal of Applied Meteorology*, *40*(10), 1683–1701.

Cressman, G. P. (1959). An operational objective analysis system. *Monthly Weather Review*, *87*, 367–374.

Dixon, M., & Wiener, G. (1993). TITAN: Thunderstorm identification, tracking, analysis and nowcasting: A radar-based methodology. *Journal of Atmospheric and Oceanic Technology*, *10*, 6.

Johnson, J. T., MacKeen, P. L., Witt, A., Mitchell, E. D., Stumpf, G. J., Eilts, M. D., et al. (1998). The Storm Cell Identification and Tracking (SCIT) algorithm: An enhanced WSR-88D algorithm. *Weather and Forecasting*, *13*, 263–276.

Koch, S. E., DesJardins, M., & Kocin, P. J. (1983). An interactive Barnes objective map analysis scheme for use with satellite and conventional data. *Journal of Climate and Applied Meteorology*, *22*, 1487–1503.

Lakshmanan, V. (2000). Speeding up a large scale filter. *Journal of Oceanic and Atmospheric Technology*, *17*, 468–473.

Lakshmanan, V. (2004). A separable filter for directional smoothing. *IEEE Geoscience and Remote Sensing Letters*, *1*, 192–195.

Lakshmanan, V., Rabin, R., & DeBrunner, V. (2003). Multiscale storm identification and forecast. *Journal of Atmospheric Research*, 367–380.

Lakshmanan, V., Fritz, A., Smith, T., Hondl, K., & Stumpf, G. J. (2007). An automated technique to quality control radar reflectivity data. *Journal of Applied Meteorology*, *46*, 288–305.

Oliver, M. A., & Webster, R. (1990). Kriging: A method of interpolation for geographical information system, Int'l. *Journal of Geographical Information Systems*, *4*(3), 313–332.

Turner, B., Zawadzki, I., & Germann, U. (2004). Predictability of precipitation from continental radar images. Part III: Operational nowcasting implementation (MAPLE). *Journal of Applied Meteorology*, *43*(2), 231–248.

Tuttle, J., & Gall, R. (1999). A single-radar technique for estimating the winds in tropical cyclones. *Bulletin of the American Meteorological Society*, 80, 683–685.

Vincent, L., & Soille, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *PAMI(13)*, *6*, 583–598.

Wolfson, M., Forman, B. E., Hallowell, R. G., & Moore, M. P. (1999). The growth and decay storm tracker. *Eighth Conference on Aviation* (pp. 58–62). Dallas, TX: American Meteorological Society.