Samet, Hanan. 1984. The quadtree and related hierarchical data structures. *ACM Computing Surveys* 16: 187–260.
Samet, Hanan. 1990. *The Design and Analysis of Spatial Data Structures*. Reading, Massachussetts, USA: Addison-Wesley.
Samet, Hanan. 1990. *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS*. Reading, Massachussetts, USA: Addison-Wesley.

7
Manipulations

Interpolations, geometric operations, transformations

Billy Jackson wants a method to know how to interpolate geological layers between two borehole measures. Christegonde de l'Aigliere wishes to know whether she is inside or outside Amazonia. David Marcowitz wants to go to the very centre of Russia. Maria Schwarzenberg looks to find where she crosses all country boundaries when flying from Berlin to Tokyo. Min Wong wishes to convert her digital remote sensing data to polygons and lines. José María Gallegos plans to build a quadtree out of a digitized map. Jan van Delft wants to update a map of highways in the Netherlands from new air photographs. And Krading Gingging is fixated by a pixilated key-ring.

The purpose of this chapter is to alert the reader to some issues about the manipulation of spatial data and objects, especially the use of computational geometry. When dealing with geometric information, it is necessary to have several procedures to handle spatial entities. This chapter first provides a rapid introduction to the many ways in which to manipulate geometric objects, particularly for undertaking interpolation, finding line intersections, ascertaining if a point is inside a polygon, or finding a centroid for an areal unit. We concentrate on actions which use positional information, but illustrate some which also require topological data.

The chapter also presents a framework for spatial data transformations, including geometrical, topological and between different representations. Most operations which require attribute data for spatial objects are dealt with in Chapter 8; these include algorithms for spatial analysis, using graph data, such as routing and path finding, or polygon attributes, like districting. Spatial map overlay modelling, whether using cell data or polygons, is also dealt with in the next chapter.

We do not aim to demonstrate all categories of spatial data manipulation, nor provide comparative algorithms for specific requirements. Readings in the bibliography, especially Preparata and Shamos (1986), provide extensive treatments of these topics. We believe it is more important to demonstrate the varied ways in which spatial data may have to be dealt with, and the flavour of the approaches that may be used.

7.1 INTERPOLATION AND EXTRAPOLATION

Concentrating on positional information, we present in this and the next section some aspects of geometric inference and computational geometry useful in the domain of spatial information systems:

1. Interpolation and extrapolation.
2. Line and segment intersections.
3. Point-in-polygon operations.
4. Determination of centroids.
5. Operations on polygons.

In these discussions some practical considerations also will be mentioned.

7.1.1 The interpolation and extrapolation concept

For situations in which we have limited information, possibly a planned scientific sample for selected points, lines or areas, or simply just what spatial units are available in a practical sense, it is often necessary to interpolate or to extrapolate in order to get more information, that is, to 'guesstimate' a new value or set of values. We have already alluded to this need in discussing triangulated area networks, and it also arises in allocating values or attributes from one or a set of polygons to others created by overlap; or from one or a set of linear features to others created by coincidence.

For now, we consider only **point-oriented interpolation**. That is, data for one or more points in space are used to produce estimated values for other positions at which it is not possible to record values directly. Several interpolation approaches are demonstrated in Figure 7.1. The horizontal axis is assumed to be a distance scale, although it could also be time or values of an attribute. In sequence the methods are:

1. Nearest value interpolation by which we confer the value of the nearest data point.
2. Linear interpolation, in which a simple straight line based on two points is used.
3. Spline interpolation based on three or more points.
4. Stochastic interpolation based on a pseudorandom number generator and a fractal parameter and several points.
5. Model-based interpolation.

Estimation could also use a set or an average of nearby data points rather than just the nearest. Proximal interpolation in the form of

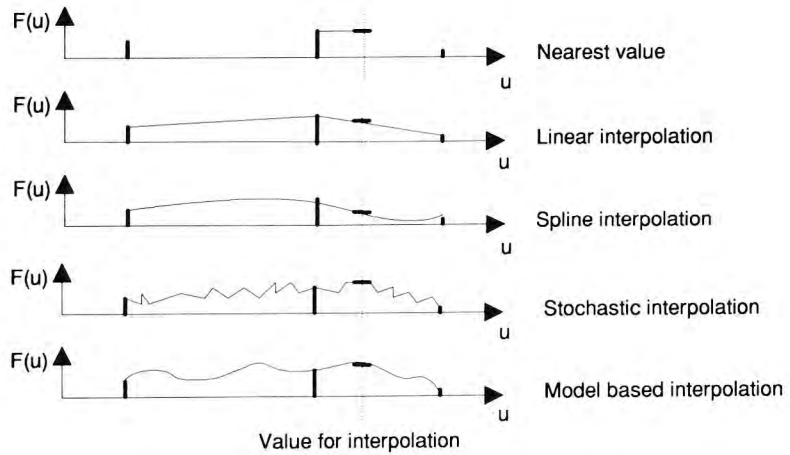


Figure 7.1 Several possibilities for interpolation.

Thiessen polygons has already been encountered in section 6.6.1. Spline interpolation was presented in section 4.5.2, and fractal interpolation in section 4.6.2.

When a more specific simulation of the phenomenon of interest is known, perhaps as a result of a mathematical model or scientific theory, it must be used in order to confer more confidence in the estimation of new values. Thus, a trigonometric function could be used to interpolate from a cyclic progression if a theory suggested a phenomenon occurs with repeated patterns. For every method, a certain level of precision must be established, and each procedure has some inherent limitations of data accuracy.

When we are outside the range of data points, we speak about **extrapolation**. As for interpolation, several kinds of estimation can be utilized (Figure 7.2):

1. Nearest value extrapolation based on the last data point or an average of neighbours.
2. Linear extrapolation based on the two last data points.
3. Spline extrapolation based on several data points.
4. Stochastic extrapolation based on fractal parameters.
5. Model-based extrapolation.

Again, the last is perhaps the more prudent approach and provides more confidence in the results. When extrapolating, as for interpolating, a certain level of precision must be given, although in particular contexts it may be difficult to establish how much error can be tolerated.

In the foregoing context, represented by the diagrams, only one spatial

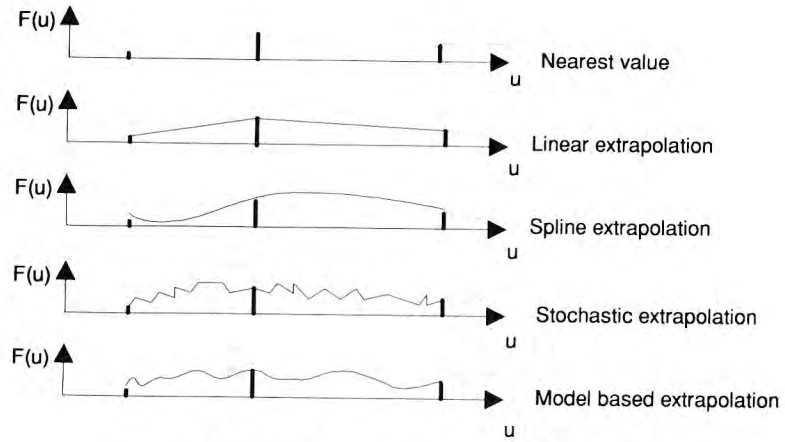


Figure 7.2 Several possibilities for extrapolation.

parameter (coordinate axis direction) was used in the interpolation or the extrapolation. However, there are more usual situations; for example, in terrain modelling, when we estimate new values for a two-dimensional surface based on neighbouring elevation values. Sometimes, we have to interpolate in one coordinate dimension, and extrapolate in the second coordinate dimension. For this type of surface modelling, the term **geometric inference** is sometimes used, meaning that we are in a complex situation characterized by:

1. Two or three space coordinate dimensions.
2. Various methods for selecting data points.
3. Either interpolation or extrapolation or both.
4. Selection of the type of procedure (linear, spline, model based, etc.).

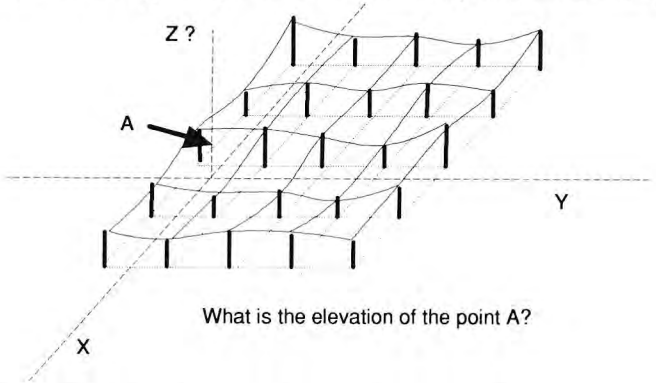


Figure 7.3 Inferring a new elevation point in a gridded terrain.

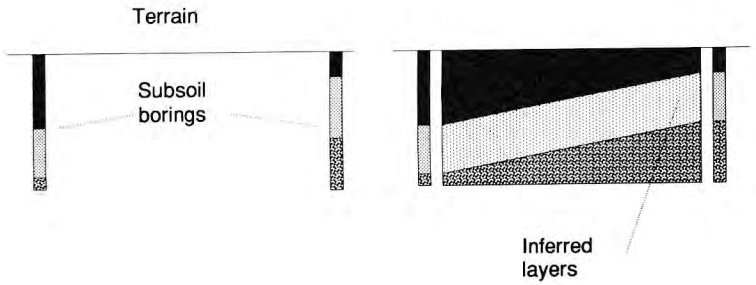


Figure 7.4 Inference of geological layers from borings.

This field-oriented approach is illustrated assuming the estimated variable z is elevation above sea level, or depth below a vertical datum. However, while other attributes, like temperature or average income can be imagined, the predicted attribute must be in the form of scalar values, not as categorical or ordered data.

Digital terrain or digital elevation models which give the observed or recorded elevation for some **privileged points**, for example along a grid or along contour lines, are often used in order to infer values for some new points (Figure 7.3). In this case, the observed values are at a regular grid spacing, and interpolation occurs along both coordinate axes, perhaps using a mathematical model like a high order polynomial. Another example, from geology (Figure 7.4), is the inference of the subterranean layer structure from borings. This time there is two-dimensional estimation, but also, in practice, the inferences are made from very few points. In general, geological interpolation or extrapolation is very difficult because there is usually very little recorded data, and often considerable real world variability even if mathematical models might be considered as appropriate for a given context.

7.1.2 Some practicalities

Sometimes the privileged points are irregularly distributed in space, and the elevations are estimated for the intersections of grid lines or a lattice of points. In this context it is most important to make a prudent selection of data points. Conceivably, all point samples could be used to obtain a value for z for positions at which no observed values exist. At the other extreme, the estimate can be based on only one known value, the nearest in space. The data points used can be chosen on the basis of several criteria:

1. The number of data points to be used for each lattice position.
2. The distance away from the lattice position.
3. The location within a compass direction away from the lattice position.
4. The interpolation or extrapolation procedure used.

The nearest value, straight line and spline interpolations use a restricted number of data points, but even so, each one of these could itself be an average of a set of points. In any event, if a procedure can use a large number of points of observed values, then the decision as to how many to use is based partially on the extra accuracy benefits that may arise relative to the effort needed to process more data. For example, for a set of 3,100 points representing counties of the USA, an estimate for a location in California is not likely to be much influenced by conditions in Florida, represented by about 100 points at a considerable distance from California.

Generally speaking, nearby data points are given more weight than those further away, and there may be a limit to the spatial range used. Sometimes only immediate neighbours are used unless some compass directions are not represented in the set of data. In this case, say, for four or eight compass sectors, each must contain at least one point of data even if it is positioned a long way from the lattice position, in order to mitigate possible effects of spatial bias.

Consequently, in geometric inference it is important to be sensitive to the facts that:

1. The number of sample points may be an unrepresentative spatial distribution.
2. There is a condition of spatial autocorrelation; that is, places closer together in space tend to be more similar than different.

The first statement means that attempts must be made to cover all space in a study area with a high enough sample of data points to reflect its nature. Otherwise some natural variability may not be detected, or the estimated values will be systematically biased or have an unacceptably high range of error. The second, the condition of **spatial autocorrelation**, implies that nearby data points receive the highest weight in averaging or modelling type interpolations, or that points beyond a certain distance can be eliminated. One modelling technique developed by mining geologists, **kriging**, specifically uses information for the spatial autocorrelation, the numerical measurement of the degree to which points different distances apart in space are alike in regard to an attribute.

Interpolation of an attribute over space will naturally be more reliable if more good data are used, but the outcome is also affected, often

substantially, by the procedure used. Dealing only with point interpolations still, important considerations are:

1. If the procedures are local or more extended spatially.
2. If they produce an outcome that honours the input data.
3. If the reality being modelled is broken or has barriers.

Some techniques are local in orientation, that is, they are applied systematically to different parts of a study area, using data limited to one point or gathered from a neighbourhood, as for proximal mapping and piecewise spline patches. Other techniques fit a continuous surface to a set of points scattered over space, as with trend surface analysis using functions like polynomials or sines, or consist of integrative measures of accessibility or space potentials. Such functions use all data points, that is they are **global interpolators**. Otherwise, they may use data from a large area, referred to as regional interpolators. Local processing has advantages of being faster because fewer data are needed, but global operations may provide better results under some conditions.

Some procedures will produce estimated values for the positions of observed data that are not the same as those observed values. Exact interpolation produces a surface that honours all known values, but some procedures fit a global or regional trend that is not constrained to the local circumstances. Kriging and B-splines ordinarily are exact interpolators; polynomial trend surfaces and moving averages are not.

As noted in earlier chapters, the interpolation may be made to recognize natural conditions like breaks in slope or barriers. The latter, usually either linear or areal in geometry, include impenetrable features like geological faults, some political boundaries, or inhospitable terrain; or permeable barriers like landscapes of different ease of transportation. Geometrically speaking, trend projections would have to be truncated or weighted, respectively, to recognize these two contexts, in order to produce abrupt or gradual changes. The latter may be easier to deal with in practice, but many situations are contrary to unbroken continuous space, if only because of the existence of anisotropicity.

Spatial interpolation may also be desired or necessary in cases of linear or areal phenomena. Examples encountered earlier include comparisons of spatial units with different boundaries or extents at different points in time, or an allocation of attributes such as the number of voters per constituency to census tracts. In addition to proportional area techniques, areal interpolation can use the intermediate steps (as discussed in section 7.3.4) of representing polygon data by sets of grid cells and then producing a continuous surface.

7.2 BASIC OPERATIONS ON LINES AND POINTS

Several of the most common needs in spatial information systems, access to phenomena encoded as entities, or determination of overlaps that result from map overlays or geometric window searches, utilize just a few basic, but very important, operations on the spatial data primitives of points and lines.

7.2.1 Line intersections

There are several contexts in which it is necessary to find the intersection of line entities. Not only is it necessary for determining topological junctions from separately digitized line features, or for finding where a line meets a map edge, but also for solving the important polygon overlay problem. We can differentiate between two contexts: one in which lines are of infinite length, and may or may not cross; the other in which there are straight line segments terminating in nodes or vertices and may or may not intersect. For representation, there are many mathematical expressions to obtain the locus of a line.

To represent a straight line in the two-dimensional Cartesian space, a **linear equation** is generally used:

$$ux + vy + w = 0$$

Some people prefer $y = a + bx$ where a is the intercept and b the slope. Unfortunately, this technique produces some difficulties in representing vertical or horizontal lines parallel to the coordinate axes, in particular because the slope in this case is zero.

For representing segments, we need to know the end-points x_A, y_A and x_B, y_B (Figure 7.5a) and the line equation. Using the linear equation cited above, where

$$u = y_B - y_A$$

$$v = x_A - x_B$$

$$w = y_A x_B - x_A y_B$$

and, considering another line, passing through points C and D, so the coordinates of the intersection are:

$$x_i = \frac{(y_C x_D - x_C y_D)(x_A - x_B) - (y_A x_B - x_A y_B)(x_C - x_D)}{(y_B - y_A)(x_C - x_D) - (y_D - y_C)(x_A - x_B)}$$

$$y_i = \frac{(y_A x_B - x_A y_B)(y_D - y_C) - (y_C x_D - x_C y_D)(y_B - y_A)}{(y_B - y_A)(x_C - x_D) - (y_D - y_C)(x_A - x_B)}$$

These equations can constitute the **point-in-line rule** which is used in Chapter 10.

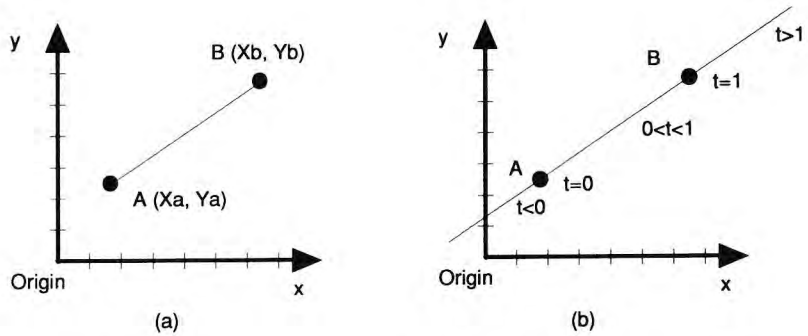


Figure 7.5 Segment represented by end point coordinates and parameter. (a) Segment represented by end-point coordinates. (b) Segment represented by end-point coordinates and parameter.

7.2.2 Segment intersections

It is rare in geomatics to have to deal with an infinite line. Generally speaking, we have line segments and for representing them we must have:

1. The line equation.
2. The end point coordinates.
3. A way to refer to only between those two end points.

The linear equation solution given above is not very convenient for these conditions. Another possibility is to use the so-called **parameter expression**, that is, have a parameter t varying from 0 to 1 for the relevant portion of the line (for example, Figure 7.5b), and take a value of 0 at one end and increase to 1 at the other. So we have:

$$x = x_A + t(x_B - x_A)$$

$$y = y_A + t(y_B - y_A) \qquad \text{where } 0 \leqslant t \leqslant 1$$

Concerning the intersections of two segments (Figure 7.6), suppose that we have two segments AB and CD. Using t and s to denote parameters, the corresponding equations are as follows. For the first segment:

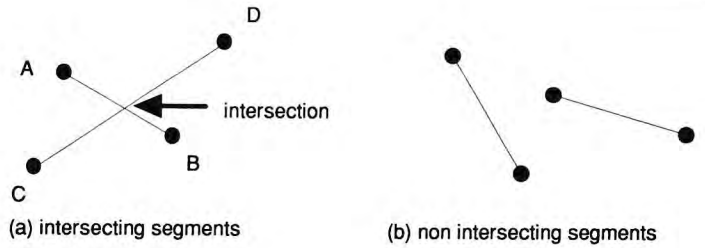


Figure 7.6 Intersection of segments. (a) Intersecting segments. (b) Nonintersecting segments.

$$x = x_A + t(x_B - x_A)$$

$$y = y_A + t(y_B - y_A)$$

For the second segment:

$$x = x_C + s(x_D - x_C)$$

$$y = y_C + s(y_D - y_C)$$

At the intersection the parameters t and s are given by:

$$t = \frac{(x_C - x_A)(y_C - y_D) - (x_C - x_D)(y_C - y_A)}{(x_B - x_A)(y_C - y_D) - (x_C - x_D)(y_B - y_A)}$$

$$s = \frac{(x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A)}{(x_B - x_A)(y_C - y_D) - (x_C - x_D)(y_B - y_A)}$$

provided that $0 \leqslant t \leqslant 1$ and $0 \leqslant s \leqslant 1$.

In order to compute the intersection for two line segments, we have two possibilities:

1. To compute the intersection point of both supporting lines and to check whether the point lies in the segments.
2. To use the parametric equations, checking whether both parameters, t and s, range from 0 to 1.

This second method constitutes the **point-in-segment rule** which will be encountered again in Chapter 10. Sometimes known as the point–vector form, this representation for a straight line has valuable properties of being able to be used for lines and line segments, and does not require special treatments for vertical or horizontal lines parallel to coordinate axes.

For either infinitely long lines or abbreviated line segments, identification of lines that may cross can conveniently take advantage of a simple

initial procedure that compares the coordinates for the corners of enclosing rectangles. (This concept is discussed in section 4.2.2.) If the bounding rectangles do not overlap at all, then the lines contained within them cannot possibly intersect. If there are many lines or segments, then preliminary sorting in ranges of x and y coordinates can help the process of finding possible intersections, even if enclosing rectangles are not used.

7.2.3 Point-in-polygon procedure

An important geometric operation, known as the point-in-polygon procedure, is to ascertain if a discrete point, or the end or vertex of a line, lies within a particular polygon. Of the several algorithms that exist, the most well-known is based on the half-line (Jordan) theorem. Let there be an (x, y) point and a polygon (for example, Figure 7.7). In order to know whether the point is inside or outside the polygon, we generate a half-line starting from this point and extending to beyond the range of the known extreme coordinates for the polygon. We then count the number of intersections of this half-line, a line projected in only one direction from the point, and the polygon boundaries. When this number is odd, the point is inside the polygon, and when even, the point is outside. The procedure incorporates a stage of counting the line intersections. In order to facilitate this procedure we ordinarily use half-lines parallel to the x-axis.

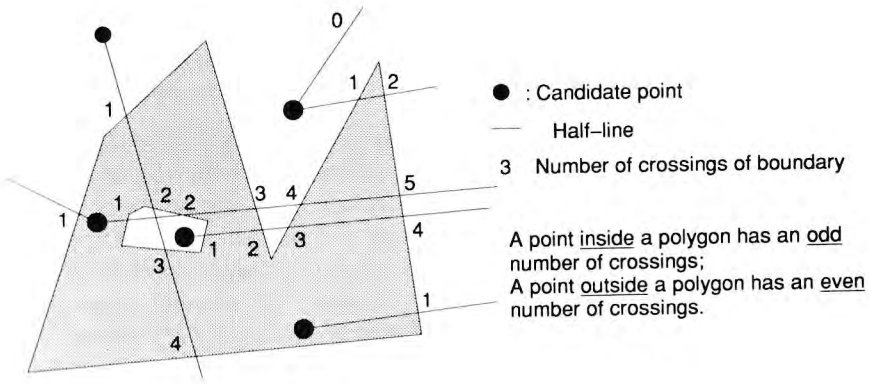


Figure 7.7 Illustration of the half-line theorem (point-in-polygon rule).

The procedure works well for so-called special cases of island polygons, polygons with holes or concave polygons. Some decision rule is needed, though, if a point falls exactly on a boundary line, or when it falls exactly on a vertex, or when a segment is collinear to the half-line. For all these situations, when the point originates from a mouse device, we usually move it slightly. When the half-line is parallel to one Cartesian coordinate system axis the algorithm for counting intersections is simpler, as explained below in the discussion of the segment-in-polygon algorithm.

In order to speed up this point-in-polygon algorithm, especially when the polygon has many edges, the minimum bounding rectangle, also called an extent, is used. Indeed, it is easier to check whether or not a candidate point is inside or outside an extent rather than for a complex polygon. When the point is outside the enclosing rectangle, it will be outside the polygon. When it is inside, though, it is necessary to apply the Jordan theorem. This speed-up procedure is usually used in database retrieval since the probability of a point being inside is very low. This algorithm will be used for the **point-in-polygon rule** (discussed again in Chapter 10).

Generally speaking, the problem is not to find whether a point lies inside or outside a single polygon, but to determine as to which spatial unit of a tessellation the point belongs, that is a **point-in-tessellation test**. A possibility is to use repeatedly the point-in-polygon algorithm for each polygon; but this solution is very time consuming. Suppose one has 1,000,000 polygons, then there is an average of 500,000 tests to perform. In other words, the search time, the time taken to find the polygon in which the single point lies, is linear; that is, it is proportional to the number of units. However, in order to undertake this task more rapidly, two acceleration techniques exist:

1. To evaluate the extent before the entire polygon is tested.
2. To organize polygons hierarchically.

The **hierarchical organization** functions as follows. Suppose we wish to know in which town in the USA an x-y point, perhaps representing a natural hazard, belongs. A naive algorithm is to apply the point-in-polygon test to all polygons representing towns. Another possibility is first to test the states, and when a state is found, to test a county, and so on. If there is even just one level of hierarchy, the number of separate tests is substantially reduced. The N polygons can be regrouped into $\sqrt{N}$ group of $\sqrt{N}$ towns, so the average number of tests is $\frac{1}{2}\sqrt{N} + \frac{1}{2}\sqrt{N} = \sqrt{N}$. Compared to the complete testing situation for our example of 1,000,000 polygons, now we have but 1,000 tests to perform instead of the expectation of 500,000. We can thereby expand the number

of hierarchy levels. For example, at three levels the average number of tests is 150, at four levels there are only 20 tests, and so on.

Point-in-polygon testing occurs for several types of need in spatial information systems work. It arises most commonly in comparing the position of a point entity relative to one or more polygons, for example, a school within a county, or a television tower in an airport zone. In a database query context it arises when the cursor is used to pinpoint a geographic location for obtaining information about a polygon drawn on the monitor screen.

The general concept of point-in-polygon can be extended to also cover point-in-volume, point-on-line or line-in-polygon. The last of these requires more tests, for there are two end points of a line segment to be evaluated, and a line segment could lie outside, completely inside or partially inside a polygon (section 7.3.1).

7.2.4 Centroid definition

At various times we need to represent a polygon (or a line) by a single point, perhaps for choosing a position for map labelling purposes, or perhaps for simplifying distance measurements to other objects. Let us imagine we wish to determine a position, called a **centroid**, which is approximately in the middle of a polygon. There are several definitions, but no one is substantially better than others:

1. Defined from vertices.
2. Obtained as a statistical bivariate median or the centre of gravity.
3. Computed as the centre of an enclosing or enclosed rectangle or of an enclosing or inscribing circle.
4. Obtained as the peak value of a surface fitted within the polygon.
5. Chosen intuitively.

The first solution is to define centroid coordinates (x_c, y_c) as the **average of all vertex coordinates** (Figure 7.8a):

$$x_c = (1/N)\Sigma x_i$$

$$y_c = (1/N)\Sigma y_i$$

As seen in Figure 7.8, when a polygon is very rugged on one side, in other words, the number of vertices is very high in some parts, the vertex centroid shifts towards those vertices. In order to avoid this condition, instead of having a vertex centroid, the centre of gravity can be employed (Figure 7.8b), or a centre can be computed arithmetically from the

corners of an enclosing rectangle or circumscribing circle, or from an inscribed circle or other regular figure. Of course, additional computational steps are necessary if centres are determined for fitted figures compared to just using coordinates data as is the case of the average of vertex coordinates.

The previous definitions hold when the polygon has no big **concavities**. In fact, in this case, a vertex or a gravity centroid can be outside the polygon and we have to move it into the interior (Figure 7.8c). Similarly, when the centroid falls inside a hole within the polygon, then it again should be moved. It is in cases such as these that a visual interactive approach to positioning the central point can be useful, or a computational procedure can be used. A very crude algorithm to calculate a centroid in this context is first to compute the centre of gravity and then, using the half-line method, test whether this point is inside or outside the polygon. If so, this point becomes the centroid; otherwise the point is moved along the line which has the longer segment forming the intersection with the polygon.

7.2.5 Some spatial statistics based on point data

Distances between two points may be readily obtained from the Pythagoras' theorem:

$$\text{Distance} = [(x_B - x_A)^2 + (y_B - y_A)^2]^{\frac{1}{2}}$$

where A and B designate the two points separate in space, or at the ends of a straight line. The formula is readily extended to other metrics, especially the Manhattan distance in which the two axial distances are summed as in:

$$\text{Manhattan distance} = [|x_B - x_A| + |y_B - y_A|]$$

That is, using absolute differences, the length between points in the two axial directions.

If distances are measured along lines or links made up of more than one straight segment, then we have:

$$\text{Length} = \sum_{i=1}^{N-1} [(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2]^{\frac{1}{2}}$$

For distances from points to lines or polygon boundaries it is necessary to find first the closest line segment and then measure distance. This process may require testing against each vertex on a line, especially if it is

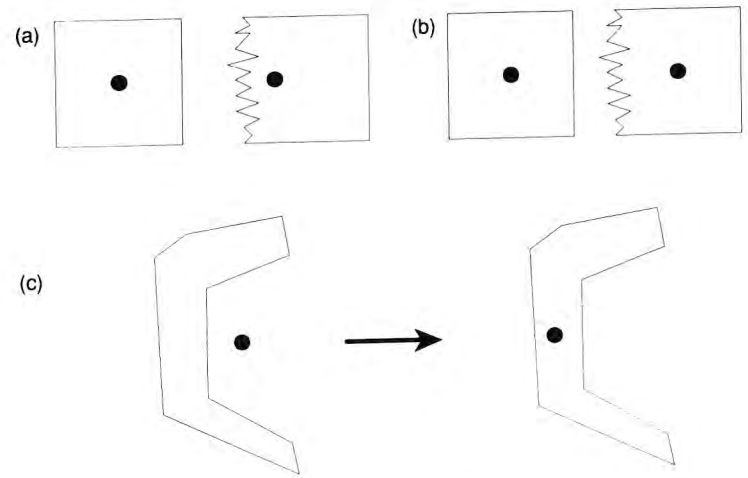


Figure 7.8 Definition of centroids. (a) Examples of centroids. (b) Examples of centroids defined as centres of gravity. (c) Moving an outside centroid.

complex, in addition to using an intersection algorithm to yield a point which is not a vertex.

The nearest neighbour statistic, a common measure for point entity distributions, requires that distances from a point to its nearest neighbour, and possibly second and higher order neighbours, be known before computing the summary statistic based on all points in the distribution. Used at times for evaluating the uniformity in a set of points prior to using geometric inference, the statistic compares the observed distribution in space with a theoretical distribution based on a process producing a random spacing pattern.

Distance measures are also involved in the computation of various spatial statistics related to the centroid or median position in two space dimensions. Ring and sector counts are based on distances from a bivariate centre, and a bivariate standard distance deviation, akin to the standard deviation, is a summary statistic of some utility.

7.3 SOME OPERATIONS FOR POLYGONS

Comparisons of different discrete polygons may require overlap of their boundaries, overlapping a rectangle or other geometric figure, or

comparing a line with a polygon boundary or interior. The first situation constitutes operations which are very common in polygon overlay, particularly unions and intersections. The second represents the box clipping problem. Other operations on single polygons viewed as separate entities include area computation, creating boundaries of buffer zones inside or outside a polygon and parallel to its boundary, and measuring shapes.

7.3.1 Intersection of lines with polygons

A common situation comprises the intersection of a segment with a polygon, or in the opposite way, the computation of what part of the segment lies inside a polygon (Figure 7.9a). In order to determine the segment intersection, we can compute the intersection of the straight line for the segment and the line segments for the polygon boundaries. If there is no intersection with the boundaries, there is no intersection of the segment and the polygon. If there are intersections of the straight line and the polygon, we can determine what parts are inside by a repetitive use of the Jordan theorem.

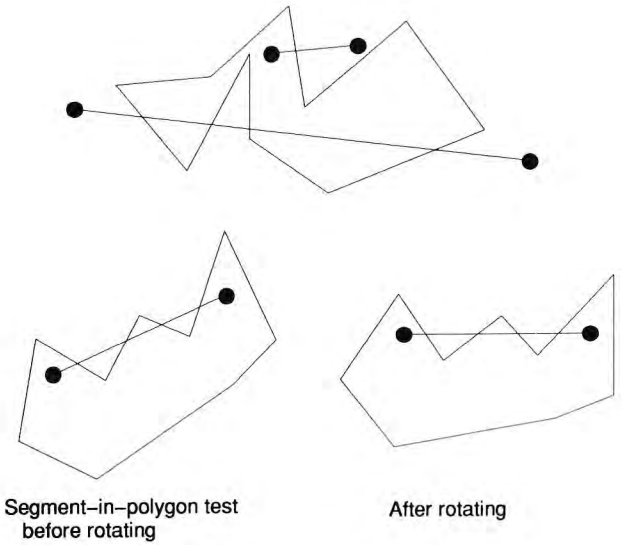


Figure 7.9 Intersection of segments with a polygon. (a) Intersection of segments with a polygon. (b) Solving the segment-in-polygon problem using the Jordan theorem.

For checking whether a segment is totally included in a polygon, it is not sufficient to test end-points because the polygons can have strange concavities intersecting this segment. Obviously we cannot examine the infinity of segment points to see whether they are all inside or outside the polygon. In order to facilitate the testing and to re-use the result of the Jordan theorem, a nice step is to rotate the polygon so that the segment under study is parallel to an axis (Figure 7.9b). After this, it is easy to count the number of intersections only by comparing coordinates. If the number is different from zero, we know that a part of the polygon is inside and a part is outside. By re-using the half-line procedure on the end-points when the number of intersections is zero, we then know whether the end-points are inside or outside, and, consequently, whether the totality of the line segment is inside or outside.

7.3.2 Union and intersection of polygons

One of the major needs and challenging problems in spatial information systems is to compute the difference, union and the intersection of polygons. For example, considering two polygons A and B (Figure 7.10a), we need to find their union; that is, their joint extent (Figure 7.10b), and their intersection, the common area (Figure 7.10c). Drawing on Preparata and Shamos (1986), who present several union and intersection algorithms, we present only possibly the simplest way, based on the **slab technique**. Each polygon is divided into parallel slabs, usually horizontal for the convenience of parallelism with the coordinate axis, as shown in Figure 7.11, created by drawing lines through the polygon vertices. This procedure creates trapezoids which are easy to compare. Figure 7.12 gives some varied examples to illustrate this method. When the edges are not intersecting inside the slabs, the comparison is

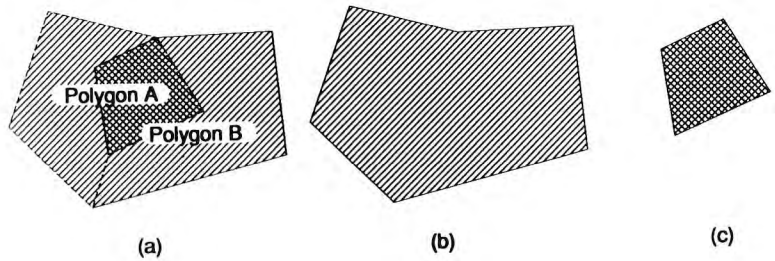


Figure 7.10 Union and intersection of polygons. (a) Two intersecting polygons. (b) Union of A and B. (c) Intersection of A and B.

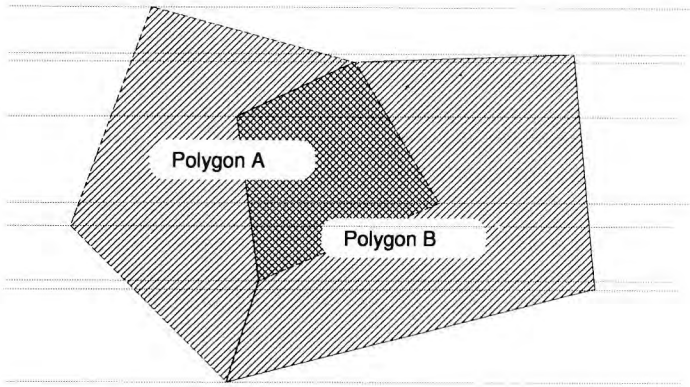


Figure 7.11 Splitting two polygons into slabs.

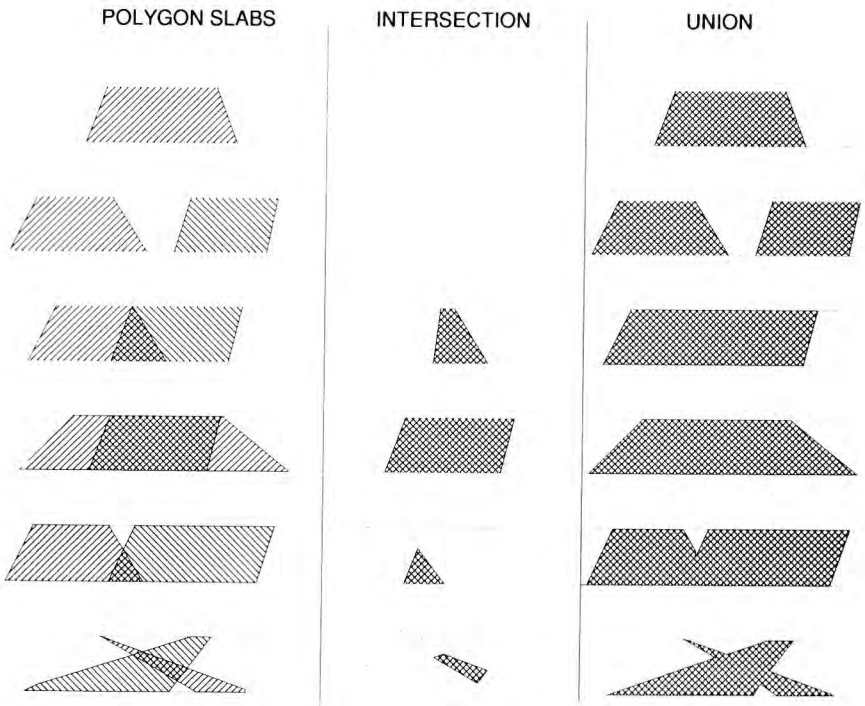


Figure 7.12 The method of slabs for finding the union and intersection of two polygons.

straightforward; otherwise, as in the last row in the diagram, some other slabs can be created passing through these intersection points.

7.3.3 Area computation

Area computation sometimes uses the trapezoid idea although a procedure using geometric cross-products is usually preferred. Lines perpendicular to the x-axis (alternatively, the y-axis could be used) are dropped to it from the vertices of polygons. This process creates a set of overlapping figures, with four line segments: the base on the x-axis, two vertical lines and an edge of a polygon. After the area of each trapezoid is computed, they are summed, including subtractions for the lower pieces. Enclaves and exclaves can be handled by this procedure by using encoded data showing the gaps or separate pieces of the polygons.

However, the classical way in computational geometry is to use the **geometric cross-product** to compute areas, and mixed products for volumes. Here we provide the procedure for only areas. For computing the area of any polygon with or without holes, if we have in total N vertices ordered in a counterclockwise sequence from 1 to N, the area is given by:

$$\text{Area} = \tfrac{1}{2}\left(\sum_{i=1}^{N-2} (x_i y_{i+1} - x_{i+1} y_i) + (x_N y_1 - x_1 y_N)\right)$$

This process, established via vector algebra using the cross-products, is more rapid than the trapezoid decomposition. The dot product device is also useful for obtaining angles between vectors, and for concatenating boundary lines for polygons using a centroid as origin for the vectors.

7.3.4 Areal interpolation

An additional interesting situation that arises for polygons is the allocation of attributes from one polygonal representation to another. This need often arises in comparing census demographic and other data for different years for spatial enumeration units that have changed between censuses. Assignment of values is easy if there is a nested hierarchy of polygon units because systematic aggregation and decomposition can be used. But, if there is overlap, then allocations have to be made for the pieces created by overlapping the polygons. The areal interpolation, a variant of point or linear based interpolation, or the transfer of attributes from one area spatial unit basis to another, may be based on any of the following:

1. Proportionate areas, assuming uniform densities of phenomena.
2. Other, correlated data.
3. A continuous surface representation of an attribute.

Not only is it necessary to create new polygons and undertake area measures by procedures described above, but it is also necessary to allocate the attribute values associated with each original polygon to the new piece. The simplest procedure, based on proportional areas, finds the area of a new polygon resulting from the intersection operation, and then allocates the attributes of the two original polygons proportional to the percentage area occupied by the new polygon. If the fortunate occurrence of another attribute distribution matching the new polygons arises, then this other variable can be used to estimate, assuming a statistical correlation exists for the two attributes.

Alternatively a field orientation, rather than a discretization implied by the use of polygons, may be a more appropriate method. In this case a surface estimation procedure will have to be used first, followed by aggregation of point estimates taken from that continuous surface. The different possibilities take advantage of the concept of a regular grid or lattice as an intermediate step.

Using an example of human population distribution, one method allocates the housing density (computed from number of houses divided by polygon area) for each areal spatial unit to its centroid, interpolates from these irregularly distributed points to a set of grid cell centroids, reconverts to the number of housing units from cell density, and aggregates to the zones for the second set of irregular areal units. An alternative procedure assumes that there will be no stepped function for the global surface and that the sum of attribute values in a zone must equal the given total. So, a cell's value is replaced by an average of its neighbours, and the sum of all cell values is adjusted to the total of the original zone. An iterative process will lead to convergence on the control values for all zones.

7.3.5 Shape measures for polygons

The shape of landscape features is an important part of the pattern recognition techniques used in image processing. Shape properties are sometimes recorded for natural phenomena for scientific purposes, especially through an approach that relates form to function, as for different types of glacial deposit or classification of coastal barrier islands. Shapes of land cover of different kinds can be quite important for wildlife, for agricultural practices or for the layout of recreational and

parkland areas. For example, long thin strips of open space can assist the movement of animal spaces through otherwise inhospitable terrain.

If only because of the concept of 'gerrymandering' associated with political redistricting, it is worthwhile to examine the measurement of shape properties of polygons. It is for supporting allegations of racial or partisan bias in drawing boundaries of new electoral districts that statistics for shape regularity or indentedness are commonly encountered. (The term gerrymandering is based on the name of Governor Gerry of Massachusetts and the form of the salamander, a shape perceived by some challengers to resemble a rather strangely formed district.)

Shape and compactness measures may utilize sets of parameters defining particular properties of areal objects, or may use data for sets of line segments fitted to the boundaries. For the former, area, perimeter, longest axis, shortest axis, centroids, radius of circumscribing or inscribing figures are encountered – shape indices are computed from two or more of these basic measures. The difficulty in producing a single measure reflects the multi-faceted nature of shape, involving conceptually distinctive elements of elongation, compactness, puncturedness and fragmentedness.

We present just two methods to illustrate the underlying computation (Figure 7.13). Firstly, the perimeter of a polygon, obtained by summing the lengths of line segments, is compared with the perimeter of a circle of the same area as the polygon. Although different classes of shape can produce the same statistical value, the measure does pick out the irregularity in figures. Secondly, again comparing with a regular figure of a circle, the departure from the perfect form can be detected by the length of radials drawn from the centre of the areal unit, as illustrated. This procedure is computationally more involved than the other, because line intersections must be determined, and requires finding a good centroid.

7.3.6 Polygon clipping

A clipping situation occurs commonly when displaying or retrieving spatial entities within a defined area like a rectangle. When one has a rectangular window on a screen or on another graphic device, it is necessary to know what part of the object (say, a house) has to be displayed on the screen. Similarly, when using enclosing rectangles for overlay purposes, particular polygon or line objects may be cut. Three situations arise, illustrated for the display context (Figure 7.14):

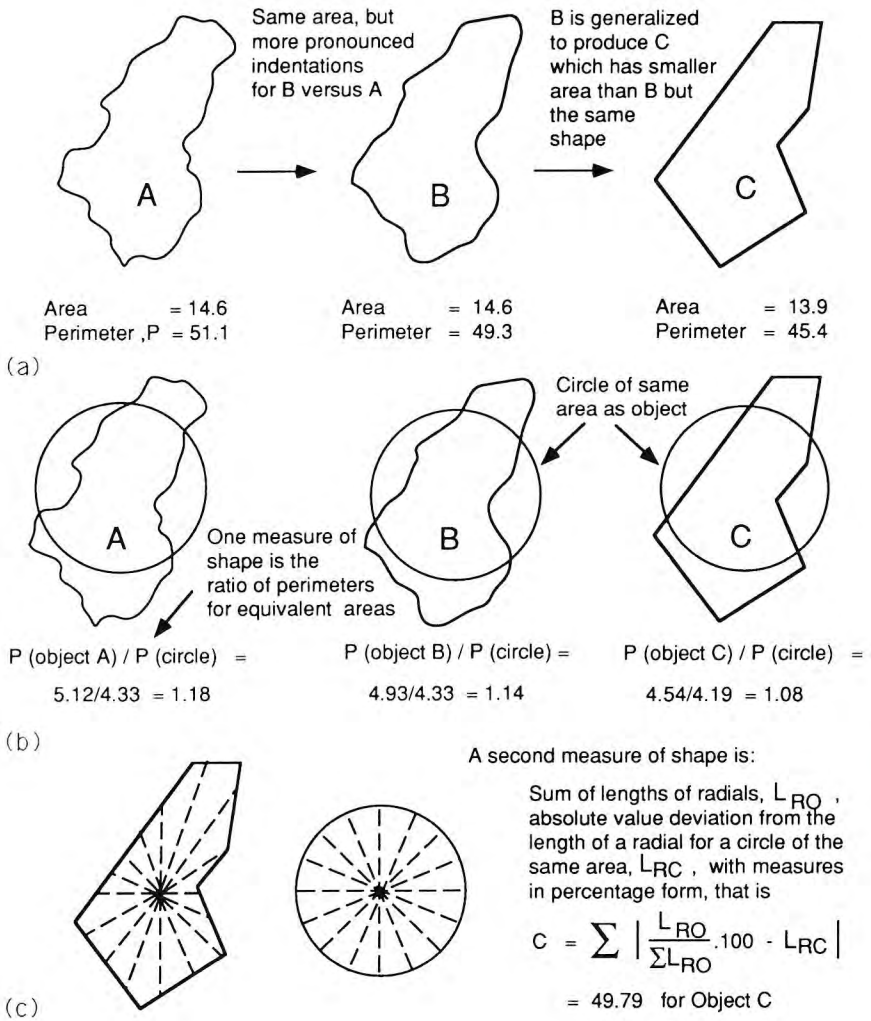


Figure 7.13 Measures of compactness. (a) Influence of generalization. (b) Compactness I Index: Simpler index based on comparison with an equivalent area circle. (c) Compactness II Index: More complex index requiring line inter-section computations. (From Thompson et al, reference cited in Chapter 8.)

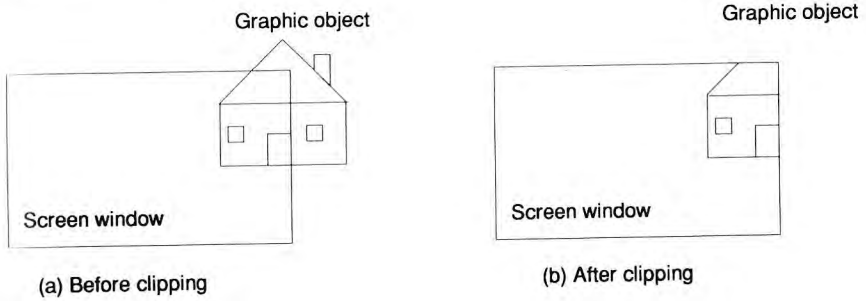


Figure 7.14 Clipping of an object by a rectangular window.

1. If the object is completely inside the window, it must be totally displayed.
2. If the object is totally outside the window, there is nothing to be done.
3. If the object is overlapped by the window, the part to be drawn must be computed and displayed.

In order to give just a flavour of a clipping algorithm, only the problem of segment clipping will be presented. In this algorithm, the two-dimensional space is divided into nine subspaces by projection of the rectangle bounding lines, and Boolean codes (binary digits) are conferred to the segment end points based on location relative to the two horizontal and vertical axes.

The first bit must be set (to 1) when the extremity is above the top line, the second when below the bottom line, the third when more at the right than the right line, and the last more at the left than the left line (Figure 7.15). Once this encoding is done, we test all extremities and the clipping must happen; in other words, we can compute the segment part to be displayed. As an output of this algorithm, the segment can be totally displayed, removed or partially displayed.

In the situation of clipping a polygonal area, it is necessary to follow both the edges of the area and those of the window, requiring an algorithm a little more complex than that just illustrated.

7.3.7 Buffer zones

One other class of spatial operations includes the creation of boundaries, inside or outside an existing polygon, offset by a certain distance, and parallel to the boundary. Referred to, respectively, by the terms **skeleton** and **buffer zones**, these new units require distance measures from selected points on the boundary of the polygon (Figure 7.16a). The skeleton is

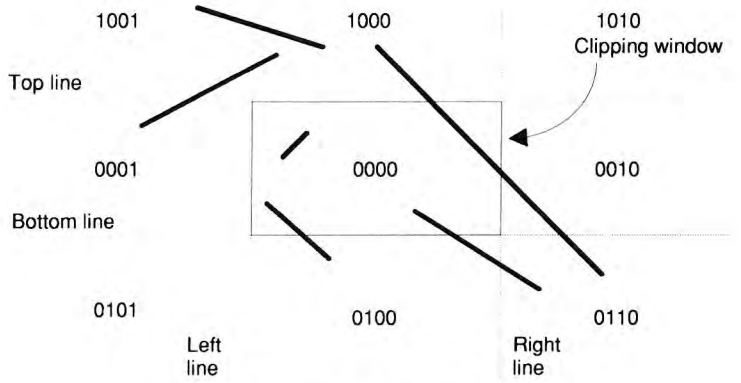


Figure 7.15 Clipping a segment.

akin to contracting the polygon by moving straight line segments in, parallel to their original position, and the buffer has lines moving outwards. Skeletons (Figure 7.16b) have some value in assisting labelling operations for polygons or edges; buffers are much used in spatial analysis and modelling.

For example, as demonstrated in Figure 2.18, buffers are used to establish critical areas for analysis or to indicate proximity or accessibility conditions. In contrast to the simpler world of proximity values for grid cell data, for vector data, computational geometry operations are required to establish buffer zones. If the concept of gradation in accessibility is required, then a sequence of buffers at selected increasing distances must be created. Similarly, for a zone within two specified distances, two new buffer boundaries have to be determined (Figure 7.16c). If a vector data representation is required, then this buffering operation is best done as a batch or background process because of the time it can take. In contrast, the equivalent operation for regular tessellations can be done much faster because it does not need to employ time consuming computational geometry operations.

7.3.8 Polygon overlay process

The matter of polygon overlays as a whole is perhaps the most challenging computational requirement for vector type spatial databases. We have already shown the need for some line segment intersection procedure; areal measures are often required too, and new spatial units have to be created. The general process of overlay to create new polygons consists of:

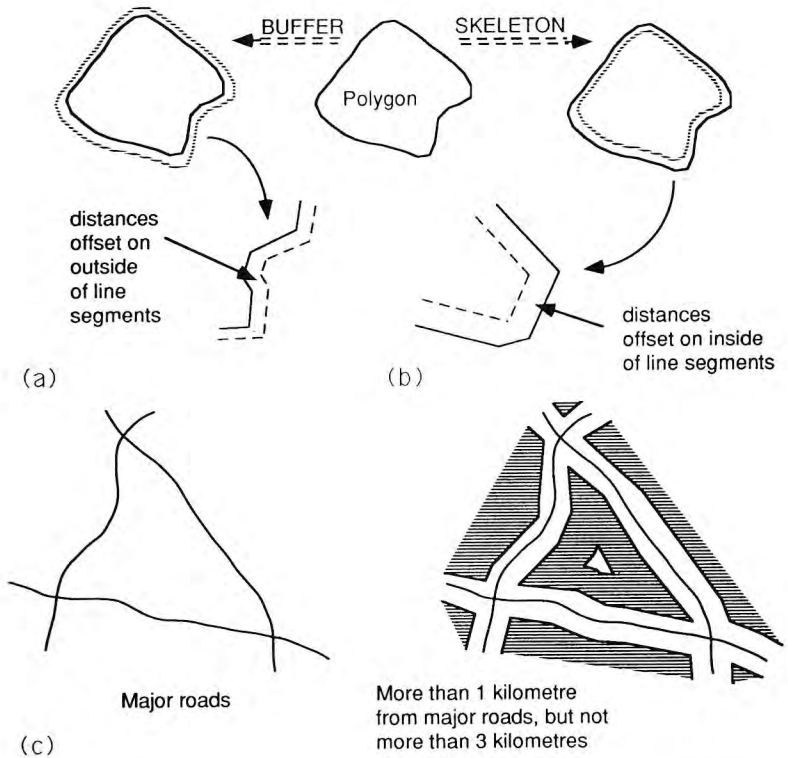


Figure 7.16 The concepts of buffers. (a) Buffer zones. (b) Skeleton zones. (c) Example of a buffer zone for major roads.

1. Identifying line segments, preferably having topology.
2. Establishing minimum enclosing rectangles for the polygons.
3. Ascertaining if line segment(s) of one polygon are inside a polygon of the overlay map by a point-in-polygon process.
4. Finding intersections of segments representing boundaries.
5. Creating records for new line segments and their associated topology.
6. Assembling the new polygons from the appropriate line segments.
7. Relabelling polygons and reallocating the attribute data if appropriate.

Taking a somewhat simple case as an example (Figure 7.17), after rectangle overlap is found, then a point-in-polygon test procedure can be used to determine if line segments fall in particular polygons. After the vertex of intersections is computed for the selected lines, the left- and right-side identifiers are applied from the original segments to the new segments; and lines making up the new polygon can be assembled. For a

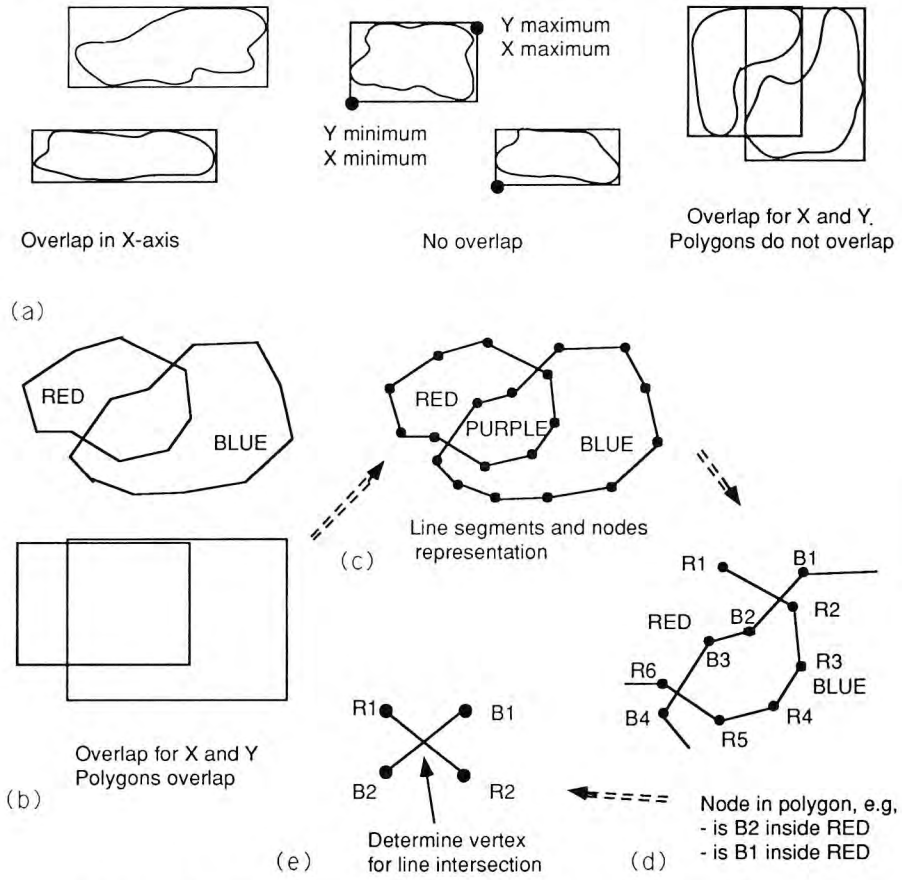


Figure 7.17 The polygon overlay process. (a) Overlap conditions for enclosing rectangles. (b) Example of overlapping polygons. (c) Node and line segment representation. (d) Point-in-polygon testing. (e) Finding vertex for intersecting line segments.

completely enclosed unit, a point-in-polygon test can be used to see if the single node lies within a larger polygon or not, unless boundaries data for ascertaining containment are explicitly encoded.

While there are ways to speed up the entire process, it is still computationally demanding of resources when there are many polygons. The time taken is influenced by the number of polygons, the number of line segments, and the complexity of the lines. Additional complications may be the need to resolve if lines touch or cross, to assess the reliability

and validity of small polygons or sliver polygons created via the overlay process, and the uncertainties of use of imprecise boundaries or tolerance values.

Sliver polygons (Figure 7.18a) must be examined and eliminated, perhaps by removal of the longest shared boundary, perhaps by a line midway between the others. Slivers may be detected, other than visually, by being small areas; having elongated shapes, as indicated by a large value for the perimeter-to-area ratio; or having only two boundary lines. Unreliability in location of new boundary lines may cause a conflict with other spatial entities, especially point features, which might now be positioned on the wrong side of a line. Such a change could have disastrous results, if, for example, a major obstacle to aircraft is shown on the wrong side of a highway. Or, one boundary, for example a river or political unit, may be unalterable in the interest of preserving 'truth'. For line entities, coincident features may have representations which have some discrepancies (Figure 7.18c).

It is also difficult to deal with the identification of containment for new polygons created by overlay processes. This is illustrated in Figure 2.17 as part of a process to create critical zones at one thousand feet from water. The buffer polygon holes were not detected directly as holes. Without special coding techniques, it is possible to detect these only visually and then take corrective action by modifying the tabulated data designating position within a buffer zone.

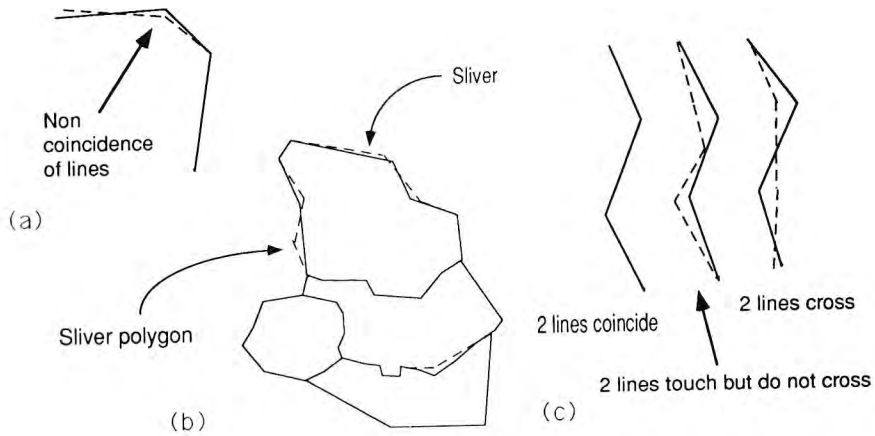


Figure 7.18 Polygon or line discrepancies. (a) Noncoincidence of lines. (b) Sliver polygons. (c) Some possibilities of line coincidence.

7.5.1 Change to regular cells

Transformation to regular cells from points, lines or polygons, already introduced in section 6.3, is perhaps easier than the reverse process. Logically straightforward, the **rasterization process** detects the occurrence of entities in the cells of the chosen size and shape, recording presence/absence, or other attributes. Basically, features are put in 'pigeon holes' by a scanning procedure, perhaps operating on rows defined by particular coordinate values, or on columns; or the combination, the two-dimensional array; or by reading lists of coordinates for points or lines. Visual encoding, the predominant practice originally for direct cell encoding from maps, is a much easier process, albeit having some practical limitations as discussed in Chapter 6.

Automated procedures may involve different types of operation:

1. Transferring Cartesian coordinates for point-and-line-entity vertices into the matrix of predetermined resolution and known positional values.
2. Use of a single scan line (row or column) or strip of several adjacent scan lines that detects intersections of cell boundaries with linear features, and records how many grid units have been passed up to the intersection.
3. For polygons, after detection of vertices, and, therefore, line segments, using a bidirectional scan traverse that knows when it reaches the edge of a polygon.

In the first situation, the coordinate values for the points or line ends are compared with the range of coordinates for each cell, or the original coordinates can be converted by a numerical factor to indicate into which cell they fall. In the extreme state, the number of cells needed can be determined from the number of coordinate positions in the original data set. Cells intermediate between line end point cells will have to be inferred in this process. Otherwise, to more directly get at the whole length of lines, the crossing of the line by the horizontal and vertical grid lines can be computed.

Except for the corner-touching case, the edge intersection will cause the two adjacent cells to be encoded with a line object. In the case of exact match at the corner intersection, all four touching cells can be encoded with presence of a line feature, or some other decision rule could be used, perhaps by looking at the other cells already encoded. In the case of lines, every cell containing even just a piece of line should be noted, otherwise lines will be incomplete in the raster representation. However, to avoid then creating fat lines in the raster, a rule of minimum

line length could be applied to the amount of edge in each cell.

For polygons, the boundaries may be encoded to cells in the same manner as for linear features, except that the attribute data for the polygons must be associated with the cells. Unless multiple attribute coding for individual cells is allowed or not wanted, a decision has to be made as to which attribute will be encoded, perhaps using a dominance rule, although this needs an extra step of computing the proportional area of a cell occupied by the different polygons. The difficulty in working with the attribute data is enlarged if the variable is continuous rather than categorical. Spatially continuous data allocated to a polygon centroid, on rasterization will be applied only to cells with points, unless the data are allocated to polygons or interpolated to a lattice before conversion to grid cells. A spatial autocorrelation function can be applied to the point data, rather than assume spatial constants in local neighbourhoods.

7.5.2 Change from regular cells to vectors

The reverse process, going from grid cells to geometric figures, usually known as **vectorization**, is more involved, partly because it requires several stages and partly because there is no topological structure already encoded in the tessellation. Ideally, we would like to:

1. Transfer topology, that is the connections and adjacencies visually apparent in the grid representation.
2. Transfer the correct shape of objects.

So there is need to vectorize and then recognize entities. For example, for sets of polygons it will be important to follow cells that make up the boundaries, creating vectors, and then assemble the vectors, including detecting junctions, to obtain polygons (Figure 7.23). Along the way, it will generally be necessary to smooth the rectangular edges of cells to eliminate the staircase effect. However, boundary smoothing is sometimes considered an optional cosmetic step.

In general, the procedure consists of the following stages. First of all, cells are coded, perhaps by binary states, so that all vertically and horizontally contiguous cells with the same attribute category can be assembled. Each such group of touching cells can then be assigned a unique label. Next it is necessary to establish the path of boundary lines by identifying the cells with different values on at least one side and then traversing the boundary from a starting point which is the corner junction of three or four cells with different values. The boundary line may be recorded as directional codes using four compass points as shown, or

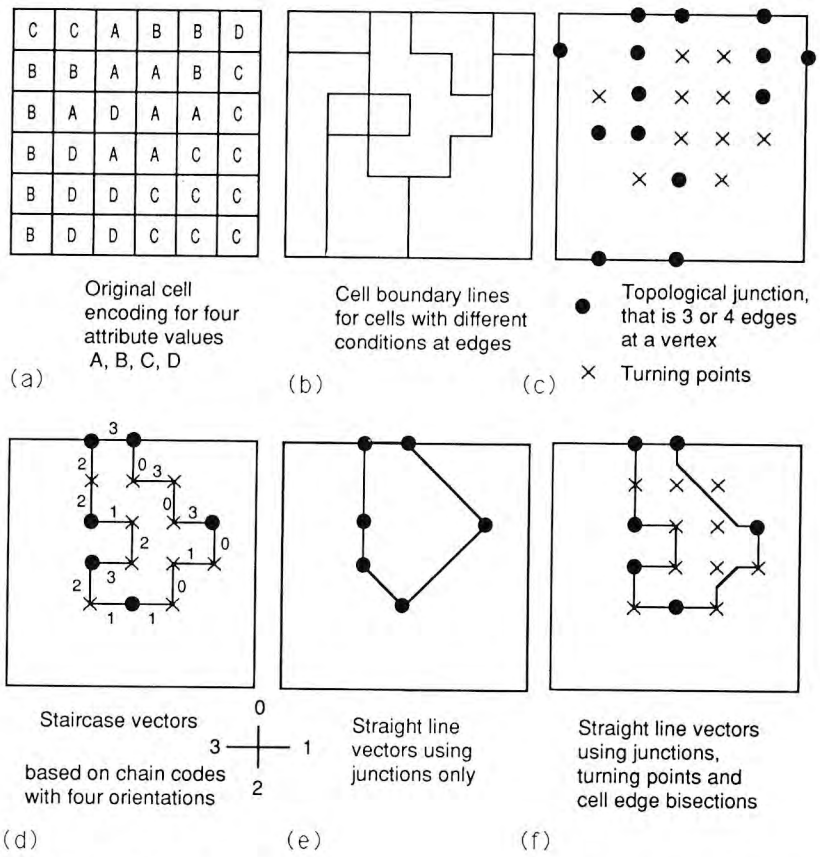


Figure 7.23 Raster to vector conversion. (a) Original cell encoding for 4 attributes (b) Cell boundary lines for cells. (c) Topological junction. (d) Staircase vectors. (e) Straight line vectors. (f) Straight line vectors using turning points.

somewhat smoothed by using eight points. Diagonals may be established, rather than simply joining corner points by straight lines, by bisecting the cell edges or a more complex procedure subdividing the cell edges based on neighbourhood conditions. Part of this process clearly involves topology construction, that is the nodes at the corners of touching polygons; and part of it attempts to create acceptable approximations for the geometry.

The construction of the vector equivalent of raster representations of lines involves line skeletonization and recognition of connectivity. In raster mode, lines have width; geometrically, they are not supposed to. So, we need to obtain a width of only one cell, generally achievable by

recognizing and eliminating cells with short lengths and allowing corner neighbours. Different algorithms exist for implementation of this general procedure (Piwowar *et al.*, 1990). Some procedures have an intermediate stage of creating graphs of vertices and edges, and simplifying by eliminating vertices through dilution when the edges are not needed for topology. Subsequently, lines can be smoothed as illustrated above for polygon boundaries.

Vectorization may be well assisted by interactive procedures. Often the smoothing can be effectively judged from the visual display of cells, clearly revealing neighbourhood conditions. Otherwise, fully automated procedures with varying success in building topology and producing acceptable geometric approximations are available. For both rasterizing and vectorizing, judgements of effectiveness can be based on several criteria; we consider accuracy of area measure, perimeter measure and minimization of feature displacement to be as important as considerations of memory space and speed of processing.

7.6 ACCESS TO SPATIAL DATA

Recalling the discussion of space partitioning by tessellations (Chapter 5) and the spatial referencing by continuous and discrete methods (Chapter 4), it is appropropriate at this point to consider the important matter of access to spatial data in a database. Generally speaking, in alphanumerical databases, access to information is based on the attributes. In spatial databases we have, in addition, location based access to data.

7.6.1 Access by identifiers and by locators

Access to spatial data will vary according to the data types and structuring. For the moment, however, we consider this matter conceptually. So we may enquire of data according to:

1. The name of an object.
2. A particular single position on the earth's surface or relative to an arbitrary reference system.
3. A specified block of space in the spatial reference system.

If it is possible to associate unique names with spatial entities, then this form of access may be preferred by many users. But not all 'places' have names, and for some features which do have unique names, there is no

clear spatial limit for the name. All things stored in a database will have a unique identifier of some kind, so that unambiguous access is always possible; but possibly not in a form comfortable to the user.

Also, different forms of representation may not lend themselves so well to straightforward use of names. For example, regular tessellations will need to have a name associated with a set of cells, either by direct storing of the feature name of each cell in the set, or by a numerical code which is translated to a name by a lookup table. If a cell forms part of several objects, then it will need to have several associated numerical codes or names. For a layered vector data model, names are conceptually attached to features like polygons; for example, lakes, not the vertices and line segments needed to demarcate them. So an additional data type of annotation may be necessary if the software does not allow combination of the primitive spatial units to a level to which names are appropriate.

Earlier discussion (section 4.2) of discrete and continuous referencing pointed to different forms of coordinates and tiling systems. Queries as to the nature of space can be executed by identifying positions or tiles and then providing the descriptive information about the identified position. However, some software systems may not be able to handle all points in space too well, recognizing only entities as encoded; that is, they lack the ability to perform nearness, or point-in-polygon or point-on-line tests quickly or at all.

It is important to be sensitive to a double requirement for accessing spatial objects:

1. Identifiers for entities or their encoded forms.
2. Locators for spatial objects or chunks of space.

Recall that locators may be geometric, either Cartesian or azimuthal, or topological (such as route mile point or postal address), or space-filling (for example, a Peano key). And the locators may be of mixed form, for example a postal code area delimited by coordinates for vertices.

7.6.2 Rectangles and strip trees

At times lines, either one-dimensional objects, or the boundaries for polygons, are accessed and retrieved by reference to the boxes that contain them. Enclosing figures, generally rectangles, but possibly circles or spheres as discussed later, serve to identify the range of values in the x and y dimensions, whatever the particular form of representation for the details of the shape of the line. As discussed in section 4.2.2, lines or

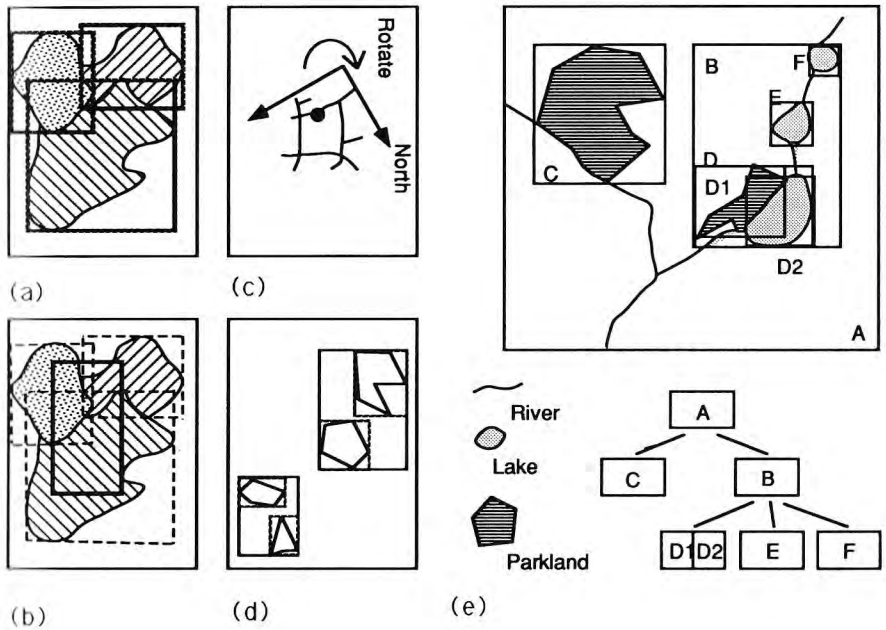


Figure 7.24 Rectangles for polygons. (a) Minimum enclosing rectangles for a tessellation of polygons. (b) Truncation of polygons for search window. (c) Rotation requirement. (d) Rectangle hierarchy for nonconnected polygons. (e) Example of an R-tree.

irregular polygons encoded geometrically can be represented by rectangles drawn in orthogonal dimensions to touch the extreme points of the object in both x and y dimensions. These minimum-bounding rectangles, shown in Figure 7.24a, are useful devices for extracting particular lines or polygons from a set, requiring specification of only the range of x and y, rather than undertaking a spatial search on coordinates for the polygon boundary vertices.

The rectangles (or cuboids), with sides parallel to the two (three) coordinate axes, may also serve to clip entities stored as polygons or lines in the interest of finding what exists within a block demarcated by a particular range of x and y (and z if required). Or the rectangle may represent a larger spatial unit, a map sheet or tile subdivision of the entire database. The fragmentation of entities has limitations, though, such as failure to retrieve an entire object (Figure 7.24b), and the fragmentation can grow to undesirable levels as boxes are made smaller in cases where the amount of data increases. Otherwise a rectangle can be used to access all objects within it, but, again, the chance of retrieving a few objects is

correlated with the size of the rectangles relative to the density of empirical phenomena at a given scale.

If the rectangles are bounding rectangles, then the number of rectangles will be equal to the number of entities, and they no longer serve a purpose of simply partitioning space. In the case of a database of different thematic layers, polygons in each layer may be enclosed by sets of minimum rectangles, leading to overlapping of bounding rectangles when searching for all thematic objects in a specified coordinate range (Figure 7.24b). It is, though, easier to compute intersections of overlapping rectangles than to find where irregular figures might cross.

The enclosing rectangle is just one of several **spatial access devices** using regular figures. Some needs may be better met if the enclosing boxes are not dependent on being parallel to coordinate axes, as, for example, when searching for objects in a rectangle rotated to a certain orientation, or a varying angle in the case of vehicle navigation displays in which the compass direction at the top of the map display varies in order to keep the vehicle icon-oriented in the direction of the vehicle (Figure 7.24c). Circles and spheres are insensitive to the rectangular axes of Cartesian coordinate systems, and are also appropriate to searching in azimuthal coordinate space. On the other hand, it is more involved computationally to fit a circle around an irregular polygon.

This type of spatial unit organization can also have a **tree structure** in order to get different spatial resolutions, but does not involve a regular partitioning of space as with the quadtree (Figure 7.24d). Indeed areas void of polygons or other objects can be ignored. Known as **R–trees** (rectangle trees) and illustrated in Figure 7.24e, this organization is preferable for unconnected polygons rather than tessellations (see also section 15.2.3).

Here, as we illustrate, we can reference the two large rectangles B and C, at one level, but to more clearly access the matching parkland within B, we go through two more levels to get to D. Access may be by identifier, or by a locator for the bottom left corner (x, y based). The intent in building the rectangles is to keep overlap to a minimum while still keeping as many features as possible completely within a single box.

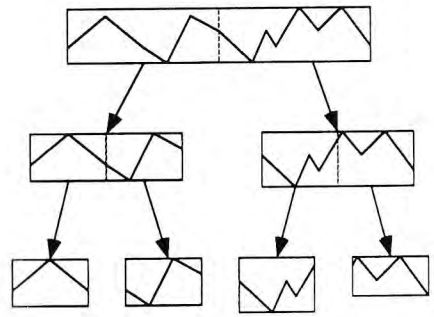
Linear features, as well as polygons, can be represented by rectangular cells. One technique, using the **strip tree**, is beneficial for operations requiring search of linear features. It is appropriate for single curves, rather than sets, being produced by successive approximations enclosed in bounding rectangles (Figure 7.25a). The process of decomposition can be stopped when strips are of a predetermined width or height. If long linear features are not split into pieces, their extents will be large, possibly being inconsistent with zooming operations for visual display changes of scale.

In cases like this the low-level clipping routines will have to be invoked.

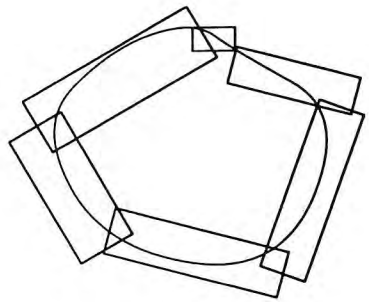
A strip tree is very nice for representation of polylines at different resolution, as might be done in the case of approximations of a coastline at different scales. Several special conditions may have to be taken into consideration, such as closed curves, by breaking an initial rectangle into two or more separate but connected pieces. A disadvantage is that here the rectangles, which can have varying orientations, are not tied into a given coordinate system (Figure 7.25b). The data tabulated will consist of the strip width and level, a code pointing to the next level block in the strip hierarchy, and geometric and other information, as shown (Figure 7.25c).

7.6.3. Sheets and tiles

The topic of division of space also arises in the creation of databases. Because much spatial data now exists in the form of separate pieces of paper, the digital equivalents are often created as discrete units even if they comprise part of a series, as for most national topographical maps, for example, Figure 4.2. While the concept of a seamless coverage for the joined base maps in digital form is laudable, there are practical matters to be dealt with in order to produce effective and accurate spatial databases



(a)



(c)

(b)

Block ID	Width of strip	Level in tree	Pointer to lower level block	Coordinates for strip start	Coordinates for strip end	Coordinates for line feature	Attributes for line feature

Figure 7.25 Trees for linear features. (a) Strip tree. (b) Strips for an enclosed polyline. (c) Representative data for a strip tree.

(Chrisman, 1990). There are two main types of space partition in this context:

1. The physical subdivisions, generally sheets representing original paper or photograph documents that are to be combined to produce a single cover.
2. Tiles produced as logical units for reasons of user querying or possible management purposes.

A sheetless database should be created so as not to have map edge matching problems resulting from mismatched positions of map features, and will avoid the problems of queries returning bad data because a sheet boundary truncated an object. However, notwithstanding the physical space savings in using logical tiles rather than physical irregular chunks as a major unit to partition a coverage, by devices like a reduction in the number of digits needed for storing coordinates by reducing the range covered by the map area, queries cutting across tiles will still occur. However, while logical partitioning is a good idea, given that user needs may not be well anticipated, there is still a need to provide for operations that cross tile boundaries, for example, to assemble the pieces of a road. A possible device is a double encoding for the spatial units (Chrisman, 1990); and another option is to have overlapping tiles.

7.6.4 Different forms of spatial address

The specification of location for spatial databases is itself not necessarily a simple matter. The conventional indication of position by coordinates does not cover all aspects associated with location. For a good understanding of the quality of locator data in spatial data processing, four elements need to be addressed:

Scale
Resolution
Precision
Accuracy

Scale (denoting the order of magnitude or level of generalization at which phenomena exist or are perceived or observed) and resolution (the size of the smallest recording unit, akin to precision, the fineness of measurement) are traditionally specified separately from the traditional Cartesian coordinate form of indicating position. Thus the scale of observation might be specified by a cartographic ratio like 1 : 1,000,000 or by reference to a unit of observation, for example nation. The resolution

may be indicated by the size of shortest length to be measured on the ground, or by a fuzzy tolerance value for a digital map.

While Cartesian coordinates may have a length (number of digits) that varies with machine precision (whether integer or real numbers in a digital computer environment), dictated usually by the hardware's word length, this length does not inherently indicate which digits are significant. The numbers themselves say nothing about scale, and carry no guidance as to the accuracy of the measurements. Something akin to the statistician's sampling error measures must be provided separately, as part of the ancillary metadata. Accuracy, the correctness of measurements, in the sense of validation against reality, is not the same as precision, which reveals the detail in recording the observed properties.

Other types of locator can be constructed to convey something of scale, resolution and accuracy as well as position (Dutton, 1991). We have already presented a locational coding scheme for quadtrees. A quadtree address, such as 321 for the block shown in Figure 7.26a, conveys position for a quadrant in the NW, NE, SW, SE sequence, and indicates resolution by the number of digits. Precision in location can be achieved by making the quadrant sufficiently small to encompass the entire object. Accuracy can be conferred by establishing a buffer strip around the object, that is, making the square larger by some quantity. Similarly, a triangular tessellation (Figure 7.26b) coding scheme denotes position, scale and resolution. In the example, the level is shown by the number of digits, and 0, 1, 2, 3 refer in a consistent way to the four

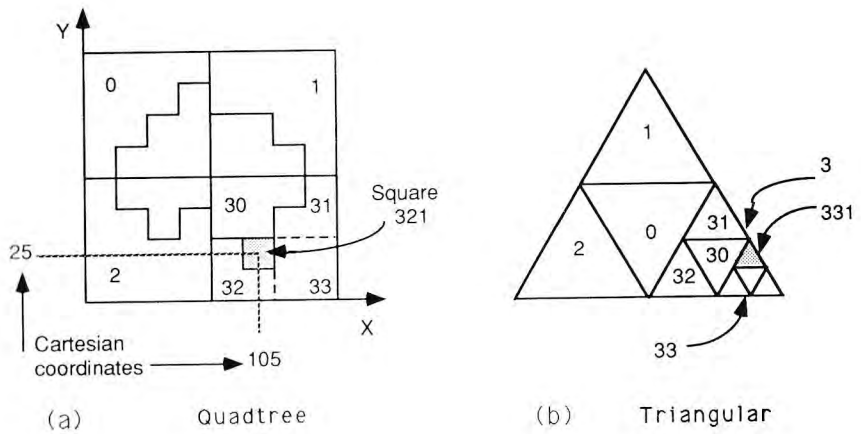


Figure 7.26 Coordinate and tessellation addressing. (a) Quadtree Z-order locational coding. (b) Polyhedral tessellation addresses.

triangles. Using an example provided by Dutton (1991) the Giza Pyramid has an address of 003201320110230. The first zero in this address at the fifteenth level indicates the global octant in the octahedron system discussed in section 6.5.

We shall return to this topic of methods of spatial addressing again later in Chapter 15. For the moment it is important to be sensitive to the multifaceted nature of establishing position of objects in space.

7.7 SUMMARY

The scope of this chapter was not to present all situations, problems, general procedures and algorithms for manipulating spatial objects. We have merely tried to give a flavour, in order to transfer something of an appreciation for data manipulations in spatial information processing. We have accordingly:

1. Emphasized the variety of transformations that may be needed.
2. Presented some methods for access to spatial data elements.
3. Identified the fundamental computational geometry operations.
4. Described basic computations needed for some spatial statistics.
5. Shown the need to both differentiate, and yet deal jointly with, the spatial characteristics of geometry and topology.
6. Established a basis for intensional rules that can be incorporated into the design and implementation of spatial information systems.

The treatment of transformations provides a framework for discussion of alterations of states for conceptual modelling. Later we will treat the topic of multiple representations, but it should be clear already that many different states can exist. The great variety of operations on the spatial data may have challenges for their implementation in software systems because there is not a simple one-to-one match between operation concept and method. Returning to the idea in Chapters 1 and 2 of a spatial information system being a toolbox of operators applied to data, we have in Chapter 7 presented some of the low level operations that may need to be performed in order to undertake spatial analysis or mapping tasks, or reorganize a spatial database. Chapter 8 covers some of the higher semantic order operations utilized in several domains.

We have presented operations involving positional data: scaling, rotation and translation; and distance, length and area measurement. We have discussed more complex needs requiring several basic operations, as in conflation, shape computation, buffering, areal interpolation and

polygon overlay. All of these have been dealt with in the domain of encoding of geometry and topology. This arises, in part, because many fundamental operations for regular tessellation representations are much simpler, but we shall come back to this point in Chapter 8. In addition, we have so far paid only scant attention to the attribute data, although, of course, interpolations and extrapolations apply to one or more spatially variable attributes.

It is important to emphasize that the algorithms and methods discussed will be used in data modelling as rules, to be covered in Chapters 10 and 11. Indeed, spatial data representation must embody not only coordinates and topology, but also some geometric semantics as rules. Whatever the form, process or need, the impact of interpolation and extrapolation on spatial information systems is the encoding of a rule or model for deriving new data from encoded initial data. The determination of centroids, line intersections, shapes and area or volume measures, buffer zones and clipping regions, requires geometric operations in the form of rules. The intensional representations may be unavoidable or they may be desired for reasons of data storage economy; but they must be recognized and provided for in addition to complete, explicit, deterministic, extensional encoding.

7.8 BIBLIOGRAPHY

Either Foley *et al.* or Preparata and Shamos will provide the reader with extensive coverage of computational geometry algorithms. The work of Davis provides extensive coverage of spatial interpolation, while Clarke contains several chapters on cartographic data transformation.

Aronson, Peter. 1987. Attribute handling for geographic information systems. *Proceedings Auto Carto 8 Conference, Baltimore*, Maryland, USA, pp. 346–355.

Ballard, Dana. 1981. Strip trees: a hierarchical representation for curves. *Communications of the Association for Computing Machinery* 24: 310–321.

Boyce, Ronald R. and W. A. V. Clark. 1964. The concept of shape in geography. *Geographical Review* 54: 561–572.

Chrisman, Nicholas R. 1990. Deficiencies of sheets and tiles: Building sheetless databases. *International Journal of Geographical Information Systems* 4(2): 157–168.

Clarke, Keith C. 1990. *Analytical and Computer Cartography*. Englewood Cliffs, New Jersey, USA: Prentice Hall.

Davis, John C. 1973. *Statistics and Data Analysis in Geology*. New York: Wiley.

Deichmann, Vive, Michael F. Goodchild and Luc Anselin. 1989. *A General Framework for the Spatial Interpolation of Socioeconomic Data*. Technical

Report, National Center for Geographic Information and Analysis, University of California, Santa Barbara, California, USA.
Dutton, Geoffrey. 1990. Locational properties of quaternary triangular meshes. *Proceedings of the 4th International Symposium on Spatial Data Handling*, July 23–27. Zurich: Switzerland, pp. 901–910.
Dutton, Geoffrey. 1991. Polyhedral hierarchical tessellations. The shape of GIS to come. *Geo Info Systems* 1(2): 49–55.
Foley, James D. *et al.* 1990. *Computer Graphics, Principles and Practice*, 2nd edn. Reading, Massachussetts, USA: Addison-Wesley.
Frank, Andrew U. 1987. Overlay processing in spatial information system. *Proceedings of the Auto Carto 8 Conference*, *Baltimore*, Maryland, USA, pp. 16–31.
Goodchild, Michael F. and Nina S-n Lam. 1980. Areal interpolation: a variant of the traditional spatial problem. *Geo-processing* 1: 297–312.
Guevara, J. Armando. 1985. Intersection problems in polygon overlay. *Proceedings of the Auto Carto 7 Conference*, *Washington, DC, USA*.
Mark, David M. 1987. Recursive algorithm for determination of proximal (Thiessen) polygons in any metric space. *Geographical Analysis* 19(3): 264–272.
Olea, Ricardo A. 1974. Optimal contour mapping using universal Kriging. *Journal of Geophysical Research* 79(5): 695–702.
Oliver, Margaret, Richard Webster and John Gerrard. 1989. Geostatics in physical geography. Part 1: theory. *Transactions of the Institute of British Geographers* 14(1): 259–269.
Openshaw, Stan. 1984. *The Modifiable Areal Unit Problem: Concepts and Techniques in Modern Geography*. Norwich, UK: Geo Books.
Peuquet, Donna J. 1981. An examination of techniques for reformatting digital cartographic data. Part I: the raster-to-vector process. *Cartographica* 18(1): 34–48.
Peuquet, Donna J. 1981. An examination of techniques for reformatting digital cartographic data. Part II: the vector-to-raster process. *Cartographica* 18(3): 375–394.
Piwowar, Joseph M., Ellsworth F. LeDrew and Douglas J. Dudycha. 1990. Integration of spatial data in vector and raster formats in a geographic information system environment. *International Journal of Geographical Information Systems* 4(4): 429–444.
Poelstra, T. J. 1990. Frankie goes to town, or how to capture reality in urban areas. In: Polydorides, Nicos D. (ed.) *Data Aquisition for Spatial Information Systems*; Athens, Greece: Urban and Regional Spatial Analysis Network for Education and Training, Computers in Planning Series 7: 56–66.
Preparata, Franco P. and Michael I. Shamos. 1986. *Computational Geometry: An Introduction*. New York: Springer-Verlag.
Saalfeld, Alan. 1988. Conflation: Automated map compilation. *International Journal of Geographical Information Systems* 2(3): 217–228.

Taylor, Peter J. 1977. *Quantitative Methods in Geography*. London, UK: Houghton Mifflin.
Tobler, Waldo R. 1979. Smooth pychophylactic interpolation for geographical regions. *Journal of the American Statistical Association* 74: 519–530.
Unwin, David. 1981. *Introductory Spatial Analysis*. London, UK: Methuen.