The Structures for Spatial Data

The large memory requirements associated with storing pictorial data are well known. For example, storing ***an ordinary frame of television*** requires at least 512x5l2 bytes, if we use three bits for two of the primary colors and two for the third.

A black and white passport photograph requires at least a 64x64 matrix with six bits per element, well above the size of a record containing whatever other information is in a passport. (A page of single-spaced typewritten text requires about 3000 bytes).

Problems of storage, search, retrieval, transmission, etc. are particularly difficult whenever pictorial data are encountered. These difficulties are somewhat counterintuitive because humans often find it easier to deal with pictures than with text.

The use of pattern recognition may result in higher compaction, but such processing requires a fair amount of computing so that one is still obliged to face the problem of storing and representing images on a computer.

The problem of data compaction has different forms in communications and in computing. In many applications, an image is created, transmitted, observed, and then discarded:

- the major consideration in communications is reducing the bandwidth required for transmission under the constraint that the processing must be done in time comparable to that required for generating and transmitting the image;
- a different situation exists in applications where a pictorial data base is needed and images must be stored for long periods. New images must be compared with old ones, or sets of images must be searched for certain features.

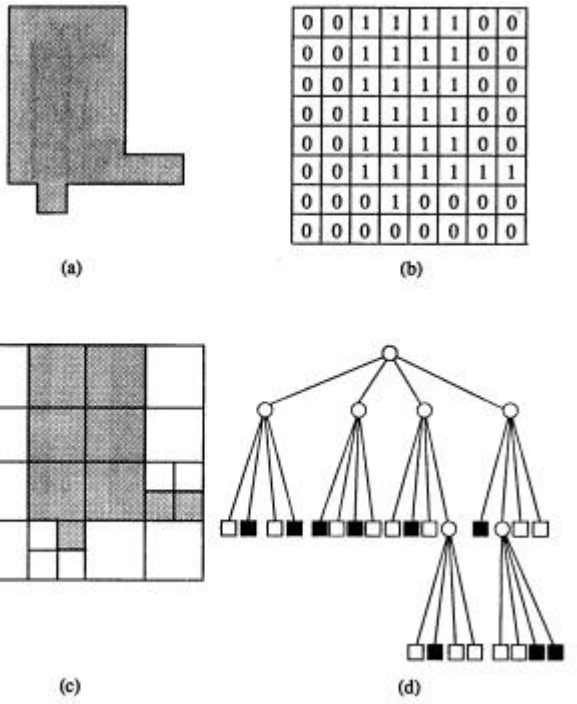
Pyramids or Quadtrees

Pyramids or quadtrees are popular data structure in both graphics and image processing, and in other spatial data. This technique is best to present and display when the picture is a square matrix *A* whose dimension is a power of ***2***, say $\boldsymbol{2^n}$.

The matrix A can be subdivided into four square matrices A_0, A_1, A_2, A_3, whose dimensions are half of A. This process can be repeated recursively ***n*** times, until the single pixel level is reached. The levels can be numbered, starting with zero for the whole picture, down to ***n*** for the single pixels.

A particular square may be labeled with one of the symbols 0, 1, 2, or 3, concatenated to the label of its predecessor square. In this way single pixels will have labels that are ***n*** characters long. We can express this arrangement as a tree, whose nodes correspond to the squares. Nodes are connected if one of the corresponding squares immediately

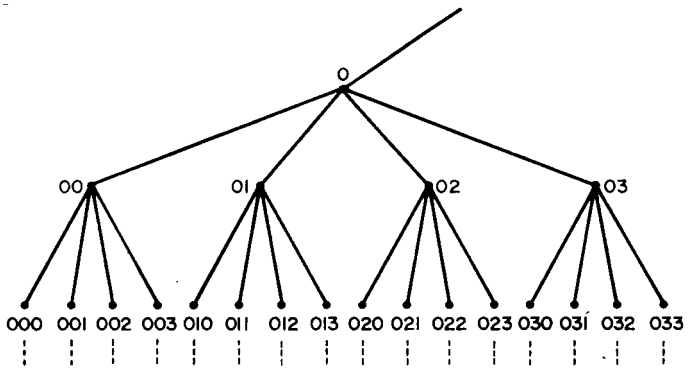
contains the other. The root of the tree corresponds to the whole picture, the leaves to single pixels, and all other nodes have down degree 4.



Since the k^{th}-level contains 4k squares, the tree has a total of

$$N = \sum_{k=0}^{n} 4^k = \frac{4^{n+1} - 1}{3} \approx \frac{4}{3} 4^n$$

nodes. Therefore, there are approximately 33 % more nodes than pixels. It turns out that for many applications the tree can be stored using only 4^n locations. Figure shows the addressing notation for an ***8 x 8*** picture:



000	001	010	011	100	101	110	111
002	003	012	013	102	103	112	113
020	021	030	031	120	121	130	131
022	023	032	033	122	123	132	133
200	201	210	211	300	301	310	311
202	203	212	213	302	303	312	313
220	221	230	231	320	321	330	331
222	223	232	233	322	323	332	333

Creating a Quadtree

We start with an image that can be accessed row by row (the case when it is stored on a tape or disk, or when it is digitized or one based on a television camera). We read in two such rows, and then examine quadruples of pixels in the order shown in figure (a):

0	1	4	5	10	11	14	15	20	21	...
2	3	6	7	12	13	16	17	22	23	...
100	101	104	105	110	111	114	115	120	121	...
102	103	106	107	112	113	116	117	122	123	...

(a)

g_0	g_4	g_{10}	g_{14}	g_{20}	g_{24}	...

$(g_1, g_2, g_3)(g_5, g_6, g_7) \cdots (g_{75}, g_{76}, g_{77})$

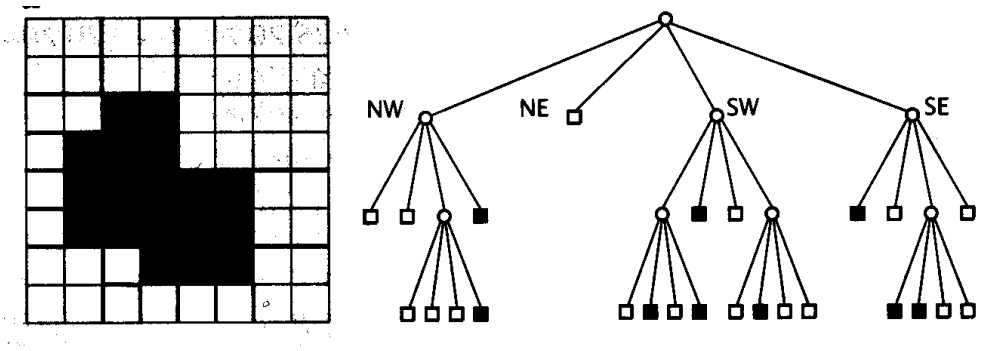
(b)

g_0	g_4	g_{10}	g_{14}	g_{20}	g_{24}	g_{30}	...
g_{100}	g_{104}	g_{110}	g_{114}	g_{120}	g_{124}	g_{130}	...

(c)

The part (a) of figure above presents so-called the ***Morton order*** of visiting the elements of matrix. Labels are in octal assuming a 64_{10} x 64_{10} image.

Infix ordering of trees gives an efficient way to visit (or asemble) all the nodes of quadtree:



$$ggwwgwwwbbwggwbwbbwgwbwwgbwgbbwww.$$

or, by using parentheses (which are not necessary at all):

$$g(g(wwg(wwwb)b)wg(g(wbwb)bwg(wbww))g(bwg(bbww)w)).$$

Reconstructing an Image from a Quadtree

If an image is stored as a matrix its transmission and display proceed along rows or scan lines. Each part is displayed in full detail and if T_c denotes the time required for the complete display, in $T_{c/2}$ we can see only the top half of the image. This is an unsatisfactory way for browsing through pictures, especially if T_c is more than one or two seconds long.

It would be better to display in $T_{c/2}$ the whole image at a coarse resolution so that the viewer can decide quickly whether there is anything interesting in it, and if there is not to abort the display. Such a "gross-information-first" display can be implemented with quad trees.

Image Compaction with a Quadtree

The representation described in the previous two sections can be improved in a number of ways. Our first concern is to eliminate the need for increased storage (or the possibility of round-off errors) for the g_j's. What we need is a technique for a one-to-one mapping of four numbers onto another four.

Proposition: Let a_1, a_2, ... a_n, be n numbers, each consisting of Q bits. The number n is assumed to be a power of two. Then their average plus the $n-1$ differences of D_2, … D_n from that average can be stored in $nQ+\log_2 n$ bits.

Let L be the sum of the n numbers, q be its quotient by n and r its remainder. This proposition shows that for the quad tree we must add two bits at each level, and this can certainly pile up. However, there is a trade- off between the bits for q and r and $a_i - q$. If the numbers a_i are nearly equal to each other, then r and $a_i - q$ will require very few bits, but q may require up to Q. If there are big differences between the a_i's, then q will be small and will require fewer bits.

This feature can be exploited to produce representations that do not require additional space. The result of proposition can be used for increasing efficiency. Instead of transmitting the differences from the average, one can send only the differences from q (each requiring at most Q bits), plus the last q. In addition, we can transmit the remainders r, each requiring only $\log_2 n$ bits (two bits for quad trees).

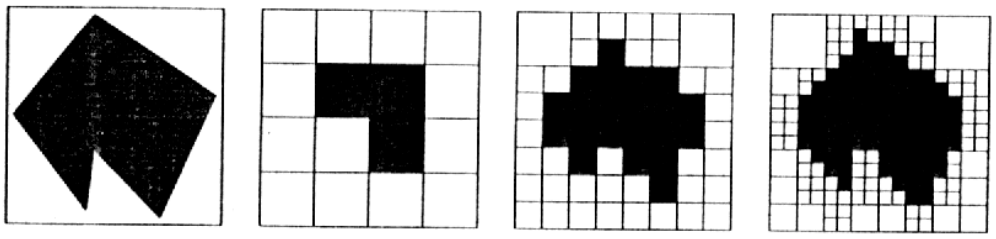
The number of remainders is the increase in the volume not in 33 %, but $(2/Q)$ times 33 %. The reconstruction can be performed by a simple modification of the given equations.

It is possible to achieve considerable savings if the image has large uniform areas. Indeed, suppose that the three differences for a block are zero. Then, instead of placing three zeros in the array, one can insert a single special symbol. (This could be the maximum value of brightness possible, since this value will not occur as a difference from the average). After the scanning of the whole image has been completed, repetitions of this symbol can be erased from all subsequent levels of the tree.

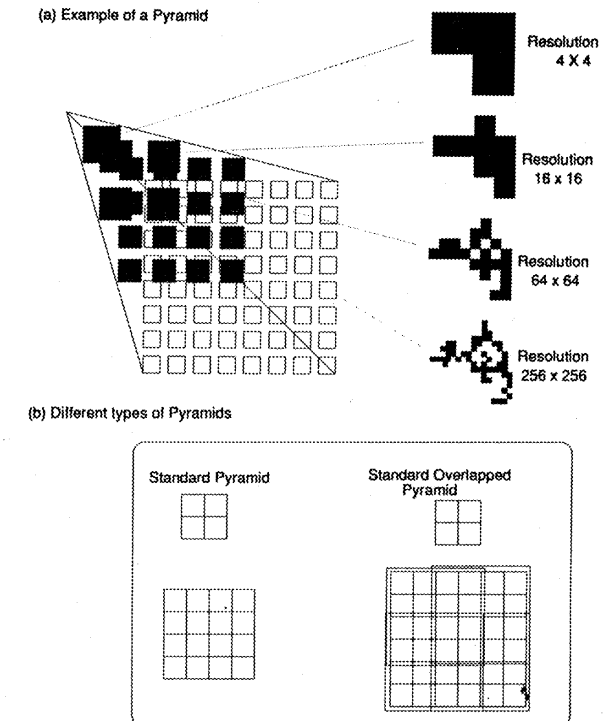
Quadtrees for Variable Spatial Resolution

The concept of variable spatial resolution implies varying sized units at a given resolution level. The choise of the shape is a special matter. The square is particular handy if the process of creating the blocks of varying size is one of the decomposing

space from a general level to more detail. For example, a polygon can be successively approximated by sets of blocks at different levels:



In the cartography, geographic information systems, and in similar applications, we are often associated with a map layer concept. Then the matrix and clustered forms are designed to work with thematic layers, meaning that the attribute data are not recorded in sequence for each and every cell. While the layering architecture is utilized for both entity-oriented and tessellation encoding of phenomena, the two types certainly do treat attribute data differently. For entity-oriented representations, attributes are separated from the spatial information in most cases, whereas for regular tessellations the positional and attribute data are associated.



Spatial **aggregation** can be achieved quite well for tessellations by combinations of adjacent cells; spatial disaggregation can be dealt with by splitting cells. Aggregations might be necessary to facilitate generalization from one resolution (geographic scale) to another. For example, the number of points in four cells can be added to get the incidence for a set of four cells, or, if the attributes are scalar, an average could be obtained. Combinations might be necessary to produce the grid cell equivalent of homogeneous regions or administrative districts.

The square cell is used most often for aggregating or disaggregating across scales. The resulting form, known as the pyramid model (above) provides a multiple scale representation, with spatial units constant for a given scale. Used often in image processing, it provides a means to remove or hide detail in order to focus on structural components like general shape, or to reveal details of form that might be hidden by too much generalization.

It is designed for rapid detection of global (overall, not earthly) features in a complex image. An ordinary pyramid has blocks of four cells combining to larger cells at higher levels, without overlap. A standard overlapped pyramid has 50 per cent overlap for adjacent blocks (b). A hierarchical representation has uses in image processing:

- for browsing at different scales,
- for simplifying mapping,
- for matching data collected at different resolutions,
- for access at different levels.

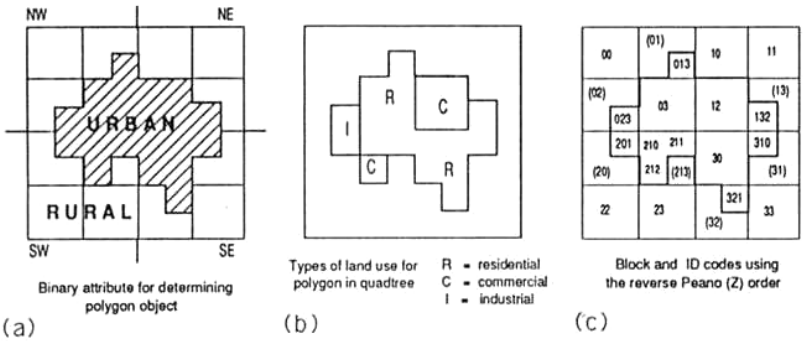
In this way, the grid cell becomes a handy spatial indexing tool, rather than a storage unit, for space filling curves.

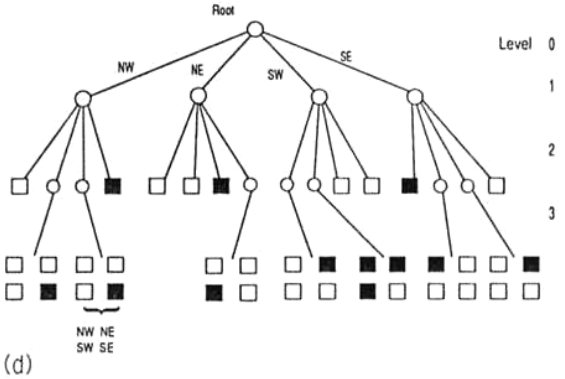
The quadtree data organization, as presented above, leads to complete table of data for the entire area, involving quadtree blocks and identifiers based on location codes, hierarchical organization, table of attributes for the urbal polygon, complete data. Different kinds of data can be treated in this hierarchical subdivision fashion. Ideally we would like to be able to:

1. treat point, line, and area data in the same way;
2. capture metrical details for entities;
3. facilitate various kinds of operations;
4. deal with different ways of measuring attributes;
5. have consistent locational referencing.

The simplest form of quadtree recognizes the presence or absence of an attribute in space, whether point, line, or area. This usually is refered to the **binary incidence** representation as colour coding, using black and white to indicate presence and absence.

A cell could contain a scalar value, or a pointer to sets of attributes under the condition that the cell is a lowest geographic unit. Thus cells may be used to represent point data, such as cities, where each cell contains one city; or linear features, say water pipes, where each cell contains a segment of a pipe or a junction of several water lines.





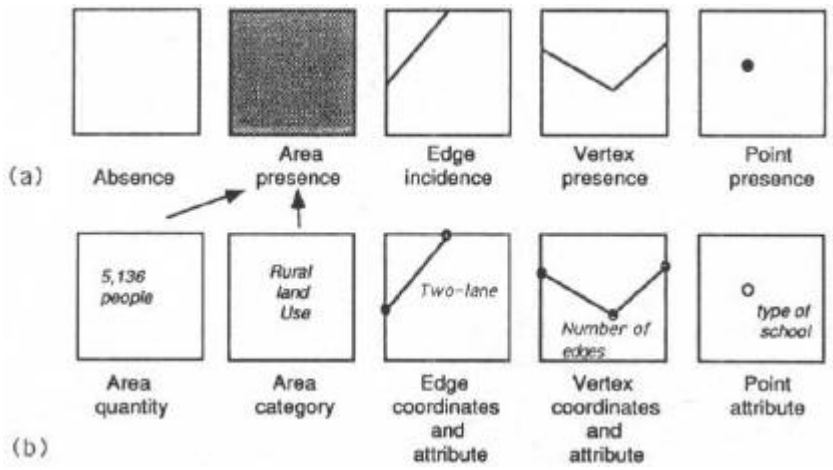
Quadtree Level 1	2	3	Land use type
		013	R
		023	I
	03		R
	12		C
		132	R
		201	I
		210	R
		211	R
		212	C
	30		R
		310	R
		321	R

(e)

(f)

Level 2	Level 3	Type of land use	Area (number of level 3 blocks)	Level 2	Level 3	Type of land use	Area (number of level 3 blocks)
00		Rural	4		202	Rural	1
01	010	Rural	1		203	Rural	1
	011	Rural	1	21	210	Residential	1
	012	Rural	1		211	Residential	1
	013	Residential	1		212	Commercial	1
02	020	Rural	1		213	Rural	1
	021	Rural	1	22		Rural	4
	022	Rural	1	23		Rural	4
	023	Industrial	1	30		Residential	4
03		Residential	4	31	310	Residential	1
10		Rural	4		311	Rural	1
11		Rural	4		312	Rural	1
12		Commercial	4		313	Rural	1
13	130	Rural	1	32	320	Rural	1
	131	Rural	1		321	Residential	1
	132	Residential	1		322	Rural	1
	133	Rural	1		323	Rural	1
20	200	Rural	1	33		Rural	4
	201	Industrial	1				

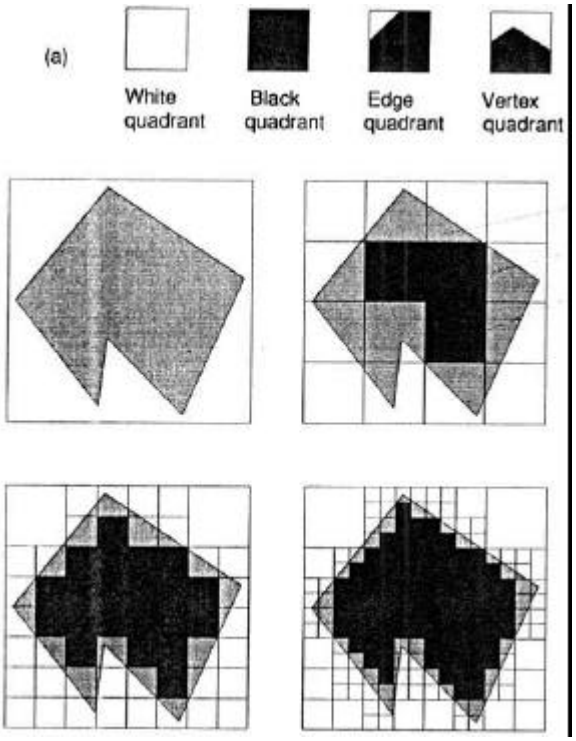
So we may define an attribute presence quadrant, an absence quadrant, an edge quadrant, a vertex quadrant and a point quadrant:



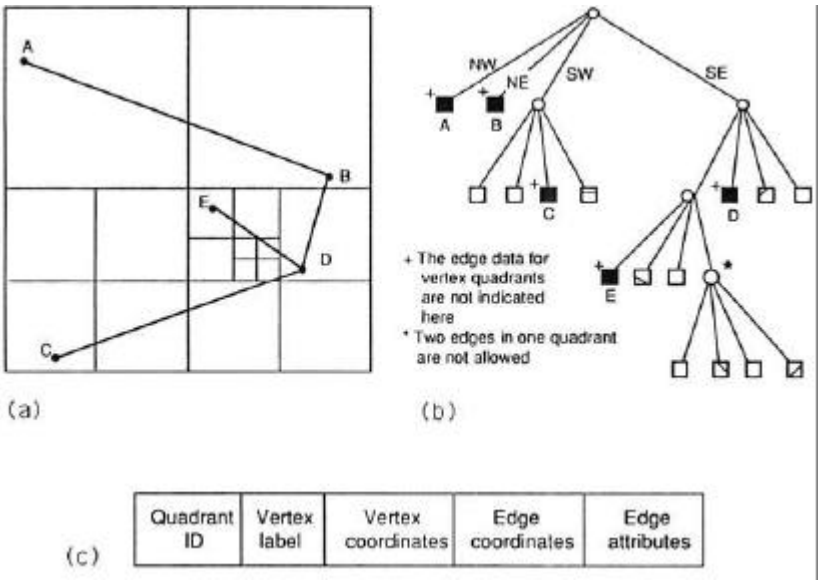
In the standard form, the geometry of edges and points is not retained, only incidence. However, as for fixed resolution regular tessellations, additional information can be encoded for cells (b). In the case of edges representing polygon

boundaries or graphs, this could consist of the *x* and *y* Cartesian or the polar coordinates to establish where the edges cross the boundary of a cell, or for vertices or points, the exact coordinates for a point within a cell.

Extended quadtrees are presented below. They have different type of quadrants, and these types can be used at different levels:



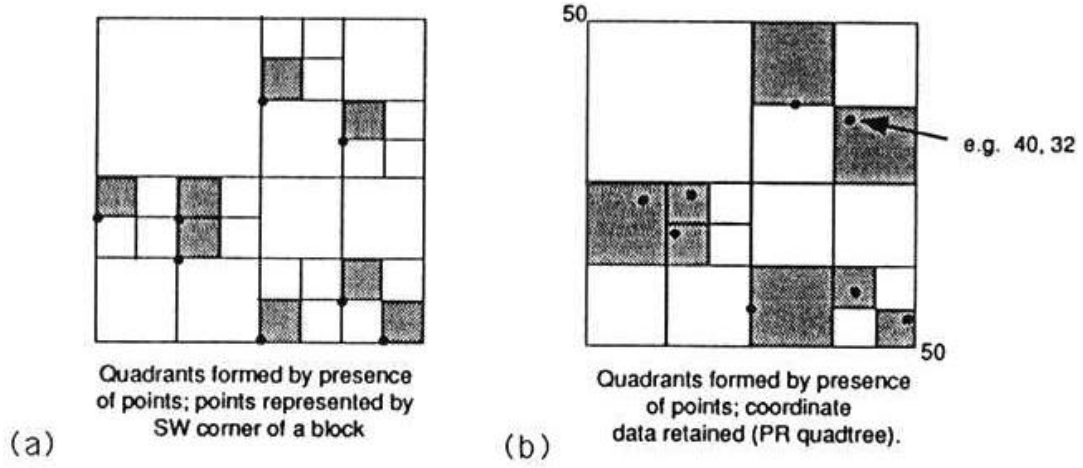
If, for linear features, the incidence is only a vertex with one or more graph edges or a piece of an edge, then a quadtree representation will look like that in:



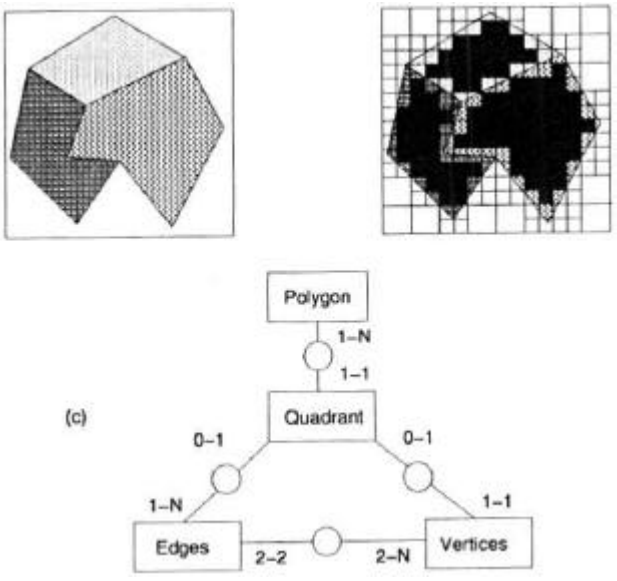
This figure represents a quadtree for line data, giving map and quadrants for line objects, tree and encoding rules, data item records. Various possibilities for encoding

exist; rules must be established before the quadtree database is created from the original data.

Unconnected points may be handled in different ways. A regular figure decomposition process could produce squares from the orthogonal coordinate space by subdividing using both *x* and *y*, with varieties depending on whether or not all four squares at a given level of decomposition were recognized (the MX quadtree) or not (the PR quadtree), The second type requires coordinate information to establish position within the block; the former does **not,** representing the point at a corner of the cell.



An example of a combined representation, based partly on segments and partly on quadrants, is given below (this is presented as a conceptual model):



Different forms of spatial address

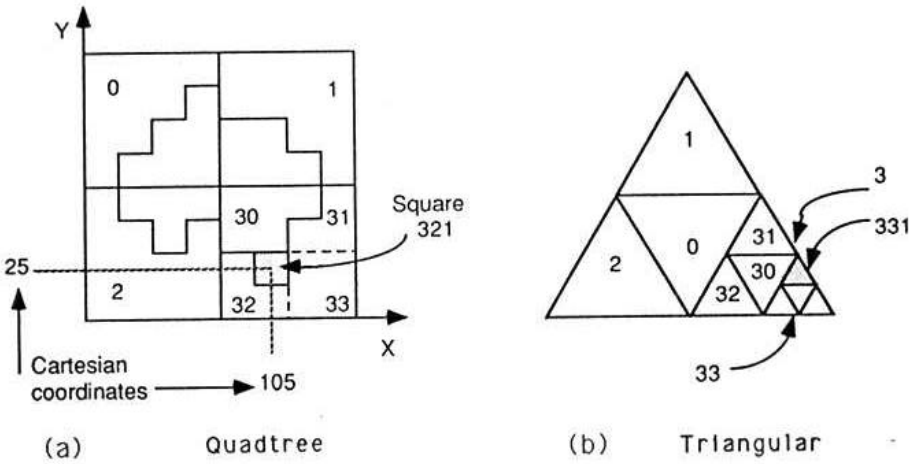
The specification of location for spatial databases is itself not necessarily a simple matter. The conventional indication of position by coordinates does not cover all aspects associated with location. For a good understanding of the quality of locator data in spatial data processing, four elements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

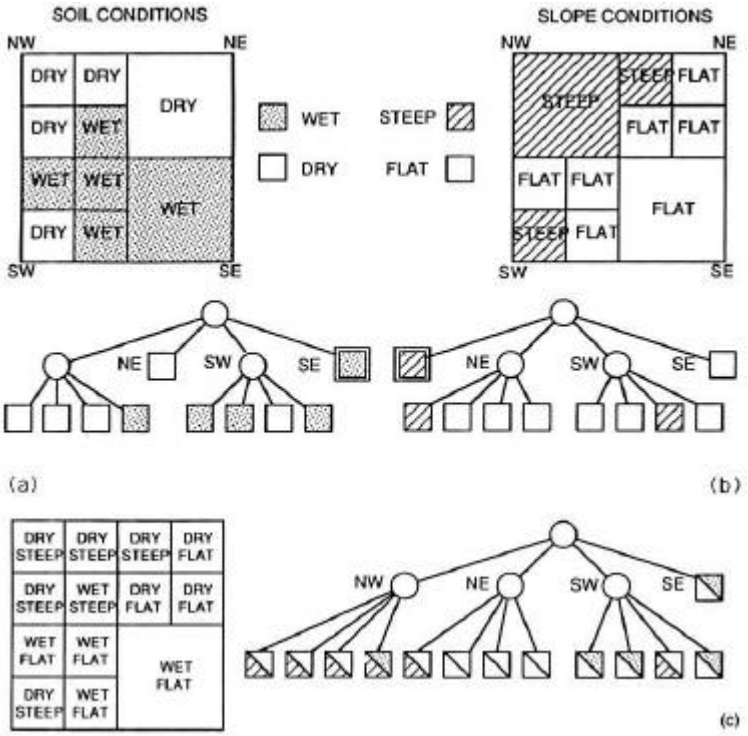
Other types of locator can be constructed to convey something of scale, resolution and accuracy as well as position. A quadtree address, such as 321 for the block, conveys position for a quadrant in the NW, NE, SW, SE sequence, and indicates resolution by the number of digits. Precision in location can 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 coding scheme denotes position, scale and resolution.



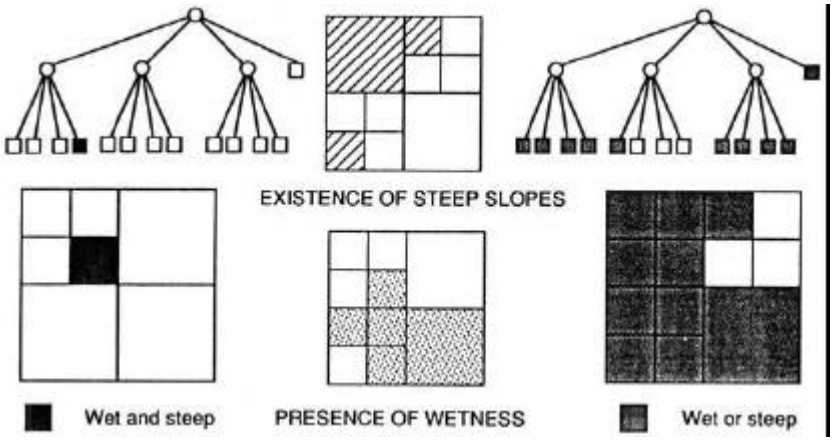
Operations for quadtree tessellations

The quadtree and pyramid varieties of regular tessellations can facilitate operations associated with grid-cell data sets, and may accomplish some actions faster. Separate quadtrees are made from the cell encoded data, and then traversal of each tree, from the top level down, takes values for comparable spatial units and creates a third map. If there is no branch at a givel level, then the value found at that level for the first attribute is assigned to each of four squares at the same level for the orther attribute.

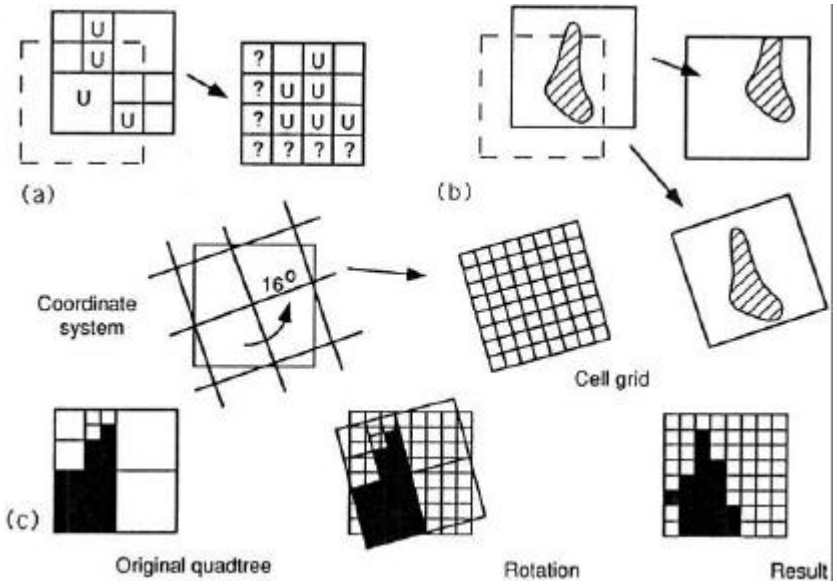
Combination of two maps via quadtrees is presented below. There are soil conditions map and tree, as well as slope conditions map and tree. then it's easy to get combined map and tree:



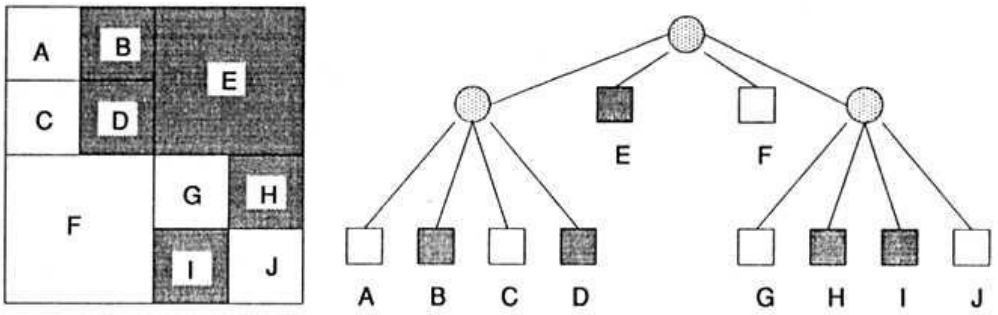
The quadtree is especially useful for **performing set operations** like union and intersection, again doing this by traversing simultaneously two thematic trees, making tests for the attribute coding of the nodes of the hierarchical structure:



Other operations like rotation and translation are not so much convenient for the quadtree structure:

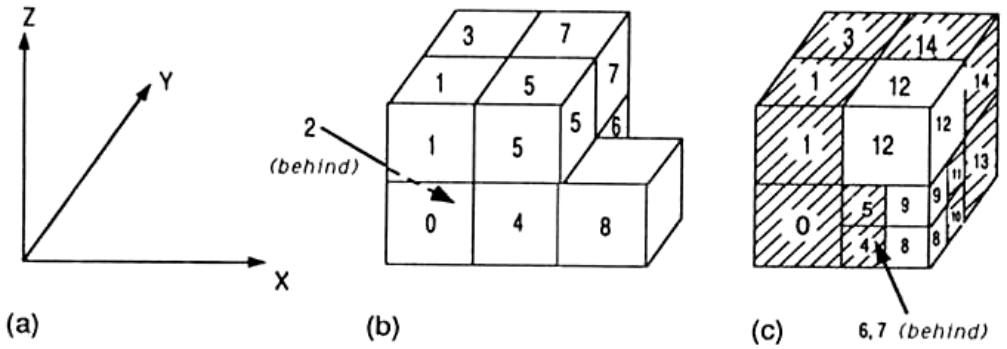


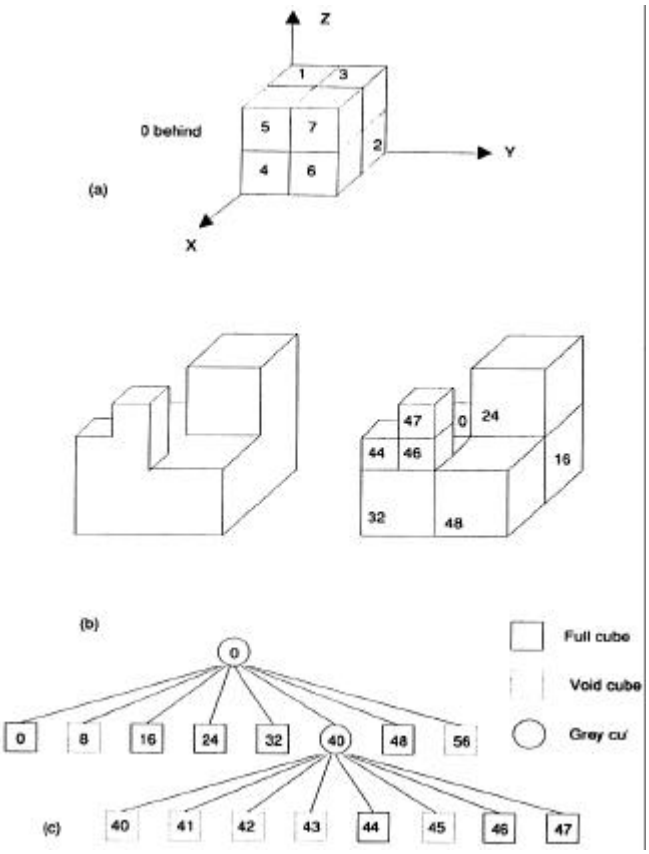
Storing quadtree as a tree:



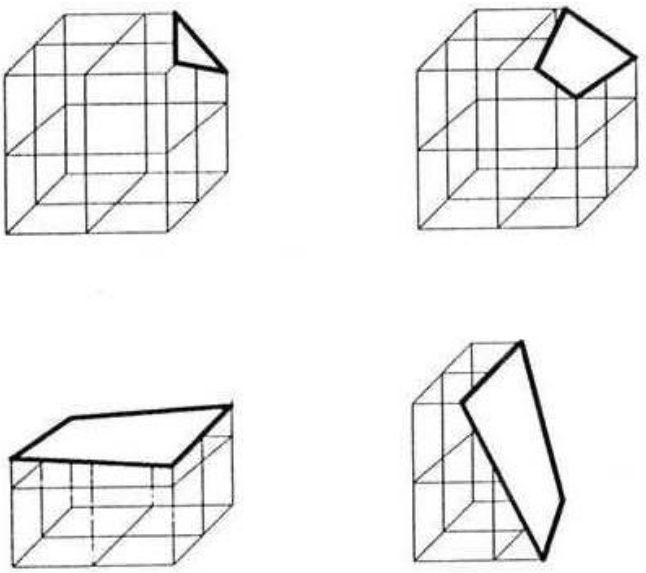
The 3-dimensional structure: octtree

If the process involves systematic splitting of space in 2-dimensional space by a rule of four, then the structure is known as a quadtree, a type of hierarchical data model. The general properties and principles for quadtrees are applicable to the 3-dimensional variant, the octtree, used to some extent for geologic modelling and representing three-dimensional solids. A 3-dimensional equivalent is known as an octtree because it involves an eightfold splitting:





As in case of quadtree, various attributes to dividing cubes can be prescribed:

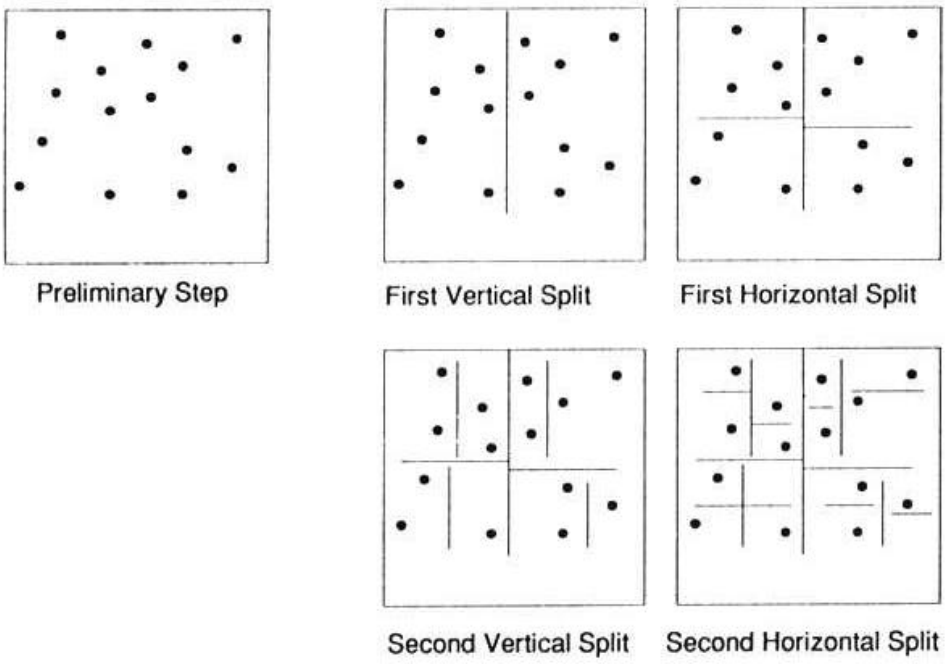


KD-trees

Hierarchical decompositions of spatial data may be undertaken on the basis of the **empirical information** to be encoded and stored in contrast to the regular subdivisions. The later are data independent; the former are data dependent. For example, a distribution of point features, such as cities, may be subdivided into rectangular, rather than square blocks, on the basis of alternating x *and* y axes. A similar process can produce two or four branches at each step. Thus, the empirical

information, the exact position of the points, governs the data structuring, not a fixed-grid scheme. The binary subdivision, which is one of a group of **K-dimensional (KD) trees,** is generally regarded as superior to the point data quadtree for operations done in sequence.

The quadtree and related structures, clearly based on a tessellated discretization of space, provide semantic value by their recognition of varying density of incidence of phenomena in space, and can deal with both vector and raster data. The hierarchical structuring cleverly addresses spatial variations at different scales, it offers the valuable adaptability property to empirical conditions and with good locational referencing provides a basis for efficient spatial access and indexing. As discussed later, Boolean operations such as union, intersection and difference are easy to perform, whereas translation, rotation and scaling are not.

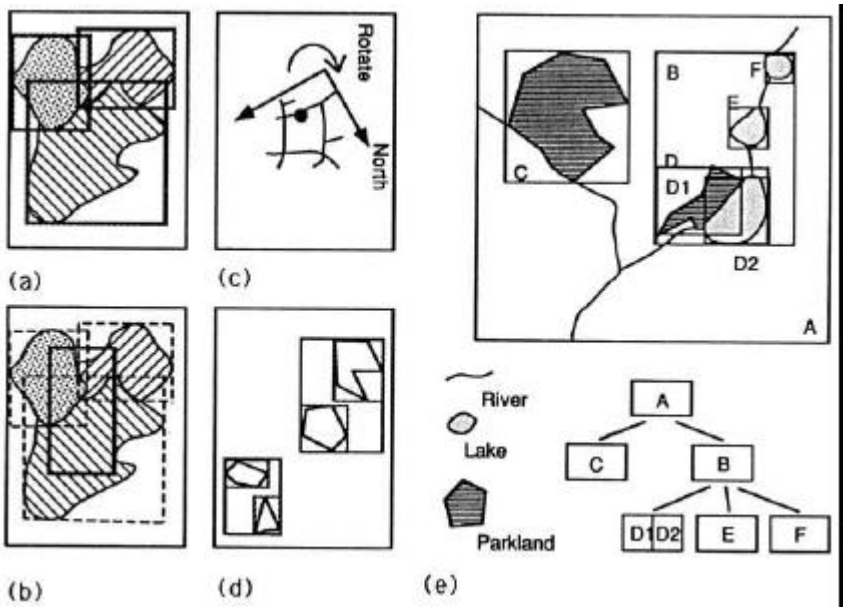


In general, the hierarchical tessellations are regarded as offering benefits in the reduction in the amount of space needed to store data for phenomena. We contrast the more extensive grid cell encoding with the quadtree, and another device, the run-length encoding. The first of these records data for each cell, demonstrated here for two different resolutions. The quadtree will use a smaller number of spatial units as produced by the hierarchical subdivision; the run-length encoding reduces data storage by recording runs of like conditions for rows (or columns) as shown. The degree to which **the space-saving methods reduce storage** depends **primarily on the amount of homogeneity** in the mapped data. The extremes are a perfectly uniform landscape, for which the quadtree block is best, or a checkered pattern in which each cell is different from all its neighbours. In this case, there is no particular advantage in using the two space-saving techniques. Alternative data storage schemes like linked lists are preferable for sparse matrices.

The hierarchical structures may be differentiated on the basis of types of data, the principles guiding or governing the decomposition process, and the type of spatial resolution. However, because they are based on regular spatial units, they also have advantages and limitations associated with the use of grid squares. Particularly, there are limitations in dealing precisely with point and linear features, and in not explicitly addressing topological spatial properties.

Rectangles and similar trees

At times lines, either 1-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, serve to identify the range of values in the *x* and *y* dimensions. Lines or 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 (figure a), 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.

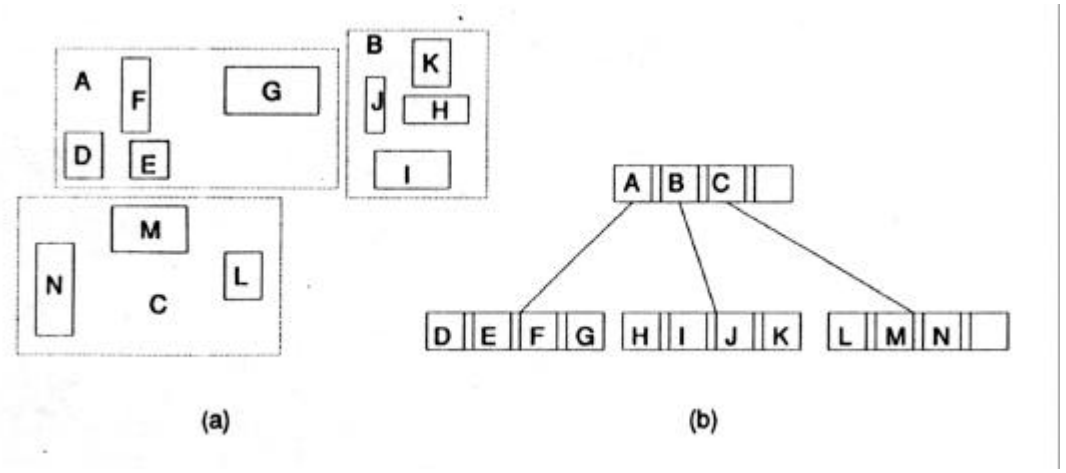


The rectangles (or cuboids), with sides parallel to the two (three) coordinate axes, may also serve to clip entities stored as polygons or lines in the interest of finding what exists within a block demarcated by a particular range of *x* and *y* (and *z* if required). Or the rectangle may represent a larger spatial unit, a map sheet or tile subdivision of the entire database.

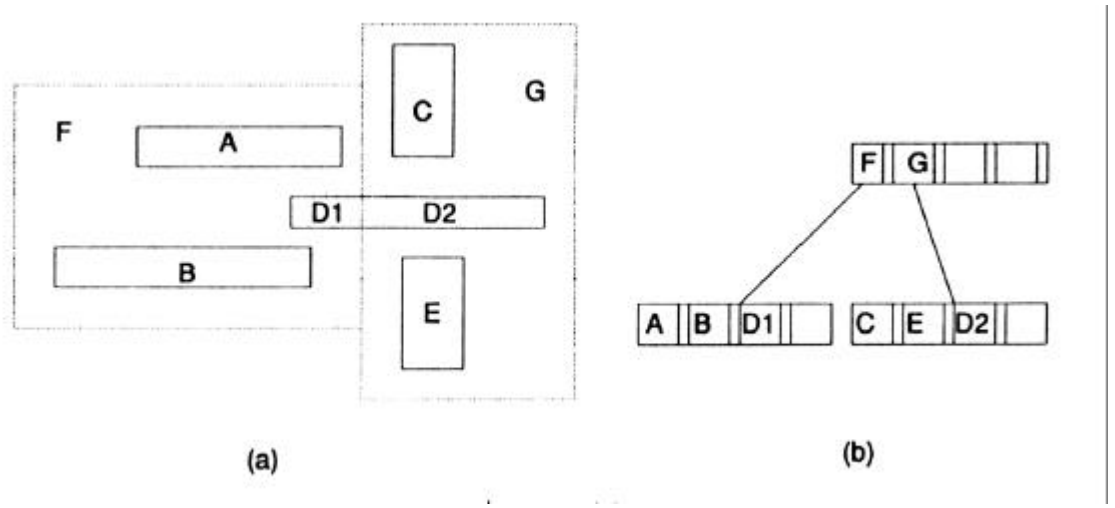
The fragmentation of entities has limitations, though, such as failure to retrieve an entire object (b), and the fragmentation can grow to undesirable levels as boxes are made smaller in cases where the amount of data increases. Otherwise a rectangle can be used to access all objects within it, but, 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 (b). It is, though, easier to compute intersections of overlapping rectangles than to find where irregular figures might cross.

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 (d). Indeed areas void of polygons or other objects can be ignored. Known as R-trees (rectangle trees) and illustrated in figure (e), this organization is preferable for unconnected polygons rather than tessellations, organized either in R-trees or in R^+-trees:



R-tree (non-overlapping rectangles and hierarchical structure)

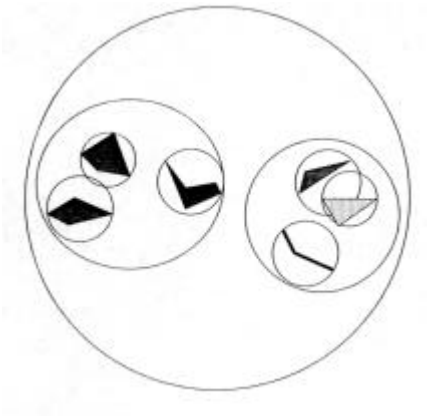


R+-tree (rectangle split by higher level rectangle and hierarchical structure)

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.

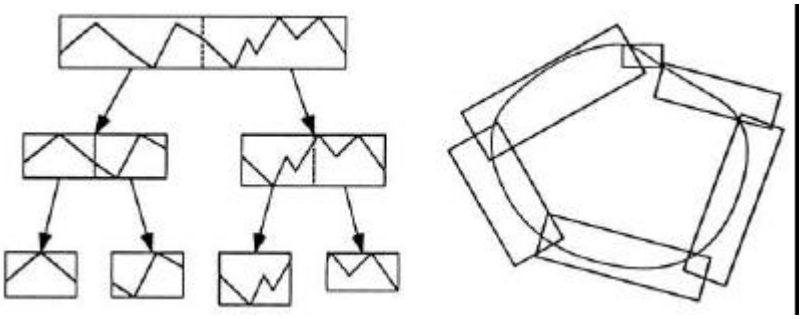
The enclosing rectangle is just one of several spatial access devices using regular figures. Some needs may be better met if the enclosing boxes are not dependent on being parallel to coordinate axes, as, for example, when searching for objects in a rectangle rotated to a certain orientation, or a varying angle in the case of vehicle navigation displays. 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:



Indexing with sphere trees

Strip trees

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 below). 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. Then 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. A disadvantage is that here the rectangles, which can have varying orientations, are not tied into a given coordinate system.

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 sheetiess 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; and another option is to have overlapping tiles.

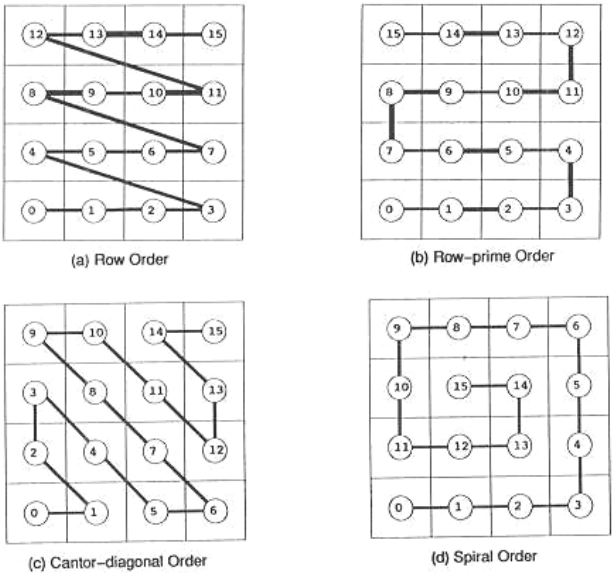
Space-Filling Curves And Dimensionality

Data processing and storage may be more economical if less information can be used to meet the same requirements. Thus area units may be represented by a centroid, a zero-dimensional object or by parametric curves. A data reduction can also occur if objects could be positioned in only one dimension rather than two or three. The matter of dimensionality is encountered in spatial information in different ways. It may arise in terms of addressing systems.

Paths through space

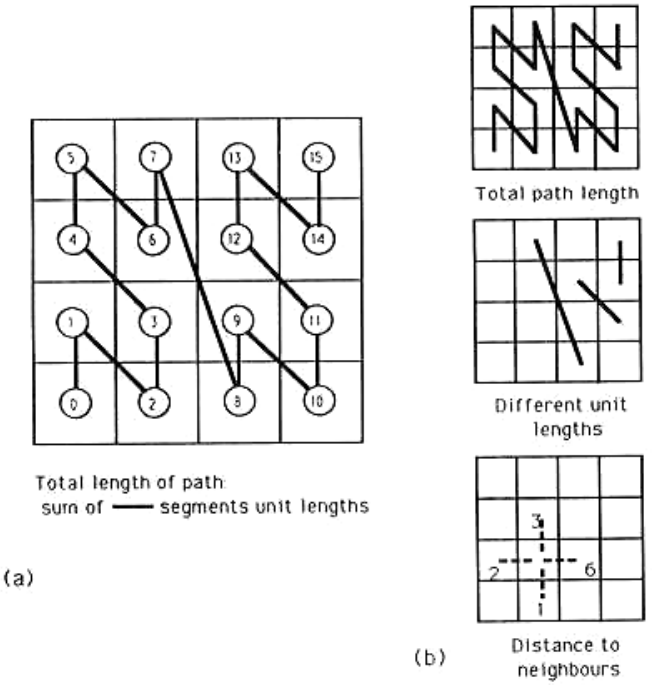
There are different orderings, that is **one-dimensional paths,** through the two-dimensional tiled space. Paths could zigzag, could go along a row in one direction and in a reverse direction in the next row, like a bidirectional computer printer (b), or could follow a path that reduces the total distance of travel through going to as many immediate neighbours as possible, and having a small number of longer connections, or could have diagonal or spiral forms (c and d).

A good sequential ordering should have certain properties that provide some conveniences in single dimension addressing for two- or three-dimensional sets of regularly shaped tiles. The path should pass only once to each tile in the two- or N-dimensional space, and neighbours in space should be adjacent on the path. The path should be useable even if there is a mixture of different sized spatial units, and should work equally well in two or three dimensions, and for connecting to adjacent blocks of space. In reality there is no ideal path; there are just orderings with some of these properties.



A comparison of different paths (resolution given) can use several measures:

1. total length of the path.
2. variability in unit lengths, where unit length is the distance from one point on the path to the next in sequence.
3. the average distance on the path from tiles to their four neighbours in space.



Comparative averages for the central block of four squares are shown in the table below along with other properties of a sixteen-tile mosaic.

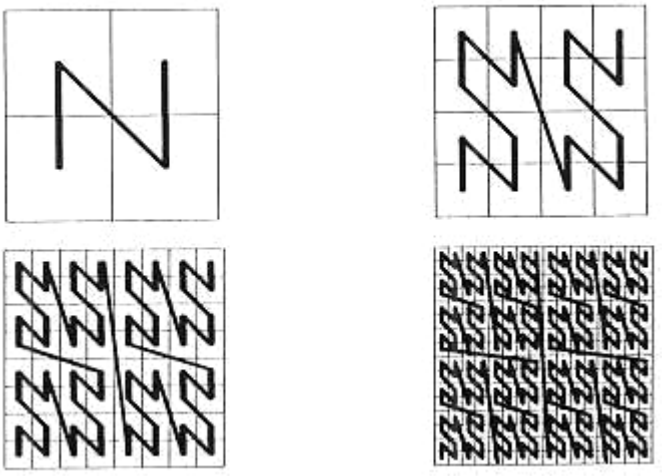
Path type	Length (approximate)	Variability	Average distance
Row	22	2	10
Row-prime	*15*	1	10
Diagonal	18	2	18
Spiral	15	1	13
N	20	3	12

Space-filling curves

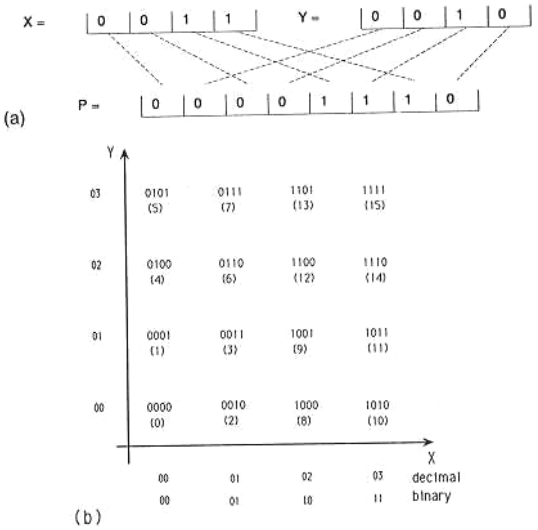
Often we talk about space-filling curves rather than paths through space. These curves are special fractal curves which have characteristics of completely covering an area or volume. While they have a topological dimension of two, their fractal dimension is two when filling an area, or three when completely occupying a volume space. Consequently, thinking of paths in space now as space-filling curves, lines that pass to all possible points in space, they should have the following properties:

1. The curve must pass only once to every point in the multi-dimensional space.
2. Two points that are neighbours in space must be neighbours on the curve.
3. Two points that are neighbours on the curve must be neighbours in space.
4. It should be easy to retrieve the neighbours of any point.
5. The curve corresponds to a bijective mapping from a multi- to a onedimensional space.
6. The curve should be able to be used for variable spatial resolution, that is, a mixture of different-sized "points".
7. The curve should be stable, even when the space becomes very large or infinite.

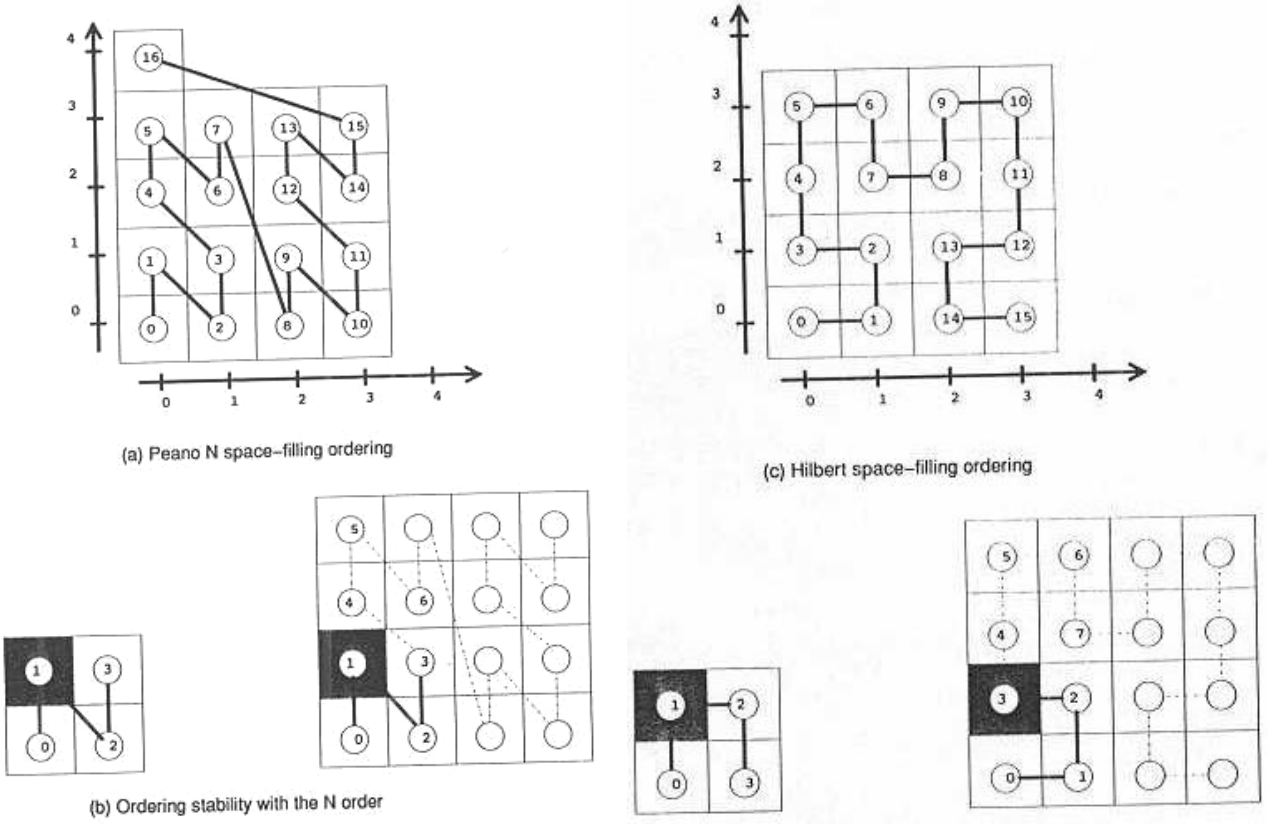
In reality, we do not possess such ideal curves, but there are some with valuable properties for our purposes. The original space-filling curve was exhibited in 1890 by the Italian mathematician Giuseppe Peano (Peano, 1890). A later variety, now known as the **Peano** or **N ordering** (a), facilitates retrieving **neighbours,** and although neighbouring points in space are not always neighbours on the curve, they generally are. It is also possible to deal with different resolutions as shown, and the curve is stable:



The **Hilbert** curve meets most of the conditions noted above, but does not provide an easy way to retrieve neighbours and is not stable. For the Peano curve the keys are easily obtained, the binary digits for the x and y values are interleaved:



Generally, the ordered paths have similar shapes at different scale levels, they are self-similar. However, the particular place of a point or tile in the sequence for a particular curve type may not be consistent across scales. While the N curve does have such **stability,** as revealed by the coded numbers, this is not so for the Hilbert curve. That is, for the Peano case, if the space is extended by doubling each side for the block of four quadrants, we see that the order of squares is not perturbed.



Space-filling curves have two principal, practical uses in the domain of spatial information systems:

- firstly, they provide some efficiencies in scanning operations, either hardware devices or searches through datafiles;

- secondly, they are used as spatial indexes, simplifying two dimensional addressing as single dimension addressing.

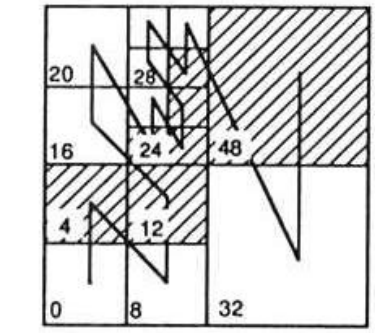
Various locational reference schemes are possible, and are indeed used for meeting different requirements.

Node level	Nodes at next level NW	NE	SW	SE	Node type	Node attributes

(a)

Locational code	Size of quadrant	Node type	Node attributes

(b)



(c)

(d)

Peano key	Size of quadrant - side length	Land use
0	2	Rural
4	2	Urban
8	2	Rural
.	.	.
.	.	.
.	.	.
28	1	Rural
29	1	Rural
30	1	Urban
31	1	Rural
32	4	Rural
48	4	Urban

One simple scheme is to consistently order the four blocks at each level in a NW, NE, SW, SE sequence, using data in the record for a tree node to point to the four nodes at a lower level if such exist (a). Numerical coding representing the NW, NE, SW and SE by integers could be used (b); some coordinate values could be used, or a space path could be employed to simplify movement through the entire set of cells, without using actual coordinate values.

Referring to figure above, coding using row and column identifiers would require more data to be stored than for a locational coding scheme, using the NW, NE, SW, SE orientations, while a Peano N path has single dimension addressing and has stable numbering across different levels of resolution. Thus the larger blocks in the quadtree would be represented by fewer positional pieces of data than the number of blocks (c), and the final table would contain items for the Peano key and quadrant size, often the number of smallest size pixels on the side of the square block.