A HYBRID APPROACH FOR AUTOMATED AREA AGGREGATION

Zeshen Wang
ESRI
380 NewYork Street
Redlands
CA 92373
Zwang@esri.com

ABSTRACT

Automated area aggregation, which is widely needed for mapping both natural and artificial features, has been getting a great deal of attention from cartographers since the manual work is so tedious and time consuming. Automated aggregation can be accomplished in either vector or raster modes. More accurate results can be achieved in vector mode, but it may cost heavy computational time especially when the aggregation distance is large and the polygon boundaries are complicated. This paper describes a method that uses both vector and raster approaches. Simple implementation of the idea can be found in research papers, but there is no detailed description about its algorithms and its industrial application. In this method, input polygons are first represented with vectors. The original polygons are then converted to patches by rasterization, and these patches are expanded to connect areas close to each other. At this point, each patch has a buffer zone around its boundary. If two areas are close, this buffer zone will connect them. This buffer is then shrunk so that the original patches and the connecting buffer zones between them are both preserved. The final step is to vectorize this result. Since simple vectorization can only produce a zig-zag boundary, the vectorization employed in the program includes different algorithms for natural features and for buildings. The results achieved by applying this approach to real world data show a good restoration of original shapes, but performance will be slow if there are a lot of input polygons and the grid cells are small.

INTRODUCTION

Geographic data must include information about both position and attribution. Most information systems only store and handle attribute data. Including spatial information makes geographic information systems more complicated. The two structures widely used for storing geographic data are raster and vector. Raster data consist of an array of grid cells, each cell referenced by a row and column number and a number of attributes. As the row and column number are directly related to the position of the associated feature, it has been well known that raster model can be very helpful for spatial searching and analysis (Burrough 1986, Aronoff 1989). To find out what is in a specific position, we need only locate the row and column number of a cell containing the position, then access the attributes of that cell by its row and column number. Vectors, on the other hand, are very good at representing the features shown on maps as lines, such as rivers, highways and boundaries. ArcInfo is basically a vector GIS software, so the input and output data are assumed to be vector data for most applications. In the hybrid approach described in this paper the raster data model, owing to its superiority for spatial searching, is used as intermediate data model.

Since area aggregation gathers nearby area features together, searching neighboring features should be the first task to conduct. A number of authors have mentioned the experiments using thickening and shrinking of rasterized area patches for aggregation (Jeworrek 1988). The idea is straightforward. Each area patch is enlarged (thickened) so that polygons that are close will touch each other. Then patches are shrunk except the cells filling the gap between the original polygons.

This idea can also be realized in the vector data model where buffer polygons can be generated for each original polygon (Muller & Wang 1992). These enlarged polygons are checked to see if their boundaries intersect. But since the polygon's boundary can be complex with slivery lines, the intersections can be too many and too dense to manipulate.

Another data model, Delaunay triangulation, has also been used by several groups of cartographers for area aggregation (Jones et al 1995). In this data model, the connection of nearby features is described by triangulation, which can be established quickly by well-studied algorithms, so spatial searching can be performed efficiently.

SEARCHING NEARBY POLYGONS

Vector data are assumed to be both input and output for this aggregation program, with the raster models serving to reach intermediate goals. The raster data model (called grid in ArcInfo) was built by applying the conversion command Polygrid to input data with a specified cell size. The original polygons are first converted to area patches in Grid. Then the rasterized patches undergo two raster operations: expansion and shrinkage. Expansion is the word in Grid for thickening.

A simple example is provided here to demonstrate these processes (Figure 1). Four polygons in 1a are converted to area patches in Grid (1b). Then the Grid function Expand is applied to them and new patches are generated (1c). Now we can see that close polygons are connected by buffers. But the patches in 1c are much bigger than their original size. To get rid of the buffer, a grid function Shrink is applied to 1c, and the results are converted back to vector mode (1d). At this point, we can see two polygon groups, each of which contains original polygons and some additional polygons generated by Expand and Shrink. There are two types of additional polygons: connecting polygons and attaching polygons. Connecting polygons must be kept in the final version, while attaching polygons do not play any meaningful role and should be excluded. In 1d, polygons 6 and 7 are connecting polygons and polygon 3 is an attaching polygon. Two conditions may generate attaching polygons. One is small concave dip in a polygon boundary. Expand fills it up but Shrink fails to remove the cells at the dip. Another case can be seen in Figure 2, where two polygons are close enough but owing to their alignment and distance Shrink separates them again.

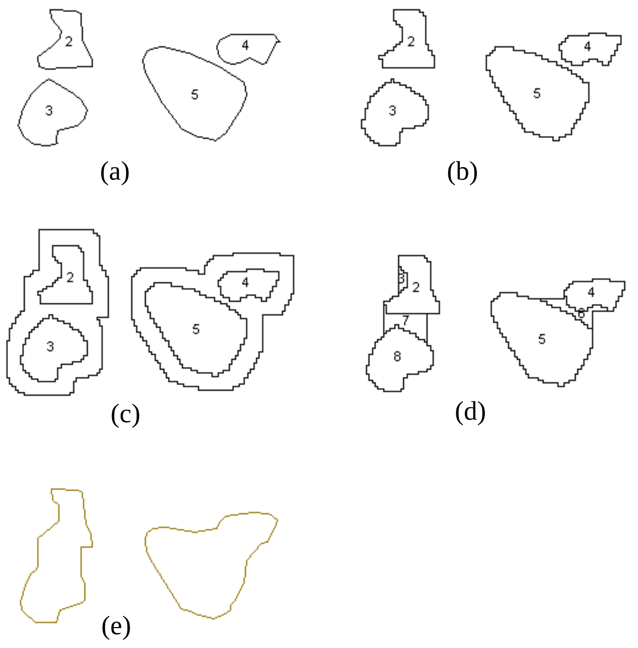


Figure 1. Processes of area aggregation in the system

When all raster manipulations are done and the results have been converted back to vector mode, the major tasks, described in the following paragraphs, include finding out the polygons in each group, searching a new boundary for polygon groups, and restoring their shapes.

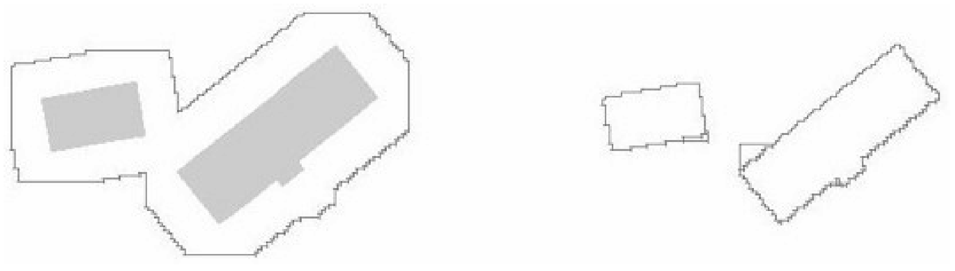


Figure 2. Shrinkage may generate attaching polygons.

SEARCHING FOR POLYGONS INSIDE A GROUP

To search for polygons inside an aggregated polygon group, we need topological information describing arcs and their adjacent polygons, information that should be available in a professional GIS. The information provided by ArcInfo includes a bounding arc list for each polygon, and left-hand and right-hand polygons for each arc. In ArcInfo, the arc list is given by the arc ID numbers that enclose the polygon going clockwise. As the outside-polygon space in ArcInfo is assigned to polygon 1, or the universe polygon, the first polygon in every ArcInfo Coverage is polygon 2 instead of polygon 1 (Figure 3).

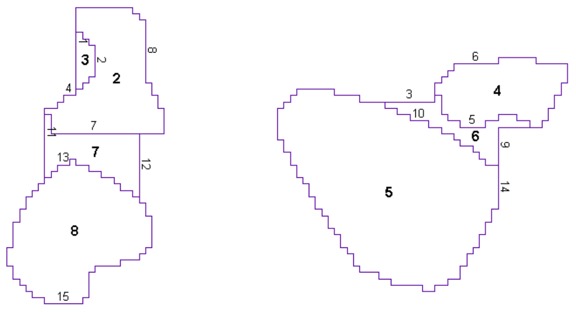


Figure 3. Polygons and arcs after expansion, shrinkage and conversion back to vector.

This figure is an enlarged version of Figure 1d. The numbers printed in bold are polygon IDs, and the small numbers are arc IDs. Group one consists of two polygons, originally 2 and 3 (see Figure 1a). These two are marked as 2 and 8 because when polygons are added some polygons will get renumbered according to their position. How to preserve the original polygon's ID will be mentioned later. For now, the task is searching the polygons for each group in this coverage. Searching follows these steps:

1) Take the first polygon of the coverage, polygon 2 as the current polygon and put the number 2 into an empty polygon array.
2) Read the polygon file to get its bounding arcs. For polygon 2, they are arcs 8, 7, 4 and 2. Append these numbers into an arc ID stack.
3) Pick up the latest-added arc ID number in the stack, which is 2, read the arc file to get its left and right-hand polygons. In this case, the outer side of arc 2 is polygon 3. It should be added to the polygon array and become the current polygon. If the outer side of an arc is the universe polygon,

skip the arc and pick up next arc in the stack. When a new neighbor polygon, polygon 3 here, is found, go back to step 2.

In step 2, when a new polygon is found, its bounding arcs will also be appended into the arc stack. Here they are arcs 2 and 1, and arc 2 will be ignored since it has been used to find polygon 3 from polygon 2, and cannot tell us about any new polygon. At this point, the updated arc stack should contain arcs 8, 7, 4 and 1.

By repeating this process, another two polygons in the same group, polygons 7 and 8, will be found and added to the polygon array. A bitmap is used to mark the polygons already found. When the arcs in the arc stack are exhausted, the polygon list for a group should be completed. After all polygons in the first group are found, the program will start to retrieve its outer rings and restore their shapes. When the first group is processed, we can check the bit map and move to next polygon to start a new group. The goal for searching polygons in one group is two-fold: to establish the relationship between input original polygons and output aggregated polygons; and to limit the search for the outer ring to the polygons inside the group.

SEARCHING THE BOUNDARY OF A GROUP

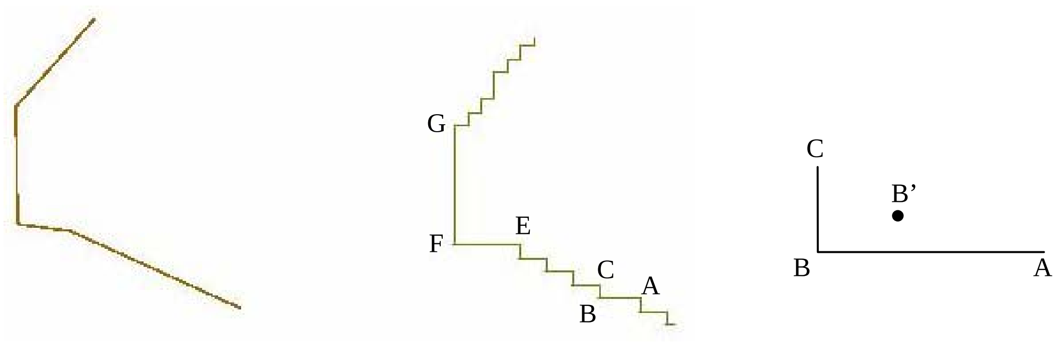
Each new polygon group must have an outer ring, and some of them may have newly-formed holes. Searching outer rings needs the same topological information as searching polygons inside a group. But the procedure is quite different. The process starts by picking up the first polygon from a polygon group, in this example, polygon 2. Its bounding arcs 8, 7, 4 and 2 can be found from reading the polygon file. The outer side polygon can be read from the arc file. If the outer side is the universe polygon, the arc should be a part of the new boundary, and its ID number will be saved into an output array. Otherwise skip it. In this example, arc 8 is the first arc that meets the criterion, it is saved as the first arc in the output arc array. The second arc 7 is then checked. Its outer side is polygon 7, a connecting polygon. Since a connective polygon should be a part of the integrated polygon, so polygon 7 now is taken as the current polygon. On its bounding arc list, arc 7 is followed by arc 12, which is adjacent to the universe polygon. So arc 12 is appended to the output arc array. In this way, the arcs that enclose the group can be found as 8, 12, 15, 11, 4 and 2. Also, the inner-side polygon for each arc should be saved for later use in shape restoration. They are polygons 2, 7, 8, 7, 2 and 2 in this example. Note that the last arc in this list is arc 2 instead of arc 1, since polygon 3 is an attaching polygon that should be excluded. An attaching polygon can be detected by checking the number of its bounding arcs. Unlike connecting polygons that have four arcs or more, this kind of polygon has only two adjacent polygons, one original polygon and the universe polygon. Therefore it has only two bounding arcs.

It is worth noting how the program identifies an original polygon from an additional polygon. When we convert an ArcInfo coverage to a Grid, the polygon ID may get lost. One way to save the ID is to take it as Grid code when we convert it so that the polygon's ID will be attached to each area patch. Even when all of the operations in raster are done and the results have been converted back to vector mode, the Grid code is still available as an item in polygon's attribute tatble.

Newly-formed holes may appear inside a group (see Figure 8). Fortunately, a hole polygon always carries the universe polygon's ID, that is 1, as its Grid code. Therefore all we need to do is check every polygon in the current group to see if it has such Grid code. If yes, we should keep its bounding arc list just as we have done for the outer ring.

RESTORATION OF SHAPE

We now have a complete output arc list. However we should not simply output the zig-zag lines without modification. We need to restore the outlines to their original shape. In our example, the output arc list of the first group includes the following arcs: 8, 12, 15, 11, 4 and 2. Since the inner-side polygon for each arc was saved, we can easily tell which arcs enclose which polygon. As arc 4, 2 and 8 all have inner-side polygon 2, and arc 15 has polygon 8, we can form two lines, one from arc 4, 2, and 8, and the other from arc 15. To derive a smooth line from such a zig-zag line, a three-point moving average is applied to every three vertices on the line, followed by applying the Douglas algorithm (Douglas & Peucker 1973) to eliminate the redundant points. Figure 4c demonstrates the three-point moving average applied to three vertices ABC (enlarged from 4b). Point B will be replaced by their “average” position B’, the center point of triangle ABC. An exception should be made when both sides are longer than a cell size. In 4b EFG has two long sides EF and FG; the second point F should be left in its original position rather than moved to the center point of the triangle EFG.



a. original line b. rasterized line c. new line will go through B’

Figure 4. Three-point average.

So far, the shape of outlines for each original polygon has been restored. One line is derived from arcs 4, 2 and 8. The second line is from arc 15. But the two lines are still separated. Natural features (Figure 3) can be simply connected by straight segments, no matter if the connecting arc is a straight line (arc 11, 12) or L-shaped (arc 9 and 3).

Smoothing by averaging only applies to natural features. For artificial features, particularly for buildings, a different approach is needed to restore shapes. Buildings are usually formed by straight-edges and right-angle corners. When a building polygon is converted to Grid, the zig-zag line corresponding to a straight building edge has the following characteristics (Figure 5):

1) Segments are either horizontal or vertical, and all the corners are right angle. Their turning directions are positive and negative alternately.

2) The length of each segment is always a certain multiple of the cell size. Every second segment is one cell long. In other words, either horizontal or vertical segments are one-cell long. The longer segments in other direction are either N or N+1 times the cell size (N is an integer). For line AB, all vertical segments are one cell long and the horizontal segments are either one or two cells long (N = 1).

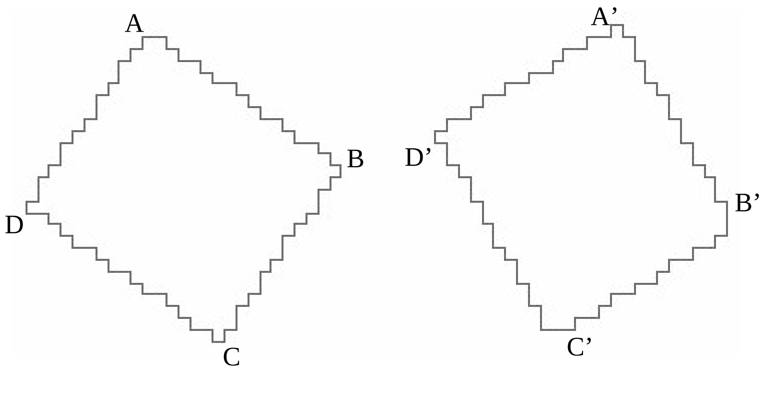


Figure 5. Rasterized building polygons.

Line DAB from Figure 5 is enlarged and shown in Figure 6. If the building's edge is long enough, we can simply connect the center points of the first short side and the last short side to form a straight line (EF and GH). When we have two consecutive straight lines, EF and GH, their intersection can be calculated to restore the corner.

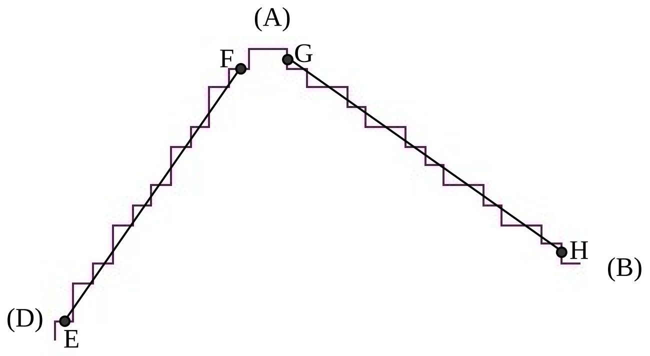


Figure 6. Shape restoration for a building's edges and corner

So far, the shape of individual buildings is restored. Unlike aggregating natural features where we simply connect two polygons by straight lines, projection is needed toreserve the building's orthogonal characteristics. Figure 7 demonstrates this. Figure 7b is a Grid representation of the original buildings from 7a after applying Expand and Shrink. Arcs 3 and 6 are outlines of original building polygons. Their shapes are restored as shown in Figure 7c. To connect the contours of two buildings, the ending segment AB of arc 3, is projected to CD, the beginning segment of arc 6. The other arc ends can be connected in the same way (Figure 7d).

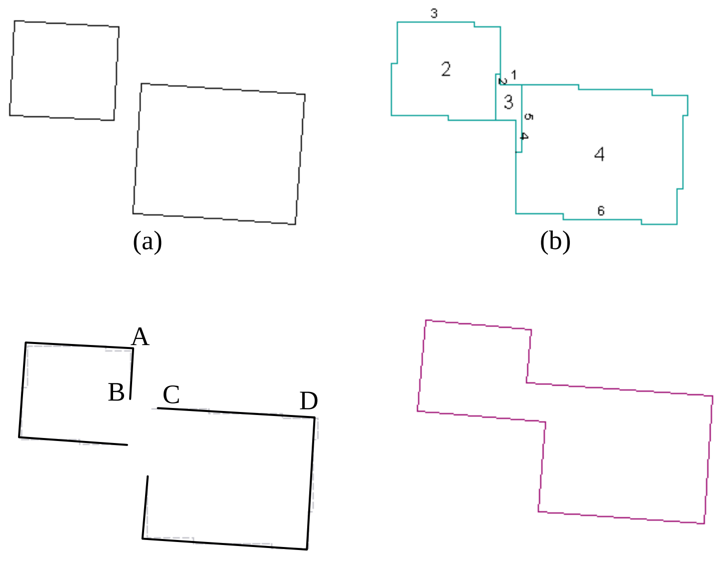


Figure 7. Join buildings by projection

REAL WORLD DATA TEST

The program has been extensively tested with real world data. Two sample data sets, one natural and the other a building cluster, are shown below to demonstrate the details of area aggregation. The first figure shows original polygons. Next to it are the results of aggregation. The bigger drawing underneath shows the intermediate results of expansion and shrinkage (gray lines) combined with the final results (solid lines).

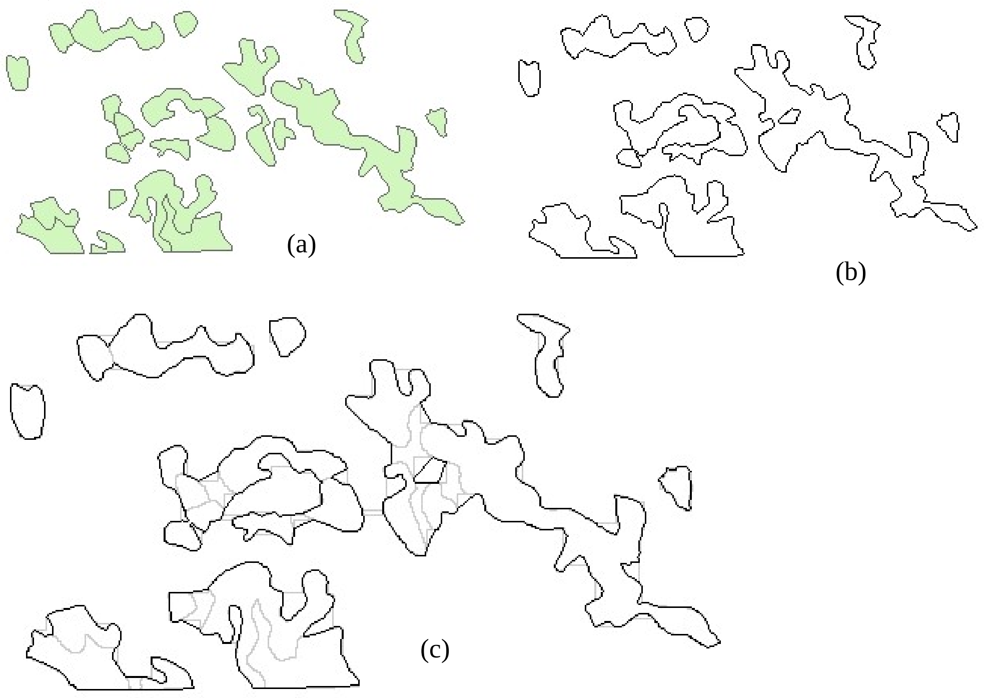


Figure 8. Aggregation of natural features.

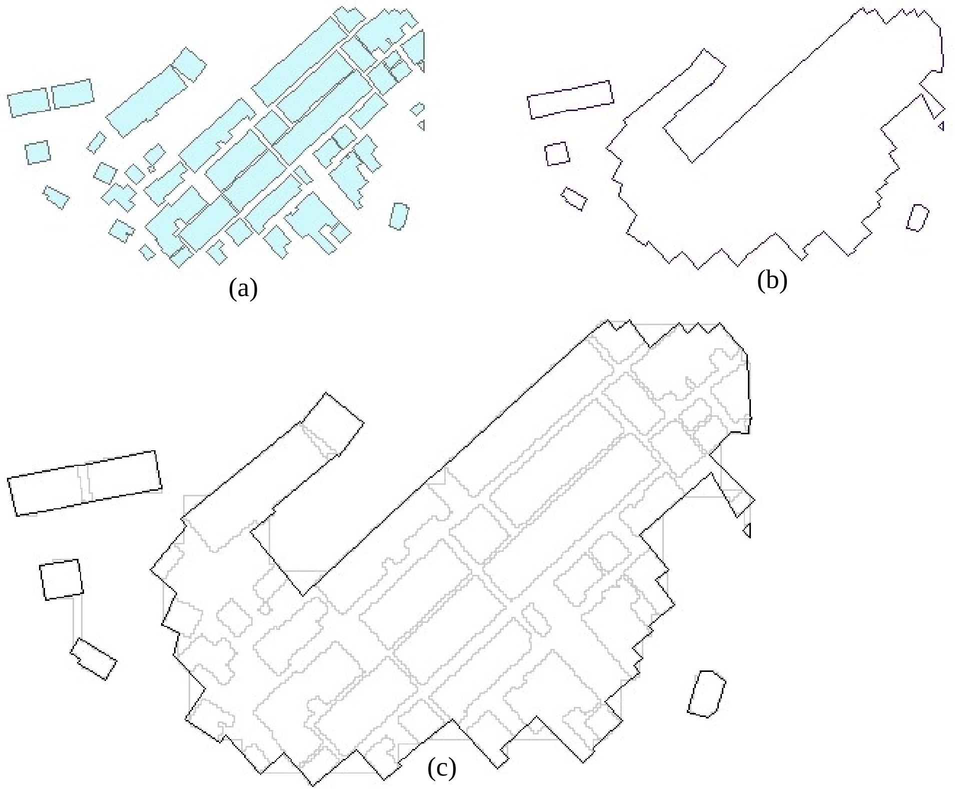


Figure 9. Aggregation of buildings.

AGGREGATION DISTANCE

A close examination of Figures 8c and 9c will reveal some instances where polygons are connected by a narrow corridor, but not connected in the final version. Generally speaking, Expand follows the user-defined aggregation distance very well. But Shrink may separate the two polygons again if they are not aligned horizontally or vertically. Suppose two buildings, A and B (Figure 10b). If both the X range and Y range of building A do not overlap with building B, they are not aligned horizontally and vertically.

If the shrinkage function does not separate them but leaves a very thin connection (Figure 10c), such a narrow connection will be detected and ignored, and the poorly connected polygons will be disconnected.

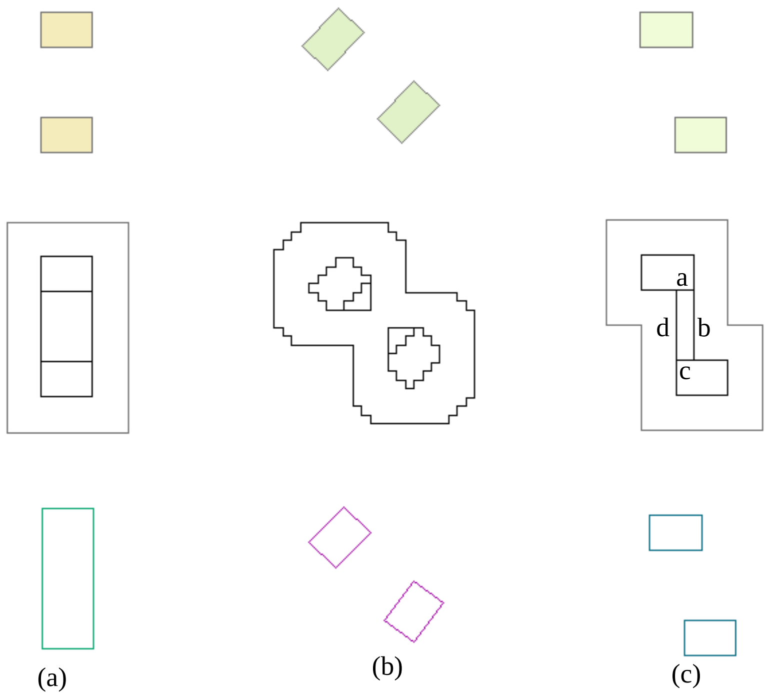


Figure 10. Special cases where close polygons may not be aggregated.

Figures 10a and 10b show two building polygons the same distance apart but with different alignments. On top are original polygons, in the middle are the results of both expansion and shrinkage, and on the bottom are aggregation results. The two buildings in (a) get aggregated but not the two in (b) owing to their alignment. But alignment of buildings should not affect the aggregation results. This is a drawback of using the raster model for aggregation, though the problem only occurs when a big aggregation distance is specified. In this example, the aggregation distance is much bigger than the buildings' dimension and should be reduced. The two polygons in 10c are about the same distance apart as those in 10a. But they are shifted, which causes the generation of a narrow connection polygon. The program will disconnect the two polygons by measuring the ratio of its length to its width. The formula used for this length-width ratio is (d+b)/(a+c). The connecting polygon won't necessarily be a rectangle like this one. In Figure 3, for example, the connecting polygon 6 is complicated. After replacing the four arcs of the polygon 6 by straight segments we can apply this formula.

CONCLUSIONS

The raster data model has been used for area aggregation for long time. This paper comes out of the experience of developing an operational program for ArcInfo GIS. Several existing Grid functions, including expansion and shrinkage, are employed in this hybrid system to reach the intermediate goals for gathering close polygons together. The focus then shifted to technical issues of the vector processes after all raster manipulations are done. The main steps there include searching polygons and boundaries for integrated polygon groups and restoring original shapes from raster-converted lines.

Tests have been applied to a variety of data sets and the results are visually pleasing and positional accuracy is acceptable if the cell size is small. Another advantage of this hybrid approach is its well-defined steps and relative simple computation. There are some drawbacks for this hybrid approach: the conversion between vector and raster models is slow, as are the raster operations, such as expansion and shrinkage, especially when the cell size is small. The aggregation distance is not consistently followed when the polygon alignment is changed and the aggregation distance is big. Manual area aggregation, not covered in this paper, occasionally requires feature to move or rotate before being merged with other features. The experiment reveals the difficulty of measuring the relative position between polygons and conducting these necessary movements before accomplishing aggregation.

REFERENCES

Aronoff S (1989) *Geographic Information Systems: A Management Perspective*, WDL Publications, Ottawa, 164-68.

Burrough P A (1986) *Principles of Geographical Information Systems for Land Resources Assessment*, Oxford University Press, Oxford, 36.

Douglas D and Peucker T (1973) *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*, The Canadian Cartographer 10(2): 112-22.

Jeworrek J (1988) Untersuchungen zur automatischen generalisierung von flachen im Rasterdaten format. Unpublished Master’s Thesis, University of Hannover.

Jones C B, G L Bundy, et al. (1995) *Map Generalization with a Triangulated Data Structure*. Cartography and Geographic Information Systems. 22(4): 317-31.

Muller J-C and Wang Z (1992) *Area-Patch Generalization: A Competitive Approach*, The Cartographic Journal 29: 137-44.