Matching and Aligning Features in Overlayed Coverages

J. Mark Ware and Christopher B. Jones
School of Computing, University of Glamorgan
Pontypridd, CF371DL, Wales, UK
+44 (0)1443 480480
jmware,cbjones@glam.ac.uk

1. ABSTRACT

The problems caused by locational error when overlaying spatial data from different sources have been recognised for some time, and much research has been directed towards finding solutions. In this paper we present a solution in the form of an algorithm that seeks to match and align semantically equivalent features prior to overlay. It is assumed that, because of locational error, semantically equivalent features will not always be geometrically equivalent. The technique has been developed to assist in the detection of change between multi-date vector-defined data sets. Initial results, obtained by applying our algorithm to land cover data, are presented.

1.1 Keywords

Geometric overlay, locational error, equivalence testing, change detection, conflation

2. INTRODUCTION

It is inevitable that spatial databases contain error of some kind. The very fact that spatial data sets have to represent real world phenomena in an abstract and generalised form means they can never be truly accurate. Error can be introduced at all stages of the database construction process ([10]). Data gathering is subject to the accuracy of the particular technique being used. For example, data derived from remotely sensed images will include errors that are due to the characteristics of the airborne platforms and sensor systems.

Further errors are introduced during data compilation. For example, the interpretation of remotely sensed images can be highly subjective, particularly when dealing with natural domains without precise boundaries. Data resulting from the processing of existing data sets can include additional error due to error propagation [15]. For example, a coverage obtained by overlaying an existing pair of coverages will contain error that is some function of the error contained in each of the source data sets.

Spatial data error can be grouped into two classes, namely, locational error and attribute error. In this paper we are concerned with locational error only, and examine it within the context of detecting change between pairs of multi-date, vector-defined coverages. The standard technique adopted for change detection involves the geometric overlay of two coverages, *A* and *B*, followed by analysis of the attributes of features in the derived coverage, C. Certain features in *A* will be semantically equivalent to certain features in *B* (i.e. they represent the same real world feature). However, because of locational error, any given semantically equivalent pair of features, F^A and F^B, will not necessarily be locationally (geometrically) equivalent. As a result, the attributes of some features in *C* will indicate change, whereas, in fact, no change has taken place. In an attempt to overcome this problem, we propose a method that seeks to ensure that semantically equivalent features are represented by precisely the same geometry. This is achieved by aligning *A* and *B* prior to overlay. The alignment process makes use of a feature matching procedure that takes locational error into consideration. Having established equivalences, matching feature pairs are replaced by a third feature, F^{AB}, within their respective coverages. F^{AB} is derived by calculating a weighted average of the two original features. Although specifically developed to assist in change detection, the algorithm presented is applicable to overlay in general.

3. PREVIOUS WORK

Data error, and the problems it causes during overlay, has been recognised for some time [2]. A number of attempts have been made at addressing the problem, many of which make use of an epsilon band [12]. When using this band, the error associated with a particular feature is represented by a zone of width epsilon around that feature. In its deterministic form, the model assumes a probability 1 that the true line lies within the epsilon band and a probability of 0 of it lying outside [1]. Alternatively, in the model's probabilistic form, confidence in a point's membership of a particular line decreases (in accordance to some probability distribution) as the point's distance from that line increases [7].

Dougenik [4] presents a program (Whirlpool) that, among other things, performs polygon overlay. Whirlpool is designed to reduce the number of sliver polygons produced as a result of overlay. It deals with error by making use of what is termed the error distance. The general idea is that the concept of line intersection is extended to include that of fuzzy line intersection. A fuzzy intersection occurs between two lines if two points, one on each line, are within the error distance of each other. Lines are broken at intersection points, and points closer to each other than the error distance are coalesced. A clustering algorithm determines which points move and which remain fixed. This algorithm is described in detail in [3]. Zhang and Tulip [17] and Pullar [13] present related work. The clustering methods used in the algorithms given in these papers work so as to move less accurate nodes to more accurate nodes.

Lupien and Moreland [11] define map conflation as the process of identifying features in two maps that represent the same real-world feature, or are in the same location, then selectively merging features and attributes of both into a third. It may also be regarded as a process of selective merging whereby the best quality elements of the two maps are selected to create a third composite map. Lupien and Moreland [11] describe an iterative conflation technique made up of two stages, namely coverage alignment and feature matching. Coverage alignment is achieved by triangle-based rubber sheeting. The rubber sheeting makes use of user defined links between matching locations in the source coverages. These links are used to construct a pair of distortion surfaces that are then used to transform features. Feature matching is based on distance measures involving points and arcs. The basic idea is that two points are matched if they lie within a specified tolerance of each other, while two arcs are matched if all points on one arc are within tolerance of the other. Saalfeld [14] presents a conflation technique in which feature matching can be achieved using both spatial information (e.g. location and shape) and attribute information (e.g. name). Matched features are again aligned using triangle-based rubber sheeting. Although originally intended for consolidating maps that are known to contain the same features, the methods employed in conflation can be adapted for other applications.

The alignment overlay technique given by Harvey [8], and later discussed in Harvey and Vauglin [9], is specifically concerned with overlays where one coverage (*A*) has a higher accuracy than the other coverage (*B*). In such circumstances Harvey suggests that, during an alignment process, the nodes (or vertices) of *A* should remain fixed. It is assumed that each coverage has an associated error tolerance. Coverage *A*'s tolerance is termed the match tolerance; coverage *B*'s tolerance is termed the shift tolerance. The basic concept of overlay alignment is if a feature F^A belonging to *A* is within match tolerance or shift tolerance of a feature F^B belonging to *B* then the respective segment of F^B should be moved to an existing or newly created node in F^A. The paper lists twelve potential matching situations that may occur between F^A and F^B, and the subsequent actions that should be taken.

The work presented by Edwards [5] and, more recently, by Edwards and Lowell [6] is of particular relevance since it deals with change detection. The earlier paper outlines a framework for characterising the spatial uncertainty of inherently fuzzy boundaries by analysing several independent interpretations of the same scene. The analysis is achieved by clustering curves (feature boundaries) based on proximity, and by characterising clusters in terms of average cluster location and a variance box. A cluster is identified as representing change if the curve-to-curve distance of its constituent features exceeds some tolerance value.

4. FEATURE ALIGNMENT ALGORITHM

We now present a description of our feature alignment algorithm. The techniques used are grouped into two main stages, namely, feature matching and feature update.

4.1 Feature Matching

Consider coverages *A* and *B* to be made up of edges E^A and E^B, respectively. A feature is defined here as being any series of connected edges. Two edges are connected if they share a vertex. Each edge e_A belonging to E^A has an associated error tolerance ε^A, while each edge e_B belonging to E^B has an associated error tolerance ε^B. In the work presented here we make use of the deterministic epsilon band. Therefore, when comparing *A* and *B*, we consider a combined error tolerance ε, where $\varepsilon = (\varepsilon^A + \varepsilon^B)$.

4.1.1 Containment Relationships

Using an epsilon band of radius ε, it is possible to define various containment relationships that can exist between a pair of edges e_A and e_B. These are: (i) e_A fully contains e_B. This is true when both vertices of e_B are within a distance ε of e_A (Figure 1, e_{B1}); (ii) e_A partially contains e_B. This is true when only one vertex of e_B is within a distance ε of e_A (Figure 1, e_{B2}), or if no vertex of e_B is within a distance ε of e_A but one or both vertices of e_A, or the interior of e_A, is within a distance ε of e_B (Figure 1, e_{B3} and e_{B4}); and (iii) e_A does not contain e_B (Figure 1, e_{B5}).

4.1.2 Containment Sets

For each edge e_B belonging to E^B we find an associated edge containment set E_c^{eB} consisting of all edges in *A* which contain it , either fully or partially (E_c^{eB} will sometimes be empty). For example, in Figure 2 the edge containment set of e_B consists of edges e_{A2}, e_{A3}, and e_{A4}.

Then for each edge e_B we inspect E_c^{eB} and from it construct sets of connected edges E_f^{eB1}, E_f^{eB1},…, E_f^{eBn}, each of which fully contains e_B. These sets are referred to as fully containing sets. In Figure 3 e_B has two such sets.

It will sometimes be the case that a containment set E_c^{eB} will contain a series of connected edges E_p^{eB} that only partially contains e_B (a partially containing set). In such an event e_B is split at a point *v*, such that the minimum distance between *v* and E_p^{eB} equals ε. Two new edges e_{Ba} and e_{Bb} are thus formed, and e_B is discarded (Figure 4). An attempt can now be made to find the fully containing sets of each of the new edges. New edges may also require splitting, but eventually edges will be found which do not have partially containing sets.

Each edge e_B is now designated either as being fully contained by *A* (i.e. the edge is associated with one or more fully containing sets) or as not being contained by *A* (i.e. the edge does not have any associated fully containing sets).

The edges of E^A are now processed in similar fashion. A slight variation is required when dealing with a partially containing set E_p^{eA}. As before, when splitting e_A at point *v* two new edges e_{Aa} and e_{Ab} are formed,

and e_A is discarded. However, certain edges belonging to B will reference e_A in their associated fully containing sets. It is necessary to replace these references with references to e_{Aa} and e_{Ab}.

4.1.3 Matching Features

We now consider coverage A. Each fully contained edge e_A has n associated fully containing sets E_f^{eA1}, E_f^{eA2},…, E_f^{eAn}. Let us first of all consider the simplest case where n equals one for all edges.

We start with a list L_A of all fully contained edges belonging to A. A single edge e_{A1} is removed from this list, and from it we build a list of connected edges F^A (e_{A1}, e_{A2},...., e_{Ai}) and a list of connected edges F^B (e_{B1}, e_{B2},...., e_{Bk}) such that : (i) F^A fully contains F^B; and (ii) F^B fully contains F^A (Figure 5).

Initially, F^A is set to e_{A1} and F^B is set equal to E_f^{eA1}. Next, edges connected to e_{A1} are examined in turn, checking each time if the particular edge is fully contained. If no directly connected edge is fully contained then processing of e_{A1} terminates; it follows that F^A consists of the single edge e_{A1}, and that that F^B consists of all edges in E_f^{eA1}. If, however, a connected edge e_{A2} is fully contained then E_f^{eA1} and E_f^{eA2} are examined. If E_f^{eA1} and E_f^{eA2} are themselves connected then e_{A2} is added to F^A, E_f^{eA2} is added to F^B, and e_{A2} removed from L_A (Figure 6).

Note that when constructing F^A and F^B, we ensure that the valency of constituent vertices does not exceed two. This is necessary for the feature update procedure (section 4.2) to work correctly. All edges added to F^A subsequent to e_{A1} in turn undergo the same processing as e_{A1}. In this way a matching pair of features of the kind shown in Figure 5 is found. Having constructed F^A and F^B, a new starting edge e_{A1} is removed from L_A and the process is repeated. This continues until L_A becomes empty.

4.1.4 Multiple Matches

In cases where n is greater than one, choosing the correct fully containing set is not straightforward. The choice will ultimately determine the make-up of F^A and F^B. At present, to simplify matters, we assume that each feature in A can be equivalent to, at most, one feature in B. In an effort to make the best matches, we have implemented techniques which process all possible combinations of fully containing sets, starting at e_{A1}, assembling all possible pairs of matching features in the process. Having done this there are several possible ways to chose the most appropriate pair (Figure 7). One technique might be to reduce ε in the hope that in doing so the number of fully connecting sets per fully contained edge reduces to one. Alternatively, we could use some additional metric, such as a shape measure, for determining equivalence. In our current implementation, we simply choose the longest matching pair of features.

4.2 Feature Update

The next step is to update A and B so that those features deemed to be equivalent are represented by the same geometry. This is done by replacing a pair of features F^A and F^B with a single feature F^{AB}. F^{AB} is found by averaging F^A and F^B. In fact we use a weighted average, where the weighting is related to the relative sizes of ε^A and ε^B. There are three steps in the update process.

4.2.1 Vertex Insertion

The first step is to add vertices to F^A and F^B such that : (i) F^A and F^B have the same number of vertices; and (ii) vertices appear at proportionally equivalent distances along F^A and F^B. Each vertex in F^A now has a corresponding vertex in F^B (Figure 8).

4.2.2 Weighted Averaging

F^{AB} is now found by calculating the weighted average of corresponding vertices in F^A and F^B (i.e. $v_{AB} = Wv_A + [1\text{-}W]v_B$, where $0<=W<=1$) (Figure 9). F^A and F^B are discarded. In situations where one coverage is of a much higher accuracy than the other (i.e. the difference in accuracy exceeded a preset threshold), the averaging procedure could be replaced by a procedure which simply selected the more accurate of F^A and F^B.

4.2.3 Feature Connecting

Each coverage now consists of two types of feature: (i) original features, consisting entirely of edges not contained by edges of the other coverage; and (ii) updated features, which are the result of steps 4.2.1 and 4.2.2. A third type of feature is required to deal with the situation where a break occurs between an original feature and a previously connected updated feature.

Currently, a break is dealt with by inserting a connecting feature. Each connecting feature consists of a single edge, which is made up from the appropriate start/end vertices of the original and updated features (Figure 10). However, it will be noted from the test results illustrated in Figure 11 that spanning breaks in this way produces artificial looking features in an around connecting features. A possible alternative, currently under investigation, is to use a weighted progressive merge.

5. EXPERIMENTAL RESULTS

The feature matching and update procedures described have been implemented as C functions on a Sun workstation. During the construction of containment sets heavy use is made of a triangle-based spatial search scheme, as described by Ware and Jones [16]. This ensures that the search for containment sets is performed efficiently.

Testing has been carried out using data supplied by the Macaulay Land Use Research Institute (MLURI). Two data sets, representing land cover in part of the Cairngorms in 1946 and 1964, have been used. Figure 11 shows the outline of a pair of polygons of a particular land cover type, one for each of the dates, together with results obtained by applying the feature alignment algorithm with a variety of error tolerance values.

6. CONCLUDING REMARKS

We have presented a technique for matching and aligning features in pairs of multi-date coverages. The method, which has been tested on land cover data, is seen to offer a number of advantages. Firstly, unlike some methods that only consider error at vertices, our method considers error around and along individual edges. As such, the correct matching of features it is not as dependent upon they way in which a particular feature has been sampled. Secondly, it considers each edge within the context of the feature to which it belongs. This provides us with the opportunity to deal more effectively with problem situations, such as multiple matches and invalid matches between parts of features. Thirdly, vertex displacement is tightly controlled and for a particular vertex is directly related to the error tolerance associated with the feature to which it belongs.

Future work will involve more rigorous testing of the algorithm, accompanied by the design and implementation of refinements concerning the merge procedure at line breaks and methods for selecting best matches. Note that the work is being carried out as part of a much larger project. This project is looking at a wider range of issues pertaining to automated change detection, such as taking account of classification uncertainty and the possibility of varying levels of generalisation (or abstraction) between different data sets. Integrating a range of algorithms into a general-purpose change detection software package is our ultimate goal.

7. AKNOWLEDGEMENTS

The authors express thanks to MLURI for supplying data and technical support. JMW was supported by the NERC grant GR3/10569.

8. REFERENCES

[1] Blakemore, M. 1984, Generalization and error in spatial databases, Cartographica, 21, 11-139.

[2] Chrisman, N.R. 1987, The accuracy of map overlays: A reassessment, Landscape and Urban Planning, 14, 427-439.

[3] Chrisman, N.R., Dougenik, J.A. and White, D. 1992, Lessons for the design of polygon overlay processing from the Odyssey Whirlpool algorithm, Proc. 5th Int. Symp. on Spatial Data Handling, 2, 401-410.

[4] Dougenik, J.A. 1979, WHIRLPOOL: A geometric processor for polygon coverage data, Proc. AUTOCARTO 4, 304-311.

[5] Edwards, G. 1994, Characterising and maintaining polygons with fuzzy boundaries in geographic information systems, Proc. 6th Int. Symp. on Spatial Data Handling, 1, 223-239.

[6] Edwards, G. and Lowell, K.E. 1996, Modelling uncertainty in photointerpreted boundaries, Photogrammetric Engineering and Remote Sensing, 62(4), 337-391.

[7] Goodchild, M.F. 1991, Issues of quality and uncertainty, in J-C. Muller (ed) Advances in Cartography, Elsevier, 113-139.

[8] Harvey, F. 1994, Defining unmovable nodes/segments as part of vector overlay: The alignment problem, Proc. 6th Int. Symp. on Spatial Data Handling, 1, 159-176.

[9] Harvey, F. and Vauglin, F. 1996, Geometric match processing: Applying multiple tolerances, Proc. 7th Int. Symp. on Spatial Data Handling, 1, 4A.13-29.

[10] Hunter, G.J. and Beard, K. 1992, Understanding error in spatial databases, The Australian Surveyor, 37(2), 108-119.

[11] Lupien, A.E. and Moreland, W.H. 1987, A general approach to map compilation, Proc. AUTOCARTO 8, 630-639.

[12] Perkal, J. 1966, On the length of empirical curves, Discussion Paper No. 10, Michigan Inter-University Community of Mathematical Geographers, Ann Arbor.

[13] Pullar, D. 1991, Spatial overlay with inexact numerical data, Proc. AUTOCARTO 10, 313-329.

[14] Saalfeld, A. 1988, Conflation: Automated map compilation, Int. J. Geographical Information Systems, 2(3), 217-228.

[15] Veregin, H. 1995, Developing and testing an error propagation model for GIS overlay operations, Int. J. Geographical Information Systems, 9(6), 595-619.

[16] Ware, J.M. and Jones, C.B. 1996, A spatial model for detecting (and resolving) conflict caused by scale reduction, Advances in GIS Research 2 (edited by M.J. Kraak and M. Molenar), 547-558.

[17] Zhang, G. and Tulip, J. 1990, An algorithm for the avoidance of sliver polygons and clusters of points in spatial overlay, Proc. 4th Int. Symp. on Spatial Data Handling, 1, 141-150.