PERGAMON

Computers & Geosciences 27 (2001) 427–440

COMPUTERS & GEOSCIENCES

# Contour correspondence for serial section reconstruction: complex scenarios in palaeontology

Malcolm J. Herbert[a,*], Christopher B. Jones[b]

[a] BECTA, Science Park, Coventry CV4 7JJ, UK
[b] Department of Computer Science, Cardiff University, PO Box 916, Cardiff CF4 3XF, UK

Received 19 October 1998; accepted 30 November 1999

**Abstract**

Serial sectioning is used in a number of areas in science as a means of viewing the internal features of a three-dimensional object as a set of two-dimensional images. The sections are used to recreate three-dimensional computer models of the original objects by constructing surfaces between associated contours on adjacent sections. This has become a common technique for medical imaging, but is also used in a number of areas in the earth sciences, including palaeontology. This paper addresses the correspondence problem, that of matching contours in adjacent sections prior to constructing three-dimensional surfaces between them. The lack of a successful automatic approach to this stage of the reconstruction process has until now hindered the exploitation of vector data consisting of vertices and edges, derived by digitising sectional data. A new growing algorithm is proposed that uses both spatial information from the object and user-supplied semantic information describing generic characteristics of specific types of phenomena. The algorithm has been used to direct the correspondence aspects of reconstruction in a number of sectioned palaeontological data sets. © 2001 Published by Elsevier Science Ltd.

*Keywords:* Serial sections; Palaeontological reconstruction; Surface modelling

## 1. Introduction

Serial section reconstruction is now a commonly used computer-based technique for medical imaging (Rhodes, 1991). There are two main types of reconstruction algorithm, which can be defined by the type of data they use. The first is the volume-based approach, which uses a set of volume-elements (or voxels) as the basis of reconstruction. In medical imaging, these voxels are captured non-invasively using scanning equipment, such as in computer-assisted tomography (CAT) or magnetic resonance imaging (MRI). This approach is now widely used in medical imaging (for example, Herman and Liu (1979) and Lorensen and Cline (1987)) and other areas of scientific visualisation (Elvins, 1992). In contrast, a surface-based method uses vector data (vertices and edges) as the basis for reconstruction. Rather than being scanned, the data will have been captured from drawings or photographs of two-dimensional sections, using a device such as a digitising tablet. While this type of data is rare in medicine, it is quite common in the earth sciences, especially in palaeontology. With a surface-based approach the object is sampled by a series of parallel sections, on which the contours represent the intersection between the object and the section. It is possible to convert from vector to raster data format in some cases, and then use a volume-based algorithm (Jones and Chen, 1994) to carry out the three-dimensional reconstruction.

Surface-based algorithms generally consist of two stages:

*Corresponding author.
*E-mail address:* malcolm_herbert@becta.org.uk (M.J. Herbert).

1. *Correspondence*: This determines the connectivity between contours on adjacent sections, therefore defining the underlying structure of the object.
2. *Surface triangulation*: Guided by the results of the correspondence analysis, this second stage of the algorithm constructs the three-dimensional surfaces of the object.

This paper concentrates on the difficulties in determining a solution to the correspondence problem. The shortcomings of existing correspondence techniques have limited the development of surface-based methods and have led to volume-based algorithms being used in preference, where the type of data permits (Udapa and Herman, 1989). There have been a number of previous automatic solutions to the correspondence problem, but none of these is capable of reconstructing objects with disjoint components (Fig. 1), such as those found commonly in palaeontology.

With most algorithms that use a voxel approach, correspondence is determined implicitly by the relationship between each cell in neighbouring sections. The algorithm of Boissonnat (1988) uses vector data, but it does not employ an explicit analysis of correspondence, as it triangulates the whole slice between a pair of sections simultaneously. The main problem with, not using a correspondence algorithm is, that connectivity between contours is decided by local, purely geometric factors. Only with well-sampled objects that do not contain holes, and where the sectioning has taken place perpendicular to the main axis of the object, will the algorithm work well. One of the main problems is that there is no way of preventing connections between contours in disjoint components, or guiding non-straightforward correspondence situations (see Fig. 2).

The three-dimensional Delaunay triangulation proposed by Boissonnat has been developed further by Geiger (1993). It has also been tested with palaeontological specimens and compared with the other reconstruction methods (Herbert and Tough, 1995). Investigations to date indicate that the approach works well with palaeontological samples that are well sampled and that do not exhibit significant change in contour position between neighbouring sections.

The surface triangulation stage has been widely developed and is extensively covered in the literature, including Christiansen and Sederberg (1978), Fuchs et al. (1977), Keppel (1975), Meyers (1994) and Tipper (1976, 1977).

A new algorithm is proposed here, CorresGrow, which uses both spatial and semantic information to derive a correspondence graph for an object. The algorithm uses a construction technique that grows the components in an object individually rather than considering every connection between a pair of contours simultaneously, as with a global method. It is felt that this type of algorithm is especially suitable for objects that contain disjoint components and where complex correspondence situations may exist.

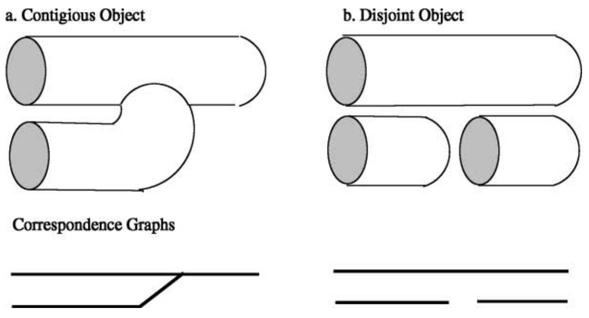


Fig. 1. Contiguous and disjoint components.

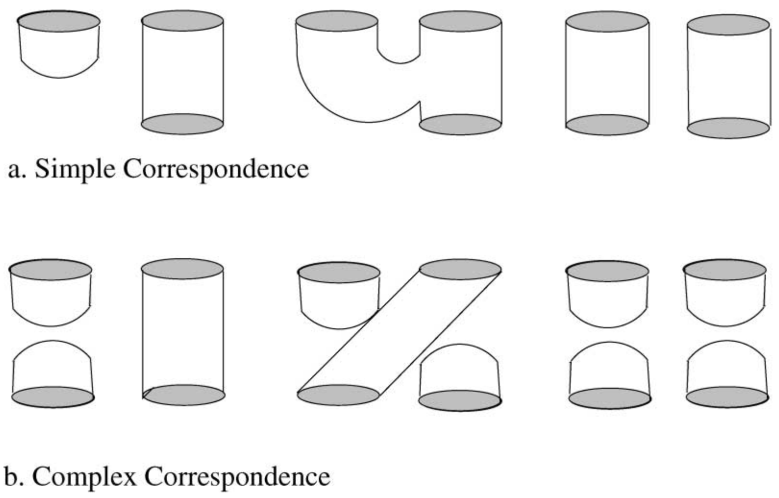


Fig. 2. Simple and complex correspondence.

## 2. Correspondence processes

### 2.1. Introduction

There have been a number of methods used for the resolution of the correspondence problem, which can be classified as follows:

1. *Manual methods*. Although in many cases a manual solution has been used to direct surface triangulation (for example, Christiansen and Sederberg, 1978; Fuchs et al., 1977; Ganapathy and Dennehy, 1982), this can be extremely time consuming for large data sets and those in which complex branches or loops may exist.
2. *Local algorithms*. The simplest form of automatic computer-based approach reduces the correspondence problem to consideration of individual pairs of sections in the object. It calculates connectivity between adjacent contours using a simple calculation, such as centroid distance or overlap. Examples of this approach include those of Ekoule et al. (1991) and Wang and Aggarwal (1985). Haig et al. (1991) combine centroid distance with topological information to confirm that all the possible connections are valid, before using contour overlap to define the correspondence.
3. *Global algorithms*. All of the possible connections between contours in an object are considered simultaneously when determining the correspondence solution, not just those lying between a single pair of sections. Global methods use a graph-based approach to calculate which edges should be included, with algorithms such as those proposed by Giertsen et al. (1990), Shinagawa and Kunii (1991), Meyers et al. (1992) and Herbert et al. (1995). There are a number of problems with this type of approach, and they are discussed below.
4. *Growing algorithms*. These adopt a hierarchical approach, attempting to locate the separate components of an object, rather than individual links between contours, before defining the overall structure of the correspondence solution. Previous solutions using this approach have been suggested by Soroka (1981) and Meyers et al. (1992), although these had a number of limitations, which have been overcome in the new implementation discussed in this paper.

The four types of approaches use a variety of information sources in compiling an underlying framework for the subsequent surface reconstruction process. Some of the information can be described as 'top–down', with high-level information being provided about the object. Manual methods rely completely on this type of information, which would be provided by the user. Local algorithms on the other hand use information from the geometry of the object directly and process it from the 'bottom-up'.

Both the global and growing algorithms carry out more information processing than the other two methods. Of these, the growing algorithm is flexible as it is possible to combine top-down and bottom-up information sources, therefore making this method potentially quite powerful.

### 2.2. Growing algorithms

Soroka (1981) proposed a growing algorithm, that constructed generalised cylinders based on elliptical cross-sections derived from the contour data. Meyers et al. (1992) used this method to build elliptical cylinders from the contours, before finding the connections between them to construct the final object. There were problems however with this style of growing algorithm, given below, which prompted them to adopt the global minimum spanning tree (MST) algorithm for the data they were reconstructing:

- *Contour ordering* — The results were dependent on the order in which the contours were specified and a contour was considered for inclusion in a cylinder on a first-come first-served basis.
- *No backtracking* — The lack of backtracking facilities meant that errors made because the wrong contour was chosen could not be corrected and that these were then propagated through the construction process.

However, the nature of the data reconstructed by Meyers et al. (1992) meant that the global algorithm limitations did not affect their results, as the shape being modelled was tree-like in form.

### 2.3. Problems with global algorithms

In a previous paper (Herbert and Tough, 1995) a new global graph-based correspondence algorithm was proposed. It is based on an MST derived from a candidate graph containing all possible edges between adjacent contours. The procedure reduces this to the final correspondence graph by selecting the edges with the lowest weight until all the contours in the graph have been included.

The algorithm proposed was distinguished from the earlier MST approach by enabling the reconstruction of those objects that were made of disjoint components. This was facilitated by determining the topological relationship between a pair of neighbouring contours based on a surroundness tree (Haig et al., 1991). In cases where a component is completely contained by another component, topologically invalid connections exist and they are removed from the graph (Fig. 3).

Although, this additional work enabled the reconstruction of a number of palaeontological specimens that contained disjoint components, there were still several fundamental problems:

1. *Cyclic features*. The algorithm was unable to find cyclic features located in the object, (such as loops or spirals), because the final graph must be a tree or a number of disjoint trees. These types of features can be found in many sectioned invertebrate palaeontological specimens. The proposed post-processing with the MST algorithm was limited in this respect, as it could only remove edges from the graph and not add them.
2. *Algorithm dependent*. The new MST algorithm used additional topological information about the relationship between pairs of neighbouring contours, but the shape of the final correspondence graph was governed by the MST process. The use of this form of geometric algorithm will always limit the type of object that can be determined, even after post-processing.
3. *Local decision-making*. The MST algorithm proceeds serially, picking the next most suitable edge from the complete set of possible connections between adjacent contours, using a weight value determined by an edge calculation. This process is described as global, as it considers the complete set of edges in the graph, but the choice of edge is still based only on information calculated between a pair of adjacent sections and therefore has little advantage over the local algorithms.

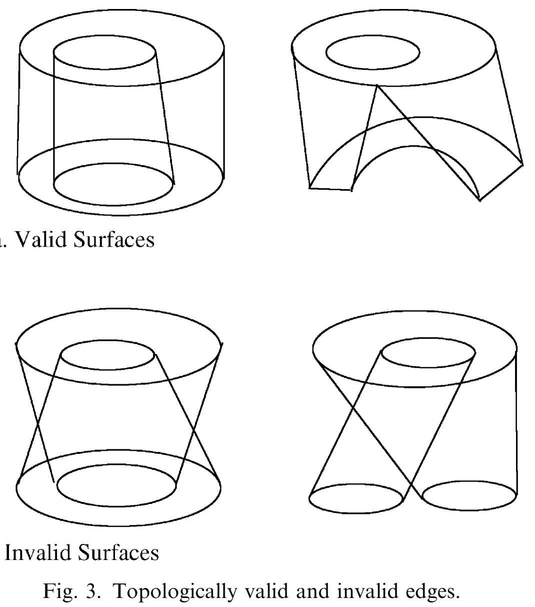


Fig. 3. Topologically valid and invalid edges.

These weaknesses in the global type of algorithm prompted an investigation into another form of algorithm, one that grows individual components of an object, before joining them together to provide the underlying skeleton of the object under reconstruction.

## 3. CorresGrow algorithm

CorresGrow is a growing algorithm which constructs each component in an object separately, before determining the branches and links between them to give a final correspondence graph. The algorithm is made up of three stages: pre-processing, component growing and branch handling (Fig. 4).

The pre-processing stage constructs the candidate graph, which contains all possible connections between contours in adjacent sections with a variety of weights calculated using several metrics attached to each connection. The information contained within the candidate graph is then used for calculating a correspondence solution. The information consists of two types:

1. Spatial information is calculated directly from the three-dimensional contours in each section. This is the main source of information used for finding each of the individual components and is processed 'bottom-up'.
2. Semantic information is provided by the user as a set of high level component descriptions. It does not specify individual connections between pairs of contours, but is an ordered list of labelled components that should exist in the object being reconstructed. As well as the label, each component also has a series of parameters which describe both its shape and its relationship to other components.

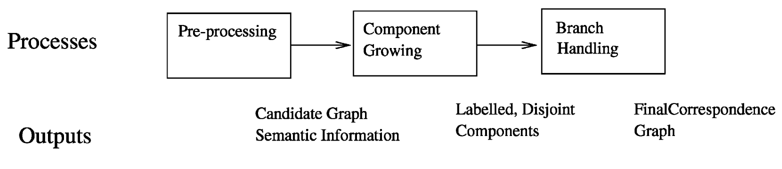

Processes
Pre-processing
Component Growing
Branch Handling
Outputs
Candidate Graph Semantic Information
Labelled, Disjoint Components
FinalCorrespondence Graph

Fig. 4. CorresGrow processes.

Both types of information are used to calculate the weights for each contour-to-contour connection, and these are discussed further below.

Component growing is the main processing stage of the algorithm, where a series of rules locate and grow each of the listed components through the candidate graph. The final branch handling stage constructs the links between the different components grown during the previous stage, primarily using the semantic information provided by the user. The use of both spatial and semantic information means that the growing algorithm can be less reliant on the methods used to construct the correspondence graph.

### 3.1. Pre-processing

The pre-processing stage accumulates the information to be used during the component growing and branch handling stages. This involves the derivation of both the spatial and semantic information (Fig. 5), from the raw contour data and from user input, respectively. Finally, before the growing process is started the topologically invalid edges are identified and removed from the candidate graph.

### 3.2. Spatial information

The spatial information is available at three levels and is used to assign weight values between each pair of contours on adjacent sections and also to remove edges that exist between unmateable contours:

1. *Contour characteristics* — These describe each contour's position, shape and size. This is then used to build the other types of spatial relationship information.
2. *Intra-sectional relationships* — They describe the correlation between contours on the same section and are then used to determine the topological validity of connections between contours on adjacent sections.
3. *Inter-sectional relationships* — These describe the connectivity between contours on adjacent sections. This information is used directly at the growing stage to determine which contours should be in each component.

The CorresGrow algorithm uses five contour characteristics: the centroid; area; major/minor axes; minimum bounding rectangle (MBR) and the compactness ratio. The centroid and major/minor axes for each contour are calculated using the algorithm proposed by Tough (1988).

The compactness ratio can be used as a measure of shape, by comparing the contour's area with the area of a circle having the same perimeter (Unwin, 1981):

$$CompRatio = \sqrt{\left(\frac{a}{a_c}\right)},$$

where $a$ is the area of the contour and $a_c$ the area of the circle with the same perimeter as the contour.

The contour characteristics are used to identify the relationships between adjacent contours in the same section and in adjacent sections. A number of *relationship measures* are used for each possible connection between two contours and these give a comprehensive set of values that can then be used by the component growing stage. These values can be of three types:

1. *Quantitative* — A numerical value corresponding to a measurement or to some function of measurements.
2. *Rank order* — A single integer value derived from the comparison of all the quantitative values calculated using each of the relationship measures, where they all emanate from the same contour. Thus for any contour in a section, the connections to contours in the adjacent section can be placed in order of preference, with a rank order value of 1 being given the most suitable.
3. *Boolean* — Some relationship measures, such as the contains measure described below, will return a true or false value rather than a quantitative measure.

There are three relationship measures used to calculate a value for intra-sectional relationships:

1. *Position*: This specifies the distance between a pair of contours based on the centroids. It also calculates approximate directional relationships according to

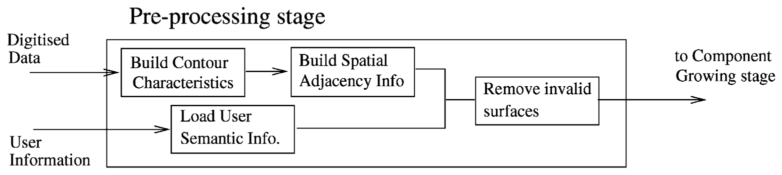


Fig. 5. CorresGrow pre-processing.

whether a contour is above, below, left or right of another contour on the section. It therefore returns both a set of quantitative and a set of Boolean values.
2. *Size*: Provides a quantitative value, expressing the difference in size between a pair of contours.
3. *Contains*: Provides a Boolean value, stating whether a contour is contained inside another contour on the section. This is used subsequently to determine the topology between contours in adjacent sections.

There are 4 relationship measures used to calculate edge values between contours on adjacent sections and they are used during the component growing stage to determine which contours occur in each component:

1. *Distance/axes combination*: This provides an indication of both change in position on the section and change in shape:
$$e(i,j) = (x_i - x_j)^2 + (y_i - y_j)^2 + (A_i - A_j)^2 + (B_i - B_j)^2,$$
where $e(i,j)$ is the edge between two contours $i$ and $j$ and $(x, y)$ is the centroid and $(A, B)$ are the major and minor axes.
2. *Minimum bounding rectangle (MBR) overlap*: This relationship measure uses the change in contour shape and position. The amount of overlap can be expressed as a value between 0 and 1. It is 0 when there is no overlap, and 1 when the contour's MBR is completely enclosed by the MBR of the other contour (see Fig. 6).
3. *Shape comparison*: This relationship measure provides an indication of change in shape between a pair of contours, based on their compactness ratios.
4. *Topological validity*: This is calculated by processing the surroundness tree for each of the neighbouring sections, using the contains intra-sectional relationship. If a contour is contained by another contour, then it is at a different, lower surroundness level to that contour. By comparing the levels between contours on adjacent sections, the validity of any connection between a pair of contours can be evaluated. Topological validity needs to be tested for a combination of edges rather than just singly (Fig. 7).

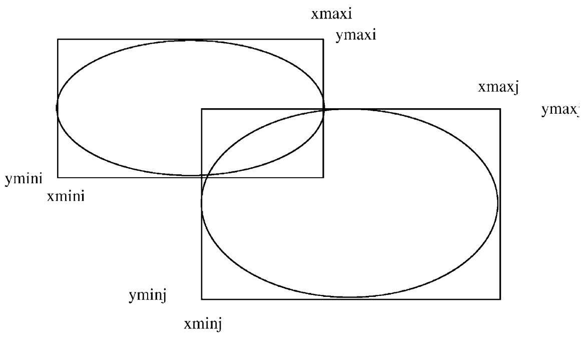


Fig. 6. MBR overlap.

All three of the non-topological measures can be combined and this reduces any weaknesses that a single measure may have when matching a particular contour combination. A candidate correspondence graph is constructed from the Inter-sectional relationship measures and includes all edges between contours on adjacent sections that have a Boolean value of true for the topological validity measure. All of these edges are then assigned the values from the other measures and are now ready to be used for component growing. By having a number of edge relationship measures it is possible to try different combinations and investigate their effect upon the shape of the correspondence graph.

### 3.3. Semantic information

In general terms, semantic information describes relationships between the major features of the object under construction. In the specific case of the palaeontological reconstructions discussed later in this paper, the semantic information is provided as a set of descriptions of the major components of the specimen.

This is currently in the form of a text file, which contains a number of component labels, each of which has a number of attributes. Each of the attributes help determine the position of the component in the object and its relationship to other components in the object. For example, the following fragment determines semantic information for the Pedicle valve component in a Brachiopod object:

```
@components = new object()

$components[0] -> label = PEDICLE;
$components[0] -> range = ALL;
$components[0] -> inside[0] = PEDICLE_INTERNAL;
```

The label attribute provides the reference by which the CorresGrow process can locate the semantic information. Other attributes provide information about the range of sections in which the component can appear, in this case a PEDICLE component must appear in all of the sections through the object.

The semantic information is used to drive the component growing process by providing the set of labelled components. It is also the main source of

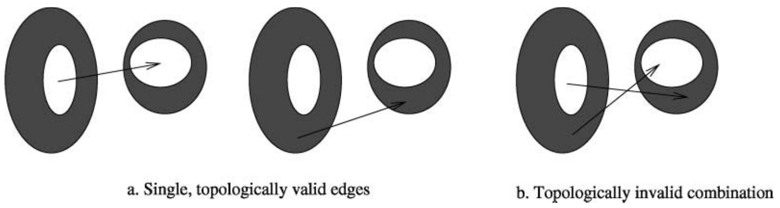

a. Single, topologically valid edges
b. Topologically invalid combination

Fig. 7. Invalid topological combination.

information in determining the actions that take place during branch handling stage (see below).

### 3.4. Component growing

The component growing stage of the correspondence algorithm takes each labelled component provided by the user as semantic information and grows this through the candidate graph.

Component growing is carried out using a four-rule system (Fig. 8). Two of the rules are used to grow each component through the sections and two are for conflict cases where more than one component is a candidate to use the same contour:

1. *Start* — Finds the first contour for each component in the object.
2. *Start-conflict* — Decides which component should use a disputed contour located by the start rule.
3. *Extend* — Allows the current component to grow to a contour in the next section, until either the component terminates or the last section in the object is reached.
4. *Extend conflict* — This is used by the extend rule, when the chosen contour in the next section is already occupied by another component.

#### 3.4.1. The start rule

The start rule will find the first contour for each component in the list provided by the user. It does this by examining each contour and deciding if it fits the semantic characteristics for the labelled component. Unlike previous algorithms this does not use a 'first-come, first-served' approach and the start rule will allow the current component to consider both free and already occupied contours. The fragment of code given below is based on the Perl source code, as are the other examples.

```
    sub start()
    {

        foreach $component (@components) {
            foreach $contour (@contours) {
        if ($contour->semantic =~ $component->semantic) { # check semantic
                if($contour->$component ! = NULL) { # contour not in a component
                    $component->first_contour = $contour; # add contour to component
                    &extend($component); # call extend rule
                }
                else {
                    &start_conflict($component); # already in a component
                }
}
        }
    }

} # end start
```

The current component keeps a record of the current metric value of the edges that are being used to extend the component through to the next section. If the contour is ‘free’, then the extend rule is used to commence the construction of the component, but if the contour is included within another contour, then the start-conflict rule allows a comparison with the component already using that contour.

```
sub start_conflict {
{
      my ($new_component)=@_;

      $contour = $component=>current_contour; # get current contour
      $old_component=$contour->component; # assign old component
      if ($old_component->label⟨⟩ $new_component->label) {
# only if not same semantic label
          $contour->component=NULL; # remove existing component
          $new_component->edge_rank=0;
          &extend($new_component); # extend the new component
          if ((($new_component->AEW < $old_component->AEW) ||
             (($new_component->AEW==$old_component->AEW) &&
             ($new_component->length > $old_component->length))) {
            $contour->component=new_component; # if new component is better fit
            }
          else {
            $contour->component=$old_component; # if not, keep old component
          }
      }

} # end start_conflict
```

### 3.4.2. The start-conflict rule

The start-conflict rule decides which of the two components should be able to use the contour by comparing the average edge weight (AEW) values when each of the components are fully grown (see Fig. 9). These values are calculated by determining the mean of the rank order values for each edge of a component. First, the existing component is removed from the candidate correspondence graph, so that it does not interfere with the growth of the new component. The new component is then extended from the now temporarily free start contour, with the use of the extend and extend-conflict rules (see below). The AEW of the component is then compared with the AEW of the removed existing component.

The component with the lowest AEW is deemed the most suitable and gains the right to use the contour. The rejected component is passed back to the start rule and the process recommences at the next contour in the object. If the existing component is the most suitable, the changes made during the extensions and conflicts of the new component are reversed by copying the candidate correspondence graph back as it was before the comparison.

This challenge to existing components by a potential new component has to be between those components that do not have the same label, otherwise components with the same structure would be produced by both new and old components, because they have the same semantic information.

If components have the same AEW, then the longest component is included in the graph, but if they have the same length, then the existing component stays in place. The combination of start and start-conflict rules mean that a labelled component may be superseded and have to be regrown at any time during the component growing process.

### 3.4.3. The extend rule

The extend rule allows the component to ‘grow’ into the next section, by selecting the best suited contour for inclusion. The rule uses the edge that has the lowest edge weight (based on the rank order value), if the connecting contour’s characteristics match the descriptive parameters in the current component’s label. If the contour is already included in another component then the extend-conflict rule is used to decide which of the two components should use the contour. Once the current component has successfully grown into the next section, the extend rule will then look at contours in the subsequent section. If no suitable contour is found for

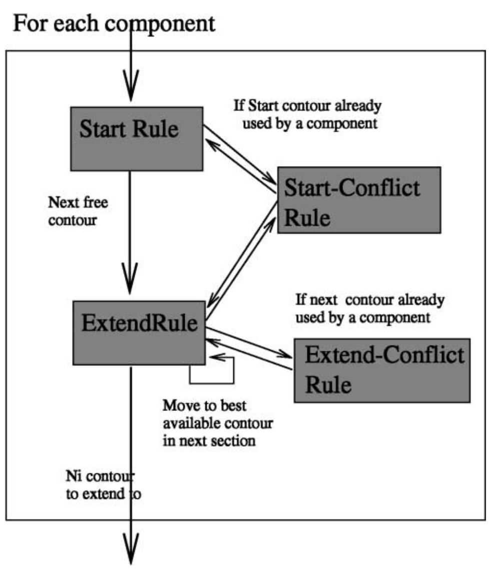

For each component
Start Rule
If Start contour already used by a component
Start-Conflict Rule
Next free contour
ExtendRule
If next contour already used by a component
Extend-Conflict Rule
Move to best available contour in next section
Ni contour to extend to

Fig. 8. CorresGrow growing process.

```
sub extend()
{
   my ($component) = @_;

   $component->edge_rank++; # increment the edge rank
   $contour = $component->edge_rank->contour; # move to next section
   if (($component->edge_rank < SEARCH_LIMIT)
      && ($contour->section = ~ LAST_SECTION)) {
     if ($contour->component ! = NULL) {
      &extend_conflict($contour,$component) # contour already used
      } else {
        $component->last_contour = $contour; # contour free and
        $component->edge_rank = 0; # and added to component
        &extend($component); # try to extend again
     }
   }
} # end extend
```

the current component, then the growing process is stopped for this component and then commenced for the next labelled component (see Fig. 10).

As the extend-conflict rule may reject a connection to the most suitable contour in the next section because another component has a better claim to it already, the next best alternative may have to be considered. It is possible to envisage, in the worst case, that the least suitable contour for the current component may be chosen by the extend rule, where instead the component should have been terminated in the preceding section.

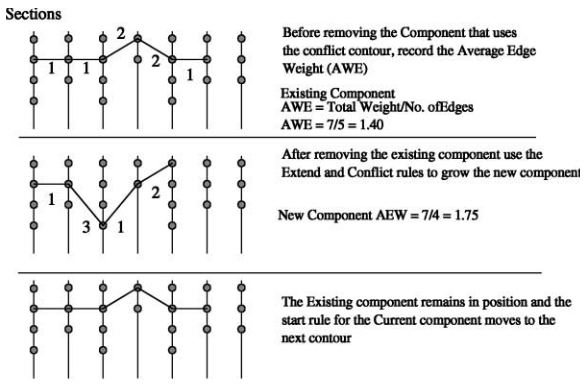

Sections
1
1
2
2
1
Before removing the Component that uses the conflict contour, record the Average Edge Weight (AWE)
Existing Component
AWE = Total Weight/No. ofEdges
AWE = 7/5 = 1.40
After removing the existing component use the Extend and Conflict rules to grow the new component
1
3
1
2
New Component AEW = 7/4 = 1.75
The Existing component remains in position and the start rule for the Current component moves to the next contour

Fig. 9. CorresGrow start-conflict rule.

To prevent this type of connection from being made, only the more suitable possibilities are considered by the growing process (the edges with the best rank order values) and, if these are better placed in other components, the current component is terminated.

### *3.4.4. The extend-conflict rule*

The extend-conflict rule is called by the extend rule when a component attempts to include a contour that is already present in another component. The rule will decide which one of the two components should contain the contour and then determine how each component should be grown from this point. If the current component gains control of the contour, the other component will have to recommence the growing process from the contour it controls in the preceding section. If the current component is unsuccessful, then it will have to attempt to use the next most suitable contour in the same section as the conflict contour.

```
sub extend_conflict() {

   $contour = @_[0];
   $new_component = @_[1];

   $old_component = $contour->component; # assign existing component
   if ($old_component->previous_contour->edge_weight < =
      $new_component->contour->edge_weight ) { # existing component uses
      $new_component->edge_rank++;      # contour
      $contour = $component->previous_contour;
   }
   else { # new component has contour
      $old_component->last_contour = $old_component->previous_contour;
      &extend ($old_component); # try to extend existing component
      $contour->component = $new_component; # to another contour on section
      $new_component->edge_rank = 0;
   }

} # end extend_conict
```

### 3.5. Branch handling

The branch handling stage of the CorresGrow algorithm uses two relative parameters in the component semantic information to determine the connectivity (or lack of it) between the components. The types of relationship are currently limited to two, namely those of adjacent and inside. Adjacent components branch in the traditional way, either by diverging or merging. Inside components exist solely inside another component and can only be connected if the internal component is joined to the outer component (see Fig. 11).

An adjacent relationship results in joining the start contour of the second component with the first component's contour in the preceding section. The Inside relationship currently handles only those contours that 'emerge' from other components and results in connecting the last contour of the second component with the first component's contour in the following section.

It may be noted that components in the semantic information that are not intended to be connected do not have any relationship information, and are hence not affected by the branch handling stage.

### 3.6. Completion of growing process

The growing process is completed when the complete list of components has been traversed, even if there are some contours not included in any component. This allows the user to define a partial reconstruction that includes only the major components. It is possible for

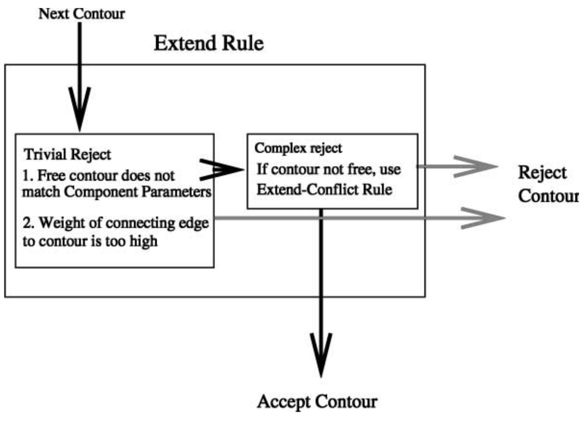

Next Contour
Extend Rule
Trivial Reject
1. Free contour does not match Component Parameters
2. Weight of connecting edge to contour is too high
Complex reject
If contour not free, use Extend-Conflict Rule
Reject Contour
Accept Contour

Fig. 10. CorresGrow extend rule.

the user to create a generic label, which contains no descriptive parameters and for this to be added to the bottom of the list of labels. It will ensure that all the contours will appear in the final correspondence graph, even if this is in an unrecognised component with no semantic information.

## 4. Palaeontological reconstructions

### 4.1. Introduction

The correspondence of a number of palaeontological specimens has been determined using the CorresGrow

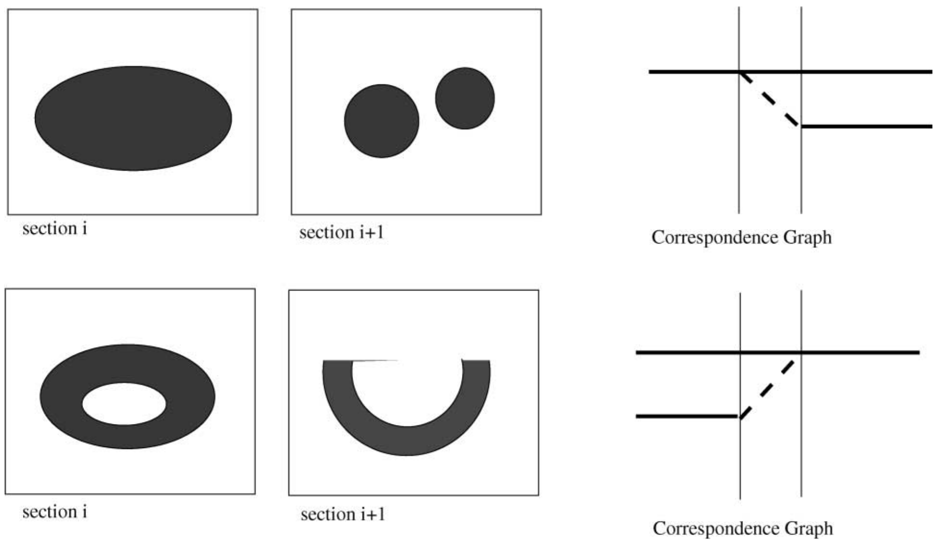


Fig. 11. Branch handling scenarios for CorresGrow.

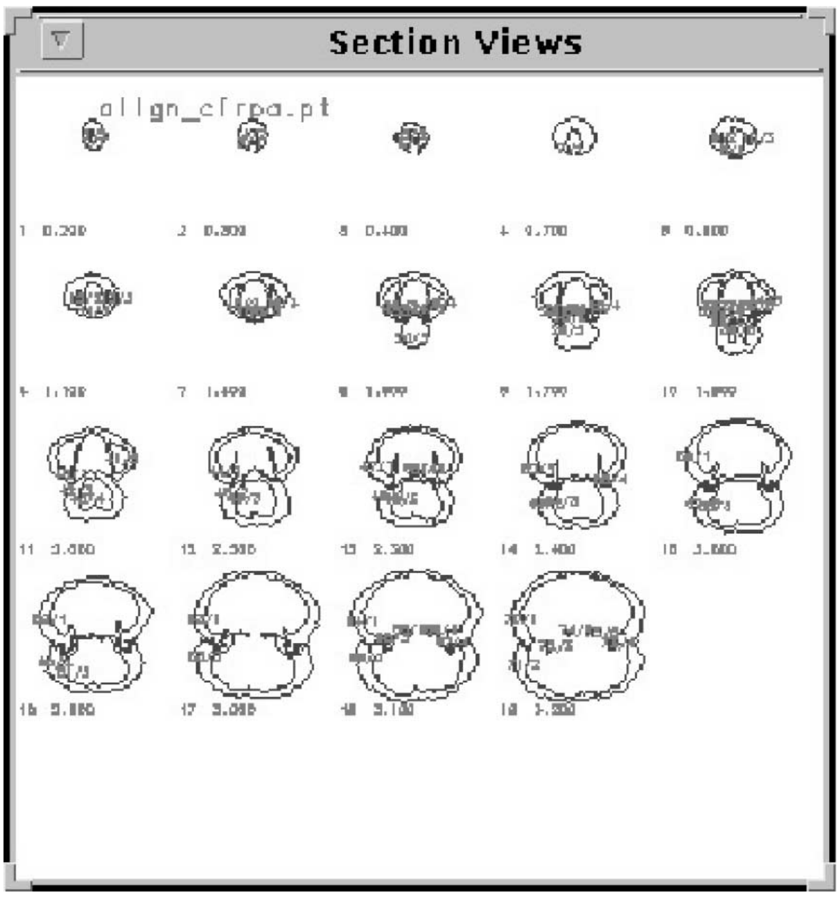


Fig. 12. Digitised sections of *Cirpa langi*.

algorithm. These have been mainly brachiopod specimens, but has also included graptolites, corals and other destructively sampled specimens. Most of the specimens have been digitised from their section drawings or sectional photographs.

The semantic information can be easily altered for different specimens as it is held in a text file that is independent of the CorresGrow algorithm itself. In the future it will be possible to construct a library of different component descriptions that can be used and tested for different reconstructions.

The correspondence of the specimen presented below was reconstructed using both CorresMST, a global-based algorithm (Herbert et al., 1995), and CorresGrow. Briefly, CorresMST does not use any semantic information, but does use the same pre-processing stage as CorresGrow, that builds the candidate graph and then removes the topologically invalid edges.

### 4.2. *Cirpa langi*

This specimen, *Cirpa langi*, a rhynconellid brachiopod was digitised from a set of published section drawings (Ager, 1956). It is recorded as a set of 75 contours in 19 sections (see Fig. 12).

The CorresMST algorithm uses the minimum spanning tree algorithm (Corman et al., 1990) to reduce the candidate graph after all topologically invalid edges have been removed. This still leaves a single tree-like graph (both the solid and dashed edges in Fig. 13), which needs post-processing to remove edges that connect disjoint components. The algorithm does locate the major components (the pedicle valve contours 1–70 for example), but it is unable to remove some of these incorrect edges (for example contours 50–52). Also correct branch edges, that link different components, were removed during the MST process (e.g., the component that uses 63–66) and these cannot then be re-inserted during the post-processing.

The CorresGrow algorithm differs mainly from the CorresMST algorithm in that it exploits semantic information in order to grow the components in the order shown in the table of Fig. 14. The performance scores in Fig. 14 show that the growing stage of the algorithm attempted to start a component 837 times and that 759 extensions were attempted.

During the growing stage the brachial component initially starts at contour 17, but it is later displaced by the first adjacent pedicle component and is subsequently started at contour 20, its correct location. The AEW score of 1.383 shows that a number of edges with a rank order value of more than 1 were used in Fig. 14. An edge that does not have a rank order value of 1 has been selected after a conflict where the semantic information has restricted the choice of contours for a component. Objects with an overall AEW of near to 1 have not had

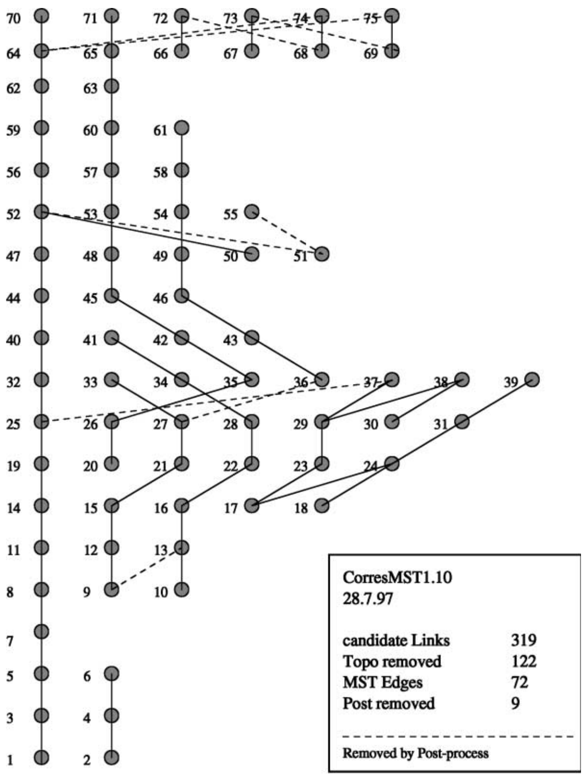


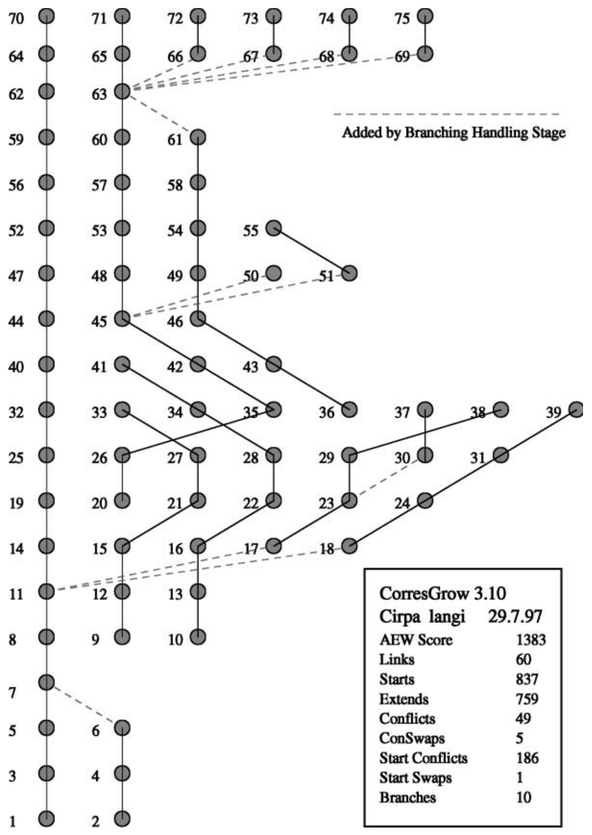
Fig. 13. CorresMST correspondence graph for *Cirpa langi*.

to rely heavily on the semantic information during the component growing, but will still use it for branch handling. A low AEW will also indicate a small number of start and extend conflicts.

The correspondence solution was then used to guide the surface triangulation algorithm in the construction of the three-dimensional models (see Figs. 15 and 16) (Herbert and Tough, 1995). As well as allowing interactive inspection of three-dimensional images, the model could also be used for morphometric analysis (Slice, 1993) or evolutionary simulations.

## 5. Conclusions

CorresGrow can successfully reconstruct sectioned palaeontological objects (Figs. 15 and 16) containing disjoint components, using both spatial information from the contour data and semantic information provided by the user. This is in contrast to a global MST algorithm, which can find the major components, but cannot always determine correctly the connectivity between them. There are a number of features in the CorresGrow algorithm which render it advantageous for reconstructing palaeontological specimens and other earth science data.



CorresGrow 3.10

Cirpa langi 29.7.97

| | |
|---|---|
| AEW Score | 1383 |
| Links | 60 |
| Starts | 837 |
| Extends | 759 |
| Conflicts | 49 |
| ConSwaps | 5 |
| Start Conflicts | 186 |
| Start Swaps | 1 |
| Branches | 10 |

Fig. 14. CorresGrow result for *Cirpa langi*.

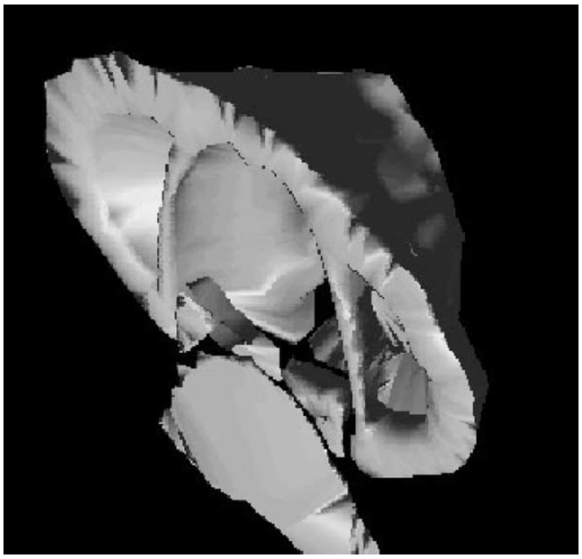


Fig. 15. Reconstruction of *Cirpa langi*.

1. *Backtracking* — No component is fixed in position until the growing stage is completed. The order in which contours are grown is only significant when component descriptors are relative to other contours in already grown components. The conflict rules allow previous decisions made by the growing process to be challenged and changed.
2. *Semantic information* — This is user input information at a high-level and does not define individual connections between contours. It can be easily changed, is entered only once and can be used with a number of different algorithms. It is used during both the component growing and branch handling stages and is the sole source of information in the latter stage.
3. *Information driven* — The shape of the correspondence graph is governed more by the information provided than by the nature of algorithm used. For example, the MST algorithm will always produce a tree-like structure made up of a single component, whereas the result of a growing algorithm is dependent only on the spatial and semantic information provided.

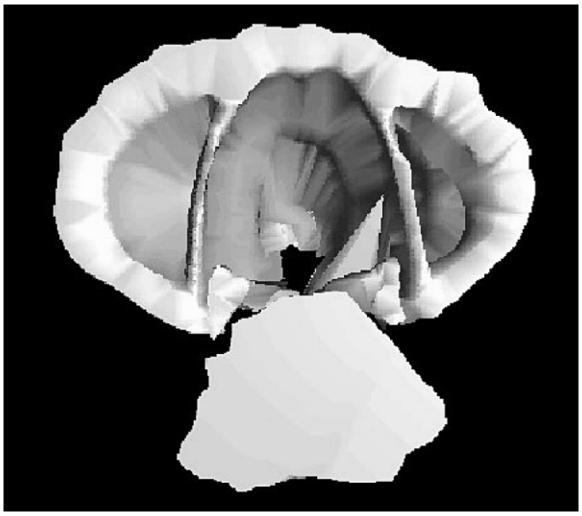


Fig. 16. Reconstruction of *Cirpa langi*.

Future work with the CorresGrow algorithm will concentrate on combining the component growing and branch handling stages. Rather than just constructing single non-branching components, the growing process will be able to identify branch points by examining spatial and semantic information and then grow two or more components from that point. Also to help with the location of cyclic or spiral features, the growing process could be extended to allow components to be extended both across sections and also back through sections which they have already passed.

Work continues with expanding the range of semantic component labels, so that the reconstruction process can tackle a wider range of objects. The CorresGrow algorithm has also been tested on data without the use of semantic information and this can identify the major components which in turn guides the user when defining the component semantic information.

## Acknowledgements

The authors would like to acknowledge the collaboration on this work by Prof. Michael Bassett, Dept. of Geology, National Museum of Wales, Cardiff. Malcolm Herbert was supported by a Studentship from the Engineering and Physical Science Research Council.

## References

Ager, D.V., 1956. A Monograph of the British Liassic Rhynchonellidae. The Palaeontological Society, London, 172pp.

Boissonnat, J.D., 1988. Shape reconstruction from planar cross sections. Computer Vision, Graphics and Image Processing 44, 1–29.

Christiansen, H.N., Sederberg, T.W., 1978. Conversion of complex contour line definitions into polygonal element mosiacs. ACM Computer Graphics 12 (3), 187–192.

Corman, T.H., Leisorsen, C.E., Rivest, R.L., 1990. Introduction to Algorithms. MIT Press, Boston, MA, 563pp.

Ekoule, A.B., Peyrin, F.C., Odet, L., 1991. A triangulation algorithm from arbitrary shaped multiple planar contours. ACM Transactions on Graphics 10 (2), 182–199.

Elvins, T.T., 1992. A survey of algorithms for volume visualisation. Computer Graphics 26 (3), 194–201.

Fuchs, H., Kedem, Z.M., Uselton, S.P., 1977. Optimal surface reconstruction from planar contours. Communications of the ACM 20 (10), 693–702.

Ganapathy, S., Dennehy, T.G., 1982. A new general triangulation method for planar contours. ACM Computer Graphics 16, 69–75.

Geiger, B., 1993. Three-dimensional modelling of human organs and its application to diagnosis and surgical planning. Technical Report 2105, INRIA BP93, 06902 Sophia Antipolis, France, 124pp.

Giertsen, C., Halvorsen, A., Flood, P.R., 1990. Graph-directed modelling from serial sections. The Visual Computer 6, 284–290.

Haig, T.D., Attikiouzel, Y., Alder, M., 1991. Border marriage: matching of contours of serial sections. IEE Proceedings — I 38 (5), 371–376.

Herbert, M.J., Jones, C.B., Tudhope, D.S., 1995. Serial section reconstruction of geoscientific data. The Visual Computer 11 (7), 343–359.

Herbert, M.J., Tough, J.G., 1995. Surface reconstruction from complex earth science data. In Proceedings of 13th Eurographics, Loughborough University, UK, pp. 85–106.

Herman, G.T., Liu, H.K., 1979. Three-dimensional display of human organs from computer tomograms. Computer Graphics and Image Processing 9, 1–21.

Jones, M.W., Chen, M., 1994. A new approach to the construction of surfaces from contour data. Computer Graphics Forum 13 (3), 75–84.

Keppel, E., 1975. Approximations of complex surfaces by triangulation of contour lines. IBM Journal of Research and Development 19, 2–11.

Lorensen, W.E., Cline, H.E., 1987. Marching cubes: a high resolution 3D surface construction algorithm. ACM Computer Graphics 21 (4), 163–169.

Meyers, D., 1994. Multiresolution tiling. Computer Graphics Forum 13 (5), 325–340.

Meyers, D., Skinner, S., Sloan, K.R., 1992. Surfaces from contours. ACM Transaction on Graphics 11 (3), 228–258.

Rhodes, M., 1991. Computer graphics in medicine: the past decade. IEEE Computer Graphics and Applications 11 (1), 52–54.

Shinagawa, Y., Kunii, T.L., 1991. Constructing a reeb graph automatically from cross sections. IEEE Computer Graphics and Applications 11 (11), 44–51.

Slice, D., 1993. GRF-ND: Generalised rotation fitting of *n*-dimensional landmark data. Technical Report. Dept. of Ecology and Evolution, State University of New York, Stony Brook, NY, 17pp.

Soroka, B.I., 1981. Generalised cones from serial sections. Computer Graphics and Image Processing 15, 154–166.

Tipper, J.C., 1976. The study of geological objects in three dimensions by the computerised reconstruction of serial sections. Journal of Geology 84, 476–484.

Tipper, J.C., 1977. A method and FORTRAN program for the computerised reconstruction of three dimensional objects from serial sections. Computers & Geosciences 3, 579–599.

Tough, J.G., 1988. The computation of the area, centroid and principle axes of a polygon. Computers & Geosciences 14 (5), 715–717.

Udapa, J.K., Herman, G.T., 1989. 3D imaging in medicine. CRC Press, Boston, 389pp.

Unwin, D., 1981. Introductory spatial analysis. Methuen, New York, 212pp.

Wang, Y.F., Aggarwal, J.K., 1985. Construction of surface representation for 3-D volumetric scene description. Computer Vision and Pattern Recognition, San Francisco, pp. 130–135.