

Washington University in St. Louis

## Washington University Open Scholarship

---

All Computer Science and Engineering  
Research

Computer Science and Engineering

---

Report Number: WUCS-82-1

1982-01-01

### Octal-Tree Spatial Sorting and its Applications

Jeffrey L. Posdamer

An octal tree subdivision recursively divides a bounded three-dimensional volume into octanta about an internal division point. This scheme has been used to represent cellular or enumerated voxel models of solid objects. Given one or more sets of points sampled from the surface of a solid, an octal tree may be generated in which each leaf node contains  $m$  or less points. By specifying the tree traversal rule, the points are accessed in a sorted order. By defining  $m=3$ , a divide-and-conquer surface triangulation algorithm may be developed which does not require special sampling conditions (such as co-planarity) on subsets... **Read complete abstract on page 2.**

Follow this and additional works at: [https://openscholarship.wustl.edu/cse\\_research](https://openscholarship.wustl.edu/cse_research)



Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

---

#### Recommended Citation

Posdamer, Jeffrey L., "Octal-Tree Spatial Sorting and its Applications" Report Number: WUCS-82-1 (1982). *All Computer Science and Engineering Research*.  
[https://openscholarship.wustl.edu/cse\\_research/888](https://openscholarship.wustl.edu/cse_research/888)

Department of Computer Science & Engineering - Washington University in St. Louis  
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160.

## Octal-Tree Spatial Sorting and its Applications

Jeffrey L. Posdamer

### Complete Abstract:

An octal tree subdivision recursively divides a bounded three-dimensional volume into octanta about an internal division point. This scheme has been used to represent cellular or enumerated voxel models of solid objects. Given one or more sets of points sampled from the surface of a solid, an octal tree may be generated in which each leaf node contains  $m$  or less points. By specifying the tree traversal rule, the points are accessed in a sorted order. By defining  $m=3$ , a divide-and-conquer surface triangulation algorithm may be developed which does not require special sampling conditions (such as co-planarity) on subsets of the sample points. By element(octal) a pre-ordering is established on the faces. From any viewpoint, surface polygons can be visited in a priority ordered fashion by appropriate tree traversal. The pre-ordering established is shown to be useful in several graphics related contexts. The ability to preprocess the faces so as to establish a viewpoint independent data structure for priority sorting leads to a useful hidden surface techniques. By appropriate modification of the traversal rule a ray-tracing algorithms may visit only faces lying on the ray in a priority order. The tree traversal visits faces in a fashion which causes reference to regions in a frame buffer to be geometrically localized, substantiating reducing page faults in a virtual buffer. The sorting process allows for simple geometric merging of sets of surface measurements which contain no fiducial information.

OCTAL-TREE SPATIAL SORTING  
AND ITS APPLICATIONS

Jeffrey L. Posdamer

WUCS-82-1

Department of Computer Science

Washington University

St. Louis, Missouri 63130

January 1982

# Octal-tree Spatial Sorting and its Applications

Jeffrey L. Posdamer  
Department of Computer Science  
Washington University  
Saint Louis, MO 63130

## Abstract

An octal tree subdivision recursively divides a bounded three-dimensional volume into octants about an internal division point. This scheme has been used to represent cellular or enumerated voxel models of solid objects. Given one or more sets of points sampled from the surface of a solid, an octal tree may be generated in which each leaf node contains  $m$  or less points. By specifying the tree traversal rule, the points are accessed in a sorted order

By defining  $m=3$ , a divide-and-conquer surface triangulation algorithm may be developed which does not require special sampling conditions (such as co-planarity) on subsets of the sample points. By assigning every polygon on a faceted surface to its containing octal element (octel) a pre-ordering is established on the faces. From any viewpoint, surface polygons can be visited in a priority ordered fashion by appropriate tree traversal.

The pre-ordering established is shown to be useful in several graphics related contexts. The ability to preprocess the faces so as to establish a viewpoint independent data structure for priority sorting leads to a useful hidden surface technique. By appropriate modification of the traversal rule, a ray-tracing algorithm may visit only faces lying on the ray in a priority order. The tree traversal visits faces in a fashion which causes reference to regions in a frame buffer to be geometrically localized, substantially reducing page faults in a virtual buffer. The sorting process allows for simple geometric merging of sets of surface measurements which contain no fiducial information.

# INTRODUCTION

## 1.0 INTRODUCTION

Recent work in the area of structured light surface digitizing techniques[1] has created the necessity to merge several sets of points measured on a single surface. Such point sets cannot contain fiducial information for registration purposes. In unidimensional problems, merging is intimately associated with sorting. Unfortunately, extensions of sorting to three-dimensions are not as direct as might be desired. Simultaneous with the need to solve this multi-dimensional merging problem, work in the use of octal tree techniques for representing three-dimensional solids became known[2]. This serendipitous coincidence resulted in the development of octal tree sorting (OTS) and its applications.

### 1.1 Definition Of Sorting

Sorting is a process to which a substantial body of literature has been devoted. The process of sorting may be defined as follows:

Given a set of elements

$$E_1, E_2, \dots, E_n$$

permute the list resulting in

$$1E_i, 2E_j, \dots, nE_k$$

where prefixes indicate position in the sorted list and suffixes position in the original unsorted list (suffixes will be omitted where they are unnecessary). In the sorted list a linear ordering relation  $R$  holds such that

for all  $i < j$ ,  $(i \in R j)$  is true.

A linear ordering relationship  $R$  has the properties that:

- a)  $(a R a)$  is true ( $R$  is reflexive)
- b) if  $(a R b)$  and  $(b R c)$  then  $(a R c)$  is true ( $R$  is transitive)
- c)  $(a R b)$  and  $(b R a)$  imply that  $(a = b)$  ( $R$  is antisymmetric)

While alternative definitions abound, the above is the operative definition. As noted above, extensive discussions of sorting are available most notably Knuth[6]. General sorting algorithms fall into several major categories including insertion, exchange, selection, merging and distribution.

#### 1.1.1 Tree-sorts -

The sort in one variable or linear sort of most relevance to octal tree sorting is Quicksort[15]. Given a list of elements to sort

$E_1, E_2, \dots, E_n$

choose an arbitrary element  $E_d$  as the division element. Create three outputs from an invocation of the algorithm: a list of elements with key less than  $E_d$ ;  $E_d$ ; a list of elements with key greater than  $E_d$ . Clearly the first and last list may be empty. For non-empty result lists, recursively apply the algorithm. Recursive invocation is halted by empty or single element output lists. The result may be organized as a tree with each choice of  $E_d$  on a node and the sublists on the left and right descendants. An in-order traversal of the tree will generate the elements in ascending sorted order. If the value of  $E_d$  is calculated, e.g. median, mean element value, midpoint of element range, etc. then the nodes will contain division values but all elements of the original

list will be distributed to the leaves of the tree.

## 1.1.2 Linear Sorting In 2-dimensions -

Linear sorting implies the use of a single key per record as the sort argument. Extension of this notion to two-dimensional data has been accomplished by suppression of one of the dimensions. A number of algorithms, especially those used for convex hull and related algorithms in 2-d, convert points to polar coordinates and sort on polar angle. Another application of linear sorting in 2-d is the treatment of a single coordinate at a time. Several rectangle measurement and nearest neighbor algorithms use a sort on a single coordinate (e.g. the x-coordinate) to break up space into strips of all coordinates lying between consecutive x values. Regions within each such strip are then ordered on y coordinates of the data lying in the strip. Unfortunately these techniques, especially polar angle sort, do not extend in an obvious way to higher dimensions.

## 1.1.3 Sorting In Geometric Complexity -

The application of the techniques of algorithm analysis to geometric algorithms has been recent. The relationship of sorting to this work is twofold: the mapping of algorithms onto sorting to demonstrate bounds and the use of sorting and sorted data to design new algorithms.

An example of the use of sorting in analysis is the demonstration by Shamos[4] that a convex hull algorithm can be used to sort a list of real values and consequently is subject to an  $O(n \log n)$  bound. It can similarly be shown that a hidden-surface algorithm can be used to sort and is thus subject to a sorting bound. In the case of image domain hidden surface algorithms, the sorting occurs in a bounded integer space which is subject to an  $O(n)$  bound (reserved seat or radix sort).

The use of sorting, especially pre-sorting in the design of algorithms has also been useful, especially algorithms which will be applied many times to the sorted data. Of particular interest has been the work on nearest neighbor problems and rectangle intersection.

#### 1.1.4 Sorting In Hidden-surface Processing -

The most widely known relationships between sorting and a class of geometric algorithm is in the area of hidden surface algorithms. Indeed, one class of algorithms is known as priority ordering. A general categorization of hidden surface algorithms has been done by noting the order in which they (sometimes implicitly) sort their input[3]. Of particular interest to octal tree sorting is the Schumaker or GE-Syracuse algorithm. This technique established a major a priori ordering of clusters of faces based on a separating plane structure. An intra-cluster minor ordering is established based on construction of a directed graph of pair-wise face priorities. The graph is traversed, assigning to each face its maximum path length from any face which cannot be obscured. The tree structure generated by the pre-ordering is



## INTRODUCTION

traversed in an order substantially independent of the viewpoint. The pre-ordering cost was quite high, however. The Newell[11] algorithm initially uses a linear sort to order faces on maximum depth, refining the sort order with a more subtle definition of R, i.e. relative visual priority with respect to the observer. Doctor and Torborg[2] demonstrated the use of an octal tree ordering on tree-modelled solids.

### 1.2 N-Dimensional Sorting

The extension of the one-dimensional or linear tree sort to Cartesian n-dimensional space will now be described. The technique is a logical extension of the Quicksort. The data to be sorted are contained within a (hyper)cubic volume bounded by  $2n$  planes, each pair corresponding to minimum and maximum values of a coordinate. A division point ( $P_d$ ) is chosen within the volume. An n-bit code may now be assigned to each point ( $P_i$ ) in the data. Each bit position  $j$  in the code is set to 0 if the  $j$ -th coordinate of  $P_i <$  the  $j$ -th coordinate of  $P_d$  else it is set to 1. Thus each (hyper)volume is subdivided into  $2^{**n}$  subvolumes coded  $0..(2^{**n}-1)$ . For non-empty subvolumes, a list of points contained therein is created and assigned to the subvolume. A tree is thus created in which the root is the wholly containing (hyper)volume. Each node of the tree has  $2^{**n}$  descendants, each descendant representing a sub(hyper)cube of its parent. A leaf is defined as a (hyper)volume containing no more than a specified number of points.

## INTRODUCTION

In three-dimensional space the tree, an octal tree, has a maximum out-degree of 8, coded 0..7(Figure 1). The two dimensional equivalent is a quad-tree[9]. Each descendant of a node is bounded by 3 pair of x, y and z planes. The creation of such a tree does not inherently produce a sorted list. The rules for traversing such a tree produce an ordering on the data it contains. The following discussion will confine itself to octal tree schema.

As a side note, another n-dimensional sorting scheme which has been demonstrated is the so-called "onion-peel" sort. Successive applications of a convex hull algorithm to a groups of points in space, with removal of points on the convex hull after each application, will induce an ordering on the set of points. The utility of such an ordering has not been shown.

### 1.2.1 Implementation Note -

A three-dimensional (octal) version of the tree sort has been implemented in Pascal(Figure 5). Two data structures are used in procedure OCTATE, the recursive octal tree generator. The structure OCTEL describes an octal element of space with eight subvolumes, each described by a substructure. By use of variant records, the substructures are permitted to be either a list of 3-space points or another OCTEL describing a suboctel. Points in 3-space are described by linked list to facilitate movement of point data between list and avoid data size limitations of fixed arrays. Input to OCTATE are a pointer to an OCTEL to be subdivided and a pointer to the list of points the octel

## INTRODUCTION

contains. A global variable contains the maximum number of 3-space points a leaf node may contain.

Informally, the procedure operates as follows:

```
PROCEDURE OCTATE(ptr-point-list,ptr-OCTEL);
  WHILE the point-list is not empty DO
    get next point P;
    generate octant code for P based on division point Pd in input
    OCTEL;
    move P to point-list associate with octant[code];
  END WHILE;
  FOR octant:=0 to 7 DO
    IF octant contains more than allowed number of leaf points
      THEN OCTATE(octant-point-list, new-octel)
      ELSE mark suboctel as empty or non-empty leaf;
    END FOR;
  END OCTATE;
```

### 1.2.2 $2^{**n}$ Recursive Division Of Space -

In creating an  $2^{**n}$ -tree subdivision of space, a strategy for choosing Pd must be established. An early use of the subdivision strategy in computer graphics was the Warnock hidden surface algorithm[10]. The technique divided 2-dimensional image space recursively until a subregion was either below the resolution of the screen or simple enough to process as a primitive case. The original subdivision strategy was to divide about the centroid of a region. Later experiments used projected vertices of the polygons and even used division along lines not parallel to the coordinate axes.

## INTRODUCTION

In experimenting with the octal tree algorithm three schemes were examined: division about the centroid of the cubic volume; division about the centroid of the data points contained in the volume ;and, division about a randomly chosen point in the volume. Keeping in mind that the situation under study was that of surface sampled objects, each of these has its own characteristics. Sampling about the centroid of the cubic volume is the simplest to compute, requires no record of the division point in the data structure but, because the points are typically on a surface which only lies in 2 or 3 of the octants of the cube, yields trees with average outdegrees in the 2 to 3 range. When the centroid of the data points are used, a more even distribution of data in the descendants occurs except when the points lie on planes parallel to the coordinate axes, a situation which came up in several simulated test cases. The use of a randomly chosen data element is adequate when measurements are truly unstructured; most often this is only a convenient intuitive notion resulting in badly skewed trees. The strategy of choice is a function of the nature of the sampled geometry data. It affects storage utilization and to a lesser extent speed.

It is important to note that the condition which halts the recursive subdivision is the presence of  $\leq N$  points in a descendant. This is distinct from the rules on spatial subdivision which limit the descent based on the size of the volume. Highly sampled areas of a surface will thus generate deeper trees than lightly sampled areas. The use of a variable limit on the maximum elements in a leaf allows the algorithm to group points, e.g. into triangles, for subsequent processes to use.

## INTRODUCTION

### 1.2.3 $2^{**n}$ Trees -

As noted above, the trees generated are true  $2^{**n}$ -ary trees. Every node can be assigned a unique code of a sequence of base  $2^{**n}$  digits corresponding to the path from the root to the node. The least significant digit in which two codes are equal identifies the common octel containing both and thus places an upper limit on their distance apart. If two trees are generated from the same enclosing volume, and use identical division points (typically using the centroid of cubic volume rule) then set operations and comparisons may be performed based primarily on the tree structure rather than geometry.

### 1.2.4 Traversal Rules -

The actual order in which data are accessed is determined by the traversal rules. Clearly, the definitions of pre-order and post-order traversal are directly applicable to higher dimensional trees. In-order traversal is less clear. A more significant problem is the choice of an order in which to visit the descendants of a node. The natural starting point is to take advantage of the order suggested by the codes used in the creation of the tree and access using 0..7 in order. This certainly produces an ordering and it is called the natural order.

The notion of visual priority suggests a more flexible ordering technique. Given the 8 octal elements (octels) composing a volume, associate with each, the corner of the parent octel it contains. Choose a point outside the volume to be a viewpoint. Calculate the distance to each corner of the parent volume and order these in ascending order.

Generate the sequence of suboctel codes corresponding to the ordered list of corner distances. Visit the contained octels in the order generated. It will be shown below that this establishes a visual priority ordering on the octels. For orthogonal projection, a single computation of the priority traversal order is sufficient. For a point close to the object, the order may have to be recalculated during traversal.

#### 1.2.5 Yuval Sort And The 3-dimensional Case -

In work related to visualizing three-dimensional models of organs derived from sets of two-dimensional CAT scan slices, an algorithm for visualizing voxel models of objects with a 100,000 or more faces was developed and implemented on a 16-bit address computer[7]. A virtual frame buffer was implemented to allow for a high-resolution z-buffer algorithm. Due to the lack of an appropriate ordering in the face file, low hit ratios resulted in poor performance. G. Yuval[8] developed a sorting technique which resulted in a 3:1 improvement in performance attributable to the regional sort property of the output. Given a set of points in an integer 3-space of precision  $n$  bits:

$$P_1, P_2, \dots, P_n$$

Construct a  $3n$  bit number, assuming that each integer is represented in binary as

$$b(n-1), b(n-2) \dots b(0)$$

the  $3n$  bit number is constructed from the  $x$ ,  $y$  and  $z$  coordinates in binary as

## INTRODUCTION

$[x(n-1)y(n-1)z(n-1)] [x(n-2)y(n-2)z(n-2)] \dots [x(0)y(0)z(0)]$

Sort the  $3n$  bit binary ( $n$  digit octal) numbers in ascending order. After sorting (by any linear sorting method) reconstruct the original binary values for the coordinates.

Essentially this technique takes the entire integer three space and applies an octal tree sort (OTS) to the points in the space, each point being assigned a unique octal code defining the path to its leaf in the tree. Effectively, this results in the property that points closely adjacent in three space are sorted closely together on the list. The property is not an absolute one, but, in general, the consequence in the image plane, for an arbitrary rotation of the object, is that portions of the frame buffer are referenced in a local pattern that substantially reduces the swapping requirements to external storage such as disc or physical frame buffer.

### 1.2.6 Octal Tree Sorting -

It is useful at this point to demonstrate that an OTS, the combination of the generation of an octal tree and specification of a traversal rule, is a true sort. Clearly, no elements of the input data are destroyed nor added during the build and traversal process so that a permutation of the original data has occurred. Using the natural traversal order we can state the ordering relation on the resultant output:

if  $(P_i R P_j)$ , for any  $i < j$   
 $(X_i \leq X_j)$  or  $(\sim(X_i \leq X_j) \text{ and } (Y_i \leq Y_j))$  or  $(\sim X_i \leq X_j) \text{ and } \sim(Y_i \leq Y_j)$  and

## INTRODUCTION

$(Z_i \leq Z_j)$

This is reflexive. Assume  $(P_1 R P_m)$  and  $(P_m R P_n)$ , if  $(P_1 R P_n)$  did not hold then the complement of  $R$  holds and

$(X_1 > X_n)$  and  $(Y_1 > Y_n)$  and  $(Z_1 > Z_n)$

which shows that there can be no value of  $X_m$ ,  $Y_m$  or  $Z_m$  which satisfies the initial assumption. Lastly,  $R$  is clearly antisymmetric. Thus for the natural order, OTS is a true sort. A similar demonstration holds for any of the 40320 traversal orderings.



## 2.0 OCTAL TREES

### 2.1 Octal Tree Representation Of Solids

The use of octal trees to represent solids, as opposed to points or surfaces, was a logical extension of published work on quadtrees and their use to represent area information in images. It is both efficient and natural to recursively subdivide space, maintain those suboctels which contain "solid" material and construct an octal tree representation of the object. This is especially useful for ab initio design in which set operations (union, intersection and difference) and their equivalent machining operations are useful. The hierarchical nature of the description relates to rough machining and its refinements.

In addition to the utility of octal tree solids in representing ab initio designs, their application to the modelling of volumetrically sensed objects was investigated. Algorithms for converting serial section CAT scan data were developed[12].

#### 2.1.1 Octal Tree Representation Of Cellular Array Solids -

For an object whose abstract description is known, the technique of model generation is top-down subdivision. Given a bounded, right rectangular parallelepiped containing a solid object. Recursively subdivide it into eight subvolumes by passing a plane parallel to each pair of faces through its centroid. This generates an octal tree. A volume may be labelled white(disjoint from the object), black(completely

## OCTAL TREES

filled by the objects material) or grey (partially filled and partially empty). If a subvolume is white or black, no more detailed description of it is necessary. If a subvolume is grey, recurse another level. Recursive descent is terminated either by generation of a white (black) subvolume or by a subvolume reaching minimum dimensions. The resultant octal tree is a compressed cellular or enumerative model of the object.

In the case of objects known by enumeration of their set of occupied volume elements (voxels), the technique is bottom-up. Essentially, groups of voxels are combined to generate larger octels which are in turn combined. The precise algorithms depend on the format in which the enumerated voxel data is stored. Algorithms have been defined for sequences of planar sections and sets of 1 by 1 columns of data.

### 2.2 Surface Triangulation From OTS Points

The sorting of point data is useful, but not the real goal of the schema. The problem with any geometric sampling technique is that a list of surface point values is of limited value in applications such as image generation or pattern recognition. A faceted surface model consists of a connection topology as well as a set of geometric data. For a variety of reasons a surface composed of triangles is desirable. A triangle is planar and therefore has a single valued normal. The connection graph of a triangle is unique within a sign change. For

these reasons, triangulation of measured surfaces has been applied to measured geometry data. Of particular interest have been a series of algorithms e.g. [13] which generated triangulated surfaces from outlines on parallel serial slices of solids. Numerous triangulation techniques exist for objects whose high-level description is known, their purpose being to prepare data for finite element analysis.

### 2.2.1 Tripunctate Leaves -

In preparing to triangulate the surface, an aid has been built into the octal tree sorted point data. Rather than requiring each leaf of the tree to contain at most one point, the leaves have been allowed up to three points. This provides a natural set of primitive triangles, lines and points to begin the triangulation process. These tripunctate leaves provide information beyond merely setting up the initial connection process. By analyzing the lengths of the sides of the primitive triangles, a measure of the density of the surface sampling can be determined. Calculation of the mean and standard deviation of the primitive triangle edge length can be performed at each recursive level of the tree.

### 2.2.2 Connection Strategies -

Two aspects of the connection strategy must be considered. First, the choice of which octels are to be connected must be made. Second, the scheme for choosing a set of edges to lay across the inter-octel boundary must be made.

## OCTAL TREES

At any point in the process, 8 octels are under consideration to be combined. There are 12 pairs of face adjacent candidates, 6 edge adjacent and 4 pair of corner adjacent (or diagonal) candidates. For each octel there is a point closest to each face adjacent boundary. Since the planes separating the octels are parallel to coordinate axes, the distance of each point to the separating planes is simply calculated. Empty octels are not considered. From the occupied, face-adjacent pairs, choose the pair of octels whose combined closest distance from the plane separating is minimum. After triangulating over the plane separating them, treat them as a single polygon. The rules for combining point, line and triangle primitives are summarized in Figure 3. If no face adjacent pairs exist, then the edge adjacent pairs are considered, then corner adjacent. Manhattan distance is used to reduce cost.

Before presenting the procedure for triangulating between the high-level polygons generated from the previous step, the use of the sampling density data will be described. As noted, for each octel, the mean and standard deviation of the lengths of the sides of its contained primitive triangles has been calculated. In any projection based surface sampling technique there can be portions of an object hidden from view, as well as artifacts e.g., as measuring the surface on which the object rests. In generating edges in the triangulation process, a limit on the length of legal edges is useful. In essence, by placing such a limit on the triangulation scheme, gaps in the data are left as open areas for which no surface is generated. Surfaces which are widely separated, will have gaps generated between them in an appropriate

manner.

A high-level polygon has more than three edges and is the result of the combination of lower level polygons. The connection process generates interior triangles and a merged boundary polygon. Figure 4 provides an example of high-level polygon merging. It should be noted that the success of this heuristic is somewhat dependent on the fact that measured objects tend to have rather well behaved surfaces and pragmatic considerations suggest that they be sampled at a density which prevents some of the nastier problems from arising. Also, external surface sampling avoids many of the problematical multiple connection problems in volume sensing.

Again, the pair of octels to be merged in chosen by the minimum sum of distances to the separating plane. The problem is reduced to a pseudo-two-dimensional one by performing some decision making in the two-dimensional orthogonal projection in a plane normal to the separating plane. The two candidate polygons A and B are represented by ordered lists of three dimensional vertices ( $a_i$   $0 \leq i \leq s$  in A and  $b_j$   $0 \leq j \leq t$  in B). Indices to the vertices are modulo  $s$  and modulo  $t$  respectively. From the sampling analysis of the primitive triangles in the containing octel, establish a maximum length permissible edge in the connection,  $L_{max}$ . An edge with length greater than  $L_{max}$  may be used to allow the algorithm to complete, but a triangle with such an edge is never recorded.

## OCTAL TREES

Connect the two vertices whose combined distance caused this pair of octels to be selected. Let the two vertices connected be  $a_k$  and  $b_l$ . Vertices  $a_{(k-1)}$  and  $a_{(k+1)}$  modulo  $s$  are now candidate vertices for connection to  $b_l$ . Similarly,  $b_{(l-1)}$  and  $b_{(l+1)}$  modulo  $t$  are candidates for  $a_k$ . At any point in the connection process there will be two candidates on  $A$  for connection to the last connected vertices on  $B$ ; likewise two on  $B$  for the last two connected on  $A$ . Of the four candidates, the connection edge with minimum Cartesian length is chosen. This converts one candidate to the status of most recently connected on its polygon, generates two new candidates and generates a triangle. If no edge of the triangle exceeds  $L_{max}$ , the triangle is recorded as a surface element. The process continues until all vertices on both  $A$  and  $B$  are vertices of a connection triangle.

Clearly, this is neither an optimal algorithm nor universal. The "well behaved" nature of real world surfaces sampled at high density obviates most problems which would require either a more universal or an optimal approach. Note that every generated triangle lies inside its containing octel and crosses known boundary faces within that octel. The primitive triangles generated by the three vertex leaf nodes of the tree lie entirely within leaf octels. Triangles spanning large octels are not present due to the  $L_{max}$  limit on edge length.

### 2.3 Inter-octel Polygons

As is clear from the description of the surface triangulation algorithm, when the primitive geometric element used for OTS modelling

is raised in elemental complexity to a triangle (polygon), problems arise with respect to the notion of an element being contained within a leaf octel. Both in the case of the triangulation algorithm, under which constraints can be placed on the polygons, and the processing of directly generated faceted models, polygons which intersect multiple octels must be represented.

### 2.3.1 By Subdivision -

The simplest solution (from the viewpoint of concept) is that of subdividing each inter-octel polygon so that its components lie only in leaf octels. Using a polygon clipping algorithm [15] the modelling polygon can be divided into components lying in leaf octels. This increases the number of polygons present in the model and, consequently, the cost of subsequently applied algorithms. The tradeoff is simplicity of implementation and maintenance of a strictly static priority relationship in the octal face tree. Loss of a strict hierarchy, as will be explained below, will force more processing into the traversal of the tree for relative visual priority determination.

### 2.3.2 By Hierarchical Assignment -

The alternative to subdividing every "large" face into leaf polygons is to assign polygons to their smallest containing octel on the tree. In the case of triangulation polygons, the size limitation ( $L_{max}$ ) will prevent very many polygons from the necessity of being placed very high in the tree structure. This approach makes the traversal process

much more complex in that all nodes may now contain elementary geometric items. A polygon's area may intersect a suboctel in which no vertex lies. This makes relative priority determination more of an insertion process than mere traversal. The process resembles the "special sort" of [11].

### 2.3.3 Octel Priority Ordering -

Consider a plane  $P_a$ .  $P_a$  divides space into two half-spaces  $aH_1$  and  $aH_2$ . An observer is located in  $aH_1$ . Any object in  $aH_1$  will have visual priority over any object in  $aH_2$  under these circumstances. For intersecting two planes,  $P_a$  and  $P_b$ , space is now divided into 4 regions, only one of which may contain a viewpoint. A binary code is applied to each region in which the first (a) bit is one for the half-space of a containing the viewpoint and the second (b) bit similarly. Region 3 contains the viewpoint. Any object in region 3 has visual priority over all other objects. Due to the intersection of the separating planes, objects in regions 1 and 2 can have no visual priority conflict. Objects in region 0 have the lowest relative priority, being capable of being obscured by objects in 3, 2 or 1. Objects are taken to be wholly contained within their regions. Note that the choice of the assignment of relative priority between 1 and 2 is a "don't care" choice unless other considerations exist. For the case of a viewpoint lying on a plane, e.g.  $P_a$ , the assignment of a code bit associated with the plane is identical in both half-spaces, e.g.  $aP_1$  and  $aP_2$ . No relative priority assignment is made among objects lying within a region.



## OCTAL TREES

The extension of this scheme to the octants generated by three intersecting planes is straightforward. For any octel, three planes of subdivision are defined, each parallel to a coordinate axis,  $P_x$ ,  $P_y$  and  $P_z$ . The eight suboctels into which the octel is divided can then be assigned relative visual priorities based on a viewpoint. Within each region, a relative priority ordering of its contents may be accomplished by a recursive subdivision and code assignment. This approach matches directly to octal tree sorting.

For viewpoints distant from the object model, orthogonal projections or fortuitous choices of viewpoint, the code calculation may occur only once for the model. For the general case, codes must be assigned at each subdivision or nodal descent. The simplicity of the calculation does not make this an expensive calculation.

For the situation in which all elements to be visualized are assigned to leaf nodes, a traversal of the tree with order determined by relative visual priority codes, the leaf nodes are visited in proper priority order regardless of the position of the viewpoint. Thus, polygons may be written into the frame buffer without any priority checking. Thus, a true priority sort occurs from a viewpoint independent data structure. The priority determination for the data structure in which polygons are assigned to their smallest containing octel is more complex.

If the technique of assignment to smallest containing octel is used, the definition of the octel record must be extended. The relationship between an octel and a polygon is tripartite: (1)the

## OCTAL TREES

polygon is wholly contained in the octel; (2)the polygon is disjoint with the octel; or, (3)the polygon intersects the octel but is not wholly contained. An octel record must now reference both a list of wholly contained polygons and a list of intersecting polygons. Note that if all vertices of a polygon lie in an octel, the polygon is wholly contained. Even if no vertices of a polygon lie in an octel, its area may intersect the volume of the octel. For intersecting polygons note: if the vertices of the polygon lie in face-connected octels, then the polygon lies entirely in the face connected pair. If the vertices of a polygon lie in an edge connected pair of octels then the polygon lies entirely in the four octels sharing the edge. If the vertices are in a set of octels sharing only the central division vertex of the containing octel, any octel may be intersected.

Consider a postorder traversal of a octal-tree in which the records are extended to represent contained and intersecting polygons. Using a traversal order based on the viewpoint, traverse the tree in post-order fashion. After processing the descendants of a node (suboctels of the octel) a priority ordered list of all polygons contained in the suboctels exists. Now process the polygons for which the octel is the smallest containing octel. Assuming that the polygons are non-intersecting and linearly separable (a consequence of surface sampling) the priority ordering relationship is a linear ordering. Therefore, insertion of the octel list is a simple linear insertion. If the list of polygons for which this is the smallest containing octel is organized based on the (face,edge,vertex) relations noted above, the insertions can be done in an in-order fashion as suboctels are

processed. The consequence of the process is a priority ordered list of all contained polygons to pass up the recursive ascent. As the polygons on the list are processed in a priority ordered fashion, groups of polygons on the list may be released for writing to the frame buffer when no additional intersecting octels remain to be processed for their smallest containing octel. This requires additional data in the polygon record in the priority ordered output lists. In situations such as sampled surface triangulation where polygon (triangle) size is limited the size limitations relieve some of the bookkeeping for this early release process.

### 3.0 APPLICATION OF OTS TECHNIQUES

The utility of a technique such as OTS is of little interest unless specific applications can be demonstrated. The use of OTS provide several of the data organization properties of linear sorting. Data which is "close" is clustered in a relatively close position in the list. Subsets of the data can be accessed by taking advantage of the sorted list structure. Due to the three dimensional nature of the OTS, data can be accessed in relative visual priority order by modifying the traversal order based on viewpoint.

#### 3.1 Geometric Surface Sampling

A number of techniques for the automatic digitization of surfaces using arrays of projected light beams have recently been described[1]. In each system, a rectangular array of beams is projected onto the

## APPLICATION OF OTS TECHNIQUES

surface of the object(s) to be digitized. The light dots created when each ray intersects the surface of the object are imaged and the image coordinates measured by one or more imaging devices. The array is regularly spaced in the projector but, due to the effects of perspective and the shape of the object, unstructured on the surface of the objects. As a consequence of the shape of the object and camera positioning which obviously must be off-axis from the projector, some rays may not be imaged.

To measure a larger portion of the surface of an object, the cameras and/or lasers may be moved around the object. If the projector is moved relative to the object or multiple projectors used, the registration between patterns is not inherent in the system. Since the surface is discretely sampled, the use of fiducial or registration marks will not, in general, be successful. Consequently, a technique which allows for merging of the sets of measurements from multiple projectors is necessary. In linear measurements, the solution would be to sort the measurements into a single list; the natural extension is to take the multiple sets of measurements and OTS them into a single tree. No reference between sets is needed and the list accommodates to the common occurrence that some portions of the object will be densely sampled while others will be lightly sampled or not sampled at all. The existence of the triangulation algorithm to generate a surface consistent with the points is of great value. As noted below, the surface triangles thus generated, may be of use in performing pattern recognition on the three-dimensional surface thus generated. It should be noted that the work on OTS described here was primarily motivated by

the multiple sampling problem.

### 3.2 Virtual Frame Z-buffers

As noted in the description of the Yuval sort, a problem can arise in the use of the Z-buffer algorithm when primary memory is too small to contain the necessary arrays. The use of an OTS list of faces, equivalent to a Yuval sorted list for the natural order traversal, allows for a localization of the references to virtual memory. This may substantially reduce execution time.

Briefly, a z-buffer hidden surface algorithm uses a model of a raster scan image in which each of the pixels is represented by a pair of values: color code (grey level) and minimum z seen to date through the pixel. Each face of the geometric model is appropriately transformed, projected onto the image plane and scan converted to a list of pixels with associated z and color code value. Pixels are conditionally written into the z-buffer subject to the condition that the z value of the polygon pixel to be written is less than the buffer pixel's z. Individual faces' references to z-buffer memory are localized due to the contiguous nature of the polygon's area. Polygon to polygon locality in the data structure is not typically maintained, especially after rotations. While the color code portion of the z-buffer pixel is write only, the z value is needed for control of the conditional write. In a typical 512 by 512 raster, only 8-bit precision on color code and z require 512KB of memory (plus an additional 512KB for true color work). While a formal statement of the OTS's localizing

property is not yet available, the results from the Yuval sort show the potential gains.

### 3.3 Ray Tracing Algorithms

A ray tracing algorithm is a computer generated image scheme which casts a line of sight ray from the viewpoint, through each image pixel to the environment. The portion of the environment model which intersects the ray closest to the viewpoint is that imaged in the pixel. The technique has the advantage of being especially powerful in the simulation of reflective and refractive phenomena. The principal difficulty of the technique is its high computational cost due to the cost of attempting to calculate the intersection of (typically)  $512 \times 2 = 256K$  rays with the polygons in the environment. Consider a set of polygonal faces. The traversal rule is set based on the viewpoint, with possibly different traversal lists at each node for sufficiently close viewpoint. A single ray is selected for casting. A priority ordered traversal of the OTS face list is performed with extensions. No off ray octel is accessed, eliminating large portions of the model space. Since the faces are accessed in priority order, traversal can be terminated as soon as no higher priority faces can be accessed. A hypothesis which warrants further investigation is that ray-to-ray coherence would allow further reduction in processing.

### 3.4 Unit Normal Sorting

Consider a set of measurements in n-dimensional space. Assume that the measurements are octal tree sorted using the centroid of data elements rule for selecting Pd. Regions with high densities of data points (clusters) will be characterized by small volume leaf nodes, greater depth in the tree and higher densities and counts of contained data points in non-leaf nodes. As noted by Warnock in his own domain, recursive subdivisions of space, controlled by the density of "interesting" phenomena, concentrate their work in regions of significance. Thus such an OTS based cluster analysis is relatively independent of the shape of the cluster.

As noted above, this work has its origins in three-dimensional surface sampling or geometry acquisition systems. Consider a system which samples several hundred or several thousand points on the surface of a physical object. The point data is then passed through the OTS triangulation algorithm yielding a densely sampled three-dimensional surface modelled with triangular faces. Clearly, the goal of such a process can be visualization or measurement. Of at least equal importance is the application of pattern recognition processes to ascertain: (1) which, of a possible library of three-dimensional objects this most closely fits; or, (2) given a three-dimensional model of a single object, determine the pose (position and rotational transformation) of the instance measured. As a consequence of the surface triangulation, highly sampled surface normal information is available. If the normals are unit in magnitude, they can be plotted in a 2-dimensional "map" projection. Planar surfaces will result in

volumetrically small, tight clusters of normals. Edges between planar surfaces will appear as linear, less dense features between the planar clusters for the pair of faces meeting in an edge. An intersection on three faces in a corner would appear as a triangle with dense vertices and linear edge features. A cylindrical surface would generate a denser linear feature; a sphere a relatively uniformly populated region. The use of geometric sorting to recognize these features allows for rapid processing.

#### 4.0 CONCLUSIONS

The octal tree sorting process allows for pre-processing geometric models in a fashion which eases a number of geometric and graphic operations. The octal tree structure allows for a relatively simple priority accessing of the surface, aiding in priority based hidden surface schemes. The local clustering property of the spatial subdivision allow for a triangulation algorithm which supports unstructured surface sampling without fiducial points. This property also reducing paging rates for virtual z-buffer implementations. Lastly, it appears that applying this approach to normal data may allow for better recognition of abstracted geometric features in three-dimensional surface sampled data.



## References

- [1] Posdamer, J.L. and M.D. Altschuler-"Surface Measurement by Space-encoded Projected Beam Systems", Computer Graphics and Image Processing, in press
- [2] Doctor, L.J. and J.G. Torborg-"Display Techniques for Octree-Encoded Objects", IEEE Computer Graphics and Applications, July, 1981
- [3] Sutherland, I.E., R.F. Sproull and R.A. Schumaker-"A characterization of Ten hidden Surface Algorithms", Computing Surveys, March, 1974
- [4] Shamos, M.I.-"Geometric Complexity", Proceedings of the Seventh ACM Symposium on the Theory of Computing, May, 1975
- [5] Bentley, J.L.-"Multidimensional Binary Search Trees Used for Associative Searching", CACM, Sept., 1975
- [6] Knuth, D.E.-"The Art of Computer Programming, Volume 3, Sorting and Searching", Addison-Wesley, 1973
- [7] Artzy, E., G. Frieder and G. Herman-"The Theory, Design and Evaluation of a Three-dimensional Surface Detection Algorithm", Computer Graphics (SIGGRAPH '80 Proceedings), July 1980
- [8] Yuval, G.-Private communication
- [9] Samet, H.-"Region Representation: Raster-to-Quadtree Conversion", Computer Science, Univ. of Maryland, 1979
- [10] Warnock, J.E.-"A Hidden Surface Algorithm for Computer Generated Half-tone Pictures", TR 4-15, Dept. of Computer Science, Univ. of Utah, Salt Lake City, UT, June, 1969
- [11] Newell M.E., R.G. Newell and T.L. Sancha-"A New Approach to the Shaded Picture Problem", Proc. ACM National Conference, 1977
- [12] Srihari, S and M.M. Yau-"Octal-tree Descriptions form Serial Slice Data", Department of Computer Science, State University of New York at Buffalo, 1981
- [13] Fuchs, H., Z.M. Kedem and S.P. Uselton-"Optimal Surface Reconstruction from Planar Contours", CACM, Oct. 1977
- [14] Hoare, C.A.R.-"Quicksort", Computer Journal, Vol. 5, No.1 1962
- [15] Sutherland, I.E. and G.W. Hodgman-"Reentrant Polygon Clipping", CACM, January 1974 @@

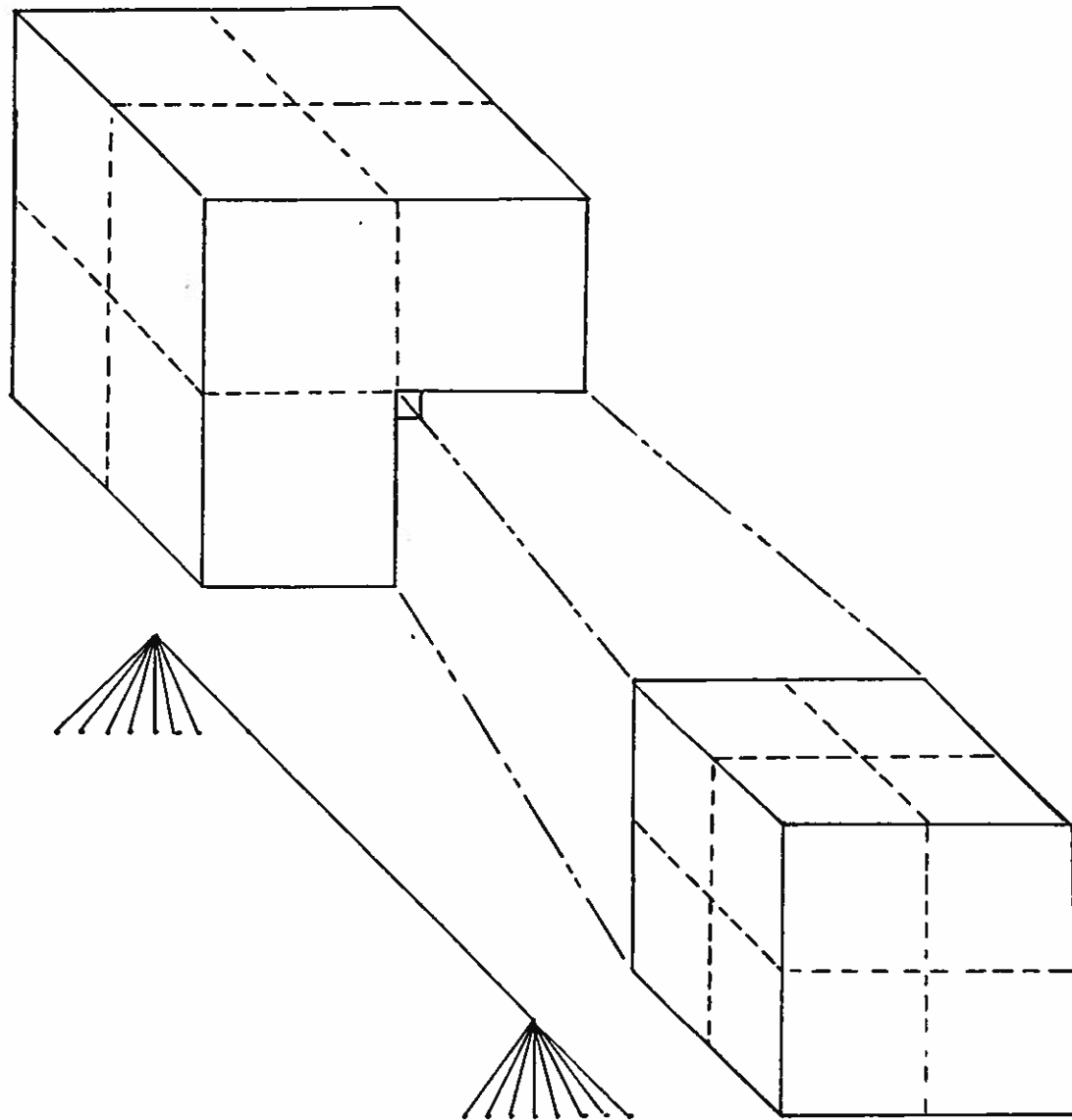
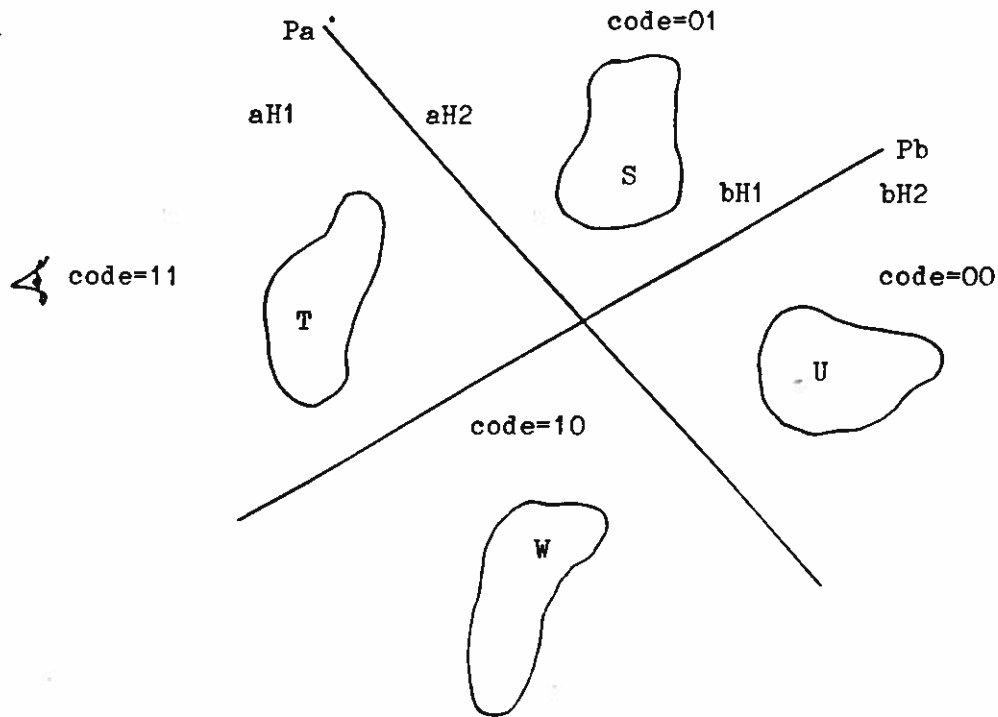
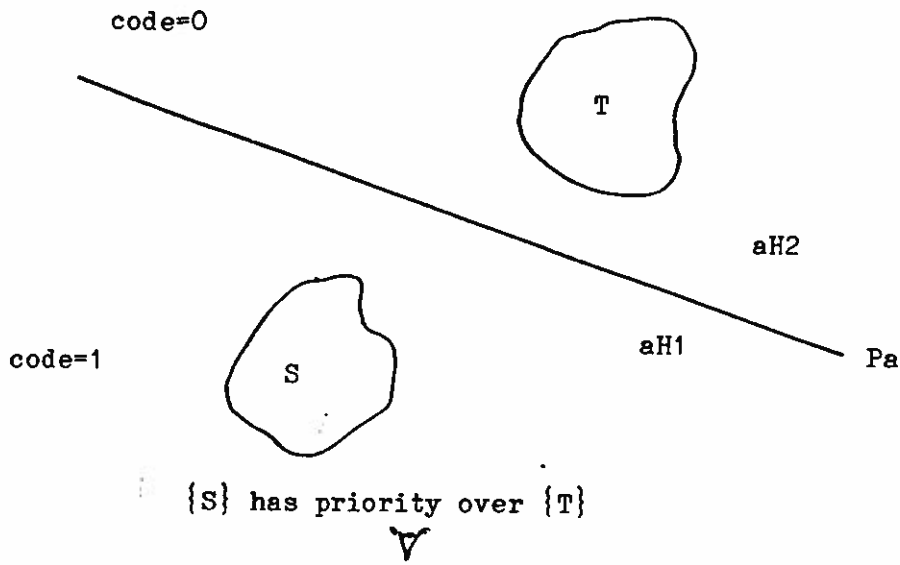



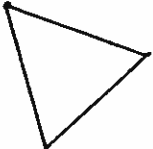
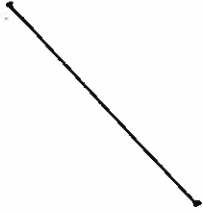
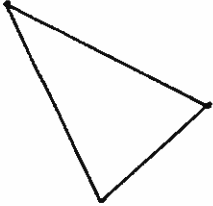
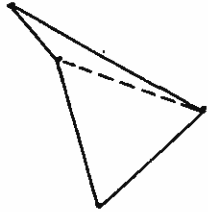

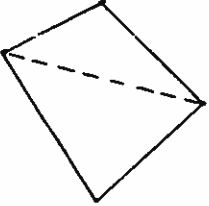
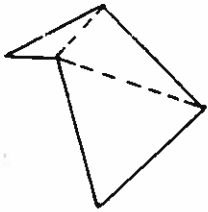

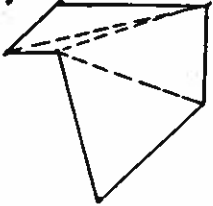
Figure 1 Recursive octal subdivision of space

CONCLUSIONS



{T} has priority over {{S}{W}} has priority over {U}

Figure 2 Priority ordering of octels

			
			
	Symmetric		
	Symmetric		



 Output result edge  
 Internal triangle edge

Figure 3 Primitive triangulation rules

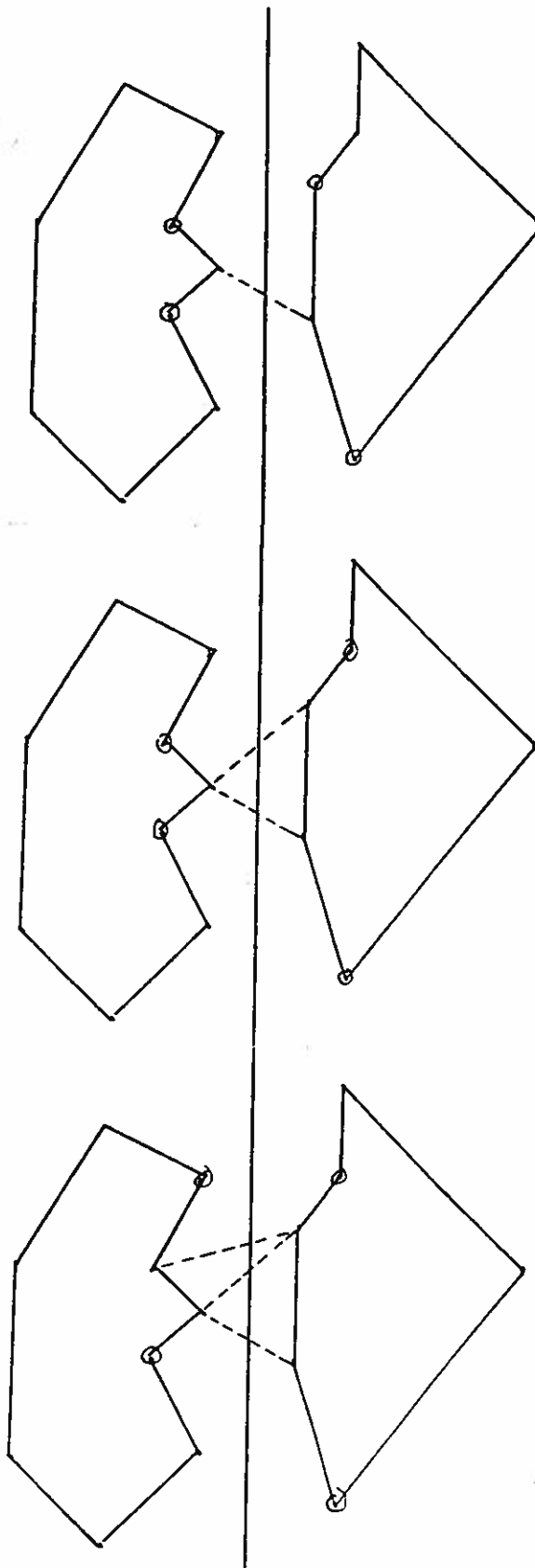


Figure 4 Example of high-level triangulation

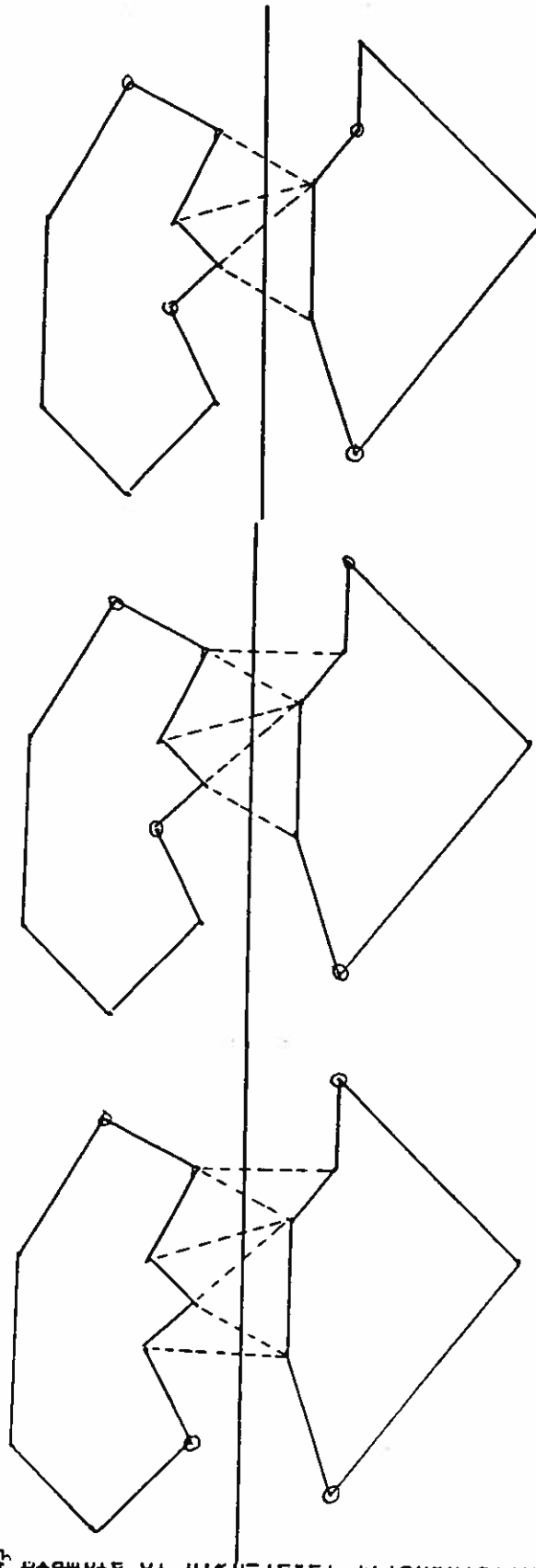


FIGURE 4. EXAMPLE OF HIGH-LEVEL TRIANGULATION

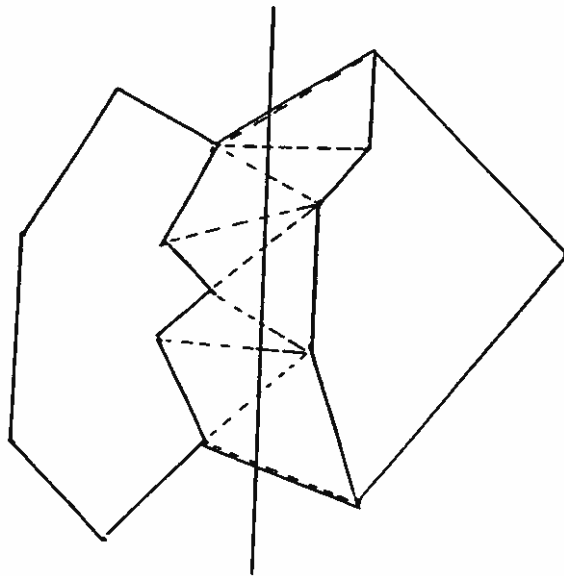
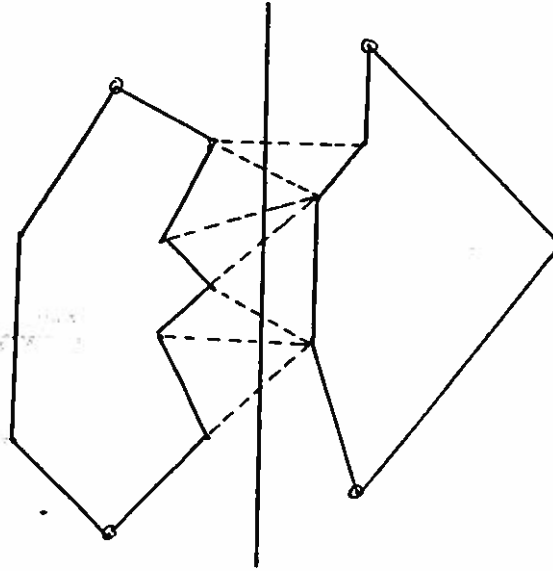


Figure 4 Example of high-level triangulation