Interpreting a 3D Object From a Rough 2D Line Drawing

Del Lamb Amit Bandopadhay

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400

Abstract

Visualizing the third dimension while designing 3D objects is an awkward process in mechanical CAD systems, given the current state of the art. We describe a computer system that automatically constructs the shape of a 3D object from a single 2D sketch. The method makes it convenient to create and manipulate 3D objects and is thus seen as an intelligent user interface for CAD and 3D graphics applications. The proposed technique is built on well known results in image analysis. These results are applied in conjunction with some perceptual rules to determine 3D structure from a rough line drawing. The principles are illustrated by a computer implementation that works in a nontrivial object domain.

Introduction

Computer-aided design of 3D objects is often impeded by the lack of proper visualization tools during the design and/or drafting process. The reasons for this are twofold. Firstly, the interaction devices associated with most CAD workstations are, for technological reasons, 2D in nature (e.g. a mouse or digitizing tablet). Secondly, 2D drafting practices dominate the design of the CAD programs. Thus a 3D structure is still generated by specifying the principal views, such as plans and elevations. Since the ultimate goal is to convey the 3D structure to the computer, it would be convenient for the designer/draftsperson to directly visualize the 3D structure, instead of having to rely on 2D views. (In fact, one of the authors, who has had some drafting experience, remembers that it is common practice to draft designs by referring to hand-drawn isometric sketches, as an aid to visualization). In this paper, we outline the basis for the design of an intelligent user interface for specifying a 3D object from a rough 2D sketch, using only 2D input devices.

It should be pointed out that some of the techniques employed by researchers in computer vision and image analysis for object interpretation are also relevant in our case (e.g. Sugihara [9], Kanatani [5], Malik [7]). However, there are important differences. Most of these techniques use numerical quantitative methods which are susceptible to inaccuracies in the input, in addition to being computationally expensive. Also, these systems work to determine structure relative to the picture plane of the image. Our goal is to use a rough sketch to determine the object structure independent of the picture plane. Hence, the quantitative methods perform more work than necessary at considerable expense. Besides, hand-drawn input adversely affects these techniques, which are not noted for their robustness. While we avoid most of the quantitative techniques available, we can profit from some of the qualitative machine vision methods used in object recognition (Sugihara [9], Hanrahan [3], Waltz [10], Huffman [4]).

Limitations and Assumptions

Strictly speaking, it is impossible to uniquely identify a 3D structure by drawing a single view. This is because, mathematically speaking, the backprojection from 2D drawing space to 3D object space gives rise to an infinite number of possibilities for 3D interpretation. However, it is common knowledge that humans have no difficulty in seeing 3D structure in 2D drawings. So there must be some implicit assumptions and simplifications that enable us to arrive (in most cases) at unique interpretations of 3D structure.

Barrow and Tenenbaum [1] give some excellent examples of this problem. For instance, an ellipse in a drawing is usually perceived as a tilted circle, yet a series of disjoint curves in space can generate the same image from a very particular angle. The point is that the circle appears as an ellipse if viewed from almost any angle. However the set of disjoint curves forms an image of an ellipse because of a fortuitous alignment of the curves and the point of view. Therefore, humans normally reject the disjoint curve scenario in favor of the tilted circle interpretation.

CH2913-2/90/0000/0059/$01.00 – 1990 IEEE

This leads us to the *general viewpoint* assumption. The implication is that there is no fortuitous alignment of curves and lines in 3D from our particular viewpoint. In any case, such alignment will be rare from a probabilistic standpoint, occurring only for a few viewing positions. Thus if the object in the drawing is rotated slightly in any direction, the arrangement of the vertices and edges does not change. This prohibits views of the object from having edges or vertices that are aligned only because the drawing represents a degenerate view.

Structural Analysis Algorithm

Introduction to Algorithm

Currently, there are several methods used to input 3D objects into CAD products. Some of these involve the user specifying 3D coordinates for each edge of an object. Others can take precisely specified multiple views of an object and interpret the 3D structure. Another method is to use solid primitives and construct the object via a series of union, intersection, and difference operations. Each of these is useful in its own way, but each also has limitations. In particular, each requires the user to provide much of the specification, or at the very least, to be well versed in certain fields (e.g. drafting).

The goal of our research is to develop a system for specifying objects that does not require specialized knowledge on the part of the user. Ideally, our system would act as a computerized sketchpad, with the interface taking rough free-hand drawings, through a digitizing tablet or mouse, and interpreting 3D structure interactively. Such a system could be used to design objects alone, or in conjunction with the other methods previously mentioned.

The key to our system for visualizing 3D structure lies in developing an algorithm that is flexible. The goal is not to interpret an object relative to some view, or to try and elaborate on the number of possible interpretations a drawing may allow. Instead, our system should quickly present the most likely interpretation and then allow the user to simply evaluate and modify the result if it is incorrect. Because this is done relative to the object itself, we have the freedom to choose any coordinate system or principal axes that we find useful. Once the object has been evaluated and accepted, it can be translated or rotated to any desired position.

The method we are using to interpret the 3D structure of an object comes from our experience in drafting. The question that arises is how to specify a 3D object quickly, naturally, and accurately. While

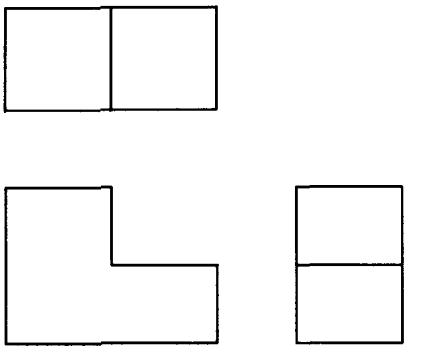

Figure 1 - Multi-View Drawing of Object

multi-view drawings communicate more information, they require some forethought, as well as some technical expertise. Single view drawings tend to display 3D information more clearly. The difference can be seen in the drawings of Figures 1 and 2.

Of the two main types of single view drawings, axonometric views are easier to interpret than perspective views for two reasons: because parallel lines on the object appear as parallel lines in the drawing, and because edges parallel to the principle axes are drawn with lengths proportional to the actual dimensions of the object. Of these types of drawings, the two most common forms are the isometric and the oblique drawings (see Figures 2 and 3). It is precisely these type of drawings that are most useful in sketching the 3D structure of an object.

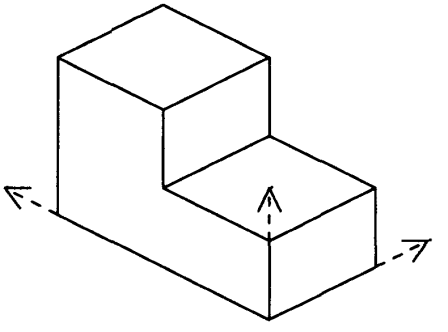

Figure 2 - Axonometric (Isometric) Drawing

One other aspect of the input drawing is whether to include hidden lines. While hidden lines can provide information about the unseen structure of an object, they also add a number of ambiguities to a drawing. Therefore, we have chosen to accept drawings without hidden lines. Another algorithm is used to infer the hidden structure of the object from the visible data given in the drawing. This may lead to an

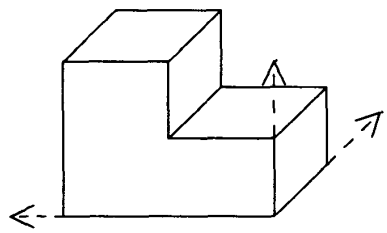

Figure 3 - Axonometric (Oblique) Drawing

interpretation that does not match what the user had in mind, but no single drawing can convey all the information necessary to prevent this. Instead, we take advantage of the interactive nature of this approach to quickly interpret the drawing and let the user modify the object to resolve any incorrect assumptions.

Inferring the 3D structure of an object is accomplished according to a very simple principle. Given knowledge of the directions of the principle axes in the drawing, the coordinates of any point on the object can be specified by calculating the distance along the X, Y, and Z-axes between this point and the origin. Likewise, if the coordinates of one point is known, the coordinates of adjacent points can be derived relative to the known point. The system makes use of this concept to determine all the coordinates of an object relative to a one junction selected as the origin. How the system accomplishes this task will become evident as we present the algorithm.

Preprocessing for Algorithm

The first step in interpreting a drawing is to obtain the input data in a form that the algorithm can understand. The drawing may be done with a digitizing tablet, on a screen using a mouse or light pen, or entered via some offline method, such as a digitized picture of the input drawing. We must first convert this input into an adjacency graph. Each vertex in this graph represents the junction of two or more visible edges of the object, with each edge representing the meeting point of two distinct planes. Along with the graph, we also compute additional information, such as the slope and length of each edge and the 2D coordinates of the each vertex in the picture plane.

Once the graph is available, the second stage is to apply a labeling to each vertex and edge in the graph. The algorithm for doing this has been taken from the method proposed by Waltz [10] in 1972. Each edge in the graph is labeled as representing a convex ridge, a concave valley, or the edge of an occluding surface for which only one of the planes incident at that edge is visible. As a part of this process, each vertex in the graph can be categorized according to the physical shape of the vertex and by the labelings of its adjacent edges. This labeling is used to reject any drawings that represent impossible objects before further processing, and to calculate information that is vital to determining the hidden structure of the object.

Description of Algorithm

Before explaining the algorithm itself, we need to define some of the terms that are used frequently. Each drawing is assigned a set of axes, representing the slopes of the *X-axis*, the *Y-axis*, and the *Z-axis*, as seen in the picture plane. The given set of axes for this drawing are referred to as the *principal axes*. Any two of these axes will form a plane, which we refer to as a *principal plane*. These planes are identified by the axis that is perpendicular to it. For example, the *Z-plane* refers to the plane that is parallel to both the X and Y-axis, and perpendicular to the Z-axis. Planes that are not principal planes are called *oblique planes*. The faces, or *regions*, in a given drawing are extracted and identified as principal or oblique by the labeling step. It is possible for a region labeled as an oblique plane to be reclassified as a principal plane when more information becomes available. Principal planes cannot be reclassified and provide the most information for interpreting the drawing.

Given an input drawing that has been appropriately processed, our goal is to assign 3D coordinates to all the vertices in the graph via the following steps:

Step 1: *Select the "best" region for coordinate assignment.* Each of the visible regions will be assigned coordinates. The order in which this is done is determined by an evaluation function. For example, regions that represent principal planes are processed before any oblique planes. Therefore, all unprocessed regions are evaluated and the one that will provide the most information is selected for processing. In Figure 4, the Y-plane would be evaluated first because it is the largest principal plane.

Step 2: *Select a reference junction for the region.* All coordinates for this region will be assigned relative to reference junction. If any junctions in this region have been assigned a coordinate prior to this point, the junction with the lowest Z-coordinate will be considered the reference point of the region. If this is the first region to be processed, no junction will have been assigned. Therefore, the junction with the lowest position in the picture plane will be chosen and assigned the 3D coordinate of (0,0,0).

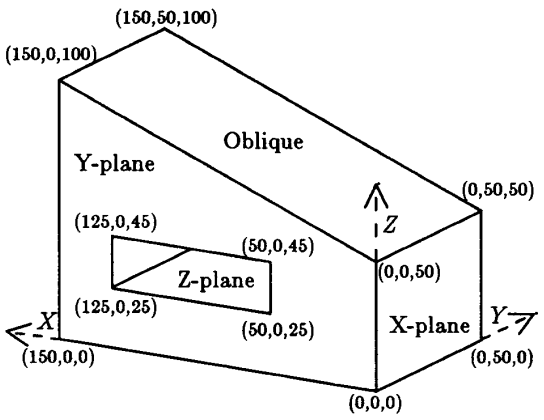

Figure 4 - Example of Coordinate Assignment

Step 3: *Assign coordinate values to all junctions in the region.* Coordinates are assigned relative to the reference point of the region, using knowledge of the equation of the plane in which the region is imbedded. How this equation is determined depends on the type of plane, and its interaction with rest of the drawing. There are four different situations that may arise at this stage of the process.

Case 3.1: *The region is parallel to a principal plane.* When the region is imbedded in a principal plane, we know the equation of the plane. Figure 4 shows an object with a region identified as a principle Y-plane. Therefore, all three-dimensional coordinates on this plane will have the same Y-coordinate. Given the coordinate of the junction selected as the reference point, the remaining coordinates are assigned using the lengths of the edges in the graph. In this case, one vertical line is 50 units long and the "horizontal" line is 150 units long. This assigns coordinates (0,0,50) and (150,0,0) to each of the new points relative to the reference junction. Even disconnected points, such as those surrounding the hole in the surface, can be assigned coordinates by determining the relative distance along the X-axis and the Z-axis from reference junction.

Case 3.2: *The region is parallel to an identified oblique plane.* Information gathered from processing regions in principal planes can be used to identify the equations of oblique planes. In Figure 4, the equation of the sloping surface can be determined from the coordinates assigned to the endpoints of the sloping edge. When the plane equation is known, coordinate assignment is accomplished in the same manner as the previous case. The only difference is that the coordinates are assigned with respect to the oblique plane equation.

Case 3.3: *The region is parallel to an unidentified oblique plane, but a symmetry rule may be applied.* When a region does not interact with principal planes sufficiently to identify its plane equation, the interpretation is ambiguous. At this point we use symmetry constraints to provide likely interpretations. Figures 5 and 6 show two objects where there is insufficient information to determine the coordinate of vertex x. In both situations, the possible assignments for x form a line parallel to the projection lines of the picture plane. In Figure 5, we can use symmetry to make a reasonable assumption. One region is a parallelogram and there is a strong human bias to see parallelograms as tilted rectangles. Since only one interpretation causes the region in question to be a rectangle, we can assume this interpretation and the coordinates can then be easily assigned.

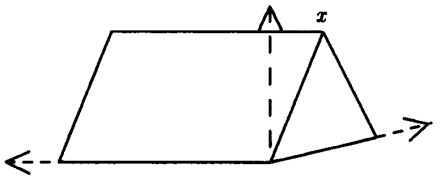

Figure 5 - Oblique Plane with Symmetry

Case 3.4: *The region orientation is unknown and no symmetry rule applies.* In Figure 6, all the visible regions are triangles. Since any three points in 3D space form a plane, there is really no way to constrain the number of interpretations so that the coordinate for x can be determined. In these cases, there is no solution except to take advantage of the interactive nature of the interface to ask the user for help in specifying the coordinate. This interaction can be done visually, by displaying extensions of two of the principal axes from each junction in question and asking the user to supply the intersection points with the third. In the figure, the user controls a line parallel to the Y-axis and moves it to specify the points y and z, which then determine the position of x. This method obtains the necessary information while retaining the visual "sketchpad" nature of the interface.

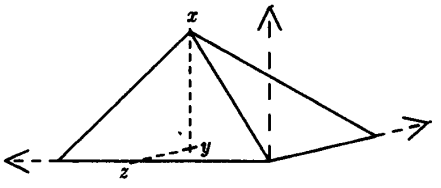

Figure 6 - Oblique Plane without Symmetry

Step 4: *Repeat Steps 1-3 until all visible junctions have been assigned coordinates.* The algorithm selects the regions in an order determined by its evaluation function. Each region is assigned coordinates according to the equation of the plane in which the region is located. As the process continues, more and more of the oblique planes are determined and, in turn, these provide more information for later evaluations. Only when presented with an area that is completely ambiguous do we stop and ask the user for some assistance. Even in this case, we can use the information at hand to phrase the query in an intelligent fashion. Once all the coordinates have been assigned to all the visible vertices, the structure analysis is complete.

Special Considerations

Disconnected Subgraphs

Some difficulties arise during the structural analysis that must be addressed before the process can continue. One such problem occurs a subgraph of the input drawing is connected solely by T-junctions, or when the subgraph is disconnected from the rest of the graph. The drawing in Figure 7 contains one example of each situation. The "bar" composed of Regions 7 and 8 is connected to the graph, but each connecting junction is a T-junction, representing an edge occluded by another surface. The hole in Region 1 is an example of a subgraph that is disjoint.

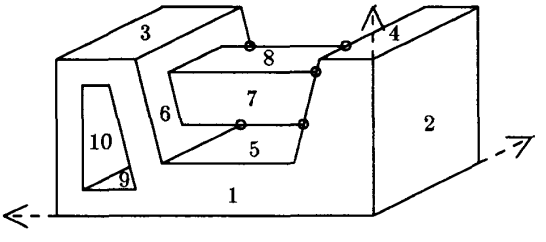

Figure 7 - Object with Disconnected Subgraphs

This type of drawing does not present a problem, but it does need to be handled carefully. Note that in each case, the object part indicated by the subgraph contacts the rest of the object at a surface, rather than an edge. It is a function of the labeling process to identify these cases and calculate which region is the point of contact. In the figure, the labeling algorithm would indicate that Regions 7 and 8 both contact Region 6 and that the hole is in the surface of Region 1. The coordinates for the points of contact can be calculated using the plane equation in the same way as any junction on the outside edge of the region. Once a coordinate is found on the subgraph relative to the rest of the graph, the process continues as before.

Coordinate Resolution Via Backtracking

A more complex problem can occur in the Step 3 of the algorithm. The assignment of a coordinate to a junction does not normally involve modifications to any other junction coordinate. An exception to this rule can occur in a particular situation. Figure 8 shows an input graph where this can occur.

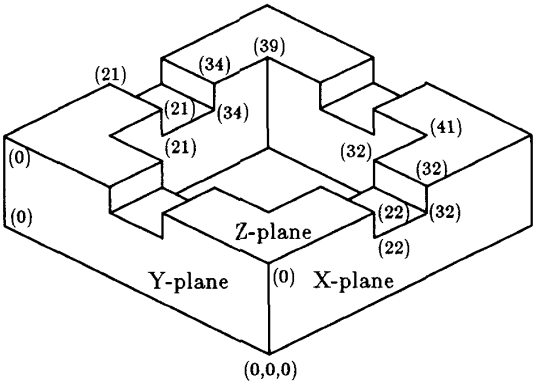

Figure 8 - Example of Object with Backtracking

Depending on the accuracy of the drawing and the order in which the regions are processed, it is possible for two junctions to be assigned coordinates that are incompatible with the equation of the plane that joins them. Note that in the figure, the coordinates of two points on the same Y-plane have different Y coordinates (39 and 41). When both of these values have already been used to generate further coordinate assignments, we have no choice but to backtrack and modify one of the coordinate values.

The backtracking algorithm is reasonably simple and it takes advantage of the fact that, as the coordinates are assigned, an additional field records where that coordinate value was derived from. Therefore, we can always work back to the junction that originally determined the coordinate value. We can then treat this junction as the root of a tree containing all the coordinates dependent on this value. When a backtracking situation occurs, the algorithm for resolving the discrepancy is as follows:

Step 1: Backtrack from the unresolved junction to the sources of both differing values. With two pointers, this can be done simultaneously. Whichever source is reached first becomes the point of change, unless that source is the origin.

Step 2: Modify the value of the source junction to resolve the discrepancy between the differing junction coordinates.

Step 3: Use a tree traversal algorithm to propagate this change to all the junctions dependent on the value that has been modified.

Though backtracking is usually an involved process that could take an exponential amount of time to resolve, that is not the case with this algorithm. First of all, the complexity of the backtracking algorithm is $O(n)$ where n is the number of junctions in the adjacency graph. This is because no junction is modified more than once during backtracking. While backtracking is not as desirable as the constant time reconciliation that usually occurs, it is not a serious inefficiency either.

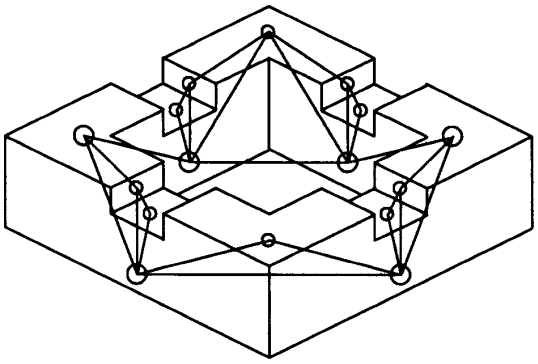

Figure 9 - Backtracking Example with Dual Graph

The conditions necessary to cause backtracking are rather complex, and not prevalent in the majority of drawings. When there is a series of junctions in a cycle, the variance of the coordinates in all three dimensions must be zero over the entire cycle. The most common junction cycle occurs around a region, but the structural analysis algorithm processes each region as a unit to prevent errors. The problem occurs when the regions are arranged so that there exists a cycle of cycles. In Figure 9, there is a representation of the dual graph for the drawing in Figure 8. Each vertex in the dual graph represents a region and the edges connect those regions that affect each other during coordinate assignment. The object in this figure has a dual graph which has a cycle of biconnected components. The same variance rule applies to this cycle as well as the simple cycles that compose it. While each of the components can be processed without difficulty, there may be a reconciliation problem when the last region in the component cycle is processed. This is resolved with a single invocation of the backtracking algorithm. While it is an unfortunate necessity, backtracking is not a significant hindrance to the interpretation process.

Structural Interpretation Example

The entire process of creating a 3D interpretation of a 2D line drawing can demonstrated with an example. This section contains a series of figures designed to show the complete process of interpretation.

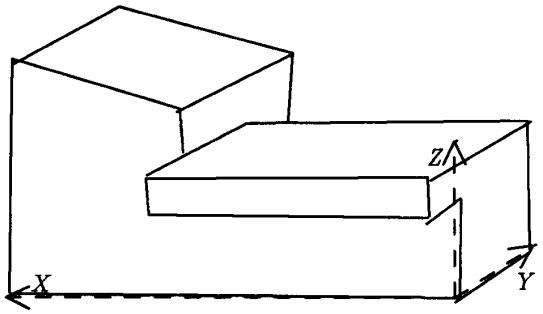

Figure 10 - Input Drawing for Graph Algorithm

The initial line drawing, shown in Figure 10, was used as input to the system. This object was chosen as an interesting, non-trivial example of a drawing that our system can process without difficulty. Notice that the original drawing contains unattached edges, misaligned vertices, and lines that are only roughly parallel. The result of the graph algorithm, with this drawing as input, is displayed in Figure 11.

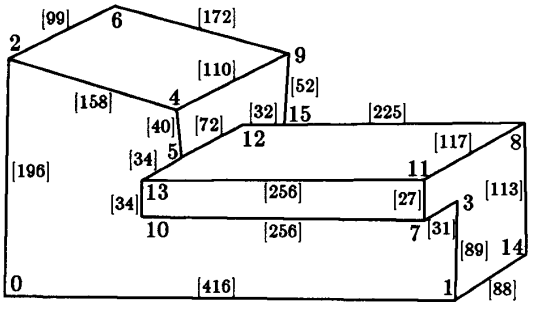

Figure 11 - Example Output of Graph Algorithm

The input drawing has been adjusted to create the corresponding adjacency graph. In this graph, all of the edges meet solidly at appropriate junctions and all vertices have been resolved. Some of the additional information that will be referenced later in the process is also displayed in this figure. This includes the number of each vertices and the length of each edge. For clarity, the edge lengths are displayed in brackets ("[" and "]").

Once the graph has been adjusted, it is processed by the labeling algorithm, with the result shown in Figure 12. The information about the graph available prior to the algorithm is still valid, but is left off this figure so that the new labeling data is clearly displayed. Each edge in the graph has been assigned a label from the set of valid Waltz labelings and each vertex has been given a category and number corresponding to an internal junction table.

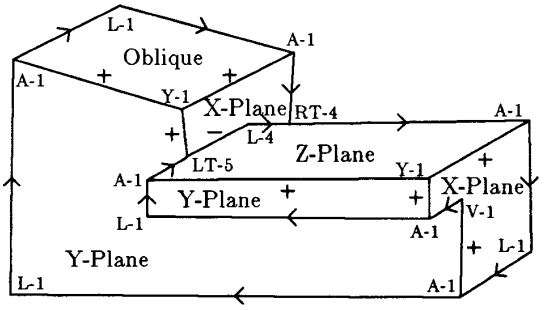

Figure 12 - Example Output of Label Algorithm

In the structural analysis phase of the process, the output of the labeling program is used as input to the algorithm. The large front region in the graph will be selected as the first region to process because it is a principle Y-plane and because of the number of vertices incident on this region. Since no other coordinates have been assigned, the lower-left vertex is selected as the reference junction of this region.

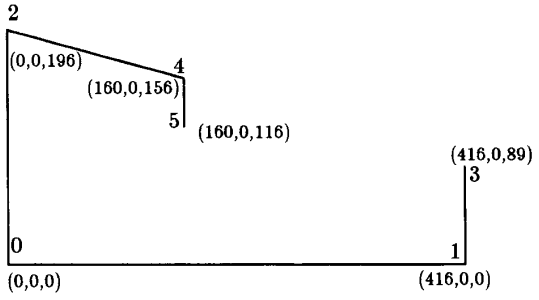

Figure 13 - Structural Analysis of First Region

Figure 13 shows what the object graph looks like after the 3D coordinates have been assigned. Note that the edge between Junctions 2 and 4 can be evaluated and given a 3D equation, now that coordinates have been assigned to its endpoints. Since this was the only unknown line equation in the drawing, the process of assigning coordinates to the rest of the vertices follows smoothly and the result can be seen in Figure 14.

One of the clever aspects of this method of coordinate assignment is that, by default, the object is adjusted to be perfectly rectangular when the principle axes indicate that this is the case. For example, the Line (4,5) in the original graph (see Figure 11) is not quite vertical. However, it is close enough to be considered parallel to the Z-axis. Therefore, when three-dimensional coordinates are assigned to the corresponding Junctions 4 and 5 in the object graph, the X and Y-coordinates are identical, forming an edge that is now perfectly vertical. Discrepancies between the lengths of edges in Figure 14 and the original line lengths in Figure 11 are the result of junction coordinate reconciliations. For example, Line (2,6) and Line (4,9) have lengths 99 and 110 respectively. But because the region formed by Vertices 2, 4, 9, and 6 forms a parallelogram (actually a rectangle) in three-dimensional space, the parallel edges must be the same length. Figure 14 shows that Edge (2,6) and Edge (4,9) both have a length of 104 along the Y-axis because the original lengths were averaged when the Edge (6,9) was added to the object graph.

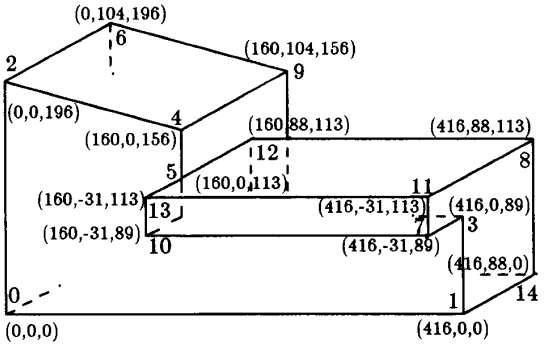

Figure 14 - Object Graph After Structural Analysis

Though the algorithm for inferring the hidden structure of the object is beyond the scope of this paper, a brief explanation is in order for completeness with this example. Based on knowledge of the junctions, the starting point of each known hidden edge is generated. These are analyzed for intersection points with other hidden lines. When two or more lines intersect and the corresponding junction is valid, this junction and its edges are added to the object graph. Any new hidden lines generated as a result of this new junction are added for consideration. For example, the first discovered intersection occurs at (0,135,0) where the hidden edges associated with Junction 0 and Junction 6 meet and form Junction 15. This process continues until the entire hidden structure of the object is determined. The final pro-

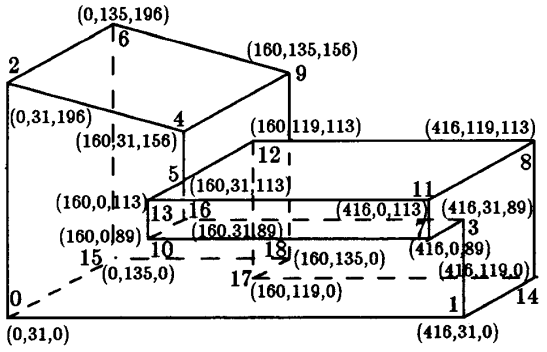

Figure 15 - Example Output of Interpretation System

duct is a graph of the 3D structure of the object, complete with valid coordinate assignments for each junction. The final result is shown in Figure 15.

Conclusion

There are several ways in which we are looking to extend the power of this interpretation system. Some of these include technical considerations, such as improvements to the user interface. Other possibilities concern refinements to the existing algorithms. One example would be a more complete system of symmetry rules to use when making assumptions about unidentified regions. Of course, the most obvious modification would be to increase the types of objects that can be processed. Currently, the system is capable of handling a reasonable subset of regular polyhedra without any special interaction from the user. In general, more complex structures need interactive user input during interpretation. This is to be expected, since in general, a polyhedral line drawing may have several degrees of freedom [9]. We are also developing methods of processing curved surfaces, based on the work of Malik [7], Stevens [8], Brady and Yuille [2], and Lee, et al [6].

One idea for processing curves would be to break each curve into an arc equal to or less than 90 degrees. Each arc can then be replaced by its tangent lines. The result is an object that can be processed by regular polyhedral methods. Once the 3D structure is determined, the arcs can be replaced to provide the proper interpretation. This method would be useful for processing cylinders and rounded surfaces, but may not be sufficient to handle more complex solids of revolution.

There are many aspects to this system of 3D object interpretation that cannot be explained in a paper of this length. Both the graph and labeling algorithms have interesting details that are not normally included in the standard routines for these tasks. There are also a number of points about the user interface, especially the interactive modules, that are essential to the usefulness of this system. In addition, the entire process of inferring the hidden structure of the object has been left out. This algorithm is similar in its complexity to the structure analysis just presented. Though this is a rather brief treatment of the work we have developed to date, it should indicate that we have a viable system for specifying 3D structure from a rough 2D sketch.

Acknowledgement

The authors wish to acknowledge the assistance of Dr. Arie Kaufman during the research and preparation for this paper and its accompanying presentation.

References

[1] H.G. Barrow and J.M. Tenenbaum, "Interpreting line drawings as three-dimensional surfaces," *Artificial Intelligence*, vol. 17, pp. 75-116, 1981.

[2] M. Brady and A. Yuille, "An extremum principle for shape from contour," *Proceedings of IJCAI-8*, pp. 969-972, Karlsruhe, 1983.

[3] P.M. Hanrahan, "Creating volume models from edge-vertex graphs," *Computer Graphics*, vol. 16-3, pp. 77-84, July 1982.

[4] D.A. Huffman, "Impossible objects as nonsense sentences," in *Machine Intelligence 6*, ed. R. Meltzer and D. Michie, pp. 295-323, Edinburgh University Press, Edinburgh, 1971.

[5] K. Kanatani, "The constraints on images of rectangular polyhedra," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 456-463, July 4, 1986.

[6] S.J. Lee, R.M. Haralick, and M.C. Zhang, "Understanding objects with curved surfaces from a single perspective view of boundaries," *Artificial Intelligence*, vol. 26, pp. 145-169, 1985.

[7] J. Malik, "Interpreting line drawings of curved objects," *International Journal of Computer Vision*, vol. 1-1, pp. 73-103, 1987.

[8] K.A. Stevens, "The visual interpretation of surface contours," *Artificial Intelligence*, vol. 17, pp. 47-73, 1981.

[9] K. Sugihara, *Machine Interpretation of Line Drawings*, M.I.T. Press, Cambridge, Mass., 1986.

[10] D.L. Waltz, "Generating semantic descriptions from drawings of scenes with shadows," in *The Psychology of Computer Vision*, ed. P. Winston, pp. 19-92, McGraw-Hill, New York, NY, 1972.