Smart Sketch System for 3D Reconstruction Based Modeling

Ferran Naya[1], Julián Conesa[2], Manuel Contero[1], Pedro Company[3], Joaquim Jorge[4]

[1] DEGI - ETSII, Universidad Politécnica de Valencia, Camino de Vera s/n,
46022 Valencia, Spain
{mcontero, fernasan}@degi.upv.es
[2] DEG, Universidad Politécnica de Cartagena, C/ Dr. Fleming
30202 Cartagena, Spain
julian.conesa@uptc.es
[3] Departamento de Tecnología, Universitat Jaume I de Castellón, Campus Riu Sec
12071 Castellón, Spain
pcompany@tec.uji.es
[4] Engª. Informática, IST, Av.Rovisco Pais
1049-001 Lisboa, Portugal
jorgej@acm.org

Abstract. Although CAD systems have evolved considerably in functionality, expressiveness and modeling power over the last decades, their user interfaces are still not suited to the initial stages of product development. While much of this functionality may be required by the sheer complexity of the tasks these systems are designed to help, we believe the user interface could benefit from simpler paradigms based on sketching and drawing to reduce unneeded complexity, especially in the conceptual design phase. In what follows, we present the CIGRO system, which provides a reduced instruction set calligraphic interface to create polyhedral objects using an incremental drawing paradigm evocative of paper and pencil drawings. Users draw lines on an axonometric projection, which are automatically beautified and connected to existing elements of the drawing. Such line drawings are then converted into a three-dimensional model in real time through a reconstruction process based on an axonometric inflation method, providing a smooth switch between 2D sketching and 3D view visualization.

1 Introduction

In spite of large strides made by commercial CAD applications in terms of descriptive power, flexibility and functionality, their user interfaces are still by and large constrained by the WIMP (Windows, Icons, Menus and Pointing) paradigm, which severely hinders both ease of learning and ease of use. Recently, work on sketch-based modeling has looked at a paradigm shift to change the way geometric modeling applications are built, in order to focus on user-centric systems rather than systems that are organized around the details of geometry representation.

There are two main approaches to sketch-based modeling. Some of this work stems from early attempts at shape interpretation in computer vision. While most of the activity in this area in the past has been focused in off-line algorithms, the growing focus on sketches and modeling has brought forth a new emphasis on approaches geared towards interactive applications. To this end, the aim of our research is to develop expeditious ways to construct geometric models. We want to generate solid and surface models from two-dimensional freehand drawings, using a digitizing tablet and a pen, an approach we have termed calligraphic interfaces. These rely on interactive input of drawings as vector information (pen-strokes) and gestures, possibly coupled with other interaction modalities. We want to keep the number of low level inter-actions to a minimum, as well as the command set (number of different gestures/strokes the user has to enter). The present text describes work at the user interface towards integrating a three-dimensional reconstruction approach into an interactive working environment, through sketch input. This environment differs markedly from previous approaches in that the speed of execution and timely feedback are more important than the ability to produce models from vectorized bitmaps in one step, as has been typical of previous efforts in computer vision.

2 Related Work

The new generation of calligraphic applications uses gestures and pen-input as commands [1, 2, 3]. In contrast to conventional drawing applications, the stylus can also be used to enter continuous-mode sketches and freehand strokes. Thus, there is a growing research interest in using freehand drawings and sketches as a way to create and edit 3D geometric models. Within this research area we can distinguish two approaches. One method relies on gestures as commands for generating solids from 2D sections and the second one, derived from computer vision, uses algorithms to reconstruct geometric objects from sketches that depict their two dimensional projection.

An example of gestural modeling is Sketch [4]. The geometric model is entered by a sequence of gestures according to a set of conventions regarding the order in which points and lines are entered as well as their spatial relations. Quick-Sketch [5] is a system oriented to mechanical design that consists of a 2D drawing environment based on constraints. From these it is possible to generate 3D models through modeling gestures. Teddy [6] allows free-form surface modeling using a very simple interface of sketched curves, pockets and extrusions. Users draw silhouettes through a series of pen strokes and the system automatically proposes a surface using a polygonal mesh whose projection matches the object contour. GIDeS [7] allows data input from a single-view projection. In addition the dynamic recognition of modeling gestures provides users with contextual menus and icons to allow modeling using a reduced command set.

The second approach, which we call geometric reconstruction, uses techniques based on computer vision to build three-dimensional geometric shapes extracted from two-dimensional images representing some kind of projection. The systems we surveyed use two main techniques. The first is based on Huffman-Clowes labeling scheme [8], [9]. The second approach treats reconstruction as an optimization problem [10]. This enables us to obtain what, from the point of view of geometry, is unrealizable: a three-dimensional model from a single projection. However, from the psycho-logical point of view it is well known that humans can identify 3D models from 2D images by using a simple set of perceptual heuristics [11]. Thus, geometric reconstruction, when cast as a problem of perception, can be described in terms of mathematical optimization. To this end reconstruction applications have been developed by authors such as Marill [12], Leclerc [13], Fischler and Lipson [14]. Other examples are Digital Clay [15], which supports basic polyhedral objects, and, in combination with a calligraphic interface for data input, uses Huffman-Clowes algorithms to derive three-dimensional geometry and Stilton [16], where a calligraphic interface is directly implemented in a VRML environment, and the reconstruction process uses the optimization approach based on genetic algorithms.

3 Overview of Operations

In contrast to surveyed work, our application provides an integrated 2D-3D environment, where users sketch and can immediately switch the point of view and see the corresponding 3D model. This real-time sketching experience is supported by implementing a minimal gesture alphabet, automatic line drawing beautification and a fast and robust axonometric inflation engine.

The objective of our system is to provide support for the preliminary phases of design, where it is not necessary to build complex CAD models but to express our visual thinking in a fast way. In this context, engineers usually employ sketches as their natural tool to express design concepts. So we have chosen a compromise between geometric complexity and fast interaction. For this reason, at this moment only polyhedral models are supported in order to be able to implement a fast axonometric inflation algorithm to reconstruct them.

Previous reconstruction-based applications include a preliminary offline 2D reconstruction stage, where the input sketch is adjusted before proceeding with the 3D reconstruction. To implement a true interactive sketching system we have developed an automatic online 2D reconstructor that operates in real time. Thus, whenever the input sketch is changed, because edges are added or deleted, the sketch is adjusted and a 3D model is automatically built and offered to the user for review. The only constraint on drawing is that the reconstruction procedure requires a single orthogonal axonometric projection of the model as input. These are the single-view representations most commonly used in engineering drawings of three-dimensional parts.

3.1 Gesture Analyzer

The gesture analyzer processes strokes generated by the user directly on the screen of a Tablet-PC or LCD tablet. The user interface is designed to minimize (we call it 'minimalist interface') the interaction with the system in an attempt to approximate it to the traditional use of pen and paper. We want the user to concentrate on sketching instead of having to navigate through menus or search for icons on toolbars. For this reason the application automatically generates the corresponding three-dimensional geometric model, and it does not need any additional information from the user. The system updates the 3D model as the user refines the geometry, so users can see the evolution of the model whenever any change in the input sketch takes place.

The gesture set used by the interface is reduced to the following commands: new edge, new auxiliary edge and remove edge (auxiliary or not). These gestures are very similar to those used when drawing on paper. The application is, therefore, in charge of recognizing the type of stroke drawn by the user and of trying to capture the designer's intention (parallelism, proximity, close points).

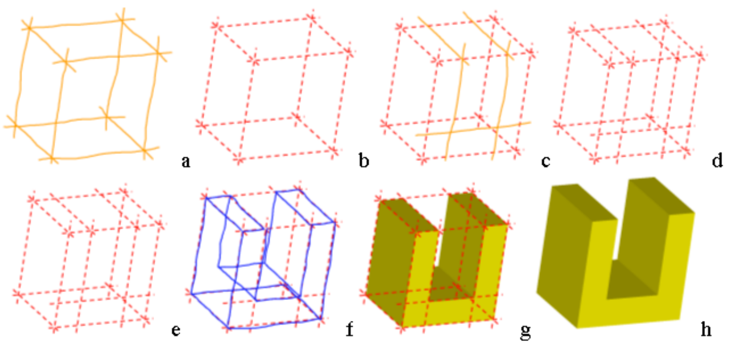

Fig. 1. Example showing the sequence of actions extracted from CIGRO. The orange color in a represents raw strokes corresponding to auxiliary lines. Cyan dashed lines in b and d represent adjusted auxiliary lines generated by the gesture analyzer. Blue raw strokes in f correspond to real geometry that is snapped to auxiliary lines in e, which provides the solid model seen in h. All of these representations can be switched on or off by means of a visualization toolbar.

The application provides both real geometry and auxiliary lines. This emulates an extended practice for making sketches. First, the user draws a set of auxiliary lines (see Figure 1.a to e) to define the main geometric features of the object and then, using this skeleton as a drawing template, the designer refines the sketch by applying pressure with the pencil and drawing over the previous template (see Figure 1.f). Our application takes into account the pressure made on the pencil. In this way it distinguishes auxiliary constructions from real geometry lines by applying a pressure level threshold (configured by the user). Auxiliary strokes tend to serve as references (constraints) for the intended model geometry strokes. This is implemented in the application using the Wintab API (http://www.pointing.com), an open industry interface that directly collects pointing input from a digitizing tablet and passes it to applications in a standardized fashion. This API makes it possible to retrieve the pressure the user applies at each point of the stroke over the tablet. This information is used by the application to distinguish between auxiliary entities (less pressure) and those of the model (higher pressure). Raw strokes provided by Wintab API are processed by the CALI library [3], which provides some components to develop calligraphic interfaces. It is based on a recognizer of elemental geometric forms and gestural commands that operates in real time using fuzzy logic. The recognized gestures are inserted in a list in order of the degree of certainty, and returned to the application. CALI recognizes the elemental geometric shapes, like triangles, rectangles, circles, ellipses, lines, arrows, etc., and some gestural commands, such as delete, move, copy, etc. At the current development level, the CIGRO application only supports sketched segments which can be recognized as entities of the "line" type or as a gestural command of the "delete" class. Users are allowed to generate faces drawing a sequence of straight edges. It is not relevant in which order edges are drawn, since the three-dimensional reconstruction method only looks at connectivity and perceptual constraints. In this way we try to afford a freedom in sketching mimicking that of pencil and paper. Edges and segments can be removed using a scratching gesture. This not only allows errors to be corrected but also enables more complicated shapes to be drawn from “simpler” forms, a term familiar to draftspersons used to sketching. When drawing a scratch gesture, the application detects the edge(s) that the user wants to delete as being those intersecting the smallest quadrilateral enclosing the scratching gesture (see Figure 2).

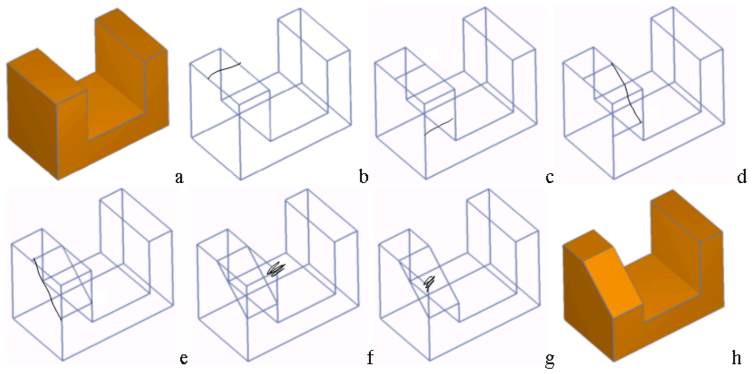

Fig. 2. Refining geometry from simple shapes

This can be used to incrementally derive more complex shapes from simpler ones, as illustrated in Figure 2, where users can construct shapes by either adding new edges or removing existing ones. The interface cooperates with the user by not requiring the presence of valid models at all times. Instead it allows geometric models which are consistent only at the face level.

3.2 Line Drawing Beautification and Snaps

Previous reconstruction-based applications usually include an offline 2D reconstruction stage, where the input sketch is adjusted. In our system we propose an online 2D reconstruction [17]. Online reconstruction provides an immediate feedback to the user, because it operates as the user draws the sketch, and it offers better integration in the calligraphic interface. This concept of 2D reconstruction is similar to drawing beautification proposed by Igarashi [6], [18].

The aim of this stage is to adjust drawing entities provided by the CALI recognizer to be used at the 3D reconstruction stage. To obtain functional input data for 3D reconstruction, we need to clean up input data and adjust edges to make sure they meet precisely at common endpoints in order to get geometrically consistent figures which can then be used for generating 3D models at the next stage. The “beautification” process has to filter all the defects and errors of the initial sketches which are inherent to their inaccurate and incomplete nature. At present, the stage of 2D reconstruction receives geometric shapes of the “line” or “auxiliary line” type as input data.

In order to provide an adequate database for the 3D geometric reconstructor, the application provides support for the following drawing aids: automatic line slope adjustment, vertex point snap and vertex on line snap. The first drawing aid consists of checking whether the new line is parallel to any of the principal axes of the sketch by considering a slope tolerance. In the case that the straight line is nearly parallel to one axis, then we adjust one or both endpoints so that the resulting line is now precisely parallel to one of the three main axes. The second analysis looks for vertices close to the line endpoints, again taking into account a vertex proximity tolerance. Should there be several such vertices, we select the one closest to that line endpoint. For endpoints of the new line which do not lie close to a model vertex, the system analyzes whether the points are close to an existing edge, taking into account a given edge proximity tolerance. If several edges match this criterion, we select the edge which lies closest to the given endpoint.

Snapping capability provides the user with continuous feedback, because it is performed in real time. Other previous systems perform these analyses offline, when the user has finished sketching and before launching 3D reconstruction. We provide a general tolerance control to soften the beautification action, because some users prefer a less automatic drawing control. After this stage, all the 2D image data are stored in a database consisting of a list of vertices and a list of edges.

3.3 Axonometric Inflation Engine

In order to provide a real-time experience, we have implemented a fast reconstruction algorithm whose clearest antecedents are presented by Sugihara [19], [20], Lamb and Bandopadhay [21]. Other reconstruction strategies based on optimization or labeling algorithms have been discarded because they are not fast enough. For this reason we have restricted the shape domain supported by CIGRO to quasi-normalon objects. We use an extension to 3D of Dori's definition of a normalon [22] as "a polyhedral object having the property that the angle between any two of its adjacent edges is 90°". Quasi-normalon typology corresponds to objects that can be reduced to normalon by deleting all edges running non-parallel to the three main directions without losing any vertices in the model. We can distinguish two classes of quasi-normalon objects:

- Class 1: we get a connected graph after simplification.
- Class 2: we get a not connected graph after simplification.

Let us suppose an orthogonal projection of an orthogonal corner like A'B'C'D' (see Figure 3.a). To reconstruct the model (i.e. to determine x, y and z coordinates of vertices A, B, C and D) the Cartesian coordinate system associated to inflation is used to trivially obtain x and y coordinates of all four vertices ($x_A = x_{A'}$, $y_A = y_{A'}$, ...) and the z coordinate of central vertex D can be arbitrarily fixed without loss of generality. Then, axonometric inflation makes use of the formulations that determine the angle between every edge (CD) and its own projection (C'D') when three orthogonal edges are connected to the same vertex (D). Next, relative z coordinates of neighbor vertices (A, B, C) are obtained from the z coordinate of central vertex (D) using Eq. 1 where $L_{C'D'}$ is the length of line segment C'D' (CD edge projection), and Z_C and Z_D are the respective z coordinates of the lateral and the central vertex that determine the reconstructed edge

$$z_C = z_D \pm L_{C'D'} \cdot \tan(\mathrm{asin}(\sqrt{\mathrm{cotg}(\alpha)\cdot \mathrm{cotg}(\beta)})) \qquad (1)$$

However, since there is no single solution for equation (1) (see Figure 3.b), the method can only be applied to normalon and quasi-normalon-type models, where the fact that their corners have to be orthogonal enables us to set this as a criterion in order to select the correct solution.

To determine the z coordinates of all vertices in a normalon polyhedron, a Kruskal algorithm is used to obtain a spanning tree formed by all edges connecting the successive central vertices. The lateral vertex connected through the longest edge is defined as the central vertex for the new corner because it is assumed that longer edges are less prone to error than shorter ones are (i.e. the relative dimensional errors are lesser, and long lines are more accurately drawn in drawing by hand). The process is repeated until all vertices in the model have been determined. In cases in which converting any of the present lateral vertices into central ones would generate a circuit (i.e. if all lateral vertices have been previously visited) the branch is abandoned and the longest as yet unexplored lateral edge is used to begin a new branch. The algorithm requires the concurrence of three orthogonal edges in each central vertex. Nevertheless, information about faces is not required and the restriction applies to central vertices only. The valence (the number of edges concurring in a vertex) of lateral vertices is irrelevant. Consequently, the approach will work if a spanning tree can be obtained where all central vertices have a valence of three, and, thus, all vertices of other valences can be determined as laterals.

In this way axonometric inflation can be applied to the class 1 quasi-normalon models where the temporary elimination of all line-segments that are non-parallel to any of the three main directions determines an equivalent normalon if no vertex is deleted and the graph remains connected.

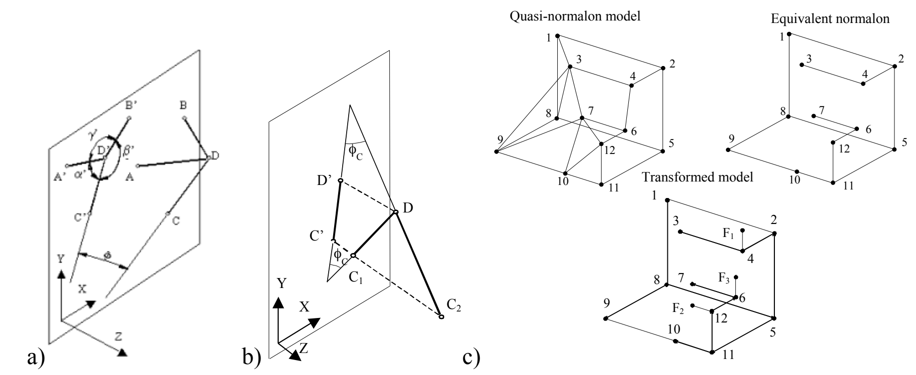

Fig. 3. a) Inflation method b) Relationship between angles in model and projection c)Transformation of a quasi-normalon model

3.3.1 Managing Class 1 Quasi-normalon Objects

Sometimes, in spite of getting an equivalent normalon, axonometric inflation cannot be applied if certain vertices are not accessible through some valid spanning tree. This is the case when some junctions are only connected to other junctions with a valence below three. For instance, in Figure 3.c, vertex 3 is only connected to vertex 4, which cannot be central because its valence is two. When the valence of a central vertex is lower than three, equation (1) cannot be used. Nevertheless, the assumption of the model to be a normalon has already been made. Hence, adding fictitious line-segments is coherent with the assumption and solves the problem. These fictitious line-segments are defined by unit length and oriented in accordance with those main directions still not present in the vertex. In Figure 3.c, fictitious edges 4-F1, 12-F2 and 6-F3 allow vertices 3, 6 and 7 to be determined. When vertices with a valence above three appear, the approach will still work if a valid spanning tree can be obtained; i.e. a tree where those vertices are not central. When this is not possible, we have limited ourselves to obtaining one of the potential models by randomly choosing three of the edges that converge in the vertex. But, moreover, for the method to be able to spread throughout the whole graph that defines the projection of the model, this graph must be connected. However, what sometimes occurs is that when the equivalent normalon of a quasi-normalon-type of model is obtained, the resulting graph is not connected resulting a class 2 object.

3.3.2 Managing Class 2 Quasi-normalon Objects

In these cases, and if the user has not drawn them, our system generates enough automatic auxiliary lines to allow the reconstruction of the model. This algorithm essentially consists of joining the graphs that have been isolated as a consequence of their being converted into equivalent normalons. Nevertheless, the graphs cannot be joined in a random fashion and the operation must comply with the following points:

1. The connecting edges must be parallel to the directions that define the orthogonal trihedron (in order to ensure that the axonometric inflation method can spread).
2. The connecting edges and the vertices that are used to connect the graphs must be coplanar.

The first operation for determining the automatic auxiliary lines is to select a connecting edge that links the independent graphs resulting from the transformation into equivalent normalon. The selection is made at random from among the set of candidate edges, although this does not mean losing the generality of the method. The second step is to replace the connecting edge with two edges that are parallel to the orthogonal trihedron, thus giving rise to a new vertex on the initial graph. Our problem is centered on how to select which two directions to take from the three offered by the orthogonal trihedron. The choice of directions is conditioned by the coplanarity criterion that means we have to detect faces from the information contained in the image and to do so we make use of an adapted version of Courter and Brewer's [23] algorithm. The process begins by detecting a face that contains the connecting edge and at least two edges that run parallel to two of the directions belonging to the main trihedron. This is a necessary and adequate condition for the connecting edge to be replaced. Once the face has been determined, the replacement process is performed in accordance with the following criteria:

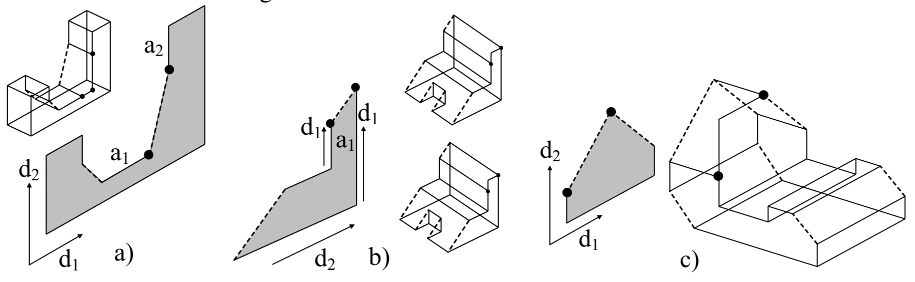

Fig. 4. Examples for the substitution of connecting edges

- If edges that run parallel to two dissimilar directions of the trihedron concur at the vertices of the connecting edge, all that needs to be done is to prolong the converging edges until they intercept each other (see Figure 4.a)
- If edges running parallel to a same direction in the orthogonal trihedron concur at the vertices of the connecting edge, the direction running parallel to the trihedron that does not concur at the connecting edge is used to determine the new vertex. In these cases there are two possible valid configurations. The random choice of one of them does not affect the generality of the method (see Figure 4.b).
- If, at any of the vertices of the connecting edge, none of the edges running parallel to the directions of the dihedron concur, then an edge running parallel to the direction of the trihedron that does not exist in the other vertex of the connecting edge will be drawn in (see Figure 4.c).

The last step in the automatic auxiliary lines generation consists of a simple check of the type of model, which is performed each time a connecting edge is replaced by a pair of edges running parallel to the orthogonal trihedron. After this check the following steps are taken:

- If the new model is connected, it is reconstructed.
- If the model is not yet connected, first step is executed repeatedly for as long as there are still connecting edges in the model.

4 Conclusions and Future Work

We have described an approach to input graphical quasi-normalon shapes using methods based on image processing. We plan to use this method as a foundation to shape input methods in current CAD systems. Instead of focusing on off-line algorithms, we aim at developing expeditious ways to construct geometric models through calligraphic interfaces. Our approach includes a three-dimensional reconstruction approach integrated in an interactive editor, using sketch input. This environment differs from previous approaches in that the speed of execution and timely feedback are more important than the ability to produce models from vectorized bitmaps in one step, as has been typical of previous efforts in computer vision. The cost for this is restricting the allowed shapes to a restricted set (quasi-normalons). However, using such shapes in conjunction with construction lines provides a sound basis for more sophisticated input techniques, including curves and surfaces. This will be approached in the future. Preliminary usability tests have shown encouraging results.

References

1. Rubine, D.: Combining gestures and direct manipulation. Proceedings ACM CHI'92 Conference Human Factors in Computing Systems (1992) 659-660
2. Long, A.C., Landay, J.A., Rowe, L.A., Michiels, J.: Visual Similarity of Pen Gestures. Proceedings of Human Factors in Computer Systems (SIGCHI), (2000) 360-367
3. Fonseca, M., Jorge, J.: Experimental Evaluation of an On-Line Scribble Recognizer. Pattern Recognition Letters, **22** (12), (2001) 1311-1319
4. Zeleznik, R.C., Herndon, K.P., Hughes, J.F.: SKETCH: An interface for sketching 3D scenes. SIGGRAPH'96 Conference Proceedings (1996) 163-170
5. Eggli, L., Hsu, C., et al.: Inferring 3D Models from Freehand Sketches and Constraints. Computer-Aided Design, **29** (2), (1997) 101-112
6. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A Sketching Interface for 3D Freeform Design. ACM SIGGRAPH99 Conference Proc. (1999) 409-416
7. Pereira, J., Jorge, J., Branco, V., Nunes, F.: Towards calligraphic interfaces: sketching 3D scenes with gestures and context icons. WSCG'2000 Conference Proc. Skala V. Ed. (2000)
8. Huffman, D.A.: Impossible Objects as Nonsense Sentences. In: Meltzer B., Michie D. (eds.) Machine Intelligence, No. 6, Edimburgh UK. Edinburgh University Press (1971) 295-323
9. Clowes, M.B.: On Seeing Things. Artificial Intelligence, **2**, (1971) 79-116
10. Wang, W., Grinstein, G.: A Survey of 3D Solid Reconstruction from 2D Projection Line Drawing. Computer Graphics Forum, **12**(2), (1993) 137-158
11. Hoffman, D.: How we create what we see. Visual Intelligence, Norton Pub., **2**, (2000)
12. Marill, T.: Emulating the Human Interpretation of Line-Drawings as Three-Dimensional Objects. International Journal of Computer Vision, **6**(2), (1991) 147-161
13. Leclerc, Y., Fischler, M.: An Optimization-Based Approach to the Interpretation of Single Line Drawing as 3D Wire Frames. Int. Journal of Computer Vision, **9**(2), (1992) 113-136
14. Lipson, H., Shpitalni, M.: Optimization-Based Reconstruction of a 3D Object from a Single Freehand Line Drawing. Computer Aided Design, **28**(8), (1996) 651-663
15. Schweikardt, E., Gross, M.D.: Digital Clay: deriving digital models from freehand sketches. Automation in Construction, **9**, (2000) 107-115
16. Turner, A., Chapmann, D., and Penn, A.: Sketching space. Computers & Graphics, **24**, (2002) 869-879
17. Oh, B.S., Kim, C.H.: Progressive 3D Reconstruction from a Sketch Drawing. In Proceedings of the 9th Pacific Conf. on Computer Graphics and Applications, (2001) 108 -117
18. Igarashi, T., Hughes, J.F.: A Suggestive Interface for 3D Drawing. 14th Annual Symposium on User Interface Software and Technology, UIST'01, Orlando, Fl., (2001) 173-181
19. Sugihara, K.: Interpretation of an axonometric projection of a polyhedron. Computers & Graphics, **8**(4), (1984) 391-400
20. Kanatani, K.: The Constraints on Images of Rectangular Polyhedra. IEEE Transactions on Pattern Analysis and Machine Intelligence, **8** (4), (1986) 456-463
21. Lamb, D., Bandopadhay, A.: Interpreting a 3D Object from a Rough 2D Line Drawing. Proceedings of Visualization´90, (1990) 59-66
22. Dori, D.: From Engineering Drawings to 3D CAD Models: Are We Ready Now?. Computer Aided Design, **27** (4), (1995) 243-254
23. Courter, S.M., Brewer J.A.: Automated Conversion of Curvilinear Wire-Frame Models to Surface Boundary Models. Comm. of ACM, **20**(4), (1986) 171-178