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
{fernasan, mcontero}@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. Current user interfaces of CAD systems are still not suited to the initial stages of product development, where freehand drawings are used by engineers and designers to express their visual thinking. In order to exploit these sketching skills, we present a sketch based modeling system, which provides a reduced instruction set calligraphic interface to create orthogonal polyhedra, and an extension of them we named quasi-normalons. Our system allows users to draw lines on free-hand axonometric-like drawings, which are automatically tidied and beautified. These line drawings are then converted into a three-dimensional model in real time because we implemented a fast reconstruction process, suited for quasi-normalon objects and so-called axonometric inflation method, providing in this way an innovative integrated 2D sketching and 3D view visualization work environment.

1 Introduction

User interfaces of most CAD applications are based on the WIMP (Windows, Icons, Menus and Pointing) paradigm, that is not enough flexible to support sketching activities in the conceptual design phase. 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. 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.

The present text describes work at the user interface, where the main objective is keeping the number of low level interactions to a minimum, as well as the command set, in order to provide support for the preliminary phases of design, where it is not necessary to build complex CAD models but to express the visual thinking in a fast way. This implies to choose a compromise between geometric complexity and fast interaction. This is accomplished by working in a particular geometric domain (quasi-normalon objects) that opens the possibility to implement three-dimensional reconstruction algorithms that operates in real-time. This environment differs from previous sketch systems 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

There are two main approaches to sketch-based modeling. The first one is based in the calligraphic interface concept that uses gestures and pen-input as commands [1, 2, 3]. These systems rely on gestures as commands for generating solids from 2D sections. Some *gestural modeling* systems are:

- 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]: 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]: oriented to 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*, derives from computer vision, uses algorithms to reconstruct geometric objects from sketches that depict their two dimensional projection. The systems we surveyed use two main techniques. The first is based on Huffman-Clowes labeling scheme [8], [9]. The second approach deals with 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, for the psychologists it is well known that humans can identify 3D models from 2D images by using a simple set of perceptual heuristics [11]. Authors such as Marill [12], Leclerc and Fischler [13], and Lipson and Shpitalni [14] provide interesting references in the development of this field. Some recent 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.
- 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, we have called CIGRO (**C**alligraphic **I**nterface for **G**eometric **R**econstructi**o**n), provides an integrated 2D-3D environment, where users sketch and can immediately switch the point of view and see the corresponding 3D model. Real-time sketching is supported by implementing a minimal gesture alphabet, automatic line-drawing beautification, and a fast and robust axonometric-inflation engine. 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 algorithm requires a single orthogonal axonometric projection of the model as input. The user interface is designed to minimize the interaction with menus or icons in an attempt to emulate the traditional use of pen and paper.

3.1 Gesture Alphabet

CIGRO processes strokes generated by the user directly onto the screen of a Tablet-PC or LCD tablet, supported by 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 allows retrieving additional information as the pressure the user applies at each point of the stroke over the tablet. Raw strokes are then 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, where they are ordered according to the degree of certainty, and returned to the application. CALI recognizes the elemental two-dimensional 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. Using a pressure threshold defined by the user, line strokes are classified as real or auxiliary line. The gesture alphabet is reduced to the following commands: new edge, new auxiliary edge and remove edge (auxiliary or not).

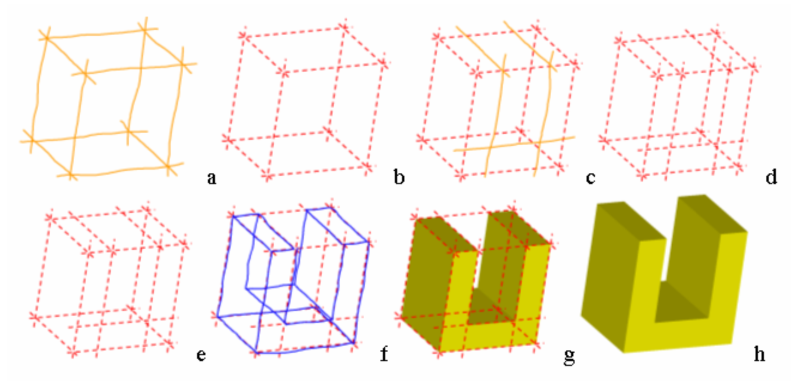

Fig. 1. Modeling sequence in CIGRO. Orange color represents raw strokes corresponding to auxiliary lines. Cyan dashed lines represent adjusted auxiliary lines. Blue raw strokes in f correspond to real geometry that is snapped to auxiliary-lines skeleton in e.

3.2 Sketching procedure

Many engineers draw a set of auxiliary lines (see Figure 1.a to 1.e) to define main geometric features of objects and then, using this framework as a drawing template, the sketch is refined by applying pressure with the pencil and drawing over the previous template (see Figure 1.f). CIGRO supports this sketching paradigm allowing oversketching of real geometry lines over auxiliary lines, which are intented to provide a set of references for snapping purposes.

Users 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. As soon as the system detect a complete face it is shaded and presented to the user (working in shaded mode).

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 by refining "simpler" forms as illustrated in Figure 2. 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.

3.3 Line Drawing Beautification and 3D Sketchy Wireframe

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] and it is aimed to adjust drawing entities provided by the CALI recognizer to be used at the 3D reconstruction stage.

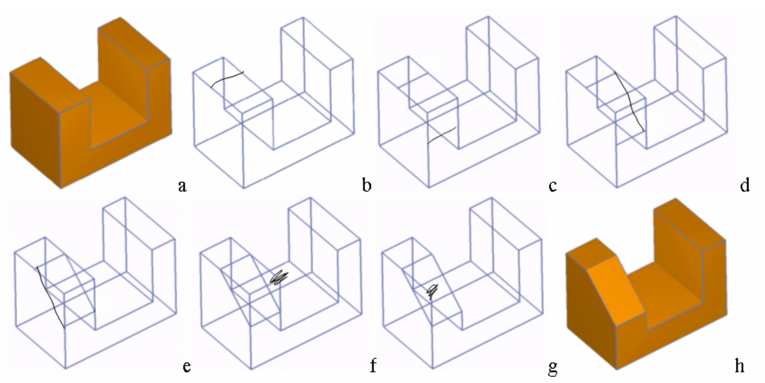

Fig. 2. Refining geometry from simple shapes

Correct work of the 3D reconstruction algorithm needs to clean up input data and adjust edges to make sure they meet precisely at common endpoints in order to get geometrically consistent figures. The "beautification" process has to filter all defects and errors of the initial sketches that 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 supports the following drawing aids:

- Automatic line slope adjustment: consists of checking whether the new line is parallel to any of the principal axes of the sketch by considering a slope tolerance. In case where the straight line is nearly parallel to one axis, then we adjust one or both endpoints so that the resulting line is now parallel to one of the main axes.
- Vertex point snap: it 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.
- Vertex on line snap and automatic line breaking: 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 that lies closest to the given endpoint. Then, the selected edge is broken at the contact point; in order to easy later delete operations, like in figure 2.e.

Snapping and adjustments are performed in real time. Other previous systems perform these analyses offline, when the user has finished sketching and before launching 3D reconstruction. A general tolerance parameter controls the beautification action, because some users prefer a less automatic drawing control. After this stage, all the 2D data are stored in a database consisting of a list of vertices and a list of edges.

The user can change the point of view and shading mode at all moment. There are three visualization modes: wireframe (fig. 3.b), shaded (fig. 3.c) and sketchy wireframe (fig 3.a and 3.d). This last mode is used to provide a hand-made look to the reconstructed model, and it is generated simply adding the z coordinate obtained from the reconstruction process in the raw 2D original sketch.

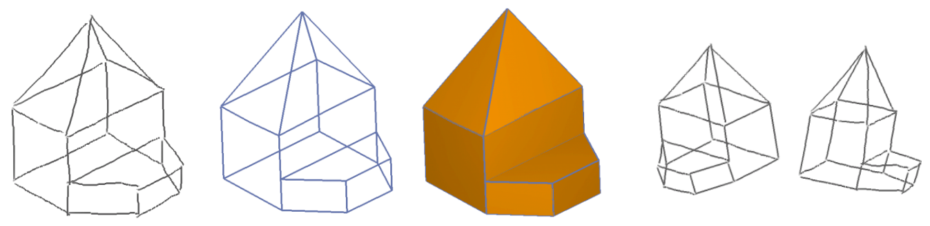

Fig. 3. Visualization modes in CIGRO

3.3 Axonometric Inflation Engine

To provide a real-time experience, we have implemented a fast reconstruction algorithm whose clearest antecedents are presented by Sugihara [19], Kanatani [20], and Lamb and Bandopadhay [21]. The shape domain supported by Axonometric Inflation is restricted to normalon objects. Where a *normalon* is rectangular trihedral polyhedron, whose corners consist of three mutually perpendicular edges (this name is an extension to 3D of Dori's definition of a normalon [22]). Other reconstruction strategies, with a wider universe of objects, have been discarded because they are not fast enough, since they rely on optimization or labeling algorithms.

Our implementation is similar to Kanatani´s, and details on the approach can be found in [20]. In brief, the spatial orientation of an arbitrary corner (with one central a three lateral vertices) is analytically determined (see Figure 4.a, where D is the central vertex and A B C are the lateral ones). From this analytical relation, coordinates of all lateral vertices are obtained. Next, the process is repeated through a propagation tree that converts already determined lateral vertices into new central vertices of neighbor corners, ensuring in this way that a topologically consistent 3D model is constructed. We use a particular version of Kruskal algorithm to obtain a spanning tree formed by all edges connecting the successive central vertices.

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 all vertices of other valences can be determined as laterals. Moreover, only edges contained in the spanning tree must be parallel to main directions. Hence, the shape domain supported by CIGRO extends to *quasi-normalon* objects, which are objects that can be reduced to normalons by deleting all edges running non-parallel to the three main directions, without losing any vertices in the model. For practical purposes, we do distinguish two classes of quasi-normalon objects:

- Class 1: when a connected graph is obtained after simplification.
- Class 2: when a not connected graph is obtained after simplification.

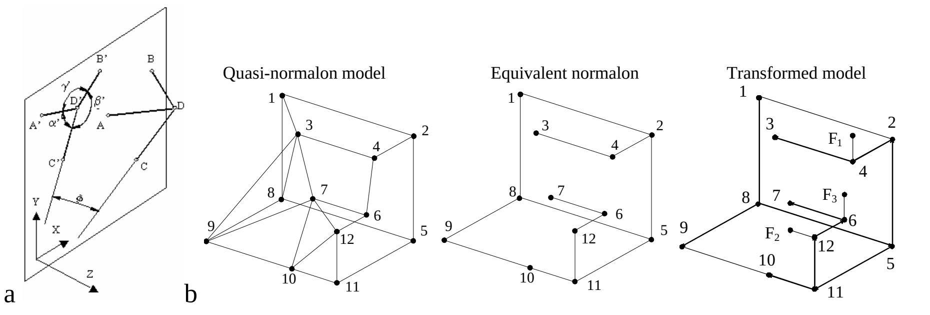

Fig. 4. a) Inflation method b) Transformation of a quasi-normalon model

3.3.1 Managing Class 1 Quasi-normalon Objects

Axonometric inflation can be applied to the class 1 quasi-normalon models as described above, provided than the spanning tree is determined in the equivalent normalon, obtained after the temporary elimination of all line-segments that are non-parallel to any of the three main directions. The equivalent normalon is valid if no vertex is deleted and the graph remains connected.

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 4.b, vertex 3 is only connected to vertex 4, which cannot be central because its valence is two. 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 4.b, 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.

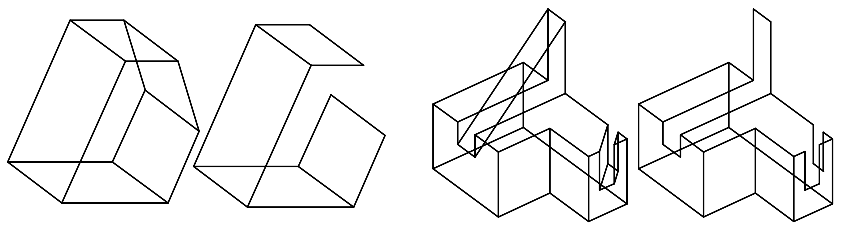

Fig. 5. Examples of Class 1 quasi-normalon-type 1 objetcs and their equivalent models

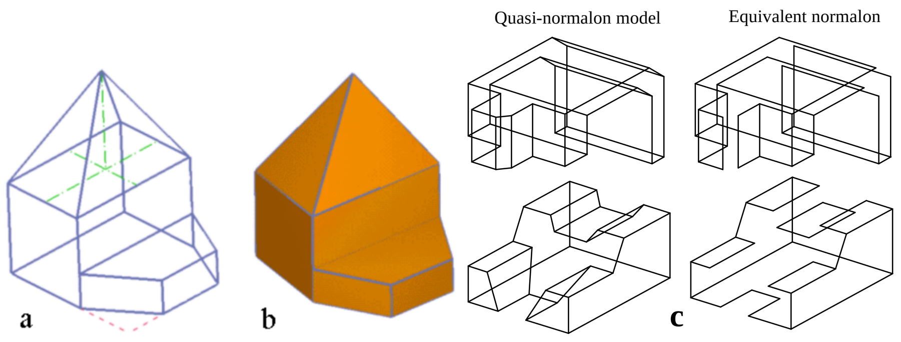

Fig. 6. Automatic auxiliary lines (dashed cyan lines in a) and class 2 quasi-normalon models

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 (fig. 6.a) to allow the reconstruction of the model. This algorithm essentially consists of joining the graphs that have been isolated as a consequence of their conversion 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 xyz-axis (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. This coplanarity condition obviously requires faces to be detected in the input 2D drawing, which is itself an encouraging task. Currently, to detect faces automatically, we make use of an adapted version of Courter and Brewer's [23] algorithm.

To determine the automatic auxiliary lines it is first selected a connecting edge that links the independent graphs resulting from the transformation into equivalent normalon (we call it hypotenuse). The selection is made at random from 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 one of the xyz-axis (we call them the cathetus), thus giving rise to a new vertex on the initial graph. The choice of two directions from the three offered by the xyz-axis is conditioned by the coplanarity criterion: i.e. 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:

- If edges that run parallel to two dissimilar directions of the trihedron converge to the vertices of the connecting edge, we simply extend the converging edges until they intercept each other (see Figure 7.a)
- If edges running parallel to a same direction in the xyz-axis converge to the vertices of the connecting edge, the direction running parallel to one axis xyz that does not converge to the connecting edge is used to determine the new vertex. Unless there are two possible valid configurations, the random choice of one of them does not affect the generality of the method (see Figure 7.b).

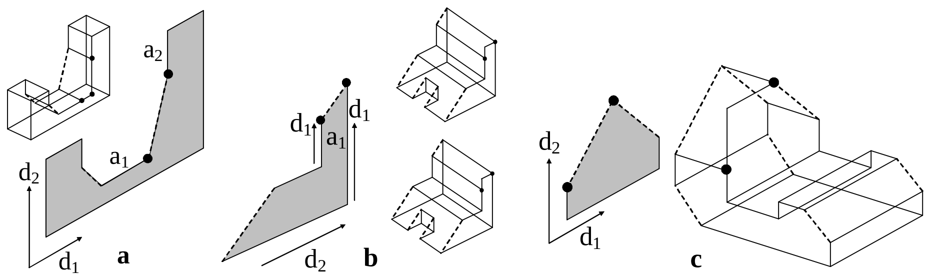

Fig. 7. Examples for the substitution of connecting edges

- If, at any of the vertices of the connecting edge, none of the edges running parallel to the directions of the dihedron converge, 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 7.c).

The last step in the automatic auxiliary lines generation describe above is 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 some of the xyz-axis. After this check the following steps are taken: If the new model is connected, it is reconstructed, elsewhere the process 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 normalons and 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.

Acknowledgments

This work was supported in part by the Portuguese Science Foundation grant 34672/99, the European Commission project IST-2000-28169, the Spanish Generalidad Valenciana grant CTIDIB/2002/51 and Fundació Caixa Castelló- Bancaixa under the Universitat Jaume I program for Research Promotion (Project P1-1B2002-08).

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**, (2000) 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., Matsuoka, S., Kawachiya, S., Tanaka, H.: Interactive Beautification: A Technique for Rapid Geometric Design, UIST'97, (1997) 105-114
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