“3D reconstruction problem”: an automated procedure

MONICA CARFAGNI, ROCCO FURFERI, LAPO GOVERNI, MATTEO PALAI, YARY VOLPE
Università degli Studi di Firenze
Department of Mechanics and Industrial Technologies
Via di Santa Marta, 3 - Firenze
ITALY
matteo.palai@unifi.it - http://www.dmti.unifi.it

Abstract: 3D CAD techniques represent today a crucial tool in almost all the design fields. Nevertheless, due to a number of well known reasons, multi orthographic view drawings are still widely used; accordingly, the conversion of 2D drawings into 3D CAD models (known in the scientific literature as the “reconstruction problem”) is still a key issue. During the last decades a number of works, dealing with the reconstruction problem, have been proposed. On the basis of these works, the authors have developed and implemented an automatic procedure that allows the reconstruction of 3D polyhedral models. The reconstruction procedure involves a number of software routines; by means of them, an initial 2D DXF file is processed and a set of 3D solutions, consistent with the initial drawing, is extracted. The obtained 3D models are subsequently output according to the most common 3D exchange formats (e.g. IGES, STEP, Parasolid, etc.). The proposed procedure and its implementation have been developed in order to achieve two main goals: to introduce researchers into the “reconstruction problem” and to create a common basis in order to methodologically compare different procedures dealing with the “reconstruction problem” itself.

Key–Words: 3D reconstruction, Engineering drawings, Computational geometry, CAD, Orthographic projections, Pseudo-wireframe

1 Introduction

Three-dimensional CAD modelers, throughout the last decades, has undergone a dramatic development, becoming, as a matter of fact, the most widespread tools in engineering design. Nevertheless, for a number of well known reasons, bi-dimensional CAD modelers are still widely employed, so multi orthographic view engineering drawings play an essential role in many engineering applications. Since many common engineering activities (e.g. FEM analysis, CFD analysis, CNC machining, rapid prototyping, etc.) require 3D CAD models, an automatic tool for 3D reconstruction can be, still today, extremely useful. The obtainment of 3D models starting from multi orthographic view engineering drawings is known in the scientific literature as the “reconstruction problem”. Since the early 1970s a number of scientific studies have been carried out confronting the reconstruction problem: these can be classified in two main families, the first one is based on B-rep (Boundary representation) methods, while the second family makes use of CSG (Constructive Solid Geometry) approaches. A useful review of the state of the art, regarding both the approaches, is provided in some recent publications [1, 3]. The present work is based on B-rep approach and focuses on the implementation of 3D CAD polyhedral model reconstruction starting from its orthographic projections. Even though the scientific literature proposed solutions to the reconstruction problem, to the best knowledge of the authors, a “researcher-friendly” reconstruction methodological and practical tool has not been yet provided. Accordingly, the aim of this work is to provide researchers and practitioners with a tool, enabling a straightforward implementation of the 3D polyhedral model reconstruction. The proposed method, though inspired by a number of studies [6, 7, 8, 9, 10, 11, 12, 5], makes use of an original approach. Since the method developed by the authors is oriented towards a systematic (step by step) description, computational optimization is intentionally neglected in order to make the procedure as much comprehensible and intuitive as possible. The paper is organized as follows: in section 2 the authors describe the reconstruction method; section 3 provides a few details about software implementation; in section 4 two case studies are presented; finally, in section 5, conclusions and future works are discussed.

2 Method

As depicted in Fig.1, the proposed procedure allows to identify and rebuild the entire set of 3D CAD models (if more than one exist) coherent with the bi-

dimensional DXF input data. As widely known Drawing Interchange Format (DXF) is a CAD data file format, developed by Autodesk® for enabling data interoperability between AutoCAD® and other software packages. DXF files contain a series of vector data describing all the geometric entities making up a drawing. The first step of the procedure consists of extracting, from a DXF file, the end-point coordinates of each geometric feature (i.e. lines, since the procedure deals with polyhedral models) and storing them into a database. The goal of this task is to obtain an ordered matrix (size $2n \times 3$) of n edges, each one defined by two triplets (see later description) of coordinates . Generally, inside a DXF file all the features are randomly stored during the drawing process. In order to assign each feature to its view (front, side and top views) the DXF file needs to be interpreted properly.

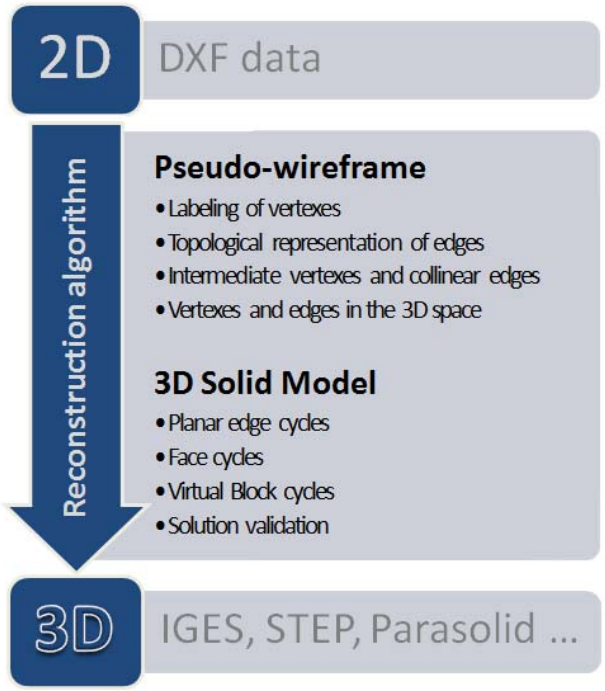


Figure 1: Method flow diagram

In particular, all the views are translated so that the DXF coordinate system (UCS) is positioned as shown in (Fig.2).

As a consequence all the features can be assigned to a view ($[x, y, 0]$, $[x, 0, z]$ or $[0, y, z]$) by inspecting the sign of their endpoints. This task is performed by means of a labeling procedure: features belonging to the top view have both negative endpoint coordinates ($\boldsymbol{x}$ and $\boldsymbol{y}$); features on front view are characterized by negative $\boldsymbol{x}$ and positive $\boldsymbol{y}$; finally, features on side view have all positive coordinates ($\boldsymbol{x}$ and $\boldsymbol{y}$). The result of feature labeling allows splitting the database into three sets of data (each one containing only the endpoints $\boldsymbol{e}_i^j$ of a single view $\boldsymbol{\Pi}^j$) [4].

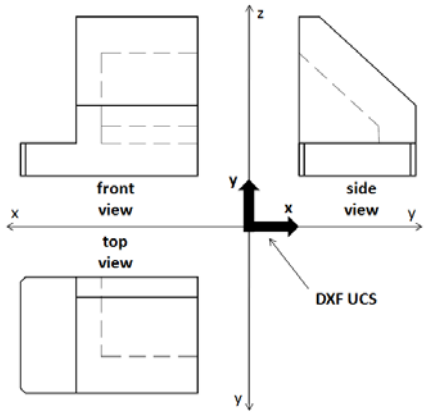


Figure 2: 2D view position

$$\begin{aligned} \boldsymbol{\Pi}^1 &= [\boldsymbol{e}_1^1, \boldsymbol{e}_2^1. \ldots, \boldsymbol{e}_a^1]_a^T \\ \boldsymbol{\Pi}^2 &= [\boldsymbol{e}_1^2, \boldsymbol{e}_2^2. \ldots, \boldsymbol{e}_b^2]_b^T \\ \boldsymbol{\Pi}^3 &= [\boldsymbol{e}_1^3, \boldsymbol{e}_2^3. \ldots, \boldsymbol{e}_c^3]_c^T \end{aligned} \tag{1}$$

Once the database is created, the procedure for re-constructing 3D models is started. As illustrated in Fig.1, such a procedure consists of the following main tasks:

- pseudo-wireframe reconstruction;
- 3D solid model reconstruction.

2.1 Pseudo-wireframe reconstruction

The pre-processed data coming from the interpretation of the DXF file may be manipulated in order to generate a mathematical representation of the 3D vertexes and edges composing the pseudo-wireframe model. This is accomplished according to the following phases [4]:

- labeling of vertexes;
- topological representation of edges;
- detection of intermediate vertexes and collinear edges;
- reconstruction of vertexes and edges in the 3D space.

2.1.1 Labeling of vertexes

Each vertex ($\boldsymbol{v}$) is labeled with a progressive number (n) so that a topological data structure is created.

$$\boldsymbol{v}_n = [x_n, y_n, z_n] \Rightarrow label(\boldsymbol{v}_n) = n \tag{2}$$

2.1.2 Topological representation of edges

Each edge ($\boldsymbol{e}$) is defined by means of the set of labels of its vertexes [4].

$$\boldsymbol{e}_j = \begin{bmatrix} x_h, y_h, z_h \\ x_k, y_k, z_k \end{bmatrix}_{2\times 3} \Rightarrow label(\boldsymbol{e}_j) = [h,k]_{1\times 2} \quad (3)$$

2.1.3 Detection of intermediate vertexes and collinear edges

Though an object is represented by a univocal set of projections, these can be drawn by using different combination of geometric entities, i.e. the same projections can be represented by different DXF files. Therefore, in order to obtain a univocally defined vectorial representation covering all the possible configurations, an approach for processing different DXF files of the same drawn object is provided. First, for each edge, an iterative procedure checks the existence of intermediate vertexes (due to possible intersection of edges with no associated vertex). If no intermediate vertex is found, the procedure stops; otherwise, each intermediate vertex causes the creation of two new edges. This procedure, called "segmentation", is applied to each set of edges belonging to a projection, thus the original edge sets are updated. After the segmentation task, an edge collinearity check is performed (this phase is fundamental when two non contiguous vertexes are linked by two or more edges all collinear one with each other). The collinearity can be operatively detected by means of the logical product of concatenation (i.e. two edges sharing the same vertex) and parallelism. When an edge collinearity is detected, a new edge, defined by the union of the two collinear ones, is added to the original set. The final result of this procedure is to define three matrixes (named "junction matrixes") that, similarly to "adjacency matrixes" defined by graph theory [2], relate vertexes and edges. In Fig.3 a graphical representation of the three matrixes, referring to the example of Fig.2, is depicted. Note that in Fig.2 segments are to be intended as arcs belonging to the graph defined by junction matrices (rather than as geometric entities), while vertexes represent graph nodes (though they also represent the real geometric vertexes).

2.1.4 Reconstruction of vertexes and edges in the 3D space

Once obtained the junction matrix for each projection view, it is possible to build a "pseudo vertex skeleton" according to the widely known approach proposed by Wesley and Markowsky [10]. The set of 3D vertexes, obtained by means of this approach, is a super-set of the real vertexes of all the possible 3D objects which can be represented by the starting DXF file. Fig.3 shows the pseudo wireframe obtained from the dataset depicted in Fig.3.

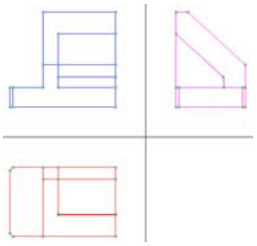


Figure 3: Graphical representation of 2D database

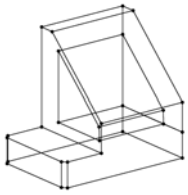


Figure 4: Pseudo Wireframe

2.2 Solid model reconstruction

Once the pseudo-wireframe model has been reconstructed, the algorithm proceeds to 3D model reconstruction according to the following further tasks:

- detection of planar edge cycles;
- detection of face cycles;
- creation of virtual block cycles;
- solution validation.

2.2.1 Detection of planar edge cycles

In this task face detection is performed according to a 3-phase procedure:

- the detection of the planes defined by all the couples of pseudo-wireframe edges sharing an endpoint;
- the identification of all the edge cycles lying on each plane;
- the definition of the faces by means of the analysis of each edge cycle.

The first phase basically checks two different conditions: an edge must lie on at least two planes; a plane must include at least three edges. From the algorithm point of view, these checks are performed by means of normalized cross products between couples of edges sharing an endpoint (which results in the vector normal to the plane defined by the two edges); then,

ISBN: 978-960-474-270-7

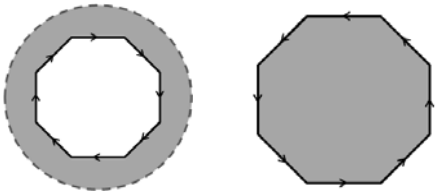
each cross product result is organized in a row vector $\boldsymbol{p}_i$ defined in Eq.4. Vector $\boldsymbol{p}_i$ contains both the cross product result (in the form of its components along x, y and z axes) and the label n of the shared endpoint. All vectors are finally arranged in the form of a matrix (called "matrix of planes") of size $k \times 4$, where k is the number of planes.

$$\boldsymbol{p}_i = [x_i, y_i, z_i, n] \tag{4}$$

Once the "matrix of planes" is compiled, the second sub-procedure (second phase) computes the edge cycles by inspecting each identified plane according to [10]. At the end of this operation, a first result is represented by a set of edge cycles, each one linked to its belonging plane. This result has to be further refined by assigning to each cycle the right path direction: clockwise (CW) or counter clockwise (CCW). In order to achieve this goal, the sum value (S) of all the inner angles of each cycle is computed so that:

- if $S = -360°$ the cycle is CW directed (Fig.5a);
- if $S = 360°$ the cycle is CCW directed (Fig.5b).

Finally, in the third phase, the detection of the face sets is performed. Assuming that, by ideally walking on the cycle boundary, the faces always lie on the left side, it is clear that only the CCW cycles can originate faces, since CW one do not delimit a finite area region.



(a) a CW cycle (b) a CCW cycle

Figure 5: Planar edge cycles

Each face ($\boldsymbol{f}$), commonly known as "virtual face", is described by a row vector storing the label of the linked plane and the list of all the edges representing the face boundary. The final result of this task consists of an $nf \times (n_{max} + 1)$ matrix where: nf is the number of faces, while n_{max} is the number of edges surrounding a face (among the identified faces).

2.2.2 Detection of face cycles

It is known that the set of faces resulting from the previous task generally represents a superset of the face set describing the actual 3D solid [10]. In order to complete the reconstruction process, it is, thereby, necessary to identify the set of elementary 3D blocks (commonly known as "virtual blocks") whose combinations provide the final 3D geometries. The block detection task is performed analogously to the face detection one; the main difference between the two is represented by the space dimensionality: blocks are managed in 3D space while faces in 2D one. From the algorithm point of view, this task requires a "warm up" phase for the compilation of two auxiliary matrixes: F1 (size $\#\boldsymbol{e} \times max\{\#fe_i\}$) and F2 (size $\#\boldsymbol{f} \times max\{\#ef_i\}$).

In the first one, each row is devoted to an edge $\boldsymbol{e}_i$ and contains the labels of all the faces $\boldsymbol{fe}_i$ having $\boldsymbol{e}_i$ among their boundary edges. The second matrix allows a fast detection of the edges $\boldsymbol{ef}_i$ composing the boundary of each face; each row is devoted to a face $\boldsymbol{f}_i$ and contains the labels of all the edges $\boldsymbol{ef}_i$ bounding it. Once these two auxiliary matrixes have been compiled, the procedure carries on detecting the loops of faces (blocks). The face cycle identification is a close analog of the one devised for the planar edge cycles, described in the previous section. The procedure stops when the two sides of each face has been used in cycle detection process. The implemented procedure related to the cycles of faces is inspired by a work of Yan, Chen and Thang [12]. Referring to Fig.6, given two faces, $\boldsymbol{F1}$ and $\boldsymbol{F2}$, and assuming as positive the normal vector pointing out from the sheet:

- in cases A and D the vectors normal to both provenience and destination faces have the same sign; positive in case A and negative in case D.
- in cases B and C the vectors normal to provenience and destination faces have opposite sign; in B the positive normal vector belongs to the destination face while, in C to the provenience one.

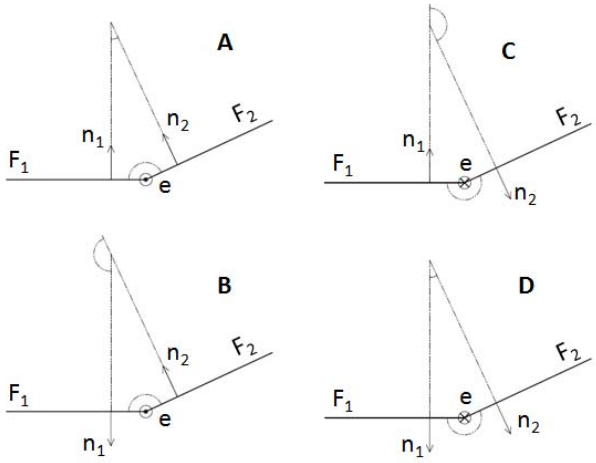


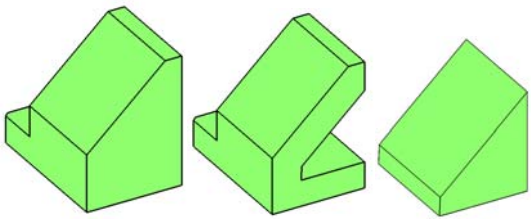
Figure 6: Face cycle interpretation

By examining Fig.6 no particular difficulty seems to arise in order to perform the face cycles retrieval; actually, the correct identification of the normal vectors referring to the edge $\boldsymbol{e}$ it is not trivial from a mathematical point of view. Accordingly, the qualitative

approach presented in [12] has been formalized as follows, in order to overcome possible ambiguities and to obtain a strict, unequivocal management of the loop detection. By means of a series of cross product comparisons involving the two face normals and the edge e , the authors provide a method that allows to correctly identify the right configuration among the four ones depicted in Fig.6.

2.2.3 Creation of virtual block cycles

The last task of the reconstruction procedure deals with block management in order to provide complete set of block combinations leading to 3D objects whose projections are coherent with the starting ones. This task starts by computing the volume of each single block so that the one with the maximum volume is discarded. In fact, it is clear that the biggest block is the external shell (Fig.7a). The other n blocks are combined in order to obtain $\sum_{k=1}^{n} C_{n,k}$ combinations, where $C_{n,k}$ is the binomial coefficient. Referring to Fig.7, the possible block configurations are: block B, block C, blocks B and C.



(a) Block A (b) Block B (c) Block C

Figure 7: Virtual blocks

2.2.4 Solution validation

In order to evaluate the solution correctness all the obtained configurations are verified according to a two step procedure. First, each configuration is compared with the original set of 2D views (first validation): if the block configuration originates a set of projections matching the original ones, it is candidate to be a correct solution. This first validation, however, is not sufficient to establish the solution correctness since the blocks are still disjoint. In fact, the joining process removes the edges on the boundary of the shared faces. As a consequence, it is necessary to perform a second validation on the set of candidate block configurations. The blocks, making up such configurations are, thereby, merged (boolean union) and a further comparison with the original views is performed (second validation). Performing the merge phase only on the candidate configurations, leads to a considerable reduction in computation time. The block configurations satisfying the two validation procedures represent the set of correct 3D solutions (Fig.8).

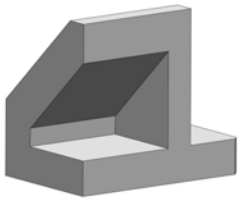


Figure 8: 3D CAD solution

3 Software implementation

The entire set of algorithms, performing all the procedures described in section 2, is implemented by means of two software packages, Matlab® and Rhinoceros®. The first one, which is widely spread in the scientific community, is mainly used as a mathematical kernel, while Rhinoceros® by means of custom scripts, allows to easily translate Matlab® output results into 3D models and operate them according to the described procedure. Algorithms described in sections from 2.1 to 2.2.2, have been designed and implemented using Matlab®; for the ones described in sections 2.2.3 and 2.2.4 both Matlab® and Rhinoceros® have been concurrently used.

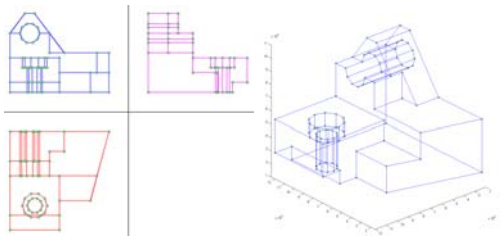
4 Case studies

The testing of the proposed methodology has been carried out on a series of case studies. In Figs.9 and 10 the key reconstruction phases for two case studies are illustrated by showing:

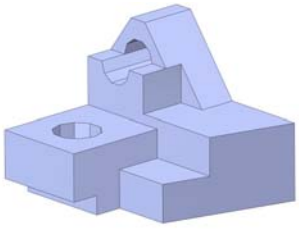
- the 2D database achieved from the manipulation of the 2D DXF file (MatLab®);
- the 3D pseudowireframe model (MatLab®);
- the final 3D solid model (Rhinoceros®).

5 Conclusions

In this work an orderly, unambiguous and automatic procedure to cope with the reconstruction problem from the implementation point of view is provided. The proposed method, developed with reference to polyhedral objects, allows the reconstruction of the entire set of 3D CAD models represented by a given set of 2D orthographic views (in DXF format). The procedure has been designed like an open source tool for researchers who want to deal with the “reconstruction problem”. In order to assess its effectiveness, it has

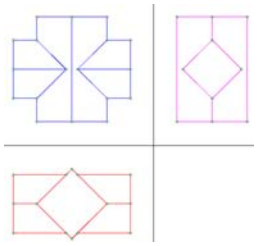


(a) 2D database (b) 3D pseudo wireframe

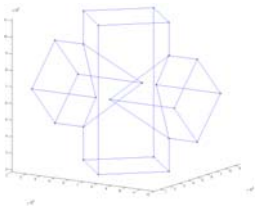


(c) 3D solid model

Figure 9: Snapshots of study case A



(a) 2D database



(b) 3D pseudo wireframe

(c) 3D solid model

Figure 10: Snapshots of study case B

been implemented using Matlab® and Rhinoceros® software packages and tested on a number of case studies. Future work will be addressed to a more general methodology dealing with the reconstruction of 3D CAD models starting from 2D views of objects defined by edges with arbitrary geometry.

References:

[1] COMPANY, P., PIQUER, A., AND CONTERO M, MNAYA, F. A survey on geometrical reconstruction as a core technology to sketch-based modeling. *Computers & Graphics 29*, 6 (2005), 892–904.

[2] DIESTEL, R. Graph theory. *Springer*, (2006).

[3] FAHIEM, M. A., HAQ, S. A., AND SALEEMI, F. A Review of 3D Reconstruction Techniques from 2D Orthographic Line Drawings. *Geometric Modeling and Imaging (GMAI '07)* (July 2007), 60–66.

[4] FURFERI, R., GOVERNI, L., PALAI, M., AND VOLPE, Y. From 2D Orthographic views to 3D Pseudo-Wireframe: An Automatic Procedure. *International Journal of Computer Applications IJCA 5*, 6 (2010), 12–17.

[5] GONG, J., ZHANG, G., ZHANG, H., AND SUN, J. Reconstruction of 3D curvilinear wire-frame from three orthographic views. *Computers & Graphics 30*, 2 (2006), 213–224.

[6] IDESAWA, M. A System to Generate a Solid Figure from Three View. *Bulletin of JSME 16*, 92 (1973), 216–225.

[7] INOUE, K., SHIMADA, K., AND CHILAKA, K. Solid Model Reconstruction of Wireframe CAD Models Based on Topological Embeddings of Planar Graphs. *Journal of Mechanical Design 125*, 3 (Sept. 2003), 434–442.

[8] LIU, S. Reconstruction of curved solids from engineering drawings. *Computer-Aided Design 33*, 14 (2001), 1059–1072.

[9] MEERAN, S., AND PRATT, M. Automated feature recognition from 2D drawings. *Computer-Aided Design 25*, 1 (1993), 7–17.

[10] WESLEY, M., AND MARKOWSKY, G. Fleshing out wire frames. *IBM Journal of Research and Development 24*, 5 (1980), 582–597.

[11] WESLEY, M., AND MARKOWSKY, G. Fleshing out projections. *IBM Journal of Research and Development 25*, 6 (1981), 934–954.

[12] YAN, Q., CHEN, C., AND TANG, Z. Efficient algorithm for the reconstruction of 3D objects from orthographic projections. *Computer-Aided Design 26*, 9 (1994), 699–717.

ISBN: 978-960-474-270-7