From 2D Orthographic views to 3D Pseudo-Wireframe: an Automatic Procedure

From 2D Orthographic views to 3D Pseudo-Wireframe: An Automatic Procedure

Rocco Furferi* rocco.furferi@unifi.it
Lapo Governi* lapo.governi@unifi.it
Matteo Palai* matteo.palai@unifi.it
Yary Volpe* yary.volpe@unifi.it

* Università degli Studi di Firenze
DMTI – Department of Mechanics and Industrial Technologies

ABSTRACT
Three-dimensional CAD models are usually used by designers because of their multiple uses (visualization, simulation, machining). However, nowadays, multi orthographic view engineering drawings are still widely used. Accordingly, a conversion tool for obtaining 3D CAD models from 2D drawings (known as the ”reconstruction problem”) is a very useful approach in a broad range of applications. The significant interest for the reconstruction problem is witnessed by the large number of works presented in the last three decades. The main object of the present work, by integrating different approaches suggested by a number of authors and rearranging them into an orderly, unambiguous and automatic procedure, is to provide a tool to help researchers and practitioners who want to deal with the reconstruction problem.

In detail the authors propose a systematic tool that allows the reconstruction of a 3D pseudo-wireframe starting from a 2D vectorial input. Such a tool is discussed in detail and has been implemented into MatLab® environment in order to validate and test the procedures. Extensive testing, carried out on a number of case studies, has demonstrated the effectiveness of the presented approach.

Keywords
3D reconstruction, pseudo-wireframe, engineering drawings, computational geometry.

1. INTRODUCTION
3D CAD modelers are recognized as one of the mostly used tool in engineering design. Both solid and surface CAD models have become crucial for a large number of CAE techniques (e.g. visualization, simulation, CNC machining, …). Anyway, multi orthographic view engineering drawings have been widely used up to latest decade and still are, so they play an essential role in traditional engineering. Actually, many products are still designed by means of orthographic views. Moreover, many engineering tasks involve modification of existing design, thus an automatic tool for reconstructing a 3D CAD model, starting from multi orthographic view engineering drawings, would prove to be particularly useful in many applications. In addition to its research significance, this kind of tool would ease a number of practical issues, mostly in the field of automatic conversion of digitized engineering drawings into 3D CAD models.

In the past three decades many scientific studies have been carried out confronting this topic, that has come to be known as geometrical reconstruction or simply reconstruction problem. The reconstruction problem has been studied since the first 1970s and a large number of works can be found in scientific literature. These can be divided in two different families:

1. wireframe-oriented approaches, that are also known as B-rep (Boundary representation) methods;
2. volume-oriented approaches, also called CSG (Constructive Solid Geometry) methods.

A useful review of relevant published works, regarding both B-rep and CSG approaches, is provided by two recent publications [1, 2]. Recently the preferred approach for performing 3D reconstruction has been the B-rep based one. This is mainly due to the fact that the CSG approach is less suitable to support complex shapes and usually requires heavier user interaction compared to the B-rep one.

It is commonly accepted that the reconstruction problem can be split into two main phases: the first is the reconstruction of the pseudo-wireframe model (set of all possible wireframe models that can be originated by an assigned set of orthographic views [3]); the second is the reconstruction of the 3D solid (or surface) model(s) from the obtained pseudo-wireframe model and coherent with the assigned orthographic views [4].

This work focuses on the first phase, i.e. the reconstruction of the pseudo-wireframe model. In spite of the huge literature on the reconstruction problem, almost all methods, proposed by several different authors, are mainly described by a conceptual point of view, so that to derive an orderly procedure covering the necessary steps from 2D data to pseudo-wireframe model always requires a great effort and a considerable amount of work. The most challenging tasks that are to be faced, when trying to derive such a procedure are related to the presence of ambiguities in the methodology description and to the lack in enumerating all possible cases that can be found when having to deal with real-life drawings. For instance, Yan et al. [5] describe a conceptually flawless method to detect 3D edges on the basis of a table of possible configurations; nevertheless such a method has to be improved by adding more new configurations in order to derive a comprehensive and unambiguous operative procedure.

The aim of the present work is to provide researchers and practitioners with an orderly and automatic procedure enabling a straightforward implementation of the pseudo-wireframe model reconstruction. The works proposed by the scientific literature, which confront the reconstruction problem for curvilinear objects usually present, quite "tricky" approaches, generally involving heavy user interaction [6, 7, 8, 9]. At the moment, though very

interesting and promising, such approaches are not well established in the scientific community. Since existing approaches are certainly better recognized in the case of polyhedric objects, the authors decided to confront the reconstruction problem for this kind of geometric entities. The proposed method, though inspired by a number of studies [3, 4, 5, 10], makes use of an original approach oriented to the implementation task mentioned above.

Since the method developed by the authors, based on the B-rep approach, is oriented toward a systematic (step by step) description, computational optimization is intentionally neglected in order to make the procedure as more comprehensible and intuitive as possible.

In the paper, after the description of the procedure (from 2D data input to pseudo-wireframe model reconstruction), obtained results are presented by means of some reconstruction examples. Finally, some hints to possible future work are provided.

2. METHOD

As described in the previous section, the main objective of the present work is to provide an automatic procedure for reconstructing a pseudo-wireframe model starting from a set of 2D projections. More in detail, the devised method uses a DXF file, as input, and provides, as output, a 3D pseudo-wireframe model. According to [11], a representation of a drawing can be generated in some neutral file formats such as DXF (Drawing Exchange Format) developed by Autodesk. This format, compiled in binary or ASCII formats, allows to share CAD drawings among several different CAD software packages.

Given three projections of an object in the DXF format, the detection of each entity (line, circle, etc.) composing it is straightforward.

Unfortunately, such a file format does not provide topological information, i.e. the entities in a DXF file are in no logical order so that no explicit information regarding their connectivity is accessible. In other words the information regarding the spatial position of each projection (and of all the edges lying on it) is not available in the DXF file. For such a reason, a method for separating the drawing into three views is needed before further processing can take place. If $\mathbf{\Pi}^i$ is defined as the i^{th} projection ($i=1,2,3$) in the orthographic view system, several approaches for separating the three views can be used (e.g. [11]).

By using one of these approaches, the result is the creation of two families of sets:

1. $\mathbf{V}^i$, with $i=1,2,3$, represent the three sets of vertices, each one corresponding to a projection in the orthographic view system (i.e. $\mathbf{V}^i$ is the set of vertices of $\mathbf{\Pi}^i$).
2. $\mathbf{E}^i$, with $i=1,2,3$, represent the three sets of edges, each one corresponding to a projection in the orthographic view system (i.e. $\mathbf{E}^i$ is the set of edges of $\mathbf{\Pi}^i$).

On the basis of the six sets described above, $\mathbf{V}^i(j)=\mathbf{v}^i_j=[x^i_j, y^i_j, z^i_j]$ represents the j^{th} vertex in the i^{th} orthographic view.

By definition, each vertex lies on one of the coordinate planes. Consequently, one of the elements of $\mathbf{v}^i_j$ is always equal to zero (see Figure 1).

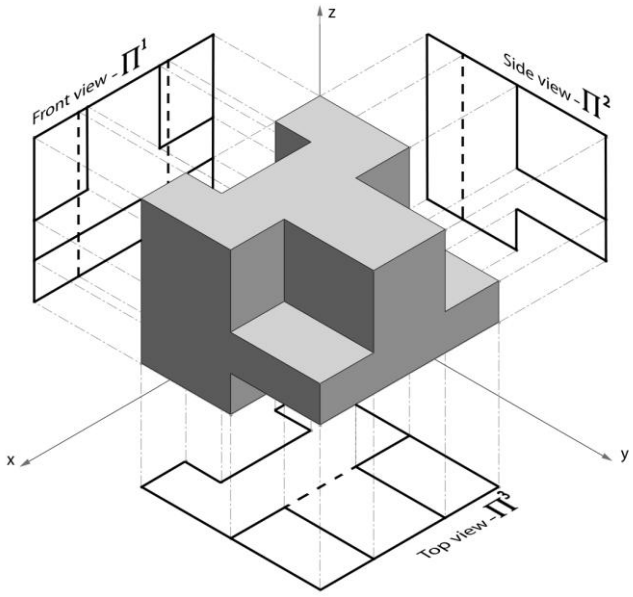


Figure 1. 3D projected object

The j^{th} edge in the i^{th} orthographic view is identified by its two vertices $\mathbf{v}^i_h$ and $\mathbf{v}^i_k$ as follows:

$$\mathbf{e}^i_j=\begin{bmatrix}\mathbf{v}^i_h\\ \mathbf{v}^i_k\end{bmatrix}_{2\times 3} \tag{1}$$

The three views can be expressed as follows:

$$\begin{aligned}\mathbf{\Pi}^1&=\left[\mathbf{e}^1_1,\mathbf{e}^1_2,...,\mathbf{e}^1_a\right]^T_a\\ \mathbf{\Pi}^2&=\left[\mathbf{e}^2_1,\mathbf{e}^2_2,...,\mathbf{e}^2_b\right]^T_b\\ \mathbf{\Pi}^3&=\left[\mathbf{e}^3_1,\mathbf{e}^3_2,...,\mathbf{e}^3_c\right]^T_c\end{aligned} \tag{2}$$

where a, b and c are, respectively, the number of edges in $\mathbf{\Pi}^1$, $\mathbf{\Pi}^2$ and $\mathbf{\Pi}^3$.

Finally the set of orthographic projections may be defined as:

$$\mathbf{OBJ}=\left[\mathbf{\Pi}^1,\mathbf{\Pi}^2,\mathbf{\Pi}^3\right]^T \tag{3}$$

where the size of matrix $\mathbf{OBJ}$ is $2a+2b+2c\ \times 3$.

In this pre-processing task, three more matrices are generated, one for each projection, containing the coordinates of the projection vertices. These structures can be represented as follows:

$$\begin{aligned}\mathbf{V}^1&=[\mathbf{v}^1_1,\mathbf{v}^1_2,...,\mathbf{v}^1_\beta]^T_{\beta\times 3}\\ \mathbf{V}^2&=[\mathbf{v}^2_1,\mathbf{v}^2_2,...,\mathbf{v}^2_\gamma]^T_{\gamma\times 3}\\ \mathbf{V}^3&=[\mathbf{v}^3_1,\mathbf{v}^3_2,...,\mathbf{v}^3_\varphi]^T_{\varphi\times 3}\end{aligned} \tag{4}$$

Where β, γ and φ are respectively the number of vertices in $\mathbf{\Pi}^1$, $\mathbf{\Pi}^2$ and $\mathbf{\Pi}^3$.

It is important to emphasize that the data structure described above may also be obtained starting from a different data source like, for instance, a vectorial database provided by the use of image processing techniques. In this case the only difference is in the edge extraction procedure. For this reason the approach described in the next task may be considered as stand-alone.

2.1 3D reconstruction of edges and vertices

Pre-processed data may be manipulated in order to generate a mathematical representation of the 3D pseudo-wireframe model. With the aim of obtaining such a model, a series of tasks has to be carried out:

1. labeling of vertices;
2. topological representation of edges;
3. intermediate vertices and collinear edges;
4. vertices and edges in the 3D space.

Though the first task allows to label 2D vertices, is necessary to verify that vertex sets of each projection are complete. Due to the drawing procedure, as a matter of fact, it is possible that some intersecting edges do not share any vertex. For such a reason, before labeling vertices, is necessary to check for the existence of possible additional 2D ones. In order to perform this task, in each projection, each couple of edges is checked for intersection and the sets are consequently updated.

2.1.1 Labeling of vertices

Once known the set of orthographic projections, it is possible to perform a reconstruction of vertices and edges in the 3D space. Anyway the data structure defined above may result in a large amount of data to be treated, especially for complex polyhedric objects. In order to reduce the information to be processed, a conversion of the geometric data into topological ones is mandatory in order to obtain an acceptable computational efficiency.

Accordingly, it is possible to create a topological data structure starting by labeling each vertex of each projection with a progressive number. As a result we obtain:

$$\mathbf{v}_n^i = \left[x_n^i, y_n^i, z_n^i \right] \Rightarrow \mathbf{v}_n^i = n \tag{5}$$

2.1.2 Labeling of edges

Each edge may be defined by means of the set of labels of its vertices. As a consequence each edge $\mathbf{e}_j^i$ can be rewritten as follows:

$$\mathbf{e}_j^i = \begin{bmatrix} x_h^i, y_h^i, z_h^i \\ x_k^i, y_k^i, z_k^i \end{bmatrix}_{2\times 3} \Rightarrow \mathbf{e}_j^i = \left[h^i, k^i \right]_{1\times 2} \tag{6}$$

Accordingly, only 3 parameters (h,k,i) are now used to properly identify each edge instead of 7 parameters previously defined ($x_h^i, y_h^i, z_h^i, x_k^i, y_k^i, z_k^i, i$).

Moreover the size of the matrix **OBJ** becomes $(a+b+c)\times 2$ instead of $2\cdot(a+b+c)\times 3$.

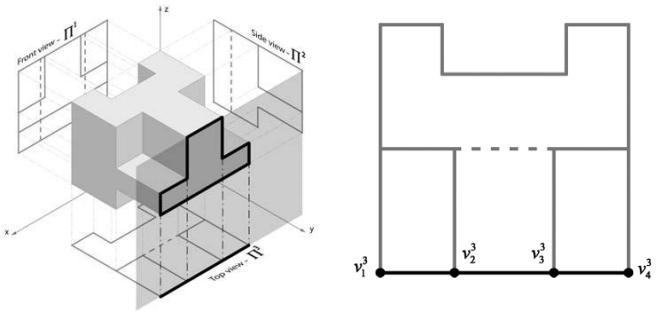
2.1.3 Intermediate vertices and collinear edges

Though an object is represented by a univocal set of projections, these can be drawn by using different combination of geometric entities: the segment highlighted in Figure 2b (as shown, for instance, in a printed copy of the drawing) in the DXF file can be made up of a number of straight vectors (from 1 to 8 as shown in Figure 3a).

Such combination of vectors, anyway, is uncorrelated with the one which would be generated by the projection of the object's 3D edges lying on the plane orthogonal to the view and whose trace contains the original segment (Figure 2a). In other words, the same projection can be represented by different DXF files.

Therefore, an approach for the processing of different DXF files of the same drawn object is provided, in order to obtain a univocally defined vectorial representation, comprising all the possible configurations.

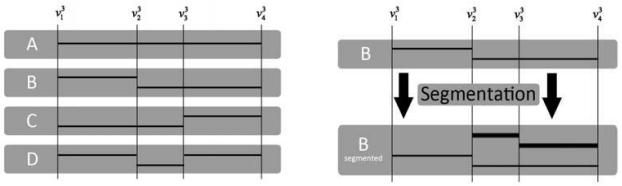
First, for each edge, an iterative procedure checks for the possible existence of intermediate vertices. If no intermediate vertex is found, the procedure stops. Otherwise the found intermediate vertex causes the creation of two new edges (unless one of them already exists). This task, called "segmentation", is performed for each set of edges belonging to a projection, thus adding new edges to the original set. Referring to Figure 2b, supposing that the portion of projection highlighted is represented by the configuration "B" of Figure 3a, the "segmentation" process produce the results shown in Figure 3b.



(a) Plane of projection (b) Projected view

Figure 2. Projection on an orthogonal view

Particularly, it is important to note that two new edges, highlighted in Figure 3b, have been generated in the projection. Note that any of the old edges is substituted or deleted after the "segmentation" procedure. For each projection $\mathbf{\Pi}^1$, $\mathbf{\Pi}^2$ and $\mathbf{\Pi}^3$ the dimension of the sets of edges is updated.



(a) Possible combinations of edges (b) "Segmentation" procedure

Figure 3. "Segmentation" procedure

According to the above procedure, the results of the segmentation task applied to the highlighted part of the projection in Figure 2b, are shown in Figure 3b. Thus, the two highlighted edges in have been added to the set of edge.

After the "segmentation" task, a check of collinearity of edges is performed. As shown in Figure 4, this phase is fundamental when two non contiguous vertices are linked by two or more edges all collinear one with each other.

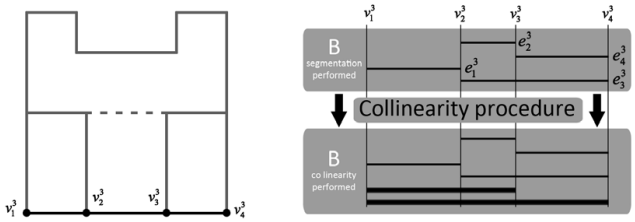


Figure 4. "Collinearity" procedure

If this check is neglected (or inaccurate) it could happen that two visually identical projections are described by two different datasets. When a collinearity of edges is detected, a new set of edges is added to the original one as described below. The collinearity can be detected as the logical product of concatenation (two edges sharing the same vertex) and parallelism. This logical process allows a reliable and straightforward approach for collinearity detection.

In order to describe the concatenation and parallelism relationship among the edges, the following two matrices has been defined:

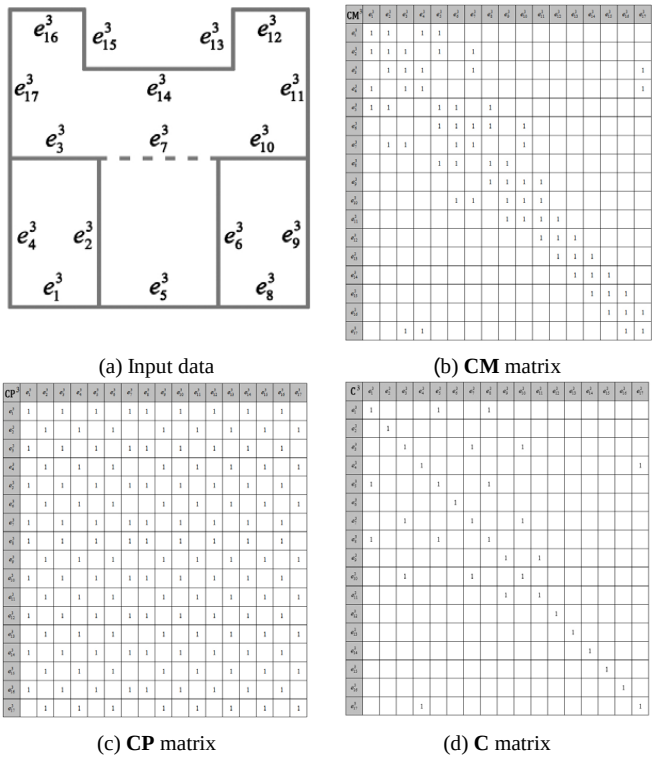
1. the concatenation matrix $\mathbf{CM}^i$ of the projection $\mathbf{\Pi}^i$;
2. the parallelism matrix $\mathbf{PM}^i$ of the projection $\mathbf{\Pi}^i$.

Since a single projection can be considered as a planar graph [12], matrices $\mathbf{CM}^i$ are defined similarly to the adjacency matrix in graph theory.

Accordingly, each matrix $\mathbf{CM}^i$ is a logical matrix whose elements $CM^{i_{s,t}}$ are equal to 1 when the s^{th} and the t^{th} edges of $\mathbf{\Pi}^i$ are concatenated and equal to 0 elsewhere. Matrices $\mathbf{PM}^i$ are also logical and their elements $PM^{i_{s,t}}$ are equal to 1 when parallelism subsists between the s^{th} and the t^{th} edges of $\mathbf{\Pi}^i$ and 0 elsewhere.

In order to further clarify the structure of these matrices an example is provided in Figure 5. More in detail the example is referred to the projection depicted in Figure 5a.

The collinearity between edges may be defined by the matrices $\mathbf{C}^i$ (Figure 5d) obtained as the element by element product between matrices $\mathbf{CM}^i$ and $\mathbf{PM}^i$, as illustrated in the example of Figure 5. Obviously, the collinearity between the s^{th} and the t^{th} edges in the $\mathbf{\Pi}^i$ projection subsists only when the element of matrix $\mathbf{C}^i$ (i.e. $C^{i_{s,t}}$) is equal to 1.



(a) Input data (b) **CM** matrix

(c) **CP** matrix (d) **C** matrix

Figure 5. "Collinearity" datasets

It is important to remark that the elements of the matrices $\mathbf{C}^i$ represent the collinearity relationship only for a couple of adjacent edges. In order to check such relationship for more than two adjacent edges, further processing is necessary. If matrices $\mathbf{C}^i$ are all zero matrices, no collinear edges exists in the three projections. Otherwise each non zero matrix $\mathbf{C}^i$ has to be checked column by column, starting from the top of the matrix, in order to detect all the collinear set of edges.

The final result of this procedure is to redefine the matrices $\mathbf{C}^i$ so that the position of nonzero elements in each column represent the collinear edges. All the permutations of the collinear edges that are not already stored in the matrices $\mathbf{\Pi}^i$ are then appended as new edges.

2.1.4 Vertices and edges in the 3D space

Once obtained a database of edges and vertices for each projection view, it is possible to build a pseudo vertex skeleton. According to the widely known approach proposed by Wesley and Markowsky [3] the set of vertices found by means of this approach is a super-set of the real vertices defining all the possible solutions of 3D models. This set, represented by $\mathbf{\Lambda}$ matrix, is known as the *pseudo vertex skeleton*.

An additional check verifies the possible presence of multiple identical rows $\boldsymbol{\lambda} \in \mathbf{\Lambda}$; these possible row groups are so simplified and only one of them is preserved inside the matrix $\mathbf{\Lambda}$. The result of this task is a new dataset structured as follows:

$$\mathbf{\Lambda} = \begin{bmatrix} x_1 & y_1 & z_1 & v_1^1 & v_1^2 & v_1^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_\varepsilon & y_\varepsilon & z_\varepsilon & v_\varepsilon^1 & v_\varepsilon^2 & v_\varepsilon^3 \end{bmatrix}_{\varepsilon \times 6} \quad (7)$$

where, for each row, the first three elements represent the spatial coordinates of the 3D vertex while the 4th, 5th and the 6th element identify, respectively, the 3D vertex projection on $\mathbf{\Pi}^1$, $\mathbf{\Pi}^2$ and $\mathbf{\Pi}^3$.

Once the matrix $\mathbf{\Lambda}$ has been compiled, four additional phases are required to accomplish the 3D edges construction task:

1. Construction of 3D edges that are not orthogonal to any projection;
2. Construction of 3D edges that are orthogonal to $\mathbf{\Pi}^1$;
3. Construction of 3D edges that are orthogonal to $\mathbf{\Pi}^2$;
4. Construction of 3D edges that are orthogonal to $\mathbf{\Pi}^3$.

In order to construct 3D edges that are not orthogonal to any projection, for each 3D vertex $\bar{\boldsymbol{\lambda}}$ a set of edges of $\mathbf{\Pi}^1$, sharing the $\bar{\lambda}_4$ 2D vertex, is stored in a temporary vector $\boldsymbol{\omega}$. For each 2D edge $\omega = \left[\bar{\lambda}_4, v_h^1\right] \in \boldsymbol{\omega}$, the procedure extracts all the rows $\boldsymbol{\lambda}$ so that $\lambda_4 = v_h^1$ and stores them in the $\mathbf{\Xi}$ temporary matrix, defined as follows:

$$\mathbf{\Xi} = \mathbf{T} \mid \mathbf{\Lambda} = \left[\begin{array}{c|cccccc} t_1 & x_1 & y_1 & z_1 & v_1^1 & v_1^2 & v_1^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ t_\eta & x_\eta & y_\eta & z_\eta & v_\eta^1 & v_\eta^2 & v_\eta^3 \end{array}\right]_{\eta \times 7} \quad (8)$$

where $\mathbf{T}$ allows to link each row between matrices $\mathbf{\Xi}$ and $\mathbf{\Lambda}$.

For all the combination between $\bar{\boldsymbol{\lambda}}$ and each row $\boldsymbol{\zeta} \in \mathbf{\Xi}$ the following conditions are evaluated:

1. $\exists \mathbf{e} = \left[\bar{\lambda}_5, \zeta_6\right] \in \mathbf{\Pi}^2$ (9)
2. $\exists \mathbf{e} = \left[\bar{\lambda}_6, \zeta_7\right] \in \mathbf{\Pi}^3$ (10)

If these two conditions are verified, the 3D edge, delimited by $\bar{\boldsymbol{\lambda}}$ and $\boldsymbol{\zeta}$, is appended to the list of 3D edges $\mathbf{\Theta}$.

At the end of the above procedure, matrix $\mathbf{\Theta}$ contains only 3D edges that are not orthogonal to any projection; therefore a further procedure to construct 3D edges that are orthogonal to each coordinate planes ($\mathbf{\Pi}^1$, $\mathbf{\Pi}^2$, $\mathbf{\Pi}^3$) has to be performed.

For instance, referring to $\mathbf{\Pi}^1$, such a procedure can be explained as follow. For each 3D vertex $\bar{\boldsymbol{\lambda}}$ the procedure extracts all the rows $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ so that $\lambda_4 = \bar{\lambda}_4$ and stores them in the $\mathbf{\Xi}$ temporary matrix, defined in eq. 8.

For all the combination between two rows $\zeta_h, \zeta_k \in \mathbf{\Xi}$ the following conditions are evaluated:

1. $\exists \mathbf{e} = \left[\zeta_{h,5}, \zeta_{k,5}\right] \in \mathbf{\Pi}^2$ (11)
2. $\exists \mathbf{e} = \left[\zeta_{h,6}, \zeta_{k,6}\right] \in \mathbf{\Pi}^3$ (12)

If these two conditions are verified, the 3D edge, delimited by ζ_h and ζ_k, is appended to the list of 3D edges $\mathbf{\Theta}$.

The result of these phases consists of two sets. One of them, $\mathbf{\Lambda}$, represents the list of 3D vertices, while the second one, $\mathbf{\Theta}$, represent the list of 3D edges.

Such sets mathematically represent, eventually, the pseudo-wireframe model of the object.

3. RESULTS

The mathematical procedure provided in section 2 has been implemented into the MatLab® environment. The resulting software has been thoroughly tested with a large number of case studies.

The test process has been carried out as follows:

Step 1: development of a 3D model for each object selected for the test (see Figure 6);

Figure 6. 3D model (Step 1)

Step 2: extraction of the orthographic projections from the 3D model, in the form of a DXF file, (see Figure 7);

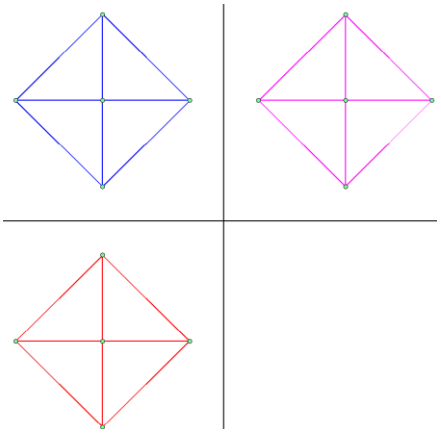


Figure 7. Orthographic projections (Step 2)

Step 3: processing of the DXF file by means of the presented reconstruction procedure thereby obtaining the test object's pseudo-wireframe (see Figure 8);

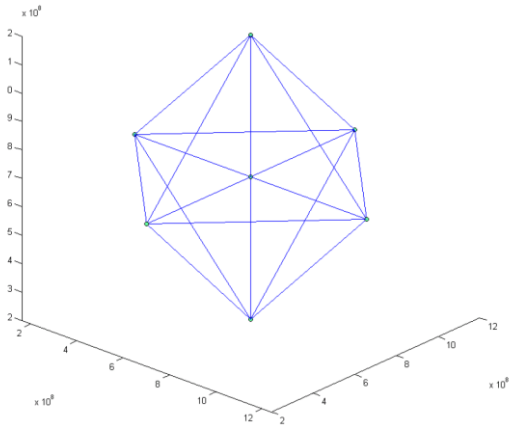


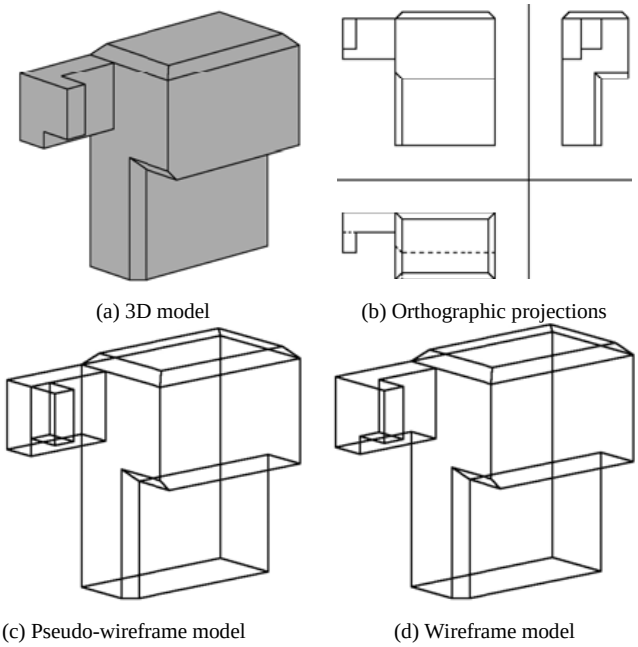
Figure 8. Pseudo-wireframe model (Step 3)

Step 4: comparison between the test object's actual wireframe model and its pseudo-wireframe one.

More in detail, in Step 4, it is necessary to prove that:

- the actual wireframe model is a subset of the obtained pseudo-wireframe one;
- the exceeding edges (which can be found in the second model, but not in the first) can be projected on segments actually existing in the orthographic projections obtained in Step 2.

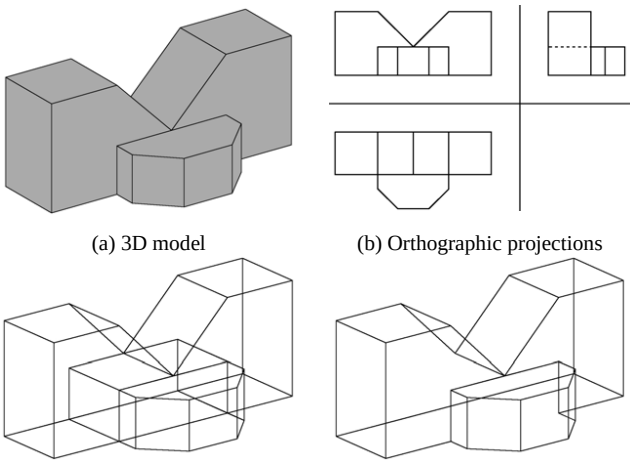
In Figure 9 and Figure 10, the results of pseudo-wireframe model reconstruction for a small selection of examples are presented.



(a) 3D model (b) Orthographic projections

(c) Pseudo-wireframe model (d) Wireframe model

Figure 9. Reconstruction example - case A



(a) 3D model (b) Orthographic projections

(c) Pseudo-wireframe model (d) Wireframe model

Figure 10. Reconstruction example- *case B*

4. CONCLUSIONS

In this work an orderly, unambiguous and automatic procedure to cope with the reconstruction problem from the implementation point of view is provided. Particularly, the proposed method allows the reconstruction of the pseudo-wireframe starting from 2D vectorial data.

The presented procedure has been designed like a support tool for researchers who want to deal with the reconstruction.

In order to assess its effectiveness, the procedure has been implemented using Matlab programming language and tested on a number of case studies. The presented results demonstrate the functionality and the reliability of the provided method. Future work will be addressed to the second phase of the reconstruction problem; accordingly it will deal with the reconstruction of 3D solid (or surface) model(s) starting from the pseudo-wireframe ones, obtained by means of the presented procedure.

REFERENCES

[1] Company, P., Piquer, A., and Contero M, Mnaya, F., 2005. "A survey on geometrical reconstruction as a core technology to sketch-based modeling". Computers & Graphics, 29(6), pp. 892–904.

[2] Fahiem, M. A., Haq, S. a., and Saleemi, F., 2007. "A Review of 3D Reconstruction Techniques from 2D Orthographic Line Drawings". Geometric Modeling and Imaging (GMAI '07), July, pp. 60–66.

[3] Wesley, M., and Markowsky, G., 1981. "Fleshing out projections". IBM Journal of Research and Development, 25(6), pp. 934–954.

[4] Wesley, M., and Markowsky, G., 1980. "Fleshing out wire frames". IBM Journal of Research and Development, 24(5), pp. 582–597.

[5] Yan, Q., Chen, C., and Tang, Z., 1994. "Efficient algorithm for the reconstruction of 3D objects from orthographic projections". *Computer-Aided Design*, **26**(9), pp. 699–717.

[6] Gong, J., Zhang, G., Zhang, H., and Sun, J., 2006. "Reconstruction of 3D curvilinear wire-frame from three

orthographic views". Computers & Graphics, 30(2), pp. 213–224.

[7] Zhang, A., Xue, Y., Sun, X., Hu, Y., Luo, Y.,Wang, Y., Zhong, S.,Wang, J., Tang, J., and Cai, G., 2004. Reconstruction of 3D Curvilinear Wireframe Model from 2D Orthographic Views.

[8] Inoue, K., Shimada, K., and Chilaka, K., 2003. "Solid Model Reconstruction of Wireframe CAD Models Based on Topological Embeddings of Planar Graphs". Journal of Mechanical Design, 125(3), September, pp. 434–442.

[9] Liu, S. et Al., 2001. "Reconstruction of curved solids from engineering drawings". Computer-Aided Design, 33(14), pp. 1059–1072.

[10] Gujar, U. G., and Nagendra, I., 1989. "Construction of 3D solid objects from orthographic views". Computers & Graphics, 13(4), pp. 505–521.

[11] Meeran, S., and Pratt, M., 1993. "Automated feature recognition from2D drawings". Computer-Aided Design, 25(1), pp. 7–17.

[12] Diestel, R., 2006. Graph Theory. Springer.