Automatic 3D CAD models reconstruction from 2D orthographic drawings

Chao Zhang, Romain Pinquié, Arnaud Polette, Gregorio Carasi, Henri de Charnace, Jean-Philippe Pernot

Title Page (with Author Details)

Automatic 3D CAD models reconstruction from 2D orthographic drawings

Chao Zhang, Romain Pinquié, Arnaud Polette, Gregorio Carasi, Henri De Charnace, Jean-Philippe Pernot

--Chao Zhang, chao.zhang@ensam.eu, Arts et Métiers Institute of Technology, LISPEN EA 7515, HESAM Université
--Romain Pinquié, romain.pinquie@grenoble-inp.fr, Grenoble INP, G-SCOP, Université Grenoble Alpes
--Arnaud POLETTE, Arnaud.POLETTE@ensam.eu, Arts et Métiers Institute of Technology, LISPEN EA 7515, HESAM Université
--Gregorio Carasi, gregorio@cognitive-design-systems.com, Cognitive Design Systems
--Henri De Charnace, henri@cognitive-design-systems.com, Cognitive Design Systems
--Jean-Philippe PERNOT, Jean-Philippe.Pernot@ensam.eu, Arts et Métiers Institute of Technology, LISPEN EA 7515, HESAM Université

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Automatic 3D CAD models reconstruction from 2D orthographic drawings

Anonymous submission #2198

ARTICLE INFO

Article history:
Received May 16, 2023

Keywords: Multiple-view 2D engineering drawings, 3D shape reconstruction, wireframe and surface modeling, pattern matching, clustering algorithm.

ABSTRACT

This paper introduces a two-stage approach that automatically generates 3D CAD models from 2D orthographic drawings. First, a pattern-matching algorithm is proposed to reconstruct a network of 3D edges by matching 2D edge features extracted from the multiple views of the 2D drawing. Second, starting from the resulting 3D wireframe and generated graph, a loop detection algorithm allows identifying possible loops of faces. Then, a clustering algorithm recognizes the faces from the set of detected loops. The reconstruction ends while trimming and stitching the faces to get a watertight ready-to-use 3D CAD model. This approach has been validated on a public dataset composed of several thousands of 3D shapes, and it achieved 99.59% of well-reconstructed models in F-score.

1. Introduction

Thanks to their 3D shape modeling capabilities, CAx tools and especially CAD modelers have become mainstream to support and structure the design, engineering and manufacturing activities all along the Product Development Process (PDP). The Digital Mock-Up (DMU) also plays an important role as it offers modelling and data storage possibilities making exchange and collaboration more efficient throughout the entire life-cycle. Most of the time, it serves as a reference model used for downstream applications. However, despite this craze for the development of full 3D applications, 2D drawings still play an essential role in engineering practices, and in many cases serve as the definitive design documentation that guides the manufacture or assembly of products. Naturally, shifting from 3D shapes to 2D representations is straightforward for designers who can easily generate 2D drawings from 3D models, and then use them as a contractual reference for the exchanges with their suppliers for instance. Conversely, automatically generating 3D CAD models from 2D drawings is a much more difficult task, and suppliers often have to rely on 2D information and thus cannot benefit from the full power of 3D representations. As a consequence, when changes to the shapes are expected, engineers first have to follow a time-consuming reconstruction process starting from scratch. Indeed, the information embedded in 2D drawings cannot be used directly in current CAD systems, and the automatic conversion of 2D drawings into 3D shapes remains a challenge, and a hindrance to improving performance and competitiveness.

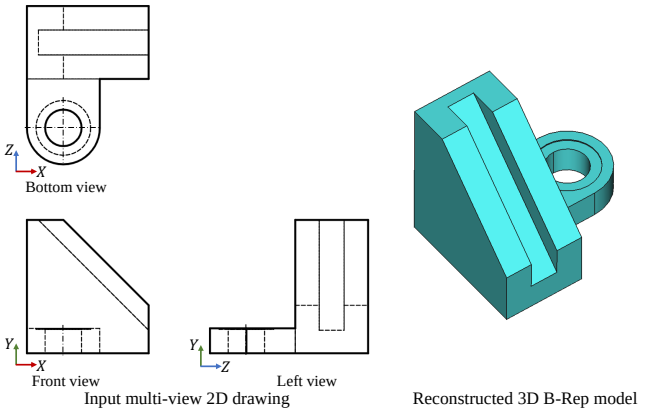

Fig. 1. Example of automatic 3D CAD model reconstruction (right) from a 2D orthographic drawing (left).

*Corresponding author: Tel.: +0-000-000-0000; fax: +0-000-000-0000;
e-mail: example@email.com (Corresponding Author Name)

Reconstructing 3D CAD models from 2D orthographic drawings is a topic that has received much attention over the last decades, as it is still a time-consuming and tedious manual process that requires a good knowledge of the underlying engineering rules. Indeed, despite the existence of commercial advanced modeling tools, it is still impossible to start with a 2D drawing depicting several views of a solid to automatically reconstruct its 3D CAD model. This is even more challenging when considering multi-view 2D composite drawings of several assembled parts, as in this case all details are often not fully available. Current methods often focus on geometric analysis and feature recognition, but they still face two main limitations:

- The algorithmic complexity is high. Existing methods have to search the space for all possible 3D edges and surfaces, and the search space grows rapidly with the number of edges and vertices in 2D drawings.
- The ambiguity of reconstructing 3D surfaces. To identify faces, an important step is to find cycles of edges that correspond to surfaces of the B-Rep model. Usually, faces are enclosed by two or more loops, but the existing methods are not good enough to compose the loops to the corresponding faces, and faces with several holes requiring multiple trimming operations can hardly be reconstructed.

Indeed, for a given input 2D drawing, there exists several reconstruction strategies, and automatically optimizing the way models are reconstructed is a challenging task. This is the objective tackled in this paper. Here, the 2D drawings have been vectorized in a preprocessing step. Readers are invited to refer to current state-of-the-art approaches [1, 2, 3] for more details on how to transform different types of 2D drawings to vectorized format. Therefore the preprocessing step is not part of this paper. One example resulting from the proposed approach is shown in figure 1. It takes a 2D orthographic engineering drawing as input, and the output is a watertight and ready-to-use 3D CAD model constructed following the proposed rule-based two-stage approach. More precisely, vertices and edges are first extracted from the 2D drawing. Then, a pattern-matching method is proposed to reconstruct the 3D edges and wireframe from the extracted vertices and edges. The algorithm focuses on co-edges and convert them to graphs. In the stage of face reconstruction, a new efficient method is proposed to detect loops in the graphs. Moreover, surface identification is made by a clustering algorithm that handles the issues of coplanar loops. The reconstruction ends while trimming and stitching the faces to get a watertight ready-to-use 3D CAD model.

In summary, the contribution is threefold: (i) a robust algorithm based on pattern matching to reconstruct various classes of 3D edges and capable of generating a 3D wireframe from a 2D drawing; (ii) an efficient loop detection algorithm able to identify the faces from the loops in a reduced search space; (iii) a clustering algorithm to identify the faces from a set of 3D loops in a wireframe, and address the issue of ambiguity in reconstructing 3D faces, especially when multiple loops define the inner boundaries of a face. Meanwhile, the approach has been tested and validated on the public Fusion360 dataset [4], and it achieves 99.81% and 99.59% in reconstructing the wireframe and faces, respectively. The method has also been tested on samples extracted from the ABC dataset [5]. The complete dataset is also made publicly available for future benchmarking of other algorithms at the following URL: `https://doi.org/10.5281/zenodo.7785223`.

This paper is organized as follows. Section 2 depicts the prior works related to 3D reconstruction from 2D drawings. Section 3 introduces the overall methodology for the automatic reconstruction of 3D CAD models and details the modules. Section 4 describes the implementation and evaluation of the approach, and section 5 concludes the paper and discusses future works.

2. Related works

Automatically reconstructing 3D CAD models from 2D orthographic drawings has been a long-standing problem since the 1970s. The approach of converting 2D drawings to B-Rep models was first formalized in [6, 7]. Since that time, several ideas have been explored and are discussed in this section. This ranges from the construction of intermediate wireframes that then serve as a support to reconstruct the 3D CAD models, to more recent deep learning-based approaches.

3D wireframe reconstruction. The automatic conversion of 2D drawings to 3D wireframes is the early stage of the 3D reconstruction approach. Several methods focused on generating polyhedral objects from their orthographic views [8]. Furferi et al. [9] developed a MatLab tool that reconstructs 3D wireframes from 2D vectorial drawings by labeling of vertices and topological representation of edges. Gorgani et al. [10] use the fuzzy theory based on expert knowledge to analyze and estimate the wireframe and surface from 2D drawings. However, these kinds of methods did not consider wireframes with curved edges. Since the curved edges and surfaces are the essential parts of CAD models, special attention must be paid to dealing with this problem. Lu et al. [11] proposed a quadric surface fitting method to extract feature curve networks from 3D models. But, the method is not working for 2D data. Zhang et al. [12] used five points method to construct conic curves by matching projections in different views. However, the method may fail to determine the corresponding relationships between the three views because the points are arbitrarily chosen on 2D curves. Gong et al. [13] proposed a method for reconstructing 3D wireframes of curved objects based on decision trees. The decision trees search the projective properties in three views and then construct 3D edges, but, the algorithm complexity is high because of the duplicated decision trees. As an extension, in [14, 15], the authors presented hybrid wireframe, which consists of surface features, edges, and vertices to complete the information of wireframe. High algorithmic complexity and limitations of curved edge handling make these methods limited in practice.

3D CAD model reconstruction. To obtain the reconstructed 3D models, Gong et al. [14, 15] convert the hybrid wireframe to the B-Rep model by graph theory and surface features matching methods. However, the hybrid wireframe is complex to define,

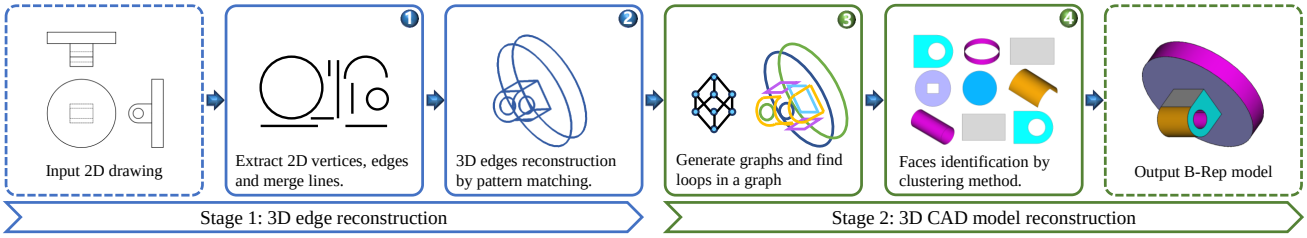

Fig. 2. Overview of the rule-based approach that automatically generates a watertight and ready-to-use 3D CAD model (right) from a 2D orthographic drawing (left). 2D edges are first extracted (Step 1) and used as input of a 3D edges reconstruction algorithm based on pattern matching (Step 2). The resulting wireframe is then analyzed to define all possible loops of edges (Step 3) that are then clustered to identify the faces of the B-Rep model (Step 4).

and it has to compute the relationships between edges for each view, such as adjacency, symmetry, and coincidence. In addition the surface features are obtained with edges matching between three views. Therefore, those operations have to traverse the edge space of the 2D drawings several times. To find surfaces in wireframes, Varley et al. [16] proposed a polynomial-order algorithm that handles the 3D wireframes, and find the possible loops of faces in the wireframe based on the shortest-path algorithm. Zakharov et al. [17] introduced a spectral graph matching algorithm to construct 3D faces of models from wireframes. In the method, the faces are constructed based on closed loops. However, these methods cannot solve the coplanar problem of two nested loops. Moreover, some of these approaches work only for 3D models with planar surfaces [8, 10, 16].

In the work of [18], the authors proposed an approach that reconstructs the entities by extrusion and revolution operations from 2D drawings, but only two types of features (holes and pockets) are recognized in the method. A hint-based method is proposed by Lee et al. [19] to handle solids of revolution from orthographic views. In their method, circular edges are as the clues in the first view, and vertices and edges in other views will match the circular edges to construct objects. In [20], the authors presented an approach based on elements of knowledge retrieval and graph theory methods to represent 3D models. The local graphs are merged into a global graph and then transformed into 3D models. However, defining a knowledge system that includes all types of features in 2D drawings is difficult. In [21], a method of reconstructing 3D models from 2D drawings by extracting features is proposed. In this method, the sketches of cuboids, cylinders, holes, and fillets are defined as features. Moreover, Harish et al. [22] have introduced a method that uses OpenCV to detect the contours from the 2D views. Alwan [2] proposed a hybrid method to convert images of technical drawings to vector data. Features extracted from images were utilized to recognize the shapes. However, these methods based on feature recognition are not sufficiently generic and only work on limited types of 2D drawings and 3D shapes. These limitations impede the reconstruction methods to efficiently process the more complex 3D CAD models.

Deep learning models. Recently, there is increasing interest in extending deep learning-based approaches for 3D CAD modeling [23, 24, 25, 26, 27, 28], such as transformer and CNN-based neural networks. Li et al. [25, 29] proposed a sketching sequence-based CAD modeling method, in which the sketches are parsed to CAD operations and the line groups of sketches are segmented by a sequence-to-sequence neural network. In addition, Wu et al. [26] represent a CAD model as a sequence of CAD commands, which can then be converted into a B-Rep model. In [28], Willis et al. proposed a transformer-based engineering sketch generation network to address the task of engineering drawing generation. And in [4], the authors construct CAD models using the sketch and extrude sequence. Wang et al. [24] presented a face identification method that uses a transformer-based classification network to find edge sequences from line drawings. The deep learning-based approaches have achieved impressive results for CAD modeling. However, to the best of our knowledge, there is no model accepting orthographic drawings as input. Finally, such deep learning-based approaches never reach 100% of successful reconstruction (e.g. creation of non-manifold models, missing faces), and it can sometimes be more difficult to exploit the generated models in industrial processes than to start from scratch.

In conclusion, in this paper, some of the identified drawbacks are overcome and the proposed approach handles curved edges and non-planar multi-trimmed surfaces, with a very good success rate and very efficiently in terms of computation time.

3. Automatic 3D CAD model reconstruction

3.1. Overall approach and hypotheses

This paper introduces a rule-based two-stage approach that reconstruct 3D CAD models from 2D orthographic drawings in a fully automatic manner. The two main stages are shown in figure 2, each composed of several steps. First, in a preprocessing step, 2D drawings are input from a *.svg* file, and geometric entities contained in the file are parsed. In step 1, all three views are separated, and 2D edges and vertices with 2D coordinates are extracted from each view. The pattern matching algorithm then reconstructs 3D edges from 2D edges in step 2. At the end of the first stage, the complete wireframe is reconstructed. In the second stage, step 3, the co-edges with vertices are converted to a graph, and all possible loops of edges are detected. In step 4, the clustering algorithm composes the loops to identify faces. Once the faces have been identified, trimming and stitching is performed using the associated loops of edges. Finally, a watertight and ready-to-use B-Rep model is output. The approach relative to these steps is detailed in sections 3.2 and 3.3.

In order to cope with the widely varying possibilities offered by the norms, as well as with the freedom they allow to repre-

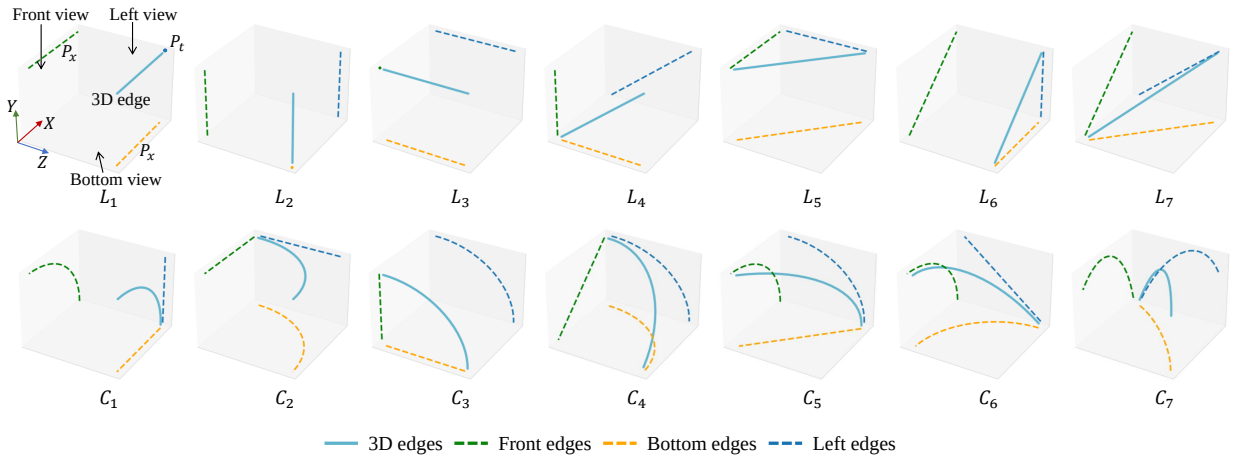

Fig. 3. Taxonomy of the 14 possible patterns of 3D edges (straight edge L_i or curved edge C_i) together with their 2D edge features in the 2D views (front, bottom and left) based on the projective properties of 2D orthographic drawings.

sent any 3D shapes by means of 2D orthographic views, several hypotheses are to be formulated:

- Input 2D drawings are provided by means of *.svg* files, i.e. they are already vectorized or have been vectorized prior to entering the reconstruction process, e.g. from images, but this additional step is not detailed in this paper and the reader is invited to refer to known techniques;

- The 2D drawings are considered as complete, i.e. they contain all the information needed to fully define all the shapes of the 3D counterpart. A single part is represented, no composite drawings. There are enough views, and all vertices, edges, hidden lines, axes are available to avoid ambiguity in defining the final shape of the object. Only manifold CAD models are considered;

- They are drawn using current norms (either US or EU), but all the possibilities offered by the norms are not yet exploited. For instance, partial views or hatched sections are not handled in this paper;

- The dimensions of the reconstructed 3D CAD model can be directly obtained from the coordinates of the vertices in the 2D views, possibly corrected by a scale factor. To cover most cases, and without any restrictions, three views are used in this paper: the Front, Bottom, and Left views are (X, Y), (X, Z), and (Z, Y) respectively;

- All coordinates of the vertices, edges, and faces in the 2D drawings and CAD models are within a tolerance consistent with the ones of the CAD modelers.

The above-mentioned hypotheses clearly circumscribes the field of application of the approach in order to prepare the validation stage. Anyhow, some of these hypotheses are already being reconsidered in our current researches, for instance the possibility to deal with any configuration not necessarily using Front, Bottom and Left views.

3.2. Reconstructing 3D edges from a 2D orthographic drawing

This step raises two central questions: how to identify the 2D edges in multiple views of a 2D orthographic drawing and extract all the information useful for the reconstruction, and how to match the 2D edges to restore the corresponding 3D edges. The principle of the pattern-matching method is first presented, and its two steps are formalized in the next subsections.

3.2.1. Principle of the pattern-matching method

The 2D edges in the orthographic views are generated from 3D edges using the parallel projection principle. Thus, the projective properties of 3D edges make the 2D edges in three views have unique patterns, which are useful for determining and reconstructing the 3D edges. More precisely, several rules stand. A straight edge in 3D space must be projected into at least two straight segments and at most one vertex in the three views. Similarly, a curved edge must be projected into at least one curved edge, and at most two straight segments. These characteristics of the three views of a 2D drawing make up the unique patterns to match the different types of 3D edges. Possible patterns are shown in figure 3, and each of it is characterized by a 2D edge feature in each view as listed in table 1. In this table, P_x, P_y and P_z characterize projections parallel to x-axis, y-axis and z-axis respectively, P_t refers to a projection that is a vertex, I if the projection is inclined with respect to the reference frame, and A if it is an arc, i.e. a curved projection. For instance, L_1 is a straight 3D edge whose projection in the front and bottom views are aligned with the x-axis, and whose projection in the left view is a vertex. Reversely, if a 2D edge is parallel to the x-axis in the front view, and if one can find both another 2D edge in the bottom view that is also parallel to the x-axis and with the same x coordinates, and a vertex in the left view with same (z, y) coordinates, then these three features (P_x, P_x, P_t) match and the 3D edge with pattern L_1 can be reconstructed. This reconstruction exploits the x, y and z coordinates extracted from the 2D drawing, and these values are used to identify possible patterns.

Table 1. Patterns of 3D edges and their 2D edge features in the different views.

	3D edges	2D edge features		
		Front (F_f)	Bottom (F_b)	Left (F_l)
Lines	L_1	P_x	P_x	P_t
	L_2	P_y	P_t	P_y
	L_3	P_t	P_z	P_z
	L_4	P_y	P_z	I
	L_5	P_x	I	P_z
	L_6	I	P_x	P_y
	L_7	I	I	I
Curves	C_1	A	P_x	P_y
	C_2	P_x	A	P_z
	C_3	P_y	P_z	A
	C_4	I	A	A
	C_5	A	I	A
	C_6	A	A	I
	C_7	A	A	A

3.2.2. *Extraction of 2D edges (Step 1)*

The 2D edges and vertices taking part to the reconstruction process are extracted from a *.svg* file. Therefore, the file is first parsed to extract the views containing all the information. The front, bottom, and left views are (X, Y), (X, Z), and (Z, Y), respectively. In this paper, three types of 2D edges are considered: line, arc of circle and circle. In addition, each edge is bounded by two vertices that are the start and end points of the edge, and their coordinates are to be used to reconstruct the 3D edges. However, for an arc of circle, an additional point is needed to fully defined its geometry, and this information is to be post-processed while computing an additional vertex in the middle of the arc. This extra point corresponds to (z_m, y_m) in the left view of figure 4. Similarly, for a circle, as the start and end points match, two extra points are to be defined, one located for instance in the first quarter, i.e. (x_q, y_q), and one in the middle, i.e. (x_m, y_m) in the example of figure 4. All the vertices are stored within their respective sets $\mathbb{V}_f$, $\mathbb{V}_b$, and $\mathbb{V}_l$, corresponding to the front, bottom, and left views, respectively. Each 2D edge refers to either a visible or hidden line.

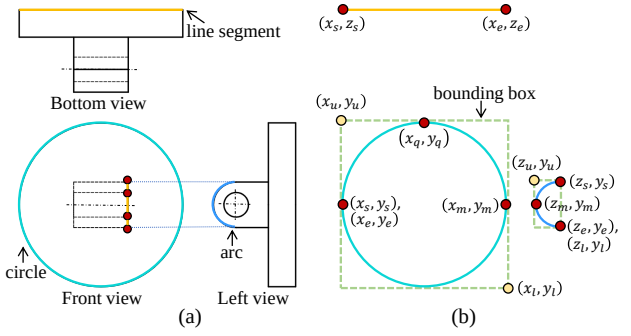

Fig. 4. Three types of 2D edges: line, arc of circle, and circle (a); and the corresponding sampling strategies and bounding box (b).

According to the retrieved coordinate values, the 2D edge feature can be assigned and the bounding box can be computed for each edge. These information will then be exploited for 3D edge reconstruction using pattern matching. Thus, a 2D edge is flagged as either a P_x, or P_y, or P_z, or I or A in the corresponding view (Table 1). The axis-aligned bounding box of a 2D edge is a minimal rectangle containing the 2D edge and aligned with the axes of the reference frame (Fig. 4.b). Each bounding box includes upper-left and lower-right corner coordinates. For the case of a straight 2D edge, i.e. either an axis-oriented or an inclined one, the bounding box is reduced to the coordinates of its start and end points.

Unfortunately, the 2D drawing does not provide all of the 2D linear edges of the entities (Fig. 4.a), and for instance for the arc in the left view it is impossible to find an explicitly matched line segment in the front view because the regarding line segment was split into three lines by the vertices. In other words, the 2D edges regarding the spatial edges of each view are not always available in the 2D drawing file. Thus, the line segments are merged into possible matchable line segments before further processing. There can be any number of edges n waiting for merge. The edges candidate to the merge should be defined so that the line segments are collinear and they share vertices. The start and end coordinates of the merged line segments are from the candidate edges (unshared vertices). In this paper, $n = 2$ and 3, i.e., 2 and 3 line segments are merged to the new edges and represent them in the same format as the extracted edges.

All extracted and merged 2D edges participate to the definition of the edge sets $\mathbb{E}_f$, $\mathbb{E}_b$ and $\mathbb{E}_l$, one for each view:

$$\mathbb{E}_f = \left\{e_f^1, e_f^2, \cdots, e_f^{N_f} \mid e_f^i = \{P_f^i, B_f^i, F_f^i\}, i \in [1..N_f]\right\} \quad (1)$$

$$\mathbb{E}_b = \left\{e_b^1, e_b^2, \cdots, e_b^{N_b} \mid e_b^j = \{P_b^j, B_b^j, F_b^j\}, j \in [1..N_b]\right\} \quad (2)$$

$$\mathbb{E}_l = \left\{e_l^1, e_l^2, \cdots, e_l^{N_l} \mid e_l^k = \{P_l^k, B_l^k, F_l^k\}, k \in [1..N_l]\right\} \quad (3)$$

where N_f, N_b, and N_l are the number of edges in respectively the front, bottom, and left views. The coordinates of the end points of the 2D edges, as well as the coordinates of the possibly added sampled points for arcs or circle and circles, are defined as follows:

$$P_f^i = \left[(x_f^{is}, y_f^{is}), \cdots, (x_f^{ie}, y_f^{ie})\right], i \in [1..N_f] \quad (4)$$

$$P_b^j = \left[(x_b^{js}, z_b^{js}), \cdots, (x_b^{je}, z_b^{je})\right], j \in [1..N_b] \quad (5)$$

$$P_l^k = \left[(z_l^{ks}, y_l^{ks}), \cdots, (z_l^{ke}, y_l^{ke})\right], k \in [1..N_l] \quad (6)$$

The coordinates of the two corners, i.e. the upper-left (u) and lower-right (l) corners, of the axis-aligned bounding boxes are defined as follows:

$$B_f^i = \left[(x_f^{iu}, y_f^{iu}), (x_f^{il}, y_f^{il})\right], i \in [1..N_f] \quad (7)$$

$$B_b^j = \left[(x_b^{ju}, z_b^{ju}), (x_b^{jl}, z_b^{jl})\right], j \in [1..N_b] \quad (8)$$

$$B_l^k = \left[(z_l^{ku}, y_l^{ku}), (z_l^{kl}, y_l^{kl})\right], k \in [1..N_l] \quad (9)$$

The 2D edge features in the different views are defined as follows (Table 1):

$$F_f^i, and\ F_b^j, and\ F_l^k \in \left\{P_x, P_y, P_z, I, A\right\} \quad (10)$$

Finally, each edge set $\mathbb{E}_f$, $\mathbb{E}_b$ and $\mathbb{E}_l$ is parsed to remove duplicate 2D edges in the different views. These duplicates appear because the 2D drawings are automatically generated by CAD software. This may happen when for instance a hidden line (represented by a dashed line) is perfectly coincident with a visible one (represented by a continuous line). However, there would be no duplicates if the 2D drawings were for instance originated from the vectorization of an image.

3.2.3. Pattern matching-based 3D edge reconstruction (Step 2)

The pattern matching method relies on the comparison of the 2D edge features and bounding boxes in the different views. The 2D edge features of the edges in $\mathbb{E}_f$, $\mathbb{E}_b$ and $\mathbb{E}_l$ are first matched and the pattern of each candidate 3D edge is identified. Then, for each candidate 3D edge, the bounding boxes of the corresponding 2D edges are compared, and if they match then the 3D edge is reconstructed directly using the x, y, and z coordinates of the matched 2D edges. Details of the method are provided in Algorithm 1. The matching of the bounding boxes is evaluated using the following criteria:

$$Matched(B_f^i, B_b^j) \leftarrow \sqrt{\left(x_f^{iu} - x_b^{ju}\right)^2 + \left(x_f^{il} - x_b^{jl}\right)^2} < \epsilon_t \quad (11)$$

$$Matched(B_f^i, B_l^k) \leftarrow \sqrt{\left(y_f^{iu} - y_l^{ku}\right)^2 + \left(y_f^{il} - y_l^{kl}\right)^2} < \epsilon_t \quad (12)$$

$$Matched(B_b^j, B_l^k) \leftarrow \sqrt{\left(z_b^{ju} - z_l^{ku}\right)^2 + \left(z_b^{jl} - z_l^{kl}\right)^2} < \epsilon_t \quad (13)$$

When one of the criterion is met, then the corresponding bounding boxes are called matched bounding boxes. If the three conditions are satisfied, then a complete match between the three views is obtained, and this identifies one of the patterns between L_4 and C_7 (Table 1). Here, ϵ_t is a threshold that has been tuned empirically to $\epsilon_t = 0.005$ to take into account the accuracy issues.

Algorithm 1 Pattern matching-based 3D edge reconstruction.

Require: 2D edges of three views.
Ensure: Reconstructed 3D edges.

$\mathbb{E}_f = \left\{e_f^1, e_f^2, \cdots, e_f^{N_f} \mid e_f^i = \left\{P_f^i, B_f^i, F_f^i\right\}, i \in [1..N_f]\right\}, \mathbb{V}_f$;
$\mathbb{E}_b = \left\{e_b^1, e_b^2, \cdots, e_b^{N_b} \mid e_b^j = \left\{P_b^j, B_b^j, F_b^j\right\}, j \in [1..N_b]\right\}, \mathbb{V}_b$;
$\mathbb{E}_l = \left\{e_l^1, e_l^2, \cdots, e_l^{N_l} \mid e_l^k = \left\{P_l^k, B_l^k, F_l^k\right\}, k \in [1..N_l]\right\}, \mathbb{V}_l$.
for e_f^i in $\mathbb{E}_f$ **do**
  for e_b^j in $\mathbb{E}_b$ **do**
    if $(F_f^i = P_x)$ & $(F_b^j = P_x)$ **then** ▷ see Table 1 L_1.
      Compute: $P_t = (z_b^j, y_f^j)$; $z_b^j \in P_b^j, y_f^i \in P_f^i$.
      if $Matched(B_f^i, B_b^j)$&$(P_t \in \mathbb{V}_l)$ **then**
        Reconstruct L_1 **3D edge.**
        Continue
      end if
    end if
    for e_l^k in $\mathbb{E}_l$ **do**
      if $(F_f^i = P_y)$ & $(F_l^k = P_y)$ **then** ▷ see Table 1 L_2.
        Compute: $P_t = (x_f^i, z_l^k)$; $x_f^i \in P_f^i, z_l^k \in P_l^k$.
        if $Matched(B_f^i, B_l^k)$&$(P_t \in \mathbb{V}_b)$ **then**
          Reconstruct L_2 **3D edge.**
        end if
      end if
      if $(F_b^j = P_z)$ & $(F_l^k = P_z)$ **then** ▷ see Table 1 L_3.
        Compute: $P_t = (x_b^j, y_l^k)$; $x_b^j \in P_b^j, y_l^k \in P_l^k$.
        if $Matched(B_b^j, B_l^k)$&$(P_t \in \mathbb{V}_f)$ **then**
          Reconstruct L_3 **3D edge.**
        end if
      end if
      if $\left(F_f^i, F_b^j, F_l^k\right)$ match a pattern in $\{L_4 \sim C_7\}$ **then** ▷ see Table 1 $L_4 \sim C_7$.
        if $Matched(B_f^i, B_b^j, B_l^k)$ **then**
          Reconstruct the 3D edge matched in $L_4 \sim C_7$**.**
        end if
      end if
    end for
  end for
end for

3.3. Identification of faces from 3D edges

Once the 3D edges and wireframe reconstructed, the next step consists in identifying the loops of edges taking part to the definition of the faces of the B-Rep model. This is the second stage of the proposed approach and it is composed of two steps (Figure 2): detection of loops in a graph (Step 3), and then clustering of the loops to define the faces and their inner and outer boundaries used for the trimming operations (Step 4).

3.3.1. Edge loop detection (Step 3)

Starting from the obtained 3D wireframe, a graph $G = (V, E)$ is created where the nodes $V_i \in V$ are connected by links $e_k \in E$, and where each node (resp. link) of the graph corresponds to exactly one vertex (resp. edge) of the 3D wireframe. Here, the links in the graph then correspond to edges in the wireframe and to co-edges in the B-Rep model. Thus, each link in the graph is involved in two loops, and the related 3D edge joins two vertices and is shared by exactly two faces. This is illustrated on figure 5.a, where e_5 is a co-edge pointing from V_4 to V_5 and shared by faces f_1 and f_2.

A face of a B-Rep model is bounded by an alternating cycle of vertices and edges forming an outer loop in which no vertex or edge appears twice. Optionally, several inner loops can also be used to trim the face, and define more complex shapes such as holes or pockets. To find the loops and faces more efficiently, a new method based on the unique planar surface is proposed. This allows us to avoid searching through the space of all possible edge loops. Basically, starting from a randomly chosen edge, the loop identification process tries to successively append vertices and edges to the loop until forming a closed loop, where all vertices are co-planar. Then, for each initialized edge, all the loops involved in it are identified at once, and the initialized edge is removed from the graph. These operations efficiently reduce the search space for finding loops of a face. Details are provided in Algorithm 2.

Some steps of the proposed edge loop detection algorithm are illustrated in figure 5, wherein hidden lines have been omitted for sake of clarity. Figure 5.b shows how Loop #1 can be defined starting from the random start edge e_5 with two vertices V_4 and

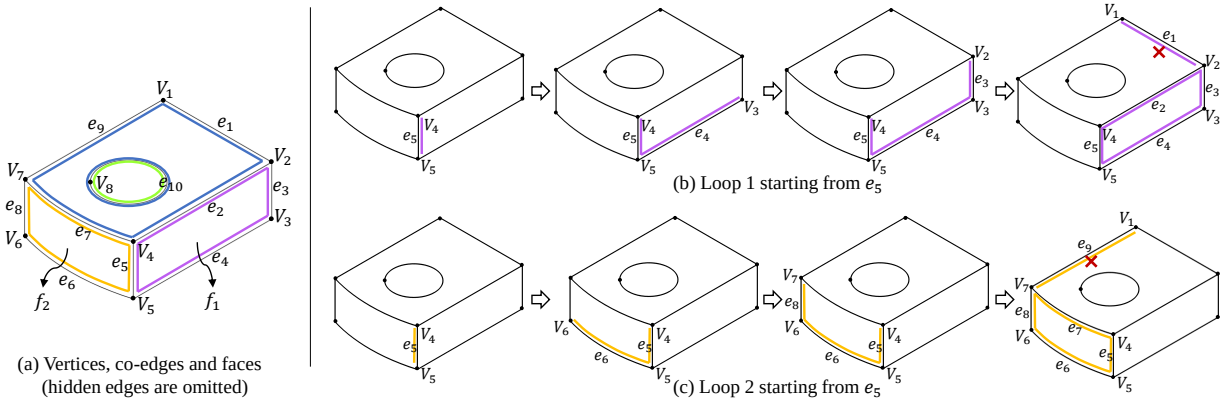

Fig. 5. Detection of edge loops in a wireframe composed of co-edges connecting exactly two faces of the B-Rep model to be reconstructed: a) loops of co-edges taking part to the definition of the faces, b) identification steps of loop #1 starting from a randomly selected co-edge e_5, c) identification steps of loop #2 associated to the same co-edge e_5.

Algorithm 2 Detection of vertices-edges loops.

Require: Edges-vertices Graph, $G = (V, E)$.
Ensure: Edges-vertices loops.

```
G = {e_1, e_2, ..., e_n | e_i = (V_j, V_k)}   ▷ e_i is the 3D edge; V_j, V_k are the vertices in 3D space.
while G not empty do
    e_i, randomly selected from G.
    V_j, V_k ← e_i.                          ▷ Start node and next node of the loop.
    Initialize: ℓ_c ← [V_j, V_k]             ▷ Initialization loops ℓ_c.
    𝕍_n ← G(V_k)                             ▷ 𝕍_n is a set of nodes connected with V_k.
    Initialize: S_d[i] ← {V_k : 𝕍_n}         ▷ S_d is a dictionary with nodes and connected nodes.
    while S_d not empty do
        V_p, 𝕍_c ← S_d[l]                    ▷ Index last element (l) in S_d to get next node V_p and its connected nodes set 𝕍_c.
        for V_c^i in 𝕍_c do
            if V_c^i not in ℓ_c then
                Compute: n⃗_c and d_c          ▷ Compute normal and bias of last three vertices in ℓ_c.
                if (n⃗_c, d_c)=(n⃗_l, d_l) then
                    ℓ_c ← ℓ_c[l] = V_c^i      ▷ Append V_c^i to ℓ_c and update.
                    𝕍_n^i ← G(V_c^i)          ▷ New next node, V_c^i, and its connected nodes set, 𝕍_n^i.
                    S_d[l] ← {V_c^i : 𝕍_n^i}  ▷ Append V_c^i and 𝕍_n^i to S_d and update it.
                    (n⃗_l, d_l) ← (n⃗_c, d_c)  ▷ Update last normal/bias.
                end if
            end if
            if V_c^i in ℓ_c then
                Find a loop
            end if
            Remove V_c^i from 𝕍_c.
        end for
        if 𝕍_c is empty then
            Remove V_p, 𝕍_c from S_d, and update S_d.
            Remove V_p from ℓ_c and update ℓ_c.
        end if
    end while
    Remove e_i from G and update G.
end while
```

V_5. Suppose V_4 is the start vertex and V_5 is the so-called next vertex. Thus, the initialized loop is $[V_4, V_5]$. At this step, V_5 has two unvisited neighbor vertices V_3 and V_6 connected to V_5 by e_4 and e_6, respectively. The unvisited neighbor nodes are stored in a dictionary data structure S_d. Here $S_d = \{V_5 : [V_3, V_6]\}$, where V_5 is the next node as key and $[V_3, V_6]$ is the unvisited neighbor nodes as value. Once the neighbor node is visited, it will be removed from the value list. Firstly, the edge e_4 pathing to V_3 is selected and appended to the loop list $[V_4, V_5, V_3]$. The three vertices now uniquely determine a planar surface whose equation can be obtained as follows:

$$a.x + b.y + c.z + d = 0 \tag{14}$$

with d the bias, and $\vec{n} = (a, b, c)$ the normal. Then, the next vertex from V_5 to V_3 is updated, i.e. the last element of the list. The next vertex V_3 has an unvisited neighbor V_2 connected by edge e_3. Since the normal and bias of the planar surface $[V_5, V_3, V_2]$ are equal to the normal and bias of the planar surface $[V_4, V_5, V_3]$, it means that the vertices V_4, V_5, V_3, and V_2 are co-planar. Thus, the vertex V_2 can be added to the loop list, $[V_4, V_5, V_3, V_2]$. Again, the next vertex is V_2 which could be linked to V_1 and V_4. However, there is no unique planar determined by the vertices V_5, V_3, V_2, and V_1, which means that the vertex V_1 is not co-planar with V_5, V_3, and V_2. Therefore, the vertex V_1 and edge e_1 are dropped in this loop, and this avoids continuing in this part of the wireframe. The next vertex is back to vertex V_2, and the vertex V_4 can be visited. As the vertex V_4 is already in the loop list, the first loop involving e_5 is found.

Similarly, the second loop involving e_5 can be identified, i.e. $[V_4, V_5, V_6, V_7]$ in this case, as shown in figure 5.c. However, the face f_2 is not a planar face but a curved one. To address this problem, a simple but yet efficient method has been developed and relies on a face virtual replacement. In figure 6, if a curved surface contains all vertices, it can be virtually replaced during the operation of finding loops by a planar surface that contains all of these vertices. Obviously, the edge information are kept, and for instance edges e_6 and e_7 in figure 5 are still arcs of circle represented by three points. The virtually replaced planar surface is only used to calculate the co-planar relationship, which does not influence the curved surface reconstruction.

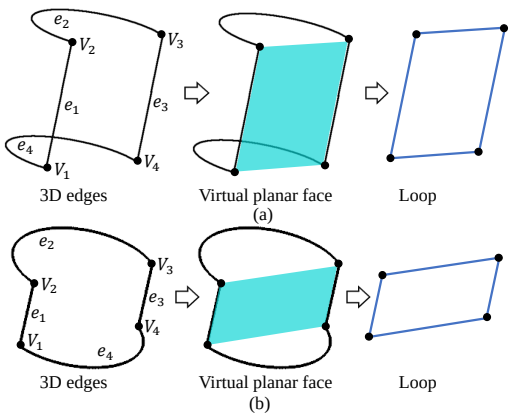

Fig. 6. Treatment of curved edges and surfaces by means of a virtual planar face that can be used to calculate the co-planar relationships.

3.3.2. *Face identification by loop clustering (Step 4)*

Each face of a B-Rep model is bounded by an outer loop of edges and, in some cases, it can also be trimmed with inner loops to model holes or pockets for instance. In step 4, a method to identify the faces from the set of loops obtained in step 3 is proposed. The face identification problem is treated as a loop classification problem. However, the number of faces and the type of faces are uncertain for each CAD model. Since there is no classification method based on supervised learning that can handle such an issue of uncertain categories, a clustering method is proposed to compose the loops defining the different faces. Figure 7 shows various configurations where faces have been identified from the detected loops using the proposed approach detailed in Algorithm 3. Inner loops are here used to trim the surfaces. The input is the full list $\mathbb{L}_s$ of N_ℓ loops ℓ_i found in the graph:

$$\mathbb{L}_s = \{\ell_1, \ell_2, \cdots, \ell_{N_\ell} \mid \ell_i = (E_i, (\vec{n}_i, d_i), B_i)\} \tag{15}$$

where E_i is the set of edges, $\vec{n}_i$ and d_i are the normal and bias of the planar face associated to the loop, and B_i is the bounding box of the loop. The set of loops is reverse sorted by the diagonal length of the bounding box. The loop are then clustered following several rules. If a face is bounded by a single loop, then the conversion from loop to face is simple (Figure 7.a). However, the faces generally include more than one loop. If a set of loops exactly fixes a planar face, the loops should satisfy two conditions: the normals and bias are equal with respect to the given accuracy, and there is the largest loop that covers all of the others (Figure 7.b). More precisely, given a new loop ℓ_i, its co-planar face is found in the set of planar faces F_p, if the euclidean distance between $(\vec{n}_i, d_i)$ and the co-planar face is under a threshold. Then, the containment relationship between the bounding box B_p^k of the co-planar face and the bounding box B_i of ℓ_i is computed. If B_p^k covers B_i, it means that ℓ_i belongs to the co-planar face, and thus ℓ_i is added to the co-planar face and the list of planar faces F_p is updated. In other situations, the new loop could not find the co-planar face, and could not be covered by the existing bounding box, which means a new face is found, and thus a new cluster is built and F_p is updated. After

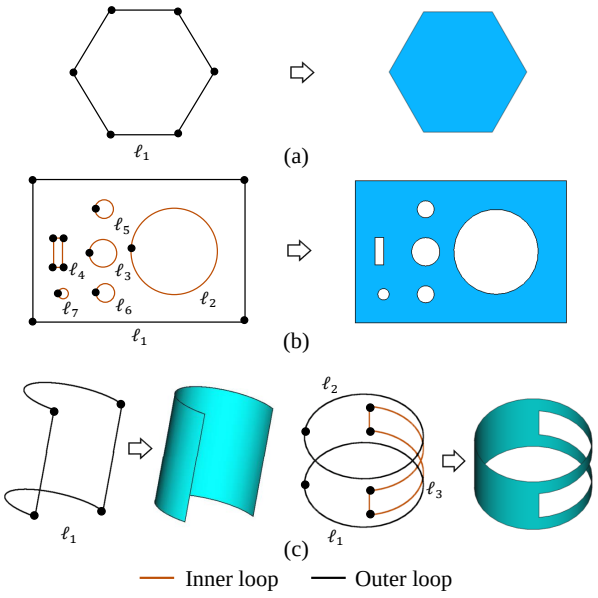

Fig. 7. Face identification from detected loops: (a) planar face bounded by a hexagonal outer loop, (b) planar face bounded by a rectangular outer loop, and trimmed by several inner loops, (c) cylindrical faces bounded with various outer loops, and possibly trimmed with inner loops.

Algorithm 3 Face identification by loops clustering.

Require: Edges-vertices loops.
Ensure: Loop clusters of faces.

$\mathbb{L}_s = \{\ell_1, \ell_2, \cdots, \ell_{N_\ell} \mid \ell_i = (E_i, (\vec{n}_i, d_i), B_i)\}$.
▷ E_i set of 3D edges, $(\vec{n}_i, d_i)$ normal and bias, B_i bounding box.
Initialize: F_p, F_c ▷ F_p set of planar faces; F_c set of curved faces.
while clusters of faces are changed **do**
  $\ell_i \leftarrow \mathbb{L}_s[i]$ ▷ Randomly selected from $\mathbb{L}_s$.
  $d^{(k)} \leftarrow \arg\min_j \left\|(\vec{n}_i, d_i) - F_p^j\right\|^2$. ▷ Find the nearest normal d and index k in F_p.
  if $d < t$ **then** ▷ t is the tolerance of the same normal.
    $B_p^k \leftarrow F_p[k]$ ▷ B_p^k is the bounding box of face f_p^k in F_p
    if $(B_i \in B_p^k)$ **then**
      $f_p^k \leftarrow \ell_i$ ▷ Loop ℓ_i belongs to face f_p^k.
    end if
    if $(B_i \notin B_p^k)$ **then**
      $f_p^i \leftarrow \ell_i$ ▷ New cluster of planar face.
    end if
  end if
  if $d > t$ **then**
    $f_p^i \leftarrow \ell_i$ ▷ New cluster of planar face.
  end if
  Update F_p
  if curved edges in ℓ_i **then**
    $f_c^k \leftarrow \arg\min_j \left(\ell_i, F_c^j\right)$. ▷ Find nearest parallel face $f_c^k \in F_c$
    if $(B_i \in B_c^k)$ **then**
      $f_c^k \leftarrow \ell_i$ ▷ Loop ℓ_i belongs to face f_c^k.
    end if
    if $(B_i \notin B_c^k)$ **then**
      $f_c^i \leftarrow \ell_i$ ▷ New cluster of curved face.
    end if
  end if
  Update F_c
end while

traversing the set of loops, all planar faces of the B-Rep model are identified.

The same strategy can be then used to find curved faces as well. However, the rules should be changed. A loop of a curved face should contain at least two curved edges, i.e. arcs of circle and circles, the bounding boxes of the loops should be parallel, the loops should not be co-planar, the axes of the loops should be aligned, and the largest bounding box should cover the others. Figure 7.c shows examples of curved faces. The loop including two arcs corresponds to a half-cylinder, and the circles and loop determine a cylinder with a hole. Two loops of circles determine a whole cylinder and only the nearest two circles are considered to construct a cylinder. The distance between the loops is a constraint to avoid the duplication of detected faces. Moreover, the reconstructed faces are constrained by the co-edges attribute, i.e., each edge is exactly shared by two faces.

4. Results and discussion

This section details the dataset, the evaluation metrics and the results obtained using the automatic 3D CAD model reconstruction technique developed in this paper.

4.1. Dataset and evaluation metrics

The dataset used for validation is built on the Fusion360 database [4]. The original 3D shapes are B-Rep models that are projected using parallel projection rule to get orthographic 2D drawings using FreeCAD software. The generated 2D drawings are stored in *.svg* files and are composed of three views (see hypotheses in section 3.1). The visible edges are represented with continuous lines, and the hidden edges are represented as dashed lines. This dataset is a subset of the Fusion360 dataset because the shapes with B-splines have been filtered out and the duplicated shapes have been removed. It finally consists of 2981 samples. The number of edges of the CAD models varies from 3 to 447, and the number of faces ranges from 3 to 151. The complete dataset includes the original models, the generated 2D drawings and the reconstructed B-Rep models, and all of them are made publicly available for future benchmarking of other algorithms at the following URL: `https://doi.org/10.5281/zenodo.7785223`.

To evaluate the performance of the 3D reconstructions, the precision, recall, and F-score are computed at both the wireframe and B-Rep levels for each of the $N_s = 2981$ samples, and then averaged on the entire database. In this paper, precision is a measure to evaluate how many of the correct reconstructed edges/faces are in all reconstructed edges/faces. The recall is a measure to evaluate how many edges/faces are correctly reconstructed in all original edges/faces. And F-Score is a measure combining precision and recall using the harmonic mean.

At the level of the wireframes, the $| W_i |$ edges of a reconstructed wireframe W_i are compared to the $| W_i^* |$ edges of the original wireframe W_i^*, and the number of edges that match $| W_i \cap W_i^* |$ is computed. An edge e matches another edge e^* if and only if the coordinates of their vertices are equal. Thus, the average precision, recall, and F-score for the wireframes are respectively computed as follows:

$$\begin{cases} p_w = \dfrac{1}{N_s} \displaystyle\sum_{i=1}^{N_s} | W_i \cap W_i^* | \, / \, | W_i | \\ r_w = \dfrac{1}{N_s} \displaystyle\sum_{i=1}^{N_s} | W_i \cap W_i^* | \, / \, | W_i^* | \\ FS_w = \dfrac{2 p_w r_w}{p_w + r_w} \end{cases} \tag{16}$$

At the level of the B-Rep models, the $| F_i |$ faces of a reconstructed B-Rep F_i are compared to the $| F_i^* |$ faces of the original B-Rep F_i^*, and the number of faces that match $| F_i \cap F_i^* |$ is computed. A face f matches another face f^* if and only if the two sets of edges defining the two faces are equal. Thus, the average precision, recall, and F-score for the B-Rep models are respectively computed as follows:

$$\begin{cases} p_f = \dfrac{1}{N_s} \displaystyle\sum_{i=1}^{N_s} | F_i \cap F_i^* | \, / \, | F_i | \\ r_f = \dfrac{1}{N_s} \displaystyle\sum_{i=1}^{N_s} | F_i \cap F_i^* | \, / \, | F_i^* | \\ FS_f = \dfrac{2 p_f r_f}{p_f + r_f} \end{cases} \tag{17}$$

4.2. Experimental results

The proposed automatic 3D reconstruction algorithm has been implemented in Python language and it exploits the possibilities of FreeCAD software. It is run on an Intel i7-12700H with 16GB RAM and 2TB SSD hard disk.

The proposed approach has been tested on the full dataset containing the 2981 samples. Some examples are provided in figure 8, wherein each test case is illustrated with its input 2D orthographic drawing, as well as with the automatically reconstructed wireframe and watertight 3D B-Rep model. The first row shows three cases in which the 2D drawings contain inclined curves and inclined line segments that are perfectly treated by the two-stage algorithm. The second row provides some complex shapes with several hundreds of edges and about one hundred faces each. For instance, the last one in this row is a gearwheel that contains 312 edges and 106 faces, with a complex topological structure, which leads to a reconstruction processing time of 52s. The third row shows examples in which the approach identifies faces composed of several inner loops. For instance, the first CAD model in this row contains a face with ten loops. The fourth row shows reconstructed shapes with curved faces, such as nested cylinders, half-cylinders, and arrays of cylinders. The fifth row includes examples of torus and target faces. The last row present the reconstructed CAD models with the pads and holes. All these examples demonstrate how the proposed approach copes with widely varying configurations and examples.

The performance of the approach can also be analysed from the resulting values of precision, recall and F-Score as summarized in Table 2. Considering the first stage, and the reconstruction of 3D wireframe from 2D drawings, the F-score, recall, and precision are 99.81%, 99.90% and 99.72%, respectively, which is really very good. For the second stage, with the reconstruction of 3D faces from the wireframes, results are also very

Fig. 8. Automatic reconstruction of watertight and ready-to-use CAD models from 2D drawings generated from Fusion360 database [4]. The approach copes with a wide range of input configurations, including straight and curved lines, and reconstructing B-Rep models possibly composed of hundreds of trimmed faces.

Table 2. Recall, precision and F-score obtained for both wireframe and face reconstructions on the complete dataset of 2981 samples.

	recall(%)	precision(%)	F-score(%)
wireframe	99.90	99.72	99.81
face	99.44	99.75	99.59

good, with a F-score, recall, and precision of 99.59%, 99.44% and 99.75%, respectively. Figure 9 shows examples of reconstructions considered as failed with the adopted evaluation metrics. For instance, the first row shows an original model with two lateral planar faces whose 3D edges have been well reconstructed and appear correctly in the 3D wireframe. However, during face identification only one face has been reconstructed, as the method is able to detect the loop that includes the two faces. Here, even though the reconstructed model is considered as failed when compared to the original one, the obtained result is correct, and could be even considered as a better quality model than the original one. The second row shows another example for which the reconstructed wireframe differs from the original one, but for which the reconstructed B-Rep is the good one, thus demonstrating that the face detection algorithm has the capability to disregard extra edges of the 3D wireframe to produce good quality B-Rep model, which is the ultimate goal as the wireframe is only used as an intermediate representation. Thus, even if some reconstructions are considered as failed with respect to the adopted metrics, they are still valid and could

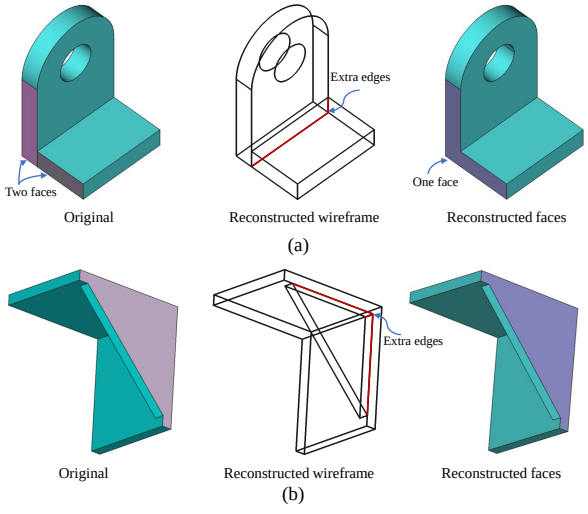

Fig. 9. Examples of reconstructions considered as failed with respect to the adopted evaluation metrics, but which can nevertheless be considered good overall: (a) the reconstructed B-Rep model differs from the original one as the two original lateral faces have been merged in a single one, (b) the reconstructed B-Rep model perfectly matches the original one, but the intermediate 3D wireframe is different.

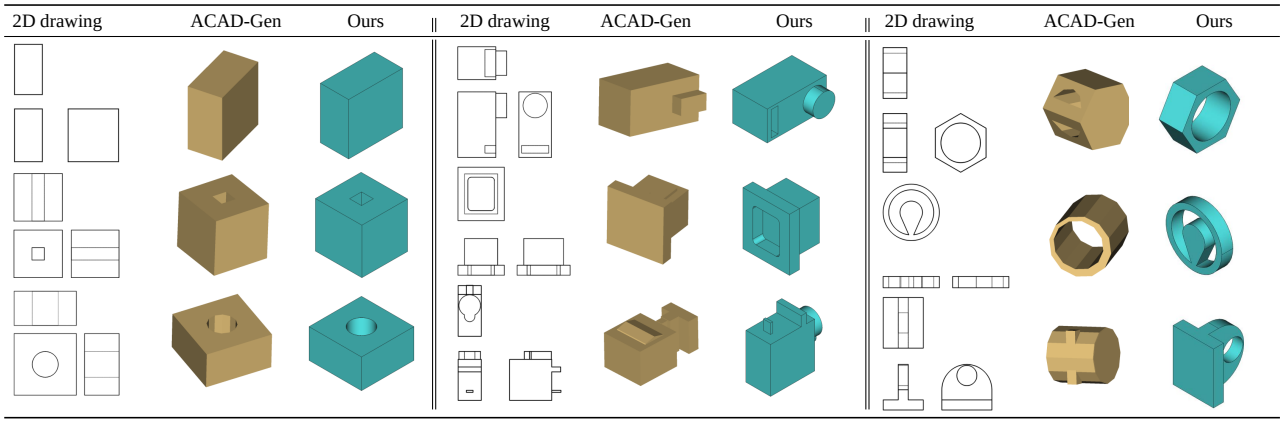

Fig. 10. Comparison results: input 2D drawings, reconstruction results by ACAD-Gen [22] , and results of the proposed approach.

even be considered as better for some of them.

The approach has also been compared to an existing method, ACAD-Gen [22]. To the best of our knowledge, this is the only related work that is open-source. Figure 10 shows the reconstruction results of the two approaches with the same input. Both methods successfully reconstruct simple models of the first column. However, ACAD-Gen cannot handle more complex models. For example, in column two, in the first sample, the circle is recognized as a rectangle, thus a cube is reconstructed instead of a cylinder; the pocket is missed in the second sample; in the last example a lot of details are lost and only some parts are reconstructed. Our approach clearly shows a better performance in the second column. The third column shows some failed cases for ACAD-Gen. Again, our approach correctly reconstructed the CAD models. In short, our approach could be handled more complex CAD models; the details are reconstructed well with good performance for curved faces and unregular planar faces.

The performance of the proposed approach can also be assessed through a computation time analysis. The complexity of the 3D CAD reconstruction directly depends on the number of entities in the input 2D drawings, on the geometrical coverage of the surfaces, and on the topology of the objects. Here, the efficiency is mainly affected by the wireframe construction, and especially for the objects with the larger amount of curved and inclined edges. This is illustrated on figure 11 that depicts the evolution of the computation time with respect to the number of faces of the CAD models to be reconstructed. When several models do have the same number of faces, the computation times are averaged. The proposed approach is efficient, and the CAD models with less than 20 faces are reconstructed within 200ms. The mean reconstruction time of models with 20−40 faces is under 700ms. It increases for models with more than 60 faces, for which the computation time is nearly 5s. The longest reconstruction time is about 70s for a model composed of about 150 faces. Figure 11 shows the fitted curve to analyze the computation time. When the number of faces rises 60, the error between the actual and fitted reconstruction time is also increased. This is because with the increasing complexity of the CAD models, the reconstruction time is also influenced by the number of edges and the topology of the models. At the end, the complete dataset composed of 2981 models is automatically reconstructed in 12.25min (735s), which is very good when compared to the time it would take to do it manually starting from scratch.

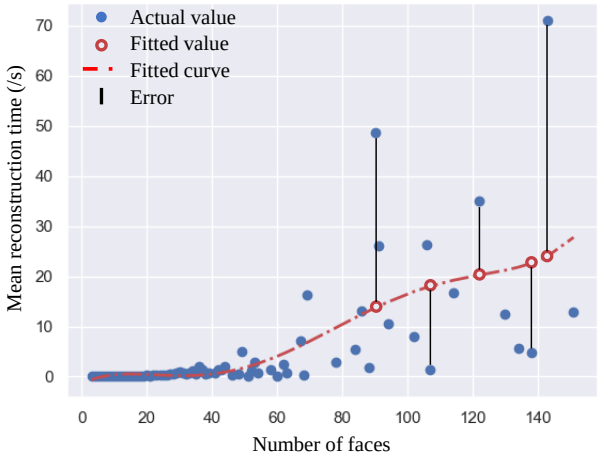

Fig. 11. Reconstruction times from 2D drawings to 3D CAD models as a function of the number of faces, and averaged when several models of the dataset have the same number of faces.

Finally, the practical performance of the proposed reconstruction approach has been evaluated on the public ABC dataset [5]. Figure 12 shows 3D wireframes and CAD models reconstructed from 2D drawings. Here again, results are very good and all of the test samples are reconstructed successfully.

5. Conclusion and future works

This paper proposes a two-stage approach to reconstruct watertight and ready-to-use 3D CAD models from 2D orthographic drawings. In the first stage, a pattern matching-based 3D edge reconstruction algorithm is used, and 3D edges of a wireframe are constructed by matching parallel projection features between the views of the 2D drawing. The method handles various types of 3D edges, including curved edges. In the second stage, a loop detection algorithm based on the unique planar surface is proposed, to detect coplanar loops in

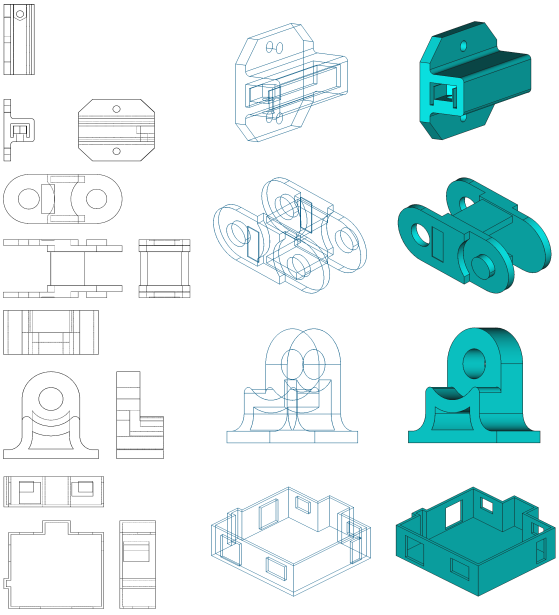

Fig. 12. Automatic reconstruction of CAD models from ABC dataset [5].

graphs. This method reduces the search space for possible loops of faces. Furthermore, a clustering algorithm is proposed for identifying and reconstructing faces defined by an outer loop and multiple inner loops used to trim the surfaces. The proposed approach efficiently and accurately reconstructs 3D models from various 2D drawings. It reaches 99.44% and 99.75% in mean recall and precision among 2981 samples, and a F-score of 99.81% which is very good. The detailed analysis of the failed reconstructions also shows that most of them are still valid, and that some of them can even be considered better than the original models. The computation times are also very good with most models reconstructed in few milliseconds.

Nevertheless, the proposed approach is suited for regular entities and cannot yet deal with complex faces, such as B-spline surfaces. Moreover, the method is limited to 2D drawings with the attributes mentioned in the hypotheses (section 3.1). However, a 2D drawing is usually mixed with dimensions and annotations, and possibly composed of additional partial views or hashed sections that the proposed approach is not yet able to capture. Thus, future works will concentrate on B-Spline surface reconstruction, and on the use of additional information extracted from annotations, partial views and hashed sections.

References

[1] Elyan, E, Jamieson, L, Ali-Gombe, A. Deep learning for symbols detection and classification in engineering drawings. Neural networks 2020;129:91–102.

[2] Alwan, S. Unsupervised and hybrid vectorization techniques for 3d reconstruction of engineering drawings. Ph.D. thesis; Ecole nationale supérieure Mines-Télécom Atlantique; 2021.

[3] Camba, JD, Company, P, Naya, F. Sketch-based modeling in mechanical engineering design: Current status and opportunities. Computer-Aided Design 2022;150:103283.

[4] Willis, KD, Pu, Y, Luo, J, Chu, H, Du, T, Lambourne, JG, et al. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. ACM Transactions on Graphics (TOG) 2021;40(4):1–24.

[5] Koch, S, Matveev, A, Jiang, Z, Williams, F, Artemov, A, Burnaev, E, et al. Abc: A big cad model dataset for geometric deep learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, p. 9601–9611.

[6] Markowsky, G, Wesley, MA. Fleshing out wire frames. IBM Journal of Research and Development 1980;24(5):582–597.

[7] Wesley, MA, Markowsky, G. Fleshing out projections. IBM Journal of Research and Development 1981;25(6):934–954.

[8] Yan, QW, Chen, CP, Tang, Z. Efficient algorithm for the reconstruction of 3d objects from orthographic projections. Computer-Aided Design 1994;26(9):699–717.

[9] Furferi, R, Governi, L, Palai, M, Volpe, Y. From 2d orthographic views to 3d pseudo-wireframe: An automatic procedure. International Journal of Computer Applications 2010;975:8887.

[10] Gorgani, HH, Pak, AJ, Sadeghi, S. 3d model reconstruction from two orthographic views using fuzzy surface analysis. European Journal of Sustainable Development Research 2019;3(2):em0081.

[11] Lu, Z, Guo, J, Xiao, J, Wang, Y, Zhang, X, Yan, DM. Extracting cycle-aware feature curve networks from 3d models. Computer-Aided Design 2021;131:102949.

[12] Zhang, A, Xue, Y, Sun, X, Hu, Y, Luo, Y, Wang, Y, et al. Reconstruction of 3d curvilinear wireframe model from 2d orthographic views. In: Int. Conf. on Computational Science. Springer; 2004, p. 404–411.

[13] Gong, JH, Zhang, GF, Zhang, H, Sun, JG. Reconstruction of 3d curvilinear wire-frame from three orthographic views. Computers & Graphics 2006;30(2):213–224.

[14] Gong, JH, Zhang, H, Zhang, GF, Sun, JG. Solid reconstruction using recognition of quadric surfaces from orthographic views. Computer-Aided Design 2006;38(8):821–835.

[15] Gong, JH, Zhang, H, Zhang, YW, Sun, JG. Converting hybrid wireframes to b-rep models. In: Proceedings of the 2007 ACM symposium on Solid and physical modeling. 2007, p. 283–289.

[16] Varley, PA, Company, PP. A new algorithm for finding faces in wireframes. Computer-Aided Design 2010;42(4):279–309.

[17] Zakharov, A, Zhiznyakov, A. Synthesis of three-dimensional models from drawings based on spectral graph theory. In: Applied Mechanics and Materials; vol. 756. Trans Tech Publ; 2015, p. 598–603.

[18] Çıçek, A, Gülesın, M. Reconstruction of 3d models from 2d orthographic views using solid extrusion and revolution. Journal of materials processing technology 2004;152(3):291–298.

[19] Lee, H, Han, S. Reconstruction of 3d interacting solids of revolution from 2d orthographic views. Computer-Aided Design 2005;37(13):1388–1398.

[20] Weiss-Cohen, M. 3d reconstruction of solid models from engineering orthographic views using variational geometry and composite graphs. Computer-Aided Design and Applications 2007;4(1-4):159–167.

[21] Tanaka, M, Kaneeda, T. Feature extraction from sketches of objects. Computer-Aided Design and Applications 2015;12(3):300–309.

[22] Harish, AB, Prasad, AR. Automated 3d solid reconstruction from 2d cad using opencv. arXiv preprint 2021;:2101.04248.

[23] Lambourne, JG, Willis, KD, Jayaraman, PK, Sanghi, A, Meltzer, P, Shayani, H. Brepnet: A topological message passing system for solid models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, p. 12773–12782.

[24] Wang, K, Zheng, J, Zhou, Z. Neural face identification in a 2d wireframe projection of a manifold object. In: Proceedings of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition. 2022, p. 1622–1631.

[25] Li, C, Pan, H, Bousseau, A, Mitra, NJ. Free2cad: parsing freehand drawings into cad commands. ACM TOG 2022;41(4):1–16.

[26] Wu, R, Xiao, C, Zheng, C. Deepcad: A deep generative network for computer-aided design models. In: Proceedings of the IEEE/CVF Int. Conf. on Computer Vision (ICCV). 2021, p. 6772–6782.

[27] Jayaraman, PK, Lambourne, JG, Desai, N, Willis, KD, Sanghi, A, Morris, NJ. Solidgen: An autoregressive model for direct b-rep synthesis. arXiv preprint 2022;:2203.13944.

[28] Willis, KD, Jayaraman, PK, Lambourne, JG, Chu, H, Pu, Y. Engineering sketch generation for computer-aided design. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, p. 2105–2114.

[29] Li, C, Pan, H, Bousseau, A, Mitra, NJ. Sketch2cad: Sequential cad modeling by sketching in context. ACM Transactions on Graphics (TOG) 2020;39(6):1–14.