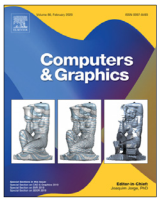
Computers & Graphics 102 (2022) 349–359

Contents lists available at ScienceDirect

Computers & Graphics

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



Special Section on SMI 2021

Extracting datums to reconstruct CSG models from 2D engineering sketches of polyhedral shapes

Raquel Plumed [a,*], Peter A.C. Varley [b], Pedro Company [a], Ralph Martin [c]

[a] *Department of Mechanical Engineering and Construction, Universitat Jaume I, Castellón, Spain*
[b] *Department of Systems and Control Engineering, University of Malta, Msida MSD 2080, Malta*
[c] *School of Computer Science, Cardiff University, Cardiff, UK*

ARTICLE INFO

Article history:
Received 30 June 2021
Received in revised form 18 October 2021
Accepted 22 October 2021
Available online 28 October 2021

Keywords:
Sketch-based modeling
CSG models
Features
Datums

ABSTRACT

Our goal is to automatically generate CAD 3D models from 2D sketches as part of a design chain where models should be procedural, containing *features* arranged in a model tree and linked to suitable *datums*. Current procedural models capture much about the design intent and are easy to edit, but must be created from scratch during the detailed design state—given conceptual sketches as used by designers in the early part of the design process, current sketch-based modeling approaches only output explicit models. Thus, we describe an approach to extract high-level information directly from 2D engineering wireframe sketches and use it to complete a CSG feature tree, which serves as a model tree for a procedural 3D CAD model.

Our method extracts procedural model information directly from 2D sketches in the form of a set of features, plus a set of datums and relationships between these features. We detect and analyze features of 2D sketches in isolation, and define the CSG feature tree by the parent–child relationships between features, and combine this information to obtain a complete and consistent CSG feature tree that can be transferred to a 3D modeler, which reconstructs the model. This paper focuses on how to extract the feature datums and the extrusion operation from an input 2D sketch.

1. Introduction

Engineers and designers typically start by sketching their ideas using pencil and paper, but current mechanical CAD applications (MCAD) are unable to use such sketches as a basis for producing 3D CAD models, which must be created from scratch during the detailed design stage. This produces a gap between the earliest stages of the design process and the rest of the product development process. Our purpose is to bridge this gap.

Our design tools must automatically extract the design intent embedded in a sketch, translate it into a CSG feature tree, and output it in some standard format, to automatically provide a 3D solid model that contains the digital product geometry [1]. This would simplify the design process—including the review of virtual prototypes [2]—by facilitating the creation of CAD models (by providing an alternative to convert design sketches into detailed CAD models), while maintaining the current advantages of exchange of CAD model data between different CAD systems, or between a CAD system and downstream application systems (CAE, CAM, etc.).

This problem is addressed by sketch-based modeling (SBM), which aims to automatically obtain 3D models from 2D sketches [3]. The main approaches applied in reconstruction modeling are optimization and inflation. Optimization has proved not to provide reliable results for SBM tasks, in view of its intrinsic numerical complexity and the possibility of being trapped in local minima. Inflation is a fairly reliable approach as long as the input is a high-quality image or drawing, but can be problematic in the presence of sketching errors. Still, the main drawback here is that these techniques provide boundary representation (B-Rep) models [4], which explicitly store low level geometric and topological information.

We propose a new strategy to extract high-level semantic information from conceptual 2D sketches before obtaining the 3D model, and construct a model tree using both procedural and explicit information. High-level semantic information captures the design intent of the model, and translating this information into procedural information allows building models in a more flexible way, so easing their subsequent re-editing or modification.

The main ingredients of the CSG strategy are: (a) modeling features and (b) datums.

* Corresponding author.
E-mail addresses: plumed@uji.es (R. Plumed), peter.varley@um.edu.mt (P.A.C. Varley), pcompany@uji.es (P. Company), Ralph.Martin@cs.cardiff.ac.uk (R. Martin).

https://doi.org/10.1016/j.cag.2021.10.013

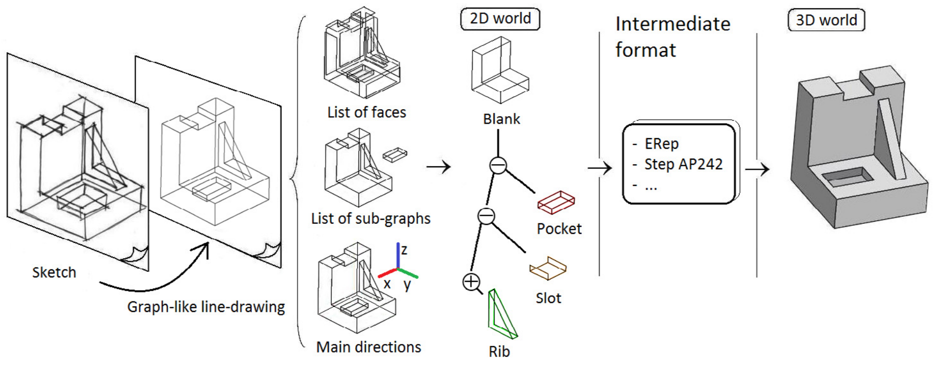


Fig. 1. Overall process to obtain a 3D model from an engineering sketch.

According to the ISO 5459:2011 standard [5], each reference element is a *Datum*, while the set of datums that define a reference system is a *Datum System*. A datum system acts as a reference system, allowing unequivocal definition of the location of any object or feature in the drawing or in space.

A *datum* is a geometrical element, which may or may not belong to the model itself. For example, the common uses of a datum plane are as a working plane or as a mirror plane. Faces of a model may also act as "on the fly" datum planes. *Contact faces* are those datum planes that represent planar surfaces belonging to a parent feature face, which act as supporting surfaces for a child feature. A *skewed symmetry plane* is the result of the affine transformation of a bilateral symmetry plane (or mirror plane) into a parallel projection.

Fig. 1 summarizes our general approach. Our input data are strokes, and our final goal is a 3D model. Sketch segmentation and vectorization provide a graph-like line-drawing, where nodes depict the vertices of the sketch and the lines linking the nodes depict the edges of the sketch. Low level graph recognition provides cues related to shape, such as subgraphs, faces, main directions, etc. All of these provide the input from which feature recognition is carried out. Then the features are linked to each other by datums and ordered in a hierarchy based on their mutual parent–child relationships (represented as a top down tree in Fig. 1). The last step consists of translating the information to a commercial CAD modeler to automatically obtain the 3D model.

We build on the work by Plumed et al. to determine the features embedded in a sketch [6]. The present work addresses the problem of completing the model tree data. It determines the cross-sections used in sweep operations to construct the features, and defines their related reference system. This information has to be represented in a way that CAD modelers can later understand.

To validate our proof of concept, we limit our polyhedral objects to solely contain features that represent prismatic solids, i.e. the result of linear sweeps (extrusions) of unchanging shapes (constant cross-section from end to end). The position of the features is as well limited to having their faces parallel to one of the orthogonal planes of the general reference system. Only basic datums are used in the model, which requires only the default front, top and right planes of a general orthogonal reference system; we do not yet detect complex datums or datums not directly linked to sweep operations.

The paper is organized as follows. In Section 2 we discuss related work. Section 3 introduces critical considerations on datums. Section 4 explains how we determine reference datums and profile data, via use of an example. Section 5 validates the approach by analyzing some examples. Section 6 discusses the strengths and limitations of the approach and summarizes our conclusions.

2. Related work

2.1. Overview

The input required by our approach is a line-drawing: a list of junctions and a list of lines, where each line connects two junctions. The conversion of sketches into graph-like line-drawings, which is typically the first stage in sketch-based modeling approaches, is not detailed here—we assume that a vectorized 2D line-drawing is already available. A summary of the state of the art in this topic to 2019 can be found in the introduction of [7]. More recently, Wang et al. propose a sketch recognition system based on the multistroke primitive grouping method to obtain a line-drawing [8,9].

Hitherto there has been much work on obtaining an explicit model from an engineering sketch, but work aimed at obtaining a procedural model from an engineering 2D sketch is more limited.

As noted in [4], 3D B-Rep models are the typical output of SBM approaches. However, reconstruction of constructive solid geometry (CSG) models is regaining interest due to recent advances in personal additive manufacturing in the form of 3D printing, and computer-aided manufacturing systems for mass customization [10]. Producing 3D digital models from sketches may shorten and simplify the CAD/CAM process, easing modifications to customizable design features and allowing non-expert end-users to produce their own designs.

There are few attempts to obtain CSG models from a *single axonometric view*. Wang and Grinstein [11] produce a CSG representation in which each feature is represented by a cuboid. Branco et al. [12] presented the IDEG system, which uses WIMP (windows, icons, menus and pointer) interaction together with perspective sketch input. Suh [13] noted that many mechanical components can be represented by a combination of linear swept volumes (LSV). Nevertheless, these pioneering approaches, most of which require user interaction and are limited to simple form features, have not been followed up.

Other approaches consider multiple orthographic views: Shum et al. [14] proposed a two-stage approach to reconstruct extruded solids. Lee et al. [15] used a hint-based approach to recognize solids of revolution from orthographic views. Other methods applied interactive reconstruction and CSG operations when interpreting sketches, e.g. GIDeS, a gestured-based system by Pereira et al. [16]; Shesh et al. [17] used CSG operations in their sketch system (SMARTPAPER) in which user feedback modifies the sketch input. Such work uses different input data from our approach (multiple orthographic views instead of a single axonometric view), and in some cases requires user interaction; furthermore, it only detects form features (i.e. those which do not convey design or manufacturing information).

Schmidt et al. [18] present an interesting strategy to create automated models with free forms, applying constraints defined by means of 3D scaffolds. The main differences with our approach is that a) the method did not search high-level semantic information like recognition of design intent and b) this approach is guided by the user as the sketch progresses, while in our case the process is off-line, it stars once the sketch is finished.

The process of detecting features in a 2D sketch by seeking indirect cues was studied by Company et al. [19–21] (at least for the case where features can be created as a result of one particular type of sweep: extrusions). An approach to determine the relationships between features to build a model tree was presented by Plumed et al. [6]. Tanaka et al. have provided further variations and improvements on this approach [22–25].

The process of CSG feature tree construction used by Plumed et al. (which searches for features in a 2D drawing) is similar to the processes of feature simplification suggested in [26–28] (which work on 3D models).

Cheon et al. [29] also proposed an approach to provide editable 3D data from a free-hand 2D sketch to a 3D CAD system. Our final goal is very close to theirs, but approaches can greatly differ depending on the nature of the input data, as they use gestural modeling—where the main problem involves differentiating stroke commands from shape strokes—whereas we work off-line with no user-interaction. This means we lack not only certain data, which may help to disambiguate certain extracted information, but also the construction history conceived by the designer.

Some recent work in the field [30] vectorizes the construction lines of a sketch to convert them to 3D, and then fits the sketched object into the 3D framework generated from the construction lines. Kato et al. [31] present a method for 3D reconstructing models outside the scope of engineering parts. Their approach outputs a 3D origami model with a plausible outer shape from an image of a flat origami piece.

Neural networks are currently a very popular research direction. One of the most effective tools found for the task for image recognition is *Convolutional Neural Network* (CNN). CNNs are frequently used for image classification and recognition because of their high accuracy, but ours is not a classification or recognition problem. In recent work [32] a CNN strategy is applied to interpret shape sketches as parametric modeling operations, incrementally creating the 3D model. An extensive review can be found in [33]. Tools which convert one 2D image to another 2D image (e.g."pix2pix" [34] can in principle be used to add labels to images (colors are labels), but no current work uses them explicitly for image analysis. Advanced 3D reconstruction neural networks have achieved impressive results in recent times, but most of them still suffer from training difficulties and loss of details, due to their weak ability to extract features [35]. The output is usually voxelized 3D models, reconstructed by layers—they are not able to recognize high-level information of features from 2D engineering wireframe sketches.

Finally, we note that recognition of features from 3D input is a well-established problem in which significant advances have been made [36], but this does not directly apply here since recognition requires 3D information which is not available in SBM.

Once we analyze the sketch to get the CSG feature tree of the model, we can obtain the reconstructed 3D model by inputting it into a 3D CAD modeler, instead of applying the usual 3D reconstruction stages of sketch-based modeling approaches. This data transfer can be represented by explicit or procedural techniques. Each has its own advantages. Explicit models provide quick access to detailed geometric information that is important for downstream applications [37]. Procedural modeling techniques such as history-based parametric feature-based modeling create 3D models from sets of rules [38], with the advantages of capturing all or part of the design intent and being easy to edit [39]. Procedural approaches which create 3D CAD models include feature-based design (FBD) and constraint-based modeling [40,41]. In this paper we only focus on obtaining a coherent and complete CSG feature tree that preserves the design intent.

We note here that most current CSG model files use proprietary formats, since standard formats for procedural models have been fully defined only recently. STEP 242, first published in 2014, is the choice for CAD import/export; the second edition ISO 10303-242:2020 [42] includes new geometric capabilities and additive manufacturing setup information.

2.2. Obtaining the CSG feature tree

Two similar approaches which obtain a CSG feature tree, Suh [13], and Plumed et al. [6], propose building a feature extrusion tree from a 2D sketch. Suh's method implicitly generates a procedure while the 3D model is being created. Plumed's approach relies on design features and considers the child–parent relationship between features. The input is a 2D graph-like line drawing, obtained by vectorizing a sketch that represents a single 2D wireframe view of a polyhedral object. The strategy used to build the CSG feature tree (Fig. 2) is to apply the reverse design order from child to parent: features more likely to be children are first detected, added to the tree, removed from the drawing, and then their parents searched for. The process continues recursively until a top level shape is reached. This top level shape is the primitive shape depicted in the simplified sketch which cannot be further subdivided into features. It becomes the root node of the tree.

Our novel ideas to complete the CSG feature tree are the stages shown as shaded in Fig. 2. We elaborate them in Section 4.

In absence of depth information, the approach relies on information provided by indirect cues, those properties of a 2D drawing that reveal properties of the 3D object that it represents. As for example, the main directions of the axonometric view, obtained as explained in Kang et al. [43]; a list of detected and numbered faces [44]; information of subgraphs and edges labeled following Varley's extended approach [45]. Other useful information can be extracted from the 2D drawing, before the recognition feature stage, such as bilateral symmetry planes for the model using the approaches of Plumed et al. [46] or Varley et al. [47] and the perimeters of subgraphs [48].

At this point, the approach is able to create a CSG tree that includes the following information about features:

- The feature's place in a hierarchy of features, based on the reverse order of feature extraction. Features extracted earlier need to be nearer the leaves of the tree.
- Parent–child relationships, created when one feature is used to construct another: for example, when a face of a feature is selected as the working plane for a second feature. This second feature is considered a child of the feature containing the working plane.

To facilitate subsequent data exchange with CAD modelers, we need to complete the CSG tree with information that focuses on the reconstruction of 3D features, in particular:

- The feature's location referenced by datums.
- Explicit information about feature geometry. This includes detailed information about its profile and the extrusion operation (length and direction).

This information completes the CSG feature tree and constitutes the novelty of our current work.

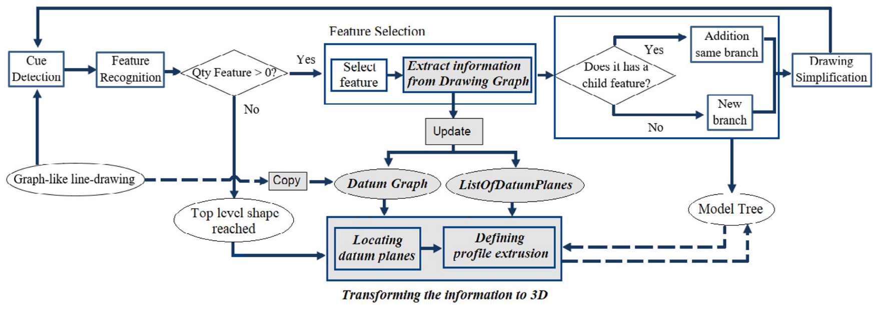


Fig. 2. Overall approach, including the new stages (with shaded area) to complete the CSG feature tree information.

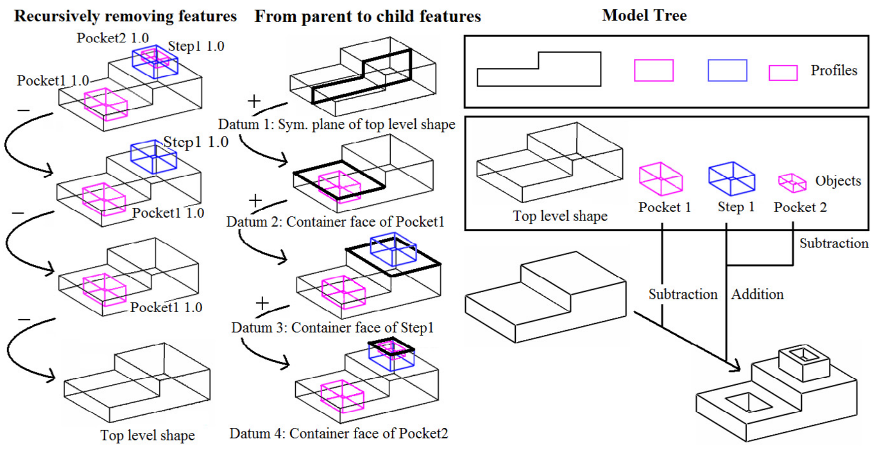

Recursively removing features
From parent to child features
Model Tree
Pocket2 1.0
Step1 1.0
Pocket1 1.0
Datum 1: Sym. plane of top level shape
Datum 2: Container face of Pocket1
Datum 3: Container face of Step1
Datum 4: Container face of Pocket2
Top level shape
Profiles
Objects
Pocket 1
Step 1
Pocket 2
Subtraction
Addition

Fig. 3. Information included in the model tree (left-feature hierarchy, center-feature datums, right-geometric extrusion profiles).

3. Previous considerations

3.1. Reference systems

To complete the CSG feature tree we have to include information about the position that each feature occupies in the model and its extrusion information (the "feature selection" stage of the general process described in Fig. 2). Fig. 3 graphically explains the strategy we use to build the CSG feature tree. Fig. 3 (left) shows the simplification process, applying the reverse design order from child to parent, explained in the previous section—in each iteration the outcome of the recognition process is a list of candidate features with a figure of merit in the range [0, 1]. Here, we address the definition of datums (Fig. 3 center), to locate the features and the information of the geometric extrusion profiles (Fig. 3 right) needed to complete the model tree.

Auxiliary reference systems must be defined for each feature. Each reference system links the feature with the rest of the 3D model. When we define auxiliary reference systems we must take into account two main considerations: (a) that cluttering the model with additional references could be detrimental or confusing—when creating auxiliary systems, it is good practice to minimize the number of links between systems, so we create an explicit reference system for each feature and, whenever possible, link it to the main reference system of the top level shape; (b) the reconstruction stage involves working with 3D objects, so our reference systems must be 3D.

A datum plane can be created at any point in the modeling process. When the user creates a datum plane explicitly, CAD tools show the datum plane as a feature on the model tree. The main advantage of explicit datums is that they can be used as a basis for future operations. Conversely, implicit datums belong to the feature undergoing creation, and they are not available for use by other features. When the user selects an implicit datum, the datum does not show on the model tree and becomes hidden, as a private property of the swept profile, after the feature is created.

Once the user defines an implicit or explicit plane, the CAD system implicitly determines the rest of the elements necessary to define an auxiliary reference system, and computes the transformation matrices with other reference systems. This strategy not only saves time, but also automatically links the new profile to the current model. Current commercial CAD modelers store datums defined by the user either on the fly (in this case, the storage is internal), or as explicitly defined references (and then the reference also appears in the model tree).

Our challenge is to define all these necessary 3D reference systems from a 2D line drawing that represents an axonometric projection of the 3D model, and define their relationships to reconstruct the model later. As we work off-line with no user-interaction, we lack previous notable information like the construction history conceived by the designer, so all our reference systems are defined by explicit plane datums. Thus, we first define the pattern reference systems for each type of feature that we can recognize in a 2D sketch.

3.2. Patterns of datum planes for each feature

We define a 3D reference system as the intersection of three orthogonal planes. Thus, for each feature of the 2D line drawing, we determine the 2D projections which best represent three mutually-orthogonal datum planes. The intersection of these three datum planes defines the origin of the auxiliary reference system of the feature and it will also act as the insertion point of the feature in the 3D model.

To date, the catalog of design features our approach can recognize in an engineering drawing is limited: isolated steps and

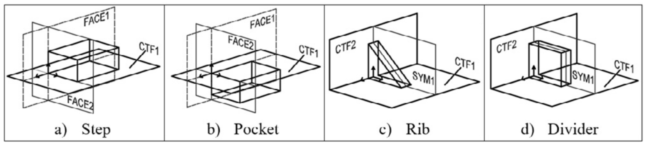


Fig. 4. Patterns of auxiliary reference systems and cross-sections for steps, pockets, ribs and dividers.

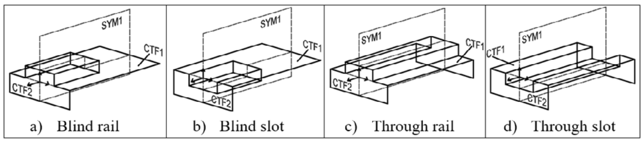


Fig. 5. Patterns of candidate auxiliary reference systems for rails and slots.

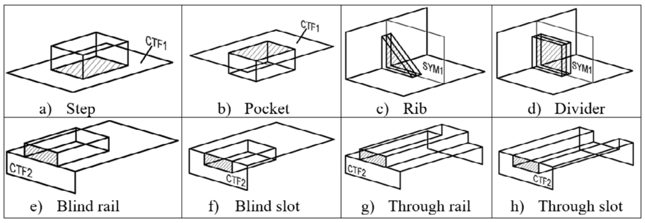


Fig. 6. Cross-sections for each type of feature.

pockets, ribs, dividers, slots and rails. All of these features are based on extruded operations of polyhedral shapes. In addition, the position of the features is limited in that the faces of any feature are parallel to one of the orthogonal planes of the general reference system (except the border face of ribs). Thus all datum planes are parallel to one of the default front, top and right planes of the main orthogonal reference system.

Figs. 4a and 4b show the default auxiliary reference systems defined for isolated steps and pockets as the intersection among a *contact face* (CTF1) and two faces of the corresponding feature (FACE1–FACE2). The origin of the reference system is chosen so that the node belongs to the face that is coplanar to the contact face and at the same time is the closest node to the origin of the main subgraph (the one related to the top level shape that contains the nodes and lines of the top level shape).

In the case of ribs and dividers (Figs. 4c and 4d), the default auxiliary reference system is determined by the intersection between two contact faces (CTF1–CTF2) and the skewed symmetry plane of the feature. The origin of the reference system is obtained as the node of the neutral plane of the rib (skewed symmetry plane in the drawing graph) that belongs to both container faces.

Rails and slots are similar (Fig. 5), regardless of whether they are through or blind. The set of datum planes consists of two contact faces (CTF1–CTF2) and one skewed symmetry plane of the features (SYM1). The origin is the node of the symmetry plane that belongs to the intersecting edge between the two container faces. This edge will form part of the cross-section of the feature and will be added to the drawing graph after the simplification of the feature. In the case of through rails and slots, in addition, the origin will be the closest node to the origin of the main subgraph.

One of the datum planes will contain the cross-section of the feature that will then be extruded. Fig. 6 shows the cross-sections according to the type of feature.

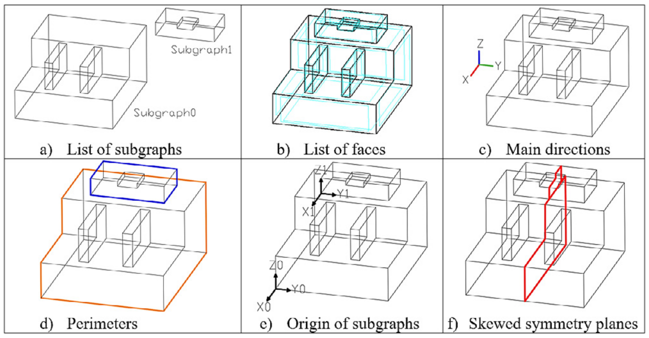


Fig. 7. Indirect cues information.

4. Obtaining 3D datums from 2D sketches

4.1. Preparatory preprocessing

The input to our overall process is a *Drawing Graph* in which nodes depict vertices of the sketch and the lines linking the nodes depict edges of the sketch. This Drawing Graph is the output of the first step of the sketch-based modeling process, the vectorization stage (illustrated in Fig. 1). After vectorization, a process of *beautification* converts all the edges of the graph that are aligned in the same direction into parallel lines [49]. Our approach is still tolerant of small imperfections that sometimes remain after beautification. We use thresholds that can be adjusted depending on the quality of the sketching the examples presented in this work a threshold of 10° is used to recognize main alignments.

In the "Cue detection" stage 2, low level graph recognition provides indirect cues related to shape, such as list of subgraphs, list of faces, main alignments, perimeter, origin node of each subgraph and skewed symmetry planes. The origin node of each subgraph is the leftmost and bottom-most node of the subgraph, and it must also belong to the perimeter circuit. Indirect cues provide the input from which "feature recognition" is carried out. All this information will become fundamental input data from which we extract the set of datums that will define the auxiliary reference systems for each feature. Fig. 7 shows an example of indirect cue detection in an L-shaped prism with two dividers and a single step. The step includes a slot in its upper face.

To build the CSG feature tree, a recursive strategy is applied following the reverse design order from child to parent, with each iteration selecting a feature and removing its lines and nodes from the Drawing Graph (the already known strategy of building a model tree).

Henceforth, our approach must feed the CSG feature tree by including information of extrusion operations and linking an auxiliary reference system for the extruded feature before the simplification process is carried out. This process takes place during the "Feature selection" stage (Fig. 2). This information must remain available outside the simplification process, so first (before the simplification process starts) a copy of the original graph is created (henceforth *Datum Graph*), so that at the same time as the Drawing Graph is simplified in each iteration, the *Datum Graph* is enriched with datum information. The datum planes are defined by closed circuits in the *Datum Graph*.

Alongside the Datum Graph, a list of datum planes is created. Each element stores the information of a datum plane including:

- The plane orientation according to the main directions of the drawing.
- The subgraph to which the feature belongs.
- List of nodes of the plane. A vector where each node is stored by its 2D coordinates.

- List of edges of the plane. A vector where the edge is stored by its endpoints (should coincide with consecutive nodes of the datum plane).
- A binary parameter to define whether the plane is a symmetry plane or not (Y/N).
- Location from the origin of the main subgraph.

Before starting the iteration process, the Drawing Graph is copied as a Datum Graph and then the skewed symmetry planes are included in the Datum Graph by adding their nodes and edges. This means that the original edges are divided when new nodes are introduced into the Datum Graph. Furthermore, these symmetry planes are also added to the list of datum planes.

We assume that the geometrical imperfections inherent in freehand sketching have been removed by the beautification stage. Hence, the coordinates of any point represented in the line drawing refer to a 2D orthogonal reference system located in the screen, which represents the projection plane. In the current stage, the data stored for nodes uses this 2D orthogonal reference system located in the screen. In the next stage, this information will be translated to 3D.

The direction of viewing in the orthogonal axonometric projection causes a distortion in the dimensions of the drawing due to foreshortening. Different scales along each of the three axes reflect this distortion. These scales are calculated from the angles of the main directions projected onto the sketching plane. In our case, we apply the following expression derived from [50]:

$$C_x = \sqrt{\frac{cos\hat{YZ}}{cos\hat{YZ} - cos\hat{XZ} \cdot cos\hat{XY}}} \tag{1}$$

$$C_y = \sqrt{\frac{cos\hat{XZ}}{cos\hat{XZ} - cos\hat{YZ} \cdot cos\hat{XY}}} \tag{2}$$

$$C_z = \sqrt{\frac{cos\hat{XY}}{cos\hat{XY} - cos\hat{YZ} \cdot cos\hat{XZ}}} \tag{3}$$

In each iteration, our algorithm considers the pattern of the reference system related to the selected feature and searches for a triplet of datum planes that fulfills the requirements of the type of feature and the cross-section defined by its nodes in the Drawing Graph. This information is obtained in two stages explained in Sections 4.2 and 4.3.

4.2. Extracting information from the 2D graph-like line drawing

The general process to extract information from 2D auxiliary reference systems is shown in Fig. 8. It takes place during the 'Feature selection'' process (Fig. 2) and consists of storing data from the Drawing graph of the cross-section and the closed circuits that will later be translated into 3D coordinates.

Once the feature is selected and in order to obtain extrusion information, the base faces of the feature are retrieved from the Drawing Graph. These faces remain constant throughout the extrusion and are parallel to the cross-section and perpendicular to the extrusion direction. From the base faces, the extrusion length is calculated as the average distance between paired nodes of the base faces multiplied by the scale corresponding to the extrusion direction.

Step and pocket features imply a subgraph disconnected from the container graph. In order to later calculate locations between nodes, all the subgraphs must be connected in the Datum Graph. The node acting as the origin of the triplet of datum planes in the step/pocket will be one of the end points of an auxiliary line that will act as a bridge between the two graphs. This auxiliary line must be the shortest line parallel to one of the main directions.

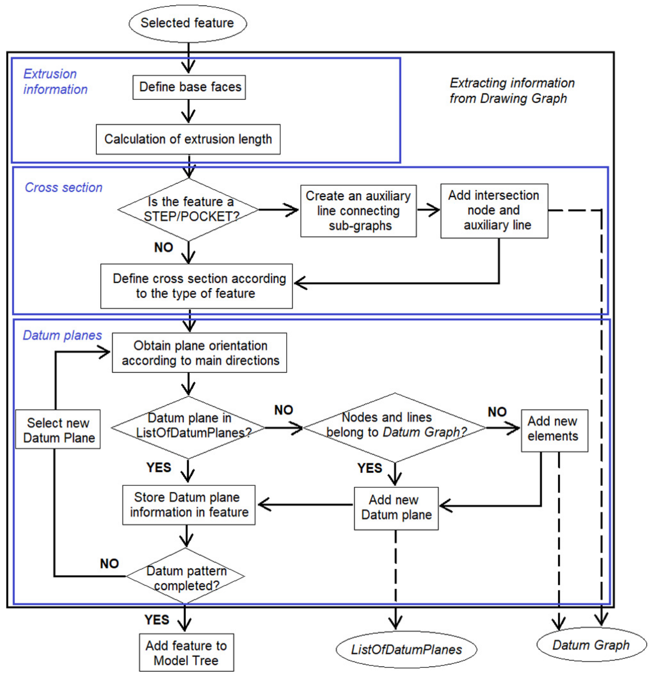

Selected feature
Extrusion information
Define base faces
Calculation of extrusion length
Extracting information from Drawing Graph
Cross section
Is the feature a STEP/POCKET?
Create an auxiliary line connecting sub-graphs
Add intersection node and auxiliary line
NO
Define cross section according to the type of feature
Datum planes
Obtain plane orientation according to main directions
Select new Datum Plane
Datum plane in ListOfDatumPlanes?
NO
Nodes and lines belong to Datum Graph?
NO
Add new elements
YES
YES
Store Datum plane information in feature
Add new Datum plane
NO
Datum pattern completed?
YES
Add feature to Model Tree
ListOfDatumPlanes
Datum Graph

Fig. 8. Flowchart of the algorithm to extract information from the 2D Drawing Graph.

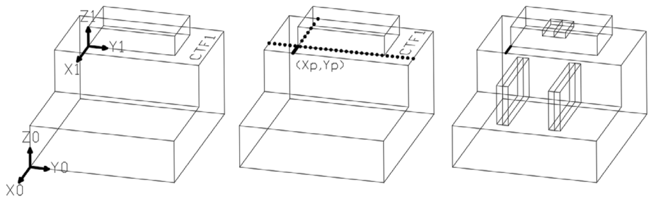


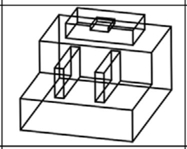
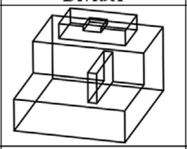
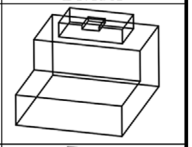
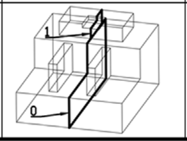
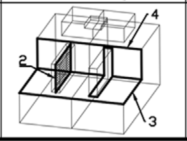
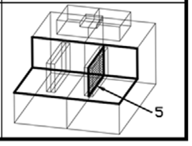
Fig. 9. Drawing Graph with the origins of subgraphs (left), obtaining auxiliary line that connects the subgraphs (center), Datum Graph including the auxiliary line (right).

Fig. 9 shows an example with a step feature (fourth iteration of the Drawing Graph in the example of Fig. 10), the origins of each subgraph are represented on the left, the auxiliary line that connects the two subgraphs is shown on the center (the rejected auxiliary lines are also shown as dotted lines. This auxiliary line intersects one of the edges of the container face and the new node (xp, yp) divides the edge of the container face. The image on the right depicts the Datum Graph in the same iteration including the auxiliary line between subgraphs.

The first datum plane defined is the sketching plane, which contains the cross-section of the feature. After obtaining the plane orientation, if the new datum plane is not already included in the list of datum planes, then it is added and the new nodes and edges are also included in the Datum Graph.

The other two datum planes are chosen to define the pattern of datum planes for the feature. Starting from the origin of the triplet, the corresponding three planes are retrieved from the Drawing Graph. The two planes with different orientation from the cross-section plane are considered as datum planes. New nodes and edges are included in the Datum Graph and the datum plane is added to the list of datum planes if it was not previously included.

Continuing with the example previously used, Fig. 10 shows the iterative simplification process of the Drawing Graph until the top level shape is reached. The Datum Graph is enriched in

Iteration	0	1	2
Feature		Divider	Divider
Drawing Graph			
Datum Graph			

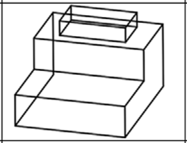
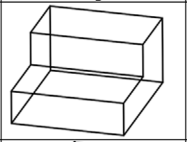
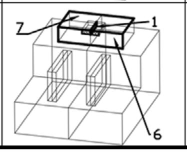
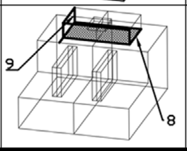
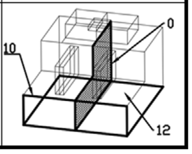
Iteration	3	4	5
Feature	Slot	Step	Top level shape
Drawing Graph			
Datum Graph			

Fig. 10. Example of the simplification process in the Drawing Graph and insertion of nodes and lines in the Datum Graph.

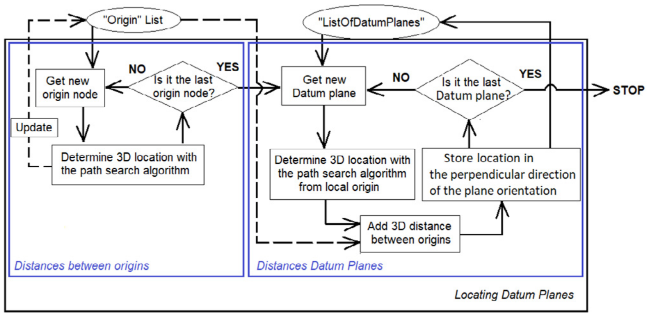

"Origin" List
"ListOfDatumPlanes"
Get new origin node
NO
Is it the last origin node?
YES
Update
Determine 3D location with the path search algorithm
Distances between origins
Get new Datum plane
NO
Is it the last Datum plane?
YES
STOP
Determine 3D location with the path search algorithm from local origin
Store location in the perpendicular direction of the plane orientation
Add 3D distance between origins
Distances Datum Planes
Locating Datum Planes

Fig. 11. Flowchart of the algorithm that calculates the position of Datum planes.

each iteration with new nodes and lines that depict new datum planes. Datum planes are numbered according to the order they are introduced in the list of datum planes.

Fig. 10 follows Plumed's method. Suh's method would start with Fig. 10(4), an extrusion that contains all of the longest lines in the drawing, it would then add another extrusion to get Fig. 10(3), then the two dividers, and finally the slot, to get the drawing in Fig. 10(0). Topologically, the results are the same, but Plumed's method gives branches of the tree in an order which better reflects the parent–child relationships of the features.

4.3. Transforming the information to 3D

Once the top level shape is reached—when the simplification process is finished (Fig. 2)—the Drawing Graph is empty and the Datum Graph contains all the information needed to move to the next stage.

In this stage, the approach converts the information from the Datum Graph into 3D information and stores the information necessary to locate and extrude each feature of the model tree. The input information is supplemented by the information from the model tree and list of datum planes.

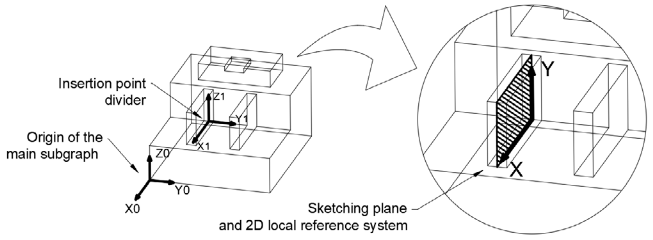

Insertion point divider
Origin of the main subgraph
Z1
Y1
X1
Z0
Y0
X0
Y
X
Sketching plane and 2D local reference system

Fig. 12. Different reference systems to locate an define the extrusion profile of the feature.

The algorithm searches for axis-aligned planes but unlike other methods such as [51], which work after inflation to $2\frac{1}{2}$D, our method is based on a path search algorithm where the inputs are two nodes of a graph representing a line drawing that only contains 2D information about a 3D model. The algorithm determines locations, measuring distances between contiguous elements and obtaining their absolute positions by combining relative locations. This simple algorithm works correctly as long as all the lines of the graph are connected. In the future, we will reduce errors by adding redundant distance calculations to reduce the impact of errors on the "beautification" of the vector drawing.

Starting from the origin of the main subgraph, the algorithm iteratively picks an unvisited node connected to the last visited node. Of the possible nodes, the closest to the destination node is selected. Then it calculates its location by adding the scaled distance along the connecting line to the appropriate coordinate (x, y or z) of the visited node. The algorithm terminates when the destination node is reached.

The algorithm is used in the following two sub-stages of the approach: (a) defining the location of each Datum plane (from the list of datum planes) with respect to the origin of the main subgraph (in this case, the algorithm starts at the origin of the main subgraph and finishes when the path reaches one of the nodes belonging to the datum plane under consideration; the searched location of the Datum Plane is the one stored in the perpendicular direction to the plane orientation) and (b) obtaining the extrusion profile of each feature from the previously stored cross-section (the profile is determined by the orientation of the sketching plane and the insertion vertex defined by the triplet of datum planes).

4.3.1. Locating datum planes with respect to the main origin

The nodes of the origins of the subgraphs previously defined as a cue (Fig. 7e) were stored in the list of origin nodes. Now, we calculate the locations from each origin to the origin of the main subgraph.

Hence, we calculate the location from the origin to each entry in the list of datum planes. The information is stored in each element (datum plane) of the list of datum planes.

The general process to locate datum planes to the main origin is shown in Fig. 11.

4.3.2. Defining the extrusion profile

Finally, the dimension of the extrusion profile can be calculated for each feature from the stored cross-section defined by the 2D coordinates of the nodes of the drawing graph. This process implies several steps.

First, we transform the 2D cross-section nodes into 3D orthogonal vertices—each node of the cross-section is replaced by a 3D vertex (x, y, z) whose coordinates store the position of the vertex with respect to the origin of the main subgraph (Fig. 12). Next, the vertices of the profile are moved to the insertion point of the

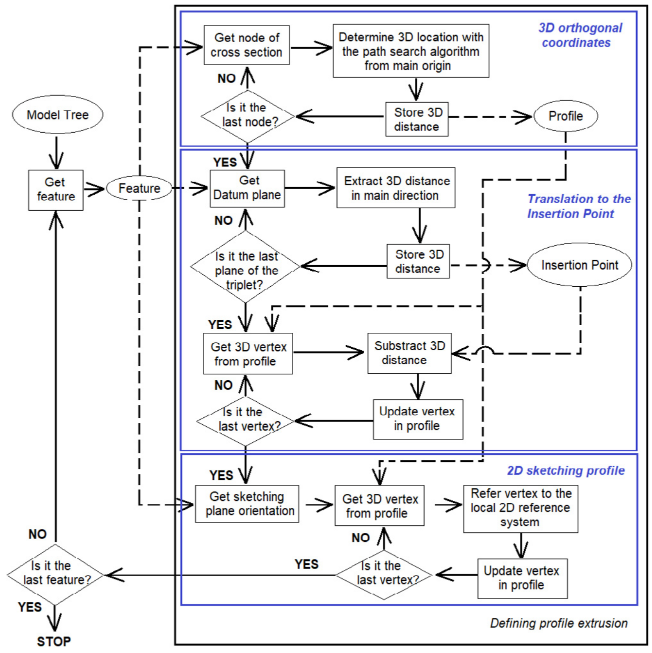

3D orthogonal coordinates
Get node of cross section
Determine 3D location with the path search algorithm from main origin
NO
Is it the last node?
Store 3D distance
Profile
Model Tree
YES
Get feature
Feature
Get Datum plane
Extract 3D distance in main direction
Translation to the Insertion Point
NO
Is it the last plane of the triplet?
Store 3D distance
Insertion Point
YES
Get 3D vertex from profile
Substract 3D distance
NO
Is it the last vertex?
Update vertex in profile
YES
2D sketching profile
Get sketching plane orientation
Get 3D vertex from profile
Refer vertex to the local 2D reference system
NO
NO
Is it the last feature?
YES
Is it the last vertex?
Update vertex in profile
YES
STOP
Defining profile extrusion

Fig. 13. Flowchart of the algorithm to define the extrusion profile of each feature.

feature and the location of the insertion point is subtracted from each vertex of the profile. Finally, the orthogonal 3D vertices of the profile are translated into a 2D local system in the sketching plane. The origin of the 2D local system coincides with the insertion point and the rules applied to define directions (x, y) are:

- If possible, the x-axis of the sketching plane remains parallel to the main x direction.
- Otherwise, the y-axis of the sketching plane remains parallel to the main z direction.

The general process to define the profile extrusion is shown in Fig. 13.

4.4. Output

The output shown here serves to proof the concept that our method can build a procedural model. It consists of a text list of features ordered from root to leaves. Each feature is defined by its extrusion information and the triplet of datum planes that locate the feature in the main reference system. This information can be used by a designer to create the 3D model using any commercial modeler. Additional information, such as the subgraph and the branch of the tree they belong to, helps convey parent–child links between features.

Fig. 14 shows this output for the example used above (Figs. 7 and 10). The root of the model tree is the top level shape, which the algorithm calls BLANK. The step and the slot belong to branch 1 and the order of the features defines that the step, which appears closer to the blank than the slot, is the parent feature of the slot and not vice versa. Dividers depict the leaves of the model tree. In engineering practice, dividers and ribs are considered features whose function is to increase the strength and rigidity of the part, and for this reason they are usually added at the end of the design process. The ID of each datum plane is a label that represents the position of the plane in the list of datum planes. The labels also help to determine whether features share datum planes. For example, both dividers have in common the planes ID 3 and ID 4, which depict the contact faces that support the dividers. The list of datum planes for each feature define the triplet of planes of its local reference system, which at the same time, define the insert point position for the extrusion profile.

Fig. 15 shows a model created with AutoCAD 3D using the output information in Fig. 14.

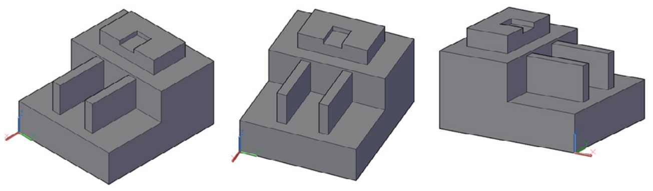
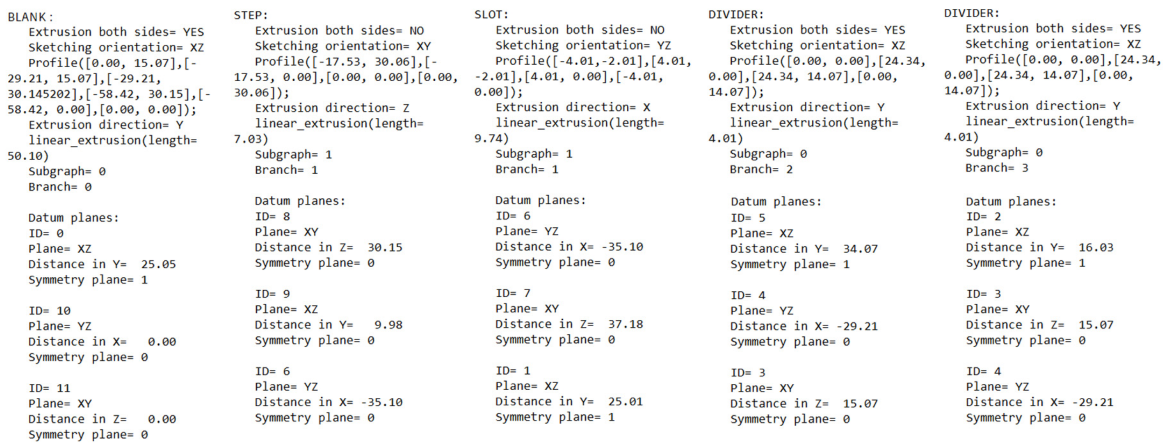


Fig. 15. Different views of the model obtained from the output.

5. Examples

As noted above, the most complicated case when completing the model tree data occurs with features that introduce disconnected subgraphs (steps and pockets). Here, we test an additional example in order to analyze whether the algorithm works properly with several nested disconnected subgraphs (Example 5.1). Next, a battery of examples is run with the algorithm and the outputs are shown in Table 1.



```
BLANK:
   Extrusion both sides= YES
   Sketching orientation= XZ
   Profile([0.00, 15.07],[-
29.21, 15.07],[-29.21,
30.145202],[-58.42, 30.15],[-
58.42, 0.00],[0.00, 0.00]);
   Extrusion direction= Y
   linear_extrusion(length=
50.10)
   Subgraph= 0
   Branch= 0

   Datum planes:
   ID= 0
   Plane= XZ
   Distance in Y=  25.05
   Symmetry plane= 1

   ID= 10
   Plane= YZ
   Distance in X=   0.00
   Symmetry plane= 0

   ID= 11
   Plane= XY
   Distance in Z=   0.00
   Symmetry plane= 0
```

```
STEP:
   Extrusion both sides= NO
   Sketching orientation= XY
   Profile([-17.53, 30.06],[-
17.53, 0.00],[0.00, 0.00],[0.00,
30.06]);
   Extrusion direction= Z
   linear_extrusion(length=
7.03)
   Subgraph= 1
   Branch= 1

   Datum planes:
   ID= 8
   Plane= XY
   Distance in Z=  30.15
   Symmetry plane= 0

   ID= 9
   Plane= XZ
   Distance in Y=   9.98
   Symmetry plane= 0

   ID= 6
   Plane= YZ
   Distance in X= -35.10
   Symmetry plane= 0
```

```
SLOT:
   Extrusion both sides= NO
   Sketching orientation= YZ
   Profile([-4.01,-2.01],[4.01,
-2.01],[4.01, 0.00],[-4.01,
0.00]);
   Extrusion direction= X
   linear_extrusion(length=
9.74)
   Subgraph= 1
   Branch= 1

   Datum planes:
   ID= 6
   Plane= YZ
   Distance in X= -35.10
   Symmetry plane= 0

   ID= 7
   Plane= XY
   Distance in Z=  37.18
   Symmetry plane= 0

   ID= 1
   Plane= XZ
   Distance in Y=  25.01
   Symmetry plane= 1
```

```
DIVIDER:
   Extrusion both sides= YES
   Sketching orientation= XZ
   Profile([0.00, 0.00],[24.34,
0.00],[24.34, 14.07],[0.00,
14.07]);
   Extrusion direction= Y
   linear_extrusion(length=
4.01)
   Subgraph= 0
   Branch= 2

   Datum planes:
   ID= 5
   Plane= XZ
   Distance in Y=  34.07
   Symmetry plane= 1

   ID= 4
   Plane= YZ
   Distance in X= -29.21
   Symmetry plane= 0

   ID= 3
   Plane= XY
   Distance in Z=  15.07
   Symmetry plane= 0
```

```
DIVIDER:
   Extrusion both sides= YES
   Sketching orientation= XZ
   Profile([0.00, 0.00],[24.34,
0.00],[24.34, 14.07],[0.00,
14.07]);
   Extrusion direction= Y
   linear_extrusion(length=
4.01)
   Subgraph= 0
   Branch= 3

   Datum planes:
   ID= 2
   Plane= XZ
   Distance in Y=  16.03
   Symmetry plane= 1

   ID= 3
   Plane= XY
   Distance in Z=  15.07
   Symmetry plane= 0

   ID= 4
   Plane= YZ
   Distance in X= -29.21
   Symmetry plane= 0
```

Fig. 14. Output.

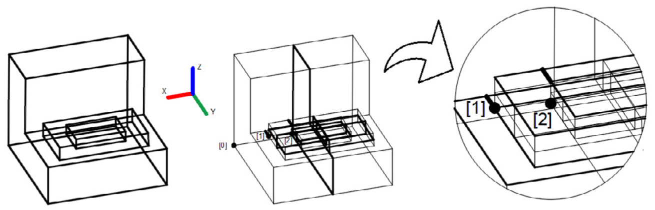
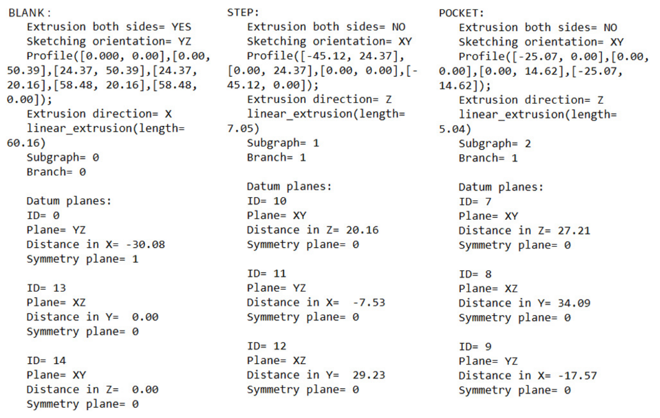


Fig. 16. Drawing Graph before simplification (left), Datum Graph after simplification (center), detail of the edge that connects the two disconnected graphs (right).



```
BLANK:
   Extrusion both sides= YES
   Sketching orientation= YZ
   Profile([0.000, 0.00],[0.00,
50.39],[24.37, 50.39],[24.37,
20.16],[58.48, 20.16],[58.48,
0.00]);
   Extrusion direction= X
   linear_extrusion(length=
60.16)
   Subgraph= 0
   Branch= 0

   Datum planes:
   ID= 0
   Plane= YZ
   Distance in X= -30.08
   Symmetry plane= 1

   ID= 13
   Plane= XZ
   Distance in Y=  0.00
   Symmetry plane= 0

   ID= 14
   Plane= XY
   Distance in Z=  0.00
   Symmetry plane= 0
```

```
STEP:
   Extrusion both sides= NO
   Sketching orientation= XY
   Profile([-45.12, 24.37],
[0.00, 24.37],[0.00, 0.00],[-
45.12, 0.00]);
   Extrusion direction= Z
   linear_extrusion(length=
7.05)
   Subgraph= 1
   Branch= 1

   Datum planes:
   ID= 10
   Plane= XY
   Distance in Z= 20.16
   Symmetry plane= 0

   ID= 11
   Plane= YZ
   Distance in X=  -7.53
   Symmetry plane= 0

   ID= 12
   Plane= XZ
   Distance in Y=  29.23
   Symmetry plane= 0
```

```
POCKET:
   Extrusion both sides= NO
   Sketching orientation= XY
   Profile([-25.07, 0.00],[0.00,
0.00],[0.00, 14.62],[-25.07,
14.62]);
   Extrusion direction= Z
   linear_extrusion(length=
5.04)
   Subgraph= 2
   Branch= 1

   Datum planes:
   ID= 7
   Plane= XY
   Distance in Z= 27.21
   Symmetry plane= 0

   ID= 8
   Plane= XZ
   Distance in Y= 34.09
   Symmetry plane= 0

   ID= 9
   Plane= YZ
   Distance in X= -17.57
   Symmetry plane= 0
```

Fig. 17. Output of Example 5.1.

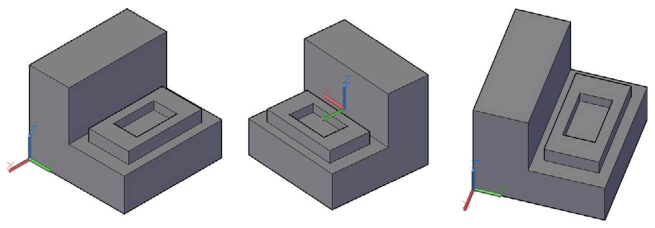


Fig. 18. Reconstructed model of Example 5.1.

5.1. Example: L-shape prism with pocket and step nested

The first case represents a L-shape prism with a step and a pocket nested (Fig. 16 left). The pocket is machined on the upper face of the step, which in turn rests on a horizontal face of the prism (the top level shape). The approach connects the pocket subgraph to one of the edges of its container face, which belongs to the step subgraph, by means of an auxiliary line. Next, the step subgraph is connected to its container face, which belongs to the top level shape. Fig. 16 right shows how the approach resolves the case properly.

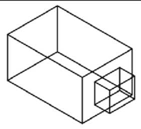
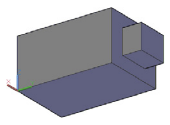
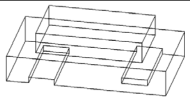
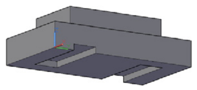
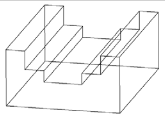
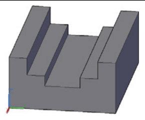
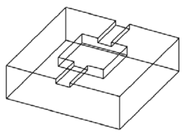
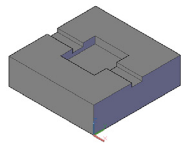
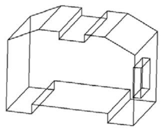
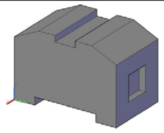
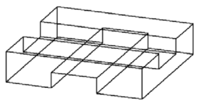
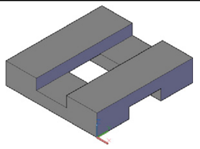
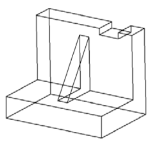
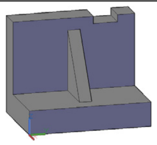
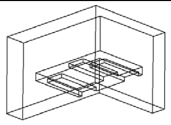
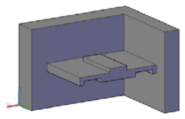
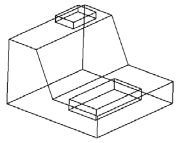
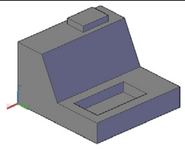
Fig. 17 shows the output. We tested it by reconstructing the model (AutoCAD 3D), as shown in Fig. 18.

5.2. Other examples

After analyzing in detail the output in Section 4.4 and Example 5.1, a battery of new examples is shown in Table 1. Examples are numbered in the first column, the second column depicts the drawing graph before the simplification and the last column shows the 3D model created with AutoCAD with the output data of the approach.

Examples 1 and 5 contain a step and a pocket respectively with extrusion direction parallel to the *y*-axis. They are useful

Table 1
Battery of examples. Column 2 represents the inputs. Column 3 depicts the 3D models manually obtained to validate the output data.

Ex	Drawing graph	3D model
1		
2		
3		
4		
5		
6		
7		
8		
9		

to analyze whether the feature orientation causes failures in the algorithm. The algorithm detects the features properly in each example and creates successfully an edge that acts as a bridge between the two subgraphs.

In addition, Examples 5 and 9 prove that although the approach is limited to work with features oriented parallel to one of the main planes of the general reference system, the top level shape can include slanted faces that are correctly depicted in the extrusion profile as slanted edges.

Examples 2, 7, 8 and 9 break any bilateral symmetry plane of the model, so features cannot define their location from a general symmetry plane. Despite this, the algorithm defines properly the position of each feature (the location of the insertion point of each feature).

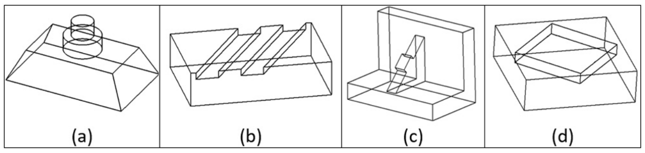


Fig. 19. Examples of features not supported by the approach.

Examples 3, 4 and 6 represent figures with nested features. Example 3 could be interpreted as a prism with symmetric lateral steps, instead, the approach recognizes two nested through slots. In fact the second slot is not recognized since the first one (the smallest one) is simplified in the first iteration. In the case of Example 4, the partial slots could be considered as a single through slot, but the approach consider two through slots, and the pocket is not recognized until both slots are simplified. Example 6 is considered as a prism with two through slots in different directions.

6. Conclusions and discussion

We advocate shifting the semantic level of SBM so as to generate procedural CAD models by (i) identifying features in the 2D sketch and (ii) arranging them around suitable datums also identified in the 2D input sketch. The procedure to identify features was described in [6], while this paper describes a suitable procedure to detect datums for linking features to enable generation of a CSG solid. This completes a demonstration that the two main stages of the procedure are feasible.

We have described the second stage of a general approach which is able to create a complete model tree from a vectorized sketch of an orthogonal polyhedral shape depicted by a pictorial axonometric-like view. The CSG feature tree depicts the reverse design history and includes constructive information such as auxiliary reference systems, which locate each feature in the model. These auxiliary reference systems are made up of datum planes such as contact faces and symmetry planes.

We have shown examples which demonstrate that the approach works properly both when the line drawing is a fully-connected graph and also for cases that contain disconnected subgraphs. The information obtained as output from our approach serves as a proof of concept to demonstrate that it can build a procedural model.

The CSG feature tree obtained in our approach conveys part of the design intent of the model. The model tree can be non-unique, as there are different solutions which reach the same final model. The order and definition of parent–child relationships, with which the designer includes the features in the model tree, defines their importance according to the designer's intention.

Nevertheless, some limitations remain as a matter of future research. The range of features that the approach is able to detect is limited to linear extrusion features with a constant profile. In future work, the range of operations that can be recognized could be extended to other modeling operations like Fig. 19a (sweep with variable cross-section, revolution etc.). This would be added to the Feature Detection stage.

The approach that has proved valid for extrusions seems readily extensible to revolutions. Hence, only the detection of general sweeps that could include variations in the profile remain to be solved. The problem is challenging, but it has been solved in 3D, where approaches to get procedural models from explicit ones are currently available.

The approach does not as yet allow for features whose extrusion direction is not parallel to the main directions (Fig. 19b), which rest in slanted planes (Fig. 19c) or whose edges in the cross-section are not parallel to the main axes (Fig. 19d). In future work, the approach should be extended to extract information of Datum planes and extrusion profiles in slanted planes.

Extending the procedure to oblique planes seems solvable by a similar strategy, while other datums like bilateral symmetry planes or revolution axes depend on procedures that are available for 3D shapes and have been studied to some extent for 2D sketches.

The model tree still lacks information about symmetry between repeated features or the use of patterns to locate them. This information can be considered as part of the design intent and enriches the CSG feature tree, and this issue should be considered in future work.

Finally, any SBM procedure must cope with the inherent imprecision of sketches. A recent contribution by Favreau et al. [52] allows vectorizing sketches with a varying level of detail, thus allowing user control on the fidelity vs. simplicity of the vectorized sketch. As a subject of future research, we advocate easing the detection of the main features and their datums by maximizing the simplicity while vectorizing, thus hiding imperfections and small features. A second analysis increasing the fidelity could be helpful to emerge complementary features (like fillets). This procedure seems suitable for an expert system trained to optimize the level of detail required in each step.

Another question that remains unresolved is how to translate model tree information into a standard language to complete the automation process so that the model can be transferred to a commercial modeler. We note here that most current CSG model files use proprietary formats, since standard formats for procedural models have been fully defined only recently.

Our next step will be to use that information to obtain a representation compatible with the ISO 10303-242 standard (a complex task since it is not a simple format). However, all the information necessary to create a procedural model is already available, so no further analysis of the 2D sketch is required.

CRediT authorship contribution statement

Raquel Plumed: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing, Visualization. **Peter A.C. Varley:** Supervision, Writing – original draft, Writing – review & editing. **Pedro Company:** Conceptualization, Supervision, Writing – review & editing. **Ralph Martin:** Supervision, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partially supported by theGeneral Directorate of Science and Research of the Generalitat Valenciana through the project with ref. GV/2021/079, and the "Transformative agreement" for open access publications of the Universitat Jaume I.

References

[1] Contero M, Company P, Vila C, Aleixos N. Product data quality and collaborative engineering. IEEE Comput Graph Appl 2002;22(3):32–42.
[2] Horvat N, Becattini N, Martinec T, Škec S. Approach to analyse the use of virtual prototypes in distributed design reviews. In: CAD conference; 2021, p. 6–10.
[3] Olsen L, Samavati F, Costa Sousa M, Jorge J. Sketch-based modeling: A survey. Comput Graph 2009;33(1):85–103.

[4] Company P, Piquer A, Contero M, Conesa J, Naya F. A survey on geometrical reconstruction as a core technology to sketch-based modelling. Comput Graph 2005;29:892–904.
[5] Geometrical product specifications (GPS)—Geometrical tolerancing—Datums and datum systems. ISO 5459:2011, ISO; 2011.
[6] Plumed R, Company P, Varley P, Martin R. Csg feature trees from engineering sketches of polyhedral shapes. In: Eurographics 2014, short papers. 2014, p. 33–6.
[7] Donati L, Cesano S, Prati A. A complete hand-drawn sketch vectorization framework. Multimed Tools Appl 2019;78:19083–113.
[8] Wang S, Wang S, He W. Multistroke grouping of online freehand axonometric sketches for mechanical models. Math Probl Eng 2020;2020. art. ID 7439341.
[9] Wang S, Wang S, He W, Qin S. Tolerance zone-based grouping method for online multiple overtracing freehand sketches. Math Probl Eng 2020;2020. art. ID 7393846.
[10] Lipson H, Kurman M. Fabricated: The next world of 3D printing. John Wiley & Sons; 2013.
[11] Wang W, Grinstein G. A polyhedral object's CSG-rep reconstruction from a single 2D line drawing. In: Proc. SPIE int. robots and computer vision iii: algorithms and techniques; 1989. p. 230–8.
[12] Branco V, Costa A, Ferreira F. Sketching 3D models with 2D interaction devices. Comput Graph Forum 1994;13(3):489–502.
[13] Suh Y. Reconstructing 3D feature-based CAD models by recognizing extrusions from a single-view drawing. In: Proc. IDETC/CIE; 2007. p. 197–206.
[14] Shum S, Lau W, Yuen M, Yu K. Solid reconstruction from orthographic views using 2 stage extrusion. Com-Aided Des. 2001;33:91–102.
[15] Lee S, Han S. Reconstruction of 3D interacting solids of revolution from 2D orthographic views. Comput Aided Des 2005;37(13):1388–98.
[16] Pereira J, Jorge J, Branco V, Ferreira F. Towards calligraphic interfaces: Sketching 3D scenes with gestures and context icons. In: WSCG. 2000.
[17] Shesh A, Chen B. Smartpaper: An interactive and user friendly sketching system. Comput Graph Forum 2004;23.
[18] Schmidt R, Khan A, Singh K, Durtengach G. Analytic drawing of 3D scaffolds. ACM Trans Graph 2008;28(5). art. 149.
[19] Company P, Varley P. A method for reconstructing sketched polyhedral shapes with rounds and fillets. In: Smart graphics. Springer Berlin Heidelberg; 2010, p. 152–5.
[20] Company P, Varley P, Plumed R, Martin R. Perceiving ribs in single-view wireframe sketches of polyhedral shapes. In: Advances in visual computing. Springer Berlin Heidelberg; 2012, p. 557–67.
[21] Plumed R, Company P, Varley P, Martin R. From sketches to CAM models: Perceiving pockets and steps in single-view wireframe sketches of polyhedral shapes. In: ACM Conf. on pervasive and ubiquitous computing. Adjunct Publication.; 2013, p. 951–8.
[22] Tanaka M, Kaneeda T. A method of reconstructing 3D models from sketches by extracting features. J. Adv. Info. Technol 2014;5(3):74–8.
[23] Tanaka M, Kaneeda T. Feature extraction from sketches of objects. Comput-Aided Des. Appl 2015;12(3):300–9.
[24] Tanaka M, Asano T, Higashino C. Isometric conversion of mechanical sketches into 3D models. Comput-Aided Des Appl 2020;18(4):772–85.
[25] Tanaka M, Terano M, Asano T, Higashino C. Method to automatically convert sketches of mechanical objects into 3D models. Comput-Aided Des Appl 2020;17(6):1168–76.
[26] Thakur A, Banerjee A, Gupta S. A survey of CAD model simplification techniques for physics-based simulation applications. Comput-Aided Des. 2009;41:65–80.
[27] Lee S. Feature-based multiresolution modeling of solids. ACM Trans Graph 2005;24(4):1417–41.
[28] Lee S. A cad-cae integration approach using feature-based multi-resolution and multi-abstraction modeling techniques. Comput-Aided Des 2005;37(9):941–55.
[29] Cheon S, Kim B, Mun D, Han S. A procedural method to exchange editable 3D data from a free-hand 2d sketch modeling system into 3D mechanical CAD systems. Comput-Aided Des 2012;44:123–31.
[30] Gryaditskaya Y, Hähnlein F, Liu D, Sheffer A, Bousseau A. Lifting freehand concept sketches into 3D. In: ACM Trans Grap. 39, (6). 2020, art. 167.
[31] Kato Y, Tanaka S, Kanamori Y, Mitani J. Single-view modeling for layered origami with plausible outer shape. In: Computer graphics forum (proc. of pacific graphics 2019), vol. 38 (7); 2019. p. 629–40.
[32] Li C, Pan H, Bousseau A, Mitra N. Sketch2cad: Sequential CAD modeling by sketching in context. ACM Trans Graph 2020;39(6). art. 164.
[33] Rawat W, Wang Z. Deep convolutional neural networks for image classification: A comprehensive review. Neural Comput 2017;29(9):2352–449.
[34] Isola P, Zhu J, Zhou T, Efros A. Image-to-image translation with conditional adversarial networks. In: IEEE conference on computer vision and pattern recognition (cvpr); 2017. p. 5967–76.
[35] Ma T, Kuang P, Tian W. An improved recurrent neural networks for 3d object reconstruction. Appl Intell 2020;50:905–23.
[36] Lee S, Han S. Rapidly finding CAD features using database optimisation. Comput Aided Des 2015;69:35–50.
[37] Pratt M, Anderson B. A shape modelling API for the STEP standard. Tech. Rep, National Institute of Standards and Technology; 2000.
[38] Shah J, Mantyla M. Parametric and feature-based cad/cam. John Wiley & Sons; 1995.
[39] Bianconi F. Towards a procedural CAD model for data exchange: problems and perspectives. In: Proc. 17th ingegraf-15th adm; 2005.
[40] Salomons O, Van Houten F, Kals H. Review of research in feature-based design. J. Manuf. Sys. 1993;12(2):113–32.
[41] Shahin T. Feature-based design- an overview. Comput-Aided Des. 2009;5:639–53.
[42] Industrial automation systems and integration—Product data representation and exchange. Part 242: Application protocol: Managed model-based 3D engineering. ISO 10303-242:2020, ISO; 2020.
[43] Kang D, Masrry M, Lipson H. Reconstruction of a 3D object from main axis system. In: AAAI fall symposium series: making pen-based interaction intelligent and natural; 2004.
[44] Varley P, Company P. A new algorithm for finding faces in wireframes. Comput-Aided Des. 2010;42(4):279–309.
[45] Varley P. Automatic creation of boundary-representation models from single line drawing [Ph.D. thesis], Dept. of Computer Science. Univ. of Wales; 2003.
[46] Plumed R, Company P, Varley P. Detecting mirror symmetry in single-view wireframe sketches of polyhedral shapes. Comput Grap 2016;59:1–12.
[47] Varley P, Takahashi Y, Mitani J, Suzuki H. A two-stage approach for interpreting line drawings of curved objects. In: Sketch based interfaces and modeling. The Eurographics Association; 2004, p. 117–26.
[48] Company P, Varley P, Plumed R. Perimeter detection in sketched drawings of polyhedral shapes. In: STAG: Smart Tools and Applications in Graphics. 2017.
[49] Company P, Varley P, Plumed R. An algorithm for grouping lines which converge to vanishing points in perspective sketches of polyhedral. Grap Recog Curr Trends Chall 2014;LNCS 8746:77–95.
[50] Gomis J, Company P. La proyección en el proceso de representación y en los sistemas de representación. Anales de Ingeniería Gráfica 1995;4(1):20–31.
[51] Varley P, Martin R, Suzuki H. Progress in detection of axis-aligned planes to aid in interpreting line drawings of engineering objects. In: SBM. 2005.
[52] Favreau J, Lafarge F, Bousseau A. Fidelity vs. Simplicity: a global approach to line drawing vectorization. In: SIGGRAPH Tech. Paper. 2016.