CSG Feature Trees from Engineering Sketches of Polyhedral Shapes

R. Plumed[1], P. Company[2], P.A.C. Varley[2] & R.R. Martin[3]

[1]Department of Mechanical Engineering and Construction, Universitat Jaume I, Castellon de la Plana, Spain
[2]Institute of New Imaging Technology, Universitat Jaume I, Castellon de la Plana, Spain
[3]School of Computer Science & Informatics, Cardiff University, Cardiff, UK

Abstract

We give a method to obtain a 3D CSG model from a 2D engineering wireframe sketch which depicts a polyhedral shape. The method finds a CSG feature tree compatible with a reverse design history of a 2D line-drawing obtained by vectorising the sketch. The process used seeks the CSG feature tree recursively, combining all design or manufacturing features embedded in the sketch, proceeding in reverse order from the most detailed features to the blank.

Categories and Subject Descriptors (according to ACM CCS): J.6 [Computer–Aided Engineering]: Computer–Aided Design—, I.3.5[Computational Geometry and Object Modeling]: Constructive solid geometry (CSG).

1. Introduction

Most designers use a pencil and paper during conceptual design, while 3D CAD models are the usual input for subsequent stages of the design process. *Sketch-based modelling* (SBM) aims to automatically obtain 3D models from 2D sketches.

Different approaches exist for SBM, but most output *boundary representation* (B-rep) 3D models. Reconstructing *constructive solid geometry* (CSG) models is regaining interest due to recent advances in personal additive manufacturing in the form of 3D printing, or computer-aided manufacturing systems for mass customization: producing 3D digital models from sketches may shorten and simplify the CAD/CAM process, easing modifications in customisable design features and allowing even non-expert end-users to produce their own designs.

We present an approach to capturing the design intent embedded in a 2D sketch and automatically producing a 3D model of it ready for use in a manufacturing process. To this end we seek a CSG feature tree that suitably combines all the design and manufacturing features embedded in the sketch.

We apply techniques of feature recognition, designed to work with *solid models*; our novel idea is to adapt these to detect geometric information at a high semantic level in a 2D *sketch*, before producing a 3D model.

We propose a method to obtain a CSG feature tree from which a CSG model can be derived. The CSG feature tree is built in reverse order from child to parent: we first detect features more likely to be children, add them to the tree, remove them from the sketch, and search for their parents. The process continues recursively until the blank is reached. This process is done on a 2D drawing using features detected in 2D, before inflation of the sketch to 3D.

Related work is discussed in Section 2. Our approach is explained in Section 3, while Section 4 details the process of determining the CSG feature tree. Results are demonstrated in Section 5; Section 6 summarises our conclusions and discusses future work.

2. Related Work

The transformation of sketches into *line-drawings* is a key task in SBM which we take for granted here. Readers interested in this topic may consult the interesting report [BZC*98]. In the rest of the paper, we assume that our input is a line-drawing.

Although 3D B-rep models are the typical output produced by SBM approaches, some attempts have been made to reconstruct CSG models. Some of them work with *single view* (axonometric like) representations: Wang and Grinstein [WG89] produce a CSG representation where every

DOI: 10.2312/egsh.20141008

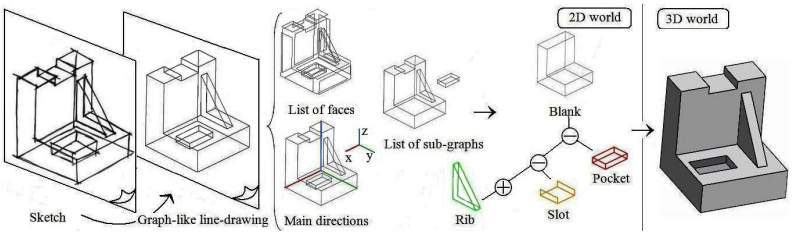

Figure 1: *Process diagram for obtaining a 3D model from a sketch.*

feature is a cuboid. Branco et al [BCF94] use WIMP interaction combined with sketch input. However these pioneering approaches seem not to have been followed up.

Other approaches focus on *multiple* orthographic views [LH05], and/or require *interaction*, neither of which we assume. Furthermore, such CSG-based reconstruction approaches only detect form features, not design or manufacturing features.

Varley [Var03] showed how to detect various design or manufacturing features. However, he did not use feature detection in a systematic way to obtain a feature-based CSG model, but to complement the main process of cue detection aimed at labelling and inflating natural drawings, as well as determining their hidden parts. Our approach is closer to Suh's work [Suh07], who proposed to build a feature extrusion tree. Nevertheless it was based on geometric features instead of design features as we do.

Recent work has been done on identifying and cataloguing the most common design features in engineering sketches [PVC13]. Some criteria which can be used to identify these design features were also discussed.

Algorithms to detect certain features in 2D line-drawings already exist, e.g. for finding design features such as rounds and fillets [CV10], ribs, slots and rails [CVPM12], and steps and pockets [PCVM13].

However, in the field of automated feature recognition, few approaches deal with 2D drawings. Meeran and Pratt [MP93] developed general rules for recognising common machining features in prismatic parts, and could even handle some features that interact with each other.

Vanderbrande and Requicha [VR93] proposed a rule-based approach to obtain a volumetric decomposition of features from the solid model. They tackle complex interactions between features through a procedure of *feature completion*. The main difference with the approach proposed here is that their input for feature detection is a 3D model. In our case, the input is a single 2D wireframe view, which necessarily contains incomplete information. Thus, detecting cues or hints that may reveal the existence of a particular design or manufacturing features becomes a critical issue.

3. Modelling Process

As shown in Figure 1, our input data are strokes, and our final goal is a 3D model. Sketch segmentation and vectorisation 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 clues or cues related to shape, such as sub-graphs, faces and main directions. All of these provide the input from which feature recognition is carried out.

Our goal here is to use features detected in the 2D drawing as input to determine the model tree. Thus, we intend to order the detected features in a hierarchy based on their mutual parent-child relationships.

Our approach is inspired in the work by Li et al [LLM06], which decomposes boundary representation models into regularity feature trees (RFTs). The main differences are that they work with 3D models instead of 2D drawings, and they do not look for design features but regularities and intended geometric relations between subparts.

We move from child to parent, i.e. from branch to root. Therefore when analysing a sketch, we apply a *reverse order strategy*: we first try to find child features and remove them from the drawing, then proceed to consider their parents. The strategy of removing already detected features aids detection of higher-level features which, due to their interactions with child features, are *masked* in the 2D drawing. The process continues until an elementary *blank* (model of a lump of material which you can work on by adding or subtracting features on it) is reached, or no more features can be detected. This work explains how we get the CSG feature tree by applying this iterative strategy.

The hierarchy of features in the model tree may be difficult to tackle when interdependencies among them appear. For instance, if one of two crossing slots is directly removed, the other may split into two. If a main feature is removed early in the search process, the remaining drawing may be difficult to interpret because of disconnected fragments.

Our final goal is to use the determined CSG feature tree to automatically obtain a CSG model ready for use in computer-aided process planning (CAPP).

4. Overview

Determining a *reverse design history* implies finding features, deciding which of them are child features, and linking them to their parents. Our approach recognises features in a drawing (see Section 4.1). It assumes that the parents are the most significant features and children are less important ones. Section 4.2 describes criteria used to classify features by importance. Next, the types of child and parent features are used to determine the corresponding Boolean operations that link them. Finally, we remove child features to help in detecting their parents (see Section 4.3) and continue the process recursively, until we get the blank (Figure 2).

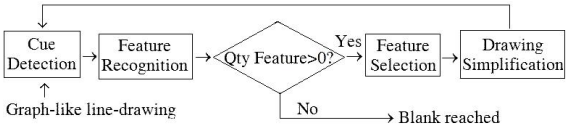

Figure 2: *Iterative process to get the blank.*

4.1. Feature Recognition

Hint-based *automatic feature recognition* (AFR) strategies are not new in SBM systems. The novelty is to adapt them to 2D recognition, as used in Figure 2, rather than 3D. Various criteria to identify such design features are discussed in [PVC13], where indeed a catalogue of the most common design features in engineering is presented. In absence of depth information, this adaptation relies on consideration of indirect cues. Currently, we use the following information:

- The main directions of the axonometric view, obtained as explained in Kang et al [KML04];
- A list of faces detected and numbered as in [VC10];
- Information on subgraphs and edge-labelling, following the approaches of Varley [Var03].

We then use algorithms to detect rounds and fillets [CV10], ribs, slots and rails [CVPM12], and steps and pockets [PCVM13]. All these processes aim to algorithmically replicate human perception of incomplete (sometimes inconsistent) sketches, and so return statistical likelihoods. The outcome is a list of candidate features with confidence values in the range [0, 1].

4.2. Feature Selection

As we try to find child features first, we assume the following hierarchical criteria when searching for features:

- Rounds and chamfers are the least important features. They are usually added at the last stage of the design process as final touches to the product.
- Ribs are usually added in later stages of design, as they are usually used to reinforce an existing design.
- Other features may be added in any order.

To disambiguate between features of equal priority, a size strategy is applied:

- Designers are likely to proceed from the largest features (which may act as containers) to the smallest (usually contained within the former). So, features with smaller bounding boxes are considered first.

Finally, our algorithm is aimed at replicating some behaviour we have observed in designers to get CSG models by way of 2D CAD applications:

- Designers usually start by drawing additive features (rails, steps and bosses) and proceed later to draw subtractive features (slots, pockets and holes).
- Designers generally draw features located in faces parallel to coordinate planes before ones in slanted faces.

Once a feature has been found, we look for a parent feature. We also use indirect cues to infer the parent–child relationship. For instance, most features we detect (apart from rounds and chamfers) have a face (referred to as the *contained* face) which rests on some other face belonging to the *container* feature. If it exists, this parent feature will be the one removed in the next iteration. This allows us to find branches of the tree by following related contained and container features until the last container face belongs to the blank. This finishes the current branch of the tree and, if further features are still detected, we start again to obtain the lowest leaf of a new branch.

Although we can process features where child and parent interact, currently we can only handle independent branches without intersection interference. We are working on extending our approach to cases where interference exists. If a feature is not supported by the system, it is considered to be part of the blank. Although no information is lost, this does not lead to optimal results. In the future, we hope to increase the catalogue of features and add a strategy to verify the blank's structure.

4.3. Drawing simplification

Once a feature has been determined, the next step involves its removal from the drawing. Simplifying the drawing implies removing edges and vertices belonging to the selected feature. This is easy enough if the feature forms a subgraph of the drawing by itself. Otherwise, the simplified drawing may require additional modification (when removing rounds and chamfers, adjacent edges must be merged and/or extended until they intersect in the corresponding vertices).

5. Results

Figure 3, 4 and 5 show examples of our approach of sequential simplification. In Figure 3, in the first iteration a slot (confidence 0.86) and a divider (confidence 0.41) are detected. After removing the slot, the divider becomes a rib.

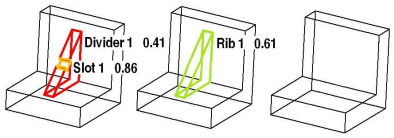

Figure 3: *Block with a rib which contains a slot.*

After removing the rib, the remaining shape is the blank, as there are no more features. In Figure 4, the smaller feature (Pocket 1) is removed. Then its step exists in a lateral face. This is because following the current branch is prioritised container feature, Step 2 is removed, even though a smaller. Finally, Figure 5 shows how the algorithm detects three slots

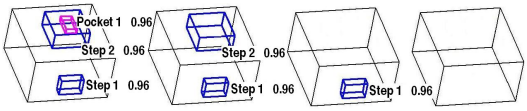

Figure 4: *A block with two independent branches.*

in the first iteration and after removing two of them, the third slot becomes a pocket. However a designer would likely perceive this as a single slot interrupted by a pocket – this would give a simpler explanation. Such interference is not yet included in the analysis, and remains a topic for improvement in future work.

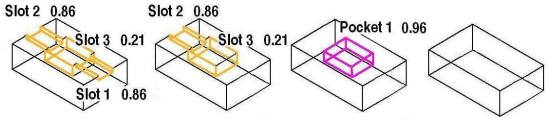

Figure 5: *Interference among features.*

6. Conclusions

We have proposed a general methodology for obtaining a CSG feature tree from a 2D drawing. Our strategy applies a recursive strategy where child features are detected and ranked by a figure of merit. The most suitable feature is selected and removed from the drawing, allowing the drawing to be simplified. This simplification eases the process of recognition since certain features that were previously masked may now be recognised; this allows us to handle some cases of child-parent feature interaction.

Currently, the approach can reach a blank from sketches with a range of kinds of design or manufacturing feature. We are continuing to devise further strategies for handling more complex feature interferences. Implementation of verification strategies will also be considered.

Acknowledge

Work partially funded by "Pla de Promocio de la Investigacio de la Universitat Jaume I", project P1 1B2010-01

References

[BCF94] BRANCO V., COSTA A., FERREIRA F.: Sketching 3D models with 2D interaction devices. *Computer Graphics Forum 13*, 3 (1994), 489–502. doi:10.1111/1467-8659.1330489. 2

[BZC*98] BLOOMENTHA M., ZELEZNIK R., CUTTS M., FISH R., DRAKE S., HOLDEN L., FUCHS H.: Sketch-N-Make: Automated machining of CAD sketches. *In DETC98/CIE-5708* (1998), 1–11. 1

[CV10] COMPANY P., VARLEY P.: A method for reconstructing sketched polyhedral shapes with rounds and fillets. *LNCS 6133* (2010), 152–155. doi:10.1007/978-3-642-13544-6_14. 2, 3

[CVPM12] COMPANY P., VARLEY P., PLUMED R., MARTIN R.: Perceiving ribs in single-view wireframe sketches of polyhedral shapes. *LNCS 7432* (2012), 557–567. doi:10.1007/978-3-642-33191-6_55. 2, 3

[KML04] KANG D., MASRY M., LIPSON H.: Reconstruction of a 3D object from main axis system. *AAAI Fall Symposium Series: Making Pen-Based Interaction Intelligent and Natural* (2004). 3

[LH05] LEE H., HAN S.: Reconstruction of 3D interacting solids of revolution from 2D orthographic views. *Computer-Aided Design 37*, 13 (2005), 1388–1398. doi:10.1016/j.cad.2005.01.007. 2

[LLM06] LI M., LANGBEIN F., MARTIN R.: Constructing regularity feature trees for solid models. *LNCS 4077* (2006), 267–86. doi:10.1007/11802914_19. 2

[MP93] MEERAN S., PRATTS M.: Automated feature recognition from 2D drawings. *Computer Aided Design 25*, 1 (1993), 7–17. doi:10.1016/0010-4485(93)90061-R. 2

[PCVM13] 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. *ACM conf. on Pervasive and ubiquitous computing, adjunct publication* (2013), 951–58. doi:10.1145/2494091.2499207. 2, 3

[PVC13] PLUMED R., VARLEY P., COMPANY P.: Features and design intent in engineering sketches. *Studies in Comput. Intelligence*, 441 (2013), 77–106. doi:10.1007/978-3-642-31745-3_5. 2, 3

[Suh07] SUH Y.: Reconstructing 3D feature-based CAD models by recognizing extrusions from a single-view drawing. In *Proc. IDETC/CIE 2007* (2007), pp. 197–206. doi:10.1115/DETC2007-35186. 2

[Var03] VARLEY P.: *Automatic creation of boundary representation models from single line drawing*. PhD Thesis. Dept. of Computer Science. Univ. of Wales, 2003. 2, 3

[VC10] VARLEY P., COMPANY P.: A new algorithm for finding faces in wireframes. *Computer-Aided Design 42*, 4 (2010), 279–309. doi:10.1016/j.cad.2009.11.008. 3

[VR93] VANDERBRANDE J., REQUICHA A.: Spatial reasoning for the automatic recognition of machinable features in solid models. *IEEE Trans on Pattern Analysis and Machine Intelligence 15*, 12 (1993), 1269–85. doi:10.1109/34.250845. 2

[WG89] 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*, 1192 (1989), 230–238. doi:10.1117/12.969737. 1