From Sketches to CAM Models: Perceiving Pockets and Steps in Single-view Wireframe Sketches of Polyhedral Shapes

Raquel Plumed
Dept. Mechanical Engineering and Construction
Universitat Jaume I
Castellón de la Plana 12071, Spain
plumed@uji.es

Pedro Company
Inst. of New Imaging Technology
Universitat Jaume I
Castellón de la Plana 12071, Spain
pcompany@uji.es

Peter Varley
Inst. of New Imaging Technology
Universitat Jaume I
Castellón de la Plana 12071, Spain
varley@uji.es

Ralph Martin
School of Computer Science & Informatics
Cardiff University,
Cardiff, UK
Ralph.Martin@cs.cardiff.ac.uk

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
UbiComp'13 Adjunct, Sept 8-12, 2013, Zurich, Switzerland.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2215-7/13/09…$15.00..

Abstract
We propose the direct production of 3D CSG models from sketches as a way of relieving the user from having to input detailed 3D CAD models. This shortens the CAD/CAM process and simplifies it, allowing non-expert end-users to produce their own designs. Early detection of features in the 2D sketch is a critical step. This paper discusses a general strategy for solving this problem, and then describes our approach for detecting steps and pockets in a 2D line-drawing obtained after vectorising the sketch captured by an input device.

Author Keywords
Sketch-Based Modelling, CSG models, Features, Steps, Pockets.

ACM Classification Keywords
J.6 COMPUTER-AIDED ENGINEERING
I.3 COMPUTER GRAPHICS: I.3.5 Computational Geometry and Object Modeling: Constructive solid geometry (CSG)
I.4 IMAGE PROCESSING AND COMPUTER VISION: I.4.8 Scene Analysis: Object recognition

I.5 PATTERN RECOGNITION: I.5.4 Applications: Computer vision

Introduction

Fabricating real-world objects and products which can be designed and built directly by the end-user is a difficult but achievable long term goal if we take advantage of different existing disciplines. One such discipline is *sketch-based modelling* (SBM), aimed at providing design tools which help designers to produce 3D digital models from the sketches produced in the earliest stage of the design process.

After sketching a conceptual design, the design process usually proceeds to a detailed design in the form of a 3D CAD model. Next, these 3D CAD models are typically used as input data to produce CAM models and plans. As 3D CAD models do not always convey explicit design and manufacturing features required in a CAM model, a stage which extracts those features is usually required.

Two different feature technologies are in use: *design-by-features* and *feature recognition*. Design-by-features adds features interactively to CAD models, based on the manufacturing-oriented operations which would be used to produce them (e.g. modelling by drilling holes, but not by extrusion, which is not a machining process). Feature recognition post-processes a CAD model to algorithmically extract manufacturing features from its 3D shape. Both technologies have proved valid workflows for high technology industries. However, neither allows end-users to design and build objects directly, as they require designers skilled in producing 3D models, and collaboration between designers and manufacturing specialists to solve the typical problems that arise during CAD to CAM conversion.

SBM may help end-users. Instead of first producing 3D CAD models from sketches, and then processing those CAD models for CAM, our vision is producing 3D CAM-ready models directly from 2D sketches. This idea is not new (see [1–3]); what *is* new is doing so by detecting semantic high-level geometric information in the form of design and manufacturing features in the 2D sketch. Our aim is to capture the design intent embedded in the original 2D sketch, and automatically produce a 3D CAM-ready model. In detail, our goal is to obtain a *constructive solid geometry* (CSG) feature tree that suitably combines all of the design and manufacturing features embedded in the sketch, after detecting them in the 2D line-drawing obtained by vectorising the sketch.

In this paper, after first reviewing the state of the art, we discuss our general strategy for deriving a CSG model from a sketch. Next, we describe our approach for detecting two specific features, steps and pockets, in such a 2D line-drawing.

State of the art

Two related fields must be reviewed: SBM with the goal of making CSG models, and detection of design and manufacturing features.

The conversion of sketches into line-drawings is not detailed here. For the rest of the paper, we assume that a vectorised 2D line-drawing is already available. Readers interested in the state of the art of this topic may read the excellent report provided by [4].

As reported in [5], 3D *boundary representation* (B-Rep) models are the typical target output for SBM approaches. However, some attempts have been made to reconstruct CSG models. Some of them work with single view (perspective or axonometric like) input: Wang and Grinstein [6] produced a CSG representation where every feature was a cuboid. This work was later extended to non-normalon polyhedra with the addition of a second CSG primitive, a tetrahedron. Branco et al [7] presented the IDeS system which used WIMP interaction combined with sketch input. Users were provided with modelling operations such as extrusion to create solids from elementary shapes. These pioneering approaches, which required interaction and were limited to simple form features, seem not to have been followed up.

Other approaches are focused on multiple orthographic views: Shum [8] proposed a two-stage method to reconstruct extruded solids. The first stage obtains a *basic solid* by sweeping each view along its normal direction. The second stage looks for *excess solid*, which is subtracted from the *basic solid*. Subsequent works broadened the domain of objects that can be reconstructed. Soni et al [9] proposed a procedure for identifying non-interacting entities in the orthographic views which depict revolution volumes, while Lee et al [10] applied a hint-based method for recognising even mutually intersecting solids of revolution from orthographic views. Further approaches apply interactive reconstruction and CSG operations when interpreting sketches, for example, Pereira et al [11] presented *GIDeS*, a gestural system, and Shesh et al. [12] introduced CSG operations in their *SMARTPAPER* sketching system, where users invoke a feedback system to modify previous sketch input.

In summary, all of these interactive CSG-based reconstruction approaches only detect *form* features (e.g. extrusions and surfaces of revolution), but *design* and *manufacturing* features are not considered.

Varley [13] detected certain design and manufacturing features. Feature detection complemented the main process of his cue detection method aimed at labelling, inflating and finding the hidden rear part of natural drawings. Feature detection was not used in a systematic way to obtain a feature-based CSG model. Recent work has been done on identifying and cataloguing the most common design features in engineering sketches [14]; criteria to identify these design features were also discussed. Algorithms to detect certain kinds of features in 2D line-drawings already exist, e.g. for finding specific design features such as rounds and fillets [15], or ribs, slots or rails [16]; much work remains to be done.

On the other hand, in the field of automated feature recognition, the main difference from the approach proposed here is that normally the input is a 3D model. Instead we seek early detection of features *before* producing any 3D model. A few approaches deal with 2D drawings. Meeran and Pratt [17] designed an experimental system for meeting automated process-planning requirements based on 2D orthographic views. They developed general rules for recognising common machining features in prismatic parts, and could even handle some interacting features. Meeran and Taib [18] presented a two-stage hint-based feature recognition system based on orthographic views. They find profiles in 2D drawings and add the third dimension to obtain the feature volume. Both approaches apply general rules to find non-interacting and interacting features.

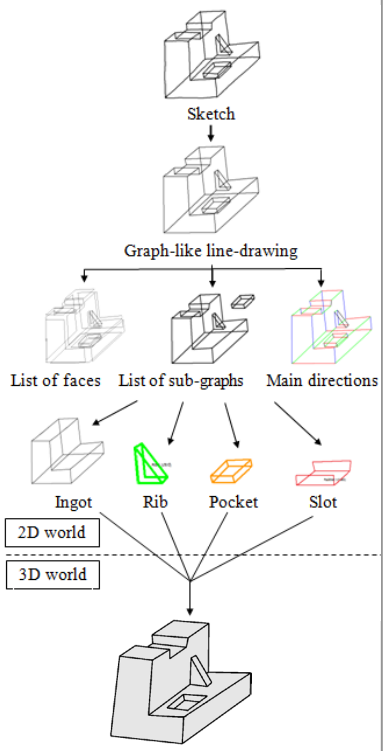

Figure 1. Process diagram for obtaining a 3D model from a sketch.

They are limited in the number of features they can deal with, and in addition features must be oriented parallel to the *x* or *y* axis. Tyan and Devarajan [19] proposed the *FlexiCAD* system, whose input is a 2D orthographic drawing, possibly obtained from a 3D model. The features are identified by geometric patterns and are classified into a hierarchical structure based on characteristic attributes. Their algorithm is limited as it only recognises features that: 1) do not mutually interact, 2) have uniform thickness and 3) are listed in their pattern library.

Approaches originally intended for producing 3D CAM models from old 2D blueprints work with multiple orthographic views. Hence, their strategies are based on extracting and organising information which is complete, although dispersed among different views. In our case, the input is a single view, which necessarily contains incomplete information; hence detecting cues or hints that may reveal the existence of a particular design or manufacturing features is a critical step.

Model tree

As noted, our final goal is to build a 3D model. Hence we have opted for a two-stage approach, based on early detection of features in 2D and defining their mutual relationships (see Figure 1).

To achieve the second step, we intend to build the model tree by recursively detecting features. Our idea is based on observation of common strategies used to build CSG models using 3D CAD applications. When a designer selects a construction strategy, it is very likely that he goes from the biggest features (which may be considered to be `containers') to the smallest ones (usually contained in the former). When the designer must model several non-interacting features of similar size, is usual to start by drawing additive features (ribs, rails, steps), while subtractive features (slots, pockets, holes) come later. The location of the features may also influence the design order; designers usually start with features located in `main' faces (i.e. parallel to the coordinate planes) rather than those located in slanted faces. Therefore, when we parse a sketch, we try to apply a `reverse design history': we first find small or secondary features and then remove them from the drawing. The search and removal process continues until a very elementary `ingot' or blank is reached, or until no more features may be detected.

Our approach is inspired by the work of Li et al [20], aimed at decomposing boundary interpretation (B-rep) models into regularity feature trees (RFTs). The main differences are that their input is a 3D model instead of a 2D drawing, and they do not look for design features, but regularities and intended geometric relations between sub-parts.

We note that detection of features and their mutual interactions is a non-deterministic process. Hence, our approach is aimed at detecting and combining compatible features while rejecting contradictory ones. To this end, we try to algorithmically replicate human perception of an incomplete (and sometimes inconsistent) sketch. For this reason, the proposed algorithm does not return definitive answers, but statistical likelihoods: it answers the question "How likely is this to be a specific feature?"

The reverse design strategy is useful in coping with mutual interdependences. For instance, after detecting a small feature (e.g. a rib), its representation is

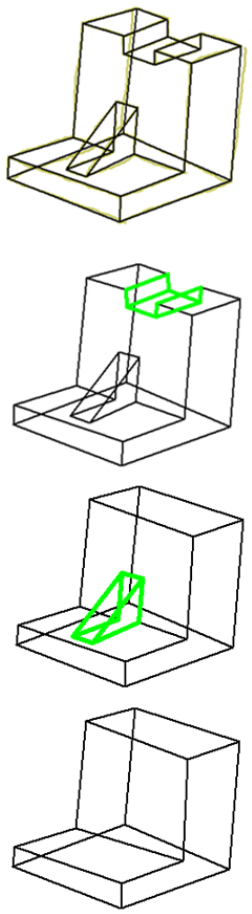

Figure 2. In the reverse-design strategy, the detected local features are sequentially removed and the remaining line drawing is repaired until the ingot is left.

removed from the drawing to help reveal its container. To this end, adjacent edges must be merged and/or extended until they intersect, before the next feature detection step starts (see Figure 2). Here, the sequence is critical, since if a main feature is removed too early in the search process, the remaining drawing may be difficult to interpret (for instance, a fragmented set of small disconnected bodies may result).

Interactions between features can also be critical. For instance, with crossing slots, if one of them is removed, the other splits into two. However other interdependences are beneficial; we can detect features that look similar. If we detect the same cues in both, and discard the less likely ones, we can still get a full set of good quality cues that allow us to safely reconstruct the common feature they represent.

Finally, when interferences are not critical, the strategy of removing already detected features can help in finding those features which, due to their dependence on other features, are masked in the 2D drawing. Obviously, the strategy for reconstructing the parts of the drawing that are hidden by the removed feature will require novel contributions (similar to those for inferring the rear part of an object used by Varley [13]).

Pockets and Steps

We describe here the process we use to detect pockets and steps, as an example of the methodology for detecting design and manufacturing features in a 2D drawing. We deal with them simultaneously, as these features are closely related. Pockets and steps are both a strong cue that the object may be part of an assembly. They provide topological and geometric information about how the assembly is to be assembled. They also may participate in the centering or positioning of assemblies.

The main hint used to find pockets and steps is that they belong to disconnected subgraphs in the graph-formed by the line-drawing.

Finding pockets and steps in single-view wireframes

Our intended output is a CSG feature tree (Figure 1). However we currently aim at an intermediate goal in the form of a list of candidate features and a figure of merit for each of them. The specific algorithm described here reports the existence of steps and pockets, and their likelihood measured by a figure of merit in the range 0-1 (see Figure 4).

The inputs for this stage are the following:

- A graph representing the line-drawing, where nodes depict the vertices of the sketch, and the edges linking the nodes depict the lines of the sketch.
- The main directions of the axonometric view, obtained by the methods in Kang et al [21].
- A list of faces. The process of finding faces in a 2D B-rep of a polyhedral object is explained in [22].
- Information on subgraphs, following the strategy of Varley [13].
- Edges labelled following the extended approach of Varley [13]: labels are *convex*, *concave* or *boundary* (distinguishing boundary loops for each subgraph).

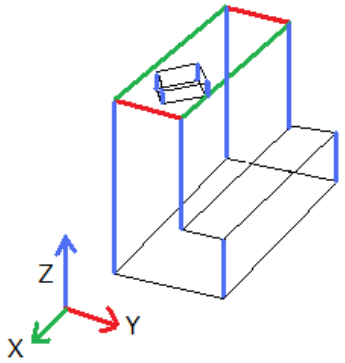

Figure 3. Offset between bases of inner subgraph is parallel to Z-axis. Hence, it is very likely the inner subgraph to be collinear to an outer face which does not contain edges parallel to Z-axis.

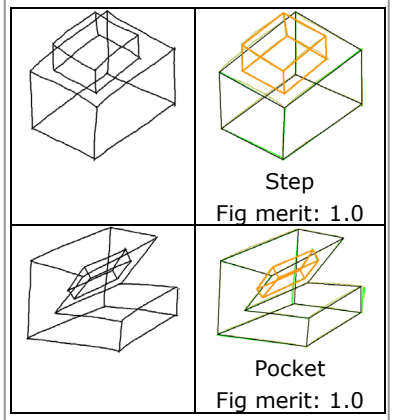

Figure 4. Two examples and their corresponding figures of merit.

There are five stages to the algorithm: A) searching for feature candidates, B) filtering by shape, C) finding outer face candidates D) outer face selection, and E) classifying the feature.

A. Searching for feature candidates

Assuming that pockets and steps belong to disconnected subgraphs, the subgraph which contains the majority of boundary edges of the object is labelled as the *outer subgraph*. The others are *inner subgraphs*.

B. Filtering by shape

Our algorithm starts analysing the shape of each *inner subgraph*. This acts as a filter, discarding subgraphs not of interest.

The algorithm searches for lateral faces, which are depicted by closed circuits of four edges. In particular, they appear as parallelograms. Slots and pockets features are generally shallow compared to their other dimensions, so the parallelograms are narrow, with two short opposed edges and two longer and parallel edges. We note that detecting parallelograms in 2D axonometric views is not trivial, as they appear slanted. If a face satisfies these conditions it is marked as a candidate lateral face, and the adjacent faces to the longest edges are labelled as candidate base faces.

Next the algorithm checks whether the number of faces labelled as lateral faces is two less than the total number of faces in the subgraph (the others are the base faces). If the inner subgraph fulfills this condition, then the figure merit is increased by 0.50. If not, it is directly rejected as a candidate pocket or step.

C. Locating the outer face

Outer faces are all faces that belong to the outer subgraph. We search for outer faces which are coplanar to a base face of the inner subgraph; these are candidates for the containing faces. Coplanarity is easy to detect in 3D. However, we need indirect hints to detect it in 2D.

A strong cue is whether either (or both) base faces of the inner subgraph are contained entirely within one or more outer faces. This increases the figure of merit by 0.15. If not, the inner subgraph is rejected directly as a pocket or step.

D. Outer face selection

When more than one candidate outer face is found, we seek the most likely one. The goal is to choose the outer face coplanar with the feature. The candidate which best fulfills the following conditions is chosen:

- If the edges of the base faces of the inner subgraph are all parallel to the edges of an outer face candidate, then it is assumed that the feature is likely to be oriented similarly to that outer face.
- When the outer face candidate has some edges parallel to any two of the main directions of the drawing, if the inner subgraph is contained in this outer face, the offset between the bases of the inner subgraph may be measured parallel to the third main direction (see Figure 3).

If an outer face is found by this process, it is labelled as a coplanar outer face and the figure of merit is increased by 0.15.

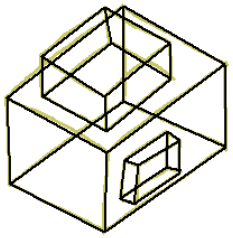

Figure 5. Upper feature is a clear step, while lateral one may be perceived both as a step or a pocket.

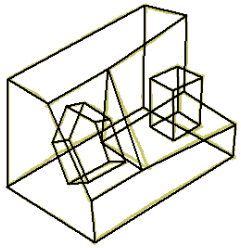

Figure 6. Non-quadrilateral steps and pockets will require previous detection of local symmetries. The example also illustrates that tall protrusions are not labelled as pockets.

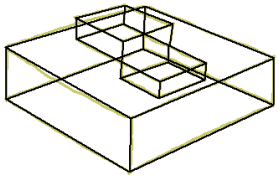

Figure 7. Mutual dependencies prevent the current version of the algorithm from finding the steps and pockets.

E CLASSIFYING THE FEATURE

Finally, as distinguishing between steps and pockets is not trivial, we search for intersections between the outer face which contains the feature (the only coplanar outer face) and the short edges of the feature:

- If intersections exist and the edges of the coplanar outer face are labelled as boundary or concave, then the feature is classified as a step, and the figure of merit is increased by 0.20 (upper feature in figure 5).
- If intersections exist and the edges of the coplanar outer face are convex, the feature is classified as a pocket, and the figure of merit is increased by 0.20.
- If there is no intersection, we search for a candidate base face of the feature coplanar with the outer face. This will be the one whose center of mass is closest to the center of mass of the outer face. Depending on the relative positions of their centers of mass with respect to an inner point of the outer subgraph, the feature is classified as a step or pocket, and the figure of merit increased accordingly. This inner point is the centroid of the point cloud formed by the center of mass of the adjacent faces connected to the outer face by a convex or boundary edge.

Our algorithm makes a choice, even for features which are neither centred nor overlapping the boundary of the outer subgraph. But we have not yet validated this choice against human perception.

Conclusions

We have proposed a general methodology for obtaining a CSG feature tree from a 2D drawing. It is based on a new approach that we call *reverse design history* as it has some methodological relationship with the well-known strategy of exploring the design tree backwards.

We have described an algorithm for searching for specific isolated design features (pockets and steps) using hints and cues embedded in 2D line drawings. The algorithm returns a list of candidate steps and pockets, together with the statistical likelihood of the associated lines representing such features.

Currently the approach is capable of resolving simple sketches with varying different kinds of design and manufacturing features. However, non-centred features that do not overlap the boundary of the outer subgraph (see Figure 5) are still difficult to discriminate. Furthermore, non-quadrilateral steps and pockets will require previous detection of local symmetries (e.g. using the approach in Piquer et al. [23]), to help find their location and discriminate between protrusions and depressions (see Figure 6). Finally, mutual dependencies prevent the current version of the algorithm from finding steps and pockets which do not have separate graphs (see Figure 7).

Acknowledgements

This work was financially supported by the Ramon y Cajal Scholarship Programme, by the "Pla de Promoció de la Investigació de la Universitat Jaume I", project P1 1B2010-01, and by the "Fundació Caixa Castelló – Bancaixa" Ref. E-2012-07.

References

[1] Shah, J.J., and Mantyla, M. Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications. John Wiley & Sons, (1995).

[2] Bloomenthal, M., Zeleznik, R., Cutts, M., Fish, R., Drake, S., Holden, L., and Fuchs, H. Sketch-N-Make: Automated Machining Of CAD Sketches. DETC98/CIE-5708. *Proc. of ASME Design Engineering Technical Conferences*, Computers in Engineering, 1998.

[3] Y.S. Suh, Reconstructing 3D Feature-Based CAD Models By Recognizing Extrusions From A Single-View Drawing, Proc. IDETC/CIE 2007, Paper No. DETC2007-35186, pp. 197-206.

[4] Olsen, L., Samavati, F.F., Costa, M., and Jorge, J.A. Sketch-based modeling: A survey. *Computers & Graphics 33*, 1 (2009), 85-103.

[5] Company, P., Piquer, A., Contero, M., Conesa , J., and Naya, F. A Survey on Geometrical Reconstruction as a Core Technology to Sketch-Based Modelling. *Computers & Graphics 29* (2005), 892-904.

[6] Wang, W., and Grinstein, G.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.

[7] Branco, V., Costa, A., and Ferreira, FN. Sketching 3D models with 2D interaction devices. *Computer Graphics Forum 13*, 3 (1994), 489-502.

[8] Shum, S.S.P., Lau, W.S., Yuen, M.M.F., and Yu, KM. Solid reconstruction from orthographic views using 2 stage extrusion. *Computer Aided Design 33*, (2001), 91-102.

[9] Soni, S., and Gurumoorthy, B. Handling solids of revolution in volume-based construction of solid models from orthographic views. *Journal of Computing and Information 3*, 3 (2003), 250-9.

[10] Lee, H., and Han, S. Reconstruction of 3D interacting solids of revolution from 2D orthographic views. *Computer-Aided Design 37*, 13 (2005), 1388-98.

[11] Pereira, J.P., Jorge, J.A., Branco, V.A., and Ferreira, F.N. Towards Calligraphic Interfaces: Sketching 3D Scenes with Gestures and Context Icons. *Proc WSCG 2000*.

[12] Shesh, A., and Chen, B. Smartpaper: An interactive and user friendly sketching system. *In Proc. of Eurographics 2004*.

[13] Varley, P.A.C. Automatic Creation of Boundary-Representation Models from Single Line Drawings, PhD Thesis, University of Wales (2003).

[14] Plumed, R., Varley, P.A.C., and Company, P. Features and Design Intent in Engineering Sketches. Studies in Comput. Intelligence, 441 (2013), 77-106.

[15] Company, P., and Varley, P.A.C. A Method for Reconstructing Sketched Polyhedral Shapes with Rounds and Fillets. LNCS 6133, (2010), 152-155.

[16] Company, P., Varley, P.A.C., Plumed, R., and Martin, R. Perceiving Ribs in Single-View Wireframe Sketches of Polyhedral Shapes. *Proc ISVC 2012*. Part II, Adv. in Visual Computing, LNCS 7432, 557-567.

[17] Meeran, S., and Pratts, M.J. Automated feature recognition from 2D drawings. *Computer Aided Design 25*, 1 (1993), 7-17.

[18] Meeran, S., and Taib, J.M. A generic approach to recognising not-interacting, nested and interacting features from 2D drawings. *Computer-Aided Design 31*, 14 (1999), 891-910.

[19] Tyan, L.W., Devarajan, V. Automatic identification of non-intersecting machining features form 2D CAD input. *Computer Aided Design 30*, (1998), 357- 66.

[20] Li, M., Langbein, F.C., and Martin, R.R. Constructing Regularity Feature Trees for Solid Models. *Springer Heidelberg 4077*, (2006), 267-86.

[21] Kang, D. J., Masry, M., and Lipson, H. Reconstruction of a 3D object from main axis system. *AAAI Fall Symposium Series: Making Pen-Based Interaction Intelligent and Natural* (2004).

[22] Varley, P.A.C., and Company, P. A new algorithm for finding faces in wireframes. *Computer-Aided Design 42*, 4 (2010), 279-309.

[23] Piquer, A., Martin, R.R. and Company, P. Skewed Mirror Symmetry for Depth Estimation in 3D Line-Drawings. LNCS 3088. (2004), pp 138-149.