Sketch Input of Engineering Solid Models: Tutorial Notes

P. Company and P.A.C. Varley
Department of Mechanical Engineering and Construction, Universitat Jaume I,
Castellón de la Plana, Castellón, Spain
{pcompany, varley}@emc.uji.es

Abstract
In this tutorial, we describe the state of the art of sketch input of engineering solid models. Firstly, we show that sketching has historically been an important aspect of engineering culture. We then discuss and classify various current approaches to computer interpretation of sketches. We present our selection of the most important algorithms used for interpreting sketches of engineering objects. Finally, we discuss some of the most interesting open problems.

1. Introduction

Slides 2 to 15 are a summary of our paper The Importance of Sketching in Engineering Culture [VC08].

Sketches are drawings which are intended as preliminary explorations, not as finished works.

We know that people understand sketches!

We are interested in sketches as they assist product designers during the creative stages of product design.

Most of you, if not all, perceive this:

Those of you who have been trained, perceive this:

Computers are blind to engineering sketches!

New computer tools are required!

! Computer-Aided Design (CAD) tools cannot solve the problem!

...because CAD applications are unable to work with:

✓ confused
✓ poorly structured
✓ incomplete ideas

Slides 2 to 15 are a summary of our paper The Importance of Sketching in Engineering Culture [VC08].
Slides 16 to 22 present the historical background to computer interpretation of engineering drawings. This information has been collated from a number of sources including [Com04], [Com07] and [CCV09].
The former goal of geometrical reconstruction was extracting information from old engineering blueprints. In other words, "archaeological" recovery of old know-how.

But the task proved difficult... because the vectorisation stage is complex...

...and because engineering drawings convey:

- 3D information represented through partial views:
  - orthographic views, particular views, cuts, etc.

- annotations:
  - dimensions, tolerances, etc.

The short term problem was solved through brute force.

Translation services were offered!

Although this goal is still alive in architecture:

In the new millennium, "next generation" systems are needed to deal with the complexity of contemporary engineering drawings.

The main goal of the reconstruction community has changed in the 1990s.

Nowadays, most of the systems are oriented toward conceptual design via sketch-based modelling.

The goal has changed over time:

\[ \begin{align*}
2D + paper & \Rightarrow 2D + computer \\
2D + paper & \Rightarrow 3D + computer \\
Conceptual design & \Rightarrow 3D + computer
\end{align*} \]

VECTORISATION → RECONSTRUCTION → SBM

The current situation in producing solid models from sketches may be summarised as follows:

- There is no general approach which solves all the SBM problems.
- Some critical features produce different bottlenecks.
- States of the art are different for every critical feature.

We propose a taxonomy of critical features:

- Number of views
- Types of surface
- Variety of inputs
- Design intent

Slides 23 to 43 present a taxonomy of existing sketch-based modelling tools, based on a similar taxonomy presented in 2004 [CPC04].
Two kinds of VIEW are distinguished for reconstruction approaches:

- multiple orthographic views
- single pictorial view

More active nowadays:

- perfect line drawings
- line drawings containing some "geometrical" mistakes
- freehand sketches

All three input types have been studied, but...

- perfect line drawings were the most frequent in the beginning...
- now (in single view approaches) we are evolving towards hand-drawn line-drawings.

Use of HIDDEN LINES in the input drawing results in two different inputs:

- wireframes (transparent models)
  - methods where the input includes all lines in the drawings
- natural (opaque models)
  - methods which reconstruct from an input which only contains the visible edges

Both have been studied, but planar surfaces are more developed

- Natural drawings have been less studied than wireframes
- The need to infer the raw of the object makes the reconstruction process more difficult

Our classification distinguishes two kind of SURFACE:

- algorithms which only accept flat surfaces
  - They are generically known as polytopes
- algorithms which accept curved surfaces
  - They are generically known as polytopes
Design Intent and CAD have been linked for many time

However, the definition of Design Intent is ambiguous

Back in 1989 Design Intent was associated with design constraints and the methods of manipulating design constraints during product design activities

...it still continues to be for many people!

When CAD people use the word “design”, they usually mean “model”

Modelling is just representing the design in some way

Design intent equates to the phrase Design for Change

This implies that you are modelling a concept that can be flexible through changes

Something has been done in the SEM sector to cope with design intent understood as design-for-change

We understand design intent as a mix of:

- Geometry
- Psychology
- Engineering

We understand design intent as a mix of:

- Geometry
- Psychology
- Engineering

Consequently, we can define Design Intent as:

The set of intentions in sketches conveyed through cues, which, when perceived, reveal regularities or features of the object

However, no practical approaches have yet considered the explicit capture of complex design intent from the input sketches!

We understand design intent as a mix of:

- Geometry...as far as it is linked to the shape
- Psychology...as far as it is not always explicit in the sketches
- Engineering...as far as it is linked to the function

We understand design intent as a mix of:

- Geometry
- Psychology
- Engineering

When geometry dominates, design intent is mainly conveyed through geometrical features which have already been studied as “regularities”

Example:

Early detection of symmetry in a 2D line-drawing

and improvement of the reconstruction process by making use of symmetry
2. Wireframe Drawings

We shall describe algorithms representative of the current state of the art in these stages:

1. **2D sketching**
   - Learn more on segmentation:

2. **2D beautification or bring up**

3. **Extraction of geometrical and perceptual information**
   - We shall describe our algorithm for finding faces.

4. **Inflating a rough 3D model**
   - We shall describe our algorithm for inflating quasi-normal faces.

5. **3D model refinement**
   - We shall describe our algorithm for optimisation-based inflation.

Slides 5 to 22 are a summary of our paper *A New Algorithm for Finding Faces in Wireframes* [VC10], explaining both why a new algorithm was needed and how the new algorithm works.
Our new algorithm:

**Data Structures:**
- **Strings** are concatenated sequences of half-edges. The shortest possible strings are single half-edges.

**Operations:**
- Two strings can be concatenated if the final vertex of the first string is also the start vertex of the second string, except that:
  1. Two strings cannot be concatenated if any other vertex appears in both strings.
  2. Two strings cannot be concatenated if the new triple of three consecutive vertices appears in reverse order in any existing face or already concatenated string.

**Data Structures and Operations (examples):**
- Starting with only the half-edges, we can concatenate AH and HA to give AHC.
- Once we have AHC, we cannot concatenate CH and HA.
- In fact, the only possible concatenation of CH is with HF, to give CHF.
- Similarly, the only possible concatenation of HA is with FH, to give FHA.

**Data Structures (continued):**
- Cyclisation is a double-concatenation where the final vertex of each string is the same as the start vertex of the other string.
- Cyclisation produces faces.

**Operations (continued):**
- The same rules apply to cyclisation as to concatenation since the purpose of the algorithm is to produce faces, cyclisation takes priority over other operations.
Slides 23 to 27 summarise the current state of the art of inflation. They bring together ideas found in a number of different papers, especially [Per68], [LS96], [CCC04] and [MVS05].
Slides 28 to 41 discuss methods for identifying and processing rounds and fillets. The theoretical basis of this work has already been presented [CV10], and a longer paper is planned which will discuss implementation and results.
3. Natural Line Drawings

Slides 3 to 15 discuss line labelling. Much of the information to be presented is also to be found in [VMS05].
Line labelling labels each line in a drawing as:

- convex
- concave
- occluding

The original purpose of line labelling was as a method of identifying and rejecting impossible drawings.

But line labelling also has many other uses:

1. Line labels indicate which edges border the visible faces or partial faces of the object and which merely occlude them.
2. The underlying vertex types implied by the junction labels limit the possible hidden topologies.
3. The junction labels constrain the geometry of any edges to be extended or added.
4. Labelling is also a useful input to inflation.

Clowes-Huffman line labelling (catalogue labelling) is a well-established technique:

1. It is very effective for drawings of objects containing only trihedral vertices.
2. There are only 18 possible ways of labelling trihedral junctions.
3. Often, there is only one consistent labelling for the whole object.

Cloowes-Huffman line labelling is less effective (when it works at all) for drawings of objects containing higher-order vertices:

- There are over 100 possible ways of labelling 4-trihedral junctions.
  - Drawings of inclined objects usually have many possible labellings.
  - Catalogue labelling is slow and unreliable.
- There are thousands of possible ways of labelling higher-order junctions (5-6-7-8-trihedral).
  - Even determining the catalogue is not practical.
- Cloowes-Huffman labelling can also lead to labellings which have no geometric interpretation.

The most important function of line labelling is to distinguish occluding from non-occluding T-junctions:

- If there is a real vertex at V, the vertex is at least 4-trihedral, so one more edge must be added.
- If there is no vertex at T – it is just the point at which one edge becomes occluded by another. There is a vertex somewhere further along the line, but we do not know anything else about it.

These differences will become important when we try to construct the complete object.

Some specific problems:

1. A junction label which normally indicates an occluding T-junction here represents an extended-K junction.
2. Traditional algorithms methods do not use geometry at all, so cannot distinguish these two.
3. Geometry affects labelling:
   - A line which separates two regions corresponding to parallel faces must occlude one or the other - it cannot be convex or concave.
4. Symmetry constrains labelling:
   - The central line corresponds to an edge with an axis of symmetry through its mid-point, so for reasons of symmetry as well as geometry it cannot be occluding.
5. Non-Local Constraints:
   - When two or more edges lie between the same two faces:
     - If the edges are collinear, the labels must be the same.
     - If they are non-collinear, the labels must be different, and at least one must be occluding.
Slides 16 to 28 discuss inflation to 2½D. This is similar to, but not identical to, the inflation problem for wireframes, as additional compliance functions are available (such as that described in [VMS04]).
Cubic Corners

\[ Z_B - Z_A = \frac{1}{2}AB\sqrt{(\tan C \tan D - 1)} \]

Note that there must be a separate mechanism for determining whether \( A \) is in front of or behind \( B \) (e.g., use depth cues). Even without labeling, we often draw them (e.g., boundary vertices are often behind internal vertices).

Junction Label Pairs:
Consider pairs of connected junctions in the drawing.

We can deduce, just from the line labels, which is the nearer (and roughly by how much).

The other shapes / Huffman solids give us several more junction label pairs.

The extended triangular solids give still more pairs.

However, adding in the 4-junctions (0.1 of them) is impractical.

No 2-label combination involving a 4-junction is common enough to justify hard-coding it in an algorithm.

Adding in the 5-junctions and beyond is not even worth thinking about.

Perpendicularity: Introduction

Assumptions to do with perpendicularity are very important:

- Perpendicularity is the most common regularity in engineering objects.
- Perpendicularity is an important part of the human perception process.

Line Parallelism

\[ n \vec{z}_A \cdot n \vec{z}_B = m \vec{z}_C \cdot m \vec{z}_D \]

Where \( m \) is the 2D length of line \( AB \) and \( n \) is the 3D length of line \( CD \).

- Easily arranged into linear or explicit equations.
- Not inherently inflationary: the trivial solution \( z = 0 \) satisfies the equations.

Face Planarity

- Can be arranged into linear equations if we include face normals as well as vertex \( z \)-coordinates as the unknowns.
- Quadrilateral faces can always be arranged into linear or explicit equations.
- Larger (pentagonal and beyond) faces cannot be arranged into linear equations if the only unknowns are the vertex \( z \)-coordinates.

N.B. making groups of four vertices coplanar does not necessarily ensure that the entire face is planar.

Face Planarity (continued)

- Not inherently inflationary: the trivial solution \( z = 0 \) satisfies the equations.
- Can be used to connect disjoint subgraphs.

Once we have chosen our compliance functions, how do we apply them?

- Linear system approaches are quickest and best.

We shall describe one such linear system approach (\cite{ numericaLorentzianTechniques}).

- Iterative approaches have also been tried:
  - They are slow.
  - There are no compensating advantages.
Slides 29 to 45 discuss the creation of hidden topology, a requirement which is unique to the interpretation of natural line drawings. There are two promising methods. The first, based on completing the object wireframe, is described in Chapter 10 of [Var03] and has not been published separately.
The second (and at the time of writing the more promising), based on reconstructing the polyhedron as the union and intersection of extrusions, is as described in [Suh07], with some minor improvements of our own. The figures in slides 38, 39 and 42 are taken from [Suh07].
4. Open Problems

Slides 3 to 16 discuss how engineers actually use pencil and paper in practice. Do current sketching tools offer all the modalities which users of pencil-and-paper expect? What is missing? This is a summary of [CV09].
We may reduce the gap between actual paper and pencil and SIB tools:
- Replicating as many modes as possible
- Adding extra features

Wrong paradigms guide human-to-computer interaction
- Command-driven and menu-driven
- No interface paradigm which would suit experienced design engineers
- Identifying such a paradigm is a difficult task in itself

Besides, ...
- Adding some current CAD operations could also help in reducing editing tasks

Final goal:
- SIB tools should be as easy to use as actual paper and pencil

To this end:
1. Hardware advances are required
   - For example, tablets have been reported to be less comfortable to use than pencil and paper because of the small gap (both in time and distance) between the cursor and the pencil tip

2. Software improvements are also required
   - User and maintenance of computers still requires technical knowledge which designers, quite rightly, do not see as part of their job
Slides 18 to 33 discuss the problem of interpreting annotated sketches. Which strokes are annotation rather than object? What does the annotation mean? How should it be applied? This section collates information from several published papers ([ICAN08], [CCV09]).
It is obvious that communication of relevant information depends on the meaning of symbols:

Misunderstanding of symbols causes technical information loss!

The problem becomes still more challenging if we realise that new standards already allow annotations in 3D models:

Today, computers are blind to these annotations!

The annotations are just "labels" added to the model

✓ Which the user can read and modify.

x but the geometrical engine does not use them, neither to construct, nor to edit or validate the model.

One interesting related open problem is interpreting sketched data input for Computer-Aided Engineering applications

It is an open problem since data is input through two alternative WIMP user interfaces:

Stand-alone CAE pre-processors

Combination of CAD applications plus downstream CAE pre-processors

CAD exports the geometry and CAE adds attributes

The input are those sketches which designers typically draw aside to fix their ideas before interacting with CAE pre-processors

The output is a file which meets the specifications of the desired analysis code

Two reasonable assumptions are:

1. The input sketches are drawn directly onto a computer screen acting as "virtual paper and pencil"

2. The user is still in the process of conceptual design and is not yet ready to progress to a detailed design stage

Hence, the goals are:

1. supply the user with a computer interface similar to classical paper-and-pencil

2. minimise the amount of information provided by the user ...

... and give the user more freedom in inputting and editing it

Our application, PreAdel, distinguishes:

1. Geometric entities

2. Symbols associated with annotations

3. Gestures associated with editing tasks (i.e. "sketched commands")

Finally, having interpreted each group in isolation, we must combine them into a whole

/ Connect the bar elements to the appropriate nodes

/ Apply the loads to the right nodes or elements

The arrow of the 300 kg force is parallel (more or less) to the arrow of the "Y" axis

They have opposite senses

The vector force should thus be interpreted as (0, -300, 0)
Slides 34 to 40 consider the possibility of sketching assemblies of parts. This is unpublished material which was briefly presented at [Com07b].
Our vision is creating a sketch-based environment…

... able to assemble different parts...

... that are not yet fully defined

References


approximate CAD models using symmetry.
Computer-Aided Design 42 (3) 183-201


