Sketch Input of Engineering Solid Models

3. Natural drawings

Pedro Company

Peter Varley

UNIVERSITAT JAUME I

REGEO

Algorithms for natural drawings

We will describe three algorithms representative of the current state of the art in three stages:

1 Line Labelling

2 Inflation to 2½D

3 Hidden Topology

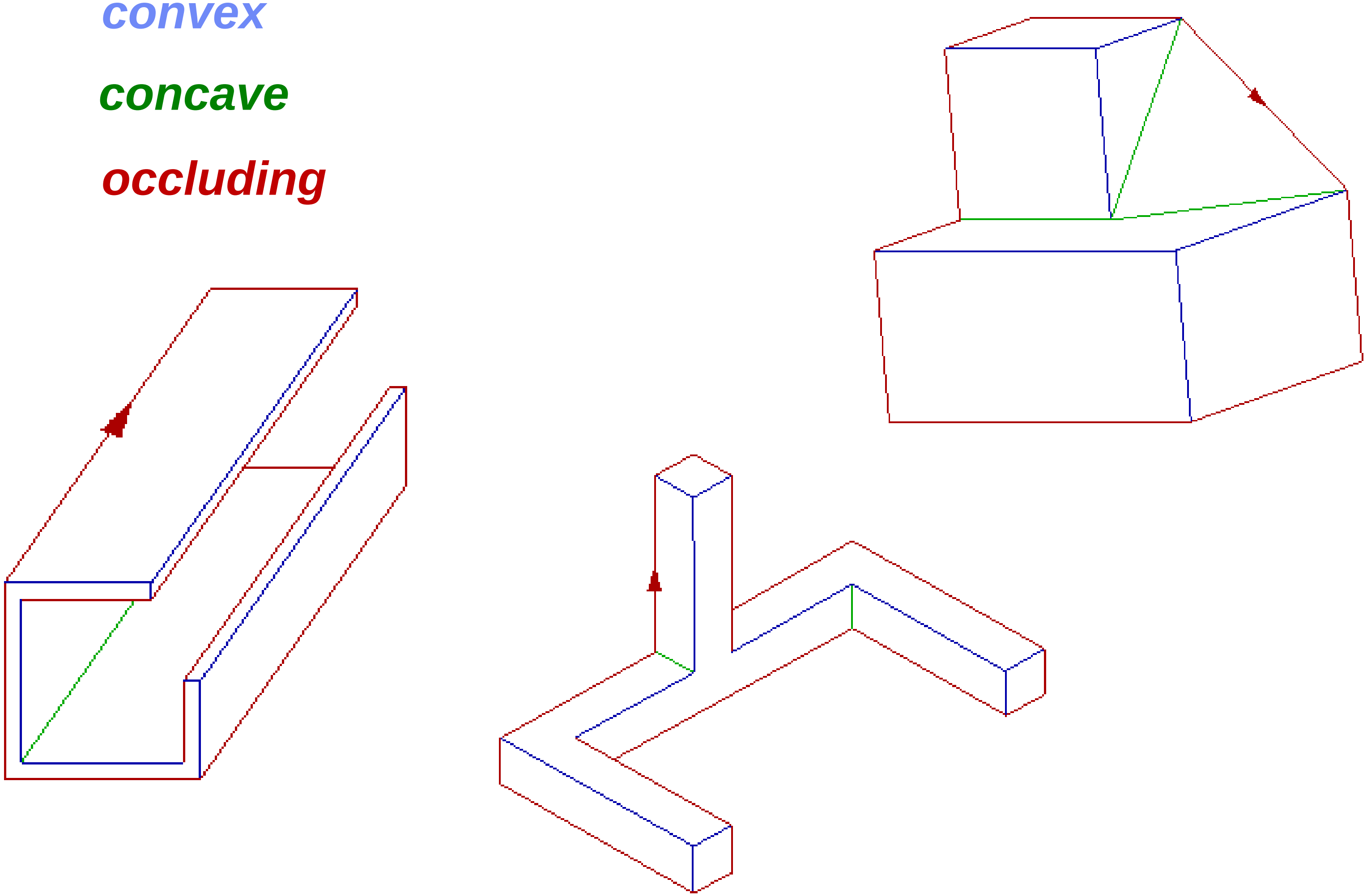
Line Labelling

Line labelling labels each line in a drawing as:

convex

concave

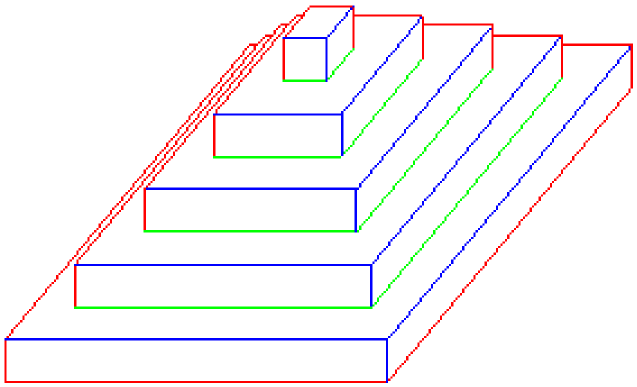
occluding



Line Labelling

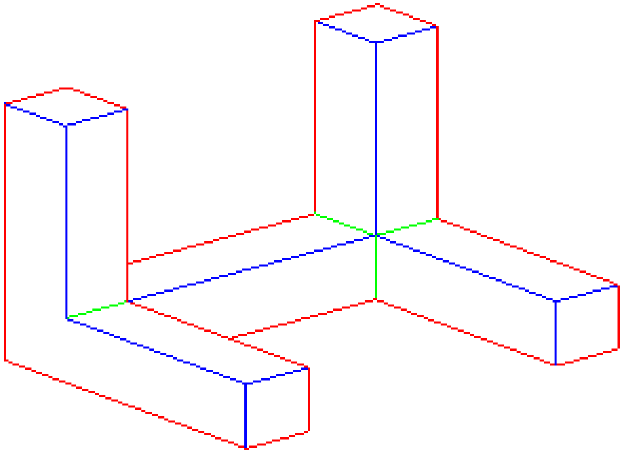
The original purpose of line labelling was as a method of identifying and rejecting impossible drawings

But line labelling also has many other uses …



1 Line labels indicate which edges bound 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

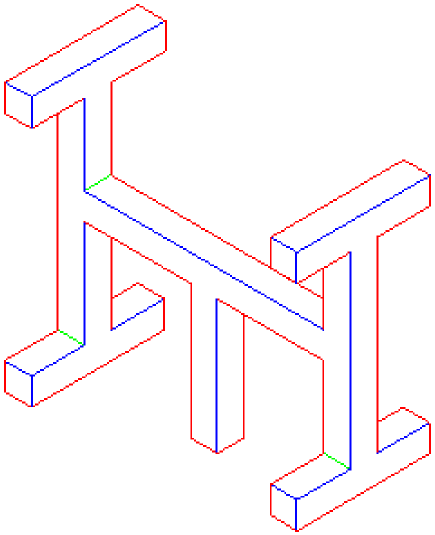


3 The junction labels constrain the geometry of any edges to be extended or added

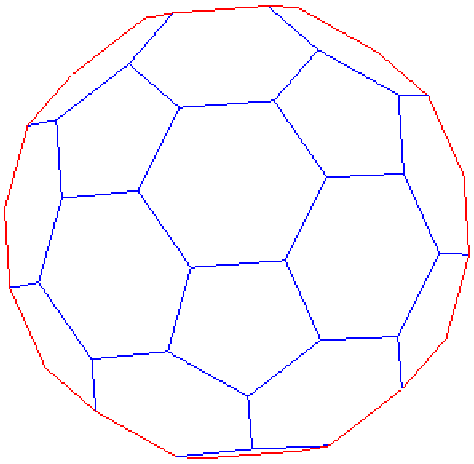
4 Labelling is also a useful input to **inflation**

Line Labelling

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



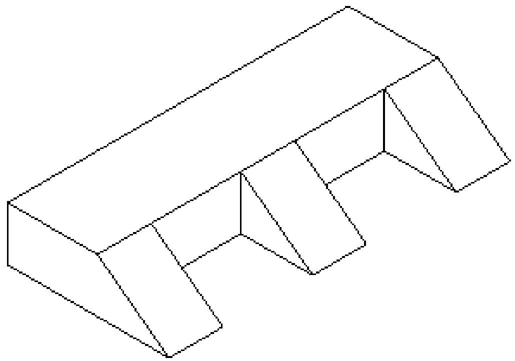
- It is very effective for drawings of objects containing only *trihedral vertices*
- There are only 18 possible ways of labelling trihedral junctions
- Often, there is only one consistent labelling for the whole object

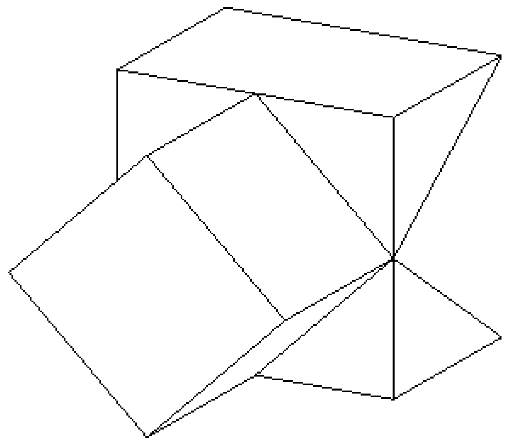


Line Labelling

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

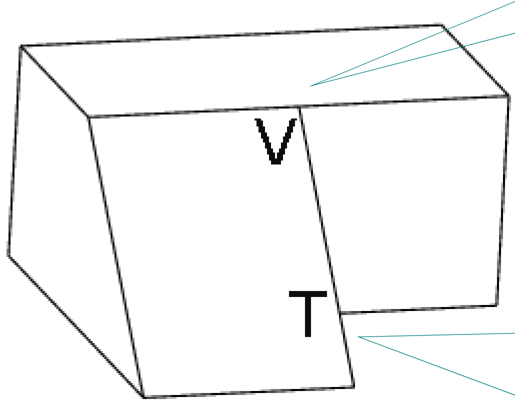
- There are over 100 possible ways of labelling 4-hedral junctions
  - Drawings of tetrahedral objects usually have many possible labellings
  - Catalogue labelling is slow and unreliable
- There are thousands of possible ways of labelling higher-order junctions (5-6-7-8-hedral)
  - Even determining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





Line Labelling

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



There is a real vertex at *V*. The vertex is at least 4-hedral, so one more edge must be added

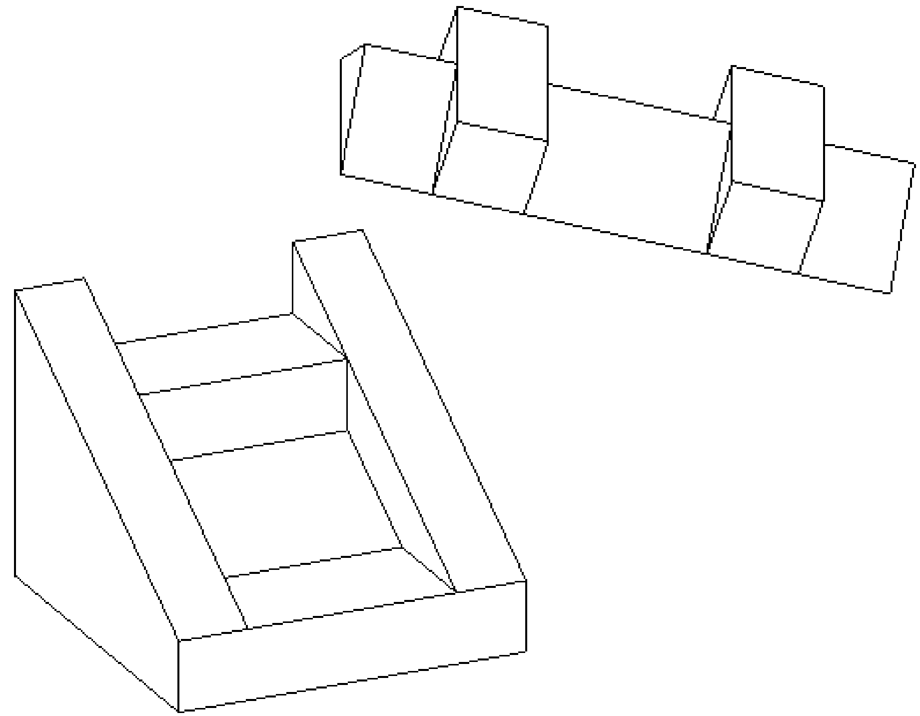
There is no vertex at *T* – it is just the point at which one edge becomes occluded by another. There is a vertex somewhere further along the line, but we do not know anything else about it

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

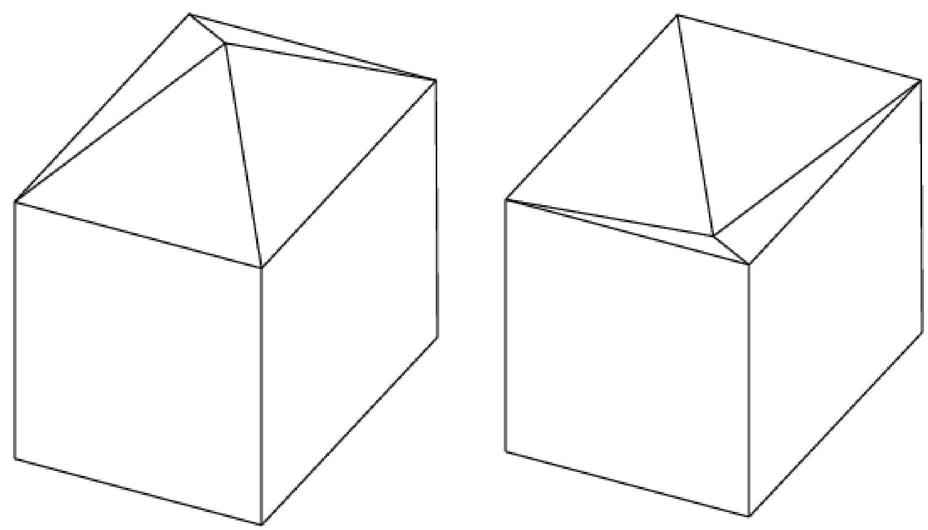
Line Labelling

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

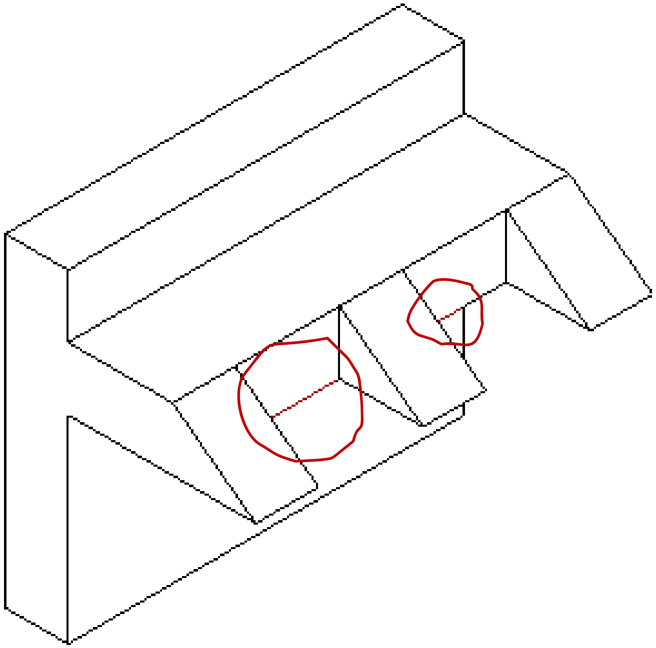


Line Labelling

3 Geometry affects Labelling:

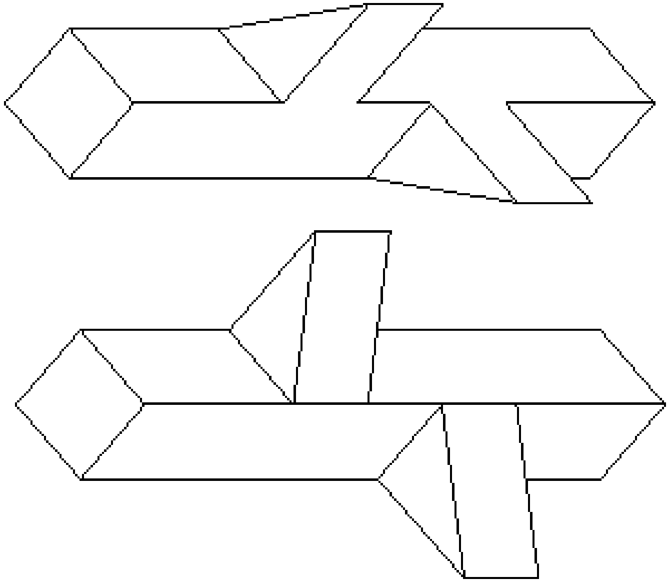
A line which separates two regions corresponding to parallel faces **must** occlude one or the other - it cannot be convex or concave

there are two such lines in this drawing



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

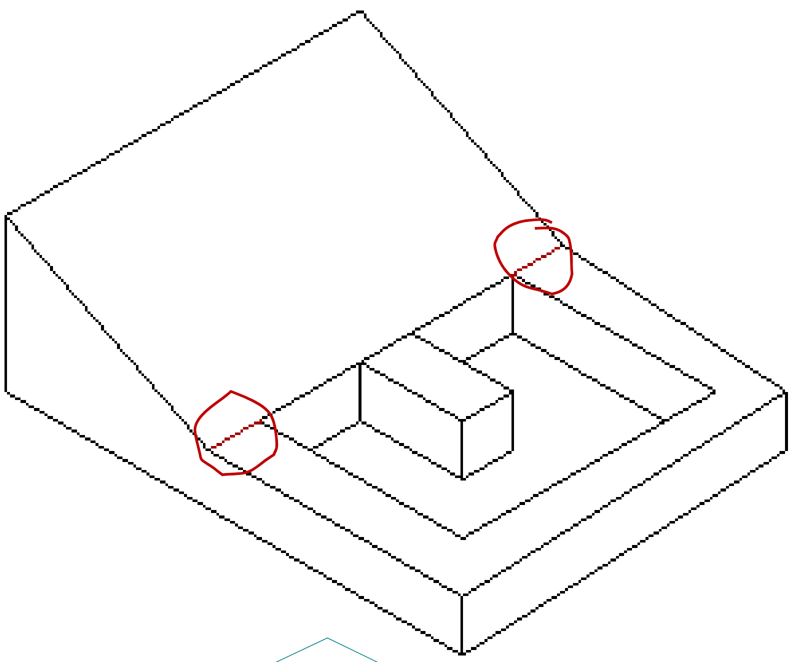


Line Labelling

5

Non-Local Constraints:

When two or more edges lie between the same two faces



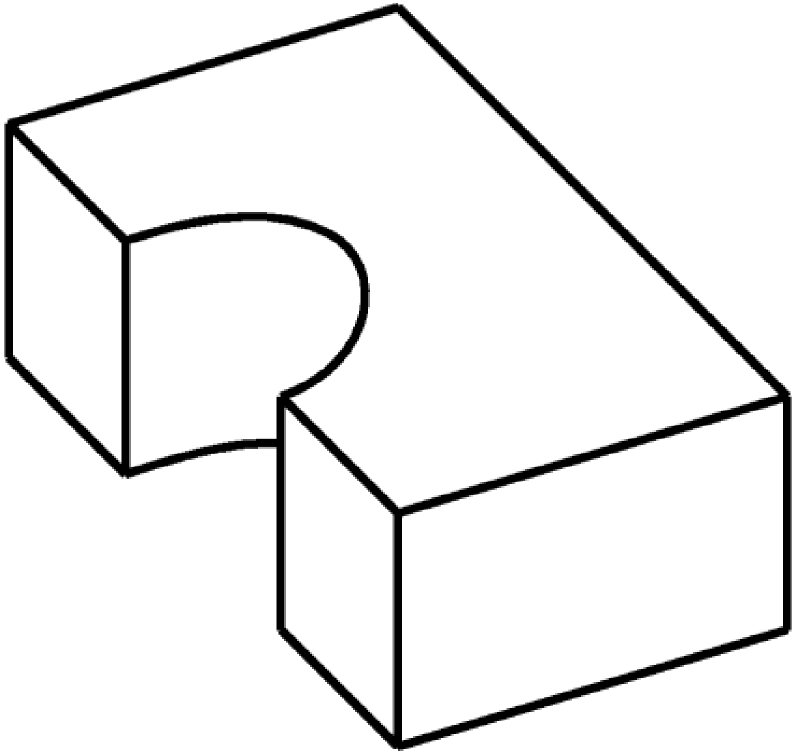
If the edges are collinear, the labels must be the same

If they are non-collinear, the labels must be different, and at least one must be occluding

Here, the two edges are collinear (and both concave)

6 Curved Objects?

In principle, drawings of curved objects can also be labelled, but there are problems



The label of one of the lines in the drawing changes from one end to the other!

State of the Art:

- ✓ Traditional line labelling algorithms solve local discrete constraint satisfaction problems
- ✓ 1-node constraints: each junction must have a valid label
- ✓ 2-node constraints: each line must have the same label at both ends
- ✗ Traditional algorithms cannot handle non-local constraints
- ✗ Traditional algorithms ignore geometry
- ✓ For trihedral drawings, there is often only one solution, so ignoring geometry does no harm
- ✗ When there are many solutions, ignoring geometry causes problems

Line Labelling

Why not determine line labels geometrically?

If we can inflate the drawing to 2½D first, all we have to do is measure the resulting geometry to determine which lines are convex, concave and occluding – we do not need catalogues or constraint satisfaction algorithms

However, line labelling is a useful *input* to inflation

How reliable would inflation be without line labels?

Answer: even without line labels, inflation is usually reliable for drawings which meet all of the following criteria:

- Most corners are cubic corners
- The drawing is not in cabinet projection or similar
- The centre of the drawing is nearer than the edge to the viewer

Line Labelling

Line labelling helps inflation, inflation helps line labelling

This suggests an alternating process, which inflates, determines line labels, re-inflates, re-labels, etc, until it converges

This represents the current state of the art, but although it is reasonably reliable it is still not perfect

It is also comparatively slow

Using a combination of the geometric insights provided by line labelling and those provided by the compliance functions discussed next seems the best way to determine frontal geometry

But there is still research to be done to determine the best combinations

Summary

✗ Line labelling is not going to go away: it is a very well-understood Valued Discrete Constraint Satisfaction Problem, and will continue to be investigated as a test of VDCSP algorithms

✗ At present, there is no reliable catalogue labelling algorithm for 4-hedral objects, and even the catalogues themselves for 5-hedral objects and beyond are too large for determining them to make sense

✓ Even if it is not possible to label a drawing completely, partial labelling remains useful

✓ Even more importantly, the geometric insights from line labelling remain true even if the algorithms used to implement it are limited

Inflation to 2½D

Inflation of natural line drawings to 2½D
is easier than inflation of wireframes:

- ✓ We still use compliance functions
- ✓ Sometimes we use the same compliance functions, but they give us more information
- ✓ If we can label the drawing, this gives us other compliance functions to choose from

Inflation to 2½D

Objectives

1 The depth ordering of adjacent pairs of visible vertices must be correct

2 Depth ordering must not be sensitive to inaccuracies in the drawing

3 Depth information must be calculated in a fraction of a second for drawings of typical engineering components

4 Depth information should be based on as little prior processing of the drawing as possible

Depth information will be used to test hypotheses, so it should not presuppose these hypotheses if this can be avoided

Inflation to 2½D

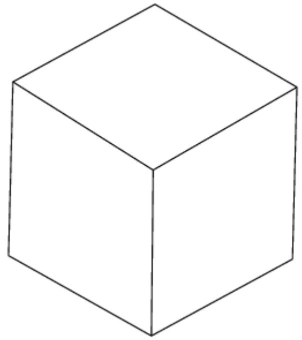
5 Depth information should be as good an interpretation of the drawing as is possible while achieving the other objectives

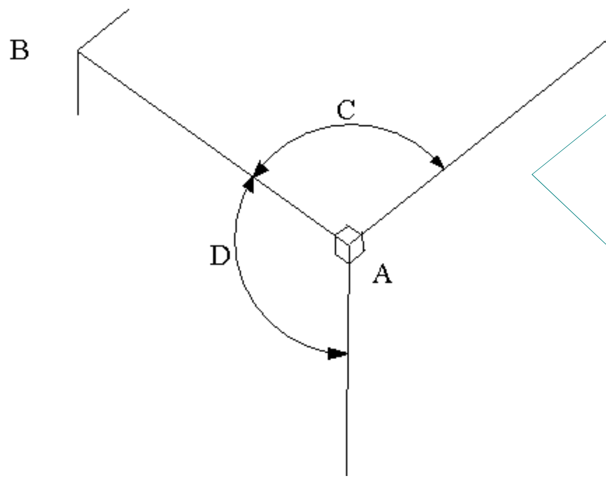
6 The results of inflation do not have to be perfect

We can add a beautification stage after completing the object topology

This will give us another chance to improve the geometry later

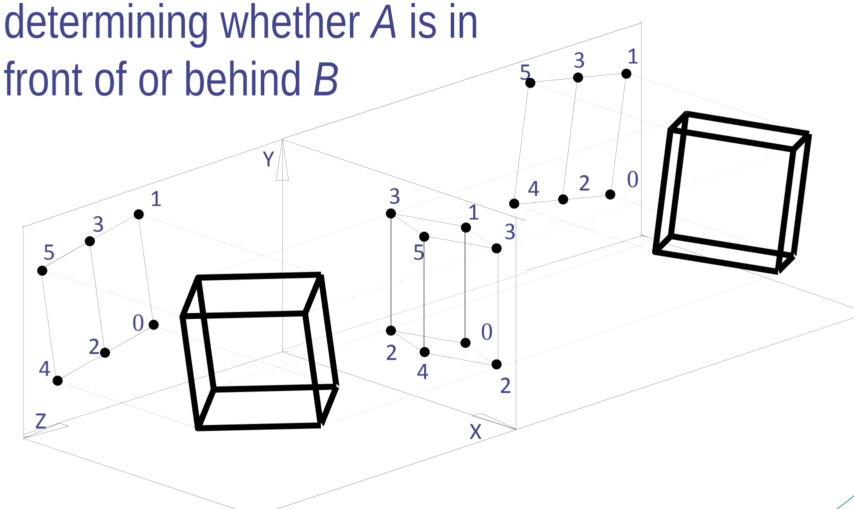
Cubic Corners





$$Z_B - Z_A = \pm AB\sqrt{(\tan C \tan D - 1)}$$

Note that there must be a separate mechanism for determining whether *A* is in front of or behind *B*



- If we have labelled the drawing, this often tells us whether *A* is in front of or behind *B* (e.g. the all-convex Y-junction, the central vertex is in front of the others)
- Even without labelling, we often have clues (e.g. boundary vertices are often behind internal vertices)

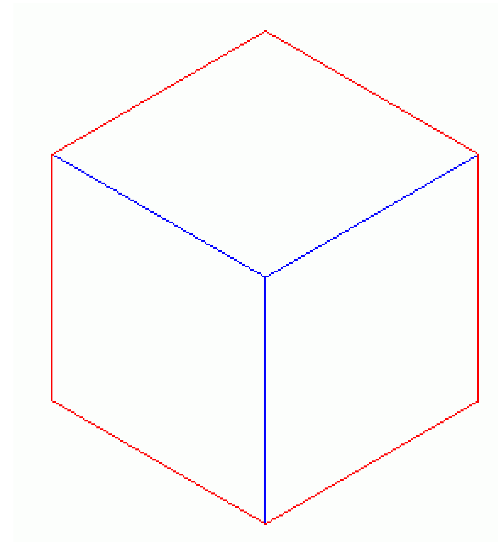
Inflation to 2½D

Junction Label Pairs:

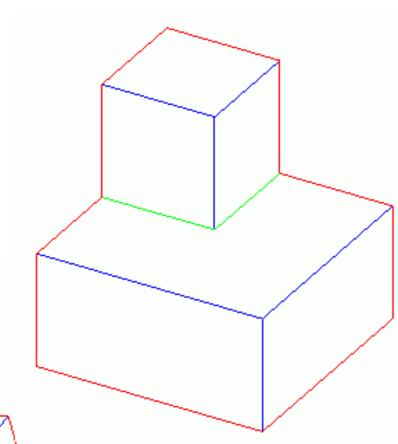
Consider **pairs** of connected junctions in the drawing

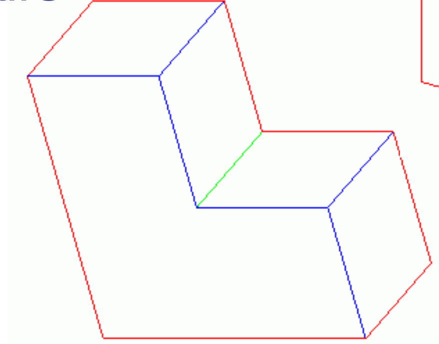
We can deduce, just from the line labels, which is the nearer

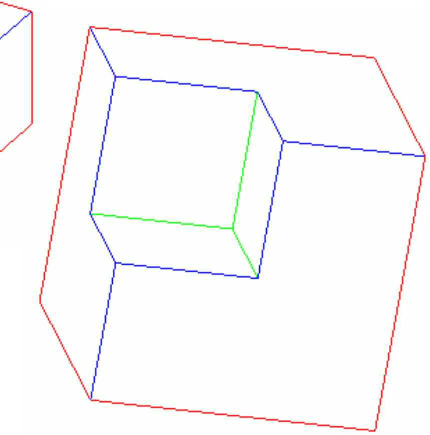
(and roughly by how much)



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

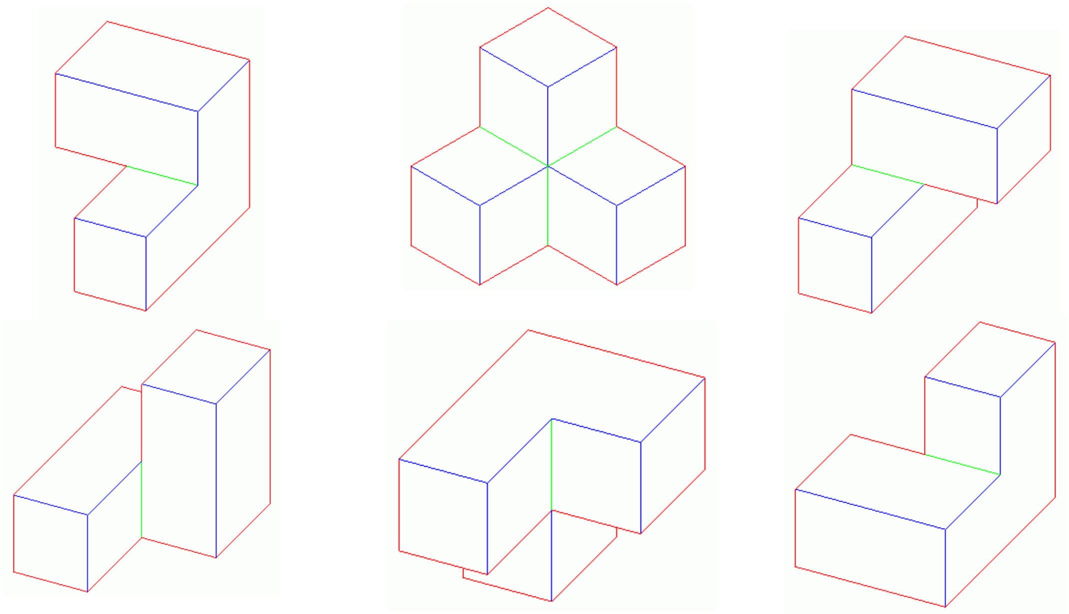






Inflation to 2½D

The extended trihedral solids give still more pairs

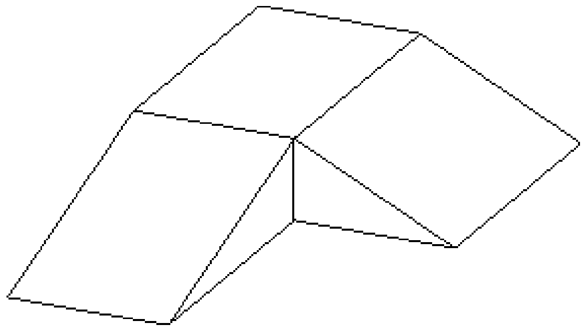


However, adding in the 4-hedral junctions (91 of them!) is impractical

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

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

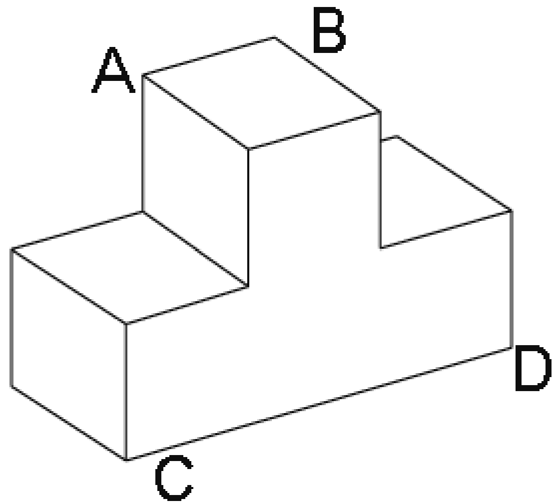
Perpendicularity: Introduction



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



$$n\, z_A - n\, z_B = m\, z_C - m\, z_D$$

Where *m* is the 2D length of line *AB* and *n* is the 2D length of line *CD*

√ Easily arranged into linear or explicit equations

√ Not inherently inflationary: the trivial solution *z=0* satisfies the equations

Inflation to 2½D

Face Planarity

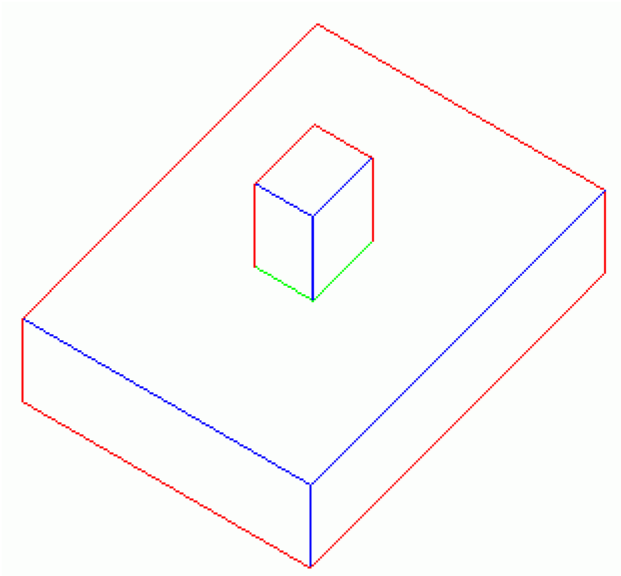
- ✓ Can be arranged into linear equations if we include face normals as well as vertex z-coordinates as the unknowns
- ✓ Quadrilateral faces can always be arranged into linear or explicit equations
- ✗ Larger (pentagonal and beyond) faces cannot be arranged into linear equations if the only unknowns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

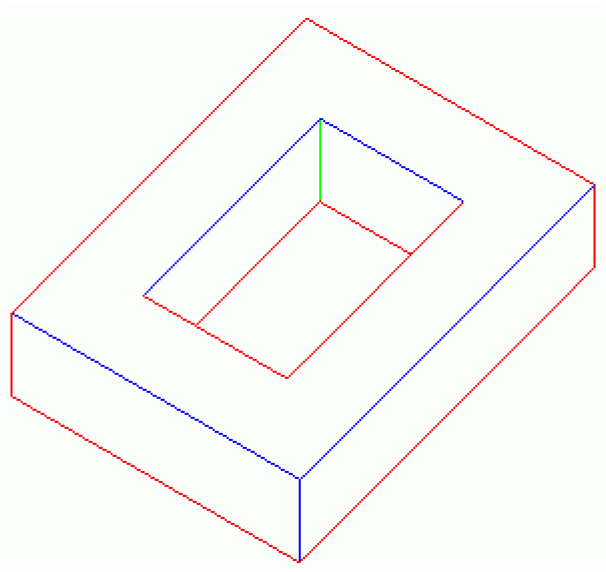
N.B. making groups of four vertices coplanar does not necessarily ensure that the entire face is planar

Face Planarity (continued)

- ✗ Not inherently inflationary: the trivial solution z=0 satisfies the equations

- √ Can be used to connect disjoint subgraphs





Inflation to 2½D

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

(More alternatives in Annex 7)

✗ Iterative approaches have also been tried

- ✗ they are slow
- ✗ there are no compensating advantages

Inflation to 2½D

The simplest and most effective approach is to use a linear system where the only unknowns are vertex *z*-coordinates

The question is then, what to include and what to leave out?

- √ Junction label pairs are good, but require a correct line labelling

  Junction label pairs on their own do not work for non-graph-connected drawings

- √ Cubic corners are nearly as good as JLP in most cases, but they do not distinguish +*z* from –*z*

  N.B. for cabinet projections cubic corners is dreadful

- √ Line parallelism is essential for good results

  for large drawings generating parallelism equations for each pair of parallel lines is too much

  it is better to generate one equation for each line, making it parallel to the ideal line of the bundle

- ✗ Face planarity is not generally recommended

  it does not help much, and makes badly-drawn drawings worse

  it is the best way to join unconnected graph segments, e.g. hole loops

Inflation: Summary

- √ Inflation using linear system of *z*-coordinates and a careful choice of compliance functions achieves its objectives

- √ This is the least problematic area of sketch interpretation

Particularly if we have reliable line labels to work with

Hidden Topology

What does the back of the object look like?

Two promising approaches, both iterative

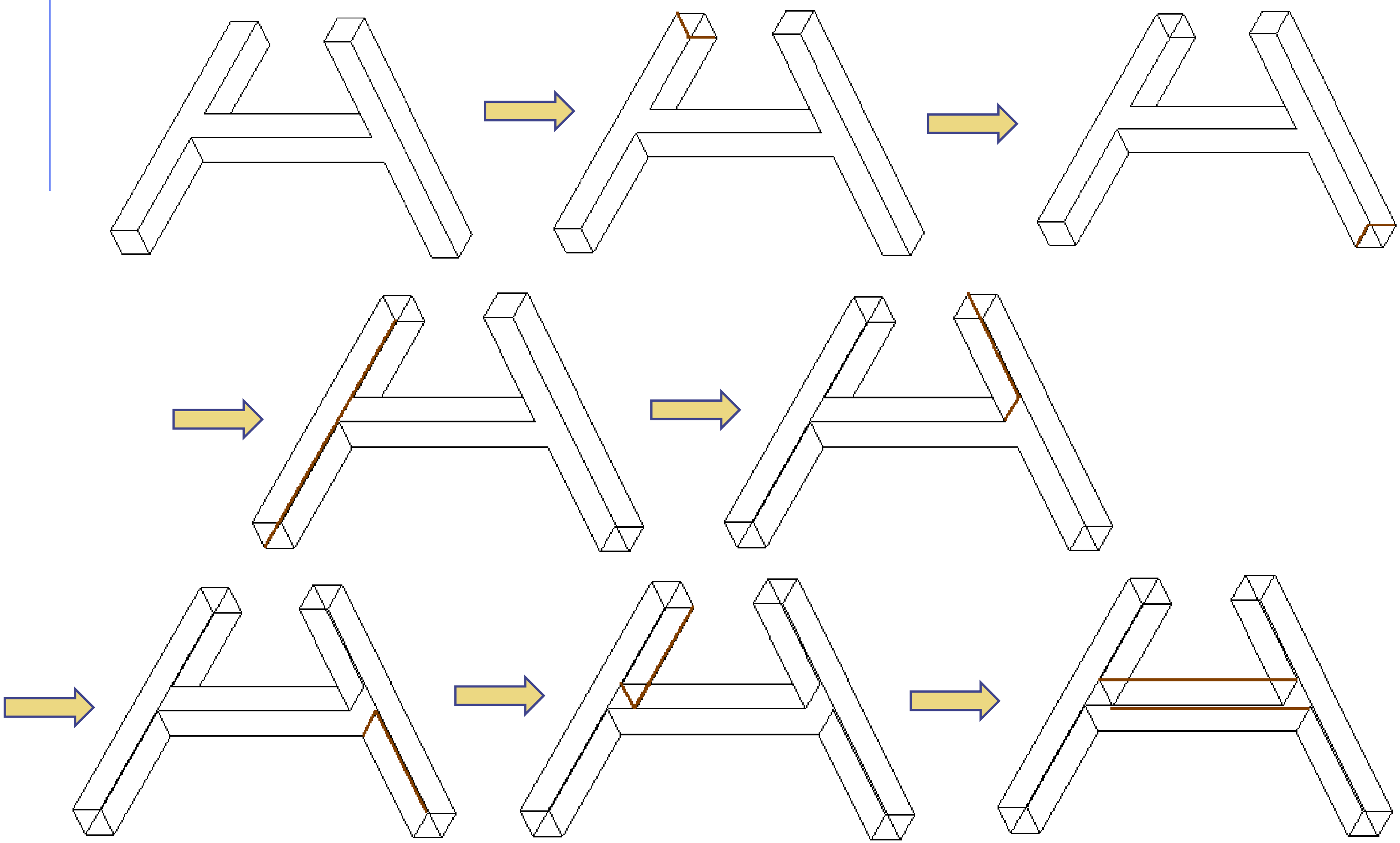
Recreate a complete wireframe

(We only need vertices and edges since we already know how to find faces)

Model the object as unions and intersections of extrusions

Hidden Topology

Recreate a complete wireframe, one or two edges at a time



Hidden Topology

This is a Greedy Approach

Sometimes it works, sometimes it does not

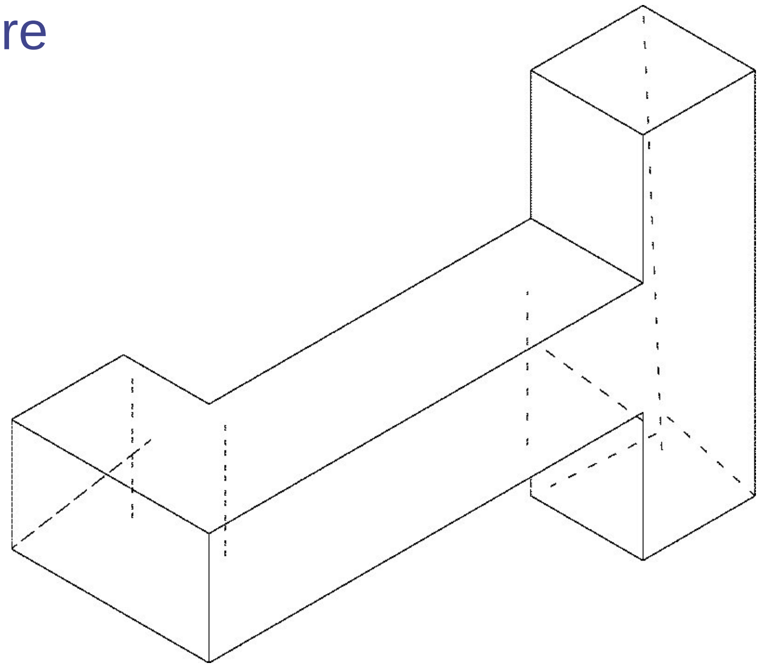
Main problems:

- ✗ Expanding it to a depth-first tree search does not help much
- ✗ When it goes wrong, the result is usually the wrong object, not an invalid object
- ✗ There is no trigger to invoke backtracking

Hidden Topology

Basic idea:

1. Extend lines in all major directions from all incomplete vertices
2. Note where the lines cross
3. Pick the best (using heuristics and probability theory)
4. Place a new vertex there
5. Add edges as required

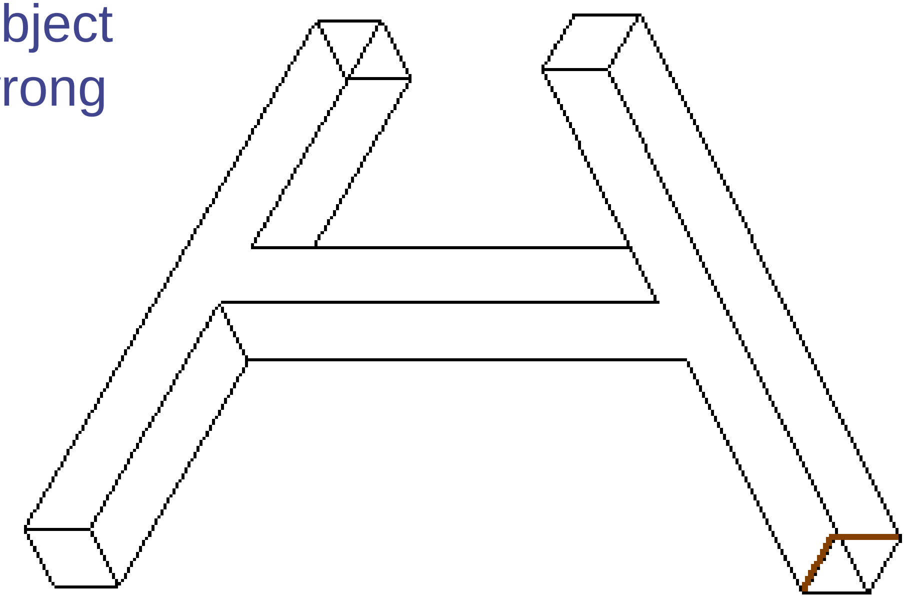


Hidden Topology

Refinement:

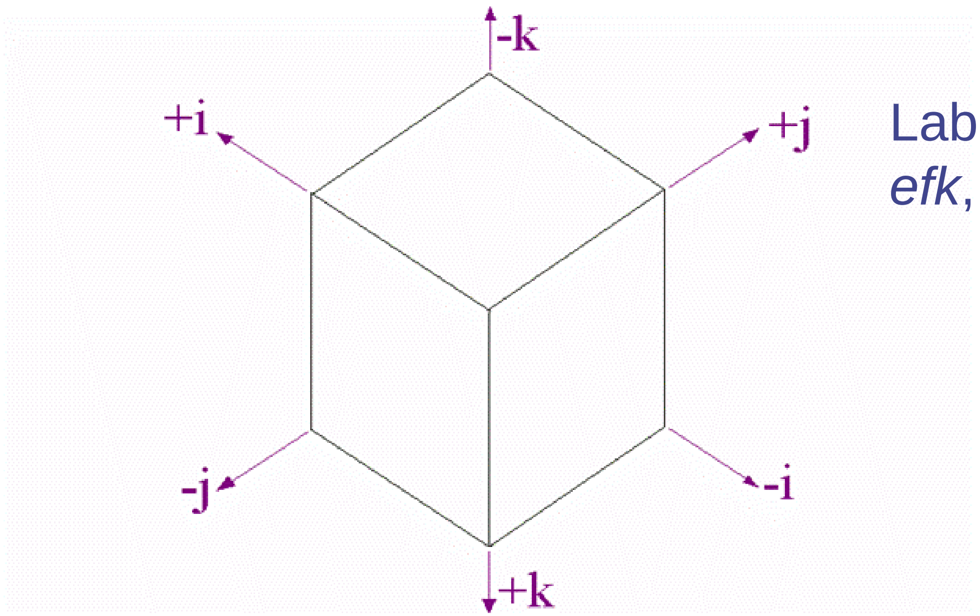
Any hypothesis which places a vertex outside the object silhouette is (probably) wrong

Any hypothesis which places (part of) an edge outside the object silhouette is (probably) wrong



Refinement: Neighbourhood matching

We can divide the space around each vertex into eight subspaces, using the three orthogonal places as half-space dividers



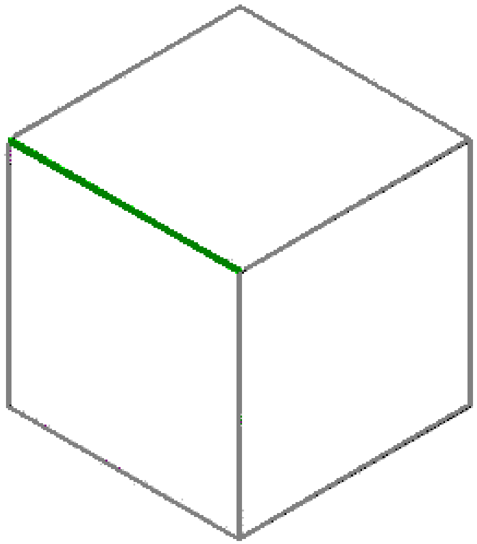
Label the eight subspaces *efg*, *efk*, *ejg*, *ejk*, *ifg*, *ifk*, *ijg* and *ijk*

efg is nearest the viewer

ijk is furthest from the viewer

Hidden Topology

Using the labelling, we can make deductions that some subspaces must be full and some subspaces must be empty



Given: the green line is *i-aligned* and *convex*

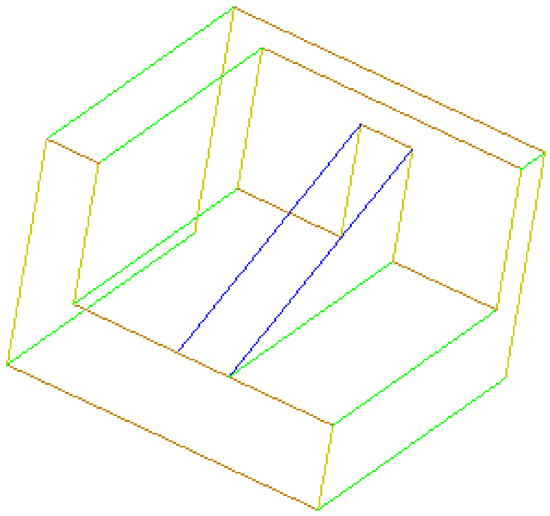
We can say that:
. One of the subspaces at the near vertex must be *full*
. Three of the subspaces at the near vertex must be *empty*
. One of the subspaces at the far vertex must be *full*
. Three of the subspaces at the far vertex must be *empty*

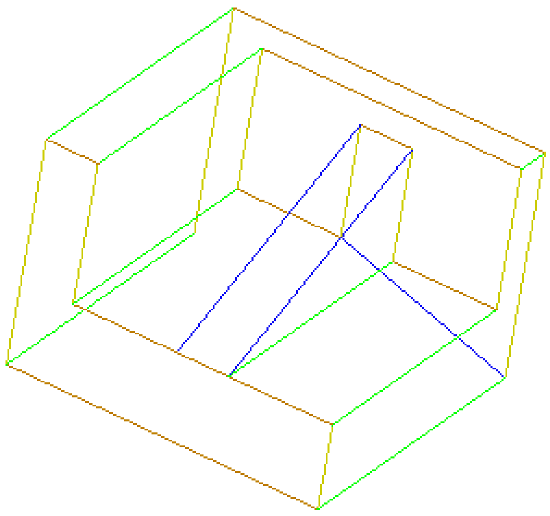
Note: at every visible vertex, the zone *efg* cannot be *full*
. this is the subspace which includes the line-of-sight

Hidden Topology

Subspaces belonging to two vertices (behind one and in front of the other) cannot be both full and empty

Any added edge which would create a subspace mismatch must be wrong





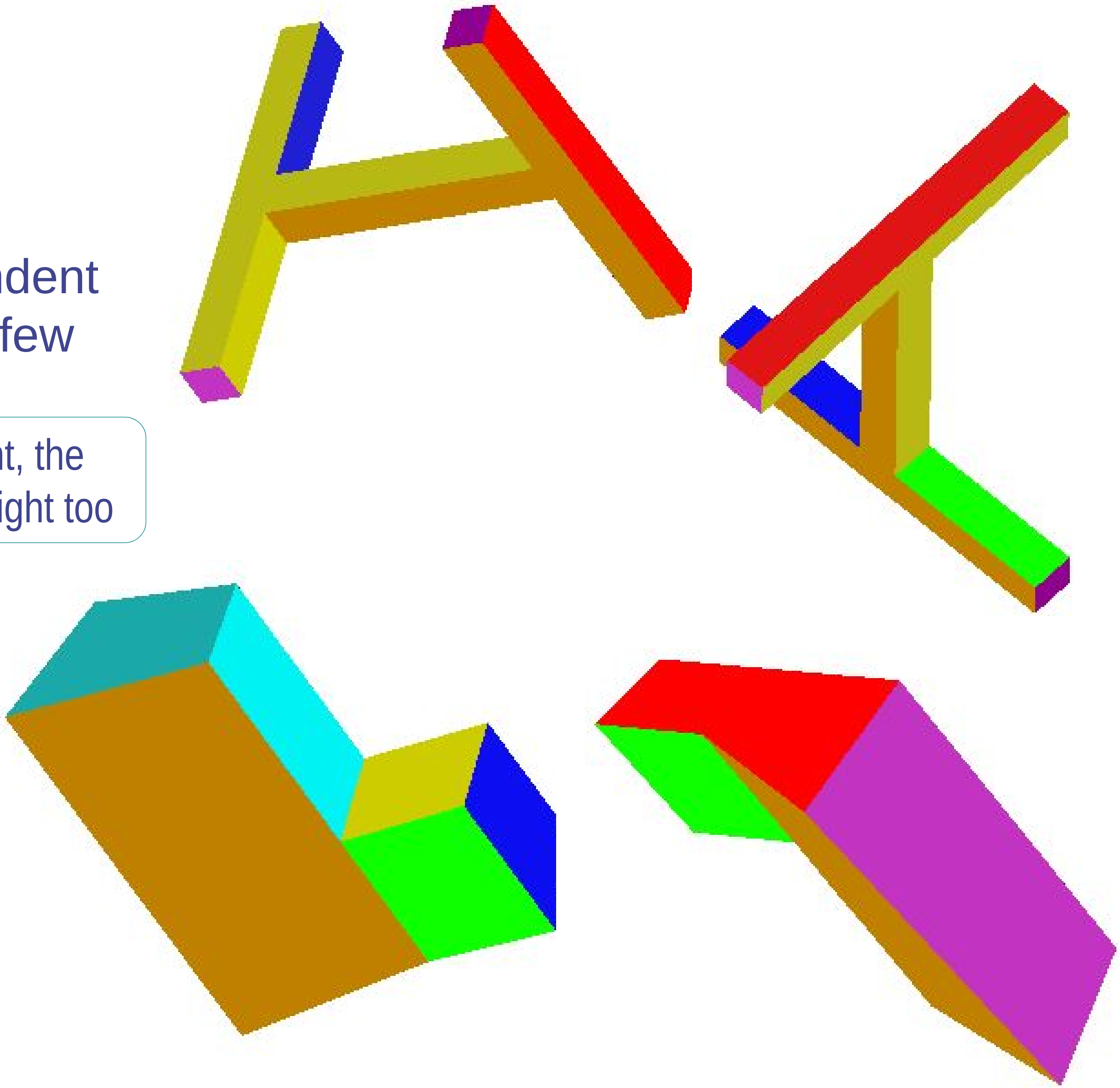
Hidden Topology

Results:

Mixed

Very dependent on the first few

if these are right, the rest is usually right too

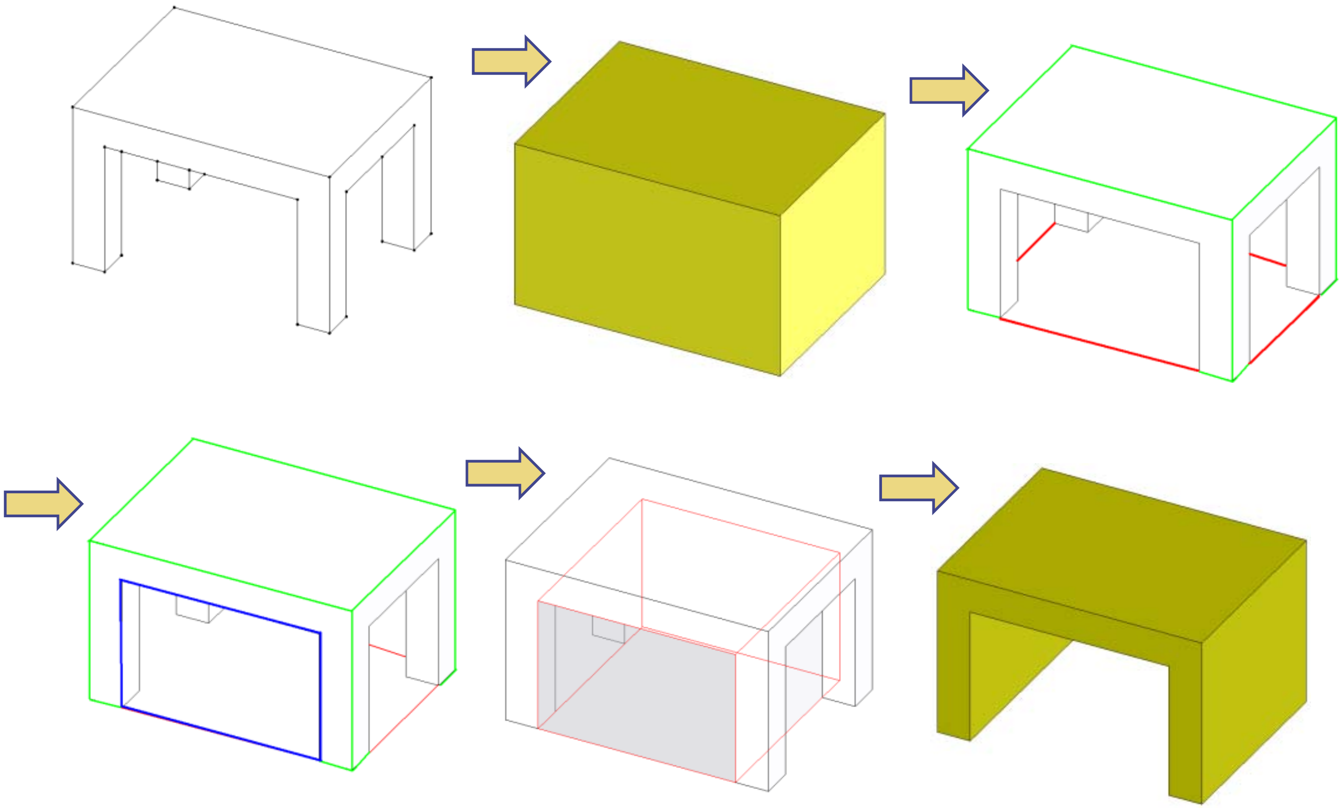


Hidden Topology

Unions and intersections of extrusions

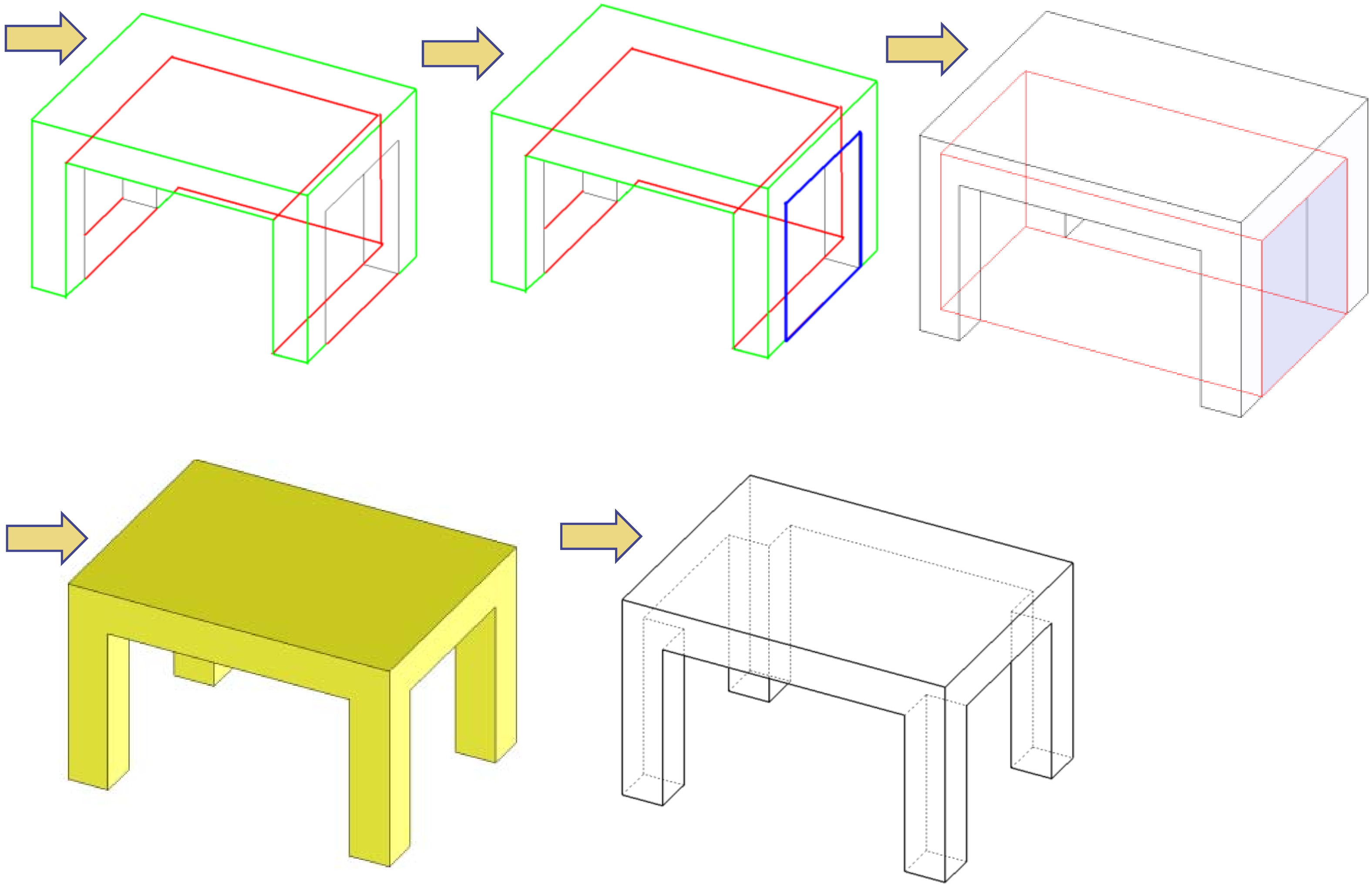
Most of this comes from: Y.S. Suh (2007) *Reconstructing 3D Feature-Based CAD Models By Recognizing Extrusions From A Single-View Drawing*, Proc. IDETC/CIE 2007)

Identify a *profile face* and extrude it



Hidden Topology

Continue identifying further *profile faces* until the entire object is modelled



Hidden Topology

A profile face is one which, when extruded along a major axis, explains some of the unidentified lines in the drawing

Some candidate profile faces are better than others and should be processed first:

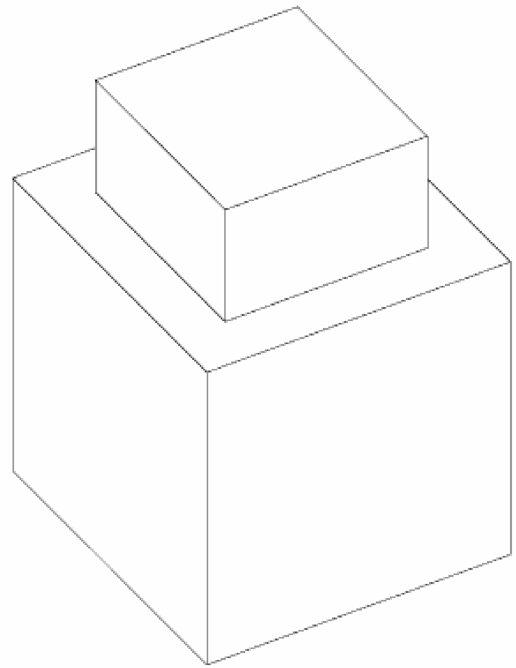
1. 2D area of the profile face: The larger the better
2. Number of profile face edges: The more the better
   (exception: triangles are better than quadrilaterals)
3. Number of extrusion lines (all in the same direction) leading away from the profile face: The more the better
4. 2D length of the extrusion lines: The longer the better
5. Number of points on profile face whose 3D positions are known: The more the better

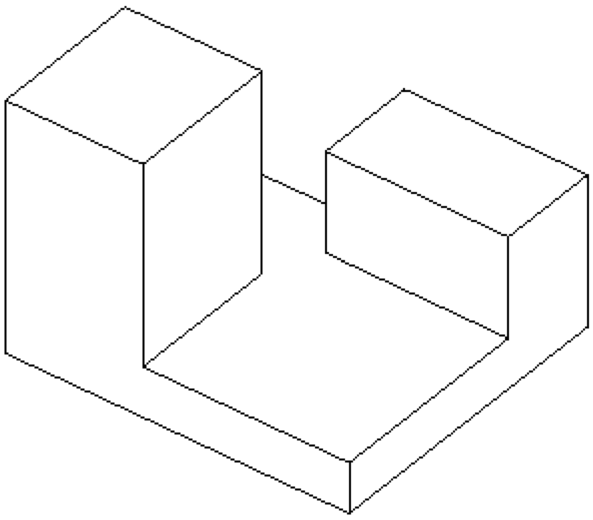
Subgraphs

Sometimes, if we break a sketch at T-junctions, we get two or more disjoint subgraphs

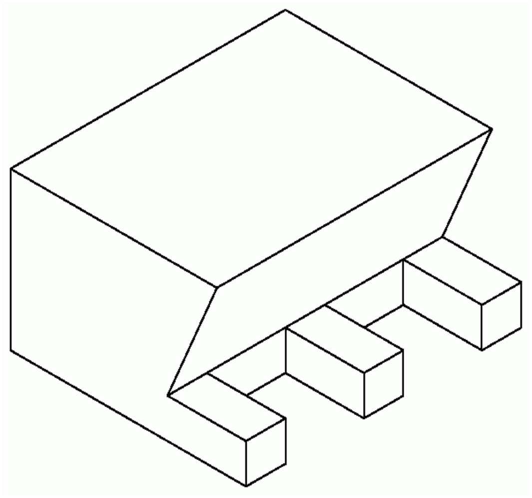
Each subgraph leads to a solid object

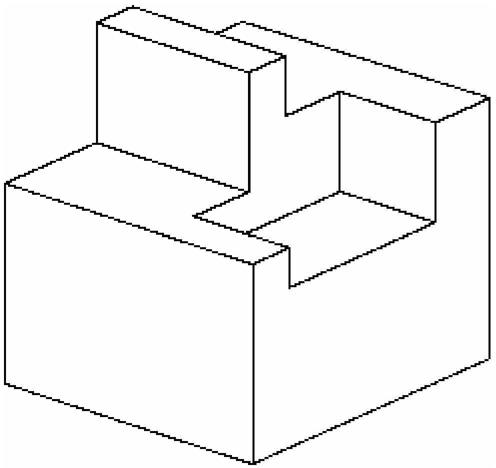
The subgraph which has the largest bounding box is treated as the base solid, and other solids from other subgraphs will be added to or subtracted from the base solid

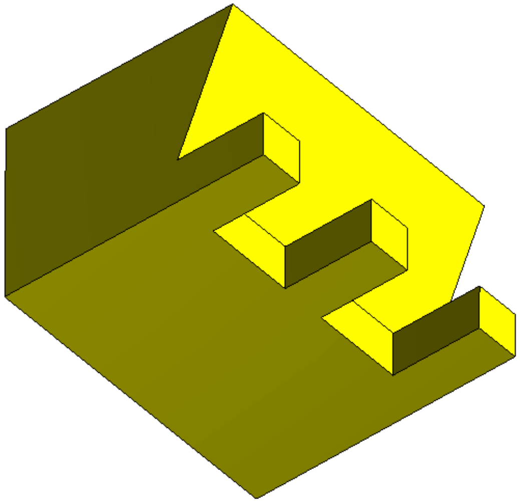


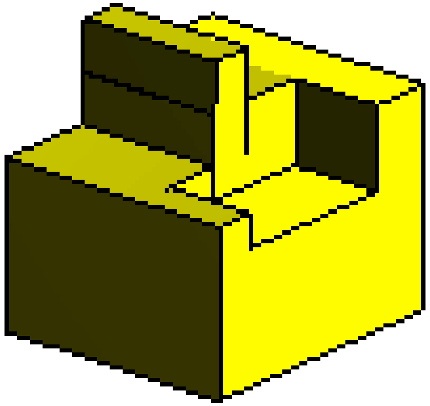


√ Results are generally good:



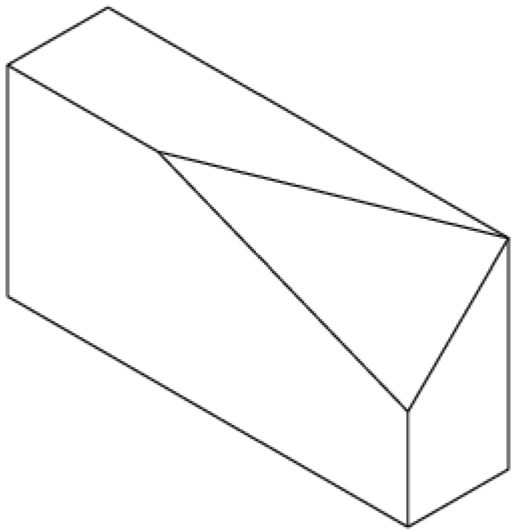




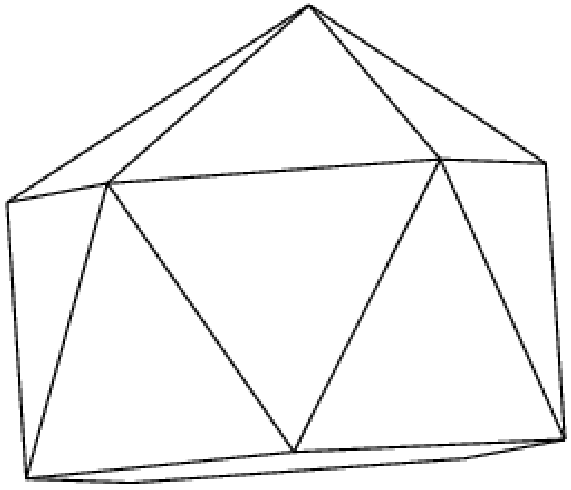


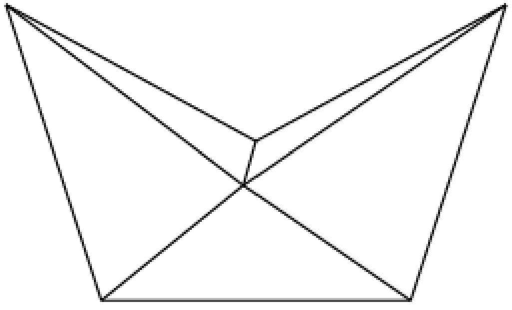
Hidden Topology

✗ But the method is limited to those objects which can be modelled as unions or intersections of axis-aligned extrusions



It cannot process these drawings:

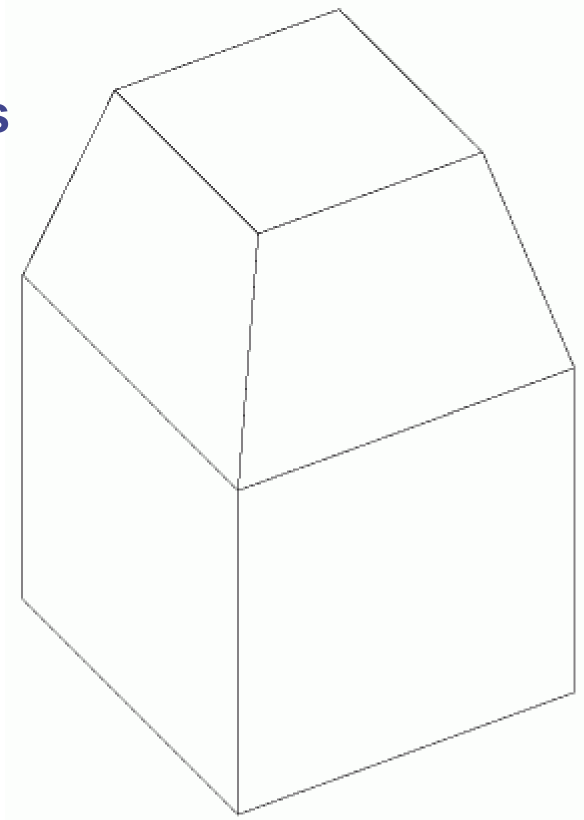


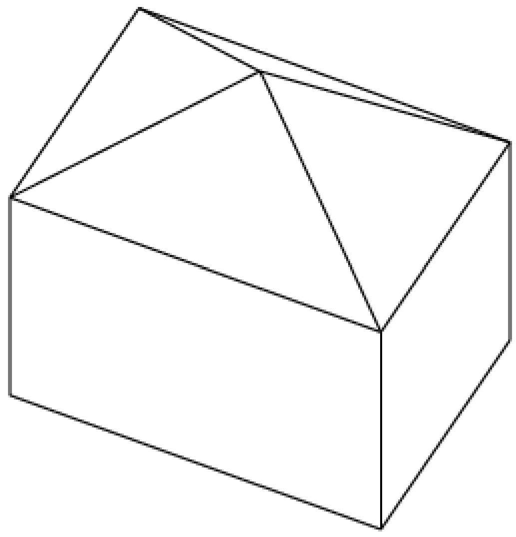


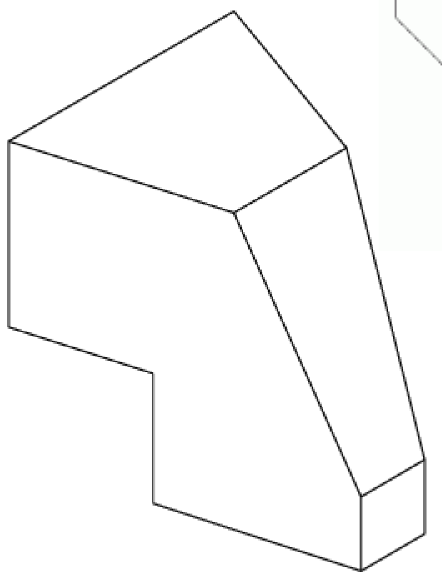
Hidden Topology

There are also a few objects which can be modelled as **unions or intersections** of axis-aligned extrusions but for which the algorithm does not work:

One or more necessary profile faces is not part of the object







Hidden Topology

If an object can be modelled as unions and intersections of extrusions, this method is usually more reliable

The alternative is creating a wireframe by projecting edges and locating their intersections

√ Is more flexible

It can, in principle, model any polyhedron

✗ But the greater flexibility gives it more opportunity to go wrong

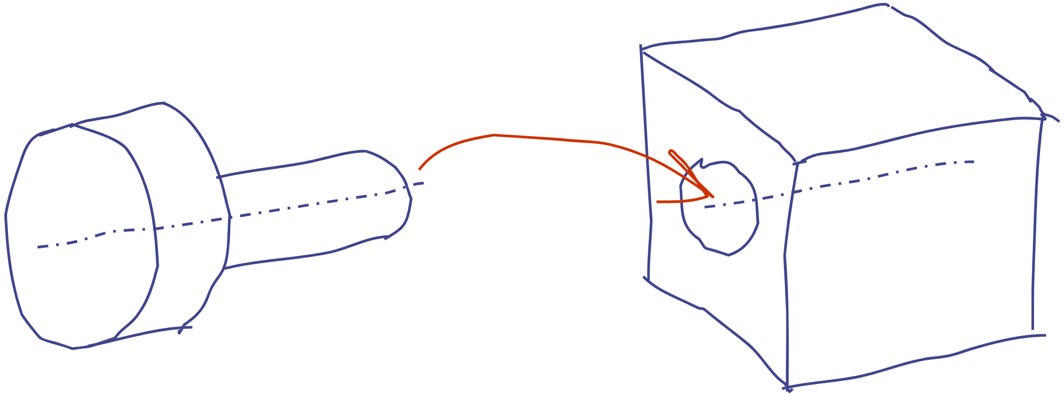
DEMO

RIBALD can be downloaded from:

http://pacvarley.110mb.com/RIBALD.html

Next session

Open problems



Starts ** time **