Finding Approximate Shape Regularities in Solid Models Bounded by Simple Surfaces

Frank C. Langbein
⟨F.C.Langbein@cs.cf.ac.uk⟩

Bruce I. Mills
⟨B.I.Mills@cs.cf.ac.uk⟩

David Marshall
⟨A.D.Marshall@cs.cf.ac.uk⟩

Ralph R. Martin
⟨R.R.Martin@cs.cf.ac.uk⟩

7th June 2001

Department of Computer Science
Cardiff University

CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

Reverse Engineering Solid Models

Data Acquisition

- Obtain 3D point clouds from a laser scanner
- Register multiple views

→

Segmentation

- Split the point cloud into subsets representing natu-ral surfaces

↓

Surface Fitting

- Find the surface type (plane, sphere, cylinder, cone, torus) of each subset
- Fit a surface of this type to the point set

←

Model Creation

- Create a solid model by stitching surfaces

Inaccurate Initial Model

- The initial model suffers from inaccuracies caused by
  - sensing errors
  - approximation and numerical errors
  - possible wear
  - manufacturing method
- **Goal:** Automatically reconstruct the *ideal* model with desired geometric regularities

Beautification

- Previous approaches:
  - ⋆ Augment the surface fitting step by constraint solving methods [Fisher,Benkő]
  - ⋆ Feature based approach [Thompson]:
    - ∗ manually identify features like slots and pockets
    - ∗ use them to drive the segmentation and surface fitting
- **Our approach:**

  Improve the model in a post–processing step called **beautification**

Beautification Strategy

Analyser

Detect potential regularities which are approximately present in the initial model

→

Hypothesizer

Select a maximal, consistent subset of likely constraints

↓ ↑

Constraint Solver

Solve a weighted constraint system using an optimization technique (quasi–Newton methods on least squares error)

←

Reconstruction

Reconstruct an improved model, fix topological problems, align the model with the coordinate axes

Geometric Regularities

- *Global* regularities: approximate symmetries
- *Local* regularities:
  - ⋆ Extract properties of B–rep model elements (faces, loops, edges, vertices) as typed feature objects
  - ⋆ Find similar feature objects of the same type by creating a hierarchical clustering structure
  - ⋆ Represent each cluster by an average feature object
  - ⋆ Find special feature objects similar to the average feature objects
  - ⋆ **Example:** Find similar cylinder radii and a special value like an integer close to the average radius

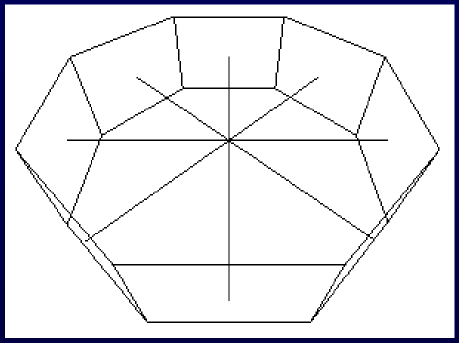
Local Geometric Regularities

Parameter

- Equal lengths
- Equal angles
- Special values:
  - integers
  - simple fractions
- Simple integer relations

Directions

- Parallel directions
- Directions with same angle relative to a special direction
- Symmetrical arrangements of directions



Axes

- Axis intersections
- Aligned axes
- Regularly positioned axes

Positions

- Equal positions
- Positions equal under projection
- Regularly arranged positions

Angle and Length Parameters

- Cluster angles and lengths separately with angular and length tolerances, to find **parameters with similar values**

Element	Parameter	Type
sphere	radius	length
cylinder	radius	length
cone	semi–angle	angle
torus	major radius	length
	minor radius	length
straight edge	distance between end points	length
circular edge	radius	length
	angle of circle segment	angle

Special Parameter Values

- Find a **special value** close to the average parameter value for a cluster:
  - ⋆ Lengths: $x = \frac{m}{n} K_l$ for length base units K_l like $1.0, 0.1, 2.54, \ldots$
  - ⋆ Angles: $x = \frac{m}{n} K_a$ for angle base units K_a like $\pi, \frac{\pi}{180}, \ldots$

    $x = \arctan\left(\frac{m}{n}\right)$

    where $m, n \in \mathbb{N}$
  - ⋆ **Special ratios** between parameters of the same type
- Basic algorithm:
  Find simple fractions $\frac{m}{n}$ approximating x with a tolerance t and $n < M$

Finding Simple Fractions

I. Find the closest integer a_0 to x

II. Find fractions for the remainder $x_0 = |a_0 - x|$ recursively; on recursion level l:

- ⋆ Let m/n be the fraction found so far with error x_l
- ⋆ For $b = 1, 2, \ldots$ approximate x_l by fractions b/a with $a = \text{round}\,(b/x_l)$ and $a \leq M$
- ⋆ Add each b/a to m/n and if the result has not been found before:
  - ∗ If the new fraction is close enough to x, report it
  - ∗ If the new error is still too large, call the algorithm recursively on the new remainder with new limit $M = M_0 M$

Parallel Directions

- **Direction:** a point on the unit sphere with antipodal points identified (projective plane)

plane	normal	straight edge	direction
cylinder	axis direction	circular edge	normal of circle plane
cone	axis direction	elliptical edge	normal of ellipse plane
torus	axis direction		

- Find **parallel directions** by clustering the projective points with

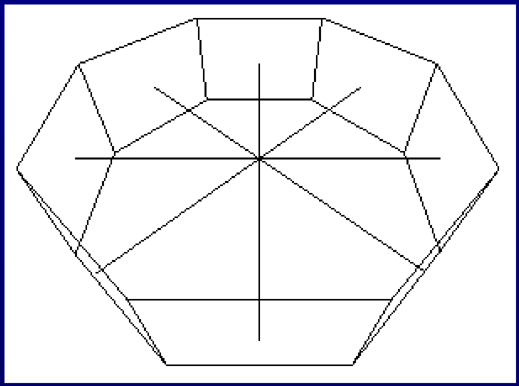
$$\angle(d_1, d_2) = \arccos\left(\left|{d_1}^t d_2\right|\right)$$

Directions in a Plane

- Find **directions in a plane**: directions on a great circle of the unit sphere
  - ⋆ For each pair of parallel direction clusters generate a plane normal
  - ⋆ Cluster the plane normals in the same way as the parallel directions

Angle–Regular Directions

- Directions in a plane might be arranged symmetrically



planar angle–regular

- For directions $\{d_j\}$ in a plane:

$$\angle(d_j, d_k) \approx m\frac{\pi}{n}, \quad m, n \in \mathbb{N}$$

with $n < \dfrac{\pi}{2t_{\text{angular}}}$

Angle–Regular Algorithm

Try all arrangements suggested by the angles:

I. Compute all angles between the directions and for each angle find base angle candidates π/n within t_{angle}

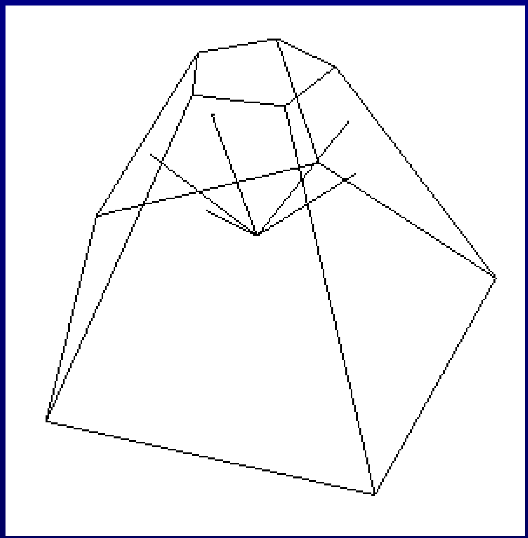
II. For each direction and its associated base angles β:

1. Try to find a planar angle–regular direction subset by checking if the angles are approximate multiples of β
2. If the direction subset is regular, accept the subset and remove base angles generating the same set

   Regular subset: – all angle multiples
   – at least every second multiple
   – at least three consecutive directions

Conical Direction Arrangements

- Directions on a small circle of the unit sphere are **directions on a cone**:
  - ⋆ Combine each triple of linearly independent parallel direction clusters to a direction cone
  - ⋆ Handle the cone directions similar to the planar case



Axes

- Find **aligned surface axes:**
  - Project positions of approximately parallel axes onto the plane through the origin
  - Cluster them
- Find **axis intersections:**
  - Compute the approximate intersection points of non–parallel axes (the centre of the shortest line between each corresponding pair)
  - Cluster them

Regular Axis Arrangements

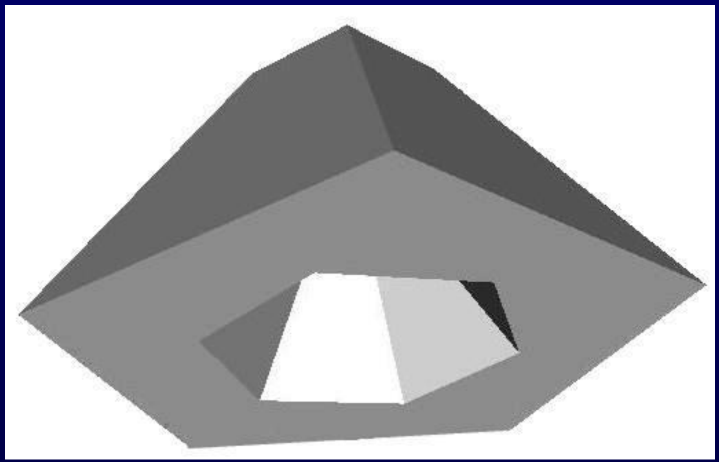
- Find **axes with equal distances on a line or a 2D grid**:
  - Project positions of parallel axes in a plane
  - Cluster all lines between pairs of the projected positions to find axis positions approximately on a line
  - Find distance–regular arrangements on these lines (similar to angle–regular arrangements)
  - If possible combine distance–regular arrangements on the lines to 2D grids

Positions

- Positions corresponding to vertices and surface root points:
  - Cluster the positions to find **equal positions**
  - Project the positions onto special planes and lines through the origin (obtained from orthogonal systems or main axes)
  - Cluster the projections to find **partially equal positions**
- Similar to regularly arranged axes, find **regularly arranged positions**

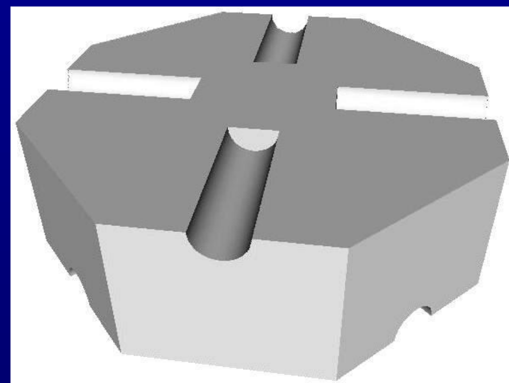
Example

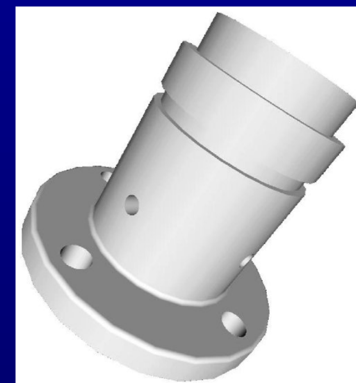
- Preliminary experiments with objects reverse engineered from simulated 3D point clouds (perturbed by 3 degrees, 0.3 length units; tolerances for 5 degrees, 0.5 length units)



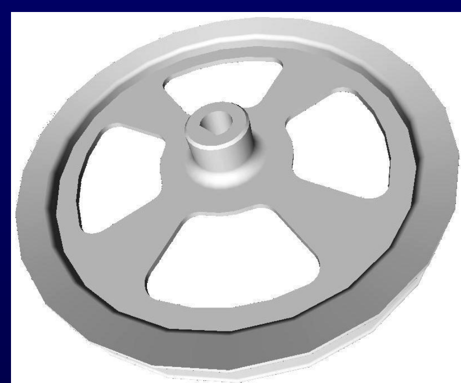
- Desired regularities found (41)
  - ⋆ conical angle regularities
  - ⋆ axis intersection points
  - ⋆ special edge lengths
- Unwanted regularities (11)
  - ⋆ parallel planes
  - ⋆ more conical angle regularities
- Missed regularities: none

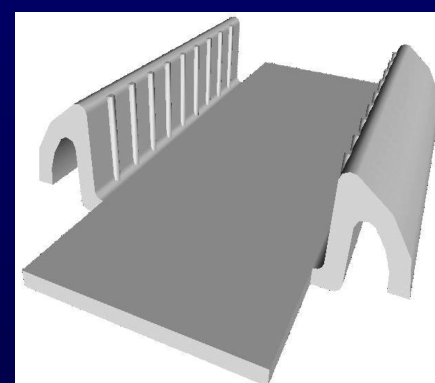
More Examples





desired:	33	20
unwanted:	0	2
missed:	0	0





desired:	76	108
unwanted:	21	27
missed:	7	5

Results

- Choosing small tolerance values results in a few, very likely regularities, but many desired regularities are missed
- Increasing the tolerance values adds the missing regularities, but also increases the likelihood of finding unwanted regularities
- For simple models there is a tolerance level which distinguishes exactly between wanted and unwanted regularities
- For more complicated models unwanted regularities can be minimized, but not avoided

Conclusions

- The presented methods find geometric regularities suitable for beautification
- Subsequent beautification steps must select an appropriate subset of regularities to generate consistent constraint systems
- The number of tolerance values used in the algorithms can be reduced:
  - ⋆ Automatically detect large tolerance jumps in the hierarchical clustering structure
  - ⋆ Add consistency checks, e.g. the intersection of n axes should be a cluster of $n(n-1)/2$ intersections of axis pairs