Recognizing Geometric Regularities for Beautification of Reconstructed Solid Models

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⟩

June 2001

Department of Computer Science
Cardiff University

Reverse Engineering

- Engineering converts a concept into an artifact
- Reverse engineering converts an artifact into a concept
- The desired result is a representation of the design intent, not a simple copy

Goal: Reconstruct an *ideal* model of a physical object with intended geometric regularities

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
- Geometric regularities have to be enforced at some stage of the reconstruction process to guarantee their presence

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

$\longrightarrow$

Hypothesizer

Select a maximal, consistent subset of likely constraints

$\downarrow \quad \uparrow$

Constraint Solver

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

$\longleftarrow$

Reconstruction

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

Geometric Regularities

- Use similarity to recognize geometric regularities approximately present
- *Global* similarities: approximate symmetries
- *Local* similarities:
  - ⋆ 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

Local Geometric Regularities

Parameter

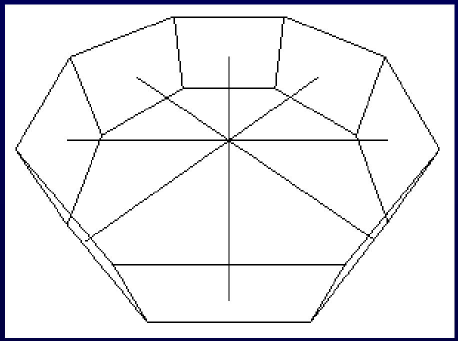
- Equal lengths
- Equal angles
- Special values:
  – integers
  – simple fractions

Loops

- Equal shaped polygons, independent of scaling

Directions

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



Axes

- Axis intersections
- Aligned axes

Positions

- Equal positions
- Positions equal under projection

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 $m, n \in \mathbb{N}$
- 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:

- $\star$ Let m/n be the fraction found so far with error x_l
- $\star$ For $b = 1, 2, \ldots$ approximate x_l by fractions b/a with $a = \text{round}\,(b/x_l)$ and $a \leq M$
- $\star$ Add each b/a to m/n and if it 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 $M = M_0 M$

Simple Fractions Example

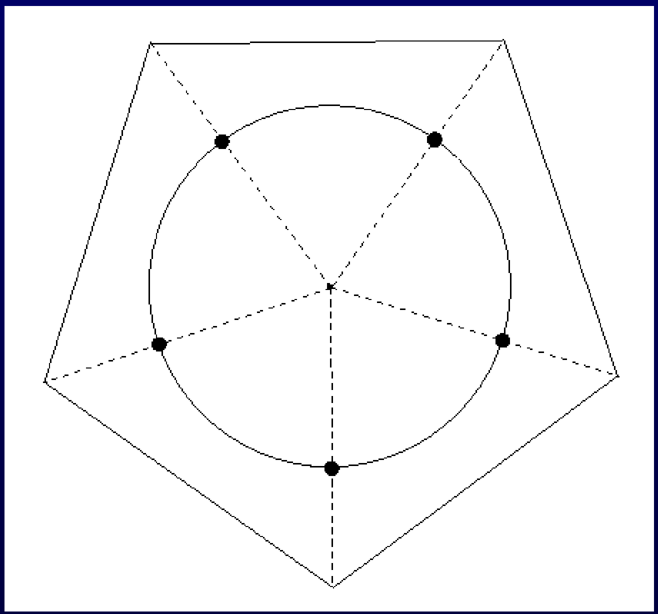
Example: $x = 0.63$ with $t = 0.05$, $M_0 = 5$ and $t_{\text{min}} = 0.01$

$$
\begin{aligned}
0.63 \quad &= 1/2 && + \quad 0.13 \\
& && \quad = 1/8 \qquad\qquad + \quad 0.005 \\
& && \qquad [\rightarrow \mathbf{5/8}] \\
& && \quad = 2/15 \qquad\quad - \quad 0.003333 \\
& && \qquad [\rightarrow \mathbf{19/30}] \\
& && \quad = 3/23 \qquad\quad - \quad 0.000435 \\
& && \qquad [\rightarrow \mathbf{29/46}] \\
&= 2/3 && - \quad 0.036667 \\
& \quad [\rightarrow \mathbf{2/3}] \\
&= 3/5 && + \quad 0.03 \\
& \quad [\rightarrow \mathbf{3/5}]
\end{aligned}
$$

Polygon Representation

- Find **similar polygons** independent of scaling
- Represent the polygon as a function on the unit circle $\mathbb{T}$:

$$f = \sum_{k=0}^{m-1} \alpha_k \delta_k$$



m : number of vertices

α_k : angle at k-th vertex

l_k : length of the line segments from vertex 0 to k

δ_k : Dirac distribution at $2\pi \frac{l_k}{l_m}$ on $\mathbb{T}$

Similar Polygons

- Compute some (~ 10) Fourier coefficients of f:

$$u_j = \frac{1}{2\pi}\langle f, \exp(-ij\cdot)\rangle$$

- Cluster the Fourier coefficient vectors using the similarity measure

$$\delta(u,v) = \sum_{j=1}^{d} \big|\, |u_j| - |v_j| \,\big|$$

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

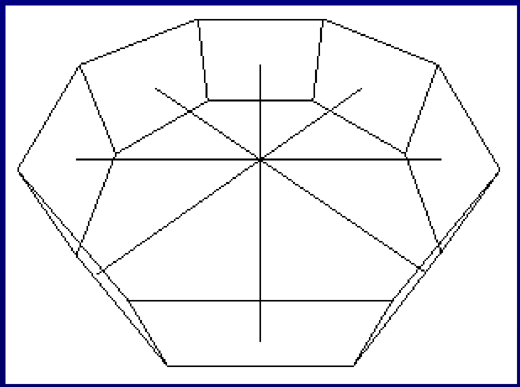
$$\angle(d_1, d_2) = \arccos\left(|{d_1}^t d_2|\right)$$

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
  - ⋆ The resulting clusters represent directions in a plane

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:

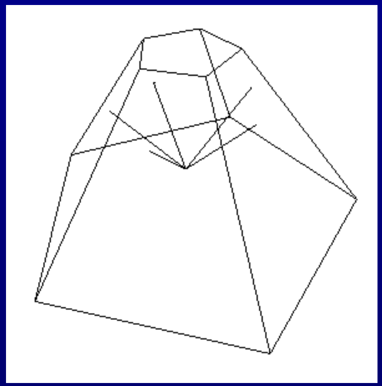
I. Compute all angles between the directions and for each angle find base angle candidates $\frac{\pi}{n}$ within t_{angle}

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



- Similar to the planar case handle directions on a small circle of the unit sphere (**directions on a cone**)
- Combine each triple of linearly independent parallel direction clusters to a direction cone; cluster them
- Detect **conical angle–regular** directions:
  - ⋆ Project directions in plane defined by the cone axis
  - ⋆ Search for planar angle–regular arrangements with base angles $\frac{2\pi}{n}$
- An orthogonal system is a special conical angle regularity

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

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**

Finite Symmetry Groups

- Finite symmetry groups of the model are determined by finitely many isometries mapping the model onto itself
- Our approach:
  - ⋆ Find symmetries of the model as point set symmetries
  - ⋆ Detect isometries as permutations which preserve the distances; the geometric realization becomes secondary
  - ⋆ Automatically choose natural tolerances reducing local ambiguity instead of finding symmetry for a given tolerance

Point Set Symmetries

- A symmetry of the model is a symmetry of a point set derived from the model (vertices, centres of spheres, tori, apices of cones)
- There is typically a point set with the same symmetries as the model
- The point set could have more, but not less symmetries
- Add a post–processing step to check if the point set symmetries also preserve geometry types and combinatorial information

Permutations

- An approximate isometry of a point set is a permutation preserving the distances between the points approximately
- The permutations are the leaves of a tree of partial injections:
  - ⋆ A partial injection is a list of point pairs where each point appears at most once as first and at most once as second element of the pairs
  - ⋆ The root of the tree is the empty list
  - ⋆ The children of a partial injection are obtained by adding one more point pair to the list

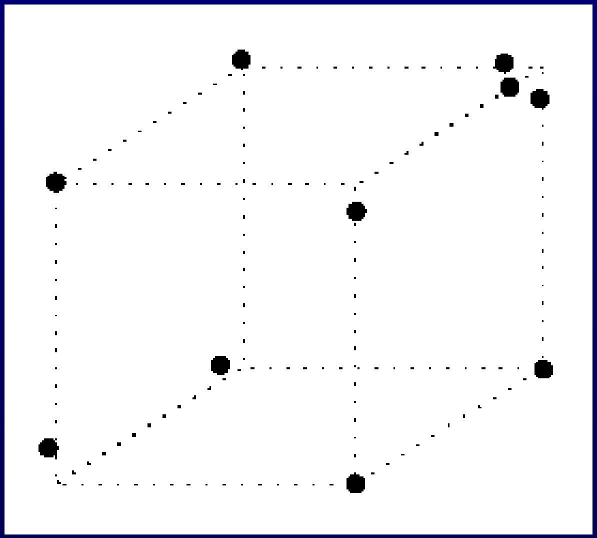
Symmetry Detection Algorithm

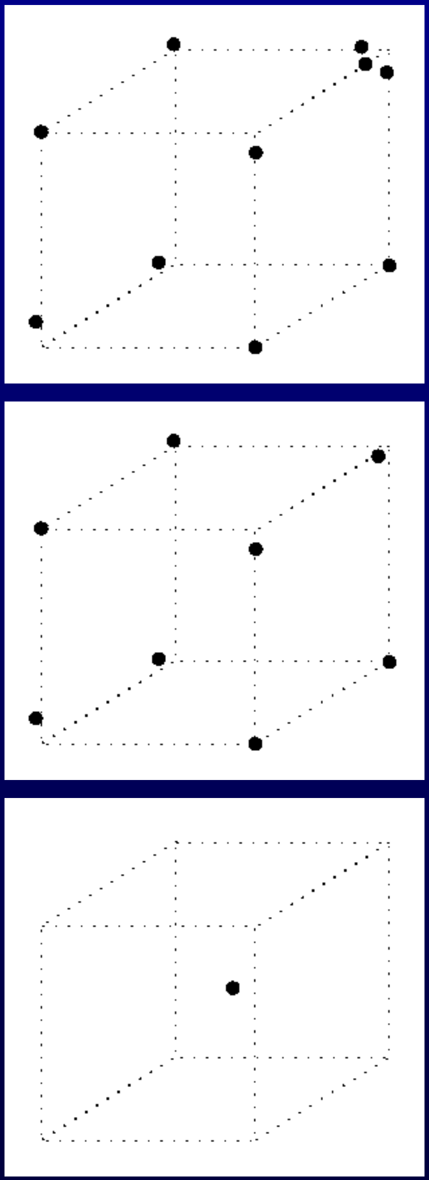
Approximate symmetry detection for point sets:

I. Create consistent clusterings of the points at different tolerance levels:
   - Each point belongs to exactly one cluster
   - All distances between the points in a cluster are smaller than the tolerance
   - Distances between points from different clusters are larger than the tolerance

II. For each consistent clustering, search the tree of partial injections to find valid isometries

Example for Consistent Clusterings





Stage II: Symmetry Analysis 1

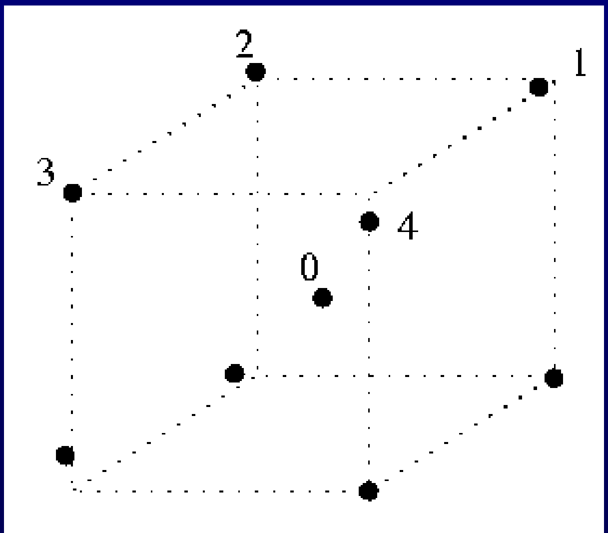
Detect distance–preserving permutations of the clustered point set

1. Find a large, non–degenerate tetrahedron whose vertices are
   - ⋆ the centroid of the clustered point set
   - ⋆ three points on the convex hull of the clustered point set chosen to be as far apart as possible from each other
     - ∗ The three points are found by maximizing the distance, the area and finally the volume

Stage II: Symmetry Analysis 2

2. Do a limited depth–first search over the tree of partial injections mapping the points of the tetrahedron:
   - ⋆ The centroid always has to be mapped onto itself
   - ⋆ Backtrack to the parent whenever the newly added point pair induces an isometry which does not approximately preserve the distances between the points
   - ⋆ Once three points are mapped, the fourth point can only be mapped to two possible locations
   - ⋆ All subsequent points are mapped to one location, thus check the remaining distances directly

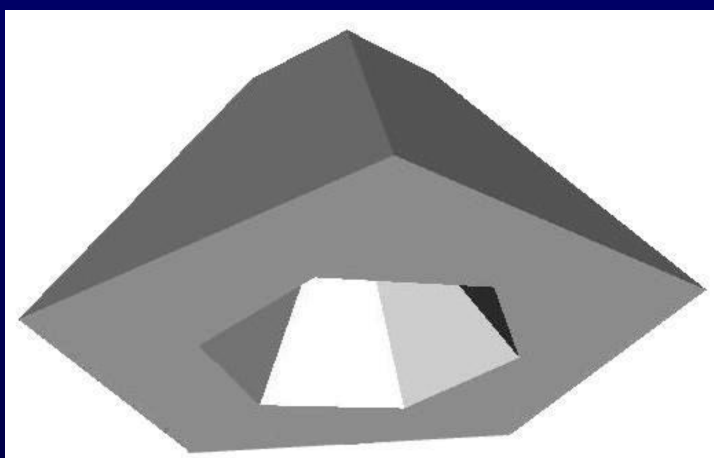
Symmetry Analysis Example



- Select tetrahedron: $0, 1, 2, 3$
- Map the centroid: $0 \longrightarrow 0$
- Map $1 \longrightarrow 2$
  - ⋆ Distance check: $d(0,1) \approx d(0,2)$
- Map $2 \longrightarrow 4$
  - ⋆ Distance check: $d(0,2) \approx d(0,4)$
  - ⋆ Distance check: $d(1,2) \not\approx d(2,4)$
- Backtrack and map $2 \longrightarrow 3$
  - ⋆ Distance check: $d(0,2) \approx d(0,3)$
  - ⋆ Distance check: $d(1,2) \approx d(2,3)$

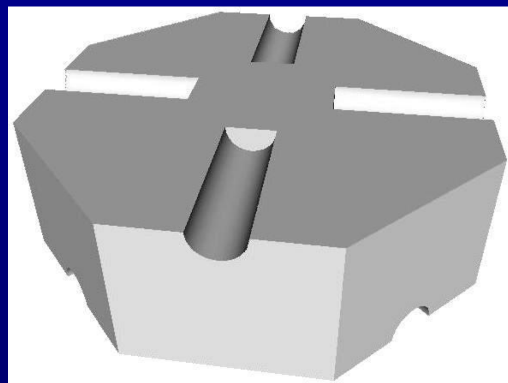
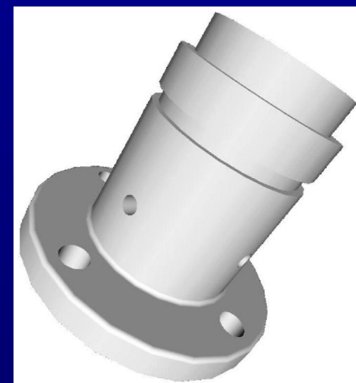
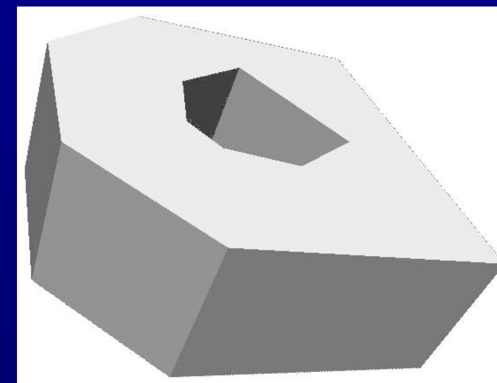
Experiments

- 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)

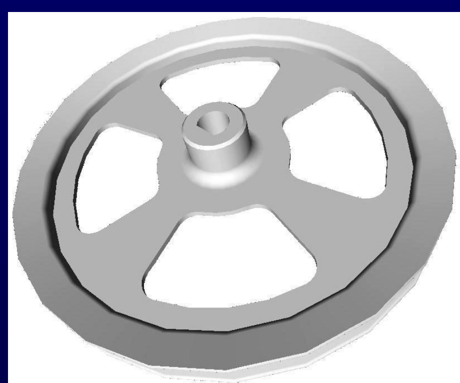
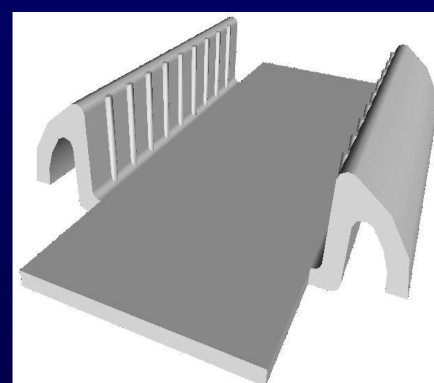
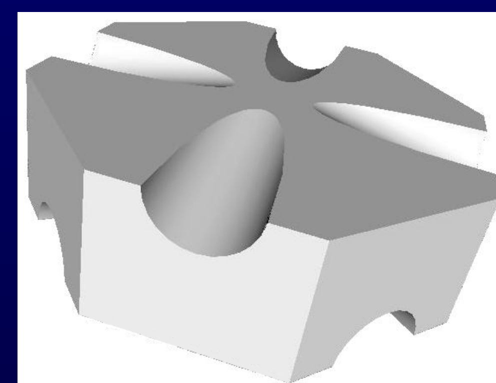


- Desired regularities found (41)
  - ⋆ conical angle regularities
  - ⋆ axis intersection points
  - ⋆ special edge lengths
- Unwanted regularities (11)
  - ⋆ parallel planes
  - ⋆ more conical angle regularities
- Missed regularities: none

More Examples

desired:	33	20	44
unwanted:	0	2	8
missed:	0	0	3

desired:	76	108	62
unwanted:	21	27	9
missed:	7	5	7

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 constraints
- 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