Recognizing Geometric Patterns for Beautification of Reconstructed Solid Models

International Conference on Shape Modelling and its Applications
Genova, Italy, 7 – 11 May 2001

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

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

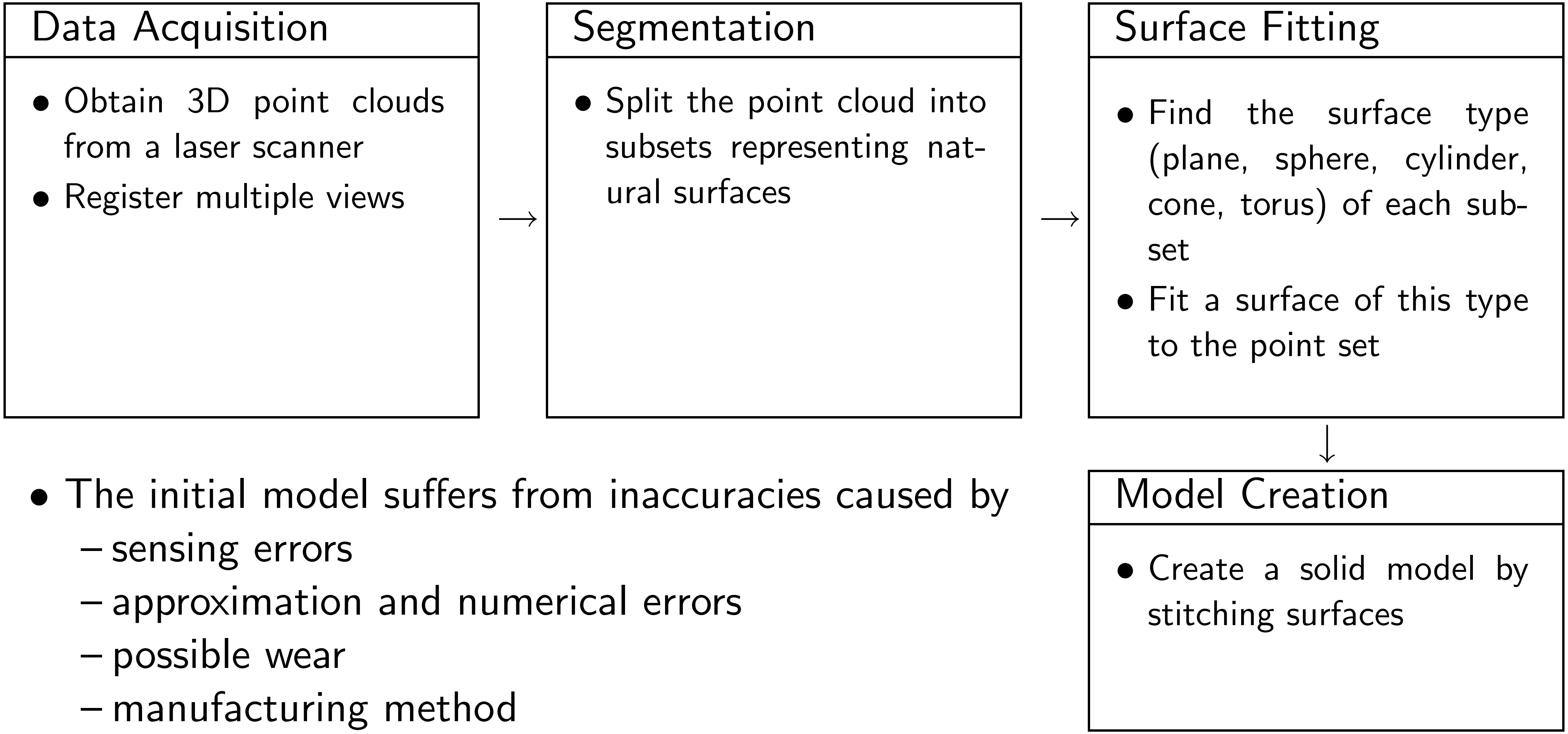
A. Dave Marshall
⟨A.D.Marshall@cs.cf.ac.uk⟩

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

9th May 2001

Department of Computer Science
Cardiff University

Reverse Engineering Geometric Models



Data Acquisition

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

→

Segmentation

- Split the point cloud into subsets representing natural 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

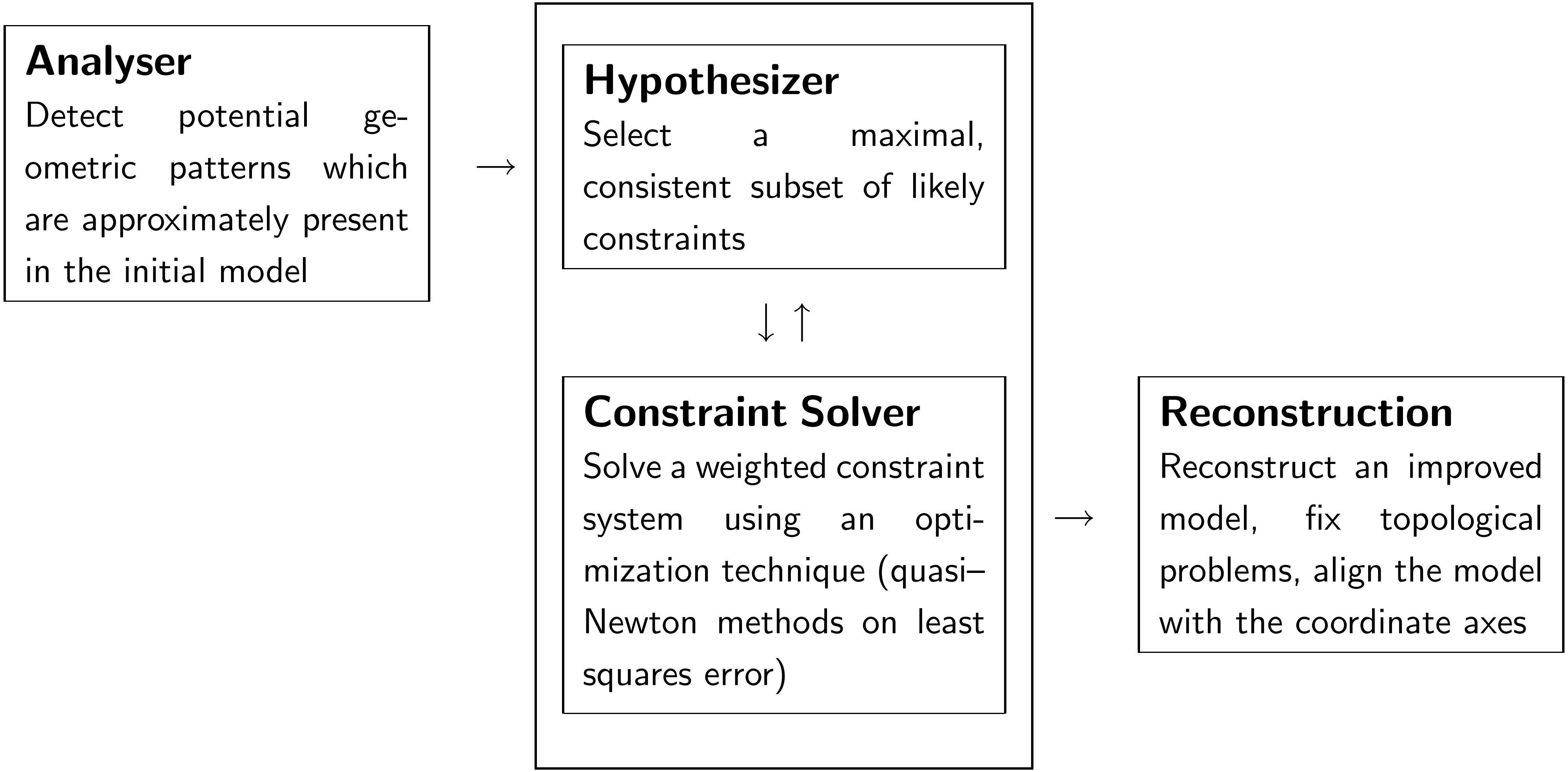
- The initial model suffers from inaccuracies caused by
  - sensing errors
  - approximation and numerical errors
  - possible wear
  - manufacturing method

⇒ **Goal:** Automatically reconstruct the *ideal* model with desired geometric patterns

Beautification

- Regular geometric patterns have to be enforced at some stage of the reconstruction process to guarantee their presence
- Previous approaches:
  - Augment the surface fitting step by constraint solving methods [Fisher,Benko]
  - Feature based approach [Thompson]:
    - manually identify features like slots and pockets
    - use them to drive the segmentation and surface fitting
- **Our approach:** Add a post–processing step called **beautification**
  - Analyse the initial model to find geometric patterns
  - Reconstruct an ideal model using geometric constraints

Beautification Strategy



Analyser

Detect potential geometric patterns 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

The Analyser:

- Which geometric patterns?
- How to detect the geometric patterns?

Geometric Patterns as Similarities

- Use similarity to recognize geometric patterns 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
  - **Example:**
    - Find a set of similar cylinder radii
    - Detect if average radius is close to a special value such as an integer

Local Patterns Identified by a Part Survey

Parameters

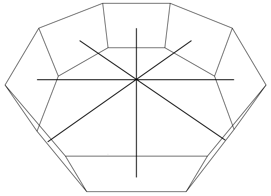
- Equal angles
- Equal lengths
- Special parameter values: integers, simple fractions

Loops

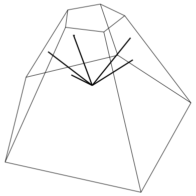
- Equal shaped polygonal loops, independent of scaling

Directions

- Parallel directions
- Direction with same angle relative to a special direction
- Symmetrical arrangements of directions in a plane



- Symmetrical arrangements of directions on a cone



Axes

- Aligned axes
- Axes intersecting in a point

Positions

- Equal positions
- Positions equal under projection

Surface Types

- Surface is approximately a plane or a cylinder

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

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

Finding Simple Fractions

- Find simple fractions $\frac{m}{n}$ approximating x in the interval $(x - t, x + t)$ with $n < M_0$
- **Algorithm:** combine a simple search with continuous fractions

I. Find the closest integer a_0 to x

II. Find fractions for the remainder $x_0 = |a_0 - x|$ recursively:

1. On recursion level l let $b = 1$ and $\frac{m}{n}$ the simple fraction found so far for x with error x_l
2. While the denominator $a = \text{round}\left(\frac{b}{x_l}\right) \leq M$

   A. If $a > b$:

   a. Let $x_{l+1} = x_l - \frac{b}{a}$

   b. New numerator $p = na \pm mb$ (depending on sign of x_l)

   c. New denominator $q = ma$

   d. If $x_{l+1} < t$, then add $\frac{p}{q}$ to the list of special values, if it is not already in it

   e. If $x_{l+1} > t_{\text{min}}$ and $\frac{p}{q}$ was not already in the list of special values, call algorithm recursively on remainder r with new limit $M = M_0 M$

   B. Let $b = b + 1$

Simple Fractions Example

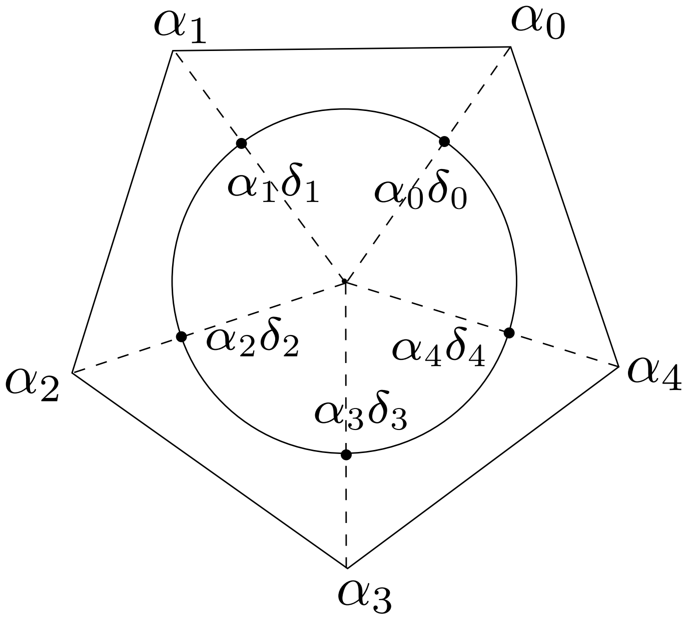
Example: $x = 0.59$ with $t = 0.05$, $M_0 = 5$ and $t_{\mathsf{min}} = 0.01$

$$\begin{array}{llllll}
0.59 & = 1/2 & + & \underbrace{0.09} & & \\
 & & & = 1/11 & - & 0.000909 \\
 & & & \quad [\rightarrow \mathbf{13/22}] & & \\
 & & & = 2/22 & - & 0.000909 \\
 & = 2/3 & - & \underbrace{0.076667} & & \\
 & & & = 1/13 & - & 0.000256 \\
 & & & \quad [\rightarrow \mathbf{23/39}] & & \\
 & = 3/5 & - & 0.01 & & \\
 & \quad [\rightarrow \mathbf{3/5}] & & & &
\end{array}$$

Similar Polygonal Loops

- 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 $\mathbb{T}$ s.t.

$$\langle \delta_k, \phi \rangle = \phi\left(2\pi\frac{l_k}{l_m}\right) \text{ for } \phi \in \mathbf{C}^\infty(\mathbb{T})$$

- 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

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

Regularly Arranged 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		

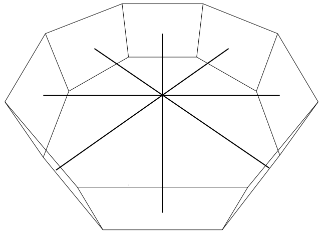
- Find **parallel directions** by clustering the projective points with

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

- 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

Symmetrically Arranged Directions

- Directions in a plane or a cone 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 $n < \dfrac{\pi}{2t_{\text{angular}}}$

- **Algorithm:** try all arrangements suggested by the angles

  I. Compute the angles α_{lk} between the directions d_l, d_k and for each angle find base angle candidates $\beta = \frac{\pi}{n}$ within the angular tolerance

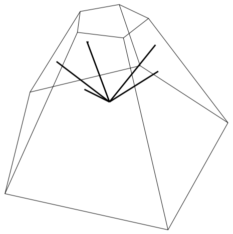
  II. For each direction d_j and its associated base angles β:

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

     Regular subset: – all angle multiples
     – at least every second multiple for $n > 4$
     – 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–regular arrangement

Special Arrangements of Axes and Positions

- 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 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**

Correcting Surface Types

- For some surfaces, the surface fitting software might not have found the correct geometric type
- Use the principal curvatures k_1, k_2 of the surface to correct the type
- For cylindrical, spherical and toroidal surfaces:

$$\begin{array}{lcl} \sqrt{k_1^2 + k_2^2} \approx 0 & \rightarrow & \text{surface is approximately a plane} \\ k_i \approx 0 & \rightarrow & \text{surface is approximately a cylinder with radius } \dfrac{1}{k_j} \end{array}$$

- For a cone with $k_1 \equiv 0$ and $k_{\mathsf{min}} \leq k_2 \leq k_{\mathsf{max}}$, $K = (k_{\mathsf{max}} + k_{\mathsf{min}})/2$:

$$k_{\mathsf{max}} - k_{\mathsf{min}} \approx 0 \text{ and } \left\{ \begin{array}{lcl} K \approx 0 & \rightarrow & \text{cone is approximately a plane} \\ K \not\approx 0 & \rightarrow & \text{cone is approximately a cylinder with radius } \dfrac{1}{K} \end{array} \right.$$

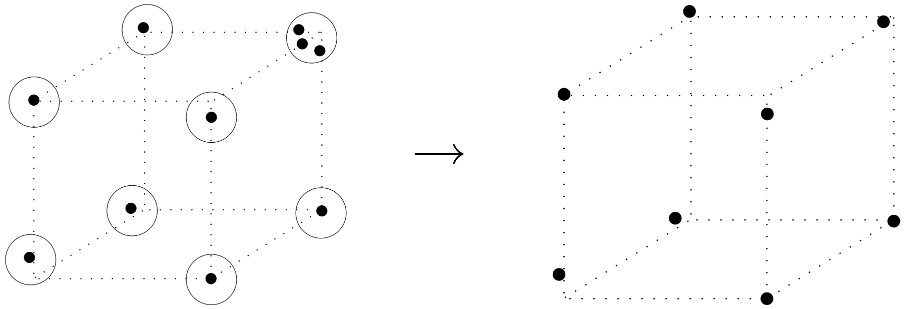
Finite Symmetry Groups

- Finite symmetry groups of the object are determined by finitely many isometries, mapping the model onto itself
- Isometries can be detected by finding isometries of point sets formed by vertices, centres of spheres, cones and tori
- An approximate isometry of a point set is a permutation which preserves 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

Finite Symmetry Groups – Algorithm Overview

- Approximate symmetry detection for point sets:

  I. Create consistent clusterings of the points at different tolerance levels such that:
     1. All distances between points in the same cluster are smaller than the tolerance level
     2. The distance between points from different clusters is larger than the tolerance level



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

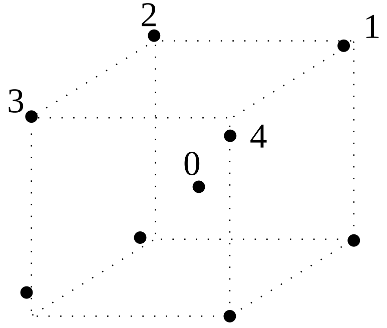
Detecting Finite Symmetry Groups

Step II: Find approximately distance preserving permutations

I. Find a non-degenerate tetrahedron whose vertices are the centroid and three points on the convex hull chosen to be as far apart as possible from each other

II. 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

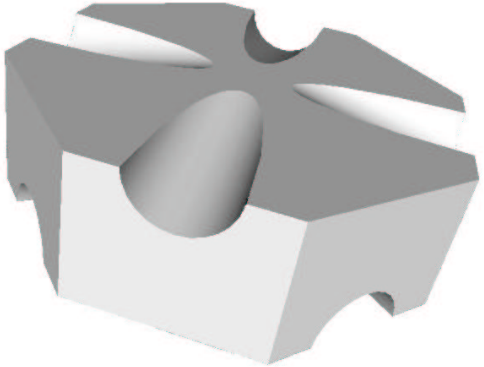


Not distance preserving:
$0 \to 0, 1 \to 2, 2 \to 4$
$[d(0,1) \approx d(0,2)]$
$d(0,2) \approx d(0,4)$
$d(1,2) \not\approx d(2,4)$

Distance preserving:
$0 \to 0, 1 \to 2, 2 \to 3$
$[d(0,1) \approx d(0,2)]$
$d(0,2) \approx d(0,3)$
$d(1,2) \approx d(2,3)$

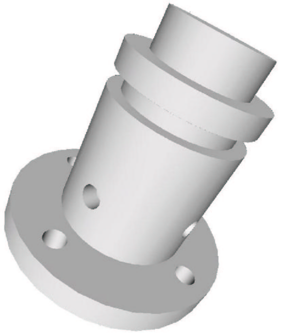
Examples

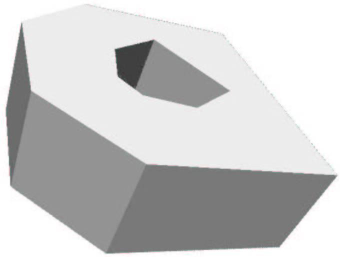
- Preliminary experiments with objects reverse engineered from simulated 3D point clouds
- Objects were perturbed by 3 degrees and 0.3 length units
- Tolerances were set for 5 degrees and 0.5 length units



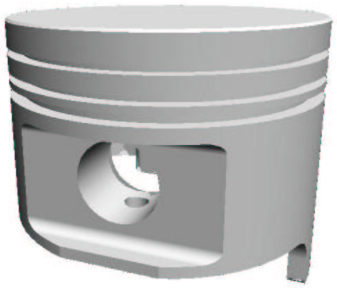
- Desired patterns found (62)
  - conical and planar angle regularities
  - axis intersection points
  - special radii and edge lengths
- Unwanted patterns (8)
  - additional conical angle regularities
  - special radii and edge lengths
- Missed patterns (7)
  - intersection point of the plane axes

More Examples





Patterns:	21	55
Unwanted:	1	8
Missed:	0	3



Patterns:	79	93
Unwanted:	5	18
Missed:	0	5

Results

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

Conclusions

- The presented methods find geometric patterns suitable for beautification
- Subsequent beautification steps must select an appropriate subset of patterns 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