Strategies for Beautification of Complex Geometric Models

F. C. Langbein

27th November 2002

Department of Computer Science
Cardiff University

CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

Beautification

- **Problem:** Reverse engineered models suffer from in-accuracies caused by
  - ⋆ sensing errors during data acquisition
  - ⋆ approximation/numerical errors during reconstruction
  - ⋆ possible wear of the artifact
  - ⋆ manufacturing method used to make the artifact
- **Goal:** Reconstruct an *ideal* model of a physical object with intended geometric regularities
- Design intent has to be considered at some stage of the process
- **Our approach:** **Beautification**, improve the model in a post-processing step

Beautification Strategy

Analyser

Detect potential regularities which are approximately present in the initial model

→

Hypothesizer

Solve a constraint system derived from the regularities which describes a complete, improved model with likely regularities

(only a subset of the constraints will be mutually consistent)

←

Reconstruction

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

Beautification Strategy

Analyser

Detect potential regularities which are approximately present in the initial model

→

Hypothesizer

Constraint Selection

Based on priorities and inconsistencies select a set of likely constraints

↓ ↑

Constraint Solver

Try to solve constraint system and indicate inconsistencies (solvability test)

←

Reconstruction

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

Key Aspects of Beautification

- Detection of suitable approximate regularities
- Selection of regularities to improve the model

  Selection criteria:

  ⋆ *intended, consistent design*

  ⋆ *solvability of the model*

- Representation of design intent in improved model

Current Beautification System

- Detect many potential approximate regularities
  - ⋆ Regular arrangements of properties as points in some space (symmetrical directions, equal radii, . . .)
- Prioritize each regularity separately with respect to
  - ⋆ Accuracy in original model
  - ⋆ Desirability/quality of regularity in general
- In order of priority add regularities to constraint system, employ test for generic solvability:
  - ⋆ If new system remains solvable, accept regularity
  - ⋆ Otherwise reject regularity
- Solve selected constraint system
- Rebuild model

Problems in Current Approach

- Current system can improve simple models:
  - ⋆ Independent, major regularities relating to most of the faces (global symmetries, orthogonal systems)
  - ⋆ Desirable regularities with high accuracy
- Problems in selecting regularities:
  - ⋆ Individual regularities rather than combinations (independent selection of angles between planes)
  - ⋆ Many dependent, ambiguous regularities for complex models
    (multiple regularities relating independent features)

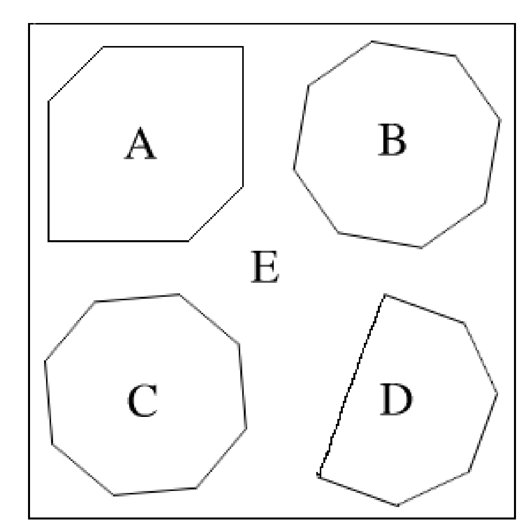
→ For complex models selected regularities are consistent w/r to solvability, but not w/r to design intent

Selection Improvements

- Develop regularity selection methods that improve handling of design intent:
- Evaluate combinations of regularities rather than individual regularities with merit functions
- Make intelligent selection decisions employing AI techniques, belief networks, etc.
- Learn which regularities and regularity combinations are common/desirable:
  - ⋆ Automatically learn from exact models
  - ⋆ Interact with user over multiple choice questions and adjustments to selected constraints

Beautification of Complex Models

- Regularity detection for complex models:
  - ⋆ Many inconsistent regularities
  - ⋆ Topological structure not considered (only regular arrangements of points/properties)
- Often complex models can be partitioned into interesting sub-parts (similar to machining features)



- ⋆ Beautification in one step has to deal with many regularities
- ⋆ Handling sub-parts separately may reduce number of regularities

Hierarchical Beautification

- Approach to hierarchical beautification:
  - ⋆ Partition model hierarchical into suitable sub-parts
  - $\longrightarrow$ Requires rules for partitioning
  - ⋆ Beautify sub-part separately as usual
  - ⋆ Re-combine sub-parts
  - $\longrightarrow$ Requires suitable relations between sub-parts (symmetries, congruences, relative orientation, location, appropriate parameter relations like equal radii, . . .)

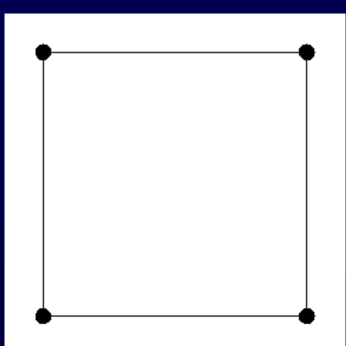
- Could also be used to find relations between different, but related objects (slot/ridge, pocket/mound, . . .)

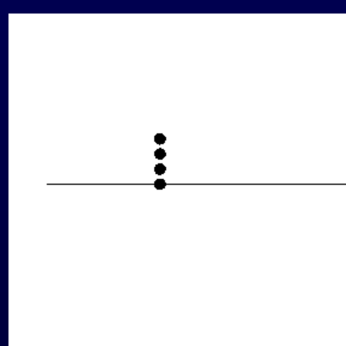
Design Intent

- After rebuilding model has certain exact regularities
- But: design intent information lost in B-rep model
- Ways to represent design intent:
  - ⋆ Store constraints with B-rep model (representation by invariant properties, Klein's Erlangner Program)
  - ⋆ Include design intent directly in representation of the model, e.g. use transformations to generate model (representation by generative sequences)
  - ⋆ Alternatives?

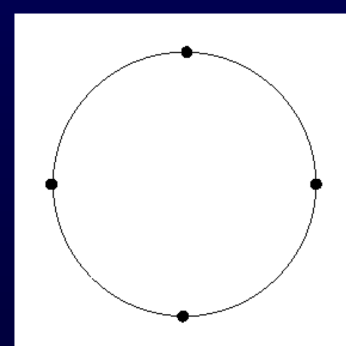
Design Intent of a Square I

- Boundary representation of square:
  - ⋆ 4 vertices, 4 straight edges (no face)
  - ⋆ Each vertex on a suitable edge pair
- Design intent of square:
  - ⋆ Equal edge lengths
    (Transformation $\mathbf{I}$ on edge lengths as points in $\mathbb{R}^1_+$)
  - ⋆ $\mathbf{k90}^\circ$ angles between edges
    (Transformations $\mathbb{Z}_4$ on edge “normals” in $\mathbb{S}^1$)





lengths invariant under $\mathbf{I}$



directions invariant under $\mathbb{Z}_4$

Design Intent and Constraints

- Find design intent by regularities:
  - ⋆ Associate B-rep elements (vertices, edges, faces) with elements of some property space (directions, positions, . . .)
  - ⋆ Detect symmetries in the property space (identity, rotation groups, . . .)
- Symmetries describe transformations which do not change the property set

Design Intent and Transformations

- Generate objects using transformations
  $\longrightarrow$ Leyton's generative theory of shape

- Basic principles:
  - ⋆ Transfer: Ability to transfer past solutions onto new problems
  - ⋆ Recoverability: Inference rules by which generative operations can be inferred from the data set

Generating a Square

- Generate line using translations

$$\mathbb{R}$$

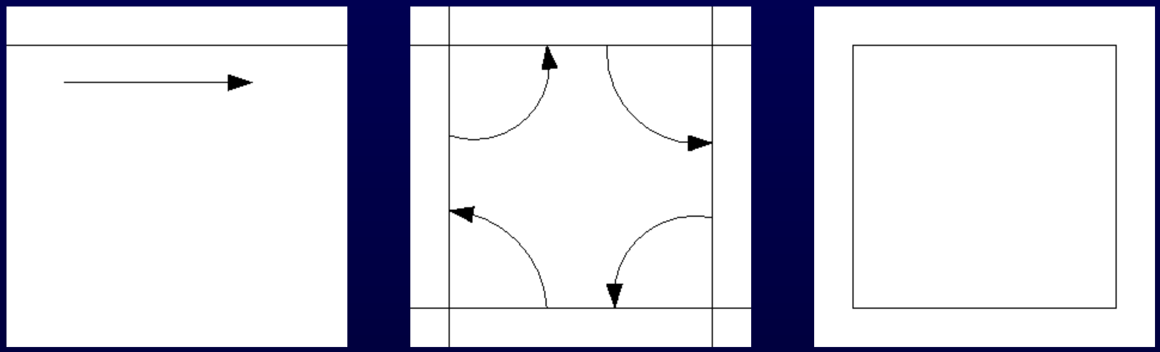
- Rotate line using $\mathbf{k90}^\circ$ rotations

  (∘ is wreath product to expand translation group $\mathbb{R}$)

$$\mathbb{R} \circ \mathbb{Z}_\mathbf{4}$$

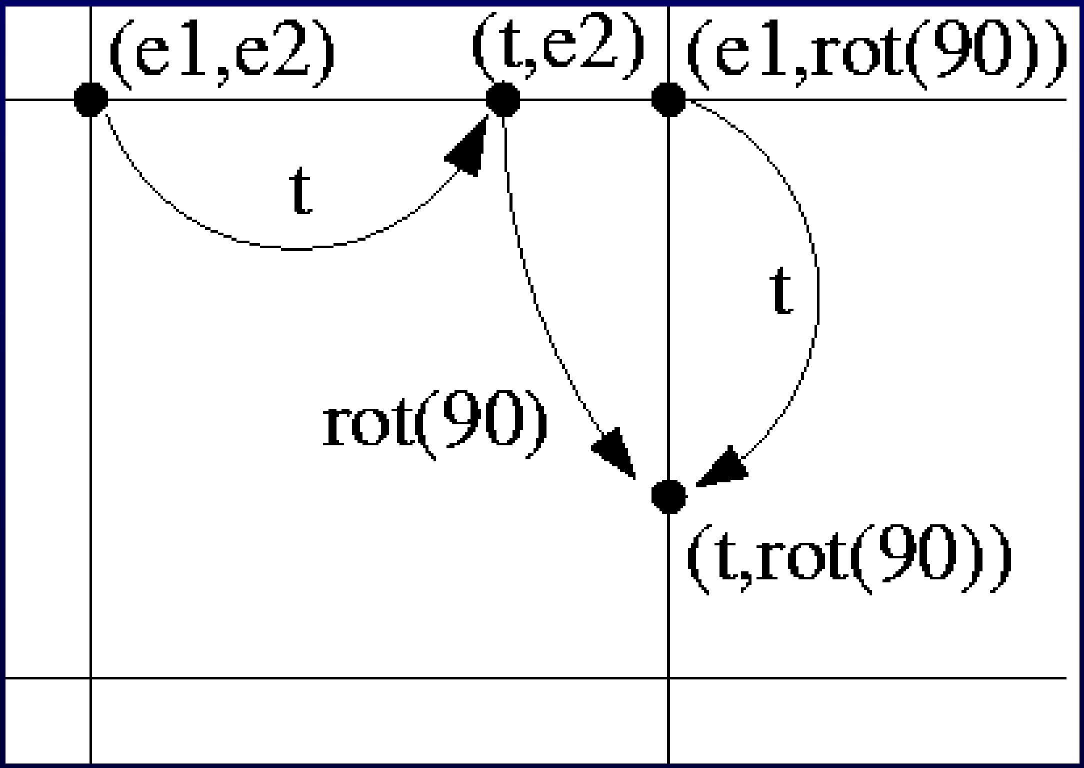
- "Cut-off" lines using occupancy group $\mathbb{Z}_\mathbf{2}$

$$\mathbb{Z}_\mathbf{2} \circ \mathbb{R} \circ \mathbb{Z}_\mathbf{4}$$



Transformations on a Square

- For simplicity ignore occupancy group $\mathbb{Z}_2$
- Point $(\mathbf{t}, \mathbf{rot}(\mathbf{k}90^\circ))$ on a square identified by translation $\mathbf{t}$ and rotation $\mathbf{rot}(\mathbf{k}90^\circ)$
- Applying transformations to the points:



Design Intent of a Square II

- Symmetries of a square (without occupancy): $\mathbb{R} \circ \mathbb{Z}_4$
  - ⋆ 4 translations $(\mathbf{T_0}, \mathbf{T_1}, \mathbf{T_2}, \mathbf{T_3})$ and 1 rotation $\mathbf{rot}(\mathbf{k90^\circ})$ give symmetry transformation

$$\langle(\mathbf{T_0}, \mathbf{T_1}, \mathbf{T_2}, \mathbf{T_3}), \mathbf{rot}(\mathbf{k90^\circ})\rangle : \mathtt{square} \longrightarrow \mathtt{square}$$
$$(\mathbf{t}, \mathbf{rot}(\mathbf{j90^\circ})) \mapsto (\mathbf{T_j t}, \mathbf{rot}(\mathbf{j90^\circ})\mathbf{rot}(\mathbf{k90^\circ}))$$

  - ⋆ Symmetries of square as $\mathbb{R} \circ \mathbb{Z}_4$ generate it
- Design intent implicitly encoded in group structure (take an edge $\mathbb{Z}_2 \circ \mathbb{R}$ and rotate it by $\mathbb{Z}_4$)
- Design intent described by generative structure

Conclusion

- Selection/detection of consistent design intent:
  - ⋆ Employ AI techniques to reason and learn about design intent
  - ⋆ Partition model hierarchical to handle complex cases
- At least two ways to represent design intent:
  - ⋆ Describe it by invariant structures
  - ⋆ Describe it by generative structures