Numerical Methods for Beautification of Reverse Engineered Geometric Models

R. R. Martin

F. C. Langbein A. D. Marshall

11th July 2002

Department of Computer Science

Cardiff University

CARDIFF UNIVERSITY

PRIFYSGOL CAERDYDD

Reverse Engineering

- Engineering converts a concept into an artifact
- Reverse engineering converts an artifact into a concept
- Desired result is a representation of the design intent, not a simple copy
- **Problem:** Reverse engineered models suffer from inaccuracies caused by
  - ⋆ sensing errors during data acquisition
  - ⋆ approximation and numerical errors from reconstruction algorithms
  - ⋆ possible wear of the artifact
  - ⋆ manufacturing method used to make the artifact

Beautification

- **Goal:** Reconstruct an *ideal* model of a physical object with intended geometric regularities
  - ⋆ Only for engineering objects with planar, spherical, cylindrical, conical and toroidal surfaces (and blends)
- Previous approaches:
  - ⋆ Augment the surface fitting step by constraint solving methods [Fisher,Benkő]
  - ⋆ Manually identify features like slots and pockets and use them to drive the segmentation and surface fitting [Thompson]
- **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)

←

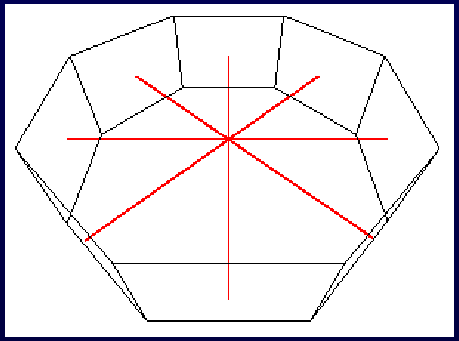
Reconstruction

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

Geometric Regularities

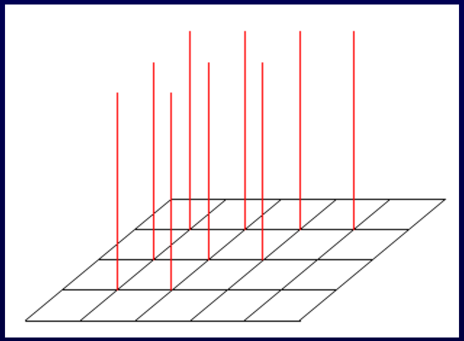
Directions

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



Axes

- Axis intersections
- Aligned axes
- Parallel axes arranged equally spaced along lines, grids or on cylinders



Positions

- Equal positions
- Positions equal under projection
- Positions arranged regularly on lines and grids

Parameter

- Equal lengths
- Equal angles
- Integer relations
- Special values:
  - ⋆ integers
  - ⋆ simple fractions

Constraint Selection & Solving

- Core of beautification problem:

  constraint selection and solving

  - ⋆ Regularities become sets of geometric constraints
  - ⋆ Model topology is described by geometric constraints
- Algorithm has to *select* desirable constraints such that the constraint system has a solution
- *Finding* a solution is a secondary task

Geometric Constraints and Objects

- Geometric constraints used to express regularities:

$\mathbf{d_1}^{\mathbf{t}}\mathbf{d_2} = \cos(\alpha)$	Const./var. angle between directions
$\mathbf{d}^{\mathbf{t}}\mathbf{d} = \mathbf{1}$	Normalize direction
$\lVert\mathbf{p_1} - \mathbf{p_2}\rVert = \lambda$	Const./var. distance between positions
$\mathbf{p} = \frac{\mathbf{1}}{\mathbf{n}}\sum_{\mathbf{k}=\mathbf{1}}^{\mathbf{n}} \mathbf{p_k}$	Average position
$\sum_{\mathbf{k}} \alpha_{\mathbf{k}}\mathbf{s_k} = \sigma$	Linear relation between scalars
$\mathbf{p} \in \mathbf{O}$	Vertex on geometric object

- Geometric object types:
  - ⋆ plane, sphere, cone, cylinder, torus, circular ellipsoid
  - ⋆ straight line, circle, ellipse
  - ⋆ vertex
  - ⋆ direction, angle, length

Selection & Solving Strategy

I. Prioritize the regularities based on
   - how well they are satisfied in the initial model
   - how common and desirable the related regularity is

II. Select initial subset $\mathbb{S}$ from all detected regularities
   - Resolve simple inconsistencies using selection rules
   - Favour regularities with high priorities

III. In order of highest to lowest priority add regularities c from $\mathbb{S}$ to a constraint system $\mathbb{C}$
   - Try to solve constraint system $\mathbb{C}$ with new regularity c
   - Add c to $\mathbb{C}$ if extended system is solvable
   - Otherwise adjust $\mathbb{S}$ using the selection rules from II as c is not used

Prioritizing Regularities

- Priorities determine which regularity to choose in case of an inconsistency
- Computed as weighted average with values in [0,1]:

$$\mathbf{w_c}(\mathbf{r})(\mathbf{c_e}\mathbf{w_e}(\mathbf{r}) + \mathbf{c_q}\mathbf{w_q}(\mathbf{r}) + \mathbf{c_b}\mathbf{w_b}(\mathbf{r}))$$

  - ⋆ with merit functions in [0,1]

$\mathbf{w_e}(\mathbf{r})$	numerical accuracy of regularity
$\mathbf{w_c}(\mathbf{r})$	constant indicating how common the regularity type is
$\mathbf{w_q}(\mathbf{r})$	specific quality/desirability depending on type and involved arrangements and constants
$\mathbf{w_b}(\mathbf{r})$	constant minimum desirability

  - ⋆ and weighting constants $\mathbf{c_e}, \mathbf{c_q}, \mathbf{c_b}$

Quality Factors

- Specific quality $\mathbf{w_q(r)}$ of a regularity is determined by factors like
  - ⋆ Special values involved (fractions with small integers preferred)
  - ⋆ Number of involved B-rep elements with common boundary
  - ⋆ Number of occupied positions for symmetrical arrangements on grids, circles, cylinders, . . .
  - ⋆ Number of B-rep elements of same geometric type
- $\mathbf{w_q(r)}$ is weighted average of merit functions for these factors

Selection Rules

- Selection rules to resolve simple inconsistencies between constraints:
  - ⋆ Constraints between the same objects with different constants
  - ⋆ Different constraints between objects which are identified by coincidence constraints
- Generic selection rule:

  A rule $(\mathbf{n_1}, \mathbf{R_1}, \mathbf{n_2}, \mathbf{R_2})$ is violated if at least $\mathbf{n_k} + \mathbf{1}$ constraints in the set $\mathbf{R_k}$ are selected for each $\mathbf{R_k} \neq \emptyset$
- One interpretation:

  If at least $\mathbf{n_1} + \mathbf{1}$ elements of $\mathbf{R_1}$ are selected, then at most $\mathbf{n_2}$ elements of $\mathbf{R_2}$ should be selected

Initial Selection

- Initially all constraints are selected
- Consecutively enforce each rule such that a maximal set of constraints with largest priorities are selected
- Basic concept for rule enforcement algorithm:
  - ⋆ If a rule is violated deselect sufficient constraints with lowest priority
  - ⋆ Check if these deselections allow the selection of previously deselected constraints by checking previously enforced rules

Rule Enforcement Example

Regularities $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ with $\mathbf{w}(\mathbf{A})<\mathbf{w}(\mathbf{B})<\mathbf{w}(\mathbf{C})<\mathbf{w}(\mathbf{D})$

$\mathbf{A}$ $\mathbf{B}$ $\mathbf{C}$ $\mathbf{D}$ 3 rules enforced in sequence:

● ● ● ● Initially all regularities selected

●: regularity selected, □: regulairty deselected

Rule Enforcement Example

Regularities $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ with $\mathbf{w}(\mathbf{A}) < \mathbf{w}(\mathbf{B}) < \mathbf{w}(\mathbf{C}) < \mathbf{w}(\mathbf{D})$

$\mathbf{A}$	$\mathbf{B}$	$\mathbf{C}$	$\mathbf{D}$
●	●	●	●
□	●	●	●

3 rules enforced in sequence:

1: $\mathbf{R_1} = \emptyset, \mathbf{n_1} = \mathbf{0}, \mathbf{R_2} = \{\mathbf{A}, \mathbf{B}\}, \mathbf{n_2} = \mathbf{1}$

At most either $\mathbf{A}$ or $\mathbf{B}$, not both, can be selected

$\longrightarrow$ Deselect $\mathbf{A}$ due to lower priority

●: regularity selected, □: regulairty deselected

Rule Enforcement Example

Regularities $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ with $\mathbf{w}(\mathbf{A}) < \mathbf{w}(\mathbf{B}) < \mathbf{w}(\mathbf{C}) < \mathbf{w}(\mathbf{D})$

$\mathbf{A}$	$\mathbf{B}$	$\mathbf{C}$	$\mathbf{D}$
●	●	●	●
□	●	●	●
□	□	●	●

3 rules enforced in sequence:

1: $\mathbf{R_1} = \emptyset, \mathbf{n_1} = \mathbf{0}, \mathbf{R_2} = \{\mathbf{A}, \mathbf{B}\}, \mathbf{n_2} = \mathbf{1}$

2: $\mathbf{R_1} = \{\mathbf{B}\}, \mathbf{n_1} = \mathbf{0}, \mathbf{R_2} = \{\mathbf{C}, \mathbf{D}\}, \mathbf{n_2} = \mathbf{1}$

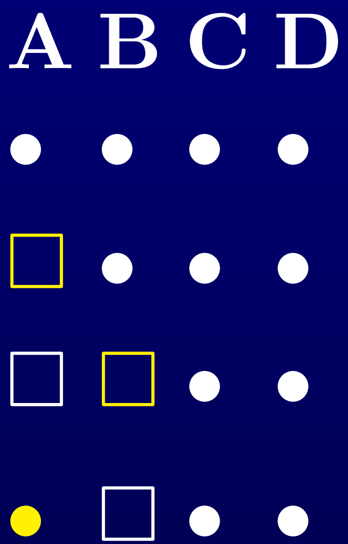
If $\mathbf{B}$ is selected, then at most either $\mathbf{C}$ or $\mathbf{D}$, not both, can be selected

$\rightarrow$ Deselect $\mathbf{B}$ due to lower priority

●: regularity selected, □: regulairty deselected

Rule Enforcement Example

Regularities $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ with $\mathbf{w}(\mathbf{A})<\mathbf{w}(\mathbf{B})<\mathbf{w}(\mathbf{C})<\mathbf{w}(\mathbf{D})$



3 rules enforced in sequence:

1: $\mathbf{R_1} = \emptyset, \mathbf{n_1} = \mathbf{0}, \mathbf{R_2} = \{\mathbf{A}, \mathbf{B}\}, \mathbf{n_2} = \mathbf{1}$

2: $\mathbf{R_1} = \{\mathbf{B}\}, \mathbf{n_1} = \mathbf{0}, \mathbf{R_2} = \{\mathbf{C}, \mathbf{D}\}, \mathbf{n_2} = \mathbf{1}$

$\leftarrow$ $\mathbf{B}$ modified, so recheck rule 1

$\rightarrow$ Can reselect $\mathbf{A}$

●: regularity selected, □: regulairty deselected

Rule Enforcement Example

Regularities $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ with $\mathbf{w}(\mathbf{A}) < \mathbf{w}(\mathbf{B}) < \mathbf{w}(\mathbf{C}) < \mathbf{w}(\mathbf{D})$

$\mathbf{A}$	$\mathbf{B}$	$\mathbf{C}$	$\mathbf{D}$
●	●	●	●
□	●	●	●
□	□	●	●
●	□	●	●
□	□	●	●

3 rules enforced in sequence:

1: $\mathbf{R_1} = \emptyset, \mathbf{n_1} = \mathbf{0}, \mathbf{R_2} = \{\mathbf{A}, \mathbf{B}\}, \mathbf{n_2} = \mathbf{1}$

2: $\mathbf{R_1} = \{\mathbf{B}\}, \mathbf{n_1} = \mathbf{0}, \mathbf{R_2} = \{\mathbf{C}, \mathbf{D}\}, \mathbf{n_2} = \mathbf{1}$

$\leftarrow$ $\mathbf{B}$ modified, so recheck rule 1

3: $\mathbf{R_1} = \{\mathbf{B}, \mathbf{D}\}, \mathbf{n_1} = \mathbf{0}, \mathbf{R_2} = \{\mathbf{A}, \mathbf{C}\}, \mathbf{n_2} = \mathbf{1}$

If $\mathbf{B}$ or $\mathbf{D}$ are selected, then at most either $\mathbf{A}$ or $\mathbf{C}$, not both, can be selected

$\rightarrow$ Deselect $\mathbf{A}$ due to lower priority

●: regularity selected, □: regulairty deselected

Rule Enforcement Example

Regularities $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ with $\mathbf{w}(\mathbf{A})<\mathbf{w}(\mathbf{B})<\mathbf{w}(\mathbf{C})<\mathbf{w}(\mathbf{D})$

$\mathbf{A}$	$\mathbf{B}$	$\mathbf{C}$	$\mathbf{D}$
●	●	●	●
□	●	●	●
□	□	●	●
●	□	●	●
□	□	●	●
□	□	●	●

3 rules enforced in sequence:

1: $\mathbf{R_1} = \emptyset, \mathbf{n_1} = \mathbf{0}, \mathbf{R_2} = \{\mathbf{A}, \mathbf{B}\}, \mathbf{n_2} = \mathbf{1}$

2: $\mathbf{R_1} = \{\mathbf{B}\}, \mathbf{n_1} = \mathbf{0}, \mathbf{R_2} = \{\mathbf{C}, \mathbf{D}\}, \mathbf{n_2} = \mathbf{1}$

$\leftarrow$ $\mathbf{B}$ modified, so recheck rule 1

3: $\mathbf{R_1} = \{\mathbf{B}, \mathbf{D}\}, \mathbf{n_1} = \mathbf{0}, \mathbf{R_2} = \{\mathbf{A}, \mathbf{C}\}, \mathbf{n_2} = \mathbf{1}$

$\leftarrow$ $\mathbf{A}$ modified, so recheck rule 1

$\rightarrow$ Cannot reselect $\mathbf{B}$ due to rule 2

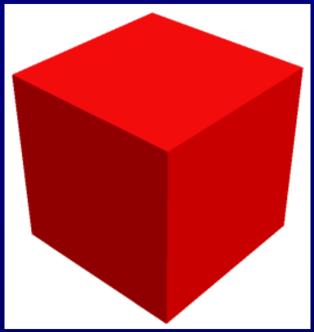
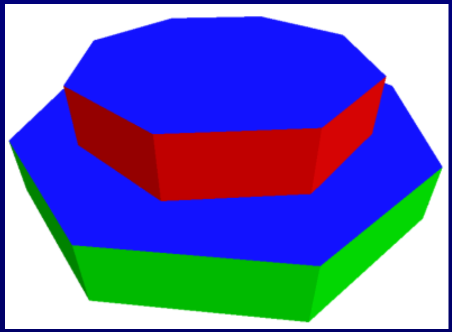
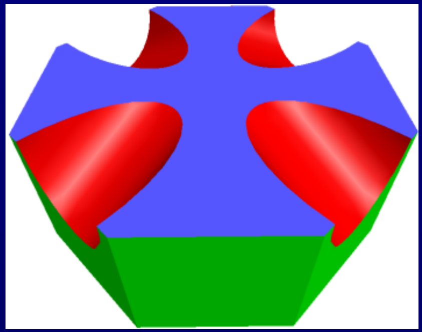
$\rightarrow$ Final selection

●: regularity selected, □: regulairty deselected

Numerical Solvability Test

- After initial selection each regularity is consecutively added to a constraint system in order of priority
- Each time the constraint system has to be checked for solvability:
  - ⋆ System is solved as numerical minimization problem (numerically stable BFGS, BFGS/Gauss-Newton hybrid, . . .)
  - ⋆ System is solvable if least-squares error is ~ 0
  - ⋆ Guarantees solvability up to numerical tolerance
- If system is solvable, the regularity is accepted
- Otherwise (conditional) selection rules involving the deselected regularity are used to check if alternative regularities can be selected

Examples

			
Detected Regularities	89	382	216
After initial selection	34	263	156
Finally selected	23	117	93
Time	$\sim$ 2h	$\sim$ 25h	$\sim$ 23h

Results

- Major regularities can be enforced on improved model
  - ⋆ Symmetrically arranged face groups
  - ⋆ Orthogonal/parallel directions, . . .
- Some problems due to ambiguities remain
  - ⋆ Choice of special values for lengths and angles not always consistent
  - ⋆ There is a choice between regularities of high quality and regularities with small errors
- Slow computation
  - ⋆ Most of the time spent on solving constraint systems (one constraint system solved for each regularity)
  - ⋆ Only proof of concept (see next slide)

Solvability Test Improvements

- To speed up algorithm: faster solvability test
- Detect structural inconsistencies without solving the constraint system
- Use degree-of-freedom analysis to build up a consistent constraint graph
- Solve the constraint system numerically only once
- Currently under investigation:
  - ⋆ Speeds up solvability test from hours to minutes
  - ⋆ Further work required to *ensure* selection of *consistent* system

Selection Improvements

- To improve quality of models: intelligent selection
- Ambiguities between approximate regularities cause inconsistent selection of regularities with respect to design intent
  - ⋆ E.g. do we have a cube with edge lengths 2 or a block with edge lengths 2 and 2.1, or $\ldots$?
  - ⋆ Local decisions can favour the block arrangements
- Make decisions in the context of the whole model, not locally with respect to inconsistencies
- Reduce/replace user-defined constants for priorities by simpler methods based on multiple-choice questions and learning

Conclusion

- We have given a general approach to improve inaccurately reverse engineered geometric models
- The current numerical method improves models, but
  - ⋆ Running time is too long
  - ⋆ Regularity selection not consistent with respect to design intent
- Current work is addressing these problems