User-assisted reverse modeling with evolutionary algorithms

Pierre-Alain Fayolle
University of Aizu

Alexander Pasko
Bournemouth University
Reverse modeling pipeline

INPUT DATA:
Triangle meshes, polygon soup, point-cloud, ...

Selection and fitting modules

Editable model (tree)

Editing, template extraction, refitting, ...

Rendering, 3D printing ...
Reverse modeling pipeline

Segmentation and fitting

CSG Tree discovery

Model editing
CSG modeling

Complex objects modeled in terms of simple primitives and operations (boolean operations, rigid transformation, etc)
Function Representation

Implicit surface:
\[ f(x,y,z) = 0 \]

Inside:
\[ f(x,y,z) > 0 \]

Outside:
\[ f(x,y,z) < 0 \]
Boolean with FRep

Given objects $A$ and $B$:

$$A = \{(x, y, z) : f(x, y, z) \geq 0\}$$
$$B = \{(x, y, z) : g(x, y, z) \geq 0\}$$

Booleans are given by:

$$A \cup B \leftrightarrow \max(f, g)$$
$$A \cap B \leftrightarrow \min(f, g)$$
$$A \setminus B \leftrightarrow \min(f, -g)$$
Boolean with FRep

R-Functions (smooth)

\[ A \cup B \leftrightarrow f + g + \sqrt{f^2 + g^2} \]
\[ A \cap B \leftrightarrow f + g - \sqrt{f^2 + g^2} \]
\[ A \setminus B \leftrightarrow f - g - \sqrt{f^2 + g^2} \]

Min/Max boolean

R-Functions boolean
Previous works (some)

Segmentation
Efficient RANSAC for point-cloud shape detection, Schnabel et al.
Direct segmentation of algebraic models for reverse engineering, Vanco et al.

Reverse engineering
Reverse engineering of geometric models-an introduction, Varady et al.

BRep to CSG
Separation for boundary to CSG conversion, Shapiro et al.
Three-dimensional halfspace constructive solid geometry construction from implicit boundary representation, Buchele et al.

Evolutionary algorithm and surface reconstruction/CSG
On the design of optimisers for surface reconstruction, Wagner et al.
Parallel surface reconstruction, Weinert et al.
Reconstruction of the ultrasonic image by the combination of genetic programing and constructive solid geometry, Yamagiwa et al.
Reconstruction for artificial degraded image using constructive solid geometry and strongly typed genetic programing, Yamagiwa et al.
Method overview

1. Pre-processing
2. Segmentation and primitive fitting
3. Construction tree discovery
4. Parameterization and refitting
Point-cloud preprocessing

1. Outlier removal
2. Alignment (in case of multiple scans)
3. Smoothing
4. Surface normals estimation
Segmentation and primitive fitting
Segmentation and primitive fitting

Given an input point cloud $P$, for each primitive $f$ from a list of candidates, find the parameter $p$ that maximizes:

$$E_1(p; f, \tilde{P}) = \sum_{i=1}^{N} \exp(-d_i(p)^2) + \exp(-\theta_i(p)^2)$$

- Penalize primitives poorly approximating the point-cloud
- Penalize primitives with normals deviating from the point-cloud normals
Results
Separating primitives

Can not be represented by applying booleans to three half planes and one disk.

A CSG representation requires an “extra” half plane (in dashed lines).

They are computed from the planes of the bounding boxes of each curved primitive.
Construction tree recovery

Goal: given a list of fitted primitives, combine them by boolean operations such that the expression represents (or approximates) the input point-cloud.
Genetic algorithm approach

Search for \( f \) implicitly defining the solid under the form: \( f_1 \circ_j f_2 \cdots f_m \)

Where: \( f_i \) are primitives and \( o_j \) are (boolean) operations.

Points \( x \) from the input point-cloud are on or near the surface, so we need to minimize:

\[
E_2(f) = \sum_{x_i \in P} f(x_i)^2
\]
Representation

Represent an individual as a list of pairs (op, L), where:

- op: index in the set of possible operations
- L: position of the primitive in the expression f

Evaluate an individual by evaluating the operations with higher precedence first (intersection and difference)
Operators

Selection: Fitness proportional selection; always include the best individuals from the previous population

Mutation: Random mutation point; stable sorting for breaking ties

Crossover: One point crossover; stable sorting for breaking ties
Parameters

Large mutation rate: 0.3
Crossover rate: 0.6
Large number of iterations (several thousands)
Population size taken to be larger than the length of an individual.
Results
Genetic programming approach

It seems natural to view the CSG expressions as trees and evolve them by genetic programming.

Given an individual $c$ and a point-cloud $P$, a raw score for $c$ (to be maximized) is computed as:

$$E_3(c; P) = \sum_{i=1}^{N} \left( \exp\left(-d_i^2\right) + \exp\left(-\theta_i^2\right) \right) - \lambda \text{size}(c)$$
Parameters

Mutation rate: 0.3
Crossover rate: 0.6
Number of iterations: (at least) 3000
Population: (at least) 100
Tree initialization: max depth of 10 and 0.7 prob. for an operation to be selected
During runtime: max depth of 20
Results

Best creature score

Graph showing the best creature score over iterations.
Results
Results

log error
-4.65472
-6
-8
-10
-12
-13.8155

Histogram of error

Frequency
Additional results
Additional results
Recovered constructive models can be parameterized to be re-used as a template model. Example of parameters: height or radius of a cylinder, height or width of a box, …

A template model can be written as: \( f(x, p) \) where \( x \) are point-coordinates and \( p \) are the parameters. Fitting \( p \) to different point-clouds corresponding to variation of the original point-cloud is done by genetic algorithm.
Results
User-assisted recovery

Many places where user assistance can be used in the pipeline:

- In the selection of possible primitives for segmentation and fitting
- In the construction tree recovery
- To extract parameters from the recovered model
User-assisted tree recovery

Incomplete model recovered (middle).
The missing handle is fitted with splines (RBF) and attached to the body with the union operation (right).
User-assisted tree recovery

The horns are incorrectly recovered (middle). The corresponding construction tree is edited to remove them. Splines (RBF) are fitted and used instead (right).
Future works

Segmentation and fitting of sweeps.
Development of tools to help automatize the parameterization.
Handle constraints in segmentation and fitting.
Detection of symmetry in the point-cloud and proper handling in the tree recovery.
Better integration of user intervention along the pipeline.