Inverse Procedural Modeling

Bedrich Benes
Purdue University
Dec 12th 2013
INRIA - Imagine

Overview

Inverse Procedural Modeling (IPM) Motivation
IPM Classification
Case studies
  - IPM of volumetric buildings
  - IPM of stochastic trees
  - Urban reparameterization
  - IPM of 2D vector images
Conclusions

Modeling

- Modeling is an open problem in CG

Traditional approaches
  - Manually
  - Scanning (and reconstruction) of real objects
  - By a code

Procedural Modeling

\[ p(t, r) = [r \sin(t), r \cos(t)] \]

Parameters \rightarrow System \rightarrow Structure

Rules
Can we find a code that generates a given structure?

**IPM Classification**

**A)** Nothing is given

*Find the system, the rules, and the parameters*

**B)** The system is given

*Find the rules and their parameters*
IPM Classification

C) The system and the rules are given
   Find the parameters

   System \arrow{?} \rightarrow \text{Structure}  \arrow{?} \rightarrow \text{Parameters} \arrow{?} \rightarrow \text{Rules}

IPM Classification

D) The system and the parameter are given
   Find the rules

   Parameters \rightarrow \text{Structure} \rightarrow ?

Manhattan-World Buildings

B) The system is given
   Find the rules and their parameters

   Vanegas, C.A., Aliaga, D.G., and Benes, B.,
   *Building Reconstruction using Manhattan-World Grammars*, Proceedings of IEEE Conference on
   Computer Vision and Pattern Recognition (CVPR) 2010

Manhattan-World Buildings

Automatically generate a 3D model of a
Manhattan-world building

*Input:*
- Geo-referenced bird’s-eye view photos
- Bounding box of the building footprint

*Output:*
- 3D model of the building represented as a grammar
A building consists of a sequence of floors \( S = \{s_1, s_2, \ldots, s_{|S|}\} \)

The external profile of \( s_i \) is a 2D polygon

Can be represented by a string of attributed letters produced by a grammar

**Generalized Rewriting Rule (GRR)**

\[
F(l) \rightarrow F(a) - F(c) + F(l - a - b) + F(c) - F(b)
\]

- **U-shape** \((a > 0 \text{ and } b > 0 \text{ and } c > 0)\)
- **Corner** \((a = 0 \text{ or } b = 0 \text{ and } c > 0)\)
- **Pushback** \((a = 0 \text{ and } b = 0 \text{ and } c > 0)\)
By image-to-geometry matching we detect “turn signals” that represent corners.

For each floor the parameters $a$, $b$, and $c$ are found via optimization (it converges to one solution).

The building is then represented by a sequence of GRR $S = \{s_1, s_2, ..., s_{|S|}\}$ and their parameters.
C) The system and the rules are given. Find the parameters


The output is a 3D model that is “alive”
**IPM of Trees**

- Novel (complicated) growth model
- Endogenous and exogenous flow
- Uses 24 parameters
- Good modeling capability

**IPM of Trees**

- Monte Carlo Markov Chain optimization with Metropolis-Hastings sampling strategy
- Similarity measure of two trees
  - Geometric similarity metrics
  - Graph distance
  - Visual similarity

**IPM of Trees**

- The output allows for random variations
- The complex 3D geometry is represented by a set of 24 parameters of the procedural model (good compression)
Urban Reparameterization

C) The system and the rules are given

Find the parameters
Use them to re-generate a city


Urban Reparameterization

The objective is urban layout editing

The user is shielded from the procedural model

High-level editing changes by indicators
- Average distance to a park
- Sun exposure
- Landscape visibility

Urban Reparameterization

A combination of two approaches

1) Forward procedural modeling by changing parameters, rules, and performing local changes
2) Inverse procedural modeling by editing indicators
Urban Reparameterization

What is the acceptable amount of changes?
Can we provide a better local control?
Can we predict the changes?
Is it too high-level?

IPM of 2D Structures

B) The system is given
Find the rules and their parameters

**IPM of 2D Structures**

**Input:** 2D vector image

**Output:** an L-system

Inspired by the previous work in symmetry detection

1. Calculate terminal symbols by inverse instancing
2. Calculate transformations, fill transformation spaces, and perform clustering
3. Analyze clusters and calculate cluster significance
4. Create non-terminal symbols and L-system rules

Update the affected clusters
Deinstancing
– Similar vector elements are coded as terminals

Transformation between two symbols
– Put all transformations into Transformation space
  Transformation = 4D Vector (2D transl., rotation, scale)
• Clustering in the transformation space
  -- Large clusters ~ significant transformations

• One transformation space for each pair of terminal symbols

• Cluster = Transformations between the same symbols

• New rule ~ new non-terminal symbol
• Clusters no longer valid
  – Update them using the new non-terminal symbol
  – Compute importance of updated clusters

Terminals
A :
B :

Non-Terminals
C :

• Generate new rules until there are no clusters
  – Axiom → Last non-terminal

Terminals
A :
B :

Axiom: S → T_{D(3)}

• Final L-system

L-system
C(m) → [A] T_{1} [B]
D(m) mod 3 → [C] T_{D(m)}
mod 3 → [C]
S → T_{1} [D(3)]

Symmetrical, but heavily randomized input is coded as L-system rules
IPM of 2D Structures

- A complex scene can have thousands of rules
- A post processing may be required
- A weighting function to give some preferences

Open Questions

- What is a set of good procedural rules?
  - Minimal grammar problem
  - Good editing properties
- What is the expression power of the IPM?
- What PM system should be used?
- Can the A case be solved?

Conclusions

- Procedural modeling is a very strong concept
- Dozens of different PM for different cases
- Hindered by the complexity of the rules
- Inverse methods can bring it to practical usage

... take a photograph of a tree/building/cloud
an app will find a PM that generates it ...