# Inverse Procedural Modeling of 3D Models for Virtual Worlds

Daniel G. Aliaga, Purdue University, USA (co-organizer)
İlke Demir, Purdue University, USA (co-organizer)
Bedrich Benes, Purdue University, USA
Michael Wand, University of Mainz, Germany

## Abstract

This course presents a collection of state-of-the-art approaches for modeling and editing of 3D models for virtual worlds, simulations, and entertainment, in addition to real-world applications. The first contribution of this course is a coherent review of inverse procedural modeling (IPM) (i.e., proceduralization of provided 3D content). We describe different formulations of the problem as well as solutions based on those formulations. We show that although the IPM framework seems under-constrained, the state-of-the-art solutions actually use simple analogies to convert the problem into a set of fundamental computer science problems, which are then solved by corresponding algorithms or optimizations. The second contribution includes a description and categorization of results and applications of the IPM frameworks. Moreover, a substantial part of the course is devoted to summarizing different domain IPM frameworks for practical content generation in modeling and animation.

**Keywords:** procedural modeling, shape analysis, inverse procedural modeling, shape editing, generative content, similarity detection, architectural modeling, geometry processing, shape

**Concepts:** •**Computing methodologies** → **Shape modeling; Mesh geometry models; Shape analysis;**

## Introduction

The demand for massive 3D models has significantly increased due to the proliferation of gaming and entertainment industry, mapping and virtualization software, and interactive applications. Models resulting from industry standard solutions vary tremendously, ranging from unorganized triangle soups to highly-structured parameterized models. The manual and reconstructed models are detailed and realistic; but they lack structure which in turn causes inefficient editing and storage. We are witnessing a renewed research interest in robust and efficient re-use of such models. One of the very interesting ways to exploit the structure inherent in existing models is by finding their procedural representation via inverse procedural modeling (IPM). In addition, automating content creation without the requirement for excessive tweaking, increase artists productivity and allow them to focus on their creative work. For this reason, the movie industry is shifting away from the manual modeling process of creating from scratch and moving towards controlling generative contents, which poses a new challenge of determining how to proceduralize existing models and support synthesis of similar plausible models.

Many of the recent advances in IPM, such as geometry based methods integrating shape editing directly to the procedural output, learning and optimization approaches that control the generative power of the procedural representation, or L-system based algorithms are specifically solved within the extents of some domains. However there is no overall analysis and collection of those solutions. Hence, there is centralized means by which researchers can become well-informed about which techniques are available for use and in which domain.

The first part of this course is a coherent review of the fundamentals of inverse procedural modeling, different perspectives on the formulation of the problem, and solutions based on those formulations. Inverse procedural modeling is usually considered as an under-constrained problem by nature; however, recent techniques address this issue by finding analogous problems in computer science, and then solve it using corresponding well-known algorithms and/or optimizations. The second part of the course presents the results and applications of those IPM frameworks; hence a substantial part of the course is devoted to introducing different domain IPM frameworks for practical content generation tasks in modeling and animation.


SIGGRAPH '16, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4289-6/16/07
DOI: http://dx.doi.org/10.1145/2897826.2927323

## Course Overview

1. Welcome and Introduction (10 mins)
   *Daniel Aliaga*

   Welcome, outline of the course, and learning outcomes. Speaker introductions. Motivations for procedural modeling (PM) and inverse procedural modeling (IPM).

2. Fundamentals of Inverse Procedural Modeling (15 mins)
   *Daniel Aliaga*

   PM background, grammar types and examples, and IPM definition and classification.

3. Low-level Priors: Inferring Shape Grammars (30 mins)
   *Michael Wand*

   An overview of IPM approaches that assume no or low level priors. Discussion of shape analysis and IPM, how building blocks can be characterized by correspondences and symmetry, and examples of methods for identifying and utilizing higher-level patterns in building block arrangements.

4. High-level Priors: Inferring Grammar Parameters (30 mins)
   *Ilke Demir*

   A look at algorithms for creating inverse procedural systems that assumes some underlying rules or properties about the system or the model. The methods include MCMC, energy minimizations and learning approaches.

5. IPM Approaches for Facades and Layouts (20 mins)
   *Michael Wand & Bedrich Benes*

   2D approaches for building facade and urban layout decompositions for inverse procedural modeling; example-based and guided/interactive methods.

6. IPM Approaches for Vegetation (25 mins)
   *Bedrich Benes*

   Guided procedural modeling for interactive content creation, and plastic trees for context-aware inverse procedural modeling.

7. IPM Approaches for Urban Areas (25 mins)
   *Ilke Demir*

   Definition of general frameworks for city-scale and building-scale proceduralization. An overview of approaches for IPM of buildings, urban sites, and their aid in reconstruction.

8. Conclusions, Future Work, and Q&A (15 mins)
   *Daniel Aliaga*

   Wrap up, review, new dimensions, and discussion.

## Prerequisites

Familiarity with procedural modeling, with a basic understanding of shape analysis and modeling.

## Level of Difficulty

Intermediate

## Audience

Industry professionals, researchers and students interested in recent techniques for inverse procedural modeling of architectural, man-made, or plant-based structures for generative content creation.

## Presenter Information


### Daniel G. Aliaga, Purdue University, aliaga@purdue.edu

Daniel G. Aliaga is an Associate Professor of Computer Science at Purdue University. He obtained his Ph.D. and M.S. degree from UNC Chapel Hill and his B.S. degree from Brown University. Dr. Aliaga's research is primarily in the area of 3D computer graphics but overlaps with computer vision. His research area is of central importance to many critically important industries, including computer-aided design and manufacturing, telepresence, scientific simulations, and education. To date Prof. Aliaga has over 100 peer reviewed publications and has chaired and served on numerous ACM and IEEE conference and workshop committees.

### Bedrich Benes, Purdue University, bbenes@purdue.edu

Bedrich Benes is a professor in the Department of Computer Graphics Technology, director of High Performance Computer Graphics Laboratory, and director of GPU Research Center at Purdue University. He obtained his Ph.D. and M.S. degrees in Computer Science from the Czech Technical University in Prague. His research is primarily in the area of procedural modeling, real-time rendering, and 3D printing. To date he has published over 100 peer reviewed publications and three textbooks.

### Ilke Demir, Purdue University, idemir@purdue.edu

Ilke is a PhD candidate in Department of Computer Science at Purdue University and a research assistant in Computer Graphics and Visualization Lab. She obtained her B.S. degree in Computer Engineering from Middle East Technical University in 2010 with a minor in Electrical Engineering. Her research interests lie in the fields of procedural modeling, 3D urban reconstruction, and interactive shape editing. She started her research carrier in KOVAN Robotics Lab, has served as the student system admin in METU, and worked at Havelsan Inc. on graphics and simulation projects. Further she worked at Pixar Animation Studios as an intern in 2015-2016.

### Michael Wand, University of Mainz Michael.Wand@uni-mainz.de

Michael Wand is a professor in the Institute of Computer Science at University of Mainz, Germany, where he is heading the visual computing group. His main research interests lie at the intersection of computer graphics, computer vision, and machine learning. Previously, he has held the position of an universitair hoofdocent at the Department of Information and Computing Sciences at Utrecht University. He has received a diploma in Computer Science from Paderborn University in 2000 and a Ph.D. from Tübingen University in 2004. From 2005 to 2007, he has visited Stanford University as a postdoc, and from 2007 to 2013, he has been a senior scientist and junior research group leader at the Max-Planck Institute for Informatics and at Saarland University.

# Content


1. *(Aliaga)* Introduction……5
   1.1. The main challenge
   1.2. Procedural modeling
   1.3. Inverse procedural modeling
   1.4. Lecturers
2. *(Aliaga)* Fundamentals ……17
   2.1. Procedural modeling
      2.1.1. Example grammars
      2.1.2. Usage and problems
   2.2. Inverse procedural modeling
      2.2.1. Motivation & Classification

   Prusinkiewicz90, Prusinkiewicz94, Mech96, Deussen98

3. Approaches for inverse procedural modeling
   3.1. *(Wand)* Low-level priors: inferring shape grammars……38
      3.1.1. Building blocks
         3.1.1.1. Direct approaches
         3.1.1.2. Optimization approaches
      3.1.2. Rules & synthesis algorithms
         3.1.2.1. Assembling tiles
         3.1.2.2. Double-cuts
         3.1.2.3. Algebraic methods
         3.1.2.4. Guided methods

   Hertzmann01, Mitra06, Merell07, Pauly08, Bokeloh09, Bokeloh10, Stava10, Vanegas10, Bokeloh12, Kalojanov12, Demir15a, Li15, Liu15

   3.2. *(Demir)* High-level Priors: Inferring Grammar Parameters……115
      3.2.1. Assumptions & Formulation
      3.2.2. Markov Chain Monte Carlo
         3.2.2.1. Random walks with local & global moves
         3.2.2.2. Metropolis-Hastings with rjMCMC
         3.2.2.3. Stochastically-ordered sequential MCMC
      3.2.3. Machine Learning
         3.2.3.1. Autoencoders in IPM
         3.2.3.2. CNNs in IPM

   Talton10, Vanegas12, Talton12, Martinovic13, Ritchie15, Yumer15, Dang15, Nishida16

4. Inverse Procedural Modeling Domains
   4.1. *(Wand/Benes)* Facades and Layouts……177
      4.1.1. Example system
      4.1.2. Modeling facade layouts
      4.1.3. Matching data
      4.1.4. Direct modeling

   Müller07, Lipp08, Xiao09, Lefebre10, Shen11, Martinovic12, He12, Bao13, Zhang13, Wu14, Dang14, Ilcik15

   4.2. *(Benes)* Vegetation……219
      4.2.1. Inverse Procedural Modeling of Trees
      4.2.2. Guided Procedural Modeling
      4.2.3. Other tree modeling work

   Benes11, Longay12, Stava12, Friedman13, Stava14, Xu15

   4.3. *(Demir/Aliaga)* Urban Areas ……258
      4.3.1. Urban Modeling Pipeline
      4.3.2. Urban IPM Pipeline
      4.3.3. Approaches
         4.3.3.1. Image-based methods
         4.3.3.2. Mesh-based methods
         4.3.3.3. Point-based methods

   Aliaga07, Hohhman09, Toshev10, Vanegas10, Markus11, Demir14, Demir15b

5. *(Aliaga)* Conclusions, Challenges and Open Problems ……307
   5.1. Challenges
   5.2. Current Projects
   5.3. Acknowledgments

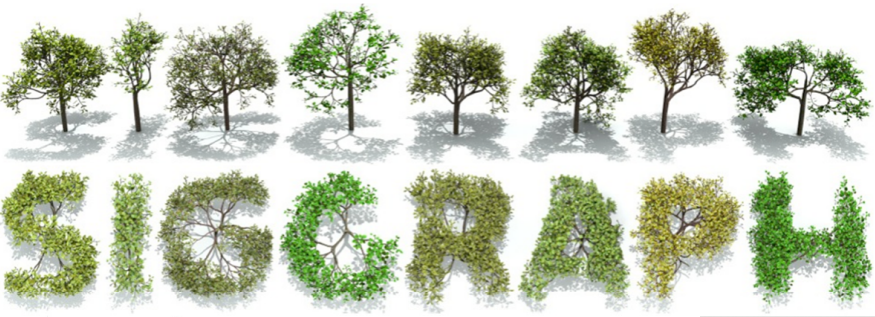
   Coming soon!

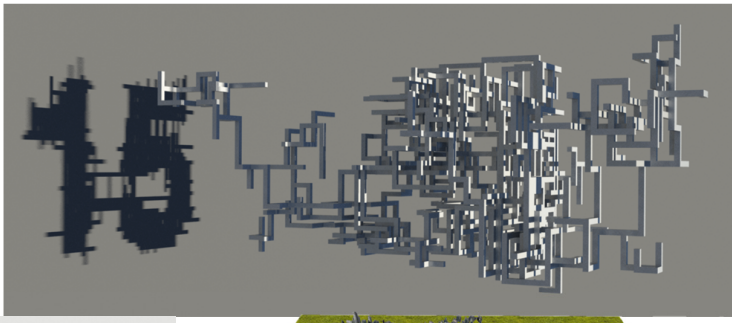

Render the Possibilities
SIGGRAPH2016
THE 43RD INTERNATIONAL
CONFERENCE AND EXHIBITION ON
&
Computer Graphics
Interactive Techniques
24-28 JULY
ANAHEIM, CALIFORNIA

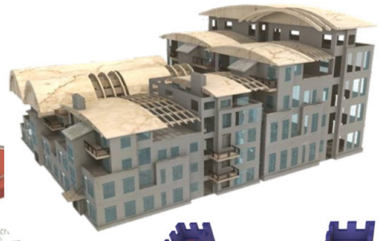
## 1 Introduction

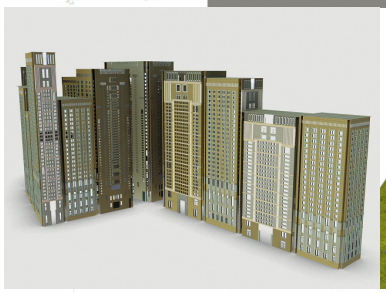
# Outline

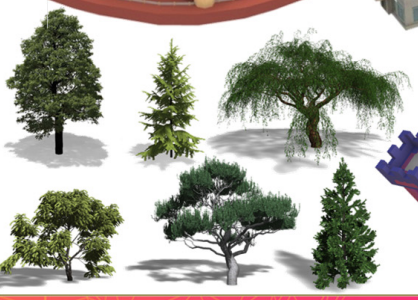
- **Introduction**
- Fundamentals of IPM
- IPM Approaches
  - Low-level Priors: Inferring Shape Grammars
  - High-level Priors: Inferring Grammar Parameters
- Inverse Procedural Modeling Domains
  - Facades & Layouts
  - Vegetation
  - Urban Areas
- Conclusions, Challenges, Open Problems

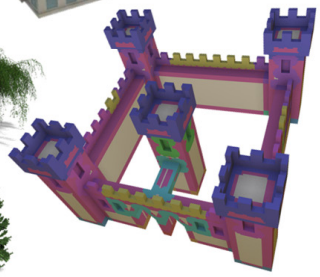

SIGGRAPH

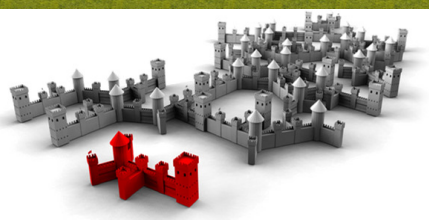












# Introduction

- What is the main challenge?
- What is procedural modeling?
- What is inverse procedural modeling?
- Why should you learn about it?
- What are the current approaches?
- Why did we decide to do this course?
- Who are we?

# What is the main challenge?

- Solving the **content problem**
  - As computing and display capabilities continually improve, audience expects ever higher quality digital content
  - Traditional tools are insufficient for increasing demand and few tools are available for efficient large-scale modeling

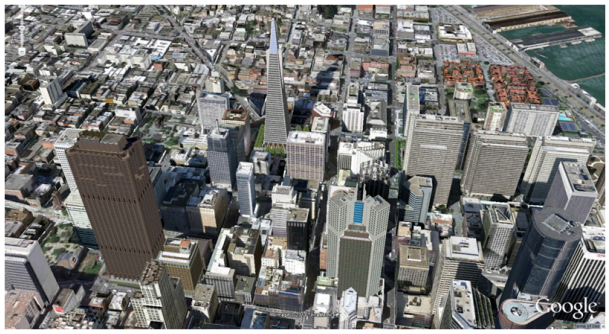
# What is procedural modeling?

- One content generation option is modeling buildings, plants, clouds, cities, and worlds using an underlying system of rules.

```
G = { terminals,
      non-terminals  →
      rules,
      axiom }
```

# What is inverse procedural modeling?

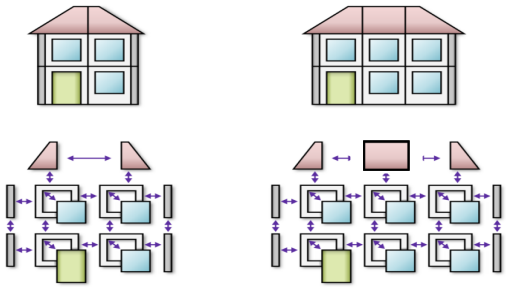
- Extracting rules and/or parameters to create a generative system from given buildings, plants, clouds, cities, and worlds.

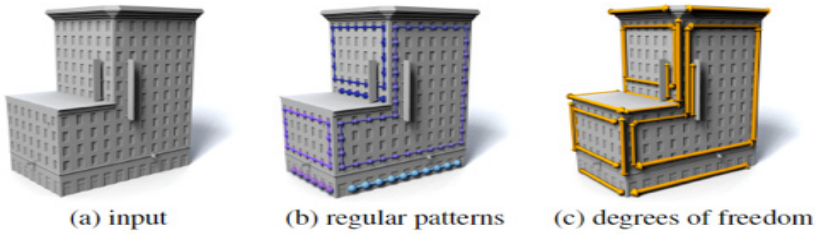



```
G = { terminals,
      non-terminals
      rules,
      axiom }
```

# What are the current approaches?

- Having low-level priors:
  - Formulations to partition geometry into components and then extract patterns and rules





# What are the current approaches?

- Having low-level priors:
  - Formulations to componentize the geometry into similarity groups to extract the patterns from.
- Having high-level priors:
  - Guiding and controlling the content generation by discovering the optimal parameters and rules for a given target.

# Why did we decide to do this course?

- We wish to inform the research community of the latest research work in the relevant areas
  - There is a recent proliferation of inverse procedural modeling publications in top computer graphics journals and conferences

# Why did we decide to do this course?

- We wish to organize and analyze the inverse procedural modeling approaches for generative systems
  - Many domains, many approaches, many formulations -- no overall organization and/or awareness.

# Who are we?

- Lecturers

Daniel Aliaga
aliaga@purdue.edu

Ilke Demir
idemir@purdue.edu

Bedrich Benes
bbenes@purdue.edu

Michael Wand
wandm@uni-mainz.de

# Outline

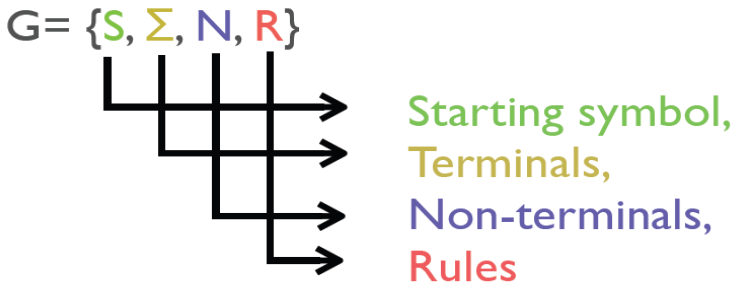
- Introduction
- **Fundamentals**
- IPM Approaches
  - Low-level Priors: Inferring Grammars
  - High-level Priors: Inferring Grammar Parameters
- Inverse Procedural Modeling Domains
  - Facades & Layouts
  - Vegetation
  - Urban Areas
- Conclusions, Challenges, Open Problems

# Fundamentals

- **Procedural Modeling**
  - **Example Grammars**
  - **Usage and Problems**
- Inverse Procedural Modeling
  - Motivation
  - Classification

# Procedural Modeling: Grammars

- A rule-based generative system loosely organized as:

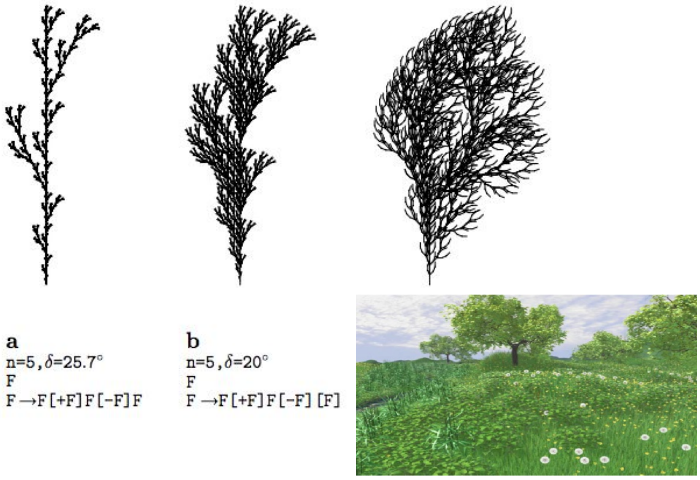


# Some Grammars

- L-systems
  - e.g., plants
- Split Grammars
  - e.g., facades, buildings
- Shape Grammars
  - e.g., architecture, sculptures, terrain, CGA (CityEngine)

# L-systems

- A parallel string rewriting system
  - Generation of plants
    Prusinkiewicz, Lindenmayer; 1990
  - Environment-sensitive
    Prusinkiewicz, James, Mech; 1994
  - Interaction (Open L-System)
    Mech, Prusinkiewicz; 1996
  - Ecosystems
    Deussen, et al.; 1998

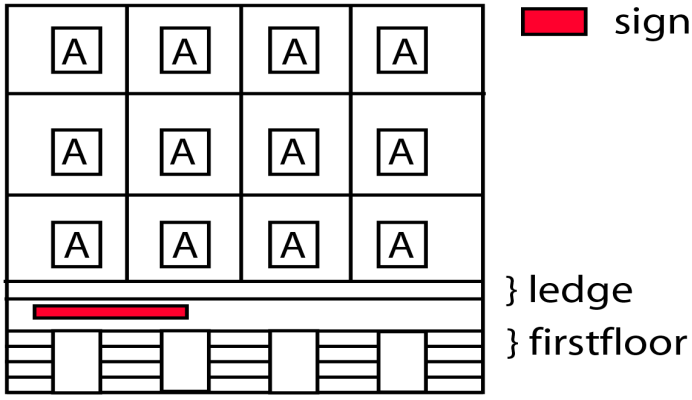


# Some Grammars

- L-systems
  - e.g., plants
- Split Grammars
  - e.g., facades, buildings
- Shape Grammars
  - e.g., architecture, sculptures, terrain, CGA (CityEngine)

# Split Grammars

- Recursively split plane/volume into subparts

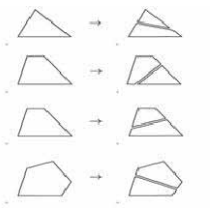


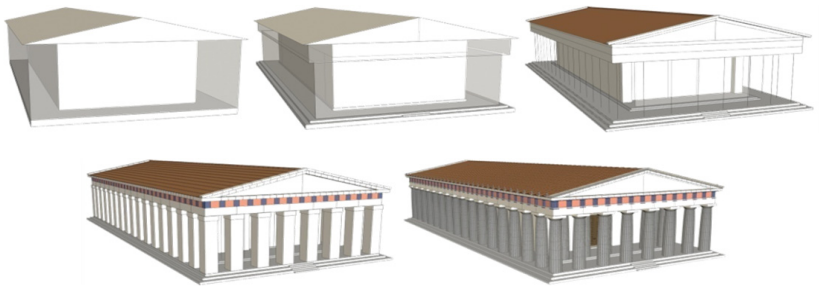
# Some Grammars

- L-systems
  - e.g., plants
- Split Grammars
  - e.g., facades, buildings
- Shape Grammars
  - e.g., architecture, sculptures, terrain, CGA (CityEngine)

# Shape Grammar

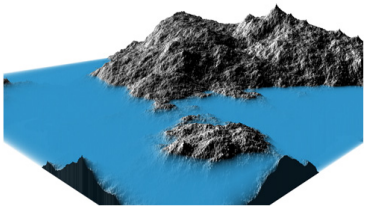
- Rules for how a shape can be transformed





# Other Generative Models

- Fractals
  - e.g., diamond-square algorithm



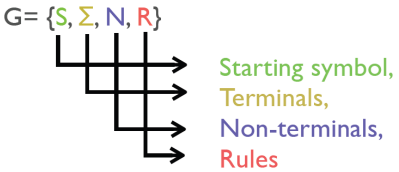
- Procedural Noise
  - e.g., Perlin Noise

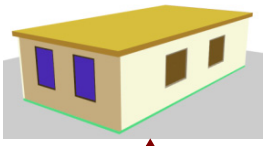
- And more!
  - e.g., continuous shape spaces, learned models

# Procedural Modeling: Example


- Seek a grammar that produces a desired 3D model

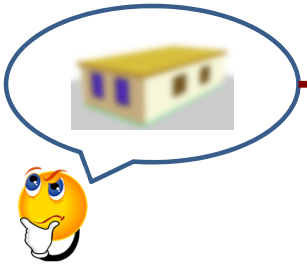
# Procedural Modeling: Example



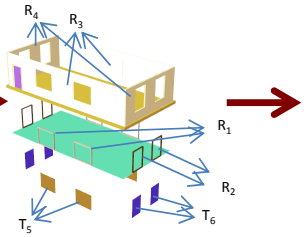



Derive an instance based on desired model parameters…

Imagine the output…

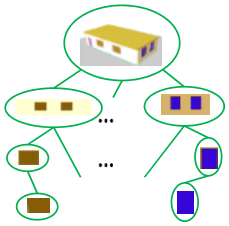


Create terminals and rules…



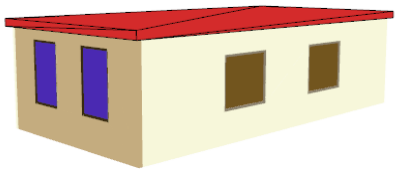

Write grammar…

```
Rule Building(S) = {                    // CREATE ENTIRE BUILDING
    a = split(S, 0.0, 0.0, 0.80, 4) // roof terminal
    c = split(S, 1.0, 1.0, 0.01, 5) // floor terminal
    B = split(S, 1.0, 1.0, 0.80, 5) // wall bbox
    D = split(B, 1.0, 0.1, 1.00, 5) // front wall bbox
    E = split(B, 0.0, 0.9, 0.00, 4) // back wall bbox
    F = split(B, 0.1, 1.0, 1.00, 5) // left wall bbox
    G = split(B, 0.9, 0.0, 0.00, 4) // right wall bbox
    R3(D); R3(E);                   // front/back rule
    R4(F); R4(G);                   // left/right rule
}
Rule R3(A) = {                          // CREATE FRONT/BACK WALL
    b = split(A, 0.0, 0.0, 0.00, 4) // wall terminal
    C = split(A, 0.4, 1.0, 0.75, 5)
    D = split(C, 0.6, 0.3, 0.00, 4) // left window bbox_
    E = split(A, 0.6, 1.0, 0.75, 6)
    F = split(E, 0.3, 1.0, 0.30, 1) // right window
    R1(D); R1(F);                   // window rule
}
Rule R1(A) = {
    // perform splits for frame and window
}
[...and add'l text for R2 and R4]
```



# Procedural Modeling: Example

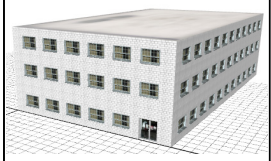
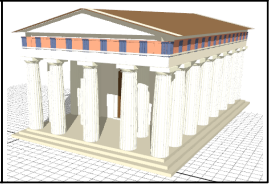
- Ultimately, to produce this building



need to write →

```
Rule Building(S) = {                    // CREATE ENTIRE BUILDING
    a = split(S, 0.0, 0.0, 0.80, 4) // roof terminal
    c = split(S, 1.0, 1.0, 0.01, 5) // floor terminal
    B = split(S, 1.0, 1.0, 0.80, 5) // wall bbox
    D = split(B, 1.0, 0.1, 1.00, 5) // front wall bbox
    E = split(B, 0.0, 0.9, 0.00, 4) // back wall bbox
    F = split(B, 0.1, 1.0, 1.00, 5) // left wall bbox
    G = split(B, 0.9, 0.0, 0.00, 4) // right wall bbox
    R3(D); R3(E);                   // front/back rule
    R4(F); R4(G);  }                // left/right rule
}
Rule R3(A) = {                      // CREATE FRONT/BACK WALL
    b = split(A, 0.0, 0.0, 0.00, 4) // wall terminal
    C = split(A, 0.4, 1.0, 0.75, 5)
    D = split(C, 0.6, 0.3, 0.00, 4) // left window bbox_
    E = split(A, 0.6, 1.0, 0.75, 6)
    F = split(E, 0.3, 1.0, 0.30, 1) // right window
    R1(D); R1(F);                   // window rule
}
Rule R1(A) = {
    // perform splits for frame and window
}
[...and add'l text for R2 and R4]
```

# Procedural Modeling: CityEngine

| | #lines | output |
|---|---|---|
| simpleBuilding.04.cga | 171 lines |  |
| CandlerBuilding.cga | 471 lines |  |
| parthenon.cga | 524 lines |  |

# Procedural Modeling: CityEngine

```
version "2010.3"

/* Attributes ************************************/
@Range(2,6)
attr groundfloor_height = 4
@Range(2,6)
attr floor_height = 3.5

@Range(1,10)
attr tile_width = rand(2.5,6)
@Range(3,50)
attr height = rand(11,30)
attr wallColor  =
    33% :  "#ffffff"
    33% :  "#999999"
    else :  "#444444"

@Range(LOD0=0,LOD1=1)
attr LOD = 0
/* Assets ***************************************/

// geometries
const window_asset = "facades/window.obj"

// textures
const frontdoor_tex = "facades/textures/shopdoor.tif"
const wall_tex = "facades/textures/brickwall.jpg"
const dirt_tex = "facades/textures/dirtmap.15.tif"
const roof_tex = "roofs/roof.tif"

# this function will get one of the 9 window textures in the assets folder
randomWindowTexture = fileRandom("facades/textures/window-*.tif")

/* Initial Shape starting rule ************/

# scale the lit to leave a small border and extrude the lot to building height
Lot --> extrude(height) Building

# inner lots are dropped
LotInner --> NIL

# split the building geometry into its facade components
Building --> comp(f) { front : FrontFacade | side : SideFacade | top: Roof}

# the front facade is subdivided into one front groundfloor
# and upper floors
FrontFacade -->
    setupProjection(0, scope.xy, 1.5, 1, 1) # setup 1.5m x 1m texture tiles along scopes xy plane (and distortion in z)
    setupProjection(2, scope.xy, scope.sx, scope.sy)
    split(y){ groundfloor_height    : Groundfloor | {~floor_height : Floor}* }

# a side facade is subdivided into one bottom floor
# and upper floors.
SideFacade -->
    setupProjection(0, scope.xy, 1.5, 1, 1) # setup 1.5m x 1m texture tiles along scopes xy plane (and distortion in z)
    setupProjection(2, scope.xy, scope.sx, scope.sy)
    split(y){ groundfloor_height    : Floor | {~floor_height : Floor}* }

# a roof texture is applied to the roof face
Roof -->
    color(wallColor)
    setupProjection(0, scope.xy, scope.sx, scope.sy)
    texture(roof_tex)
    projectUV(0)

# each floor is horizontally split into two narrow corner areas on
# each side of the floor, and into a set of window tiles in between
Floor --> split(x){ 1 : Wall | { ~tile_width : Tile }* | 1 : Wall }

...
```

```
version "2014.0"

@Order(1) @Range(28,150)
attr BuildingHeight = 62
@Order(2) @Range("Visualization","3DPrint") @Description("Generates a watertight model with less detail - suited for most 3D printers.")
attr Mode = "Visualization"


@Group("Facade",3)
@Order(1) @Range(2.5,5)
attr FloorHeight = 3.5
@Order(2) @Range(3,6)
attr GroundfloorHeight = 4.3
@Order(3) @Range(1.5,5)
attr TileWidth = 3.55
@Order(4) @Range(0.5,3)
attr CorniceOverhang = 1.2

@Group("Windows",4)
@Order(1) @Range(1.3,5)
attr WindowHeight= 2.05
@Order(2) @Range(1,4)
attr FrontWindowWidth= 2.15
@Order(3) @Range(0.5,2)
attr RearWindowWidth= 1.2
@Order(4) @Range(0,0.7)
attr SillSize= 0.26

@Group("Wall",5)
@Order(1) @Range(0,2)
attr CornerWallWidth= 1
@Order(2) @File("jpg","png","tif")
attr WallTexture= "facade/walls/bricks.jpg"
@Order(3)
attr ColorizeWall= "#FCEFE2"

@Hidden
attr onStreet= true

const winSideW= (TileWidth-FrontWindowWidth)/2
const dividerW= 0.25
const sillH= case FloorHeight>WindowHeight+SillSize: (FloorHeight-WindowHeight)*0.66 else: SillSize

# ------------------------------------
# Extrusion and facade dispatcher
# ------------------------------------

@StartRule
Footprint -->
    InsideSkin
    extrude(BuildingHeight) Solid

Solid -->
    comp(f){ front: FrontFacade  | back : RearFacade | left : SideFacade | right: SideFacade | top  : Roof }

FrontFacade -->
    set(onStreet,true)
    Facade(2,1,3,1.2)
RearFacade -->
    set(onStreet,false) # is not facing street (i.e. no shops nor cornice)
    Facade(0,0, 0,0,0)        # small windows only (marked with 0)

SideFacade -->
  case scope.sx < 30:  # catching inner backfacades (hack due to the uncommon footprint)
    RearFacade
  else:
    set(onStreet,true)          # is facing street (i.e. has shops, ledges and cornice)
    Facade(2,1,3.2,3)

...
```

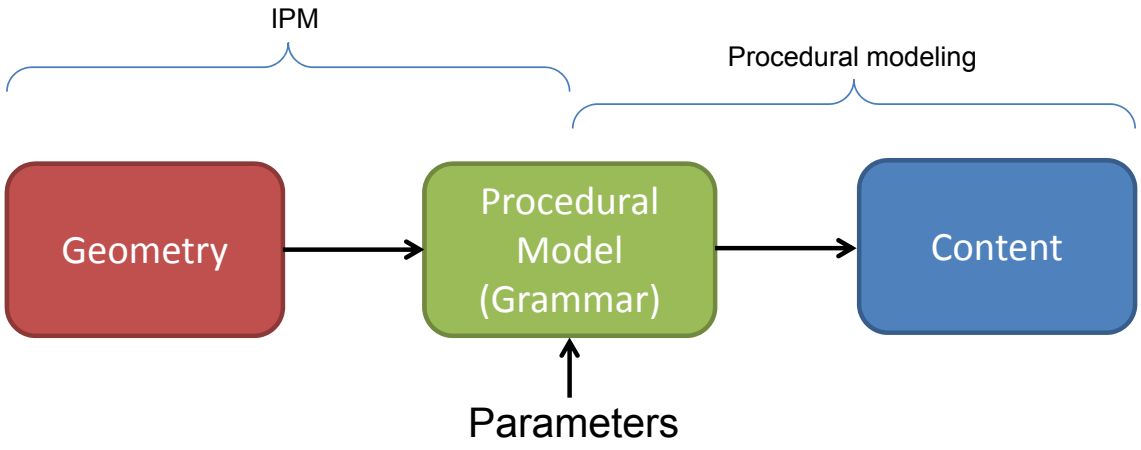
# Procedural Modeling: Observations

- Given a target object/shape, the PM system must be hand-crafted

- PM provides high detail amplification
  - Pro: lots of content
  - Con: its hard to write a PM system

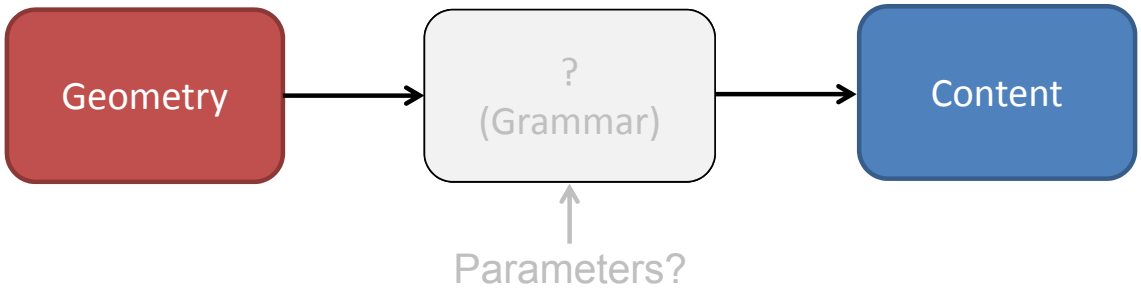
# Inverse Procedural Modeling (IPM)

- Converts existing geometry into procedural models:



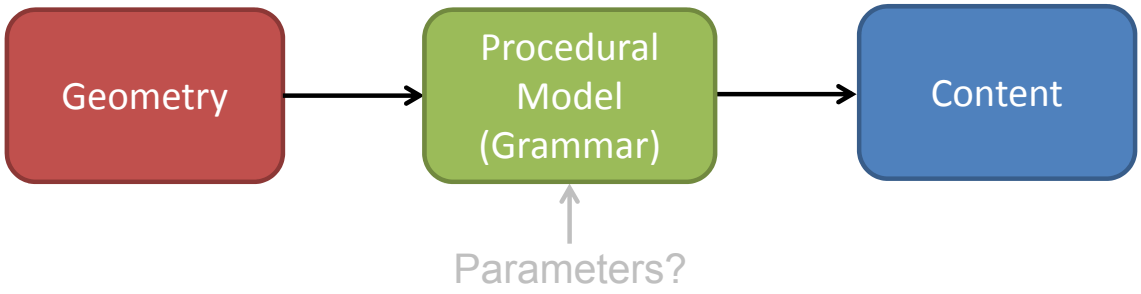
# IPM Classification

- Converts existing geometry into procedural models
- A classification:
  - 1. inferring procedural model and its parameters



# IPM Classification

- Converts existing geometry into procedural models
- A classification:
  - 2. inferring procedural model parameters (i.e., rules are known)



# IPM Classification

- Inferring procedural model and its parameters
  - Only low-level priors can be assumed
  - Usually operates directly on the geometry

- Inferring procedural model parameters
  - High-level priors can be assumed
  - PM system is given but its usage must be determined/searched

# Benefits of IPM

- Content Synthesis
  - Provides a parameterized procedural representation for easy synthesis of new content
- Reconstruction
  - Can fill-in the details
- Rendering and Compaction
  - Can compress the model and lead to faster rendering times and smaller model footprint

## 3 Approaches for Inverse Procedural Modeling

### 3.1 Low-level Priors: Inferring Shape Grammars

Render the Possibilities
SIGGRAPH2016
THE 43RD INTERNATIONAL CONFERENCE AND EXHIBITION ON Computer Graphics & Interactive Techniques

# Inverse Procedural Modeling of 3D Models for Virtual Worlds

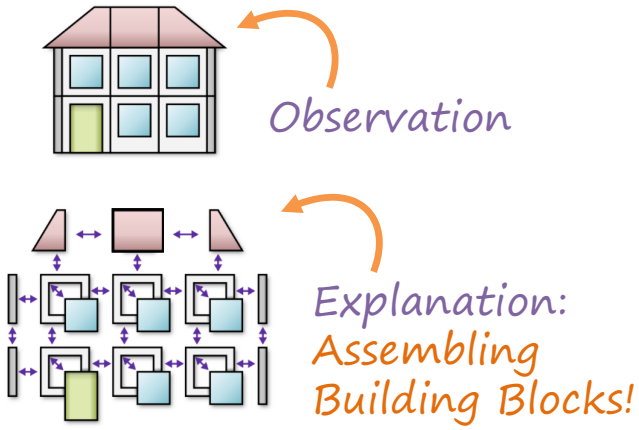
JOHANNES GUTENBERG UNIVERSITÄT MAINZ
PURDUE UNIVERSITY

Daniel Aliaga Ilke Demir
Bedrich Benes Michael Wand

# Outline

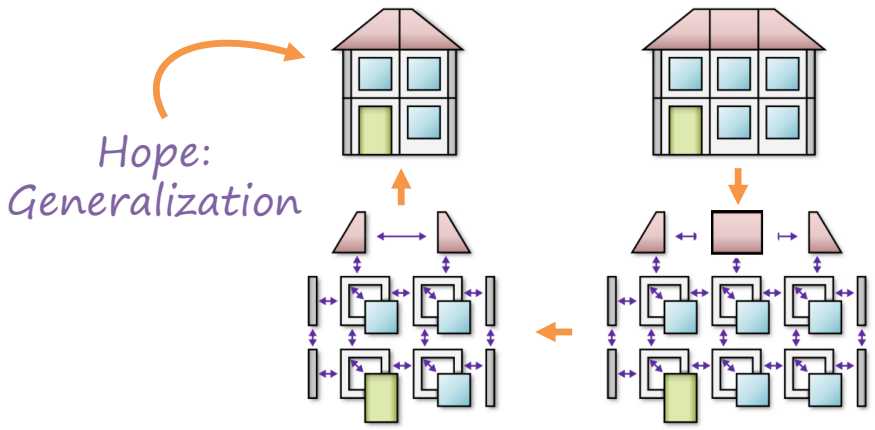
- Introduction
- Fundamentals of IPM
- **IPM Approaches**
  - **Low-level Priors: Inferring Shape Grammars**
  - High-level Priors: Inferring Grammar Parameters
- Inverse Procedural Modeling Domains
  - Facades & Layouts
  - Vegetation
  - Urban Areas
- Conclusions, Challenges, Open Problems

# Inspiration: Compositional Modeling




- Redundancy: Few types of building blocks
- Copy & paste, “simple” transformations

# Inspiration: Compositional Modeling




- Infer new “plausible” models

# Low-Level Priors

- Definition:
  - Geometric formulations with minimal assumptions
- Properties of approaches:
  - Exploiting redundancy in shapes
  - Extracting building blocks & assembly rules
  - Assuming no semantic / domain knowledge

# Low-Level Priors

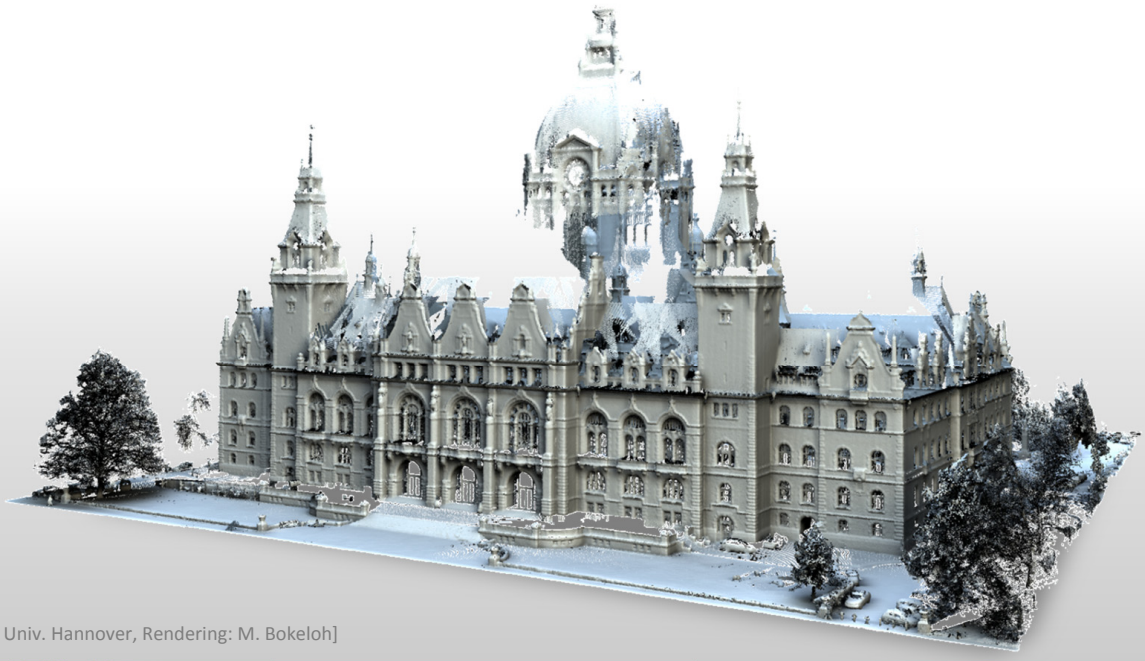
Two main steps:

- Extracting Building Blocks

- Finding Rules & Synthesis Algorithms

# Building Blocks

# What are the pieces?



[Data set: C. Brenner, IKG Univ. Hannover, Rendering: M. Bokeloh]

# Challenges

- **Modeling:** How to define building blocks?
  - Redundancy relates to symmetry
  - Define notion of redundancy
- **Computation:** Two aspects
  - Matching: Detect similar pieces
  - Segmentation: Define pieces
- Chicken & egg problem

# Defining
# Building Blocks

# Symmetry

- Operations that leave $X$ intact

$$f(X) = X$$

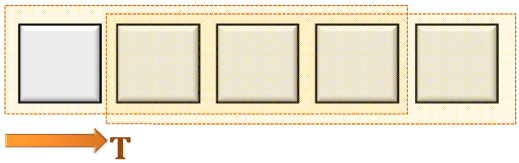
- Operations form a group


- Looking at the object:

  *Symmetry is the absence of Information*

# Symmetry Structure

## Pairwise Correspondences



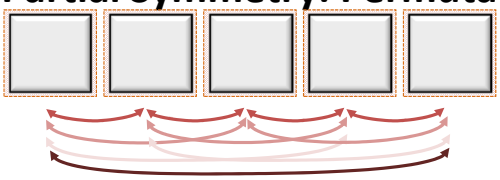

*Pairwise matches*

## Global Symmetry: Transformation Groups


*Regular transformations*

$\{T^i | i \in \mathbb{Z}\}$

## Partial Symmetry: Permutation or Pieces



*Building blocks*

# Symmetry as Redundancy

- What is symmetry?
  - Redundancy in a phenomenon
  - Can transform the data without (relevant) change
  - Compressible by parts & group actions
- Geometric symmetry
  - Global symmetry: model invariant under regular transformations
  - Partial symmetry: pieces can be exchanged without change

# Computing
# Building Blocks

# Building Blocks

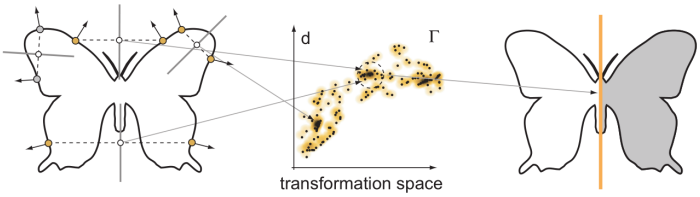
- **Direct approaches**
  - Early work: Region growing [Mitra et al. 2006], ff.
  - Regularity (grids) [Pauly et al. 2008], ff.
  - Microtiling: r-similarity [Kalojanov et al. 2012]
- **Optimization approaches**
  - Co-occurrence clustering [Li et al. 2015]
  - Set-cover constraints [Demir et al. 2015]
  - Further appr., e.g. primitives [Toshev et al. 2010]

# Computing Building Blocks

## Region Growing

# Partial Symmetry Detection


Transformation voting
+ region growing




**N. Mitra, L. Guibas, M. Pauly:**
Partial and Approximate Symmetry Detection for 3D Geometry. *Siggraph 2006.*

# Partial Symmetry Detection

**M. Bokeloh, A. Berner, M. Wand, H.-P. Seidel, A. Schilling:**
Symmetry Detection Using Line Features. *Eurographics 2009.*

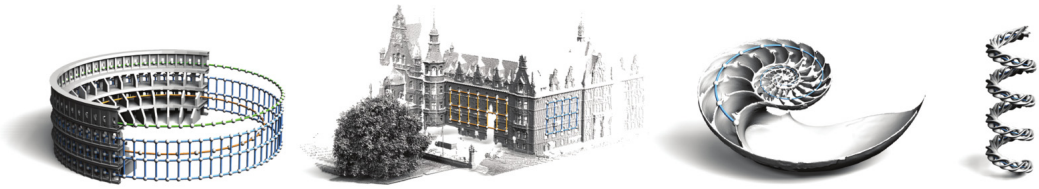
- Feature matching
- Region growing

[Data set: C. Brenner, IKG Univ. Hannover, Rendering: M. Bokeloh]

# Computing Building Blocks

## Regularity

# Regular Structure (Grids)



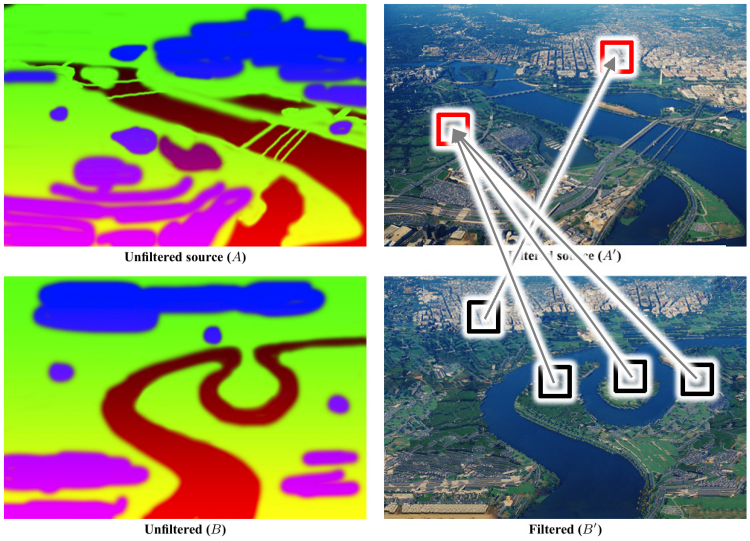
**M. Pauly, N. Mitra, J. Wallner, H. Pottmann, L. Guibas:**
Discovering Structural Regularity in 3D Geometry. *Siggraph 2008.*

- Transformation voting
- Discover transformation groups
- Commutative groups $\widehat{=}$ grids

# Computing Building Blocks

## r-Similarity & Microtiles

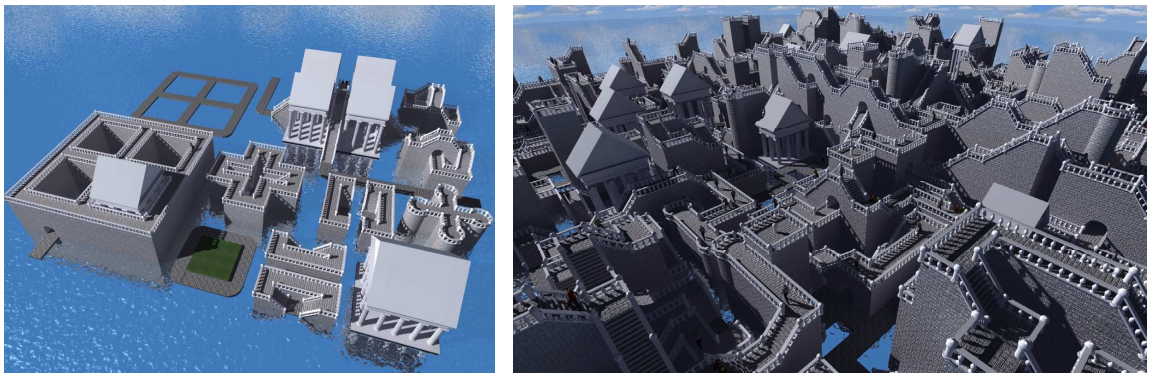
# Motivation: MRF Texture Synthesis




$k \times k$ pixel neighborhoods match

Example paper
(large body of literature)
**A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, D. H. Salesin:**
Image Analogies, *Siggraph 2001*.

# Transfer to 3D Geometry



**P. Merrell:** Example-Based Model Synthesis, *I3D 2007*.

**See also:**

**D. Cohen-Or, A. Sharf, M. Alexa:** Context-Based Surface Completion, *Siggraph 2004.*

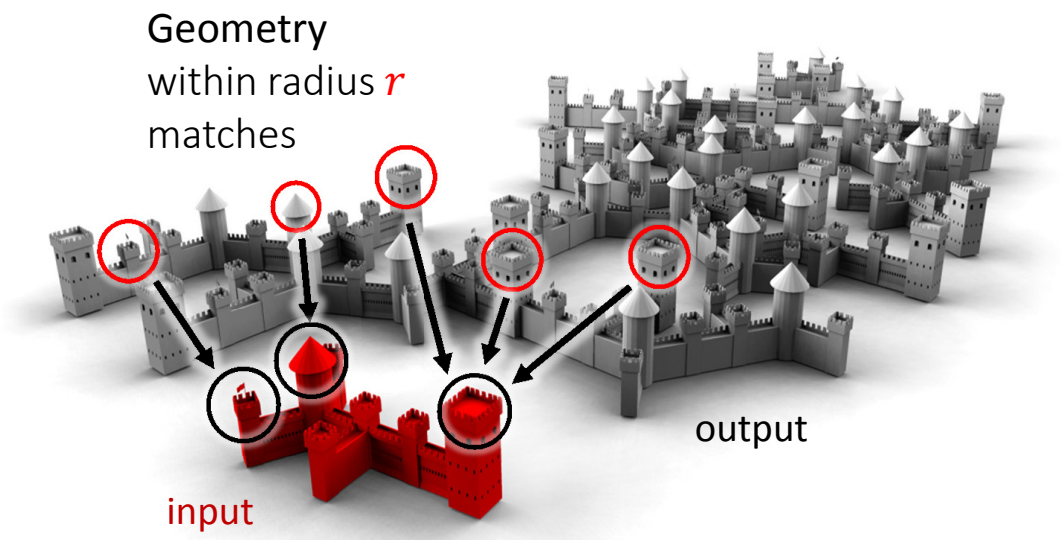
**P. Bhat, S. Ingram, G. Turk:** Geometric texture synthesis by example, *SGP 2004.*

(and more)

# Open Issues

- Problems with direct texture synthesis
  - Representation issues (regular grids? constrained geometry?)
  - Approximate solution difficult / leaves gaps
  - No structural insight
- Idea
  - Refine TS concept
  - “Exact” local similarity

# General 3D Geometry: *r*-Similarity

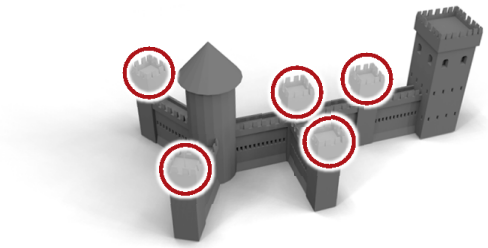


# Abstraction of *r*-Similarity

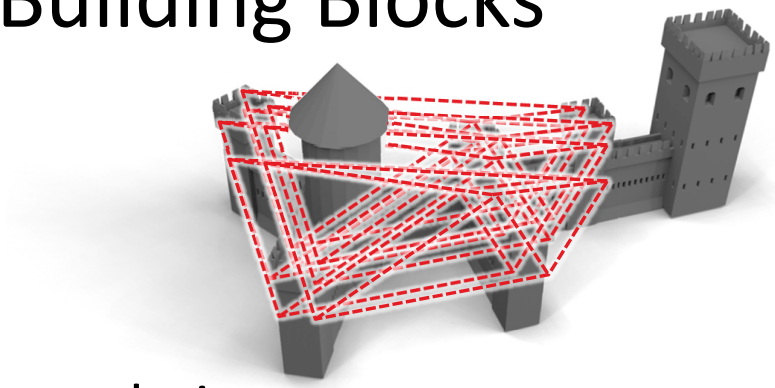
**Minimum requirement: equivalence relation**

$$\mathbf{x} \equiv_{\mathbf{T}} \mathbf{y}$$

- $\mathbf{x}$ matches $\mathbf{y}$ under transformation $\mathbf{T}$
- Symmetric, reflexive, transitive
- r-similarity model has this structure
  - Generalization possible (e.g.: graph matching)

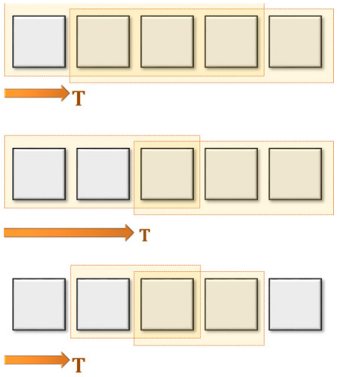


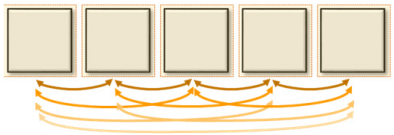
# Building Blocks



- Equivalence relation
- Start with correspondences
  - Fully-connected cliques of matching points
  - Cliques indicate building block types

# Alternative Definition



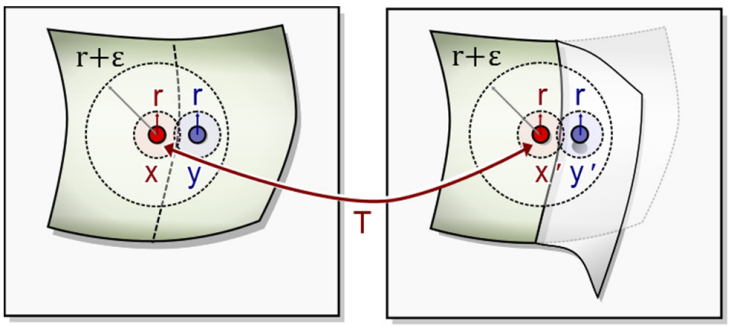
**Building blocks from "Microtiles"**

- Correspondences (equiv. relation)
- Building blocks:
  - Pieces always mapped together

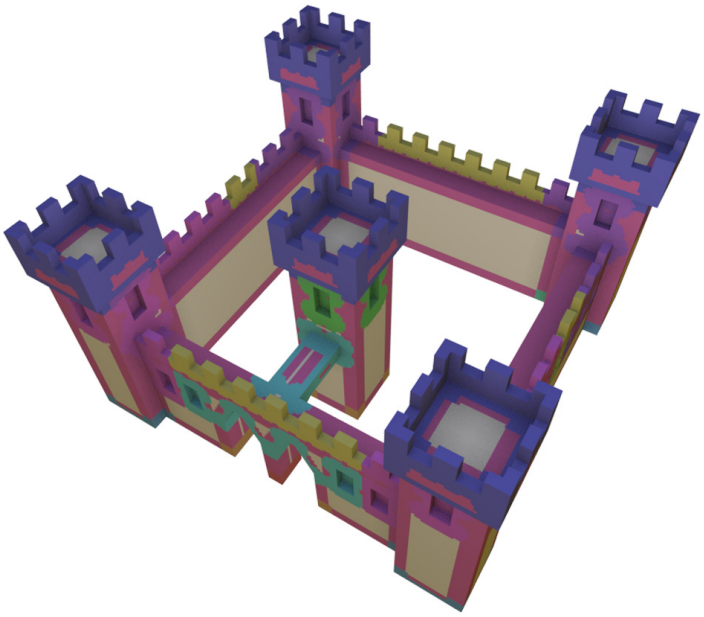
# Theory: Capture all r-Similar Shapes

**Inverse Modeling**

- All shapes $(r + \epsilon)$-similar to S can be constructed out of the $r$-microtiles of S.
  - For any $\epsilon > 0$
- The construction is unique.
- Neighborhood connection can be learned.



# Theory: Not Robust, Synthetic Data Only

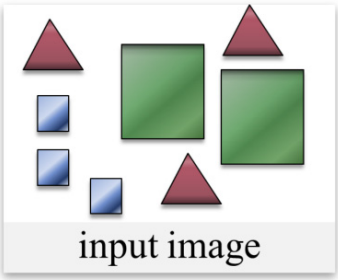


**J. Kalojanov, M. Bokeloh, M. Wand, L. Guibas, H.-P: Seidel, P. Slusallek:**
Microtiles: Extracting Building Blocks from Correspondences, *SGP 2012*.

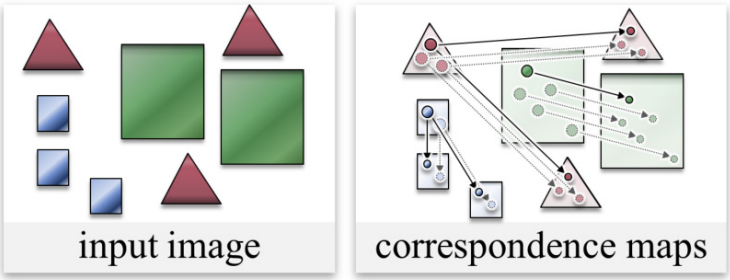
# Building Blocks: Optimization Approaches

## Co-Occurrence Clustering

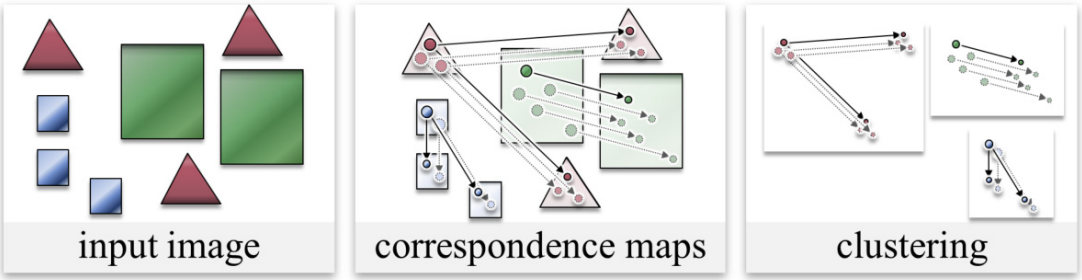
# Co-Occurrence Clustering



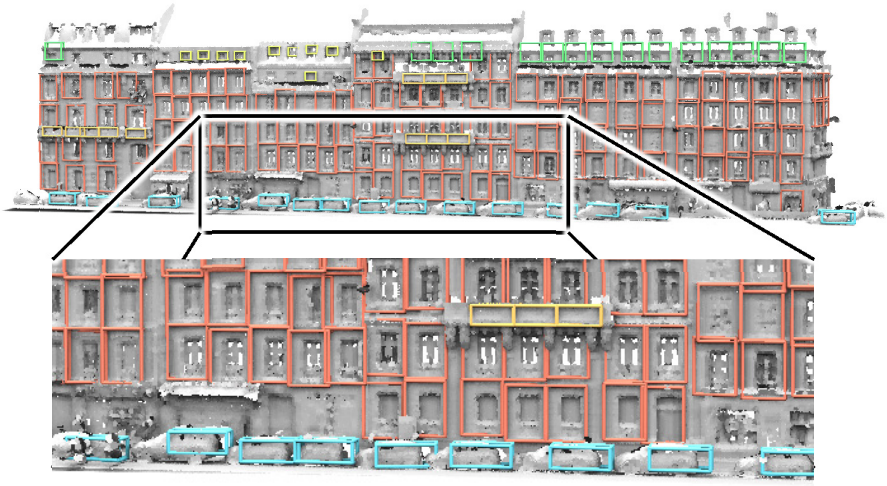
# Co-Occurrence Clustering



# Co-Occurrence Clustering

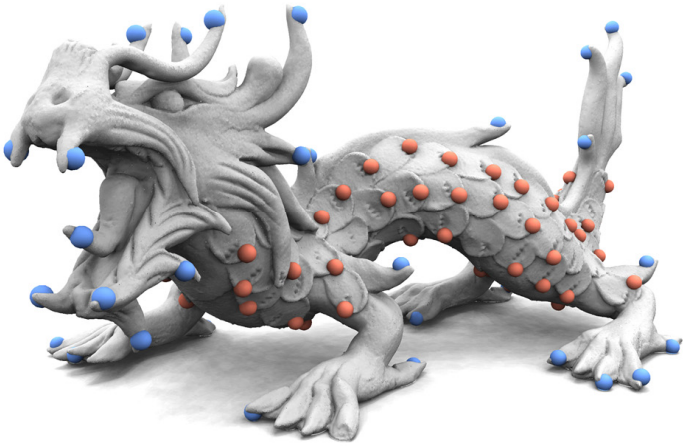


# Results



**C. Li, M. Wand, X. Wu, H.-P. Seidel:** Approximate 3D Partial Symmetry Detection Using Co-occurrence Analysis, 3DV 2015.

# Results



**C. Li, M. Wand, X. Wu, H.-P. Seidel:** Approximate 3D Partial Symmetry Detection Using Co-occurrence Analysis, 3DV 2015.

# Building Blocks: Optimization Approaches

## Set-Cover Model

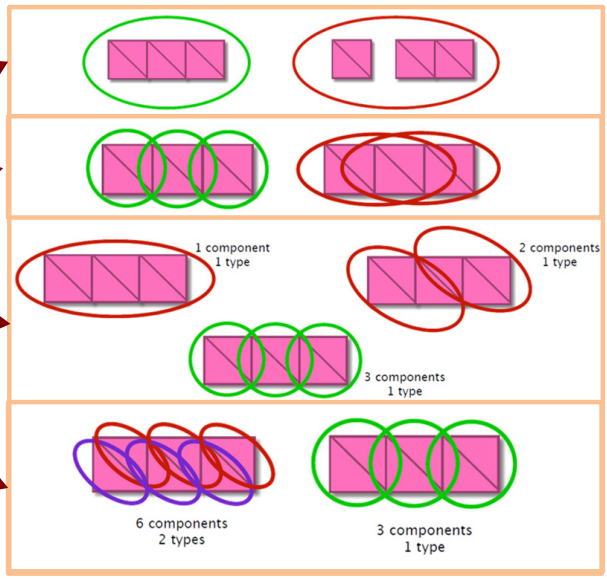
Alternative Algorithm:

# Partial Symmetry via Optimization

**Ilke Demir, Daniel Aliaga, Bedrich Benes**:
Coupled Segmentation And Similarity Detection For Architectural Models, Siggraph 2015.

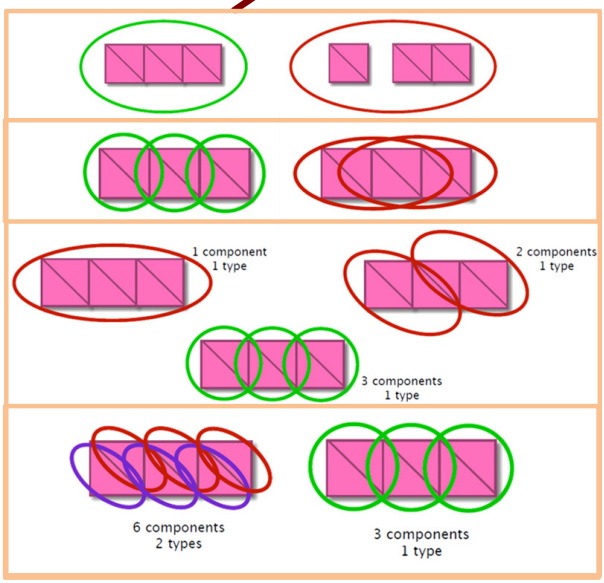
# Main Idea

- Global constraints
  - Contiguous building blocks
  - Exactly cover the model
  - Maximize instances
  - Minimize types
- Trade-off: Compact encoding
  - Reuse: Not too many types
  - Reuse: Many instances for each type



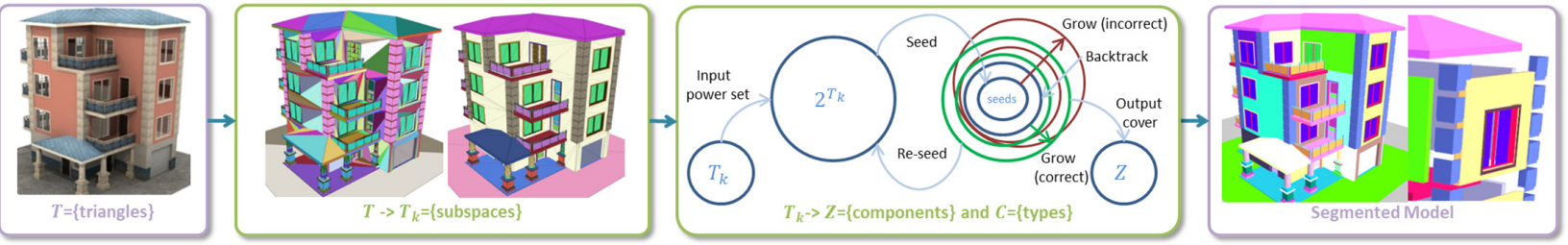
# Main Idea

Set cover problem! (NP-hard)




- Global constraints
  - Contiguous building blocks
  - Exactly cover the model
  - Maximize instances
  - Minimize types
- Trade-off: Compact encoding
  - Reuse: Not too many types
  - Reuse: Many instances for each type

# Overview

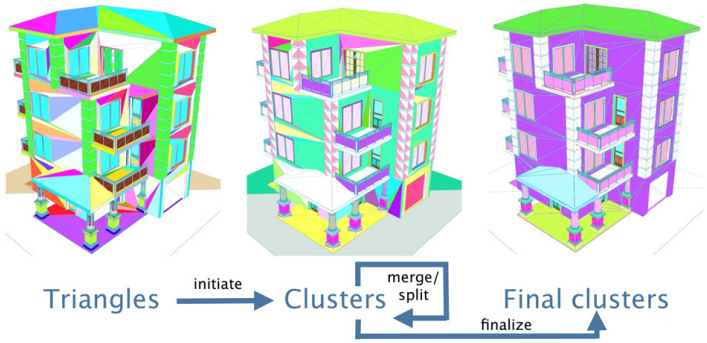



- Input: triangle soup
- Output: labeled segments (terminals!)
- Formulate as a set cover problem over triangles.
- Reduce the search space by finding partially similar clusters.
- Find the cover by volume growing per cluster.

# Algorithm

**Search Space Reduction**

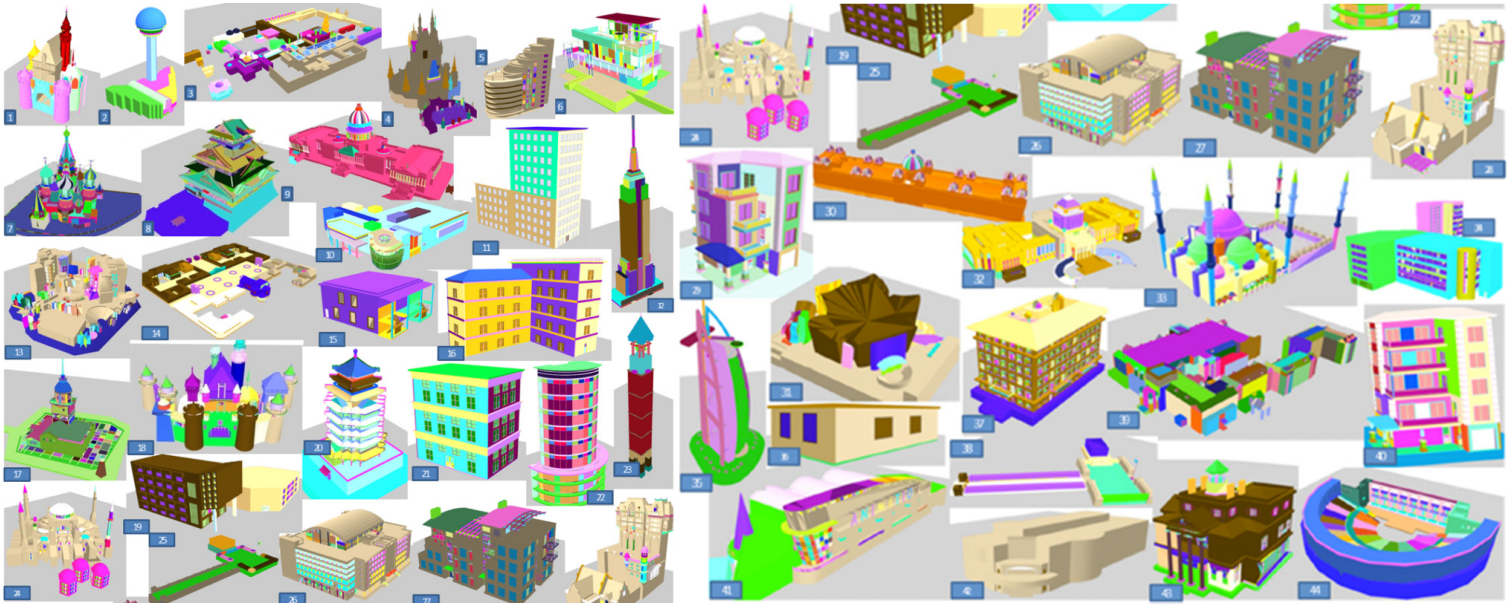
- Initiate by similarity
- Merge/split by adjacency




**Randomized Search**

- Within each search space (repeat until converges);
  - Seed triangles (smart selection on seeds!)
  - Synchronously grow convex-hulls with least deviating change
  - Compute weights and sum
- Backtrack to previous cases if not converging

# Results



# Results


|Z|: 26
|C|: 8
T: 23
|Z|: 451
|C|: 10
T: 35
|Z|: 207
|C|: 13
T: 101
|Z|: 268
|C|: 12
T: 231
|Z|: 21
|C|: 7
T: 34
|Z|: 427
|C|: 8
T: 96
|Z|: 197
|C|: 17
T: 87
|Z|: 172
|C|: 12
T: 111
|Z|: 21
|C|: 9
T: 22
|Z|: 660
|C|: 6
T: 65
|Z|: 980
|C|: 18
T: 37
|Z|: 32
|C|: 14
T: 22
|Z|: 21
|C|: 6
T: 41
|Z|: 424
|C|: 8
T: 39
|Z|: 78
|C|: 10
T: 54
|Z|: 209
|C|: 11
T: 110
|Z|: 21
|C|: 6
T: 41
|Z|: 424
|C|: 8
T: 39
|Z|: 78
|C|: 10
T: 54
|Z|: 209
|C|: 11
T: 110
|Z|: 21
|C|: 7
T: 33
|Z|: 66
|C|: 7
T: 18
|Z|: 348
|C|: 16
T: 46
|Z|: 292
|C|: 14
T: 132
|Z|: 21
|C|: 10
T: 23
|Z|: 424
|C|: 15
T: 94
|Z|: 428
|C|: 14
T: 67
|Z|: 278
|C|: 38
T: 87
|Z|: 26
|C|: 10
T: 0
|Z|: 1050
|C|: 41
T: 0.08
|Z|: 454
|C|: 179
T: 1
|Z|: 1215
|C|: 502
T: 4

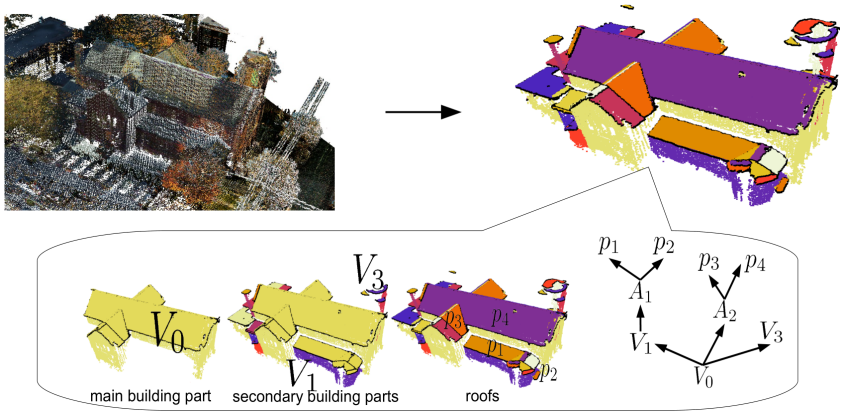
# Building Blocks: Optimization Approaches

## Other Approaches

# Primitive / Plane Detection

**A. Toshev, P. Mordohai, B. Taskar:**
Detecting And Parsing Architecture At City Scale From Range Data
CVPR 2010.

- Building point cloud as input, outputs a parse tree
- Planar patches = terminals
- Merge terminals = productions



# More Related Work

**More general, non IPM methods:**

- Conditional random field models
  - Kalogerakis et al., Siggraph 2010
- Geometric primitives as building blocks
  - Schnabel, Degnener, Klein, EG 2009
- Symmetry detection
  - Lipmann et al., Siggraph 2010
- Many more!

# Low-Level Priors

Two main steps:

- Extracting Building Blocks
- Finding Rules & Synthesis Algorithms

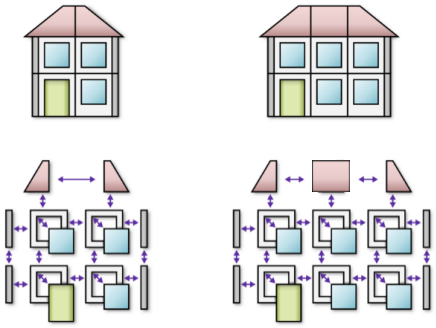
# Rules & Synthesis Algorithms

# How Difficult?

- **Strategies, sorted by worst-case costs**
  - Assembling tiles: undecidable problem
  - Double-cuts: efficient, less variability
  - Algebraic methods: grids
  - Interactively guided methods: small moves

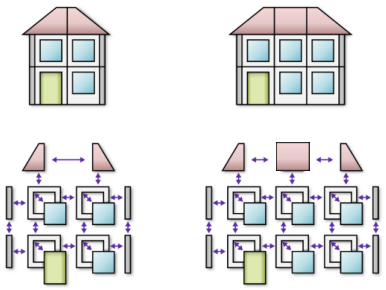
# Assembling Building Blocks

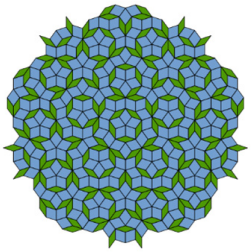
- **Shape grammar**
  - Tiling grammar
  - Glue building blocks at boundaries
- **Learning tiling grammars**
  - Very easy
  - Boundaries must match
    - Permit only “observed” combinations
    - Example data must show that combination makes sense



# Assembling Building Blocks

- **Synthesis / Assembly?**
  - How difficult is it to assemble pieces according to matching edges (dockers)?
  - Bad surprise!
  - Undecidable ("Domino problem")
    - Turing-complete
    - Emergent complexity
    - Example: Penrose tilings
      - globally non-periodic

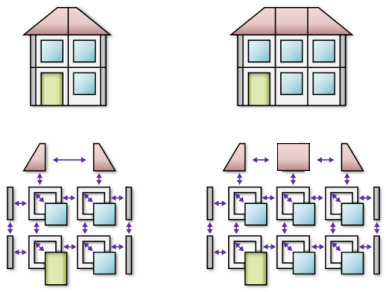




[Wikipedia, user "Inductiveload"]

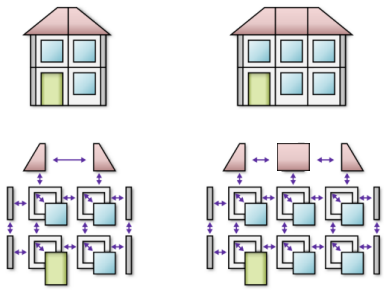
# Inverse Procedural Modeling

- Three approaches:
  - Context-free grammars
    - Rule-driven
    - “Double Cut”
  - Regularity models
    - Grids
    - Interactive guidance
  - MDL Principle



# Inverse Procedural Modeling

- Three approaches:
  - Context-free grammars
    - Rule-driven
    - "Double Cut"
  - Regularity models
    - Grids
    - Interactive guidance
  - MDL Principle



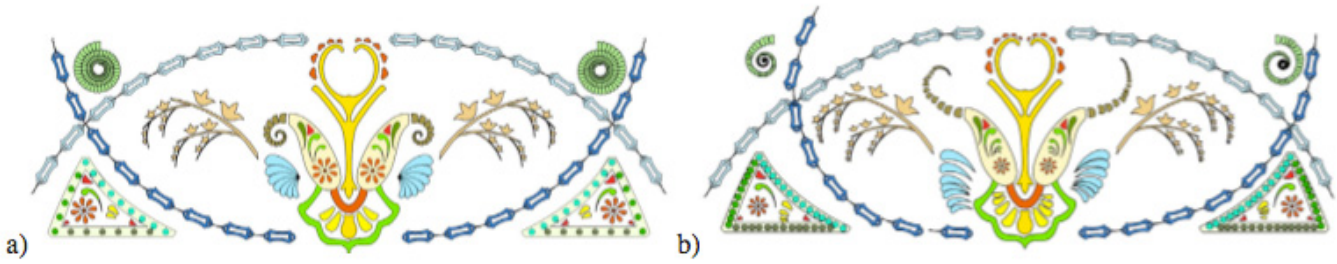
# Context-Free Grammars

**O. Šťava, B. Beneš, R. Měch, D. Aliaga, P. Kryštof:**
Inverse Procedural Modeling by Automatic Generation of L-systems,
*Eurographics 2010.*
← rule driven, vector graphics

**Martin Bokeloh, Michael Wand, Hans-Peter Seidel:**
A Connection between Partial Symmetry and Inverse Procedural Modeling,
*Siggraph 2010.*
← 3D meshes, "double cut"

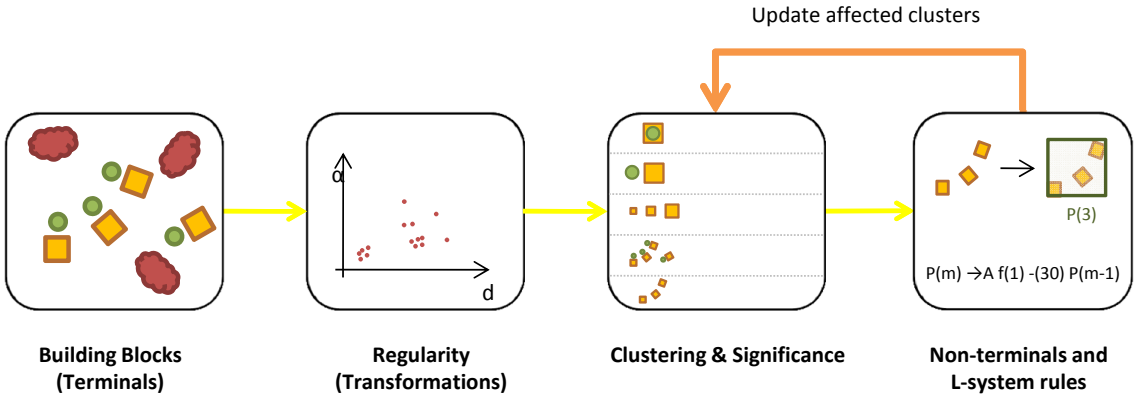
**H. Liu, U. Vimont, M. Wand, M.-P. Cani, S. Hahmann, D. Rohmer, N. J. Mitra:**
Replaceable Substructures for Efficient Part-Based Modeling,
*Eurographics 2015.*
← non-rigid part graphs, "double cut"

# Vector Graphics [Stava et al. 2010]



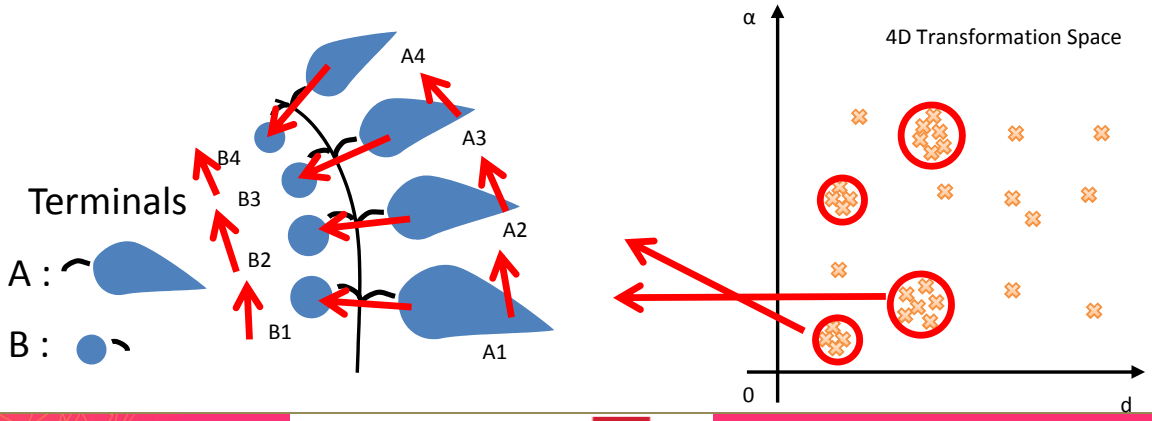

**O. Šťava, B. Beneš, R. Měch, D. Aliaga, P. Kryštof:**
Inverse Procedural Modeling by Automatic Generation of L-systems,
*Eurographics 2010.*

# Approach


Update affected clusters
α
d
P(3)
P(m) →A f(1) -(30) P(m-1)
Building Blocks (Terminals)
Regularity (Transformations)
Clustering & Significance
Non-terminals and L-system rules

# Transformation Analysis

- Clustering in the transformation space
  - Large clusters ~ significant transformations

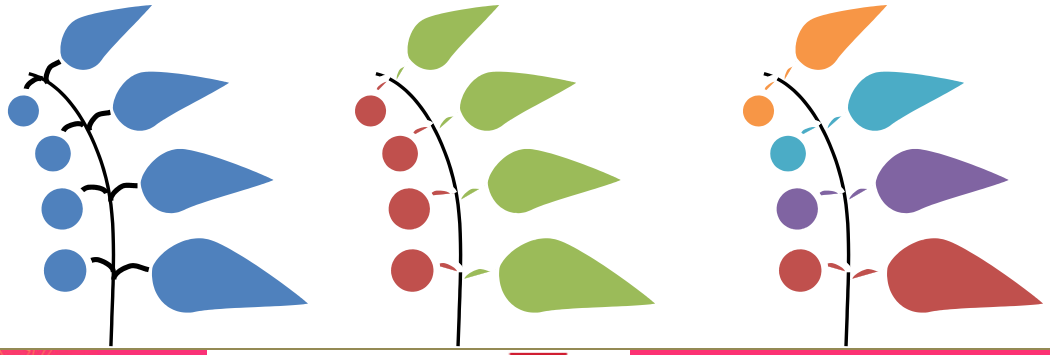


# Cluster Analysis

– Weighted cluster importance function

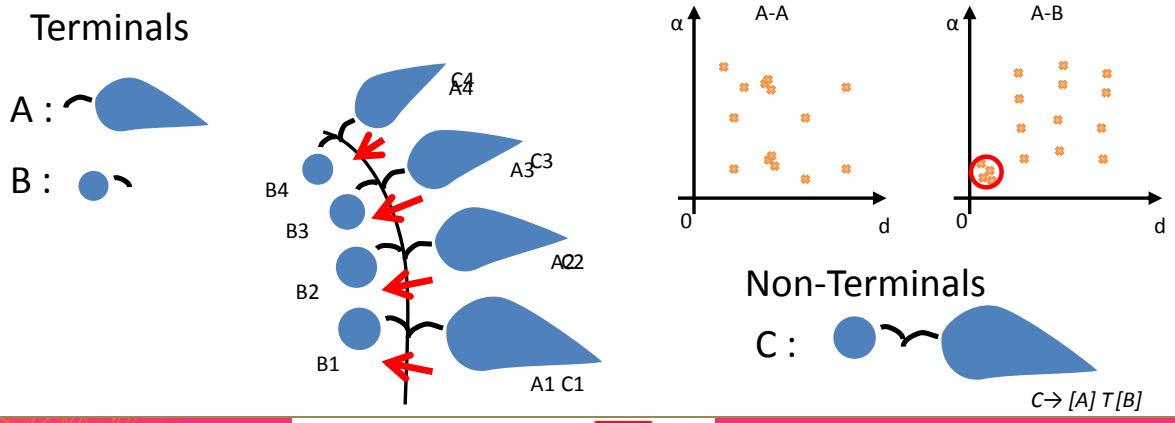
- Weights determine the final rules

*Prefer sequences* *Prefer proximity*



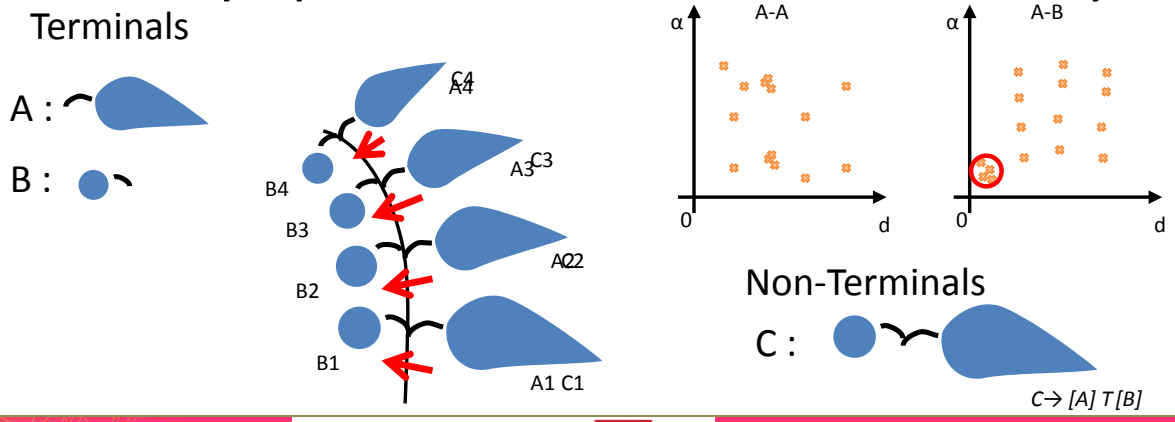
# L-system Generation

- New rule = new non-terminal symbol



# L-system Generation

- Clusters no longer valid
  - Iteratively update with new non-terminal symbol



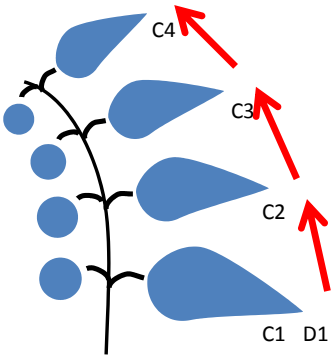
# L-system Generation

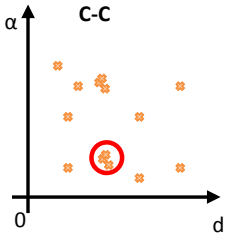
- Generate new rules until there are no clusters
  - Axiom → Last non-terminal

Terminals

A :

B :




Non-Terminals

C :

D :

Axiom: S → $T_S$ D(3)

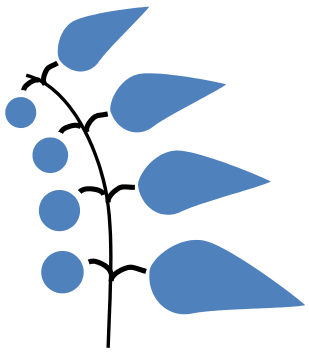
# L-system Generation

- Final L-system

Terminals

A :

B :



***L-system***

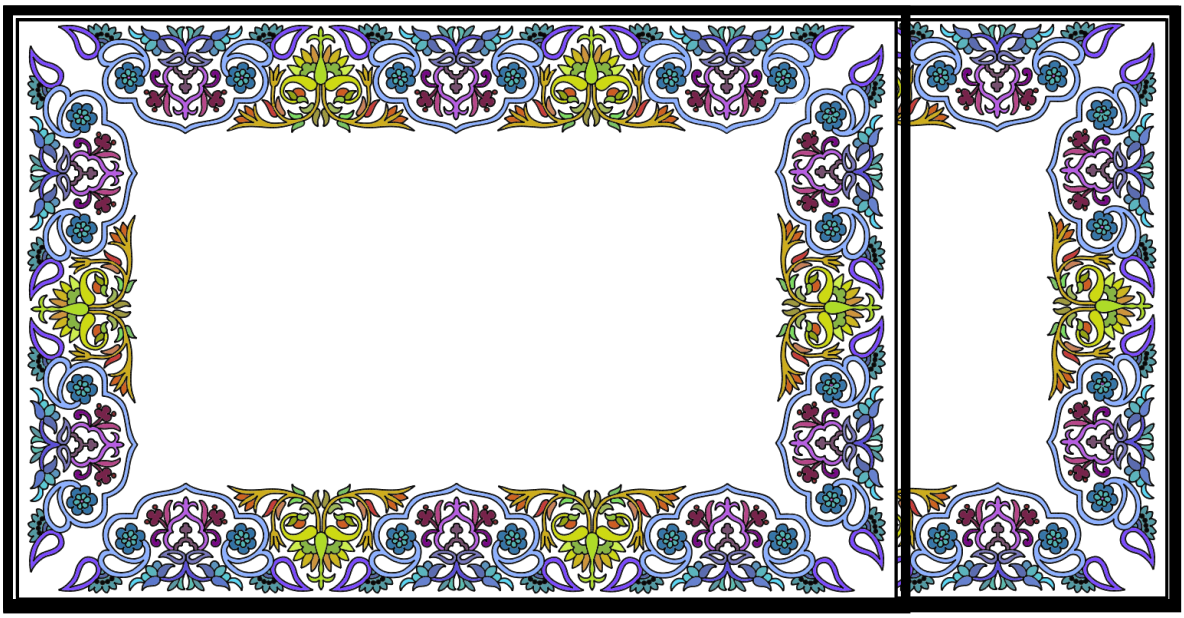
$C(m) \rightarrow [A]\ T_1\ [B]$

$D(m) : m>0 \rightarrow [C]\ T_2\ D(m-1)$

$m=0 \rightarrow [C]$

$S \rightarrow T_s\ [D(3)]$

# Results

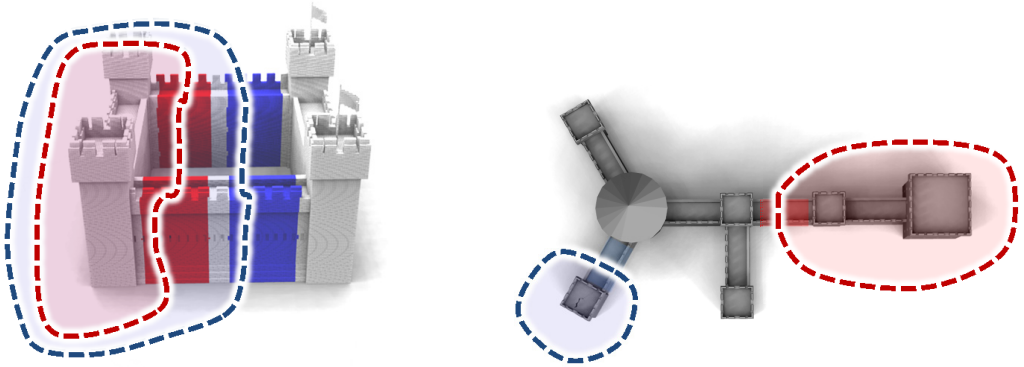


# Results

# “Double Cut” Algorithm

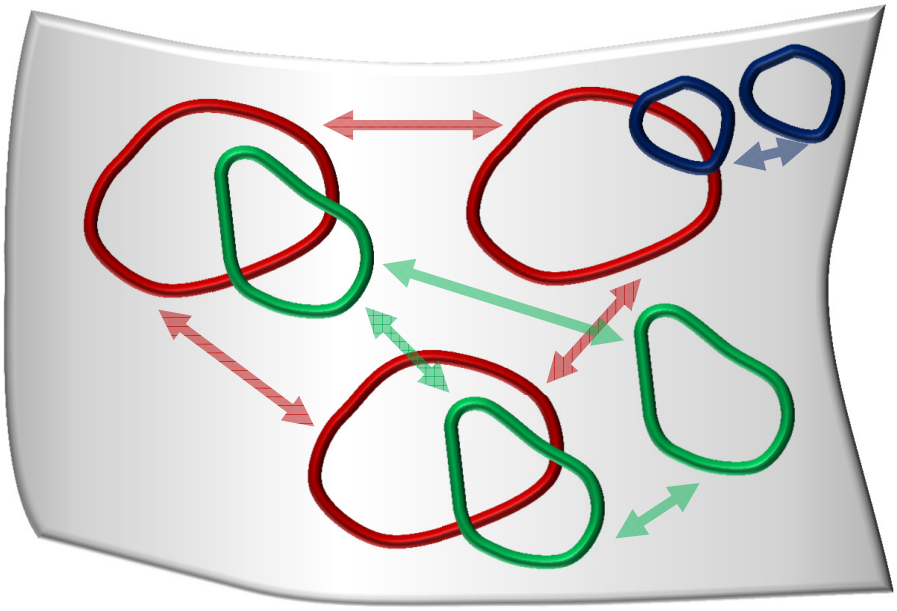
## a.k.a. “A Connection between Partial Symmetry and Inverse Procedural Modeling”

# Context-Free Grammars



- **Double-Cut Algorithm** [Bokeloh et al. 2010]
  - Enumerate cut pairs
    - No overlapping cut boundaries
  - Enumerates modification operations
  - Fast (low-order polynomial time)

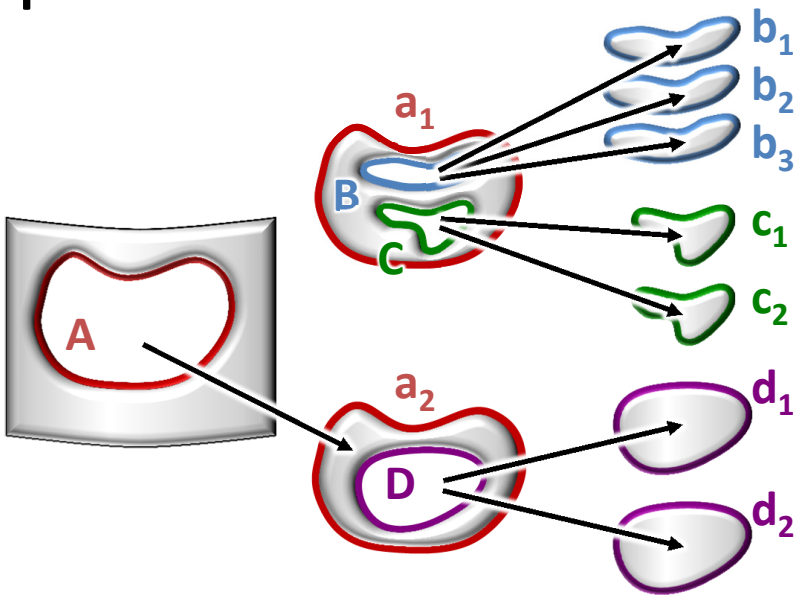
# Symmetry Boundaries



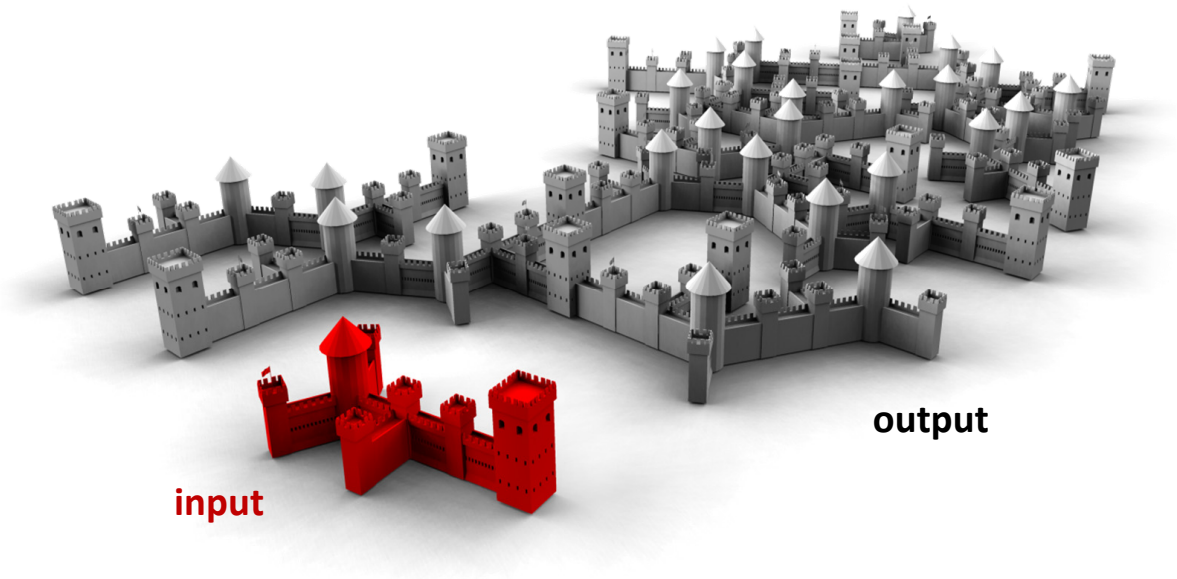
# Shape Grammar

Grammar:

| | | | | |
|---|---|---|---|---|
| A | → | $a_1$ B C | \| $a_2$ D | |
| B | → | $b_1$ | \| $b_2$ | \| $b_3$ |
| C | → | $c_1$ | \| $c_2$ | |
| D | → | $d_1$ | \| $d_2$ | |



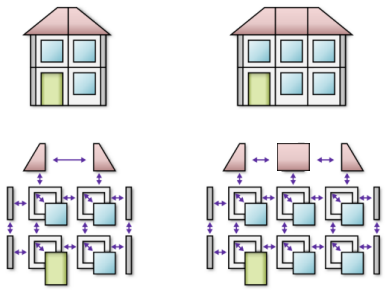
# Results



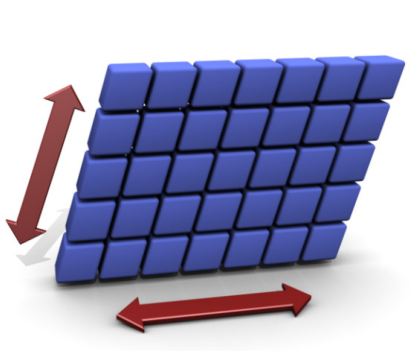
# Results

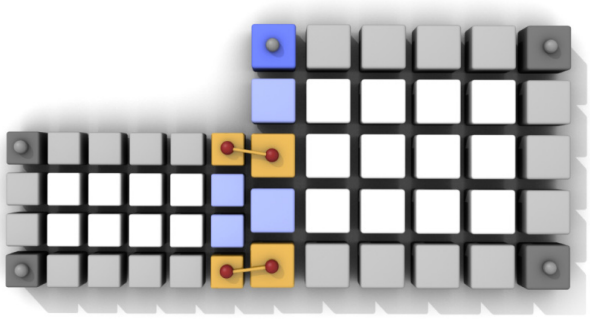
# Inverse Procedural Modeling

- Three approaches:
  - Context-free grammars
    - Rule-driven
    - "Double Cut"
  - Regularity models
    - Grids
    - Interactive guidance
  - MDL Principle

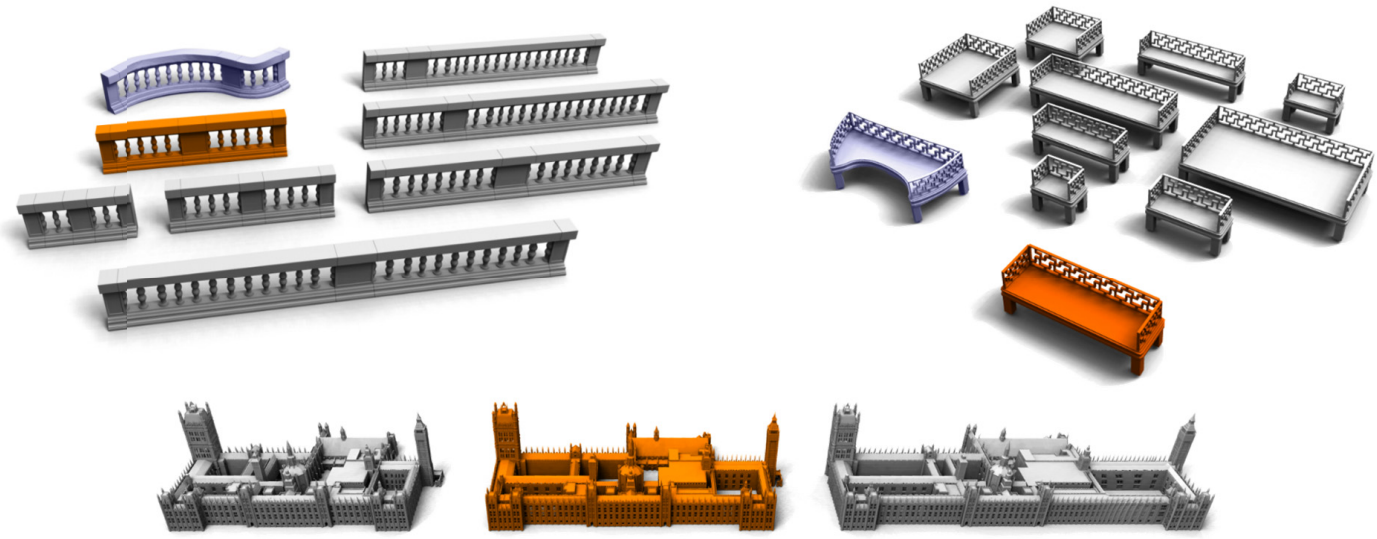


# Preserve Grid Structures



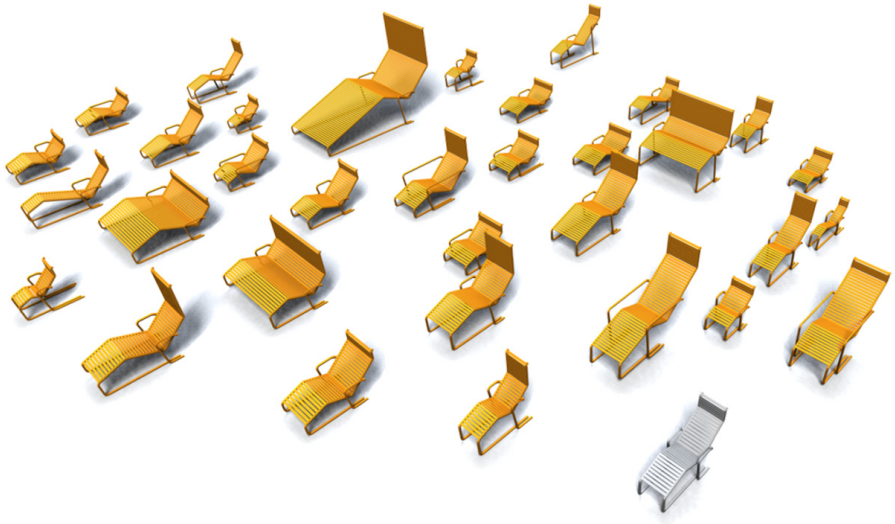


# Regularity Aware Deformation



**M. Bokeloh, M. Wand, V. Koltun, H.-P. Seidel:** Pattern-Aware Shape Deformation Using Sliding Dockers. *Siggraph Asia 2011.*

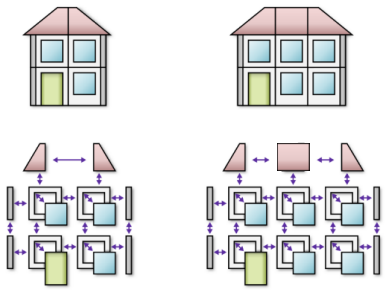
# Algebraic Shape Editing



**M. Bokeloh, M. Wand, V. Koltun, H.-P. Seidel:** An Algebraic Model for Parameterized Shape Editing. *Siggraph 2012.*

# Inverse Procedural Modeling

- Three approaches:
  - Context-free grammars
    - Rule-driven
    - "Double Cut"
  - Regularity models
    - Grids
    - Interactive guidance
  - MDL Principle




Render the Possibilities
SIGGRAPH2016
THE 43RD INTERNATIONAL
CONFERENCE AND EXHIBITION ON
&
Computer Graphics
Interactive Techniques
24-28 JULY
ANAHEIM, CALIFORNIA

Render the Possibilities
SIGGRAPH2016
THE 43RD INTERNATIONAL CONFERENCE AND EXHIBITION ON Computer Graphics & Interactive Techniques

# Inverse Procedural Modeling of 3D Models for Virtual Worlds

JOHANNES GUTENBERG UNIVERSITÄT MAINZ
PURDUE UNIVERSITY

Daniel Aliaga Ilke Demir
Bedrich Benes Michael Wand

# Outline

- Introduction
- Fundamentals of IPM
- **IPM Approaches**
  - Low-level Priors: Inferring Shape Grammars
  - **High-level Priors: Inferring Grammar Parameters**
- Inverse Procedural Modeling Domains
  - Facades & Layouts
  - Vegetation
  - Urban Areas
- Conclusions, Challenges, Open Problems

# IPM with High-level Priors

- Assumptions & Formulation
- MCMC
  - Review
  - Different approaches
- Learning
  - Review
  - Neural Networks

# IPM with High-level Priors

- **Assumptions & Formulation**
- MCMC
  - Review
  - Different approaches
- Learning
  - Review
  - Neural Networks

# Inferring Grammar Parameters

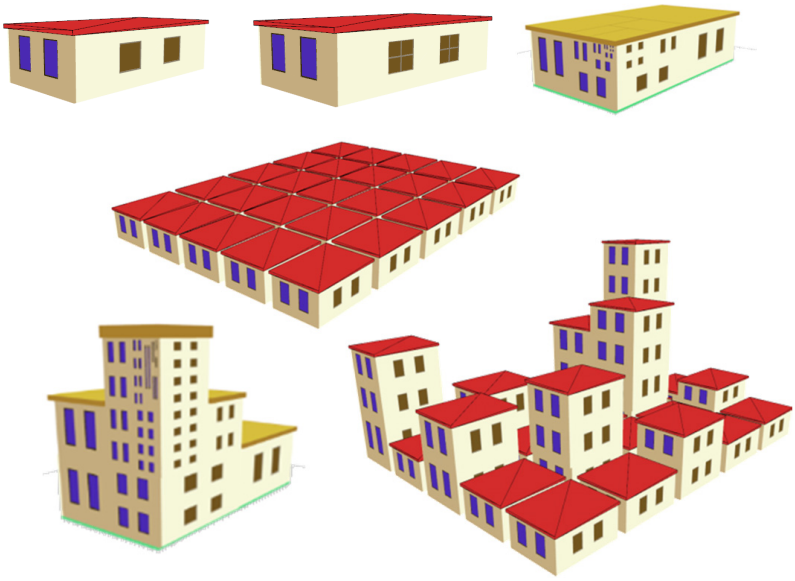
- Assumption:
  - We have a grammar!

```
Rule Building(S) = {                  // CREATE ENTIRE BUILDING
    a = split(S, 0.0, 0.0, 0.80, 4) // roof terminal
    c = split(S, 1.0, 1.0, 0.01, 5) // floor terminal
    B = split(S, 1.0, 1.0, 0.80, 5) // wall bbox
    D = split(B, 1.0, 0.1, 1.00, 5) // front wall bbox
    E = split(B, 0.0, 0.9, 0.00, 4) // back wall bbox
    F = split(B, 0.1, 1.0, 1.00, 5) // left wall bbox
    G = split(B, 0.9, 0.0, 0.00, 4) // right wall bbox
    R3(D); R3(E);                     // front/back rule
    R4(F); R4(G);                     // left/right rule
}
Rule R3(A) = {                        // CREATE FRONT/BACK WALL
    b = split(A, 0.0, 0.0, 0.00, 4) // wall terminal
    C = split(A, 0.4, 1.0, 0.75, 5)
    D = split(C, 0.6, 0.3, 0.00, 4) // left window bbox
    E = split(A, 0.6, 1.0, 0.75, 6)
    F = split(E, 0.3, 1.0, 0.30, 1) // right window
    R1(D); R1(F);                     // window rule
}
Rule R1(A) = {
    // perform splits for frame and window
}
[...and add'l text for R2 and R4]
```

# Inferring Grammar Parameters

- Assumption:
  - We have a grammar!

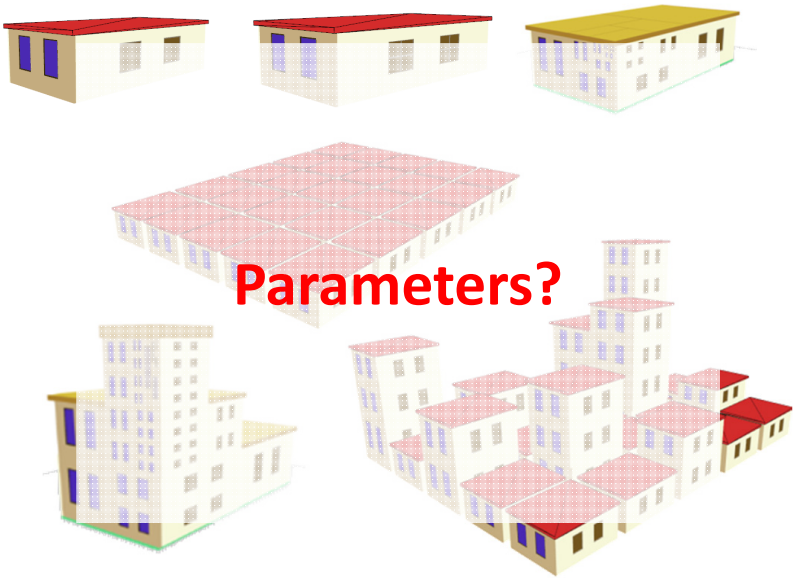
```
Rule Building(S) = {                  // CREATE ENTIRE BUILDING
    a = split(S, 0.0, 0.0, 0.80, 4) // roof terminal
    c = split(S, 1.0, 1.0, 0.01, 5) // floor terminal
    B = split(S, 1.0, 1.0, 0.80, 5) // wall bbox
    D = split(B, 1.0, 0.1, 1.00, 5) // front wall bbox
    E = split(B, 0.0, 0.9, 0.00, 4) // back wall bbox
    F = split(B, 0.1, 1.0, 1.00, 5) // left wall bbox
    G = split(B, 0.9, 0.0, 0.00, 4) // right wall bbox
    R3(D); R3(E);                     // front/back rule
    R4(F); R4(G);                     // left/right rule
}
Rule R3(A) = {                        // CREATE FRONT/BACK WALL
    b = split(A, 0.0, 0.0, 0.00, 4) // wall terminal
    C = split(A, 0.4, 1.0, 0.75, 5)
    D = split(C, 0.6, 0.3, 0.00, 4) // left window bbox_
    E = split(A, 0.6, 1.0, 0.75, 6)
    F = split(E, 0.3, 1.0, 0.30, 1) // right window
    R1(D); R1(F);                     // window rule
}
Rule R1(A) = {
    // perform splits for frame and window
}
[...and add'l text for R2 and R4]
```



# Inferring Grammar Parameters

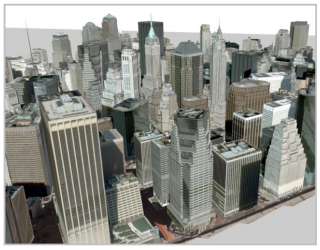
- Assumption:
  - We have a grammar!

```
Rule Building(S) = {                  // CREATE ENTIRE BUILDING
    a = split(S, 0.0, 0.0, 0.80, 4) // roof terminal
    c = split(S, 1.0, 1.0, 0.01, 5) // floor terminal
    B = split(S, 1.0, 1.0, 0.80, 5) // wall bbox
    D = split(B, 1.0, 0.1, 1.00, 5) // front wall bbox
    E = split(B, 0.0, 0.9, 0.00, 4) // back wall bbox
    F = split(B, 0.1, 1.0, 1.00, 5) // left wall bbox
    G = split(B, 0.9, 0.0, 0.00, 4) // right wall bbox
    R3(D); R3(E);                     // front/back rule
    R4(F); R4(G);                     // left/right rule
}
Rule R3(A) = {                        // CREATE FRONT/BACK WALL
    b = split(A, 0.0, 0.0, 0.00, 4) // wall terminal
    C = split(A, 0.4, 1.0, 0.75, 5)
    D = split(C, 0.6, 0.3, 0.00, 4) // left window bbox_
    E = split(A, 0.6, 1.0, 0.75, 6)
    F = split(E, 0.3, 1.0, 0.30, 1) // right window
    R1(D); R1(F);                     // window rule
}
Rule R1(A) = {
    // perform splits for frame and window
}
[...and add'l text for R2 and R4]
```



# Inferring Grammar Parameters

- Assumption:
  - We have a grammar!
- Derivation needs guidance.
  - Examples
  - User interaction
  - Metrics




# IPM as Optimization

Bayesian Inference:

- consider space of derivations $\delta \in \Delta(G)$
- define model prior $\pi(\delta)$
- take in some user specification $I$
- formulate likelihood function $L(I|\delta)$
- maximize $p(\delta|I) \propto L(I|\delta)\pi(\delta)$

# IPM as Optimization

- Looking for best parameters for model *M.*
  - Cleverly search the space
    - MCMC [**Talton10**, Talton12, **Vanegas12, Ritchie15**]
    - Lattice parsing [Martinovic13]
  - Learn the space
    - NNs [**Yumer15**, **Nishida16**, Ritchie16]
    - Learning PDFs [Dang15]

# IPM with High-level Priors

- Assumptions & Formulation
- **MCMC**
  - **Formulation**
  - **Different approaches**
- Learning
  - Review
  - Neural Networks

# MCMC Review

A *Markov Chain* is a sequence of random variables $X1, X2, \ldots$ with the Markov Property:

$$P(X_n - x | X_{n-1} - x_{n-1}, \ldots, X_1 - x_1) = P(X_n = x | X_{n-1} = x_{n-1})$$

# MCMC Review

A *Markov Chain* is a sequence of random variables $X1, X2, \ldots$ with the Markov Property:

$$P(X_n - x | X_{n-1} - x_{n-1}, \ldots, X_1 - x_1) = P(X_n = x | X_{n-1} = x_{n-1})$$

- Memoryless

# MCMC Review

A *Markov Chain* is a sequence of random variables $X1, X2, \ldots$ with the Markov Property:

$$P(X_n - x | X_{n-1} - x_{n-1}, \ldots, X_1 - x_1) = P(X_n = x | X_{n-1} = x_{n-1})$$

- Memoryless
- Properly constructed, each $X_i \equiv p(X)$ where $p(X)$ is the *stationary distribution* of the chain.

# MCMC Review

Markov chain Monte Carlo (MCMC) algorithms sample from a probability distribution based on constructing a Markov chain with the target distribution as its equilibrium distribution.

# MCMC Review

How to walk the solution space?

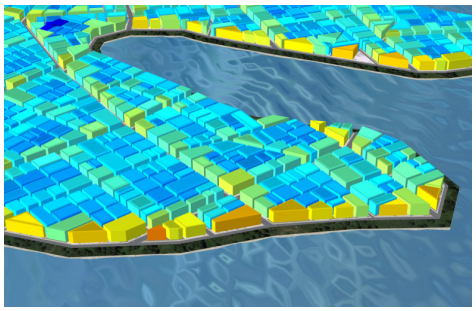
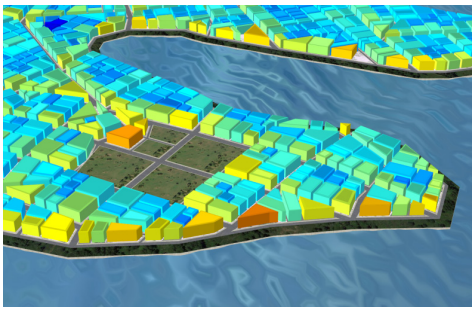
- Same problem: sample and evaluate.
  - Randomly walk: Metropolis Hastings
  - Randomly walk but multi-dim: rjMCMC
  - Ordered to allow mid-evals: SoSMCMC

# MCMC Approaches in IPM

- Random walks with local & global moves
  - Inverse Design of Urban Procedural Models (Vanegas et al. 2012)
- Metropolis Hastings with rjMCMC
  - Metropolis Procedural Modeling (Talton et al. 2010)
- Stochastically-ordered sequential MC
  - Controlling procedural modeling programs with stochastically-ordered sequential Monte Carlo (Ritchie et al. 2015)

# Inverse Design of Urban Procedural Models

- Model indicators -> PM parameters


# Inverse Design of Urban Procedural Models

- Model indicators -> PM parameters
- Error function for target indicators

$$E(\Phi_t, \Phi^*, \Gamma^*) = w_\Gamma \frac{1}{n} \sum_{\gamma \in \Gamma} \left( \frac{|\gamma - \gamma^*|}{\gamma_w^*} \right) + w_{\Phi_t} \frac{1}{m} \sum_{\phi \in \Phi} \left( \frac{|\phi - \phi^*|}{\phi_w^*} \right)$$

Distance to target indicator

Distance to desired input parameters

# Inverse Design of Urban Procedural Models

- Model indicators -> PM parameters
- Error function for target indicators
- Random walks with local & global moves

# Inverse Design of Urban Procedural Models

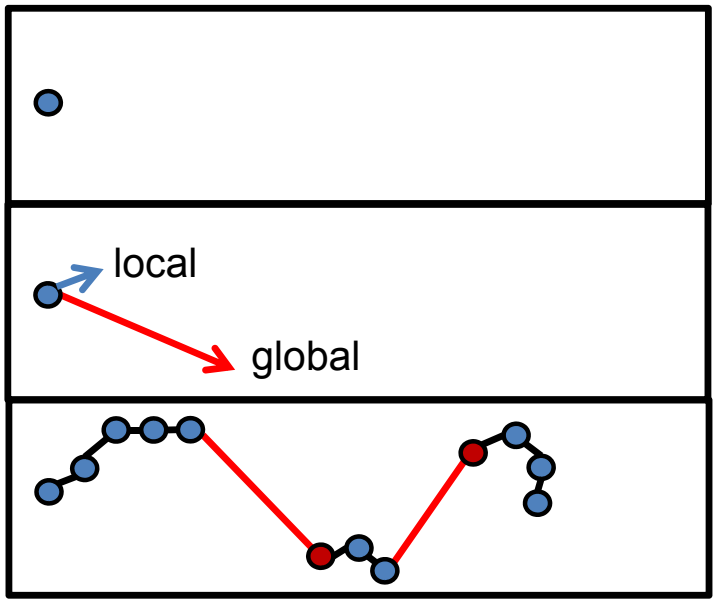
- Model indicators -> PM parameters
- Error function for target indicators
- Random walks with local & global moves
- Back propagation for interactive performance

# For each Markov Chain

1. Random seed and chain temperature

2. Local or global move

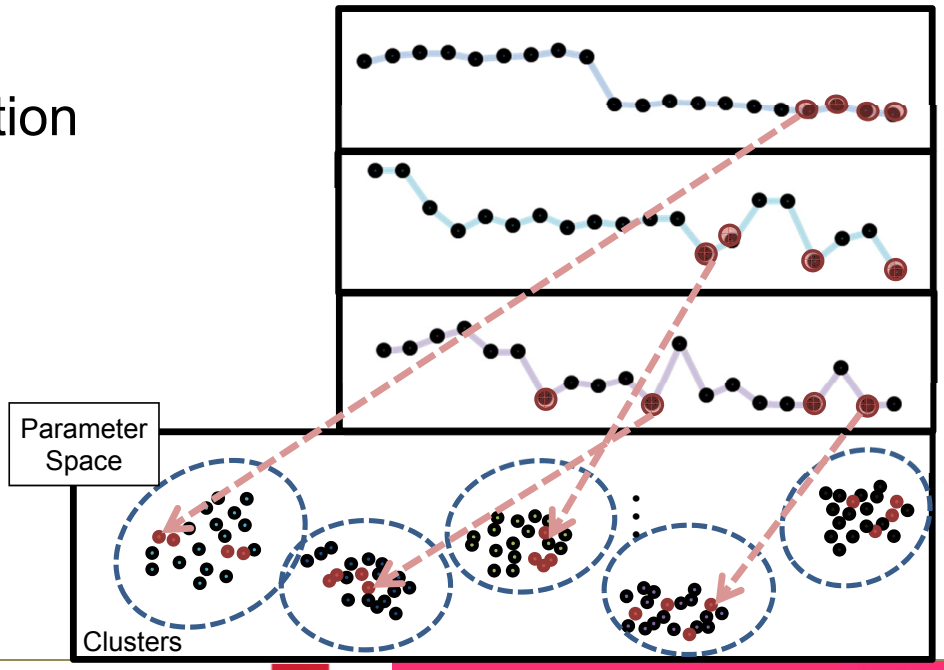
3. Random walk



# For each Markov Chain

4. Solutions selection

5. Clustering



(video)

## Result

- Initial model (forward)
- Buildings with egress (forward)
- Increase sun exposure (inverse)
- Increase interior light (inverse)
- Reduce distance to park (inverse)

# Metropolis Procedural Modeling

- Specification (sketch/volume/analytics) -> Best instance

# Metropolis - Hastings

Pick a proposal density $q(X_*|X_n)$ that can be sampled from

At each step:

- draw $X_* \equiv q(.|X_n)$
- Compute an acceptance probability

$$\alpha = \min\left(\frac{p(X_*)}{p(X_i)}\frac{q(X_n|X_*)}{q(X_*|X_n)}, 1\right)$$

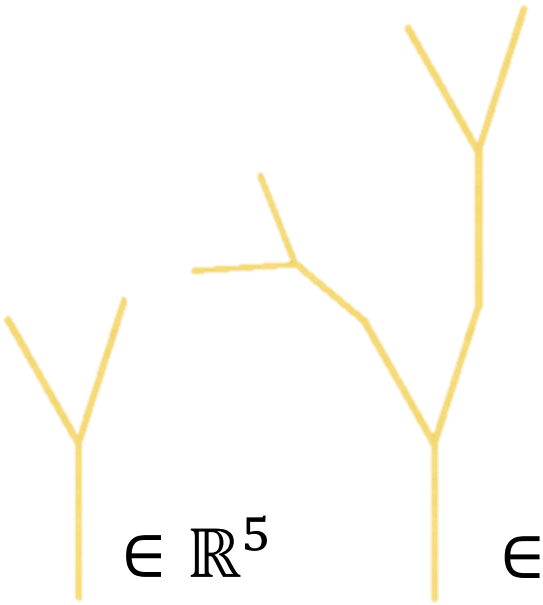
- Accept $X_{n+1} = X_*$ or reject $X_{n+1} = X_n$

# Metropolis - Hastings

MH algorithm;

- enables sampling efficiently from any function, defined over a fixed dimensional space, that can be evaluated.

# Metropolis - Hastings



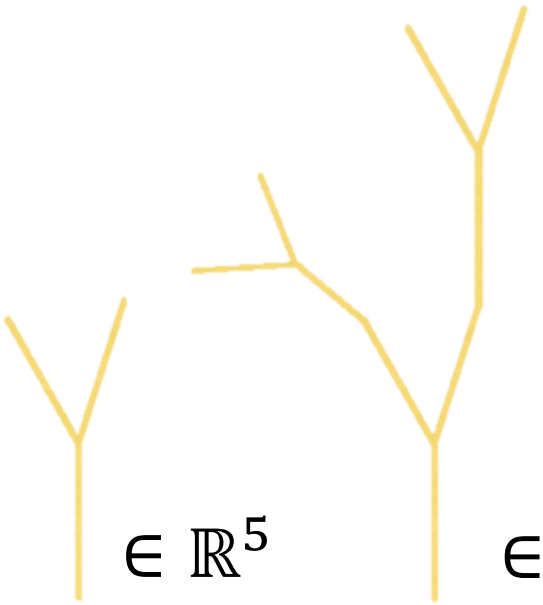
$\in \mathbb{R}^5$ $\in \mathbb{R}^{19}$

lives in $\in \mathbb{R}^{19}$

$$\alpha = \min\left(\frac{p(X_*)}{p(X_i)}\frac{q(X_n|X_*)}{q(X_*|X_n)}, 1\right)$$

lives in $\in \mathbb{R}^5$

# Metropolis - Hastings



$\in \mathbb{R}^5$ $\in \mathbb{R}^{19}$

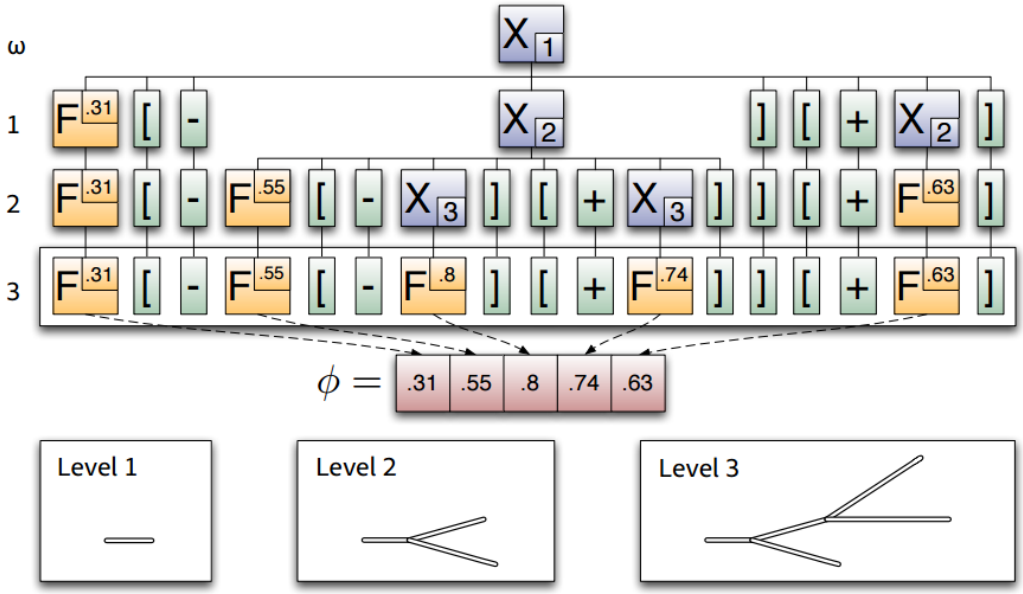
lives in $\in \mathbb{R}^{19}$

$$\alpha = \min\left(\frac{p(X_*)}{p(X_i)}\frac{q(X_n|X_*)}{q(X_*|X_n)}, 1\right)$$

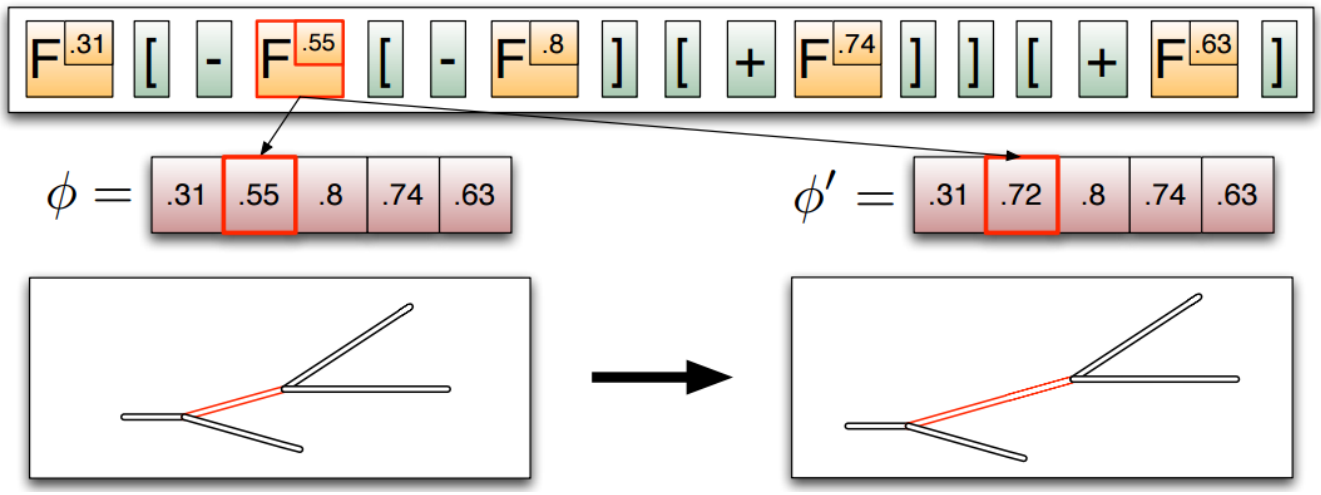
lives in $\in \mathbb{R}^5$

Use Reversible jump MCMC!

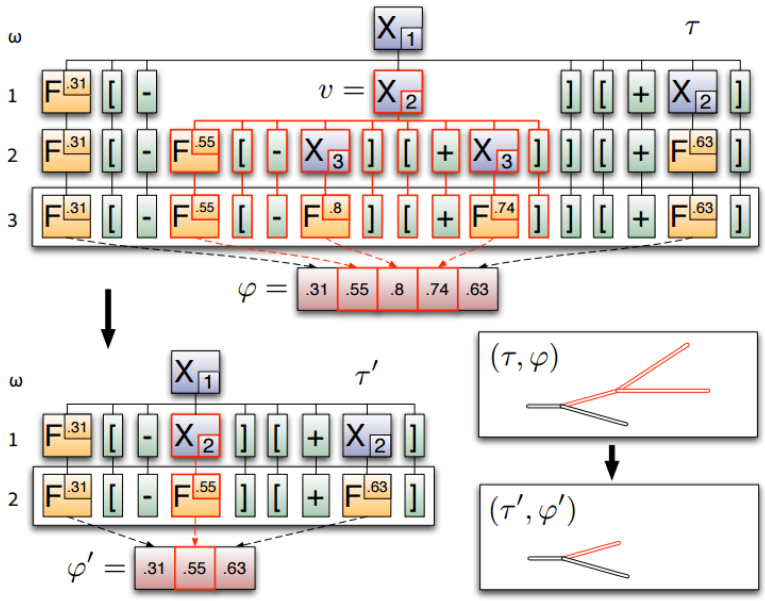
# Model Prior


ω
1
2
3
X 1
F .31
[
-
X 2
]
+
F .55
X 3
F .63
F .8
F .74
$\phi =$
.31
.55
.8
.74
.63
Level 1
Level 2
Level 3

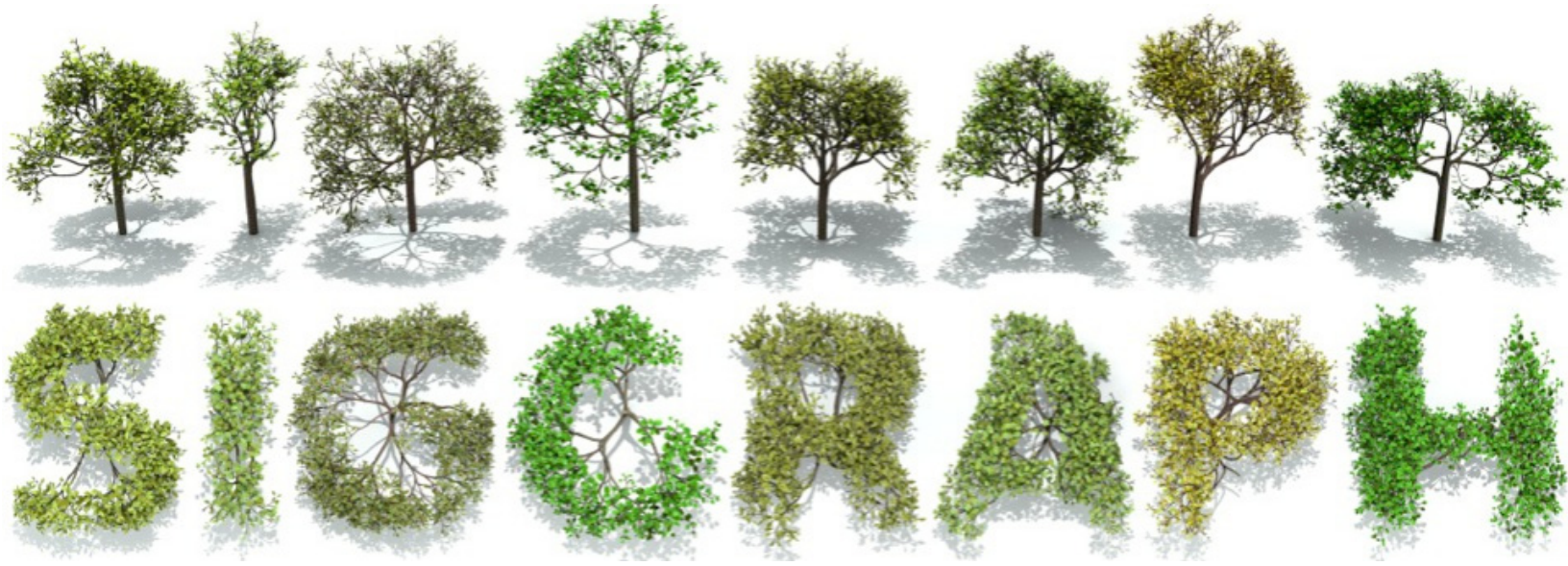
# Diffusion Moves



# Jump Moves

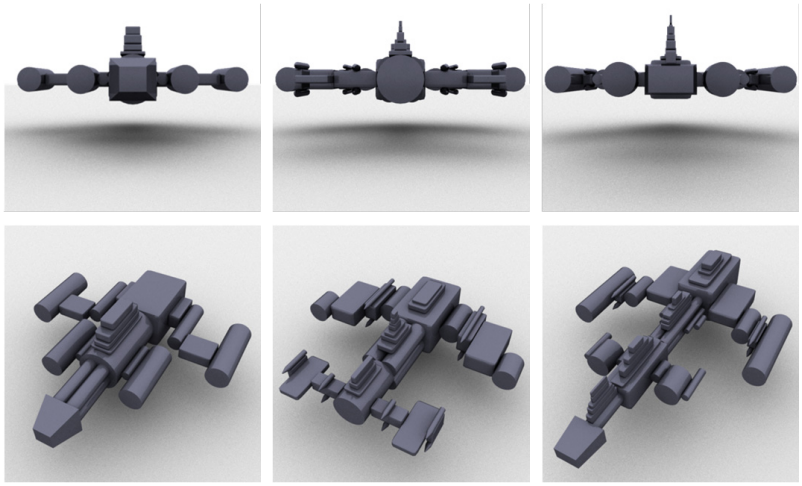


# Result



# Controlling Procedural Modeling Programs with Stochastically-Ordered Sequential Monte Carlo

- Target silhouette -> Best instance



## Controlling Procedural Modeling Programs with Stochastically-Ordered Sequential Monte Carlo

- Target silhouette -> Best instance
- MH is too slow
  - Generates whole model at once
  - No inference from middle steps
  - Use Sequential Monte Carlo!

## Controlling Procedural Modeling Programs with Stochastically-Ordered Sequential Monte Carlo

- Target silhouette -> Best instance
- MH is too slow
  - Generates whole model at once
  - No inference from middle steps
  - Use Sequential Monte Carlo!
- Stochastically order derivations

# Sequential Monte Carlo

$$p(\mathbf{x}|\mathbf{c}) \propto p(\mathbf{x})p(\mathbf{c}|\mathbf{x})$$

Random Choice Sequence

$$\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}$$

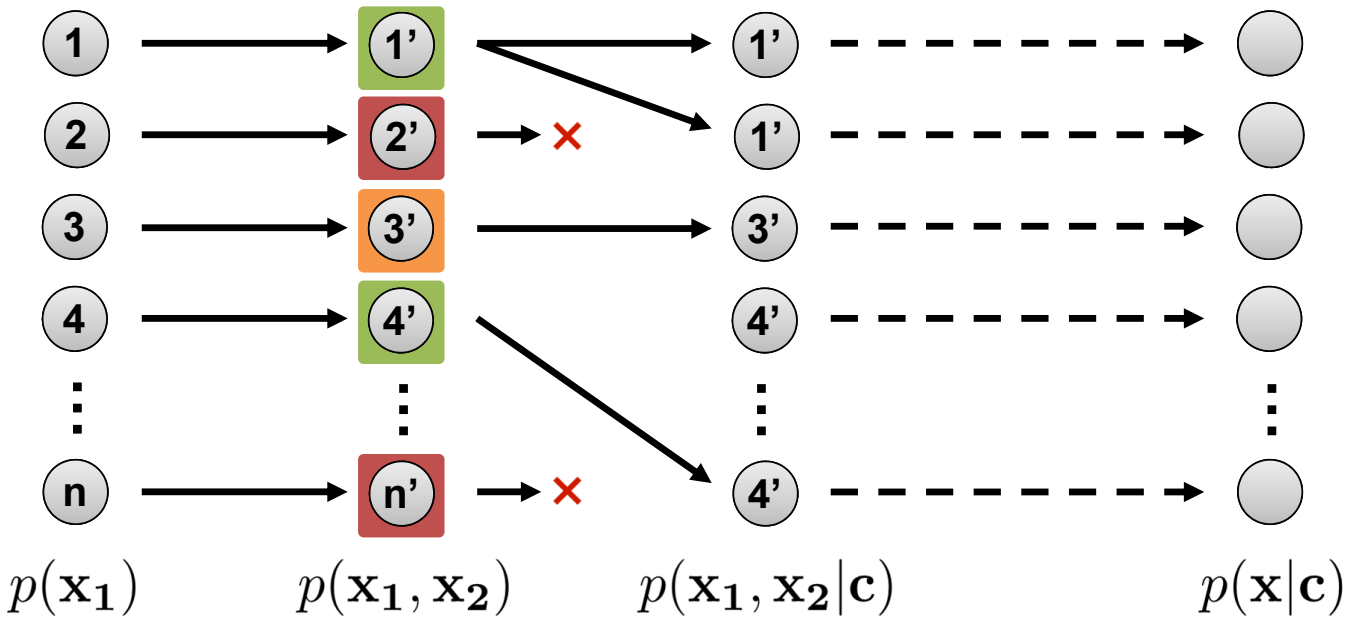
$$p(\mathbf{x_i}|\mathbf{x_1} \ldots \mathbf{x_{i-1}})$$

"Local" Likelihood

$$p(\mathbf{c}|\mathbf{x_1} \ldots \mathbf{x_i})$$

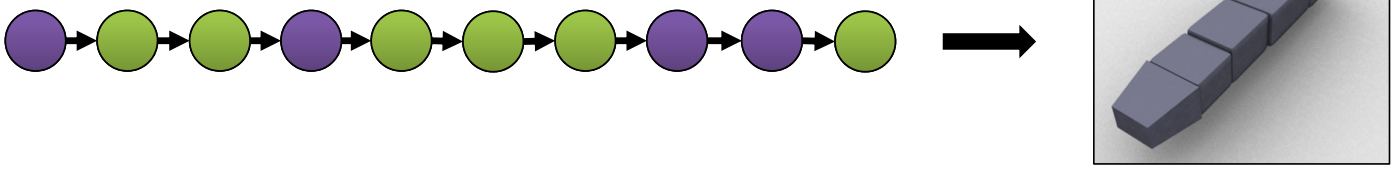
$$p(\mathbf{c}|\mathbf{x_1} \ldots \mathbf{x_n}) = p(\mathbf{c}|\mathbf{x})$$

# Sequential Monte Carlo



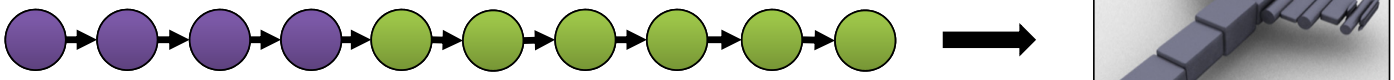
# Ordering Affects SMC

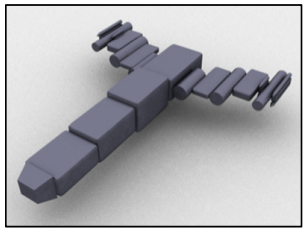
**Depth-first Ordering**





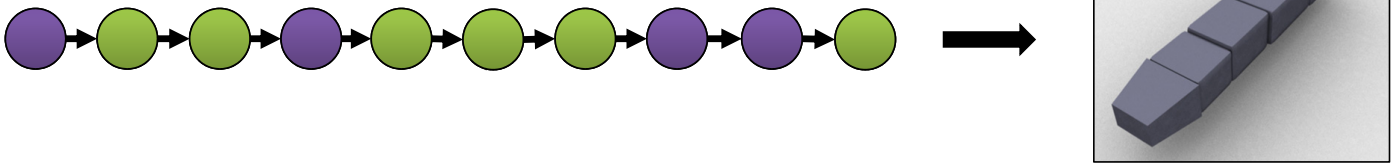
**“Body-first” Ordering**





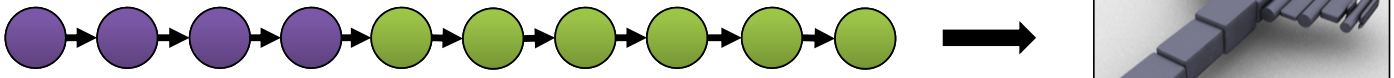
# Ordering Affects SMC

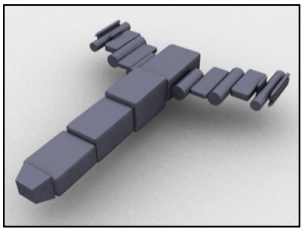
**Depth-first Ordering**





**“Body-first” Ordering**

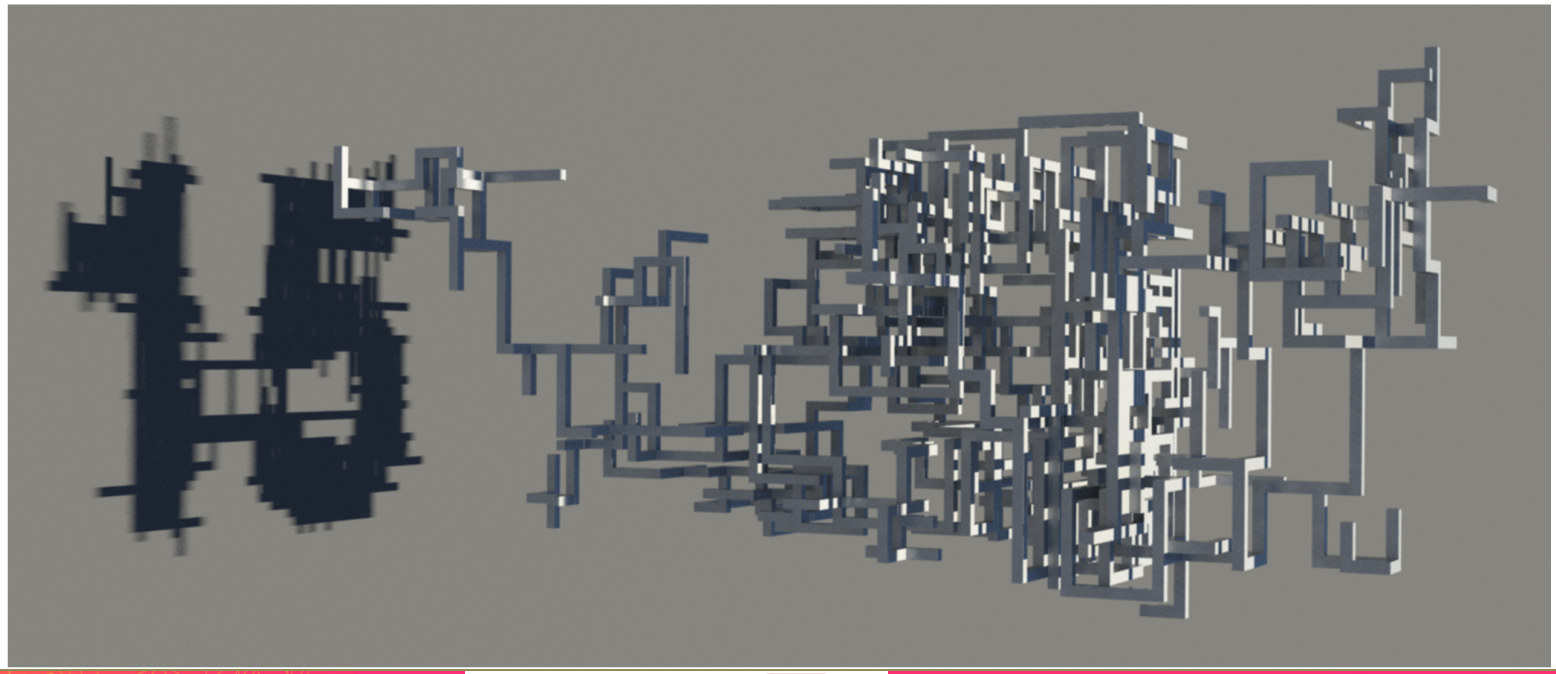




- Use SoSMC : $p(\mathbf{x}|\mathbf{c}) \longrightarrow p(\mathbf{x}, \mathbf{o}|\mathbf{c})$

Controlling Procedural Modeling Programs with Stochastically-Ordered Sequential Monte Carlo

# Results

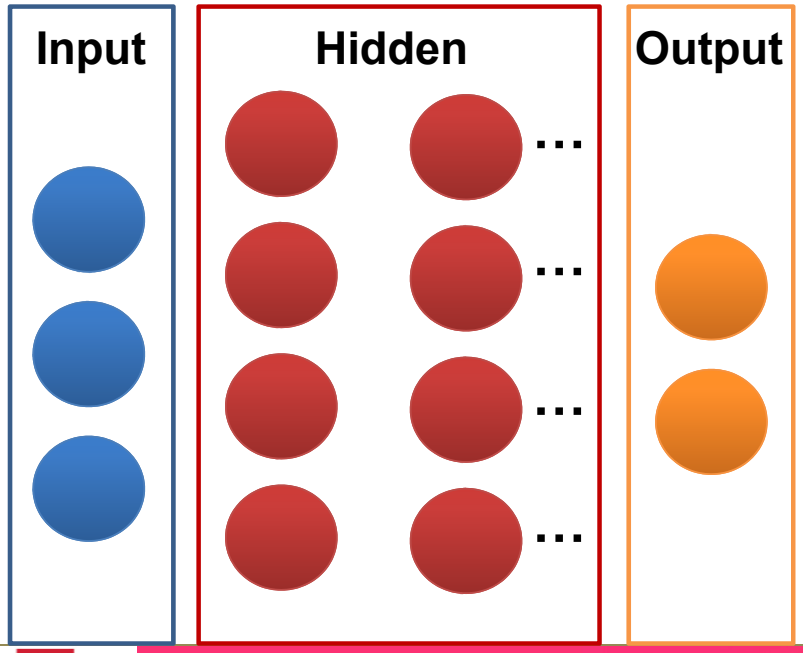


# IPM with High-level Priors

- Assumptions & Formulation
- MCMC
  - Review
  - Different approaches
- **Learning**
  - **Review**
  - **Neural Networks**

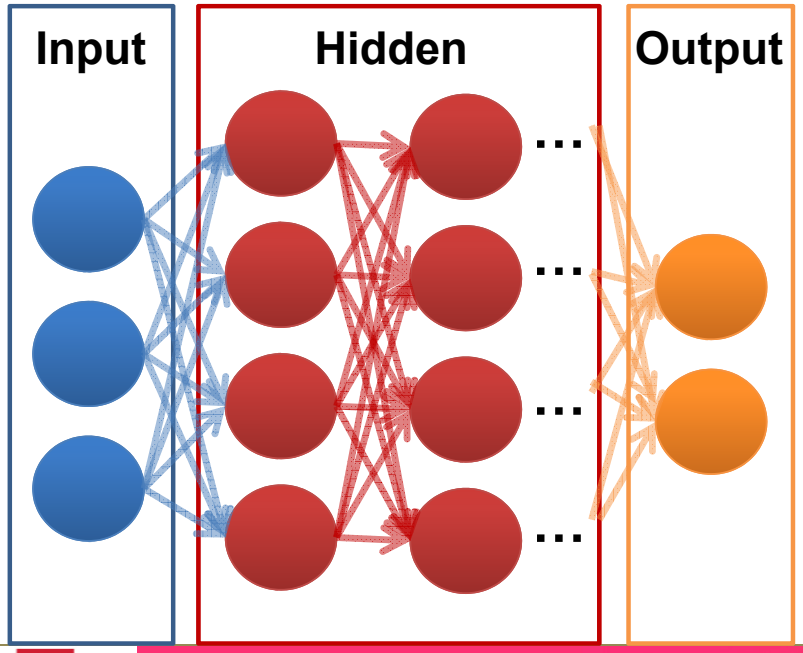
# Neural Networks

- Find the input-output relation,
  - as a function with optimal weights
  - from many samples
  - using several layers of connected nodes



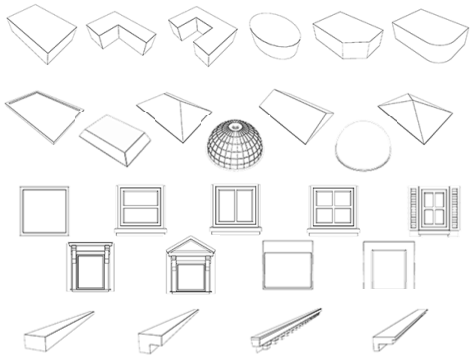
# Neural Networks

- Find the input-output relation,
  - as a function with optimal weights
  - from many samples
  - using several layers of connected nodes



# NN for IPM

**Input**

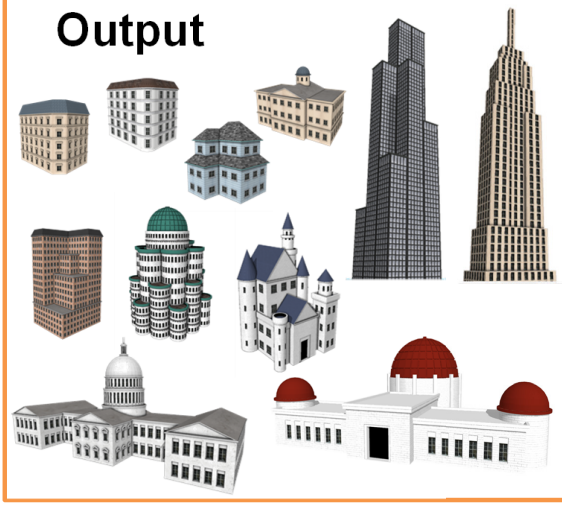


+ grammar

**Hidden**

$$O = f(I)$$
$$= \Sigma(w_{ij} I_i)$$

**Output**



# Learning Approaches in IPM

- Autoencoder
  - Procedural Modeling Using Autoencoder Networks (Yumer15)
- CNN
  - Interactive Sketching of Urban Procedural Models (Nishida16)
- MLP
  - Neurally-Guided Procedural Models: Learning to Guide Procedural Models with Deep Neural Networks (Ritchie16)

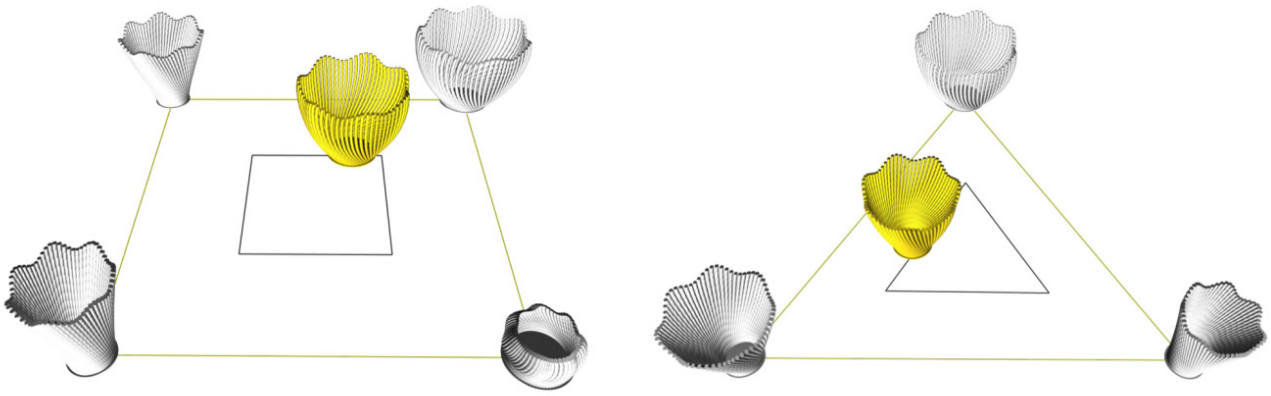
# Learning Approaches in IPM

- Autoencoder
  - Procedural Modeling Using Autoencoder Networks (Yumer15)
- CNN
  - Interactive Sketching of Urban Procedural Models (Nishida16)
- MLP
  - Neurally-Guided P… …arning to Guide Procedura… …Deep Neural Networks (Ritchie16)

Coming soon!

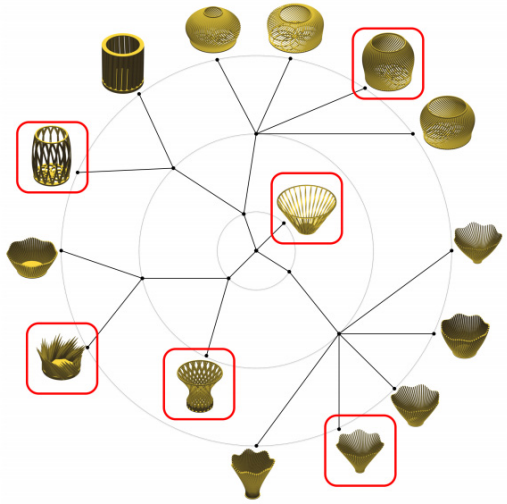
# Procedural Modeling Using Autoencoder Networks

- Procedural model -> Better exploration space



# Procedural Modeling Using Autoencoder Networks

- Procedural model -> Better exploration space
- Sample procedural space wrt shape features
  - Using C-tree + uniform sampling



# Procedural Modeling Using Autoencoder Networks

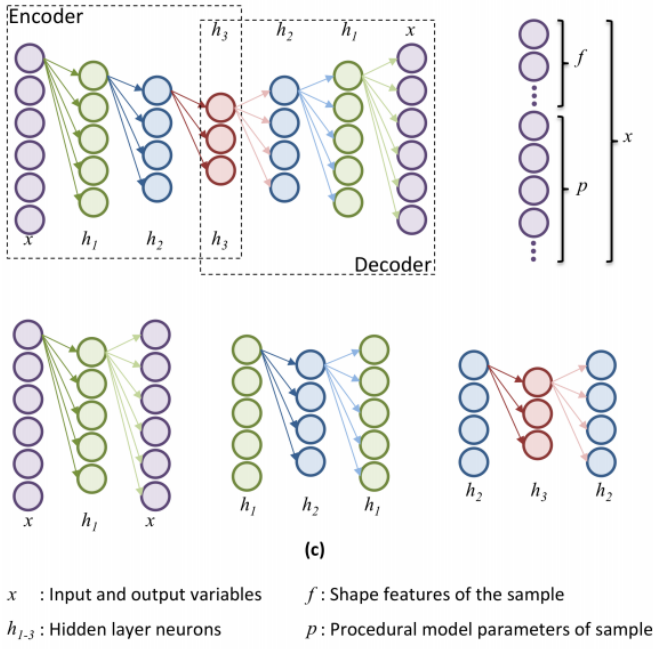
- Procedural model -> Better exploration space
- Sample procedural space wrt shape features
- Train an autoencoder with the samples

# Procedural Modeling Using Autoencoder Networks

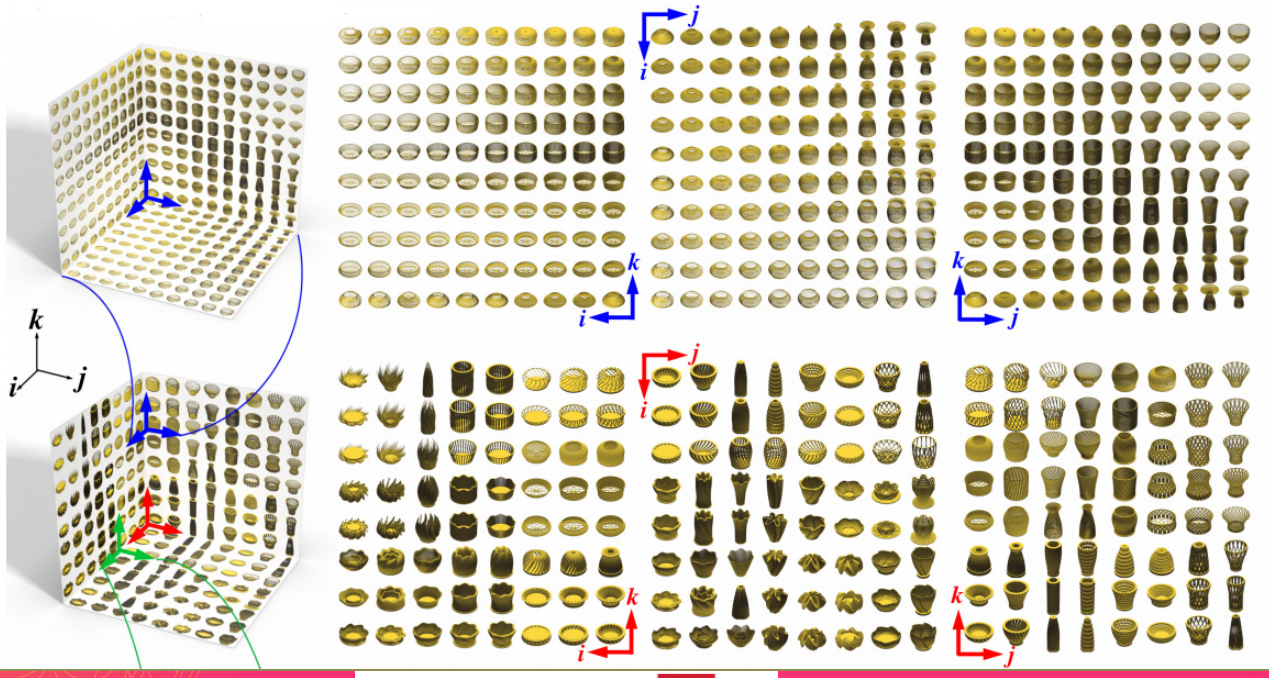
- Procedural model -> Better exploration space
- Sample procedural space wrt shape features
- Train an autoencoder with the samples
- Structure the lower dimensional derivation space

# IPM with Autoencoders

- Input/output: $p$ (parametric) and $f$ (LFD) info
- Fully connected
- 5 hidden layers
- Symmetric
- Autoencoder

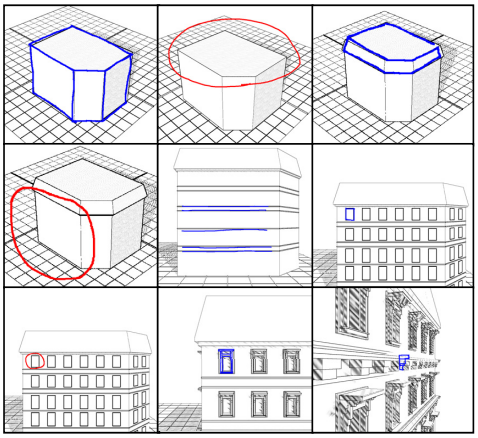


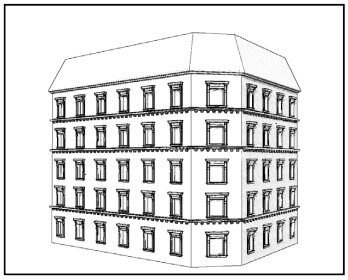
# Results



# Interactive Sketching of Urban Procedural Models

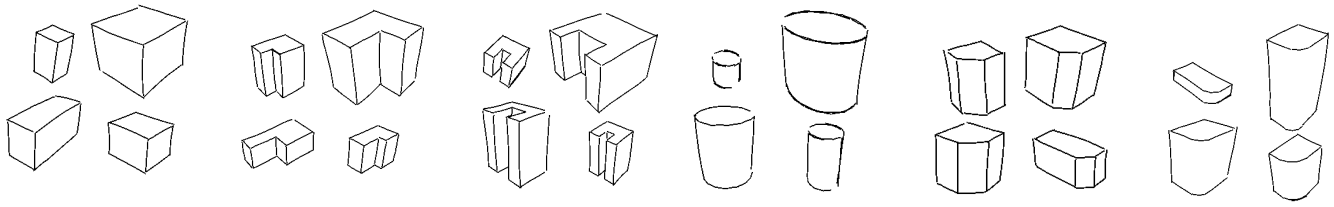
- User sketch-> Best instances





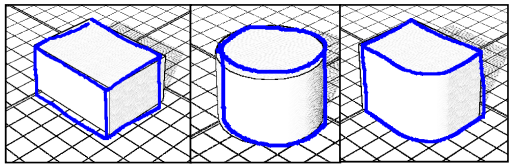
# Interactive Sketching of Urban Procedural Models

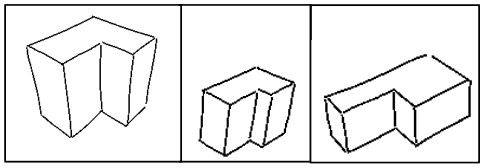
- User sketch-> Best instances
- Sample procedural snippets by rendering



# Interactive Sketching of Urban Procedural Models

- User sketch-> Best instances
- Sample procedural snippets by rendering
- Train recognition and parameter CNNs





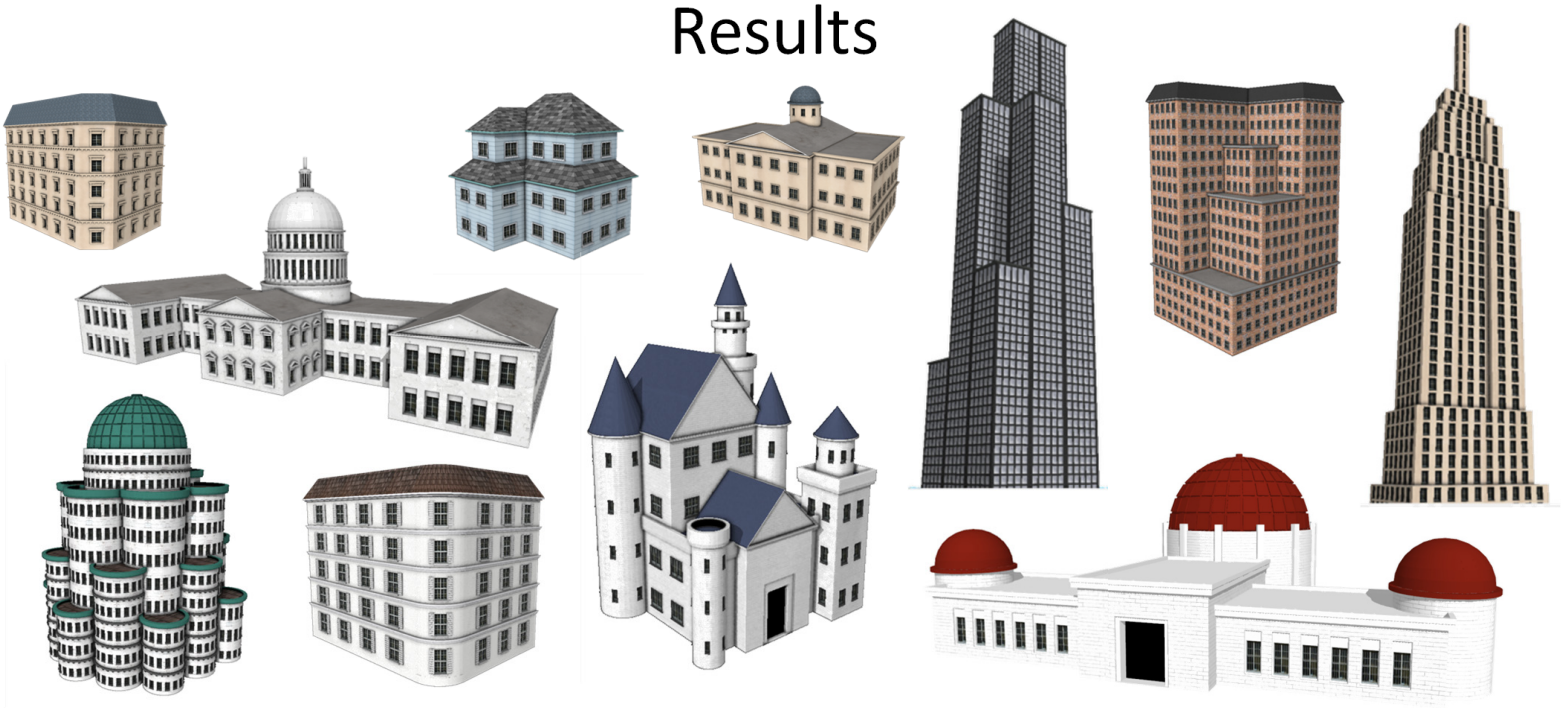
# Interactive Sketching of Urban Procedural Models

- User sketch-> Best instances
- Sample procedural snippets by rendering
- Train recognition and parameter CNNs
- Provide sketch-based procedural modeling

# IPM with CNNs

- Recognition CNN:
  - Input/output:
    Fixed view PM renders/
    pdf of snippet types
  - BVLC AlexNet
    architecture
  - Weights from Caffe
    Model Zoo
- Parameter CNN:
  - Input/output:
    Fixed view PM renders/
    normalized parameters
  - 3 convolutional +
    2 fully connected layers
  - Weights from scratch

# Results

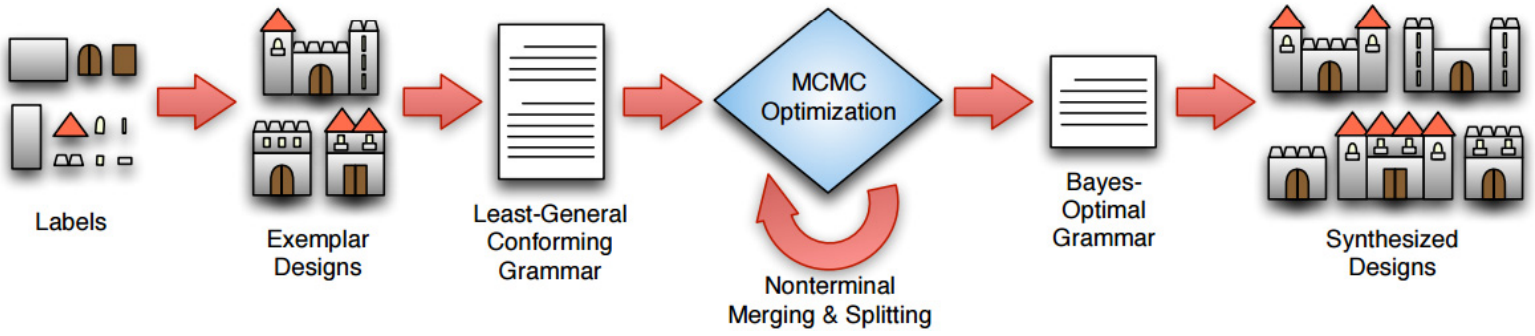


# Hybrid IPM with High-Level Priors

- Following are some papers that span both searching (e.g., MCMC) and learning categories

Other papers [Talton12]

# Learning Design Patterns with Bayesian Grammar Induction



## 4 Inverse Procedural Modeling Domains

### 4.1 Facades & Layouts

Render the Possibilities
SIGGRAPH2016
THE 43RD INTERNATIONAL CONFERENCE AND EXHIBITION ON Computer Graphics & Interactive Techniques

# Inverse Procedural Modeling of 3D Models for Virtual Worlds

JOHANNES GUTENBERG UNIVERSITÄT MAINZ
PURDUE UNIVERSITY

Daniel Aliaga Ilke Demir
Bedrich Benes Michael Wand

# Outline

- Introduction
- Fundamentals of IPM
- IPM Approaches
  - Low-level Priors: Inferring Shape Grammars
  - High-level Priors: Inferring Grammar Parameters
- **Inverse Procedural Modeling Domains**
  - **Facades & Layouts**
  - Vegetation
  - Urban Areas
- Conclusions, Challenges, Open Problems

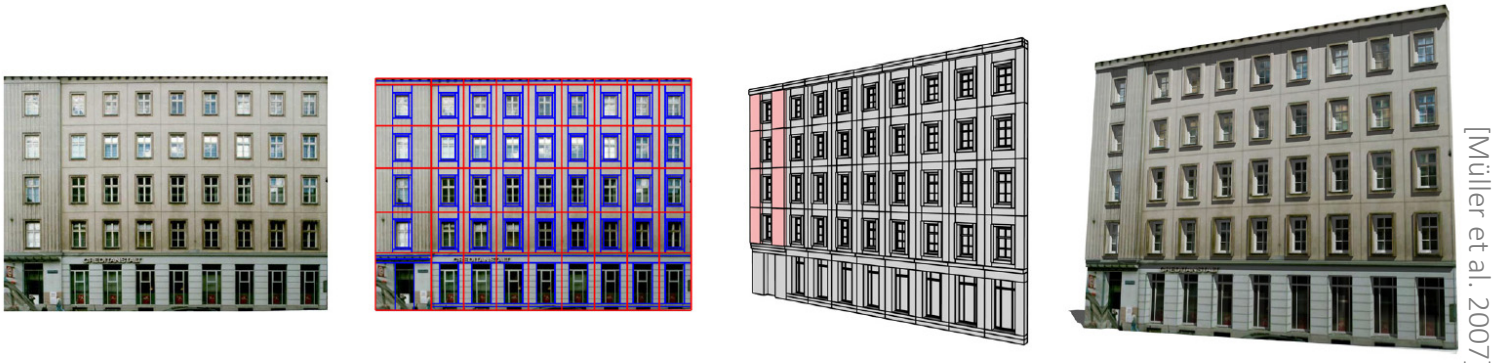
# Facades & Layouts

- Example System
  - Image-based façade parsing [Müller et al. 2007]
- Modeling Façade Layouts
  - Symmetry maximization, MDL
  - Layers, constrains, relation graphs, grids
- Matching data
  - Image parsing
- Direct modeling
  - Landmark guided
  - Offset statistics guided

# Facades & Layouts

- **Example System**
  - **Image-based façade parsing** [Müller et al. 2007]
- Modeling Façade Layouts
  - Symmetry maximization, MDL
  - Layers, constrains, relation graphs, grids
- Matching data
  - Image parsing
- Direct modeling
  - Landmark guided
  - Offset statistics guided

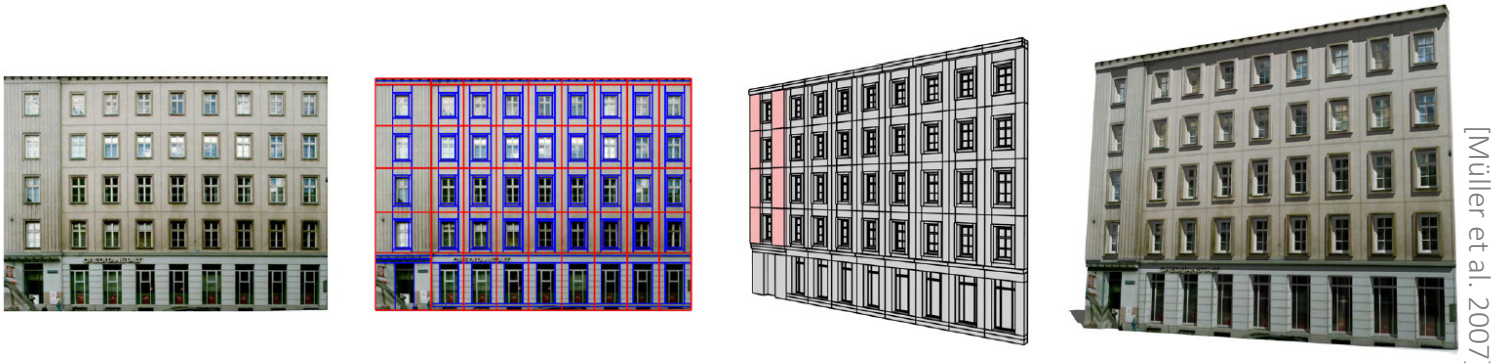
# Image-based Procedural Modeling of Façades



[Müller et al. 2007]

- **(Initial) Example-based Approach**
- **Mueller, P., Zeng, G., Wonka, P., and Van Gool, L,:** Image-based Procedural Modeling of Facades, *ACM ToG (SIGGRAPH)*, 2007.

# Image-based Procedural Modeling of Façades



[Müller et al. 2007]

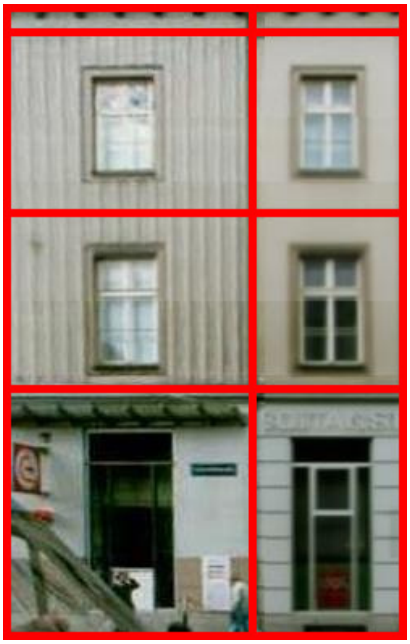
- Input: Rectified facade photograph
- Output: 3D model and a shape grammar

# Image-based Procedural Modeling of Façades

- Approach:
  1. Facade structure detection
  2. Tile refinement (split grammars)
  3. Element recognition
  4. CGA rule extraction

# Image-based Procedural Modeling of Façades

- Splitting via global optimization
  - Extraction of floors and tiles



[Müller et al. 2007]

# Image-based Procedural Modeling of Façades

- Expanding grid on a facade


[Müller et al. 2007]

(video)

# Image-based Procedural Modeling of Façades

- Local split detection

[Müller et al. 2007]

# Image-based Procedural Modeling of Façades

- Before synchronization


[Müller et al. 2007]

# Image-based Procedural Modeling of Façades

- After synchronization


[Müller et al. 2007]

# Image-based Procedural Modeling of Façades

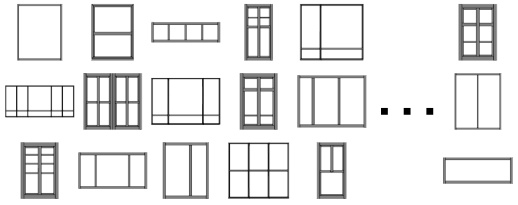
- Resulting tile subdivision

[Müller et al. 2007]

(video)

# Image-based Procedural Modeling of Façades

- Matching 3D elements
  - Compare image of rendered 3D object with captured façade tile image
  - Compare regions via mutual information



[Müller et al. 2007]

# Image-based Procedural Modeling of Façades

- CGA rule extraction
  - Semantics for floor, tiles, and elements
  - Implicitly size-independent rules

[Müller et al. 2007]

(video)

# Image-based Procedural Modeling of Façades

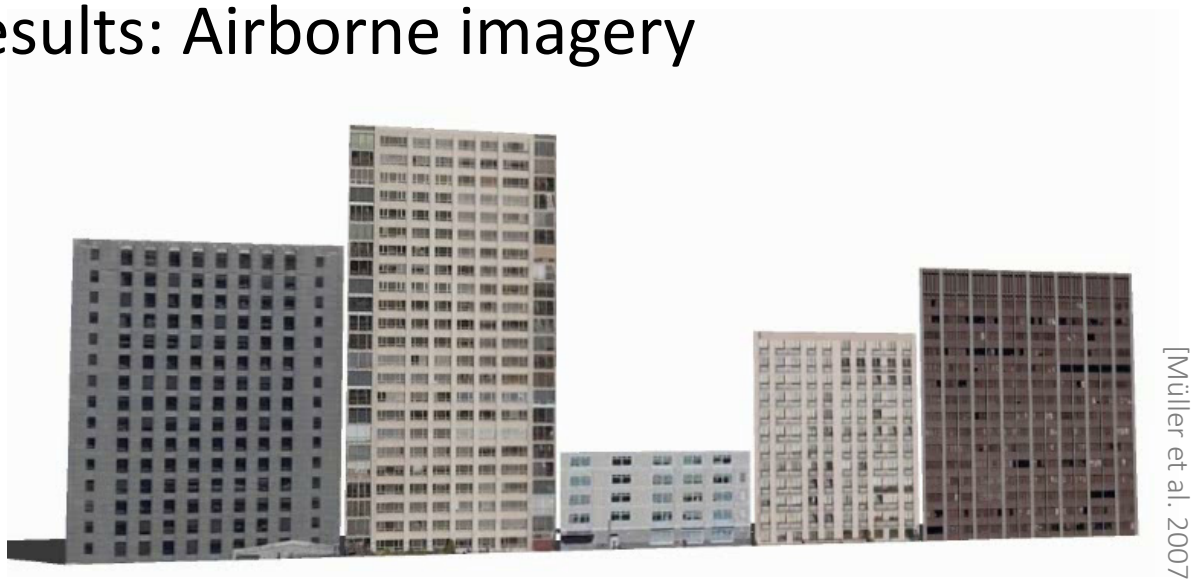
- Results: Ground-based imagery

[Müller et al. 2007]

(video)

# Image-based Procedural Modeling of Façades

- Results: Airborne imagery



[Müller et al. 2007]

# Facades & Layouts

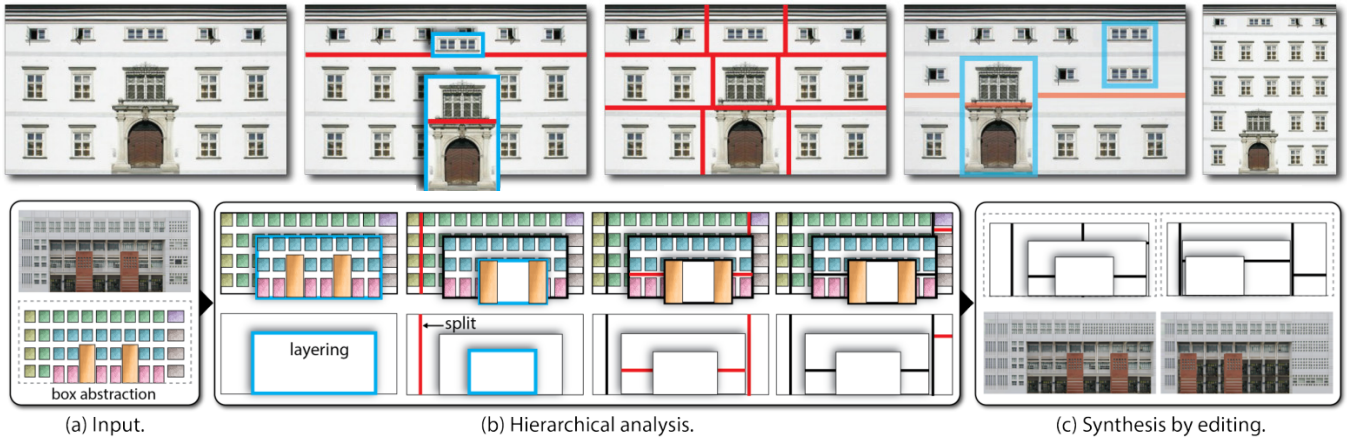
- Example System
  - Image-based façade parsing [Müller et al. 2007]
- **Modeling Façade Layouts**
  - **Symmetry maximization, MDL**
  - **Layers, constrains, relation graphs, grids**
- Matching data
  - Image parsing
- Direct modeling
  - Landmark guided
  - Offset statistics guided

# System: City Modeling

**Xiao , J., Fang , T., Zhao , P., Lhuillier , M., Quan, L.:**
Image-based street-side city modeling
ACM ToG 2009


[Xiao et al. 2009]

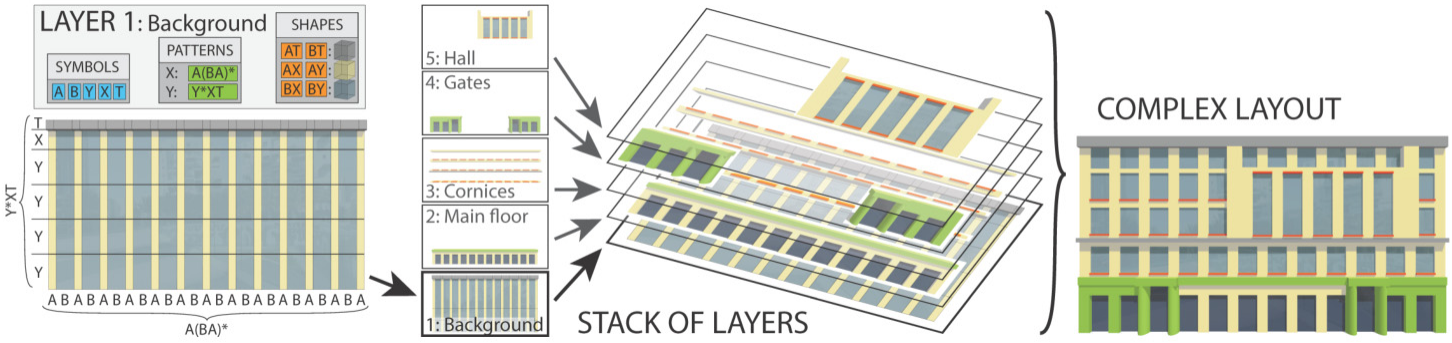
# Symmetry Maximization




[Zhang et al. 2013]

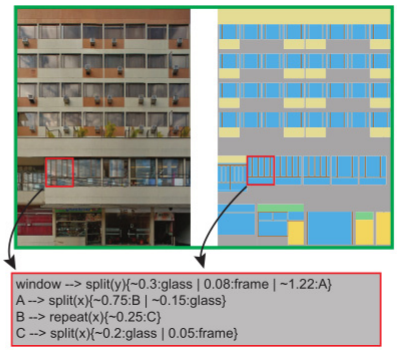
**H. Zhang, K. Xu, W. Jiang, J. Lin, D. Cohen-Or, B. Chen:**
Layered Analysis of Irregular Facades via Symmetry Maximization**,**
ACM ToG (Siggraph) 2013.

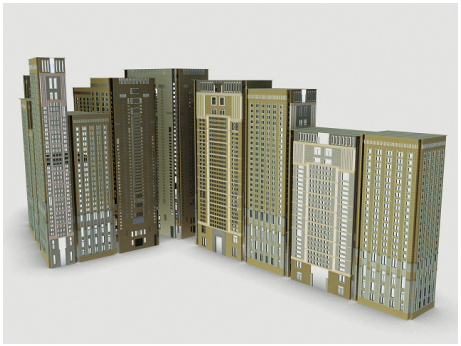
# Layered Modeling




**Ilčík, M., Musialski, P., Auzinger, T. and Wimmer, M.:**
Layer-Based Procedural Design of Façades,
Computer Graphics Forum, 34: 205–216, 2015.

# Minimum Description Length



**Fuzhang Wu, Dong-Ming Yan, Weiming Dong, Xiaopeng Zhang, Peter Wonka:**
Inverse Procedural Modeling of Facade Layouts, Siggraph 2014.

# General Principle: MDL

- **Jerry O. Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah D. Goodman, and Radomir Mech**
  Learning Design Patterns with Bayesian Grammar Induction, ACM SIGCHI 2012

- **Anđelo Martinović, Luc Van Gool**
  Bayesian Grammar Learning for Inverse Procedural Modeling, CVPR 2013

# Semi-Regular Grids
# (Graph Data Structure)


(a) Input (b) Editing output

**M. Dang, D. Ceylan, B. Neubert, and M. Pauly:**
Safe: Structure-aware façade editing,
Computer Graphics Forum, 33(2), 83–93, 2014.

# Variations from Single Example (User Guided)

(a) Input (b) Loose (c) Medium (d) Strict

**Bao, F., Schwarz, M., Wonka, P.:**
Procedural facade variations from a single layout,
ACM Trans. Graph. 32, 1, 8:1–8:13, 2013.

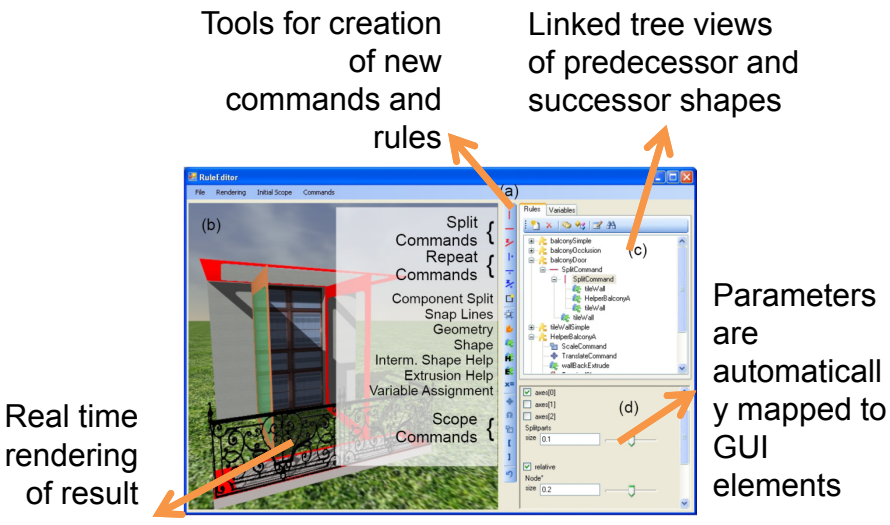
# Linking Manual & Procedural Modeling

**M. Lipp, P. Wonka, M. Wimmer**

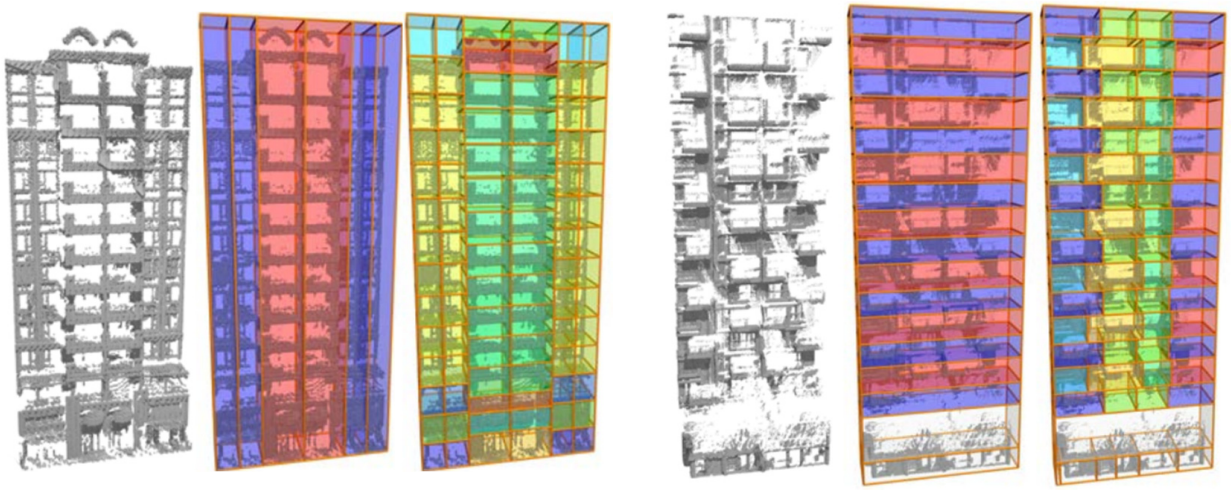
Interactive Visual Editing of Grammars for Procedural Architecture,
ACM ToG (SIGGRAPH) 2008.

# Interactive Visual Editing of Grammars for Procedural Architecture

- Approach:
  - Interactive visual editing of shape grammars
  - Allows the creation of rules from scratch without text file editing



# 3D Façades, Grid Structure

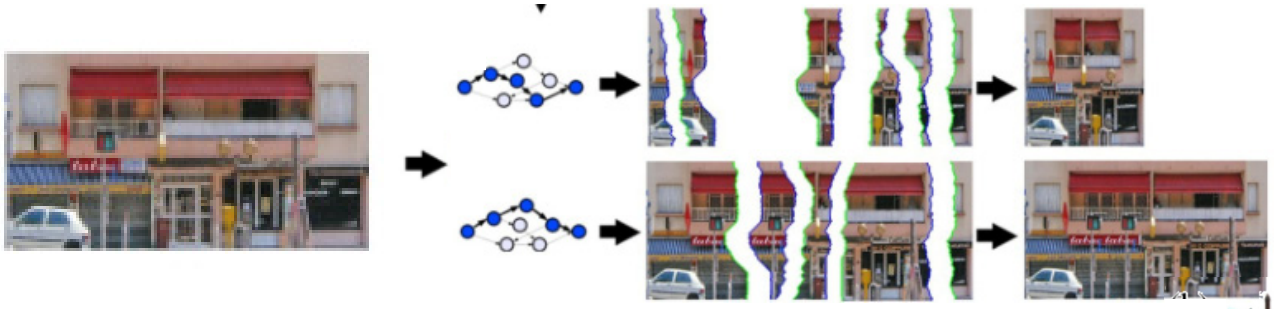


**C.-H. Shen, S.-S. Huang, H. Fu, and S.-M. Hu:**
Adaptive partitioning of urban facades,
SIGGRAPH Asia 2011.

# Facades & Layouts

- Example System
  - Image-based façade parsing [Müller et al. 2007]
- Modeling Façade Layouts
  - Symmetry maximization, MDL
  - Layers, constrains, relation graphs, grids
- **Matching data**
  - **Image parsing**
- Direct modeling
  - Landmark guided
  - Offset statistics guided

# “Double-Cut” Algorithm for Images



**Lefebvre, S., Hornus, S., Lasram, A.:**
By-example synthesis of architectural textures,
Siggraph 2010.

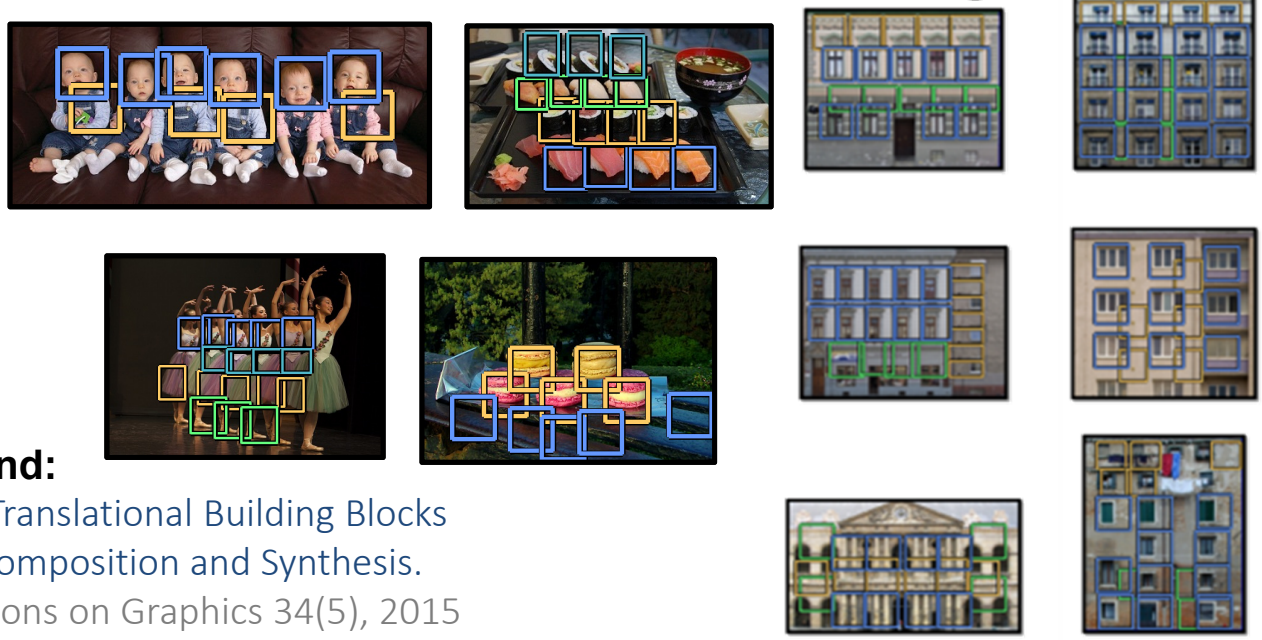
# Facade Parsing: Machine Learning

[Martinović 2012]

**Martinović, A., Mathias, M., Weissenberg, J., Van Gool, L:**
A Three-Layered Approach to Facade Parsing, ECCV 2012.

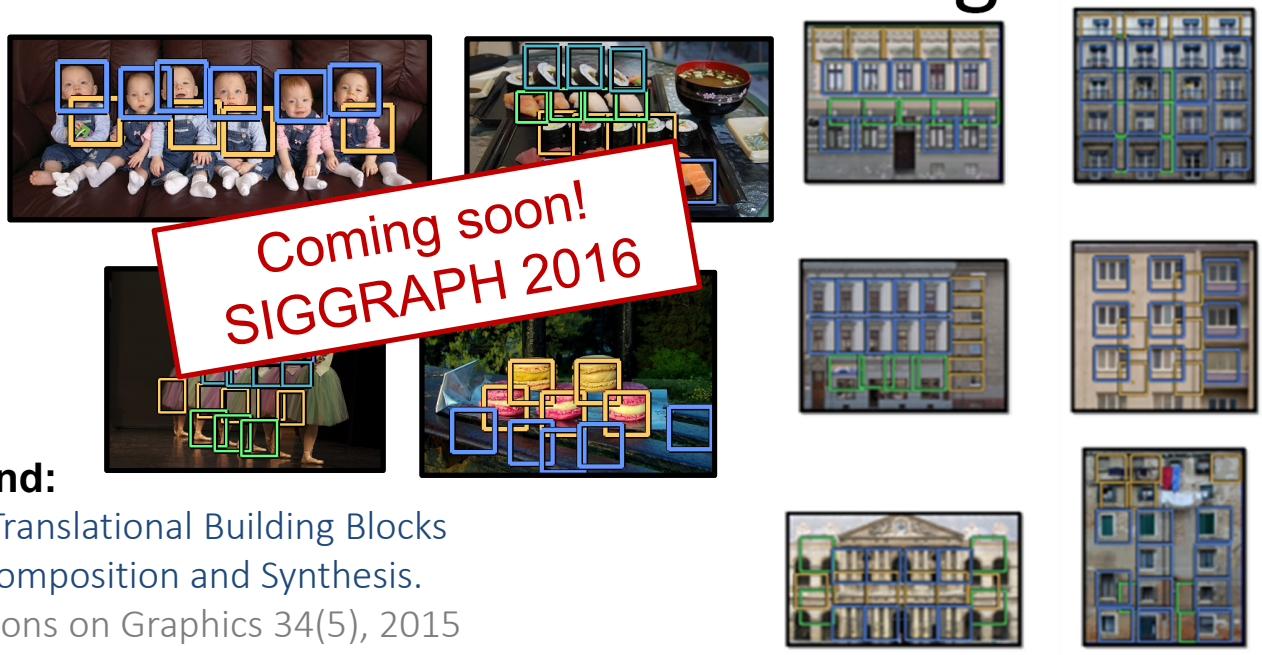
- 1st layer: RNN detector
- 2nd layer: MRF segmentation
- 3rd layer: alignment, symmetry, co-occurrence

# Co-Occurrence Clustering



**C. Li, M. Wand:**
Approximate Translational Building Blocks for Image Decomposition and Synthesis.
ACM Transactions on Graphics 34(5), 2015

# Co-Occurrence Clustering




**C. Li, M. Wand:**
Approximate Translational Building Blocks for Image Decomposition and Synthesis.
ACM Transactions on Graphics 34(5), 2015

# Facades & Layouts

- Example System
  - Image-based façade parsing [Müller et al. 2007]
- Modeling Façade Layouts
  - Symmetry maximization, MDL
  - Layers, constrains, relation graphs, grids
- Matching data
  - Image parsing
- **Direct modeling**
  - **Landmark guided**
  - **Offset statistics guided**

# Landmark Guided Graph Cut

# Problems of MRF Texture Synthesis

input

texture optimization + quilting

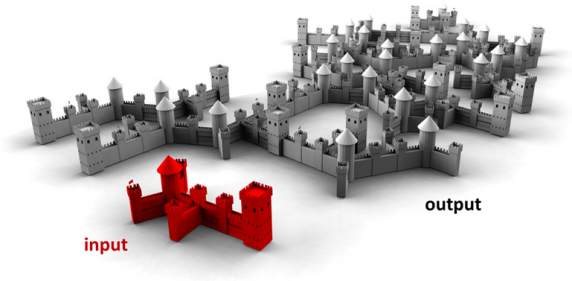
Local plausibility,
global structure lacking

# Problems of MRF Texture Synthesis

input

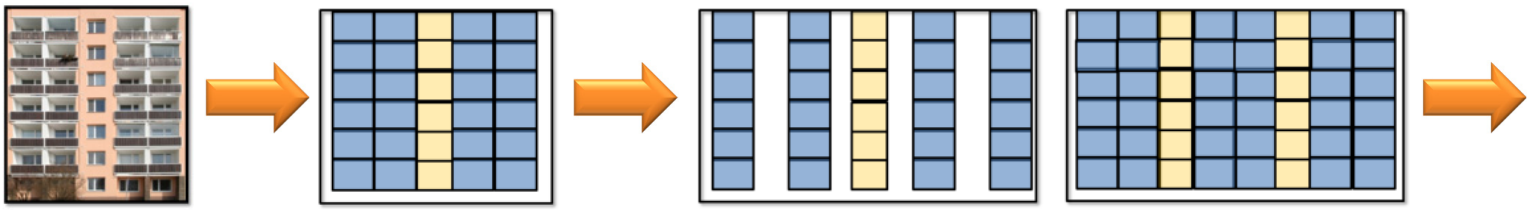
texture optimization + quilting

← Locally plausible, but global structure lacking




← globally: not (really) a castle

# Landmark Guided Texture Synthesis



input analysis resizing with building-block grammar

optimization ... ... optimization ... ... result

# Guided MRF Texture Synthesis

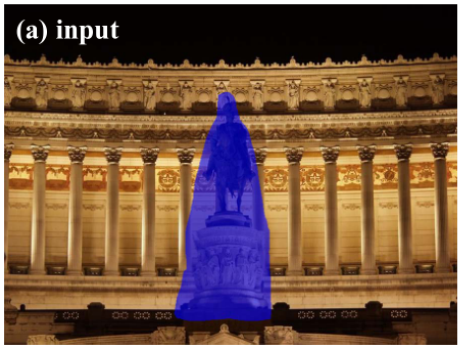
input

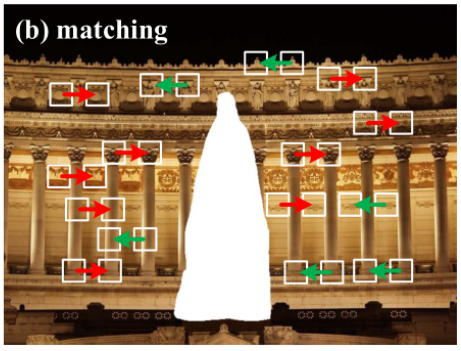
building-block guided

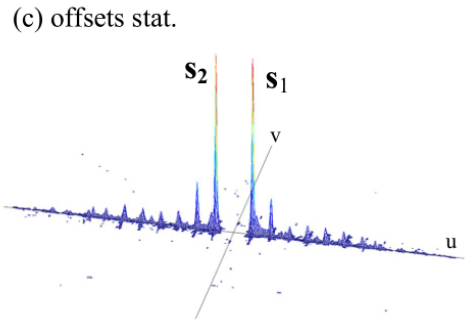
texture optimization + quilting

# Graph-Cut Texture Synthesis with Offset Statistics

# Guided MRF Synthesis




[He et al. 2012]

**Kaiming He and Jian Sun:**
Statistics of Patch Offsets for Image Completion,
*ECCV 2012*.


Render the Possibilities
SIGGRAPH2016
THE 43RD INTERNATIONAL
CONFERENCE AND EXHIBITION ON
&
Computer Graphics
Interactive Techniques
24-28 JULY
ANAHEIM, CALIFORNIA

Render the Possibilities
SIGGRAPH2016
THE 43RD INTERNATIONAL CONFERENCE AND EXHIBITION ON Computer Graphics & Interactive Techniques

# Inverse Procedural Modeling of 3D Models for Virtual Worlds

JOHANNES GUTENBERG UNIVERSITÄT MAINZ
PURDUE UNIVERSITY

Daniel Aliaga Ilke Demir
Bedrich Benes Michael Wand

# Outline

- Introduction
- Fundamentals of IPM
- IPM Approaches
  - Low-level Priors: Inferring Shape Grammars
  - High-level Priors: Inferring Grammar Parameters
- **Inverse Procedural Modeling Domains**
  - Facades & Layouts
  - **Vegetation**
  - Urban Areas
- Conclusions, Challenges, Open Problems

# IPM for Vegetation

- Inverse Procedural Modeling of Trees [Stava et al. 14]
- Guided PM [Benes et al. 11]
- TreeSketch: Interactive Procedural Modeling of Trees on a Tablet [Longay et al. 12]
- Automatic Procedural Modeling of Tree Structures in Point Clouds Using Wavelets [Friedman et al. 13]
- Procedural Tree Modeling with Guiding Vectors [Xu and Mould 15]
- Modeling Plant Life in Computer Graphics [SIGGRAPH 16!]

# IPM for Vegetation

- Inverse Procedural Modeling of Trees [Stava et al. 14]
- Guided PM [Benes et al. 11]
- TreeSketch: Interactive Procedural Modeling of Trees on a Tablet [Longay et al. 12]
- Automatic Procedural Modeling of Tree Structures in Point Clouds Using Wavelets [Friedman et al. 13]
- Procedural Tree Modeling with Guiding Vectors [Xu and Mould 15]
- Modeling Pl… …puter Graphics [SIGGRAPH 16!]

Coming soon!

# Inverse Procedural Modeling of Trees

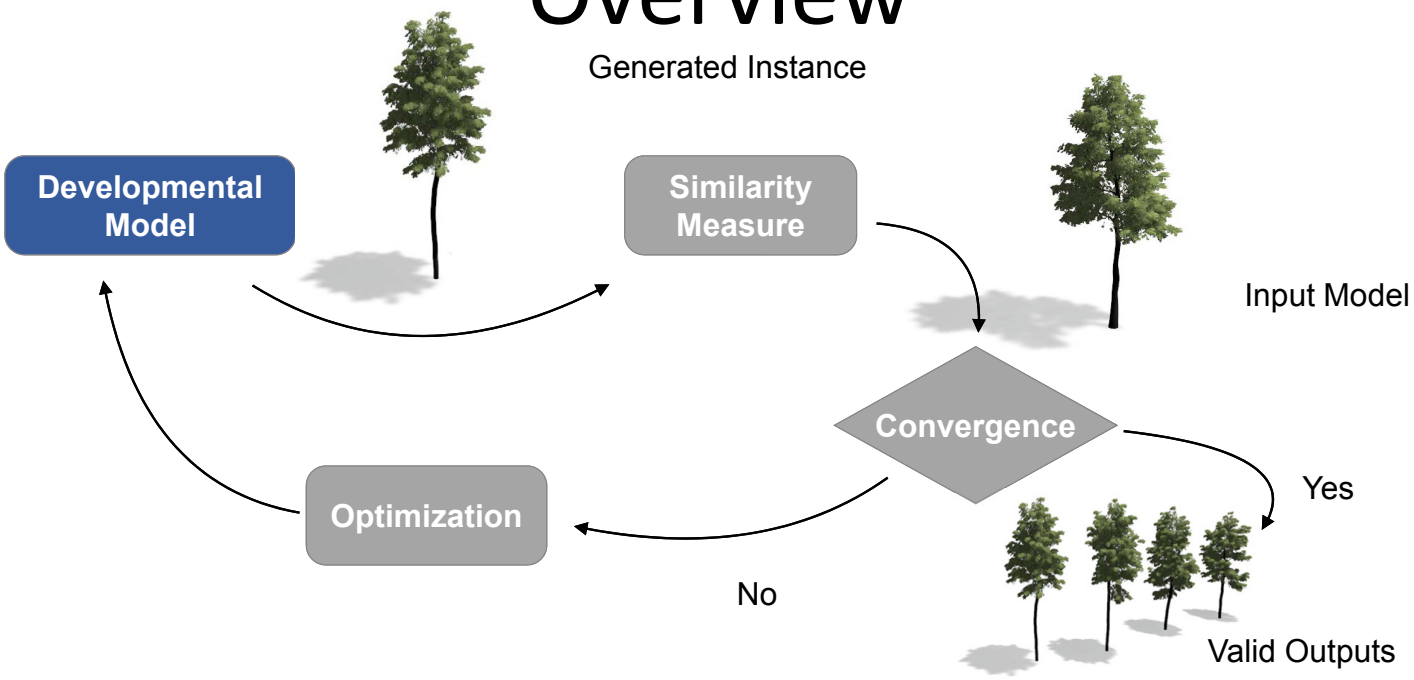
Stava, O., Pirk, S., Kratt, J., Chen, B., Mech, R., Deussen, O., and Benes, B.

*Computer Graphics Forum, 2014*

# Inverse Procedural Modeling of Trees

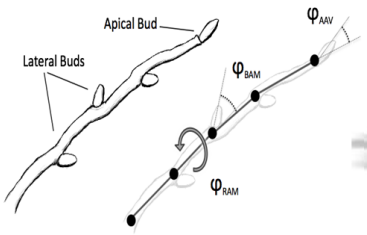
(video)

# Overview


Generated Instance
Developmental Model
Similarity Measure
Input Model
Convergence
Yes
Optimization
No
Valid Outputs

# Developmental Model

- Captures new biological findings
  [Cline et al. 2006, Cline et al. 2009]
- Geometric, environmental, and bud fate parameters
- Patch-based foliage modeling
  [Livny et al. 2011]




# Developmental Model

**Geometric Params**

Growth Rate
Internode Length
Internode Angle Factor
Apical Control Level
Apical Dominance Factor
…

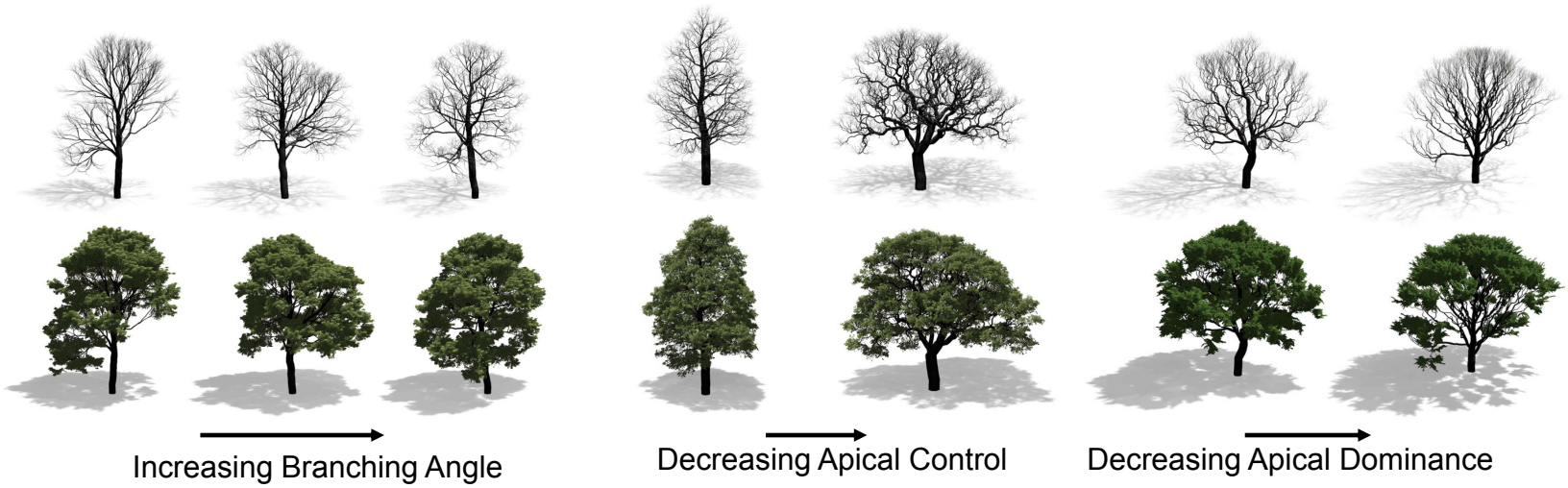
**Environment Params**

Gravitropism
Phototropism
Pruning Factor
Low Branch Pruning Factor
Gravity-bending Strength
…

**Bud Fate Params**

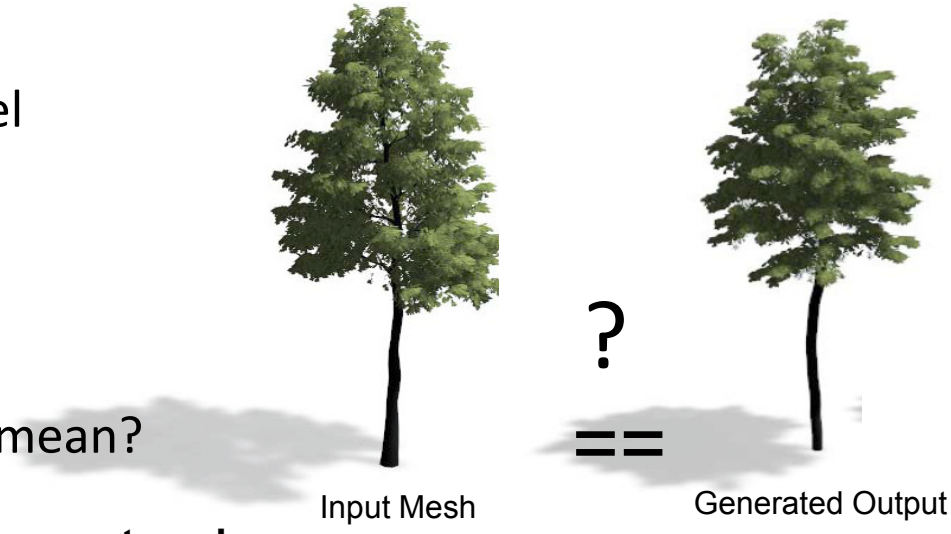
Apical Angle Variance
Number of Lateral Buds
Branching Angle Mean and Variance
Roll Angle and Variance
Apical and Lateral Light Factor
…

# Developmental Model



# Optimization

- Find parameters for developmental model
- Maximize similarity between input and generated instance
- What does "similar" mean?

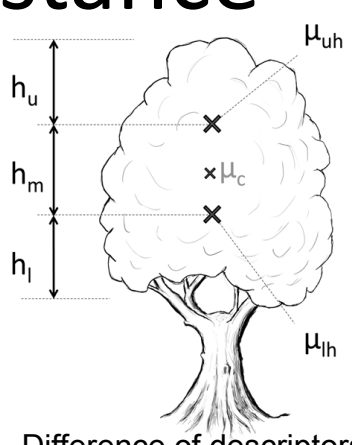


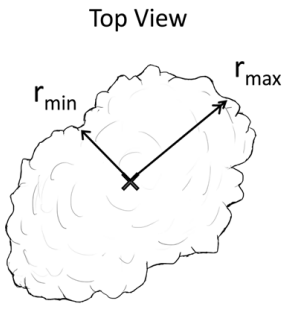

**Fitness function based on geometry, shape and structure**

# Shape Distance

- Crown shape affected by distribution of branches
- Divide tree into slabs to capture variance
- Compute shape descriptors for each slab:

**Height, radius, principal directions, leaf-branch density**




Shape Descriptors

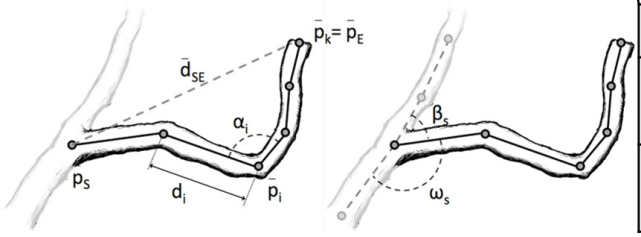
Difference of descriptors

$$\delta_{\lambda_{S,i}} = 1 - \exp\left(-\frac{(\lambda_{S,i}(\tau_1) - \lambda_{S,i}(\tau_2))^2}{2\sigma^2_{\lambda_{S,i}}}\right)$$

Normalization Factor

# Geometric Distance

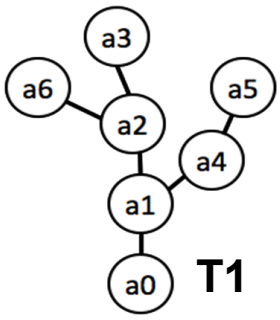
- Statistics of branch geometry computed from the tree graph
- Sample weight based on length and thickness of a branch
- Descriptors are defined as mean and variance of these samples

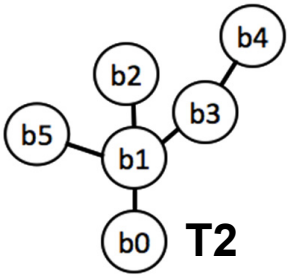



| Name | Formula |
|---|---|
| Length | $\sum_{i=1}^{k} d_i$ |
| Thickness | $\max_{\forall d_i} t_i$ |
| Deformation | $\sum_{i=1}^{k-1} \alpha_i$ |
| Straightness | $\frac{\lvert\bar{d}_{SE}\rvert}{b_L}$ |
| Slope | $\angle\bar{d}_{SE}$ |
| Sibling Angle | $\beta_S$ |
| Parent Angle | $\omega_S$ |

# Structural Distance

- Transform graph T1 into graph T2
- Costs for transforming the nodes (edit distance)
- Possible transformations: **assign, insert, delete**
- Quickly loses accuracy when geometric resolution differs




Trees

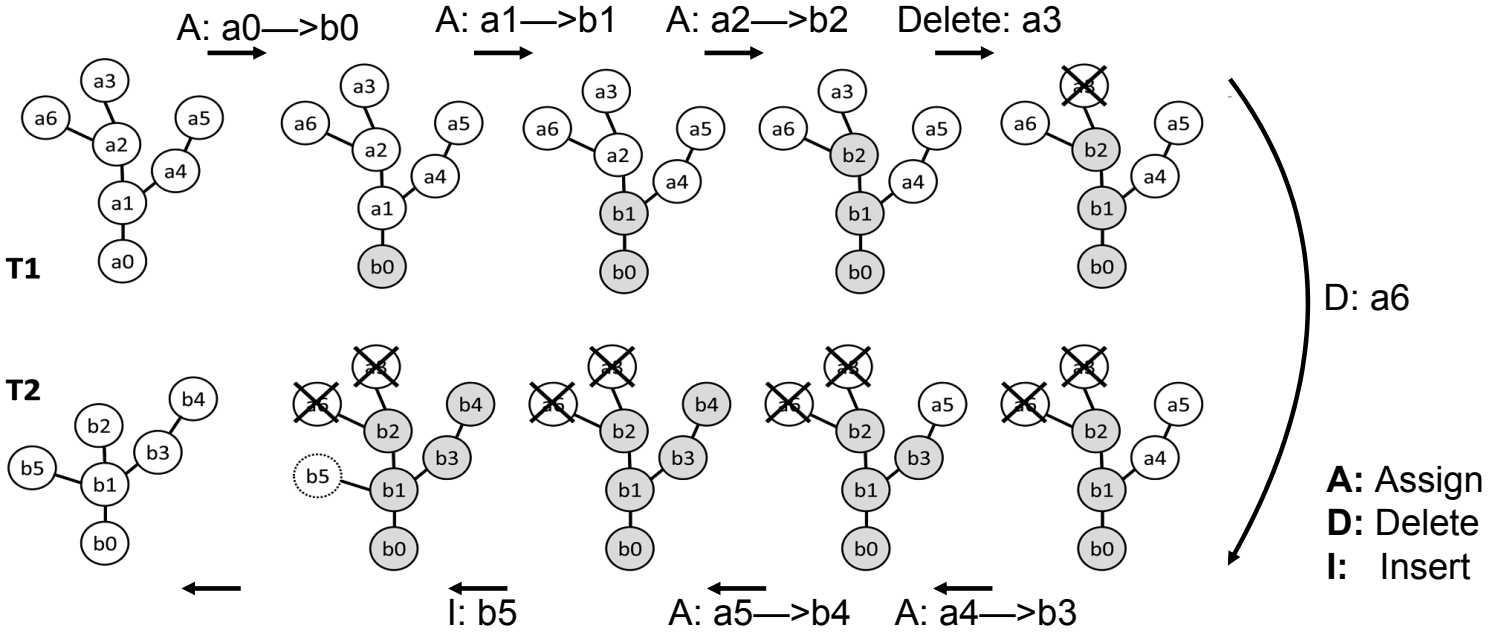
Edit distance

$$d_T(\tau_1, \tau_2) = \frac{d_N(t_1, t_2)}{2\max\left(d_N(t_1, \varepsilon), d_N(\varepsilon, t_2)\right)}$$

Structure-based distance

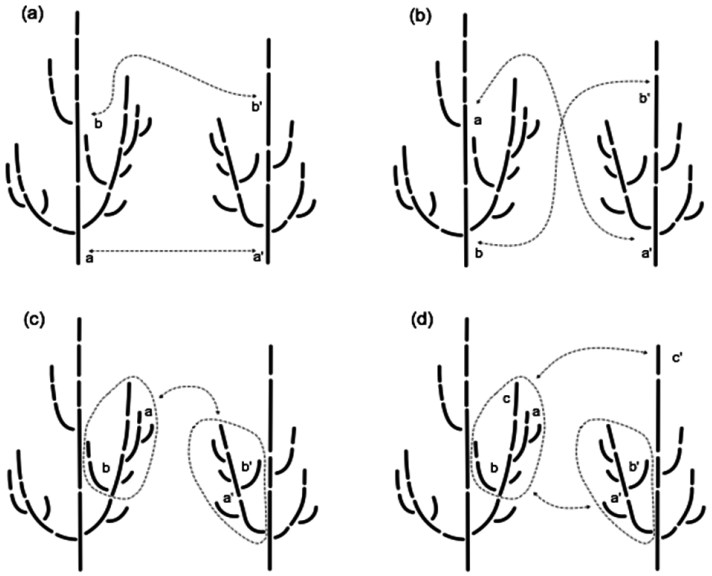
Roots

# Structural Distance



# Operator Cost Function

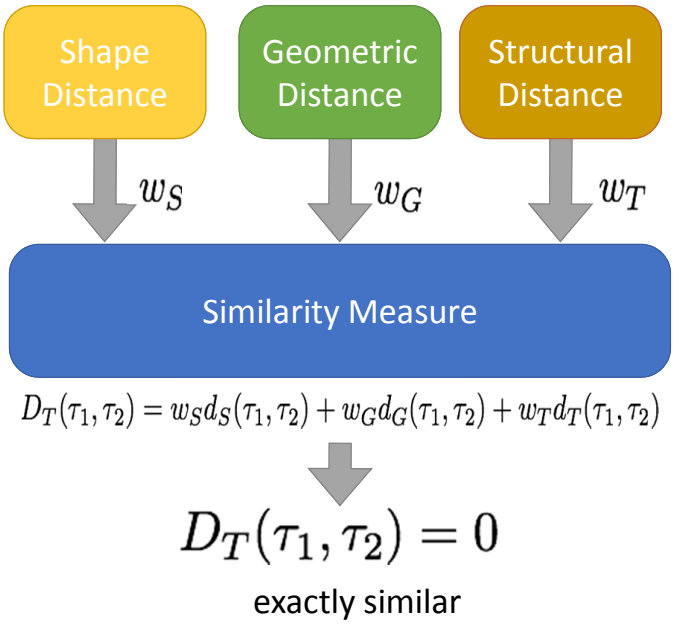
- Distance of two trees is equal to minimal costs of transforming their graphs
- Recursive algorithm that evaluates edit distance [Zhang 1996]




[Ferraro and Godin 2000]

# Similarity Measure

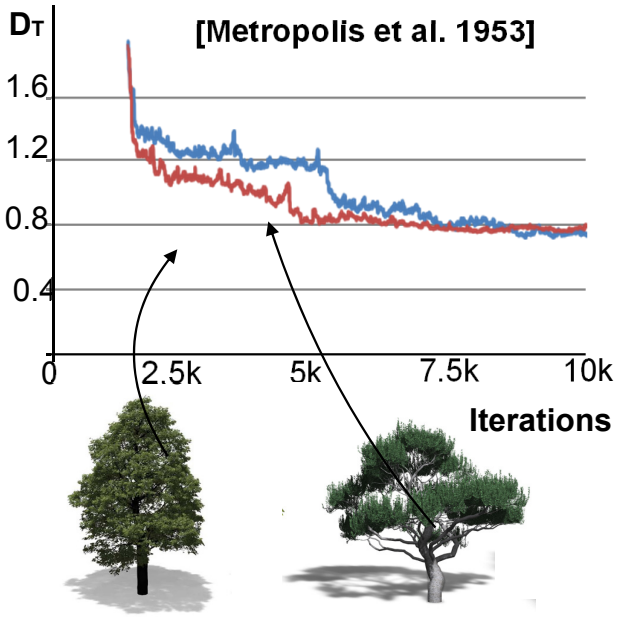
- The sum of shape-, geometry and structure-based distances
- Corresponding weights for each distance (wS, wG, wT)
- Results generated with equal weight



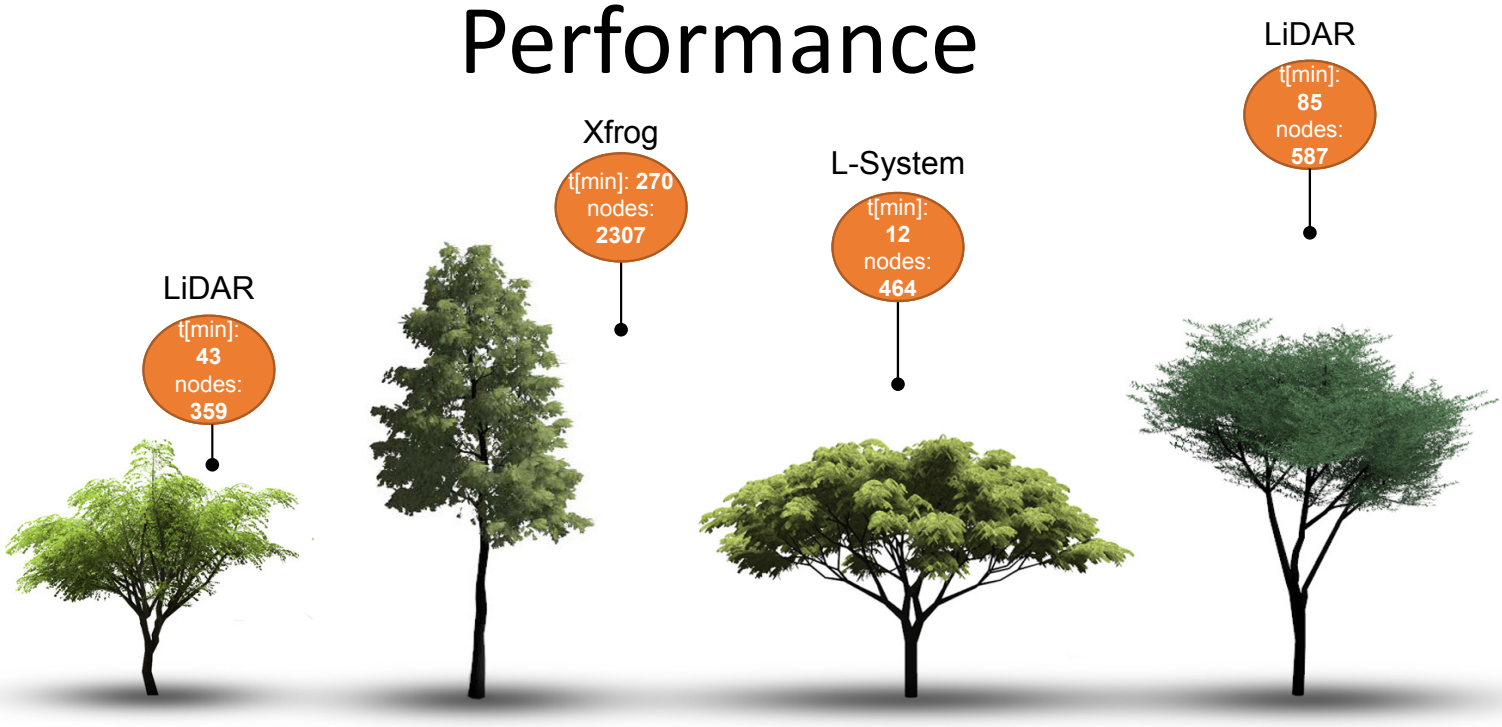
# Optimization of Parameters

- Find parameter set that generates similar trees
- Simulated annealing
- Stochastic sampling based on Metropolis-Hastings
- Solve approximate optimization problem:

$$\underset{\bar{\varphi}_{\mathcal{M}},t}{\operatorname{argmin}} \left( \sum_{\omega_j} D_T\left(\tau^r, \tau^{\mathcal{M}}(\omega_j)\right) \right)$$



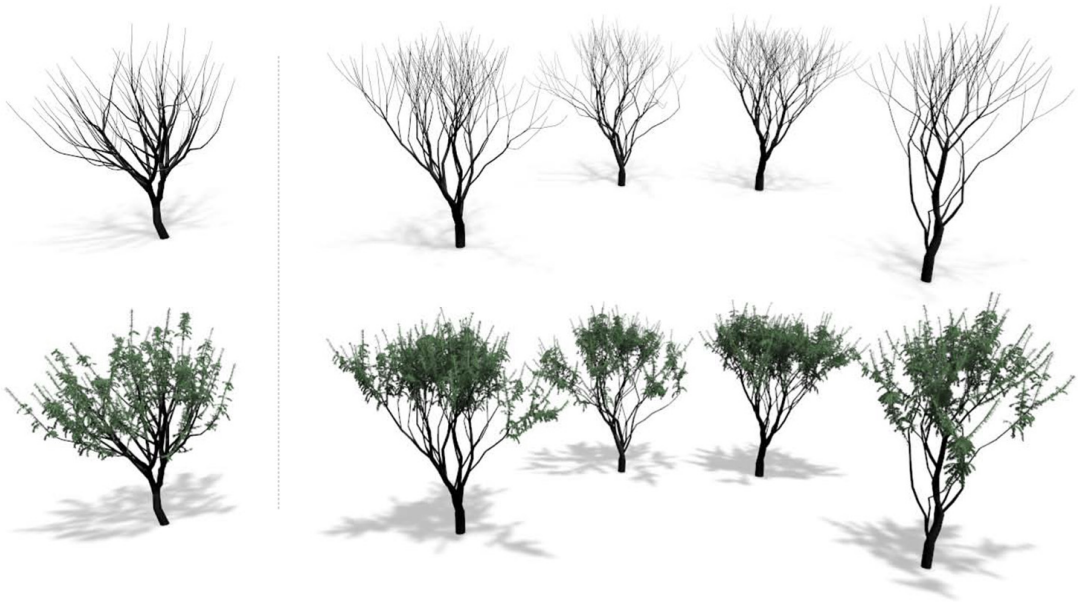
# Performance



# Results

(video)

# Results



# Parameter Changes

Increasing Age

# Environment

(video)

(video)

# Guided Procedural Modeling

Benes, B., Stava, O., Mech, R., and Miller, G.

*Computer Graphics Forum, 2011*

# Approach

- **A top-down approach**
- Divide the model into logical parts
- Isolate them inside boundaries (called guides)
- Let them communicate.

**Guided procedural model**

# Guided Procedural Models

- is a generalization of the environment

# Approach

(video)

# Approach

## Procedural Model

A:dist<$\epsilon$ →F ?X(dist) A
A:dist>$\epsilon$ →G
G→ Tree Crown

## Guided Procedural Model

Guide 1:
A →F ?X A
?X:hit a guide → “send signal” cut

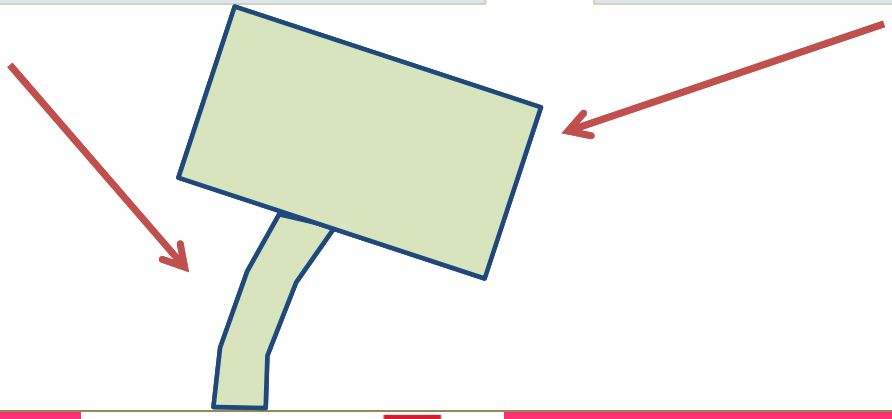
Guide 2:
G→Tree Crown

# Approach

## Guided Procedural Model

Guide 1:
A →F ?X A
?X:hit a guide → “send signal” cut

Guide 2:
G→Tree Crown



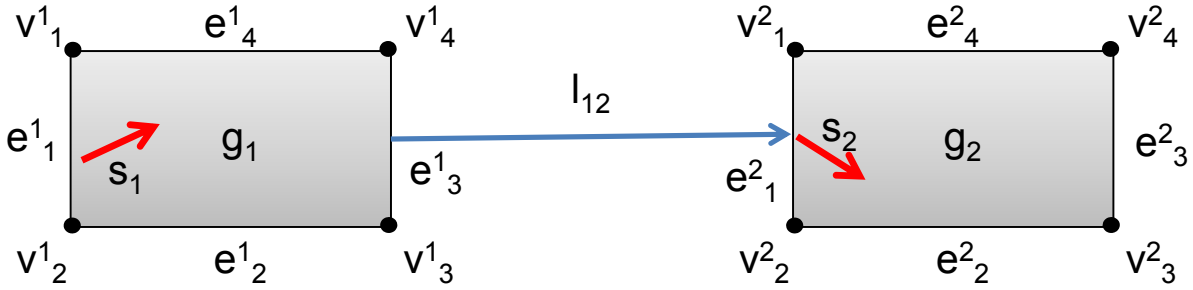
# Approach

Advantages:

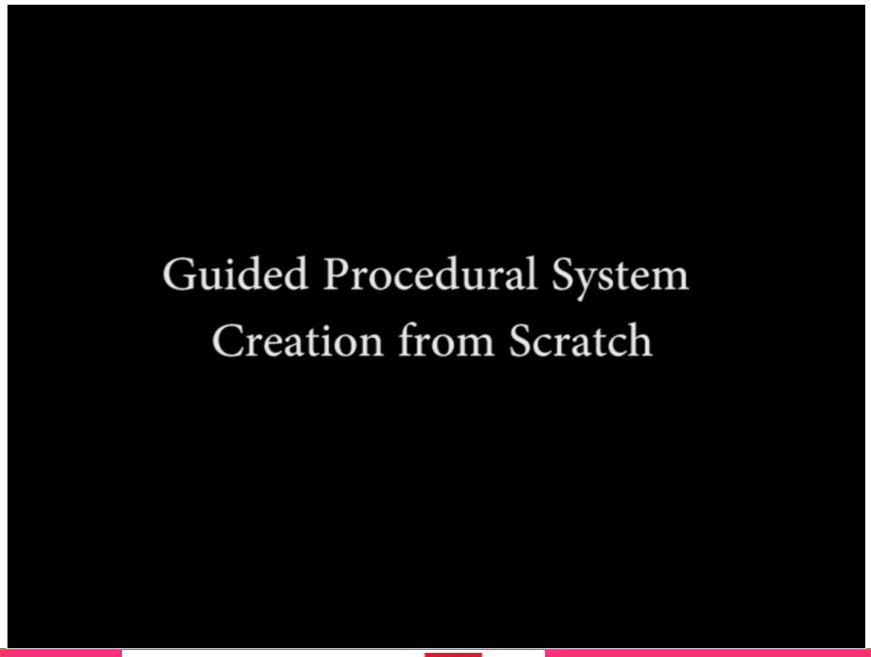
- System is separated into small and simple procedural models.
- PM execution is clear and controllable.
- PM is limited and shaped by the guide.
- A system is built top-down.

# GPM Definition

- ***Guide*** is a polygon.
- ***Link*** is a connection of edges of two guides.
- ***Seed*** initializes the PM
  (axiom, starting location, and direction).



# GPM Definition

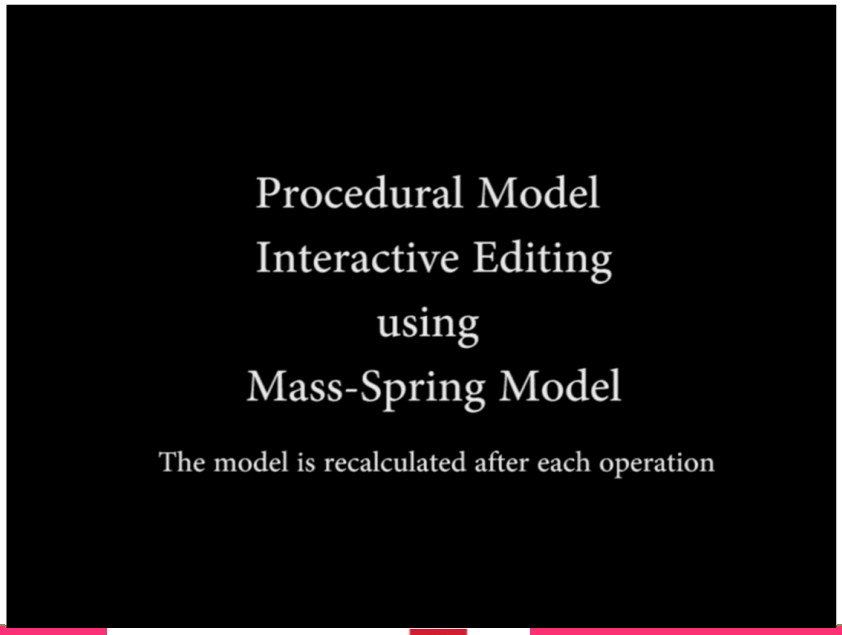



(video)

# Mass-Spring Editing

- Physics can arrange the guides.
- Guides can be locked together.
- Stiffness can be modified.

# Mass-Spring Editing




(video)

# City Layout using GPM

(video)

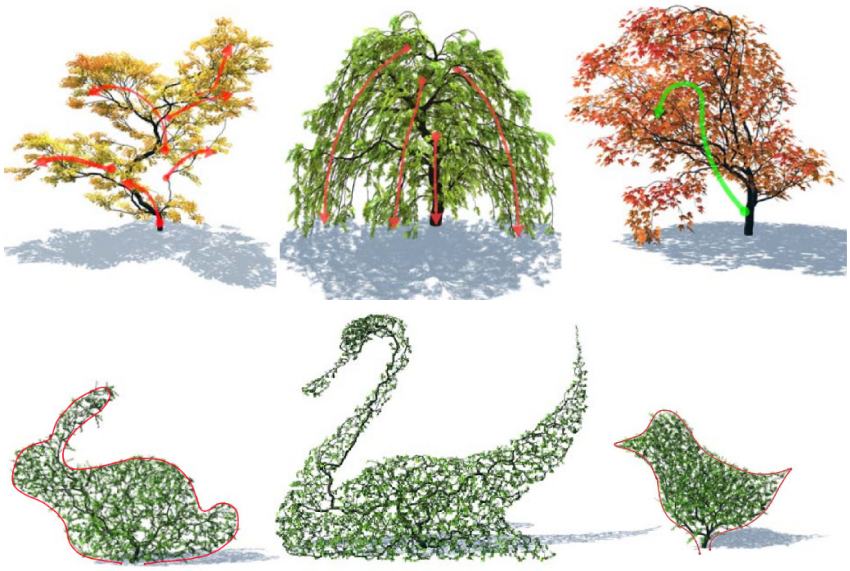
# Conclusions

- Guided procedural models are easy to design and control
- Design
  - Top-down using guides
  - Simpler PMs inside the guides
- User Control
  - Guides provide local control
  - Physics makes interaction more natural

Other papers [Longay12]

# TreeSketch: Interactive Procedural Modeling of Trees on a Tablet

- Guiding procedural generation on tree forms with a tablet
  - Self-organizing trees
  - Competition of branches
- Modeler directs the growth with a procedural brush, interleaving procedural and autonomous growth




Render the Possibilities
SIGGRAPH2016
THE 43RD INTERNATIONAL
CONFERENCE AND EXHIBITION ON
&
Computer Graphics
Interactive Techniques
24-28 JULY
ANAHEIM, CALIFORNIA

Render the Possibilities
SIGGRAPH2016
THE 43RD INTERNATIONAL CONFERENCE AND EXHIBITION ON Computer Graphics & Interactive Techniques

# Inverse Procedural Modeling of 3D Models for Virtual Worlds

JOHANNES GUTENBERG UNIVERSITÄT MAINZ
PURDUE UNIVERSITY

Daniel Aliaga Ilke Demir
Bedrich Benes Michael Wand

# Outline

- Introduction
- Fundamentals of IPM
- IPM Approaches
  - Low-level Priors: Inferring Shape Grammars
  - High-level Priors: Inferring Grammar Parameters
- **Inverse Procedural Modeling Domains**
  - Facades & Layouts
  - Vegetation
  - **Urban Areas**
- Conclusions, Challenges, Open Problems

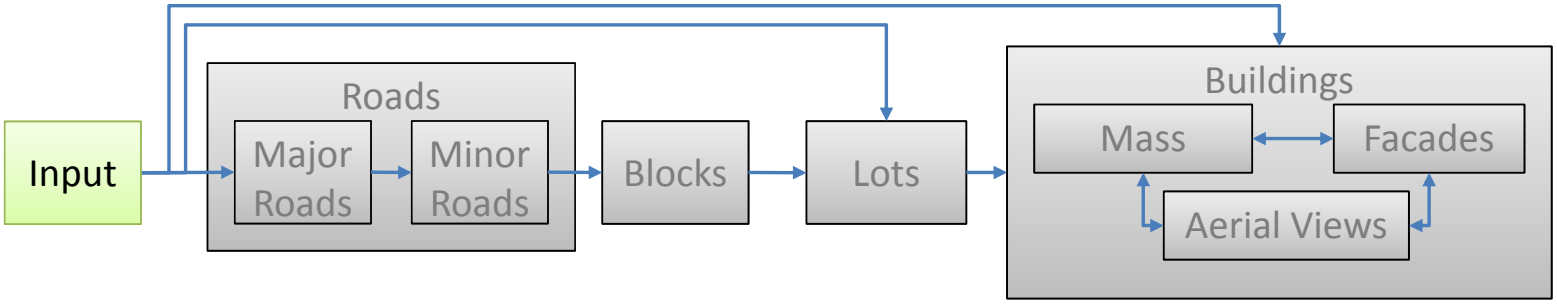
# IPM of Urban Areas

- Urban modeling
  - Forward modeling pipeline
  - Inverse modeling pipeline
- Approaches
  - Image-based methods [Aliaga07, Vanegas10]
    - Style grammars, Manhattan-world grammars
  - Mesh-based methods [Demir14, Vanegas12, Bokeloh10]
    - City proceduralization, inverse city design, double-cut
  - Point-based methods [Demir15, Markus11, Hohhman09, Toshev10]
    - Procedural editing, procedural reconstruction, CityFit, city parsing

# IPM of Urban Areas

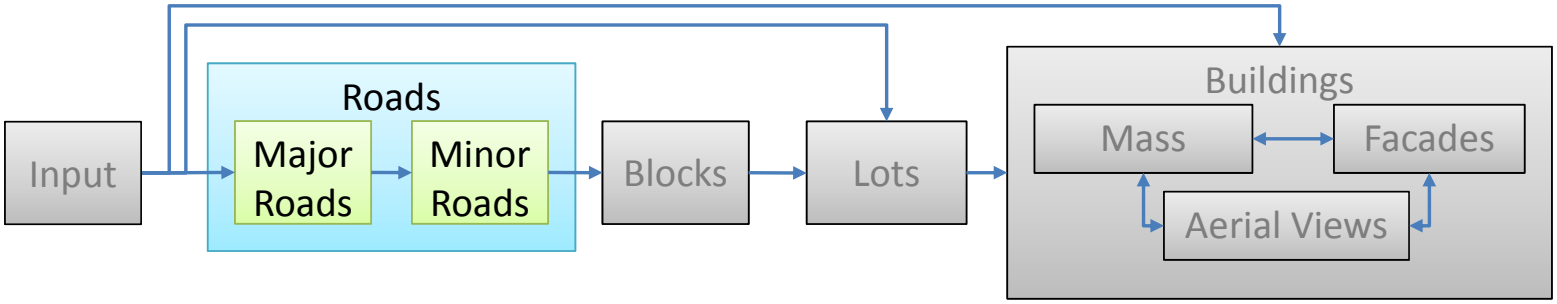
- **Urban modeling**
  - **Forward modeling pipeline**
  - **Inverse modeling pipeline**
- Approaches
  - Image-based methods [Aliaga07, Vanegas10]
    - Style grammars, Manhattan-world grammars
  - Mesh-based methods [Demir14, Vanegas12, Bokeloh10]
    - City proceduralization, inverse city design, double-cut
  - Point-based methods [Demir15, Markus11, Hohhman09, Toshev10]
    - Procedural editing, procedural reconstruction, CityFit, city parsing

# Forward Modeling Pipeline



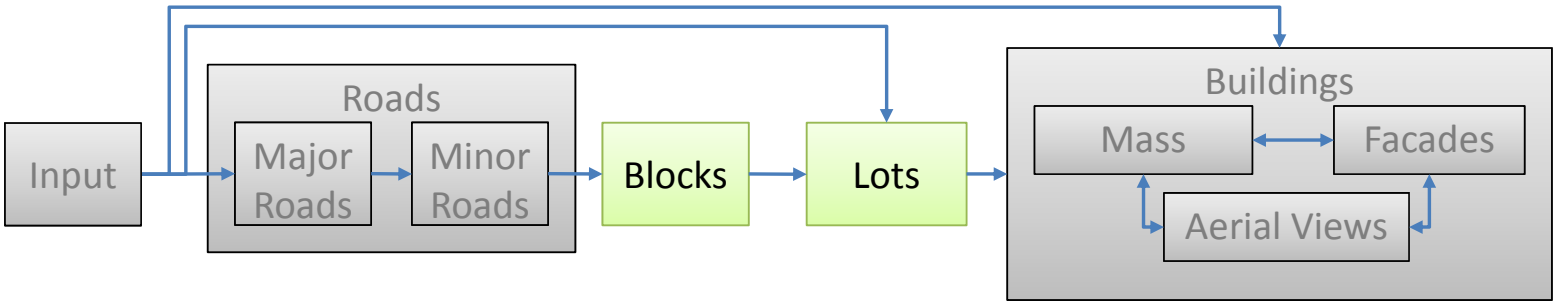

- Input
  - Target architectural design
  - Example 3D model/GIS data/Imagery
  - Socioeconomic data/Elevation data
  - Tensor field

# Forward Modeling Pipeline



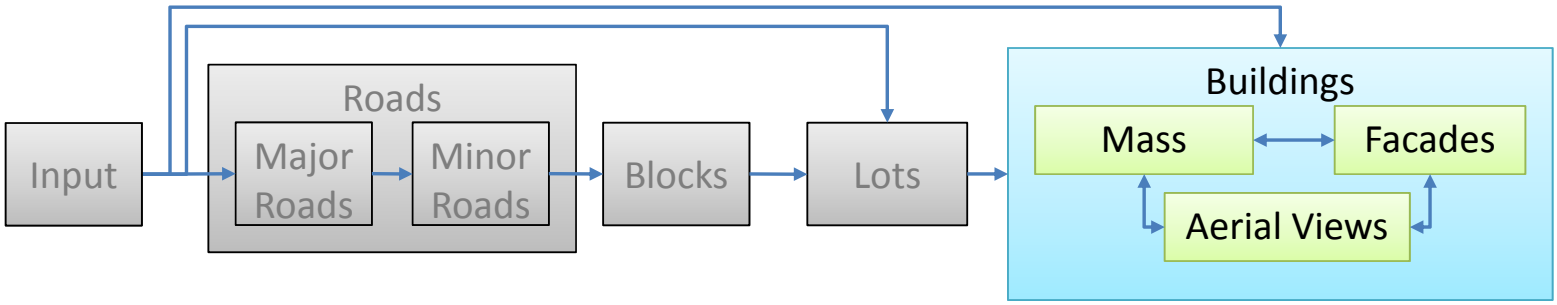

- Road generation
  - Extended L-systems
  - Hyperstreamlines
  - Directed random walks
  - Grow seeds/Traffic Simulation

# Forward Modeling Pipeline



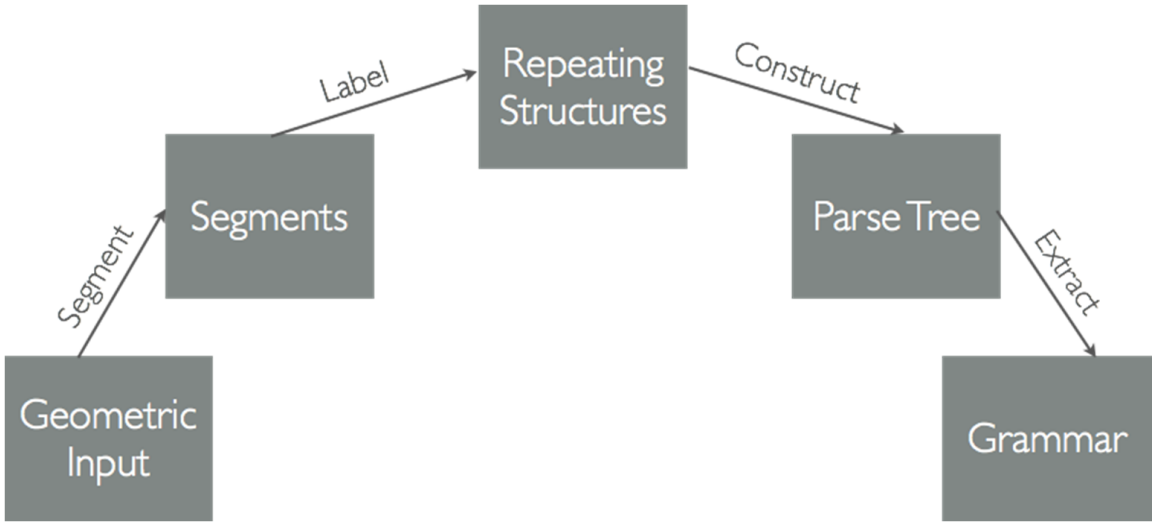

- Blocks and Lots
  - Recursive block subdivision
  - Voronoi-diagram based subdivision

# Forward Modeling Pipeline

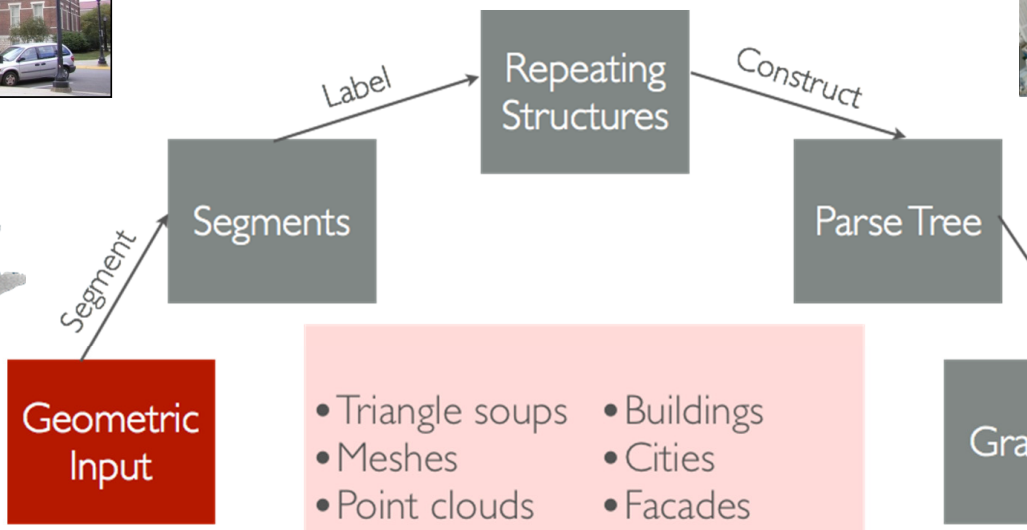



- Buildings
  - Shape/split grammars
    - Mass/façade modeling
  - Build by numbers
  - Image-based synthesis

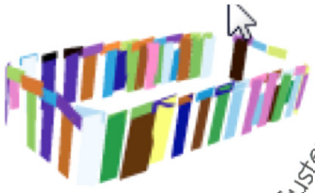
# Inverse Modeling Pipeline

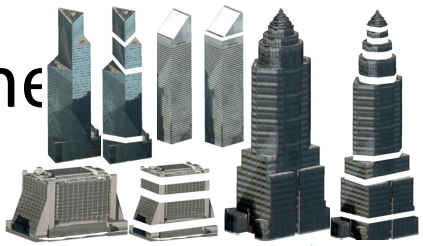


# Inverse Modeling Pipeline



# Inverse Modeling Pipeline





Geometric Input

Cluster

Segments

Label

Repeating Structures

Construct

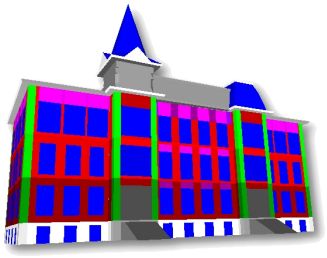
Parse Tree

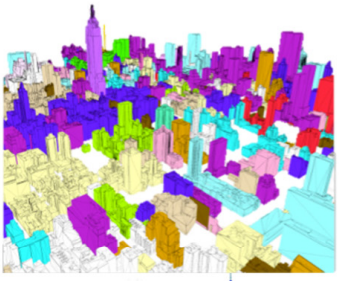
Extract

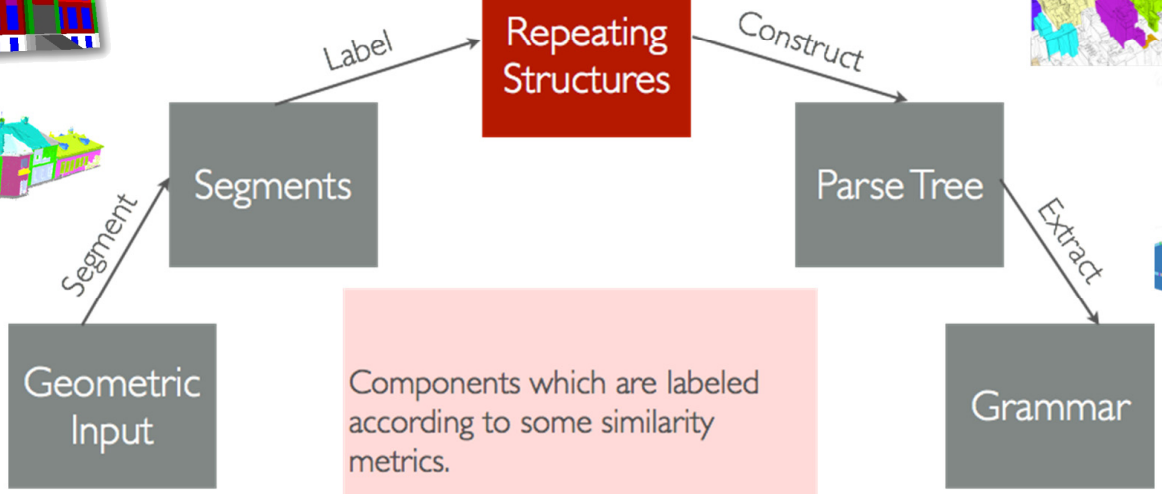
Grammar

Structures in the input that are separable from their surroundings by significant changes.

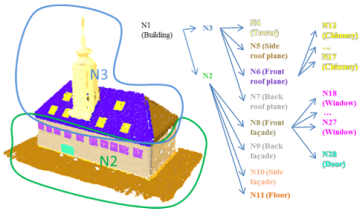
# Inverse Modeling Pipeline

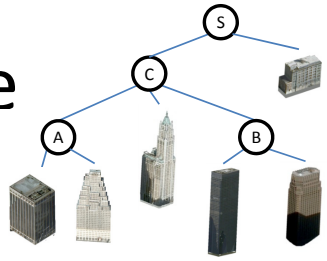






# Inverse Modeling Pipeline




Label

Repeating Structures

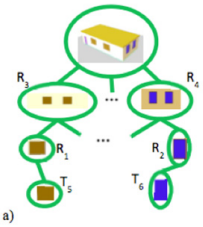
Construct

Segment

Segments

Parse Tree

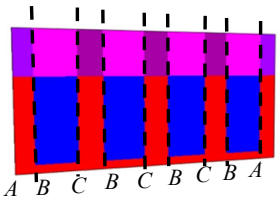
Extract




Geometric Input

Hierarchical organization of the components, that contains their spatial and/or similarity relations.

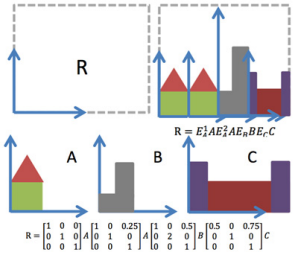
Grammar

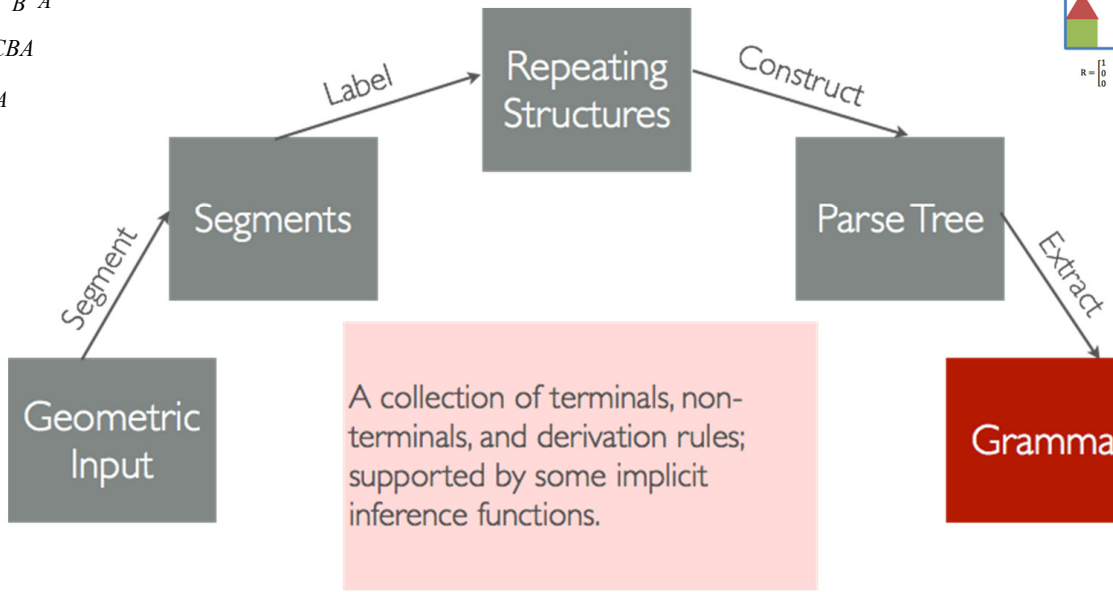


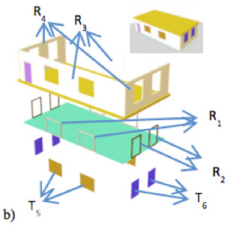

$F = ABCBCBCBA$

$F \rightarrow A(BC)^*BA$

# Inverse Modeling Pipeline



# IPM of Urban Areas

- Urban modeling
  - Forward modeling pipeline
  - Inverse modeling pipeline
- Approaches
  - **Image-based methods [Aliaga07, Vanegas10]**
    - **Style grammars, Manhattan-world grammars**
  - Mesh-based methods [Demir14, Vanegas12, Bokeloh10]
    - City proceduralization, inverse city design, double-cut
  - Point-based methods [Demir15, Markus11, Hohhman09, Toshev10]
    - Procedural editing, procedural reconstruction, CityFit, city parsing

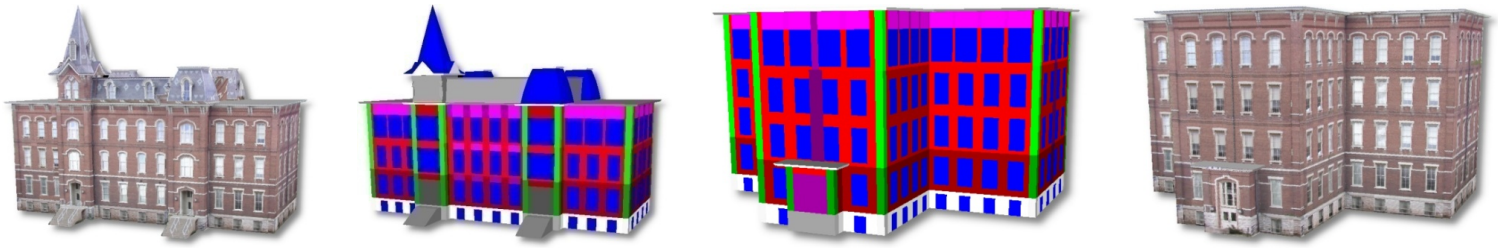
# Style Grammars for Interactive Visualization of Architecture

Daniel G. Aliaga , Paul A. Rosen , Daniel R. Bekins.

*IEEE TVCG, 2007*

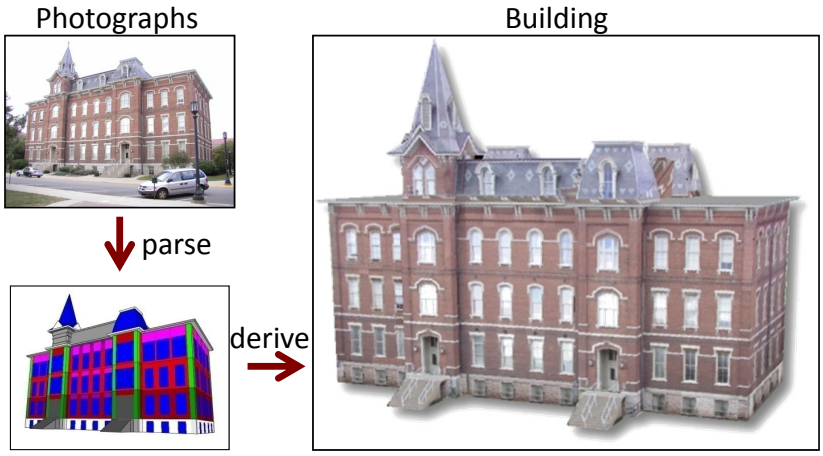
# Style Grammars for Interactive Visualization of Architecture

- Images -> Grammar instances
- Infer a grammar for creating architecture and buildings
- Enable rapid generation of a building in style of others



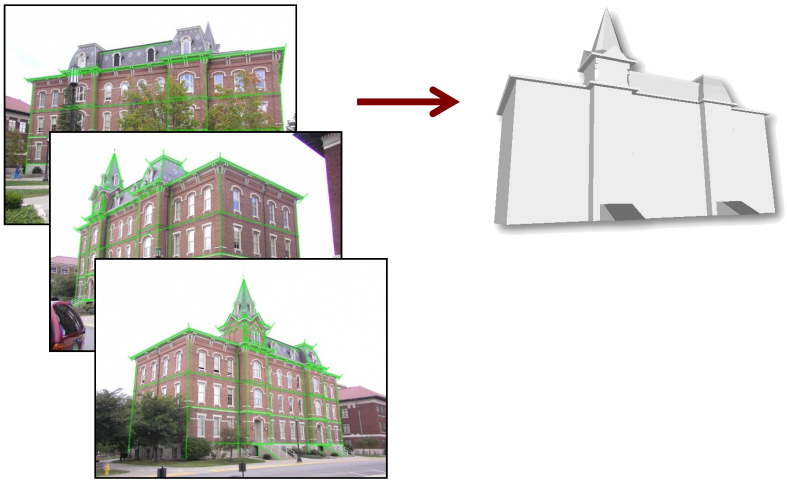
# Approach

- *parse* captured images to create a grammar
- *derive* new buildings using the data from the grammar



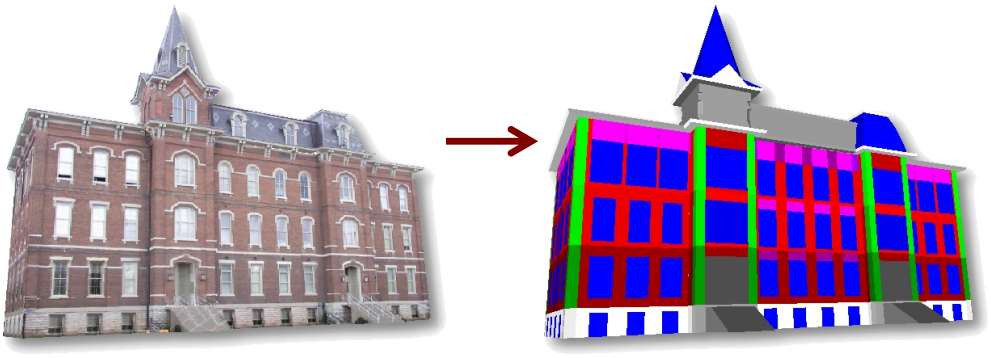
# Overview

1. Recover geometric model from sparse image set.



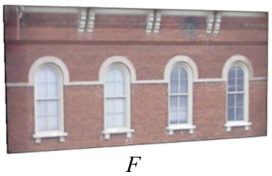
# Overview

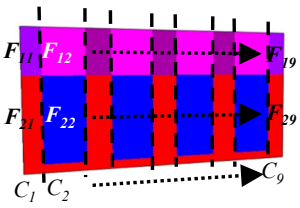
1. Recover geometric model from sparse image set.
2. Subdivide into feature regions (brick, windows, etc.)

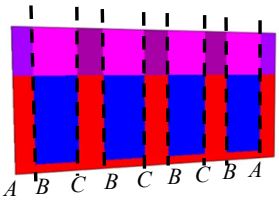


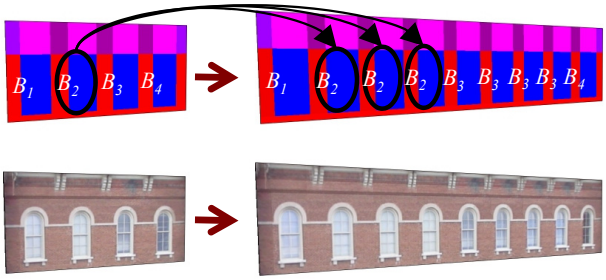
# Overview

1. Recover geometric model from sparse image set.
2. Subdivide into feature regions (brick, windows, etc.)
3. Derive and use a face schema (rule+parameters).



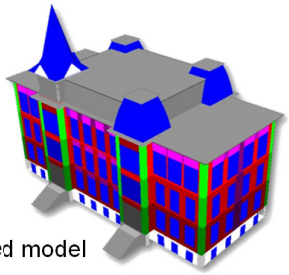

$F = ABCBCBCBA$

$F \rightarrow A(BC)^*BA$

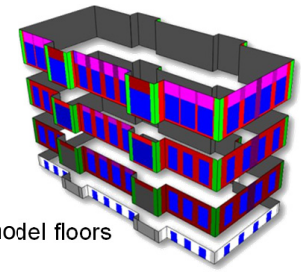
$F' = ABCBCBCBCBCBCBCBA$

# Overview

1. Recover geometric model from sparse image set.
2. Subdivide into feature regions (brick, windows, etc.)
3. Derive and use a face schema.
4. Derive and use a floor schema.



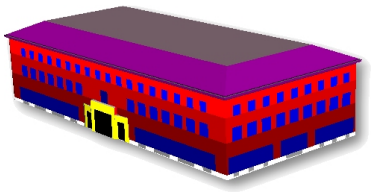
captured model

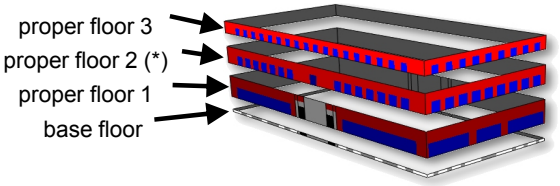


model floors

# Overview

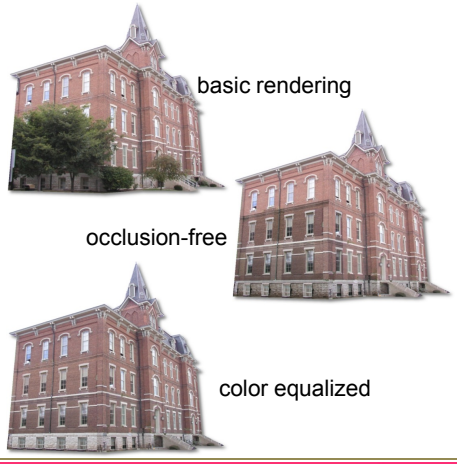
1. Recover geometric model from sparse image set.
2. Subdivide into feature regions (brick, windows, etc.)
3. Derive and use a face schema.
4. Derive and use a floor schema.
5. Derive and use a model schema.





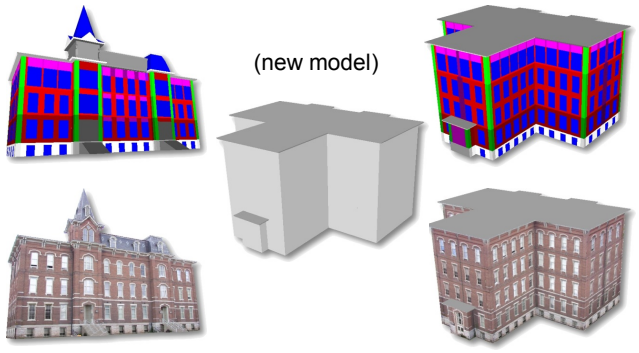
# Overview

1. Recover geometric model from sparse image set.
2. Subdivide into feature regions (brick, windows, etc.)
3. Derive and use a face schema.
4. Derive and use a floor schema.
5. Derive and use a model schema.
6. Texture the new model.



# Overview

1. Recover geometric model from sparse image set.
2. Subdivide into feature regions (brick, windows, etc.)
3. Derive and use a face schema.
4. Derive and use a floor schema.
5. Derive and use a model schema.
6. Texture the new model.
7. Generate new models!

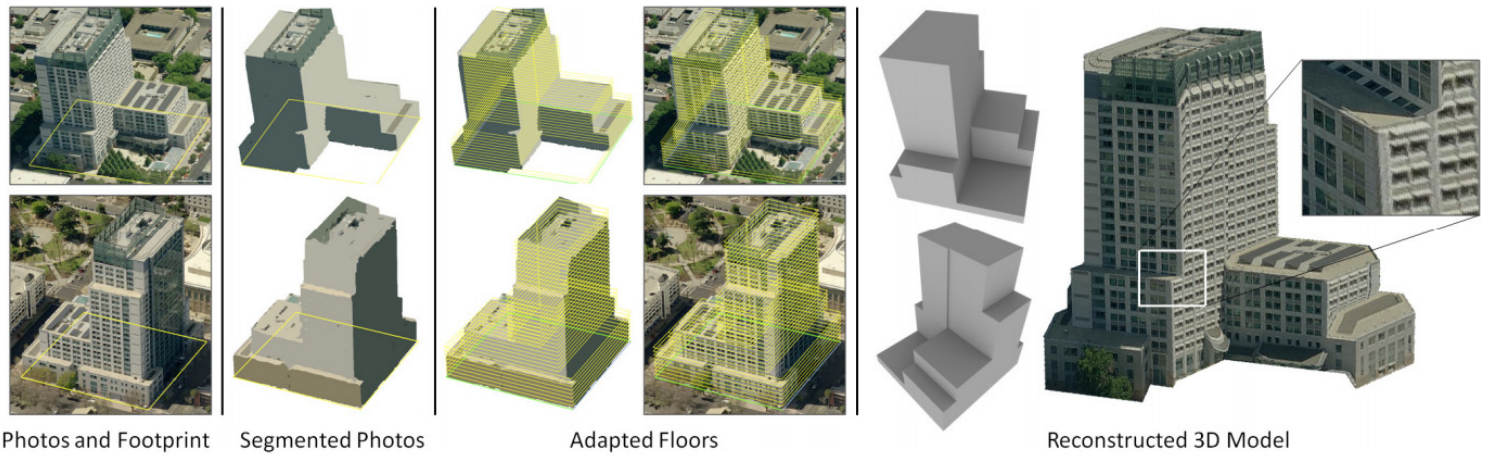


Style Grammars for Interactive Visualization of Architecture

# Results

Other papers [Vanegas10]

# Building Reconstruction using Manhattan-World Grammars



# IPM of Urban Areas

- Urban modeling
  - Forward modeling pipeline
  - Inverse modeling pipeline
- Approaches
  - Image-based methods [Aliaga07, Vanegas10]
    - Style grammars, Manhattan-world grammars
  - **Mesh-based methods [Demir14, Vanegas12, Bokeloh10]**
    - **City proceduralization, inverse city design, double-cut**
  - Point-based methods [Demir15, Markus11, Hohhman09, Toshev10]
    - Procedural editing, procedural reconstruction, CityFit, city parsing

# Proceduralization of Buildings at City Scale

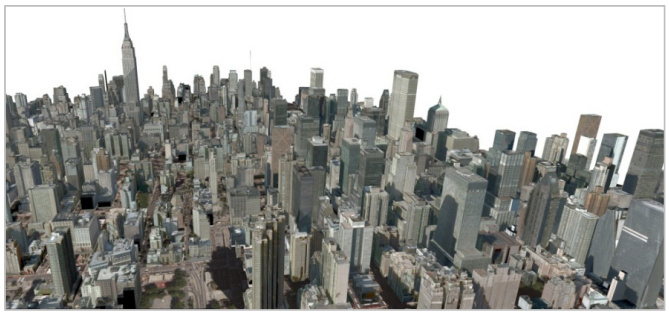
Ilke Demir, Daniel G. Aliaga , Bedrich Benes.

*International Conference on 3D Vision (3DV), 2014.*

# Proceduralization of Buildings at City Scale

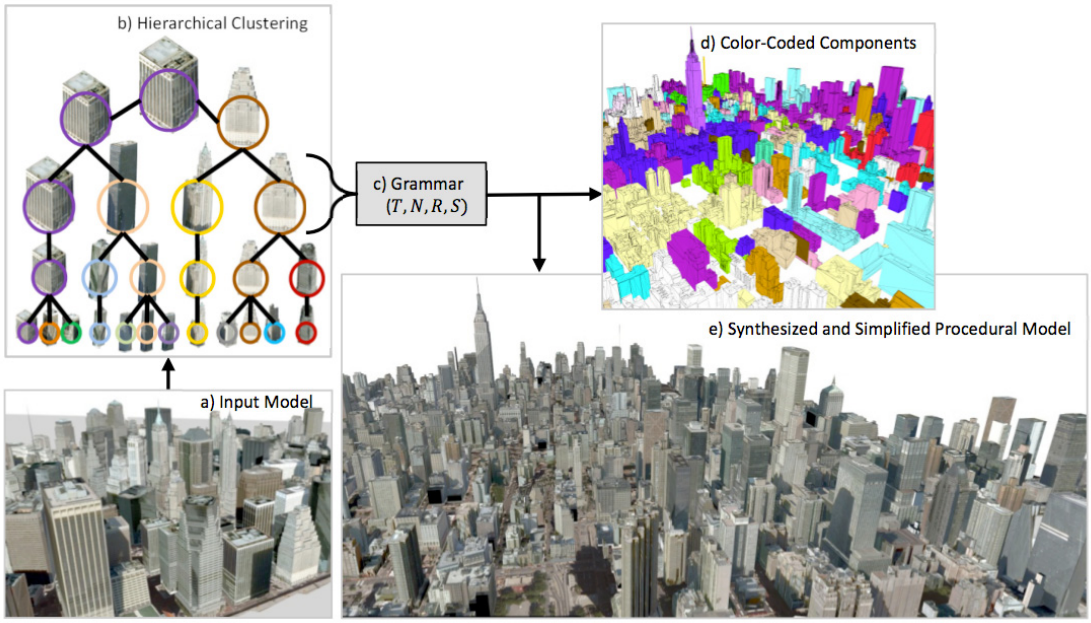
- Any urban area -> Procedural Representation
- Hierarchical simplification to organize based on similarity
- Hierarchical clustering to organize based on structure

Input model



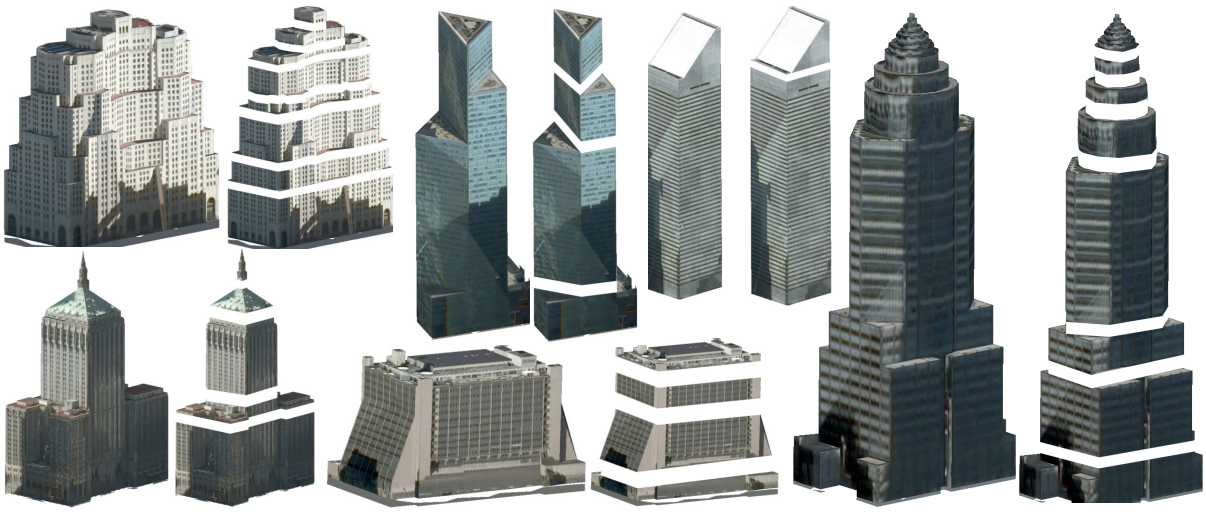
Synthesized city

# Approach


b) Hierarchical Clustering
d) Color-Coded Components
c) Grammar
$(T, N, R, S)$
e) Synthesized and Simplified Procedural Model
a) Input Model

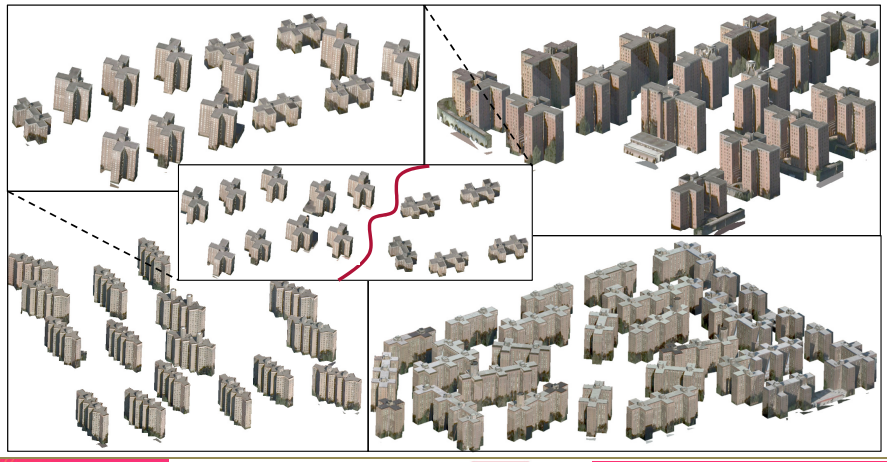
# Overview

1. Extract feature vectors by rendering.



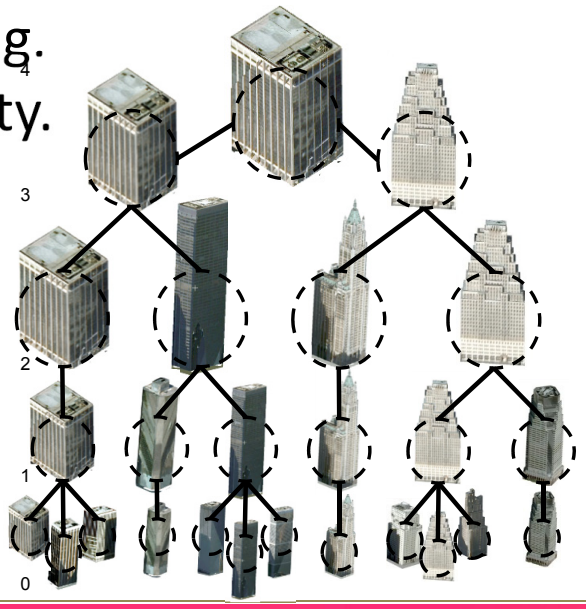
# Overview

1. Extract feature vectors by rendering.
2. Cluster based on weighted similarity.



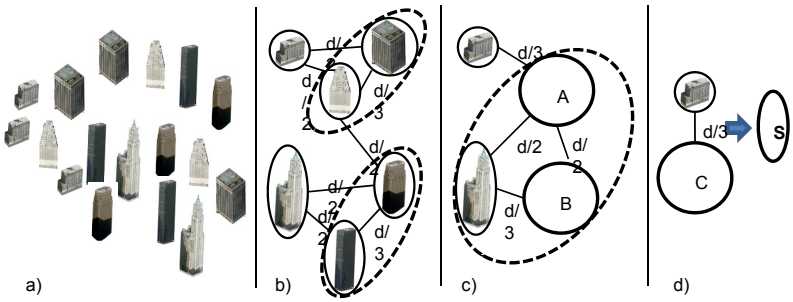
# Overview

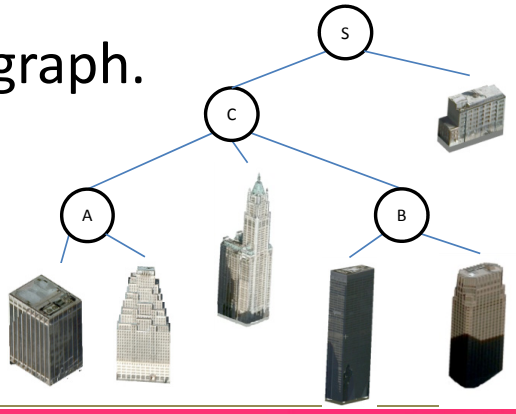
1. Extract feature vectors by rendering.
2. Cluster based on weighted similarity.
3. Select a simplification level.



# Overview

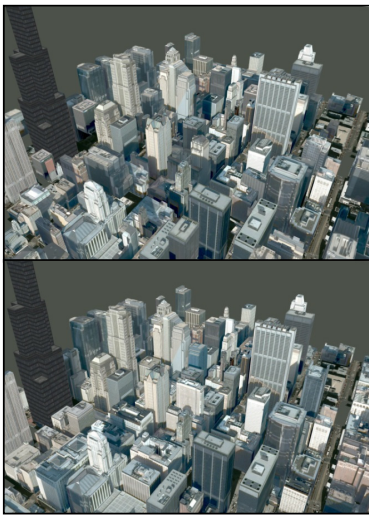
1. Extract feature vectors by rendering.
2. Cluster based on weighted similarity.
3. Select a simplification level.
4. Generate and partition the terminal graph.

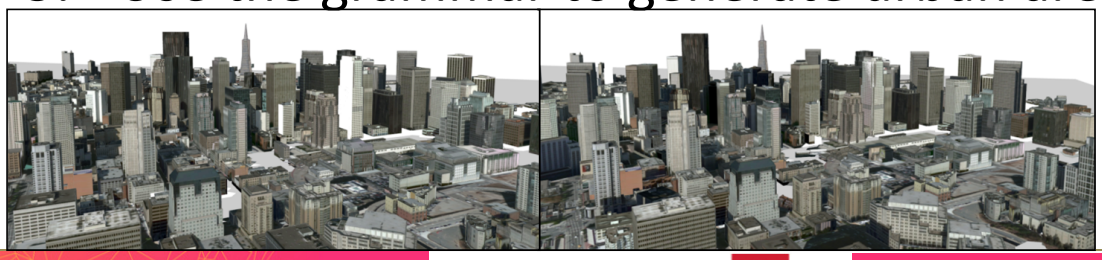


# Overview

1. Extract feature vectors by rendering.
2. Cluster based on weighted similarity.
3. Select a simplification level.
4. Generate and partition the terminal graph.
5. Use the grammar to generate urban areas!





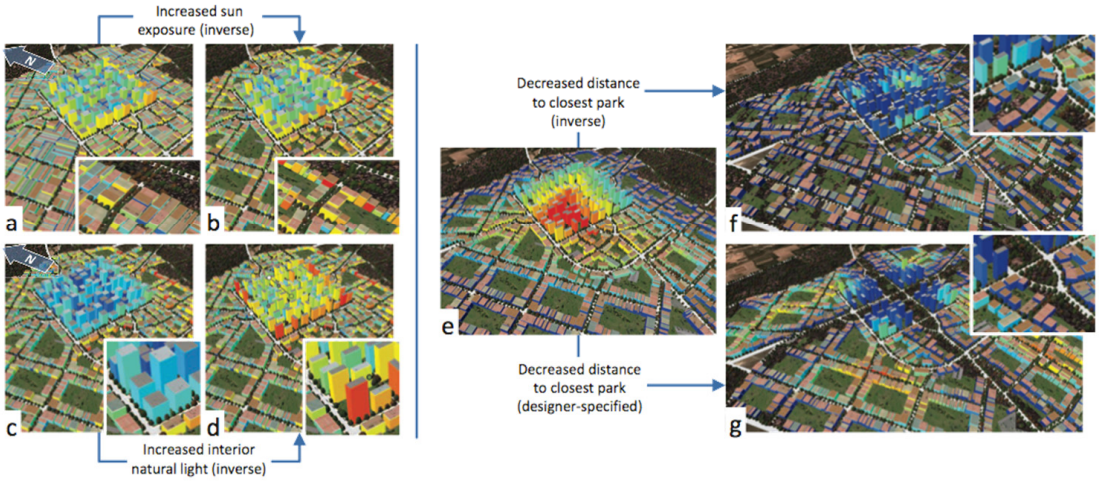
# Results (video)

De-instanced & Proceduralized
New York

Other papers [Vanegas12]

# Inverse Design Of Urban Procedural Models

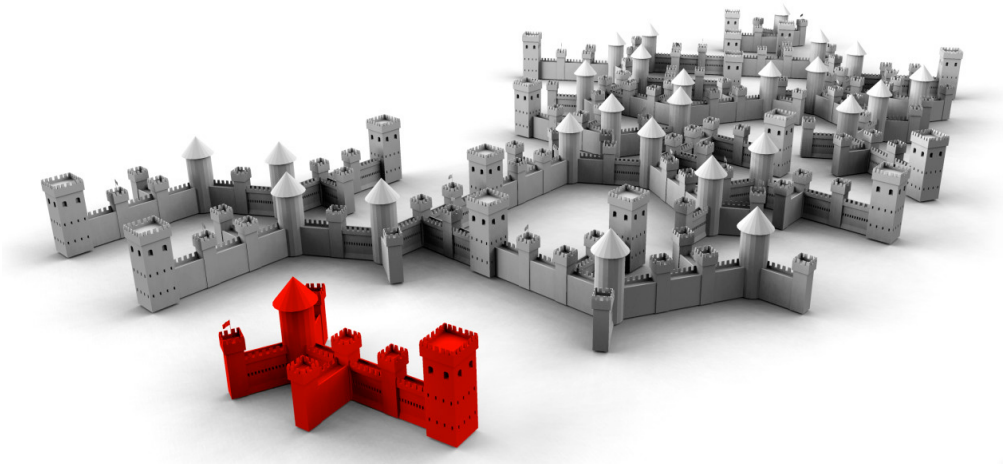
- As explained as an MCMC approach in “High-level Priors”Section



Other papers [Bokeloh10]

# A Connection Between Partial Symmetry and Inverse Procedural Modeling

- “Double-cut” as explained in “Low-level Priors” Section



# IPM of Urban Areas

- Urban modeling
  - Forward modeling pipeline
  - Inverse modeling pipeline
- Approaches
  - Image-based methods [Aliaga07, Vanegas10]
    - Style grammars, Manhattan-world grammars
  - Mesh-based methods [Demir14, Vanegas12, Bokeloh10]
    - City proceduralization, inverse city design, double-cut
  - **Point-based methods [Demir15, Markus11, Hohhman09, Toshev10]**
    - **Procedural editing, procedural reconstruction, CityFit, city parsing**

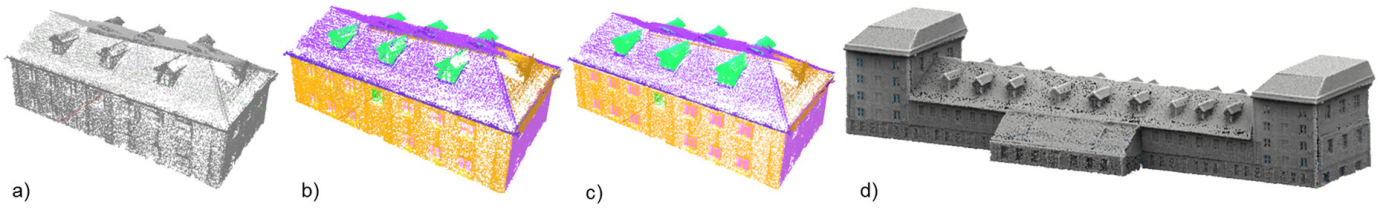
# Procedural Editing of 3D Building Point Clouds

Ilke Demir, Daniel G. Aliaga , Bedrich Benes.

*ICCV, 2015.*

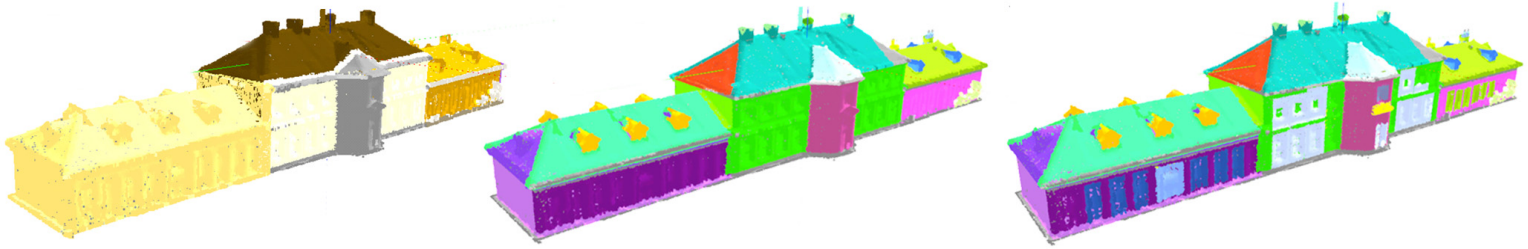
# Procedural Editing of 3D Building Point Clouds

- Any building point cloud -> Procedural representation
- A proceduralization method to convert segments into a grammar
- A completion method that exploits repetitions
- A structure-preserving direct procedural point cloud editing tool



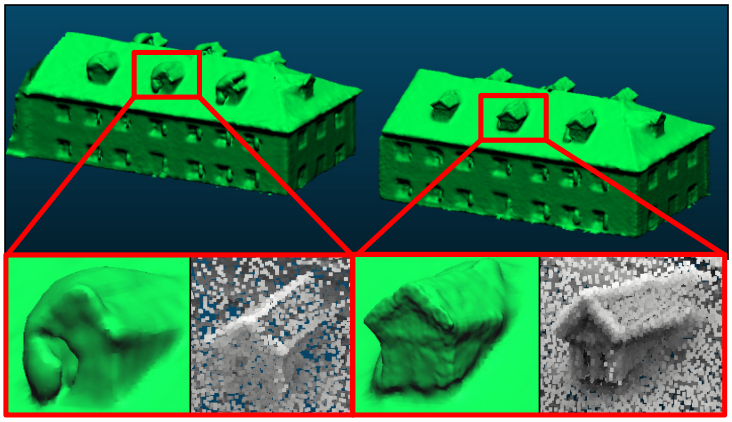
# Overview

1. Semi-auto segmentation with varying granularity. (Ransac + Euclidian + template matching)



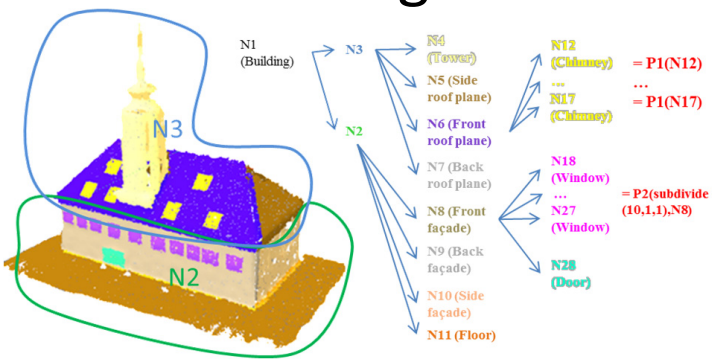
# Overview

1. Semi-auto segmentation with varying granularity.
2. Consensus model to complete the segments.



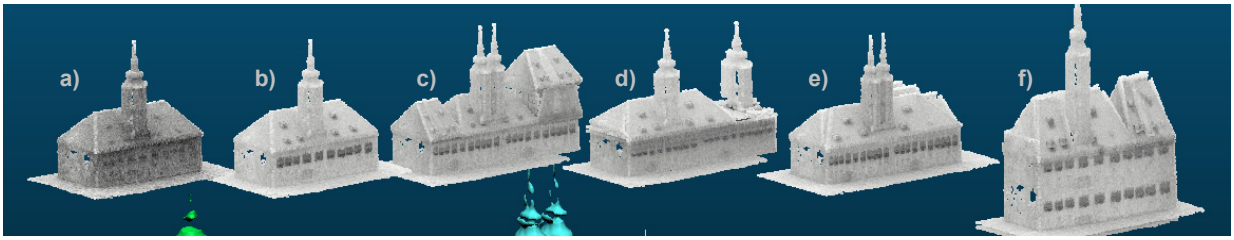
# Overview

1. Semi-auto segmentation with varying granularity.
2. Consensus model to complete the segments.
3. Proceduralization to generate the grammar.

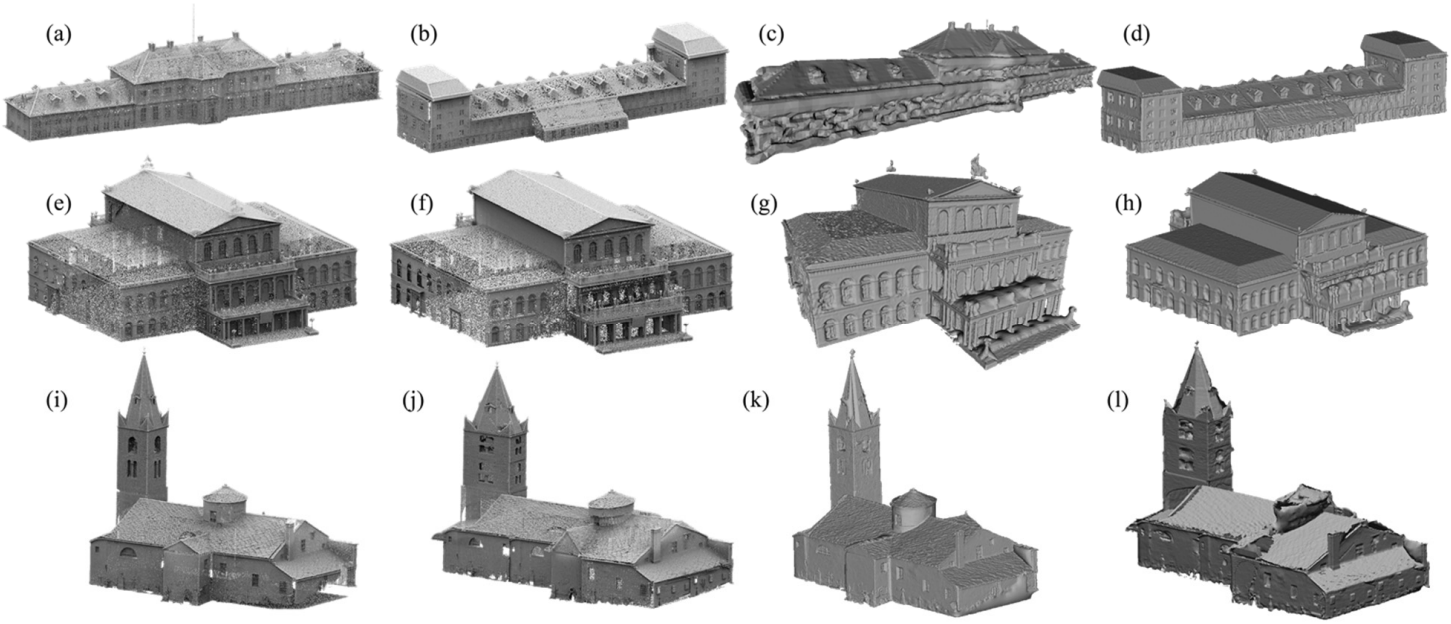


# Overview

1. Semi-auto segmentation with varying granularity.
2. Consensus model to complete the segments.
3. Proceduralization to generate the grammar.
4. Editing operations to synthesize all!



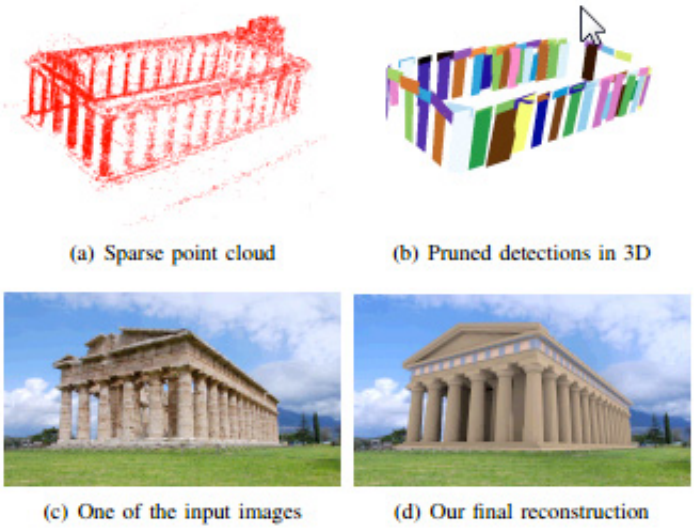
# Results



Other papers [Markus11]

# Procedural 3d Building Reconstruction Using Shape Grammars And Detectors

- Images + grammar + detectors -> reconstruction
  - Terminals: list of shapes
  - Image: assets
  - Arc3D: point cloud
  - Rules: structural info
  - Vision module: asset attributes



(a) Sparse point cloud (b) Pruned detections in 3D

(c) One of the input images (d) Our final reconstruction

# Outline

- Introduction
- Fundamentals of IPM
- IPM Approaches
  - Low-level Priors: Inferring Shape Grammars
  - High-level Priors: Inferring Grammar Parameters
- Inverse Procedural Modeling Domains
  - Facades & Layouts
  - Vegetation
  - Urban Areas
- **Conclusions, Challenges, Open Problems**

# Conclusions

- On the one hand…
  - Procedural modeling is powerful but requires writing detailed grammars
- On the other hand…
  - There are many model databases which lack structural and functional representations

# Conclusions

- Our Multidisciplinary Goal:
  - Take advantage of existing models to represent them in a more flexible way to enable modelers to automatically generate new content

# Conclusions

- However, converting existing models into procedural representations (aka **IPM**) is a significant challenge

# Summary

- This course presents
  - **Fundamentals of IPM**
  - Approaches with Low-level and High-level Priors
  - IPM Applications in Various Domains

# Summary

- This course presents
  - Fundamentals of IPM
  - **Approaches with Low-level and High-level Priors**
  - IPM Applications in Various Domains

# Summary

- This course presents
  - Fundamentals of IPM
  - Approaches with Low-level and High-level Priors
  - **IPM Applications in Various Domains**

# Current Projects

- Less structured sym
- Combine N level
- F , everywhere!
- In.nite buildings!

Coming soon!

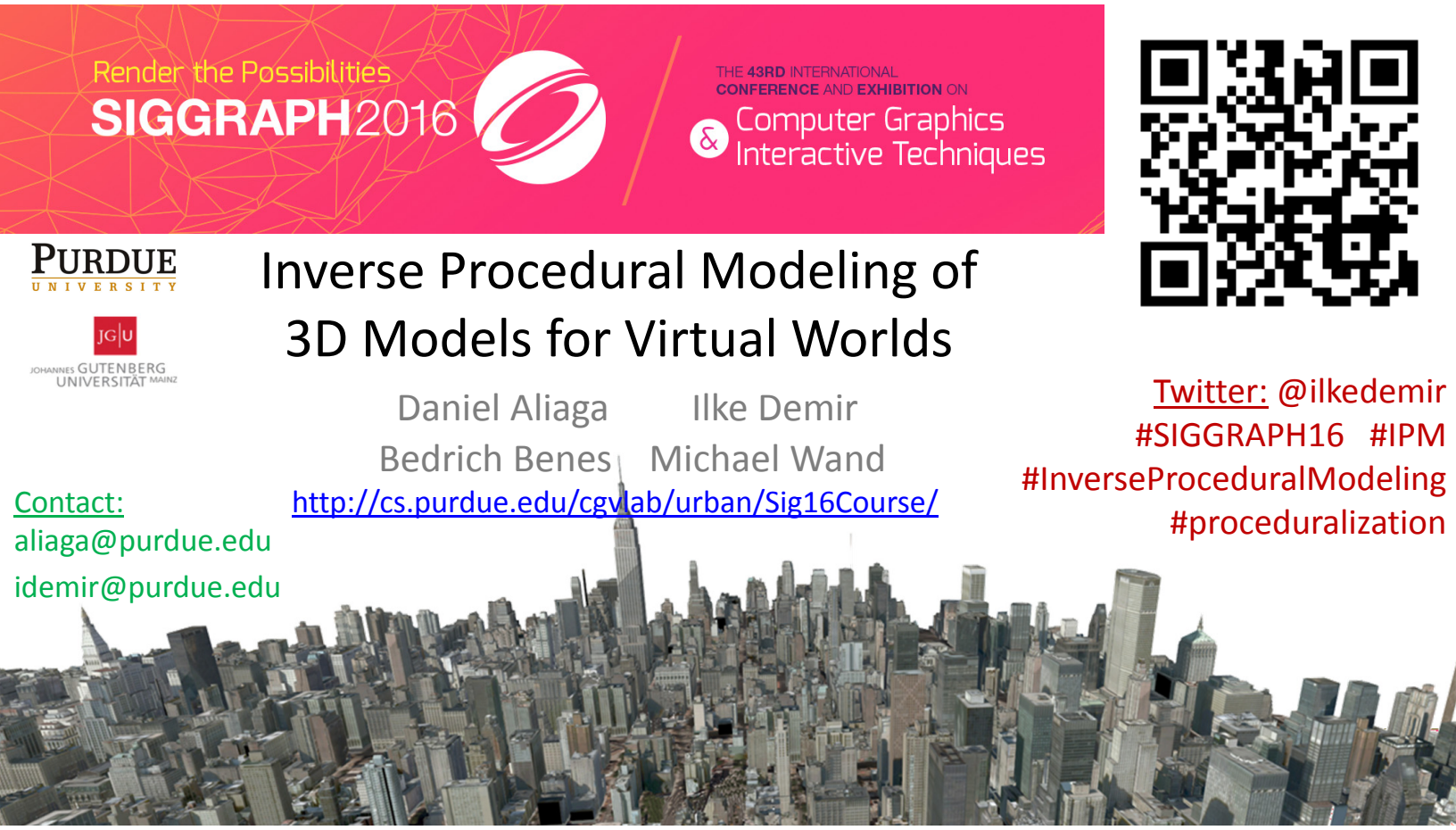
# Lecturers would like to thank…

- All contributors in the field of IPM!
- Reviewers, modelers, users, designers, colleagues
- Purdue CGVLab
- Purdue HPCG Lab

- Funding agencies:
  - NSF
  - Intel Visual Computing Institute
  - Adobe Research
  - Intel
  - Google








PURDUE UNIVERSITY

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

# Inverse Procedural Modeling of 3D Models for Virtual Worlds

Daniel Aliaga  Ilke Demir
Bedrich Benes  Michael Wand

http://cs.purdue.edu/cgvlab/urban/Sig16Course/

Contact:
aliaga@purdue.edu
idemir@purdue.edu

Twitter: @ilkedemir
#SIGGRAPH16 #IPM
#InverseProceduralModeling
#proceduralization