中国科学技术大学
University of Science and Technology of China


GAMES 102在线课程

# 几何建模与处理基础

刘利刚
中国科学技术大学

中国科学技术大学

University of Science and Technology of China

GAMES102

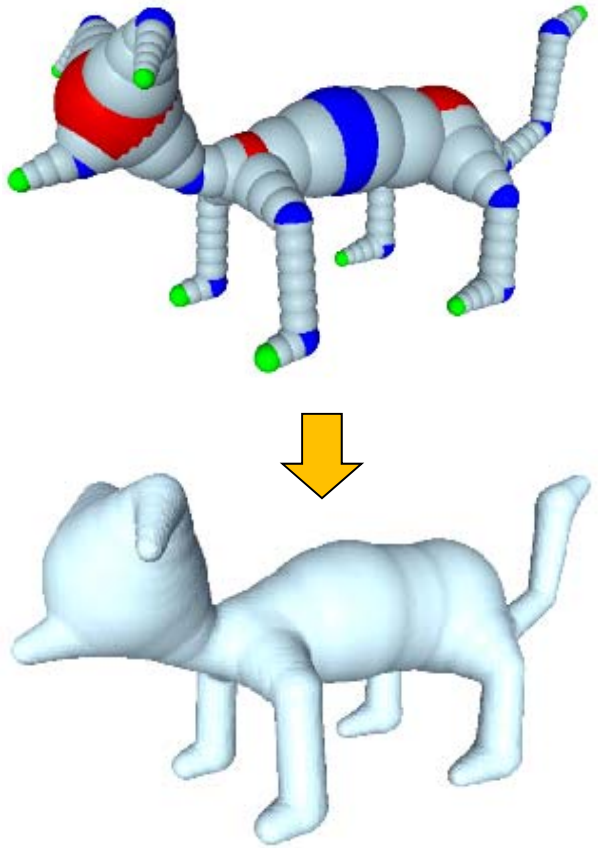
GAMES 102在线课程：几何建模与处理基础

# 纹理合成

# 回顾：曲面重建或建模

曲面重建

曲面建模

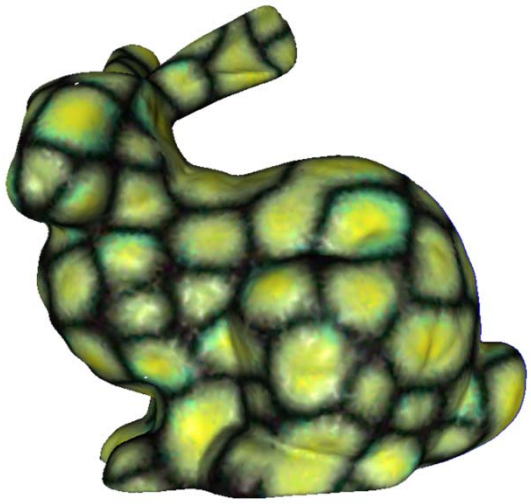


# Textured Surfaces

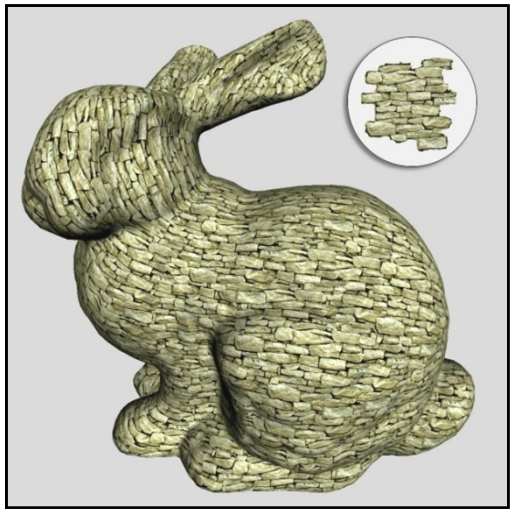
Real object

Model without texture

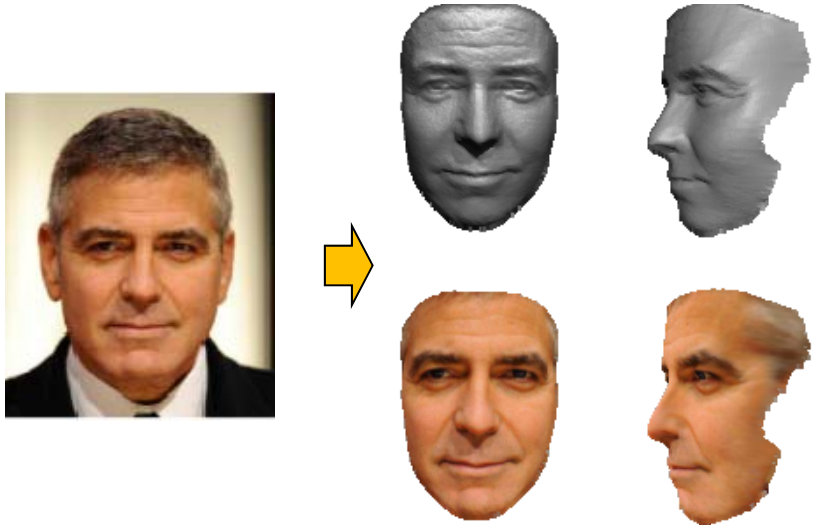
Textured model



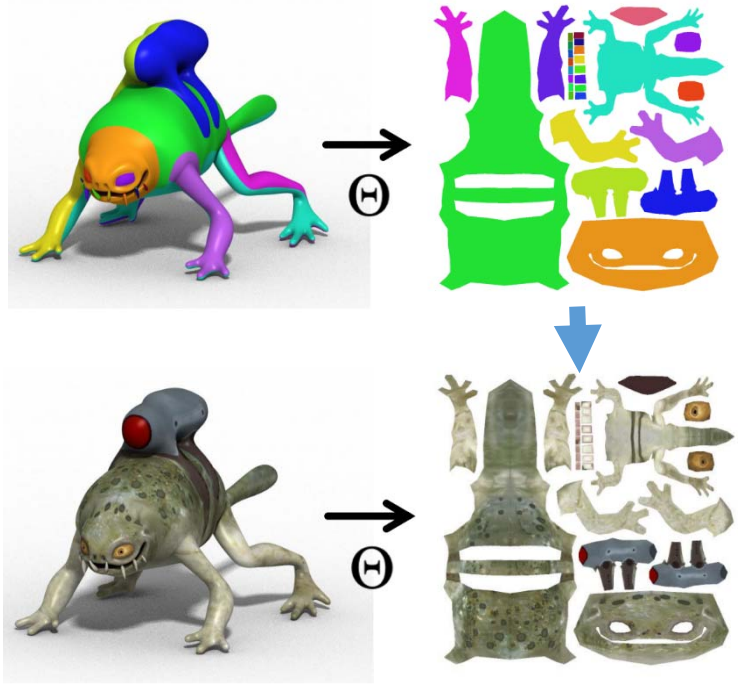
Textured models



# Texturing 3D Models

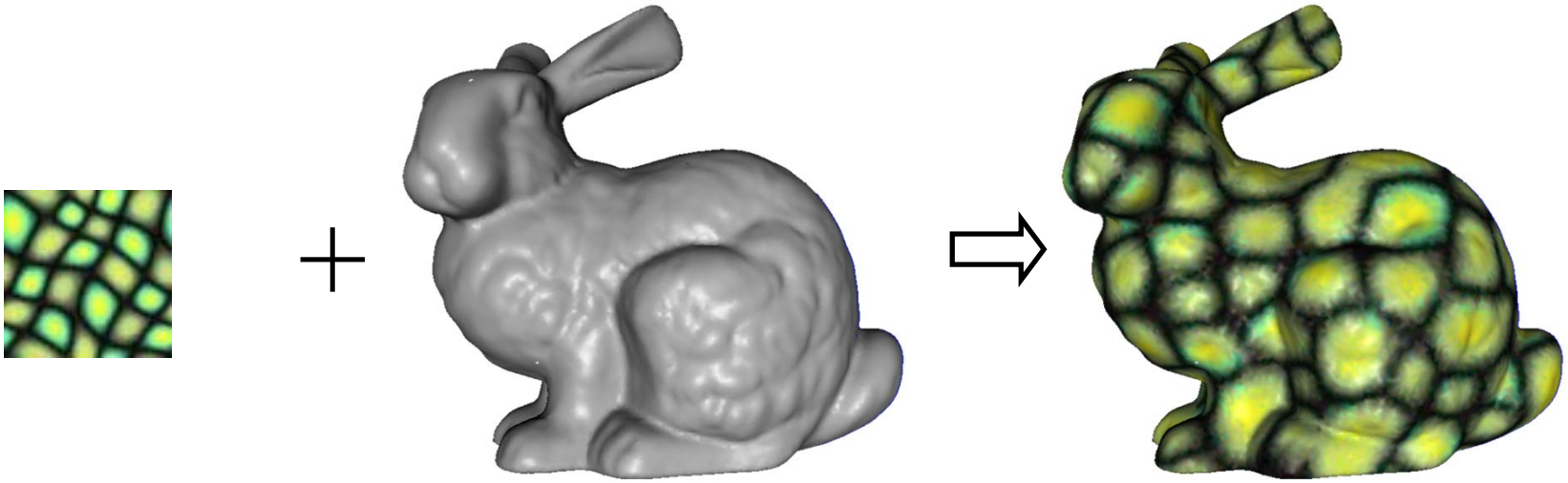


From input images

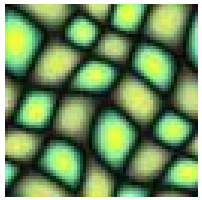


Manually designed

# Texture Synthesis

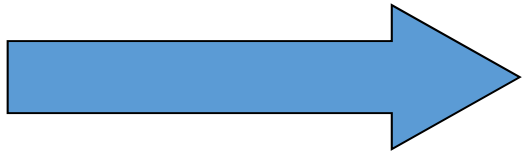


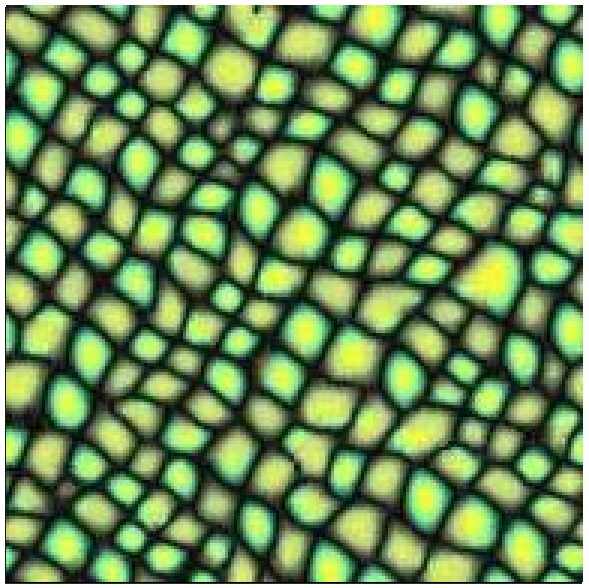
# 2D Texture Synthesis



Input
Sample

Texture Synthesis

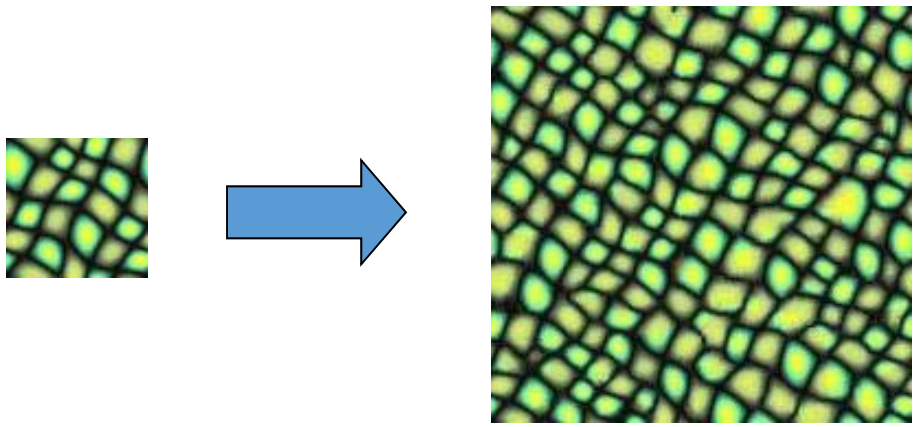




Result

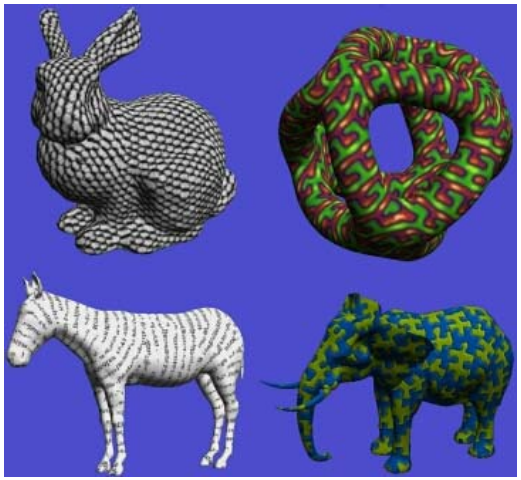
# Texture Synthesis

- Given a sample texture *I*, the goal is to synthesize a new texture *J* that
  - looks like the input sample
  - no part of result is the duplicate of the sample



*I* *J*

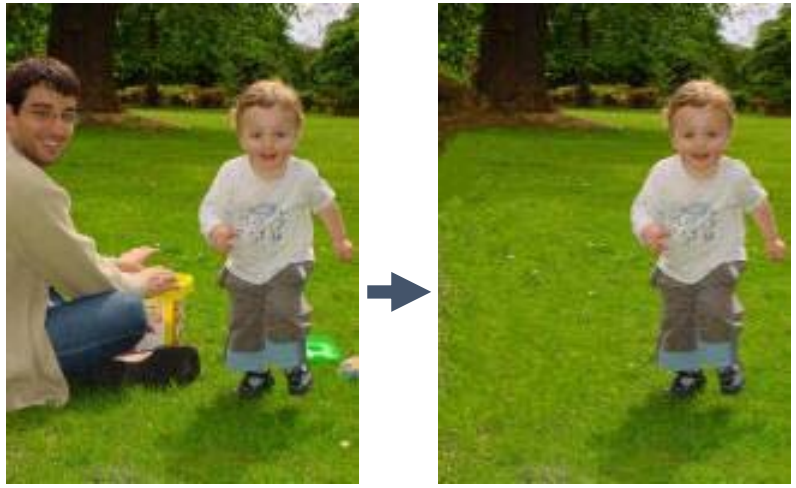
# Applications



Texture Mapping

Texture Replacement

Special Effects

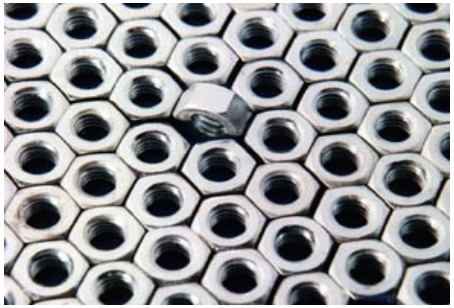


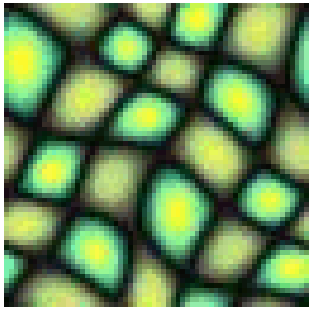
Object Removal / Hole Filling

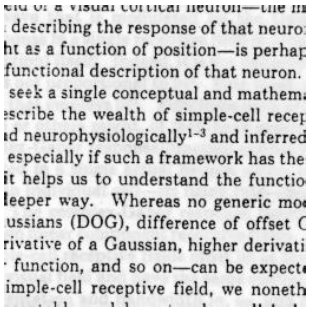
# 1. Textures

# What is texture?

- Characterized by appearance
- Elements arranged quasi-randomly



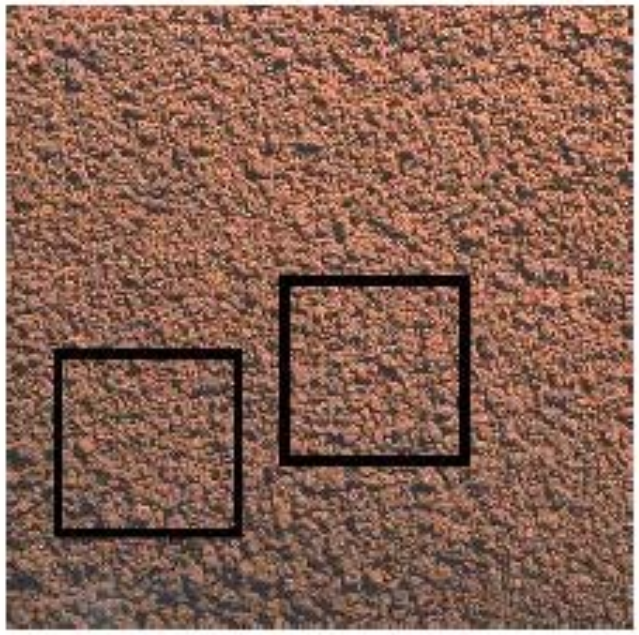






# Image VS. Texture

(a)



(b)

(a1)

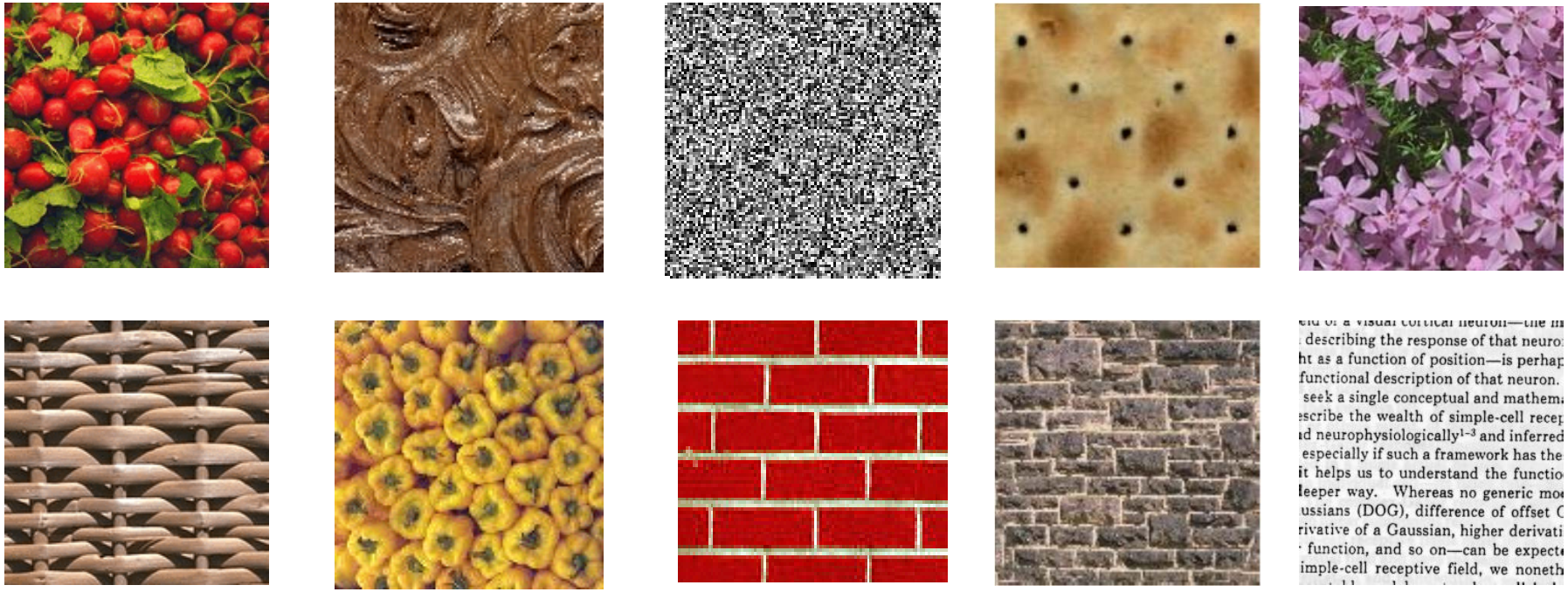
(a2)

(b1)

(b2)

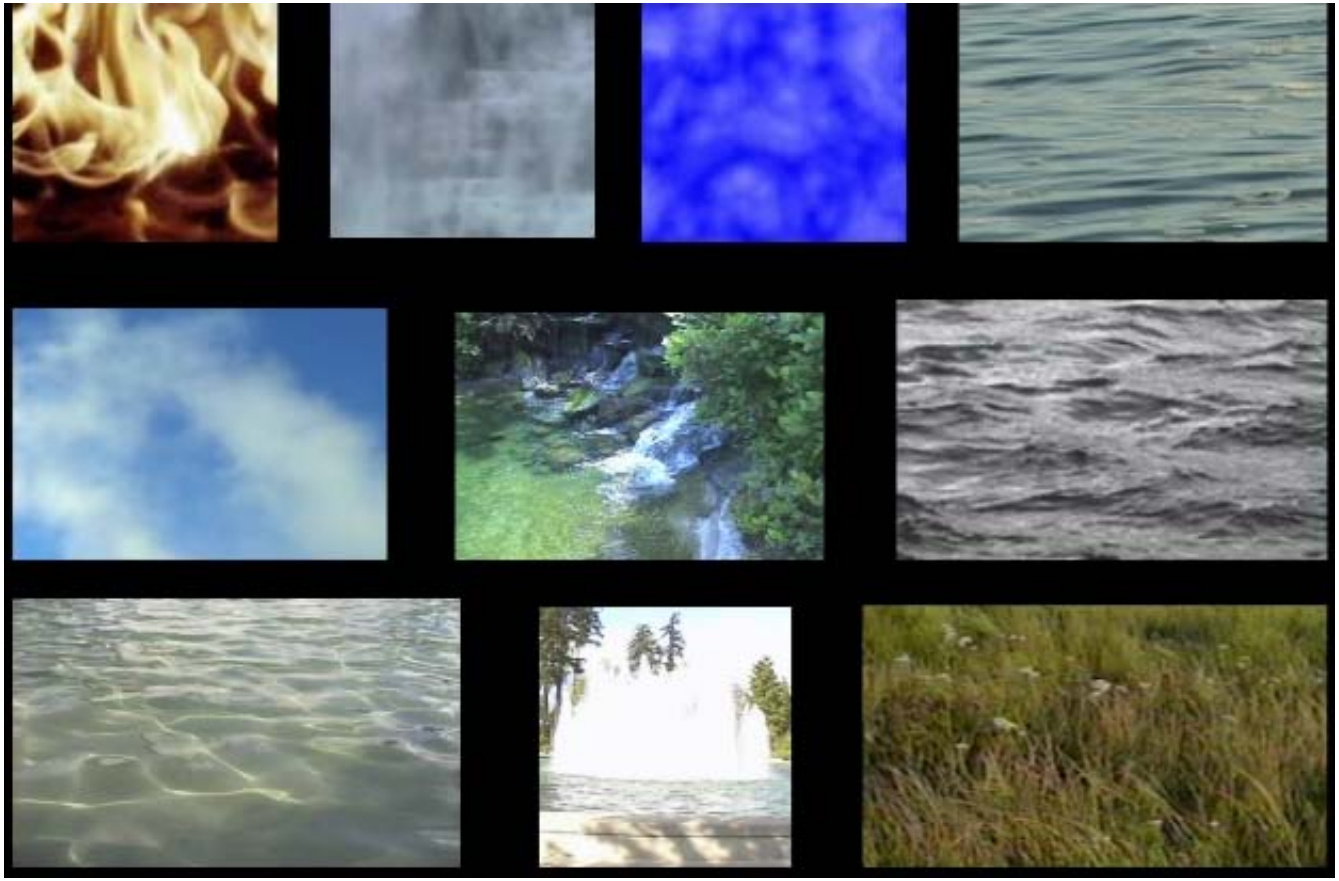
# Texture

- Images containing repeating patterns
  - Locality: small parts look alike
  - Stochastic: they never look exactly the same



# Dynamic Textures

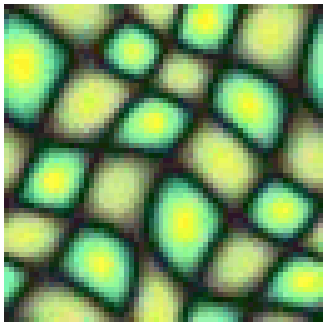
- Appearance similar even as texture evolves



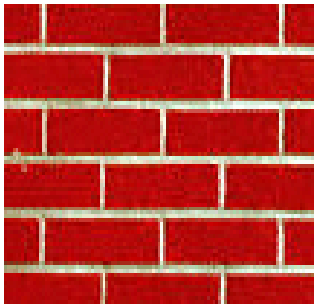
# Classification (Attribution)

**Isotropic**

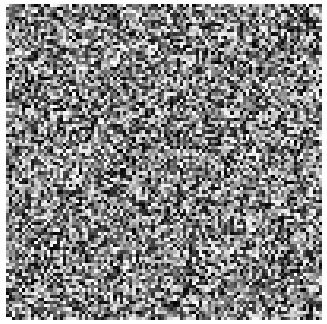
**Anisotropic**



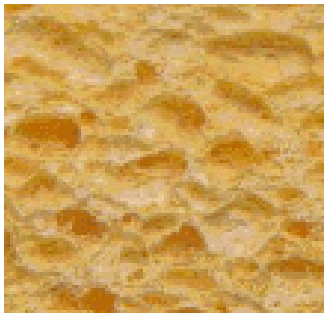
**Both?**



**Repeated**



**Stochastic**

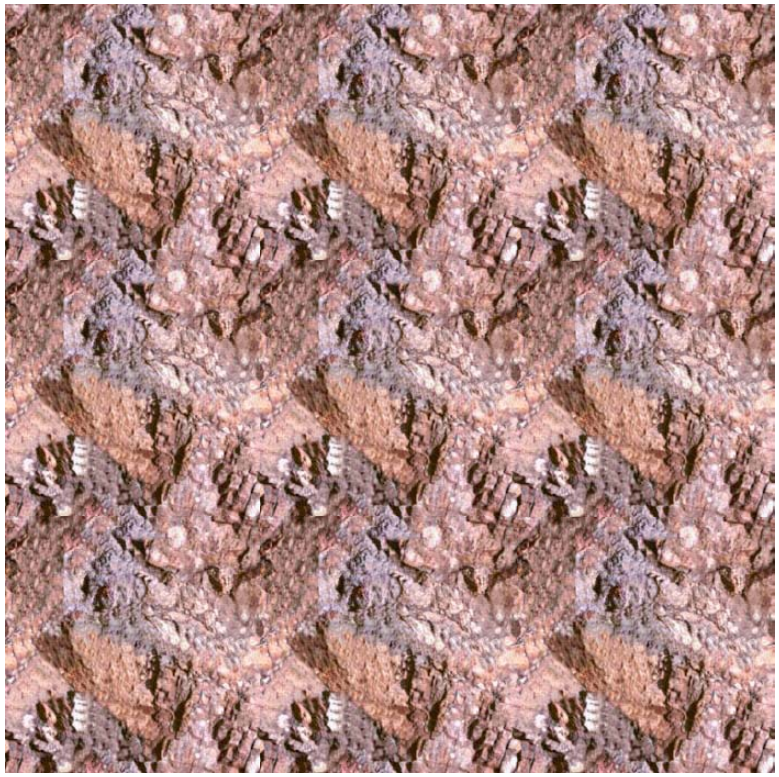


**Both?**

# 2. 2D Texture Synthesis

# Texture Synthesis:
# A Simple Solution … Tiling

small texture

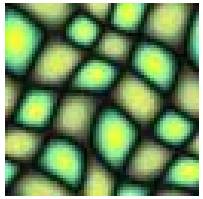


big texture?

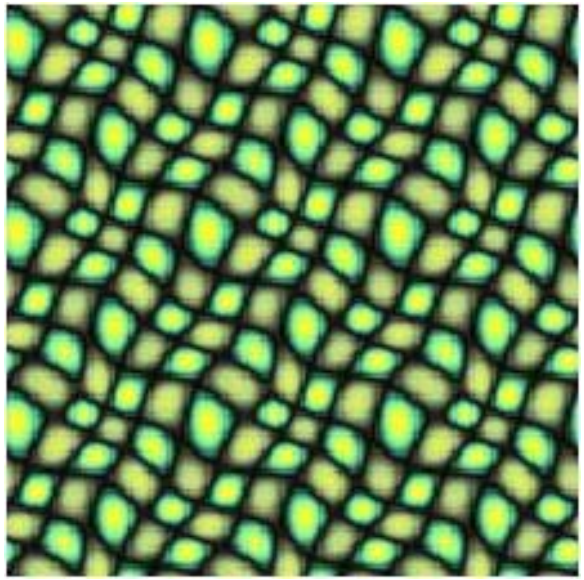
*Tiling is obvious!*

Human eyes are sensitive  to repeated pattern!

# Tilable Texture



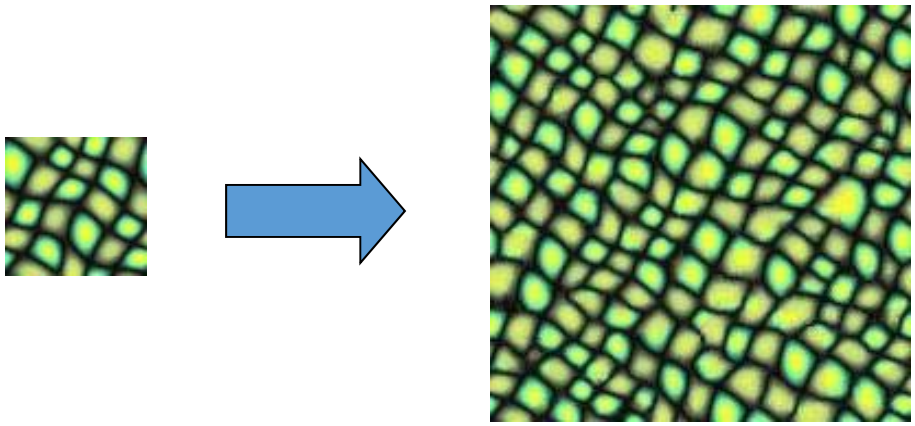
*Tilable texture*



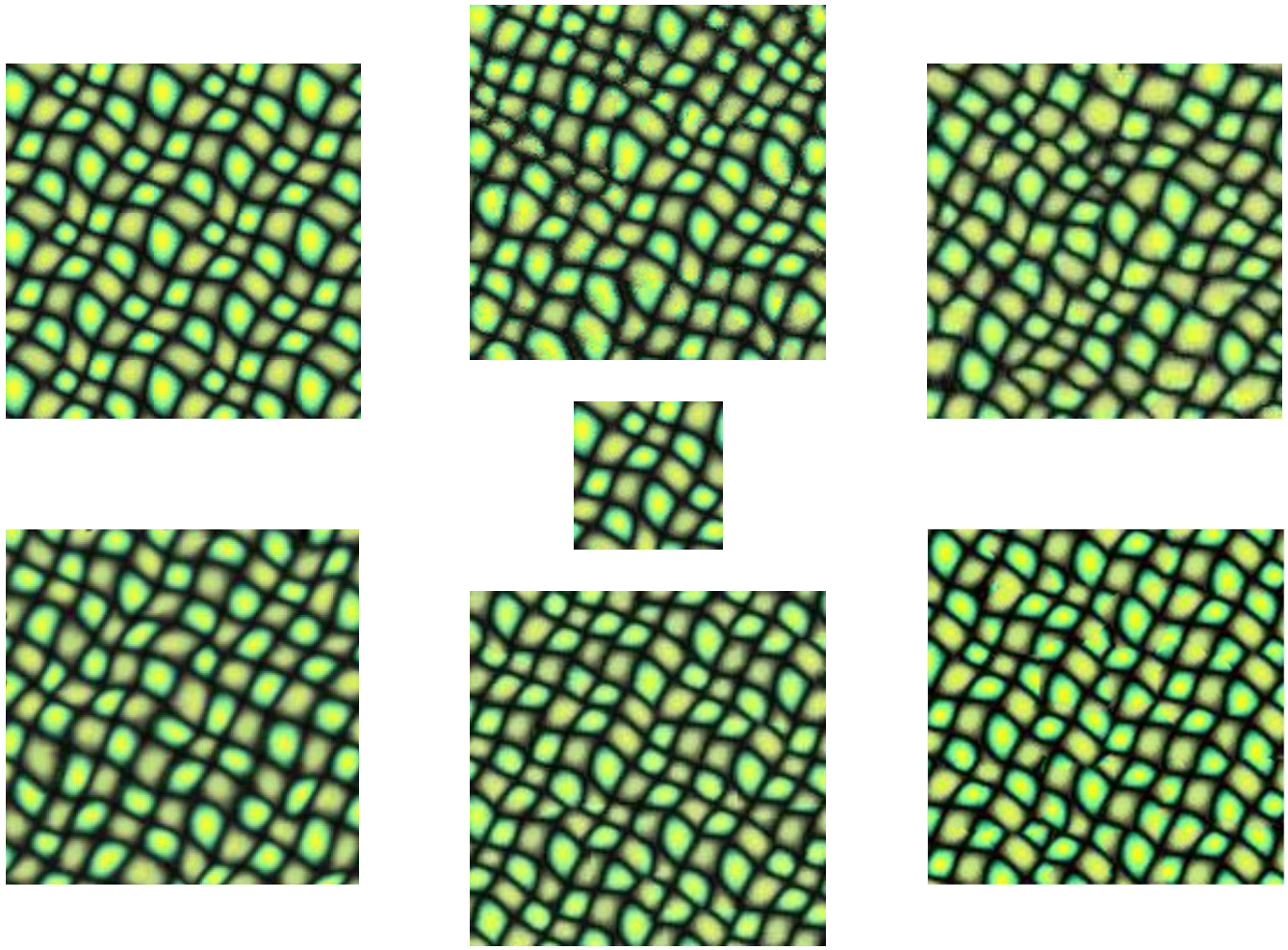
**Simple tiling**

# Desirable Properties

- Result looks like the input
- Efficient
- General
- Easy to use
- Extensible



# Ideal Solution?

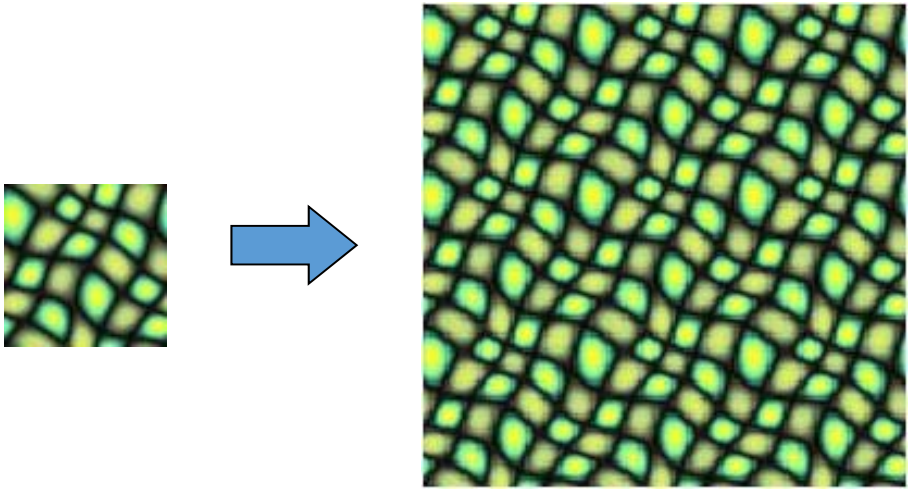


# Challenges

- Texture analysis
  - how to capture the essence of texture?
- Need to model the whole spectrum
  - from repeated to stochastic texture
- This problem is at intersection of vision, graphics, statistics, and image compression

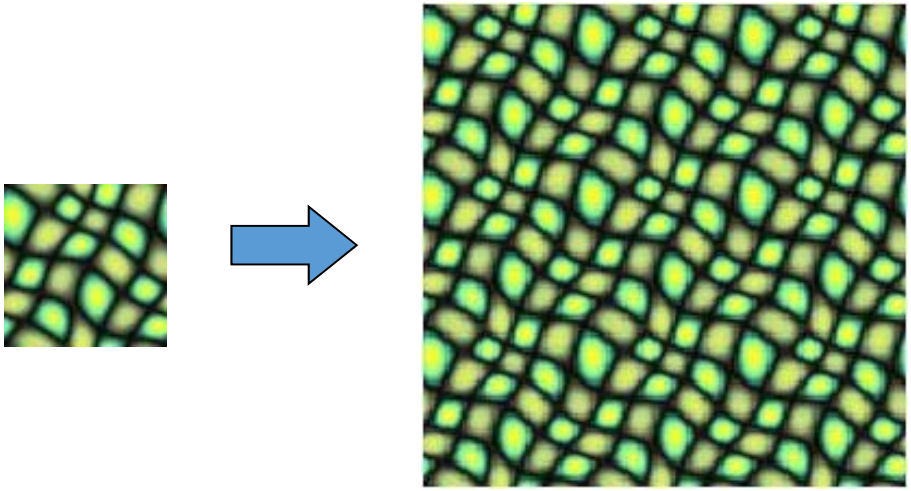
# Example-based Techniques

- Parametric Techniques
  - Compute global statistics in feature space and sample images from texture ensemble directly
- Non-parametric Techniques
  - Estimate local conditional probability density function and synthesize pixels incrementally



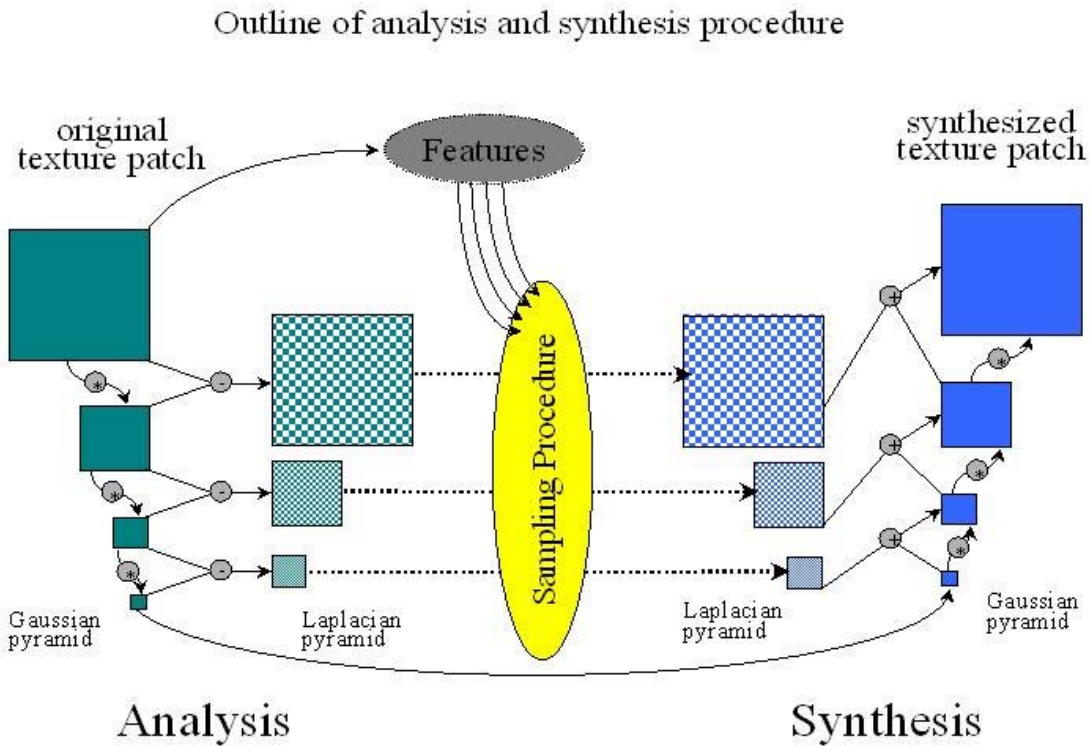
# Example-based Techniques

- Parametric Techniques
  - Compute global statistics in feature space and sample images from texture ensemble directly
- Non-parametric Techniques
  - Estimate local conditional probability density function and synthesize pixels incrementally



# Parametric Techniques

- Hypothesize a mathematical model for texture representation
- Match model parameters of input and output texture

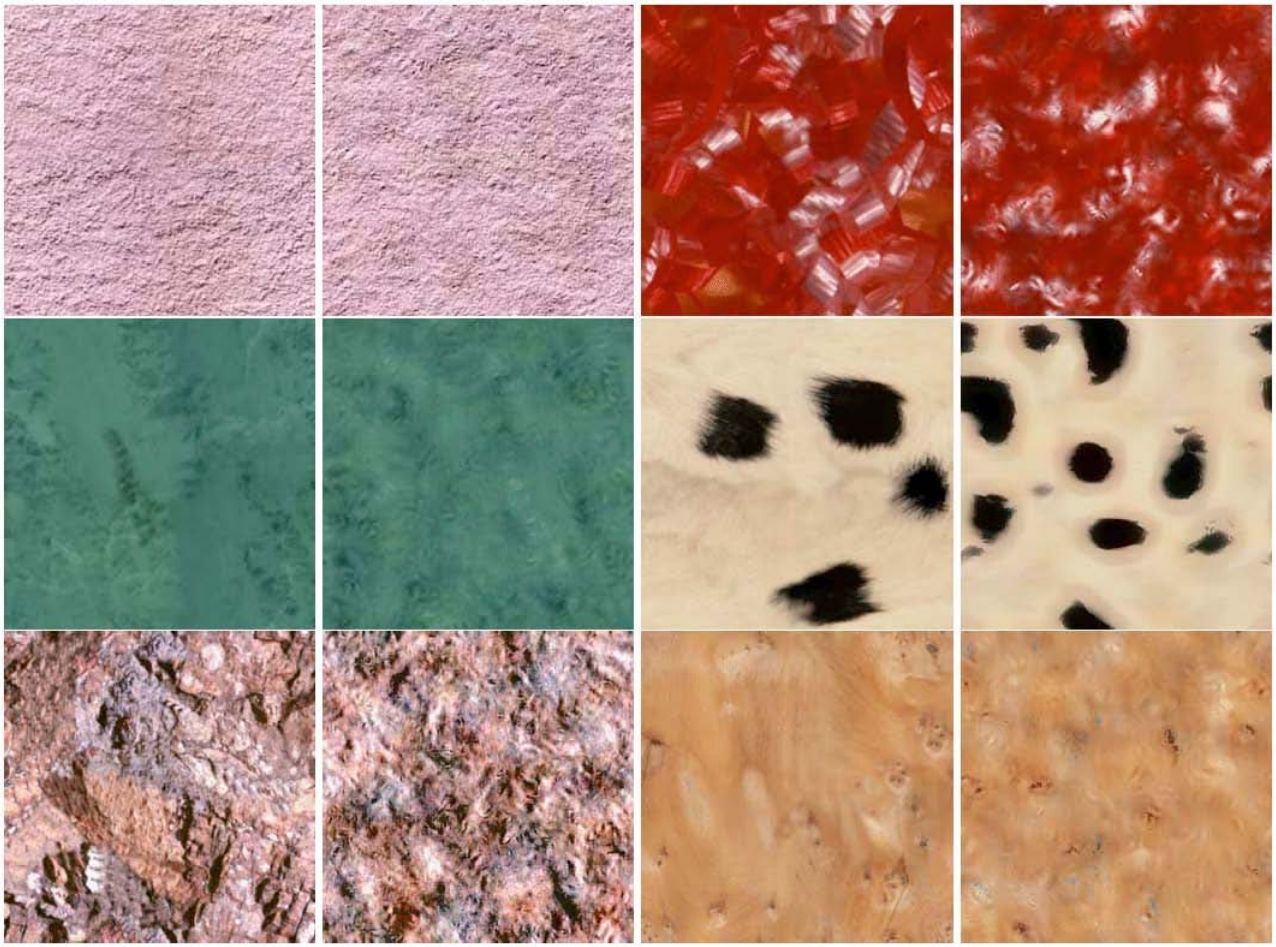


# Pyramid-Based Texture Analysis/Synthesis

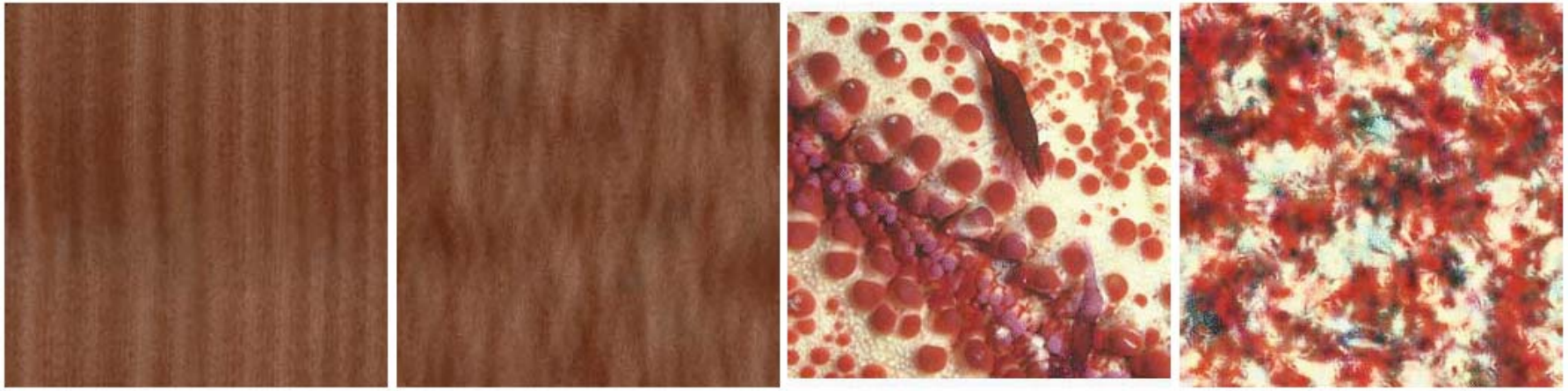
[Heeger & Bergen, Siggraph 1995]

- Initialize J to noise
- Create multiresolution pyramids for I and J
- Match the histograms of J's pyramid levels with I's pyramid levels
- Loop until convergence
- Can be generalized to 3D

# Results

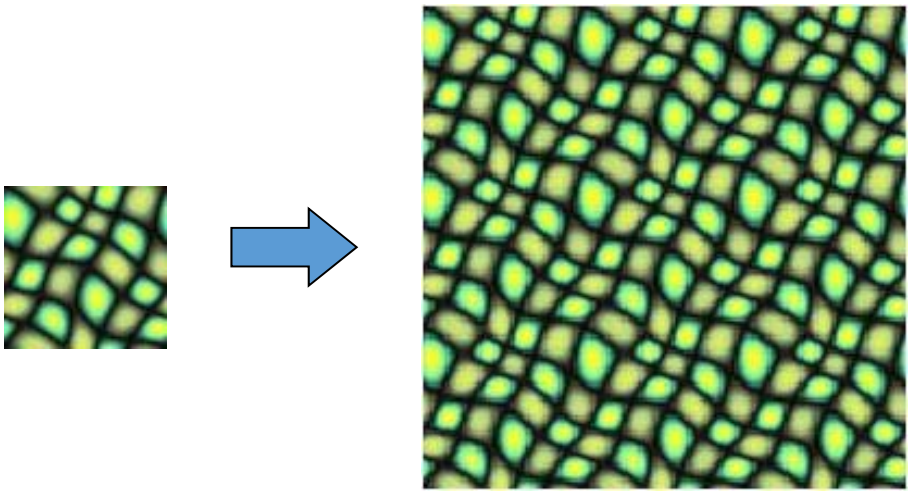


# Failures



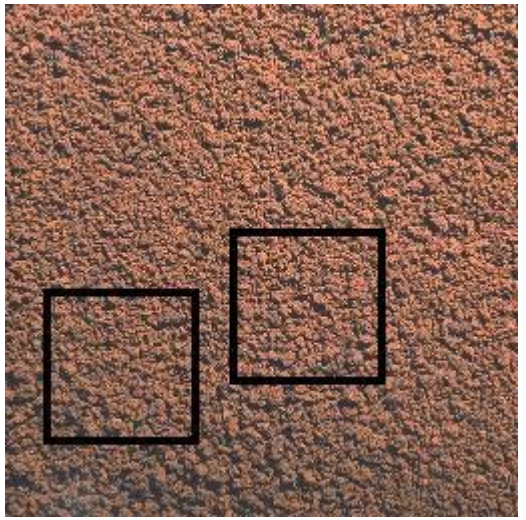
# Example-based Techniques

- Parametric Techniques
  - Compute global statistics in feature space and sample images from texture ensemble directly
- Non-parametric Techniques
  - Estimate local conditional probability density function and synthesize pixels incrementally



# Non-Parametric Techniques

- Synthesis by copying from the input
- Markov-Random Field Model
  - Pixel appearance depends only on neighborhood

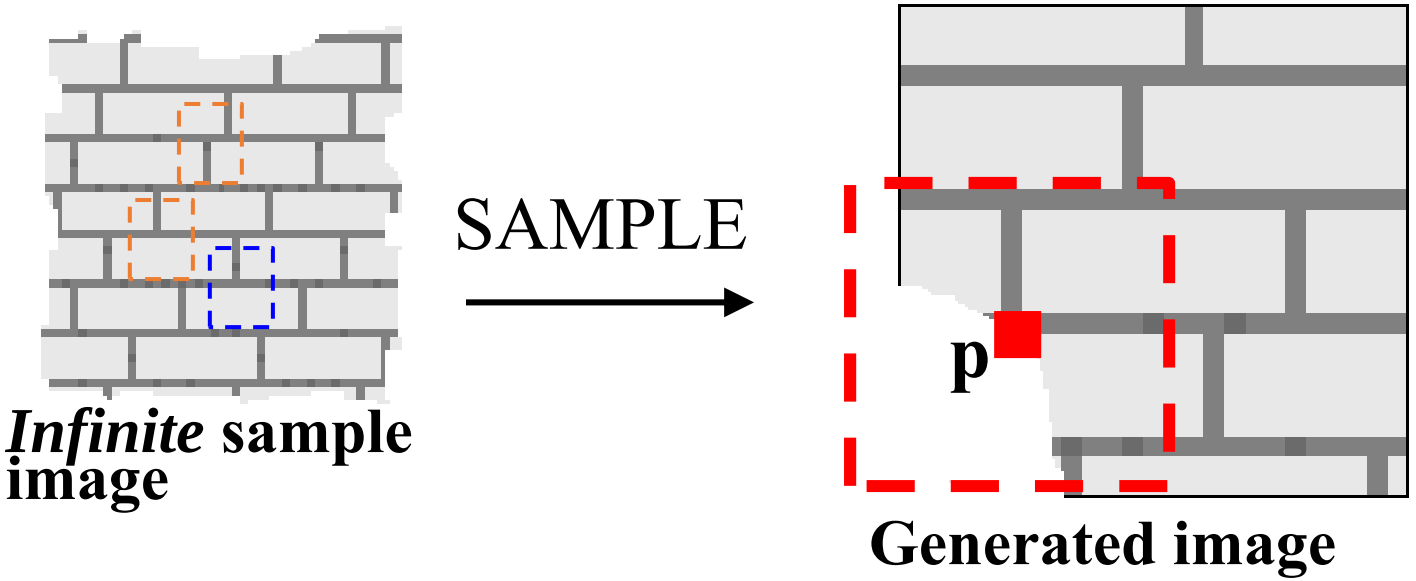


# Markov Random Field

- [Shannon,'48] proposed a way to generate English-looking text using N-grams:
  - Assume a generalized Markov model
  - Use a large text to compute prob. distributions of each letter given N-1 previous letters
  - Starting from a seed repeatedly sample this Markov chain to generate new letters
  - Also works for whole words

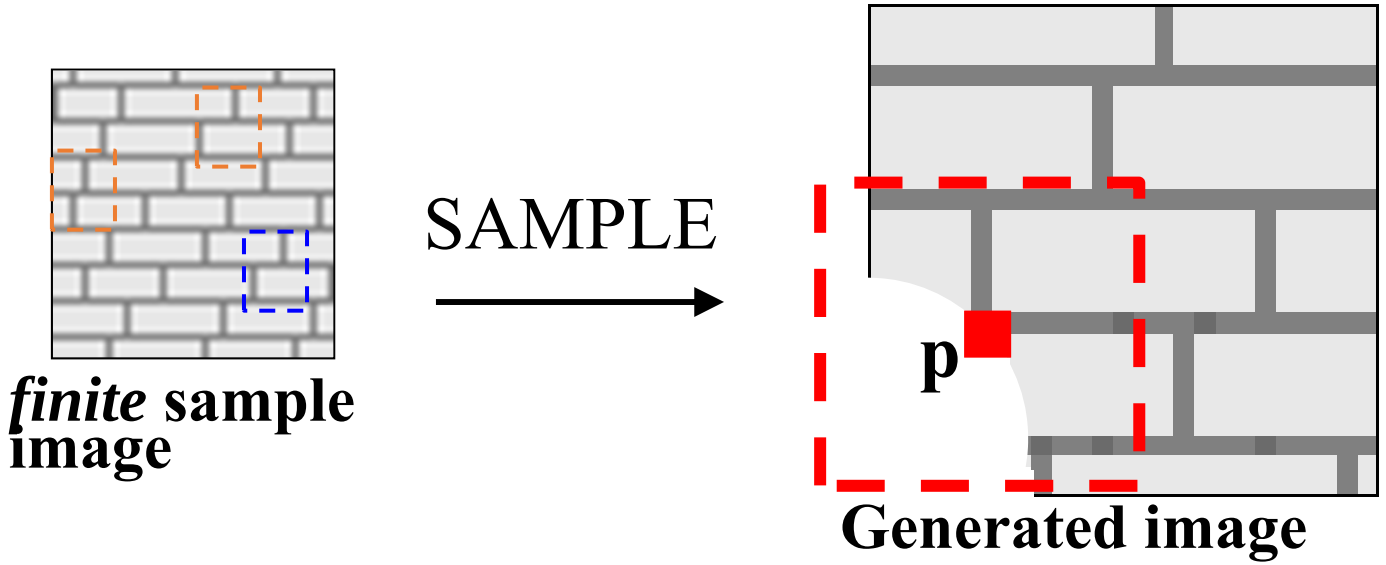
**WE NEED TO EAT CAKE**

# Synthesizing One Pixel




- Assuming Markov property, what is conditional probability distribution of p, given the neighbourhood window?
- Instead of constructing a model, let's directly search the input image for all such neighbourhoods to produce a histogram for p
- To synthesize p, just pick one match at random

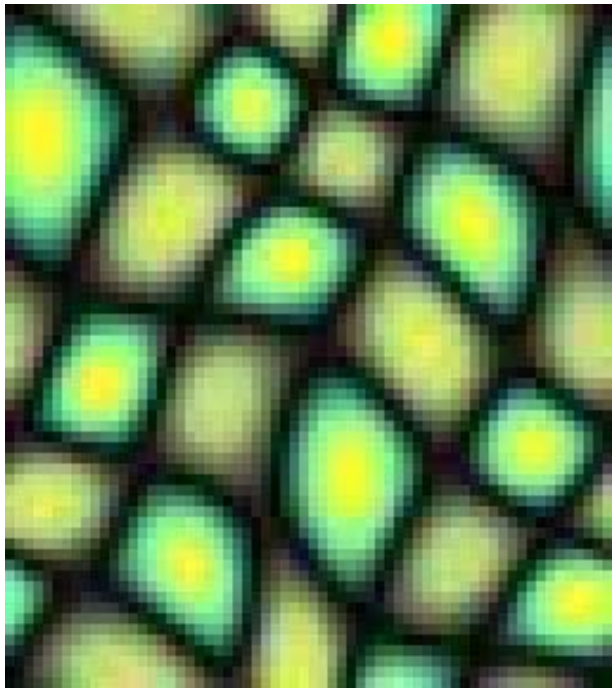
# Really Synthesizing One Pixel



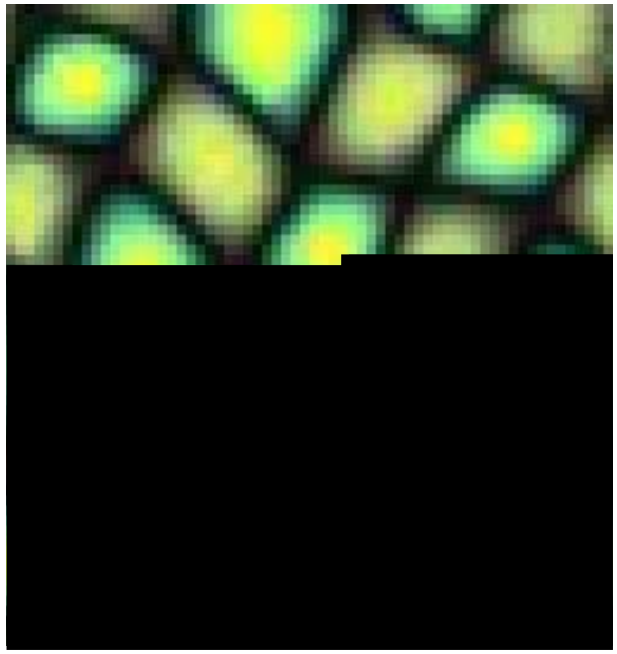

- However, since our sample image is finite, an exact neighbourhood match might not be present
- So we find the **best** match using SSD error (weighted by a Gaussian to emphasize local structure), and take all samples within some distance from that match

# Pixel-Based Texture Synthesis

[Efros and Leung, ICCV 1999]



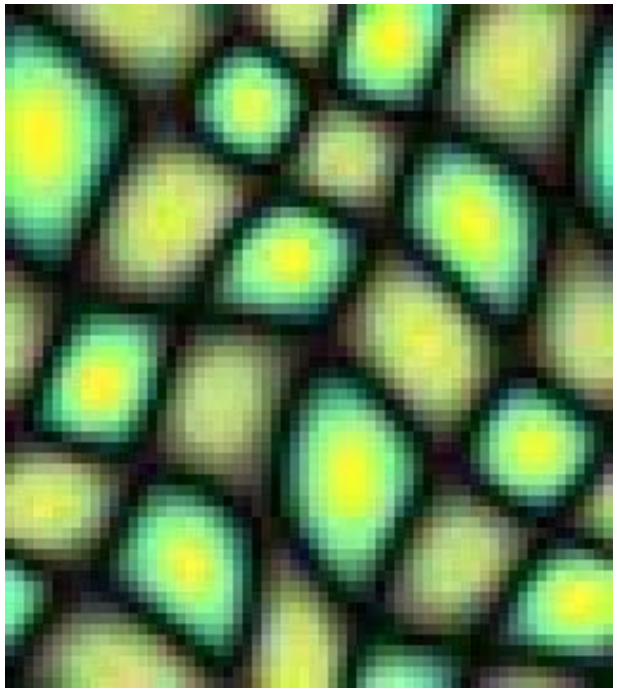
**Input**

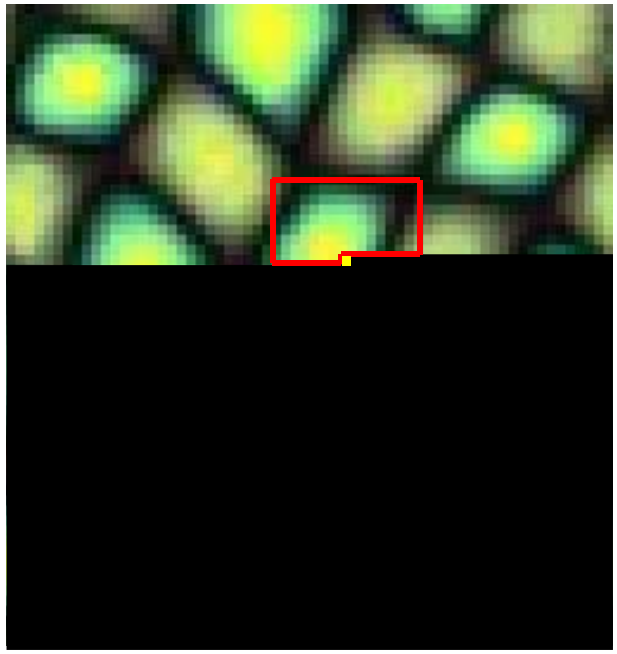


**Output**

**Step n in Algorithm**

# Pixel-Based Texture Synthesis

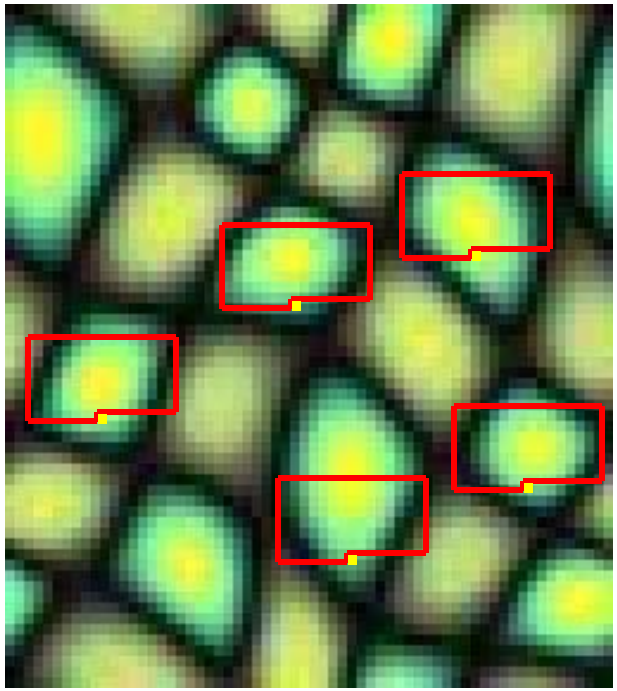


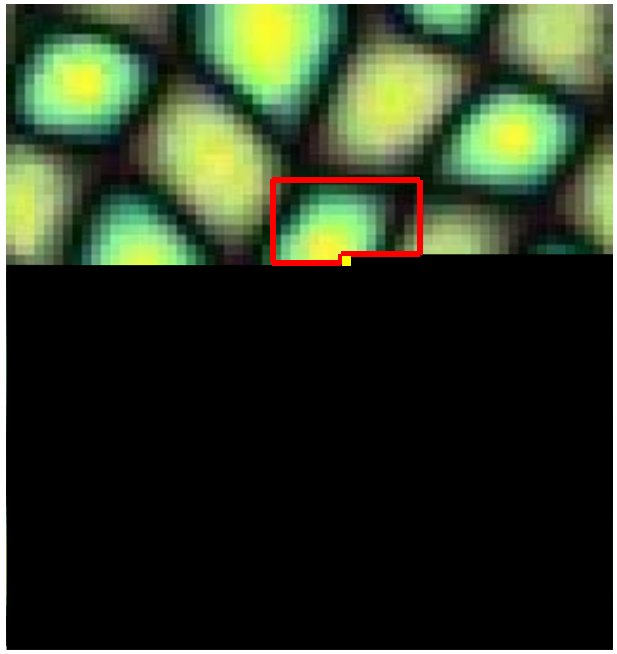


**Input** **Output**

**Build L-shaped Neighborhood**

# Pixel-Based Texture Synthesis

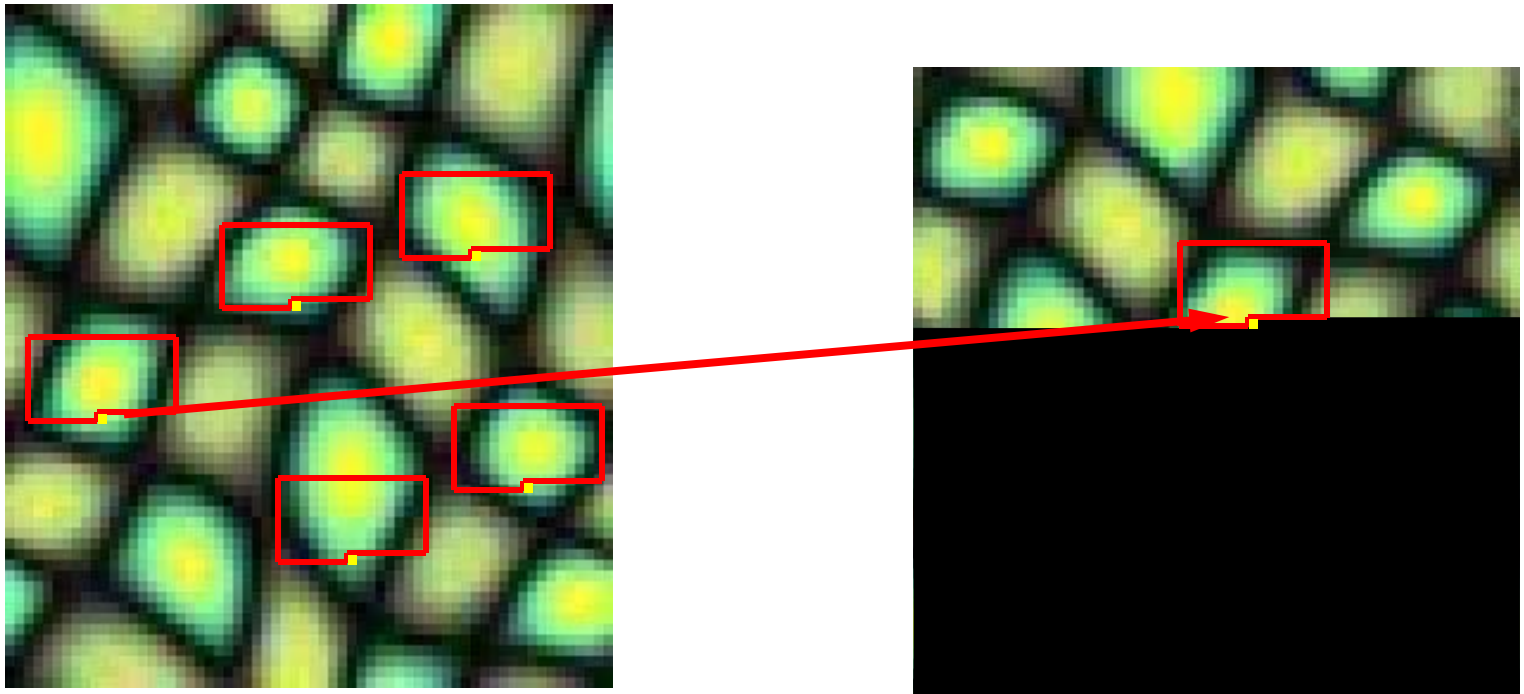




**Input** **Output**

**Find Matching Neighborhood(s) in Input**

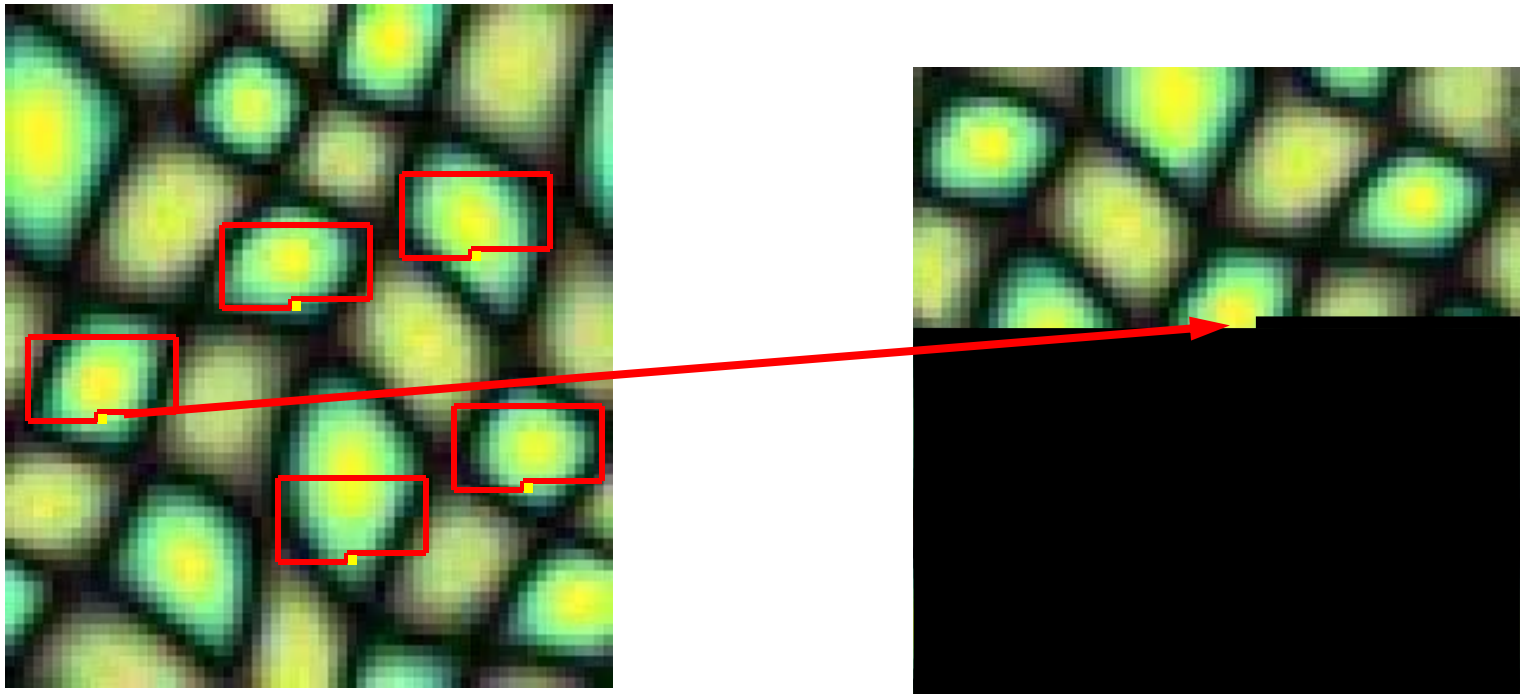
# Pixel-Based Texture Synthesis



**Input** **Output**

**Select *Best* Neighborhood from all Candidates**

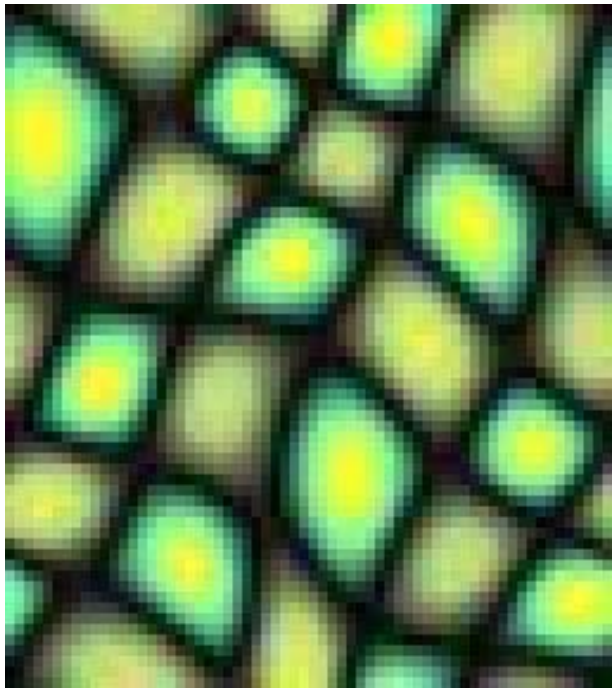
# Pixel-Based Texture Synthesis



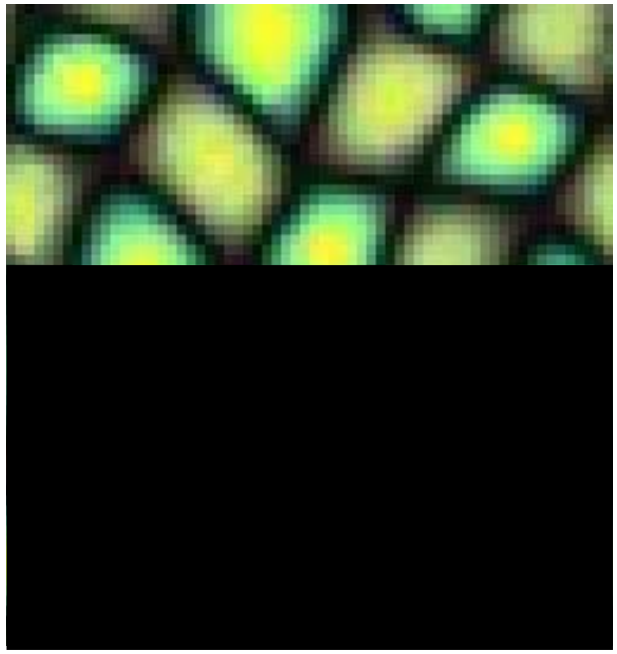
**Input** **Output**

**Copy Selected Pixel to Output**

# Pixel-Based Texture Synthesis



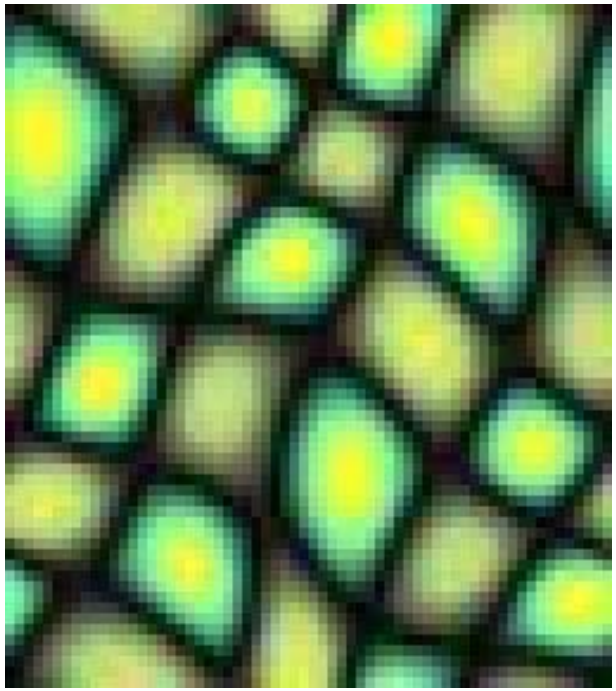
**Input**



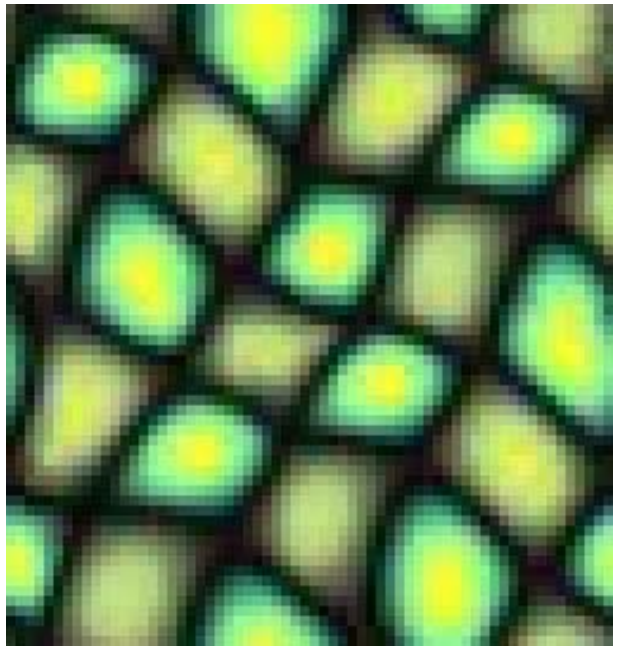
**Output**

**Repeat for all Pixels in this Row...**

# Pixel-Based Texture Synthesis



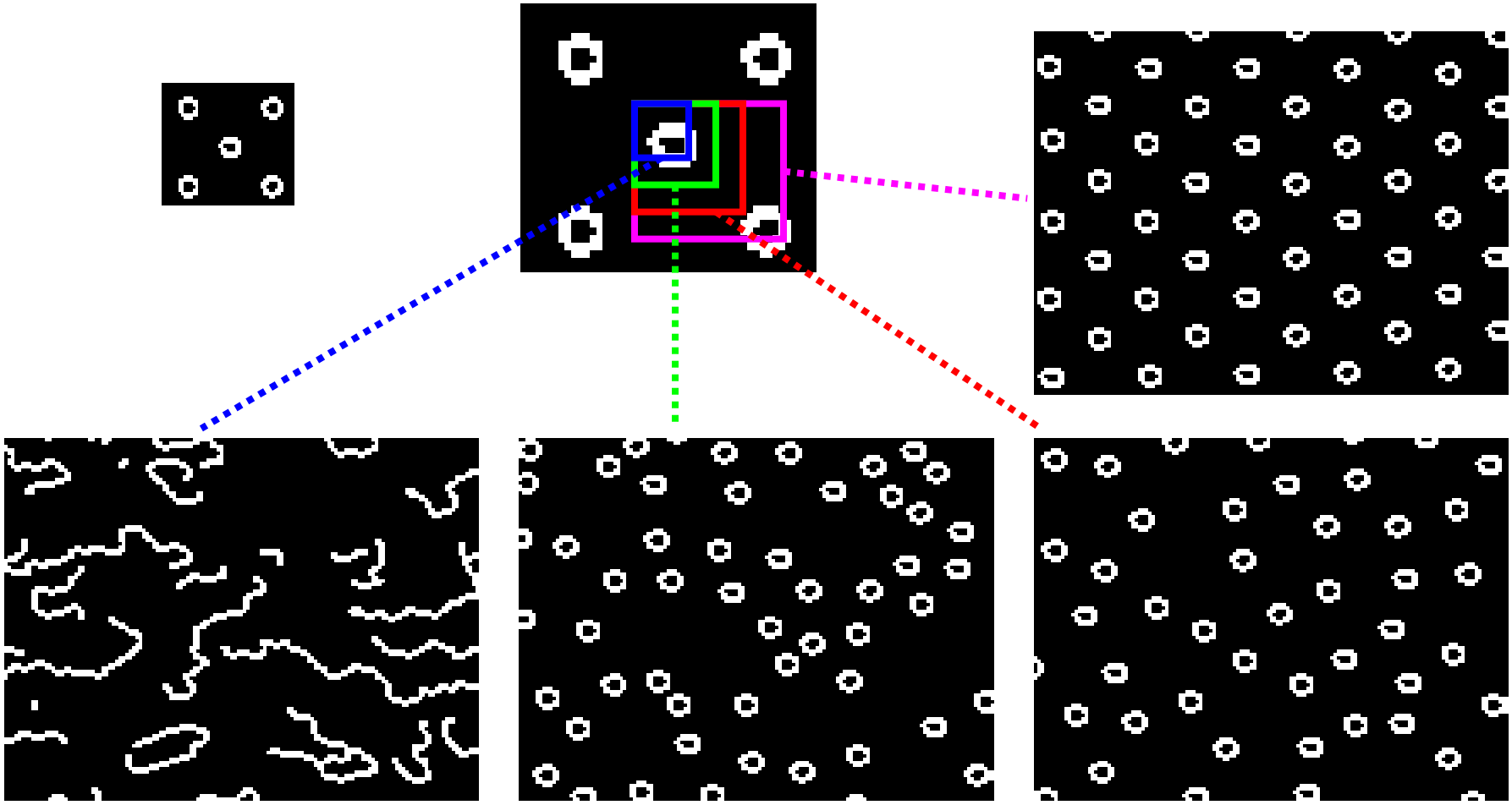
**Input**



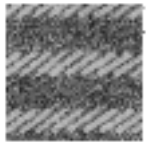
**Output**

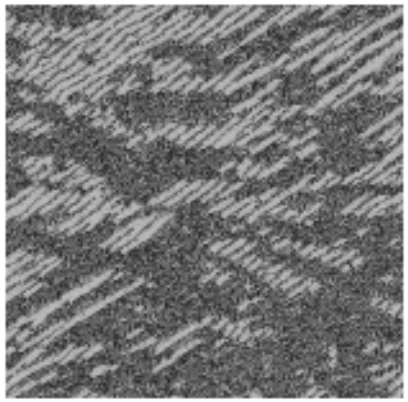
**...and all other Output Rows**

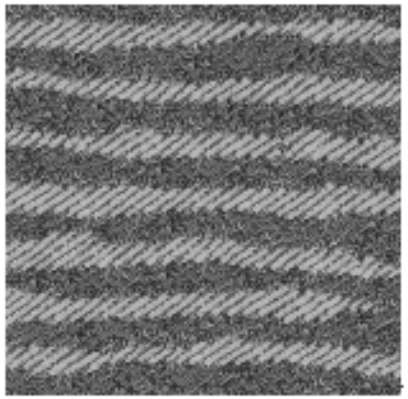
# Randomness Parameter

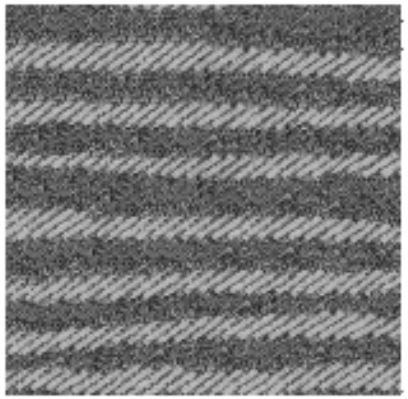


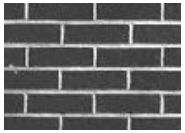
# Results

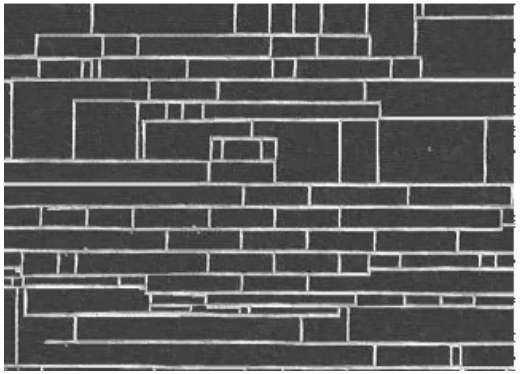


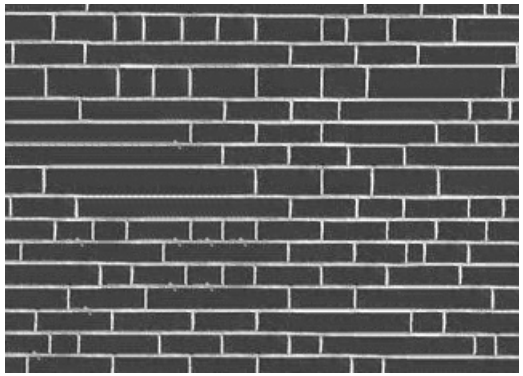


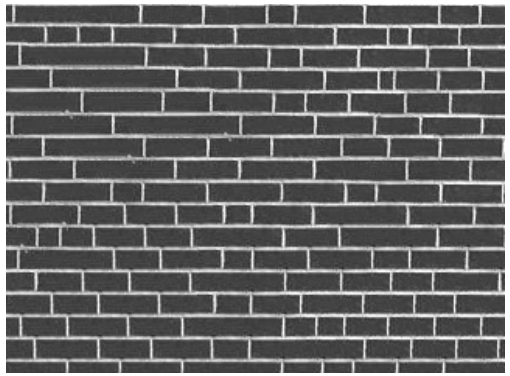












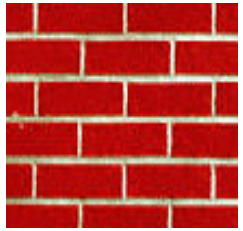
Increasing window size

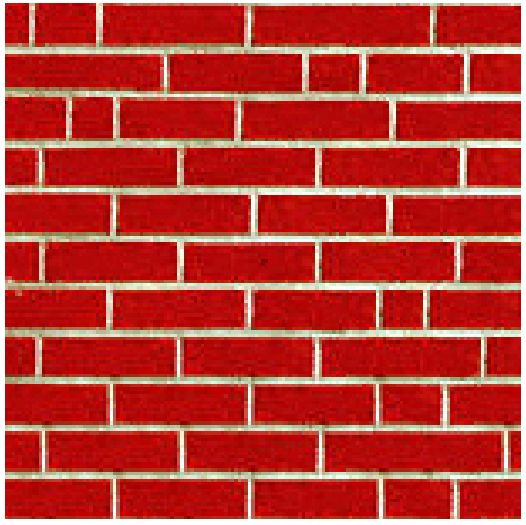
# More Results

wood

granite

brick wall

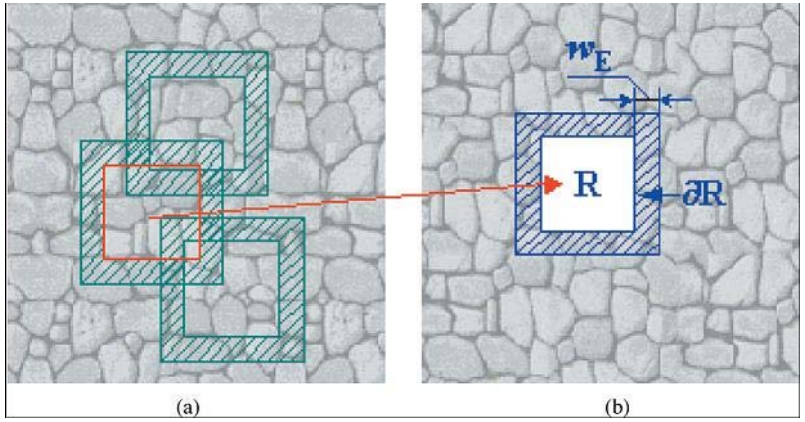


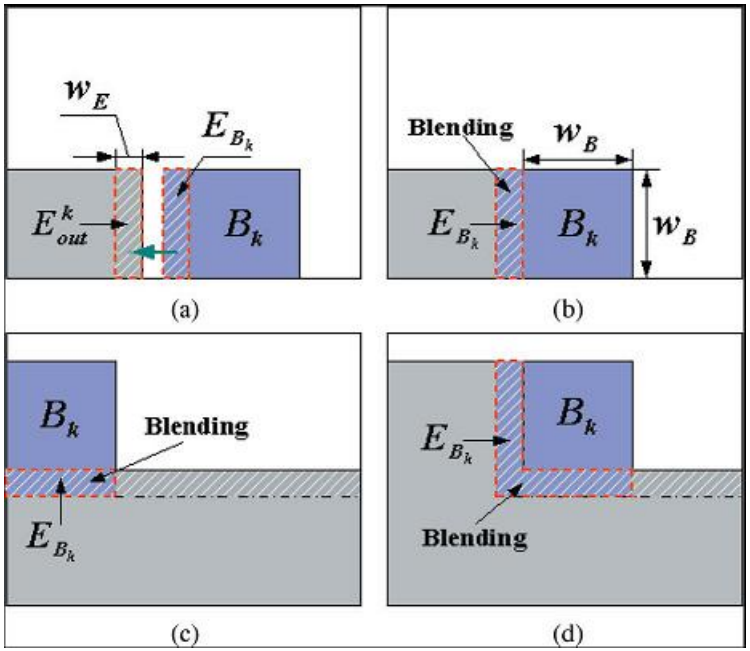


# Patch-based Synthesis

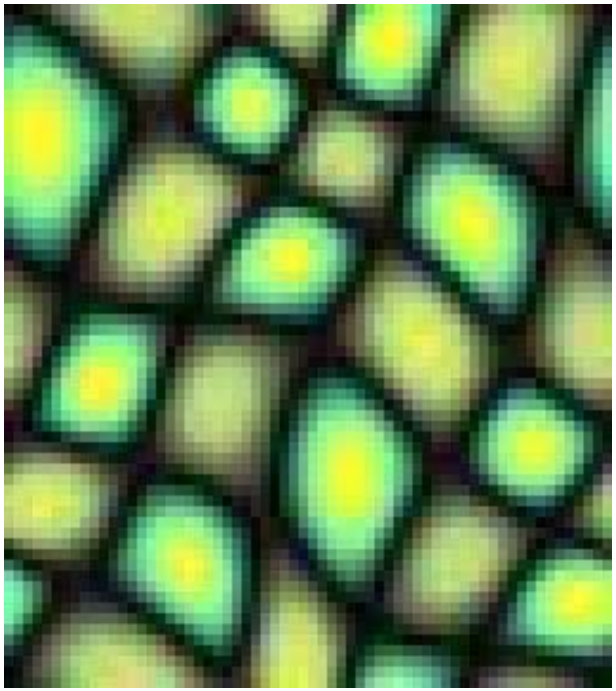
[Liang et al. TOG 2002]

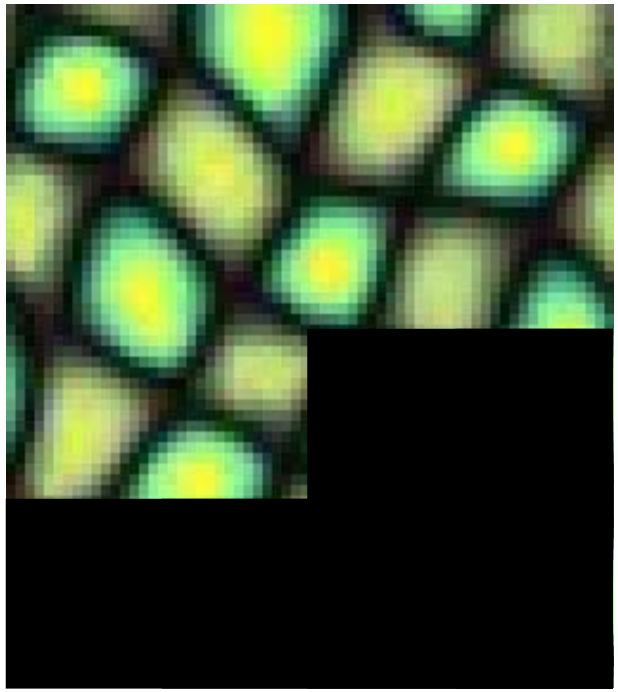
- Copy patches instead of single pixels



# Patch-Based Texture Synthesis

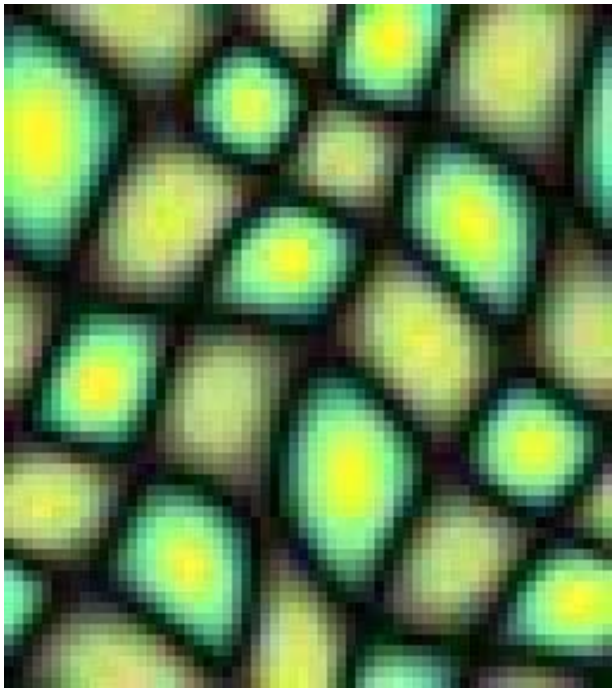


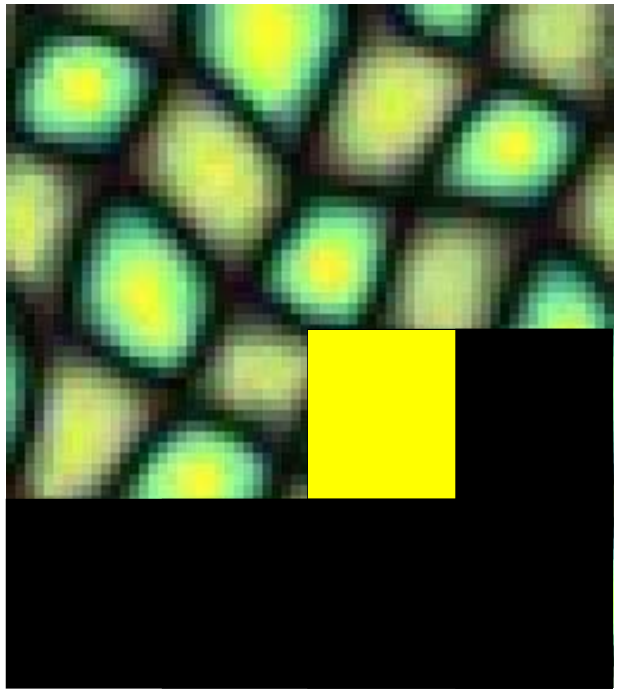


**Input** **Output**

**Step n in Algorithm**

# Patch-Based Texture Synthesis

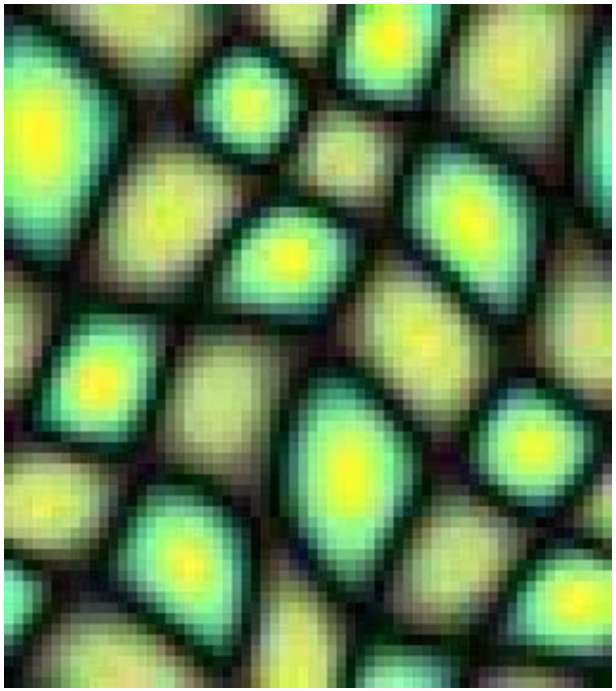


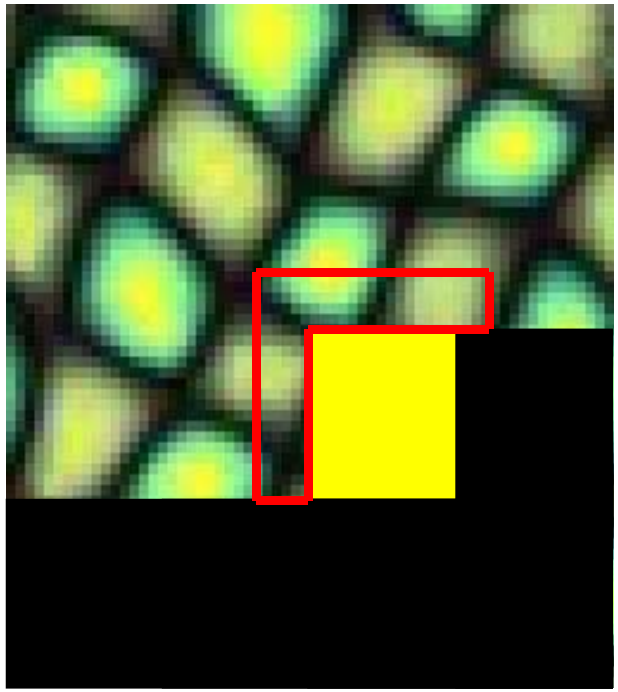


**Input** **Output**

**Patch to be Synthesized in this Step**

# Patch-Based Texture Synthesis

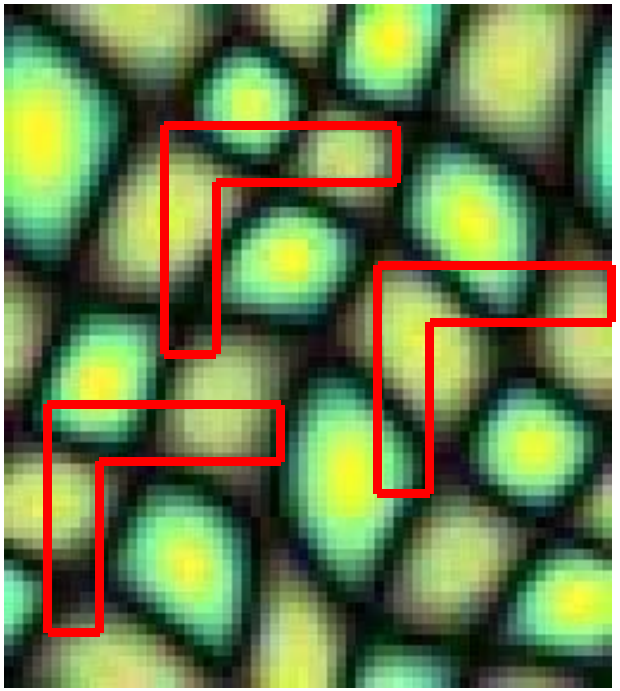




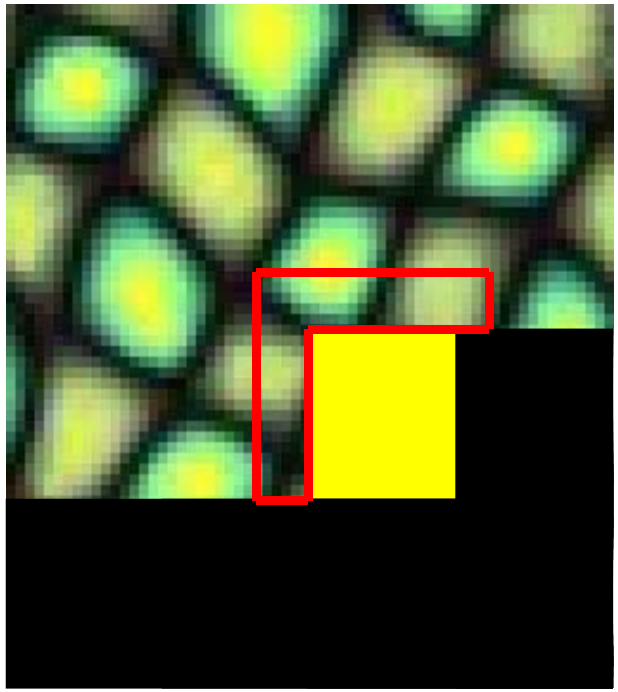
**Input** **Output**

**Build Neighborhood (Image Mask)**

# Patch-Based Texture Synthesis



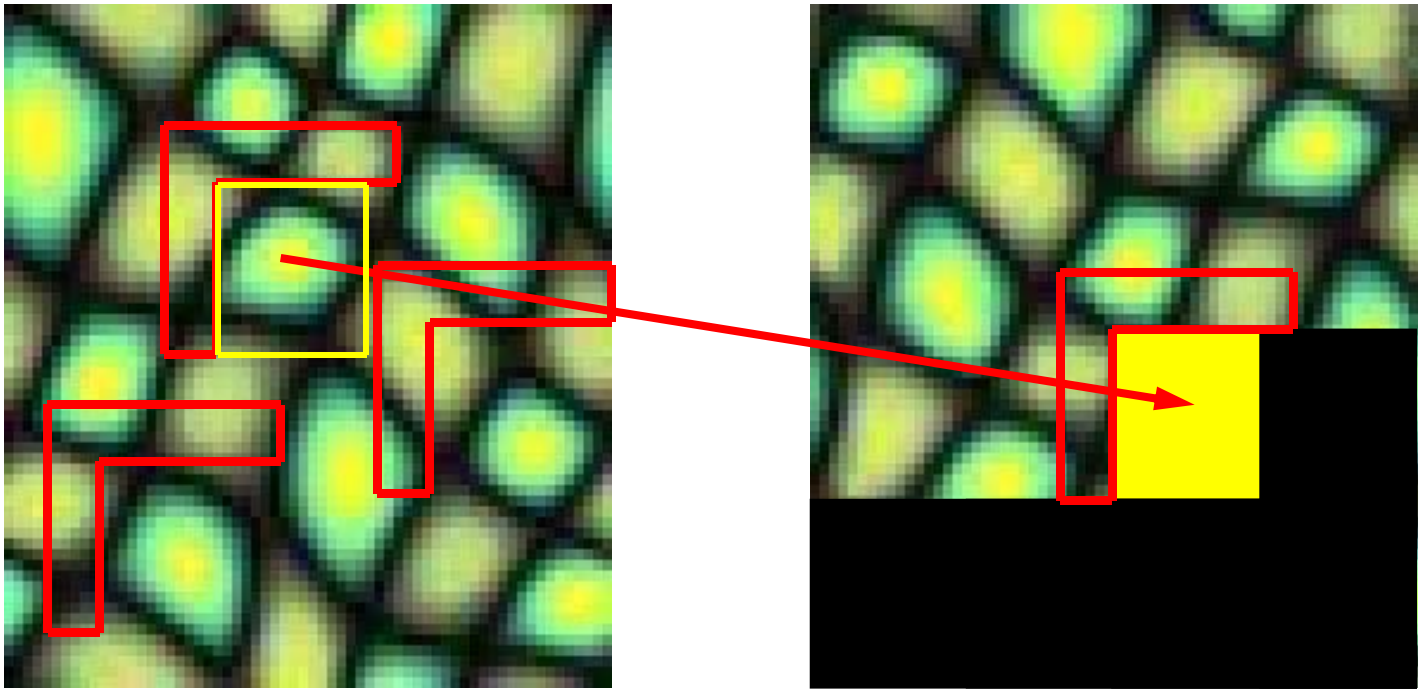
**Input**



**Output**

**Find Matching Neighborhood(s) in Input**

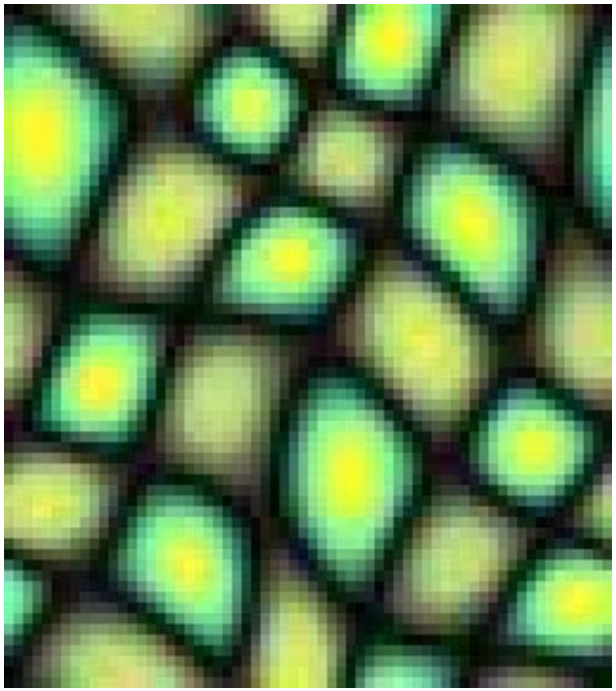
# Patch-Based Texture Synthesis

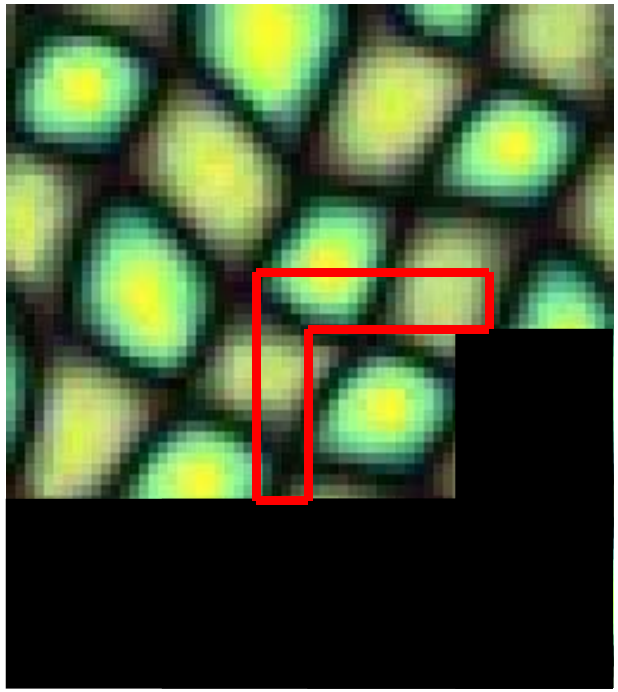


**Input** **Output**

**Select *Best* Neighborhood from all Candidates**

# Patch-Based Texture Synthesis

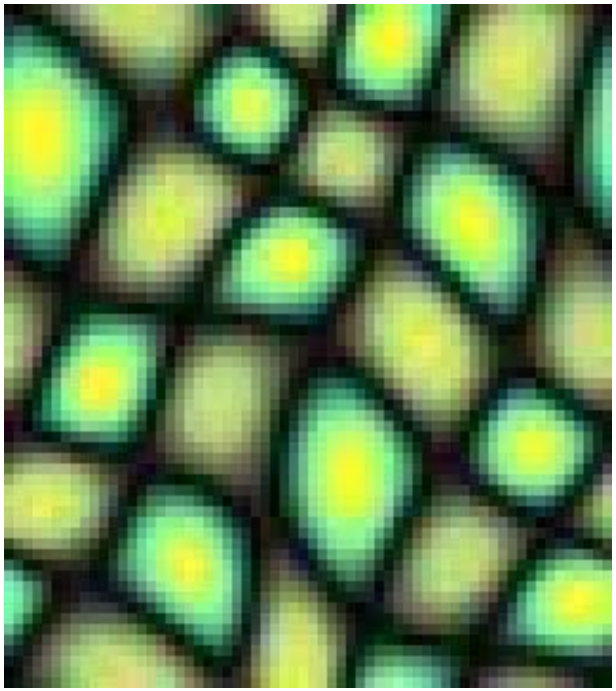




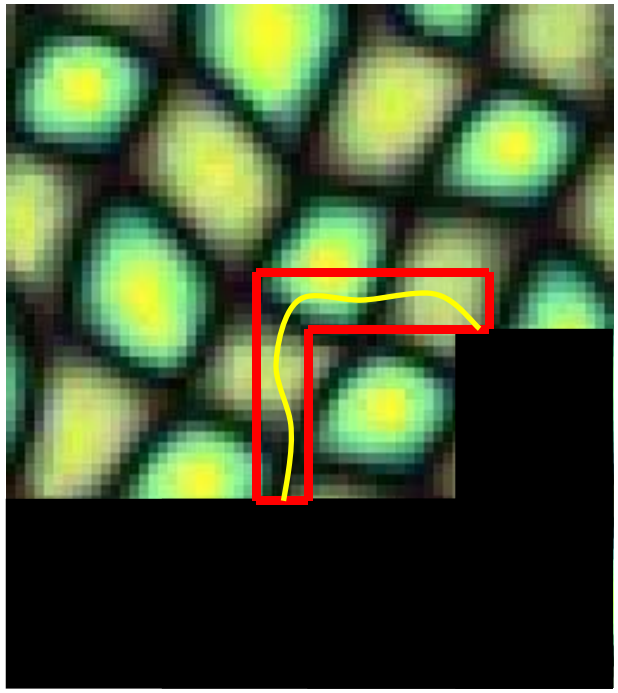
**Input** **Output**

**Copy Selected Patch to Output**

# Patch-Based Texture Synthesis



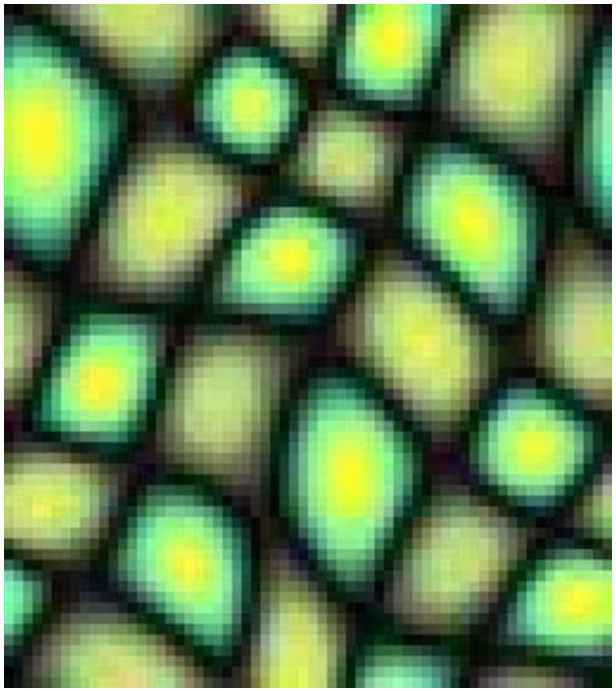
**Input**



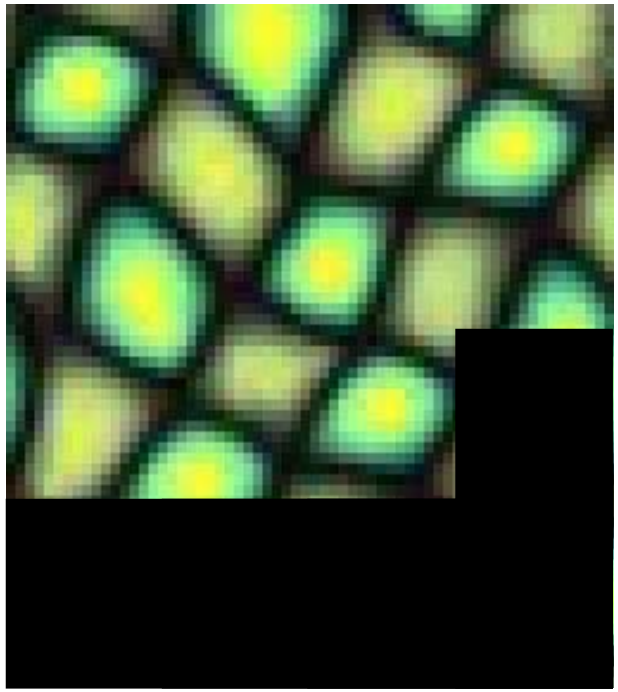
**Output**

**Apply some *Overlap Repair Strategy ...***

# Patch-Based Texture Synthesis



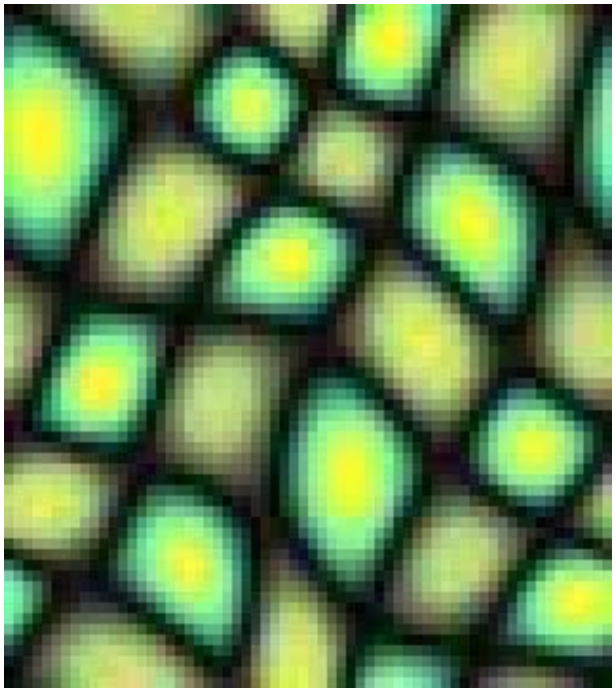
**Input**

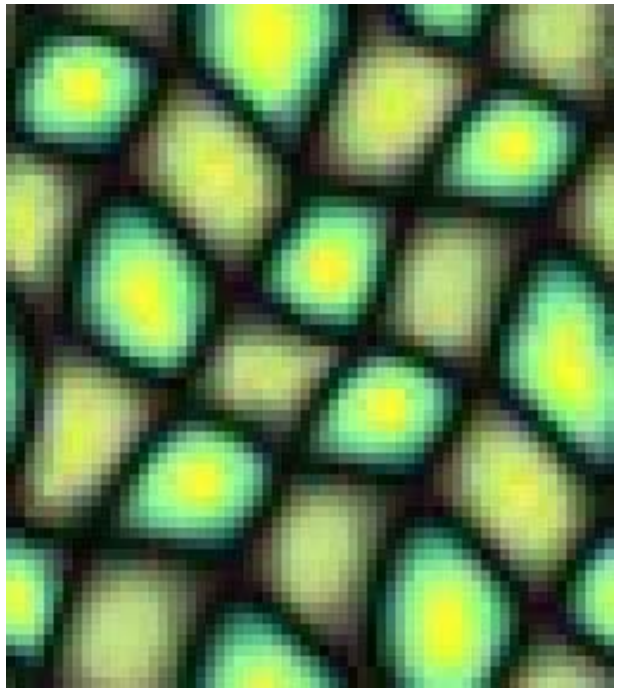


**Output**

**... And we are done with this patch**

# Patch-Based Texture Synthesis





**Input** **Output**

**Repeat for all Patches in Output**

# Synthesis Result

— High-quality Texture

Input sample

Efros and Leung

Wei and levoy

Patch-based Sampling

Input sample

Efros and Leung

Wei and levoy

Patch-based Sampling

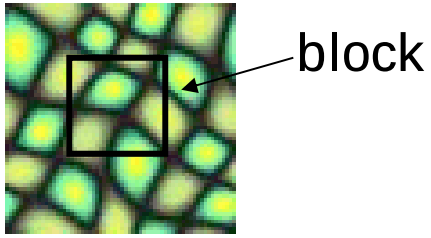
- Efros' algorithm has a tendency to grow garbage and Wei's TSVQ acceleration further aggravates this problem. In Contrast, patch-based sampling avoids growing garbage

# Advantages

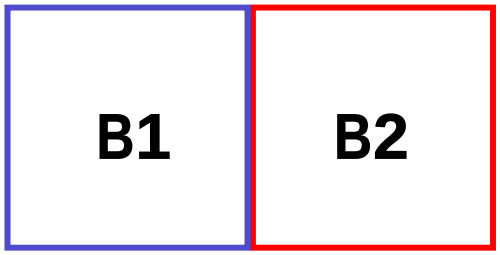
- Speed
  - Orders of magnitude faster than existing texture synthesis algorithm
  - Real-time synthesis
- Quality
  - Synthesize high-quality textures ranging from regular to stochastic
  - Avoid growing garbage
  - Synthesize subjectively better natural textures

# Mincut: Graph-cut based

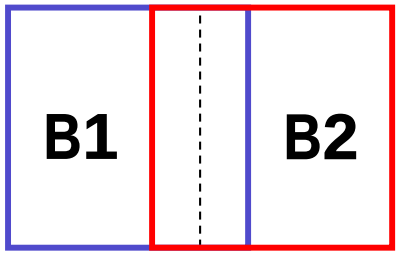
[Efros&Freeman, Siggraph 2002]



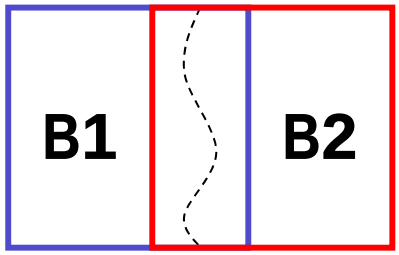

**Input texture**



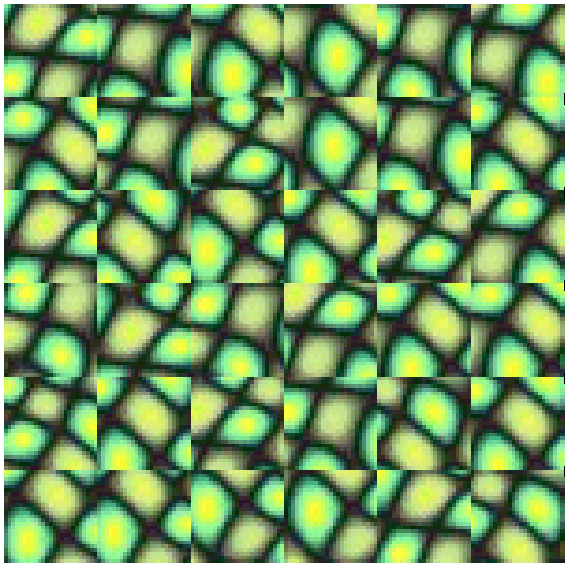

**Random placement of blocks**

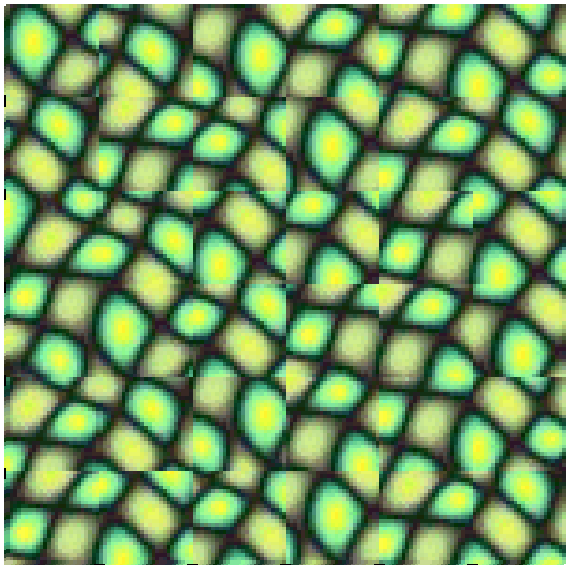


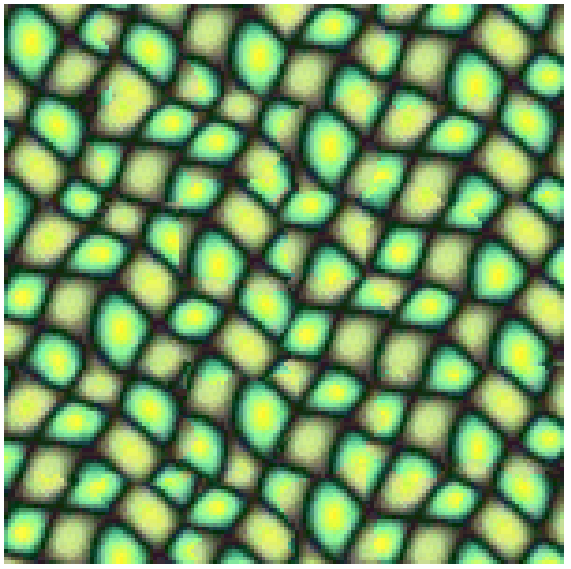

**Neighboring blocks constrained by overlap**



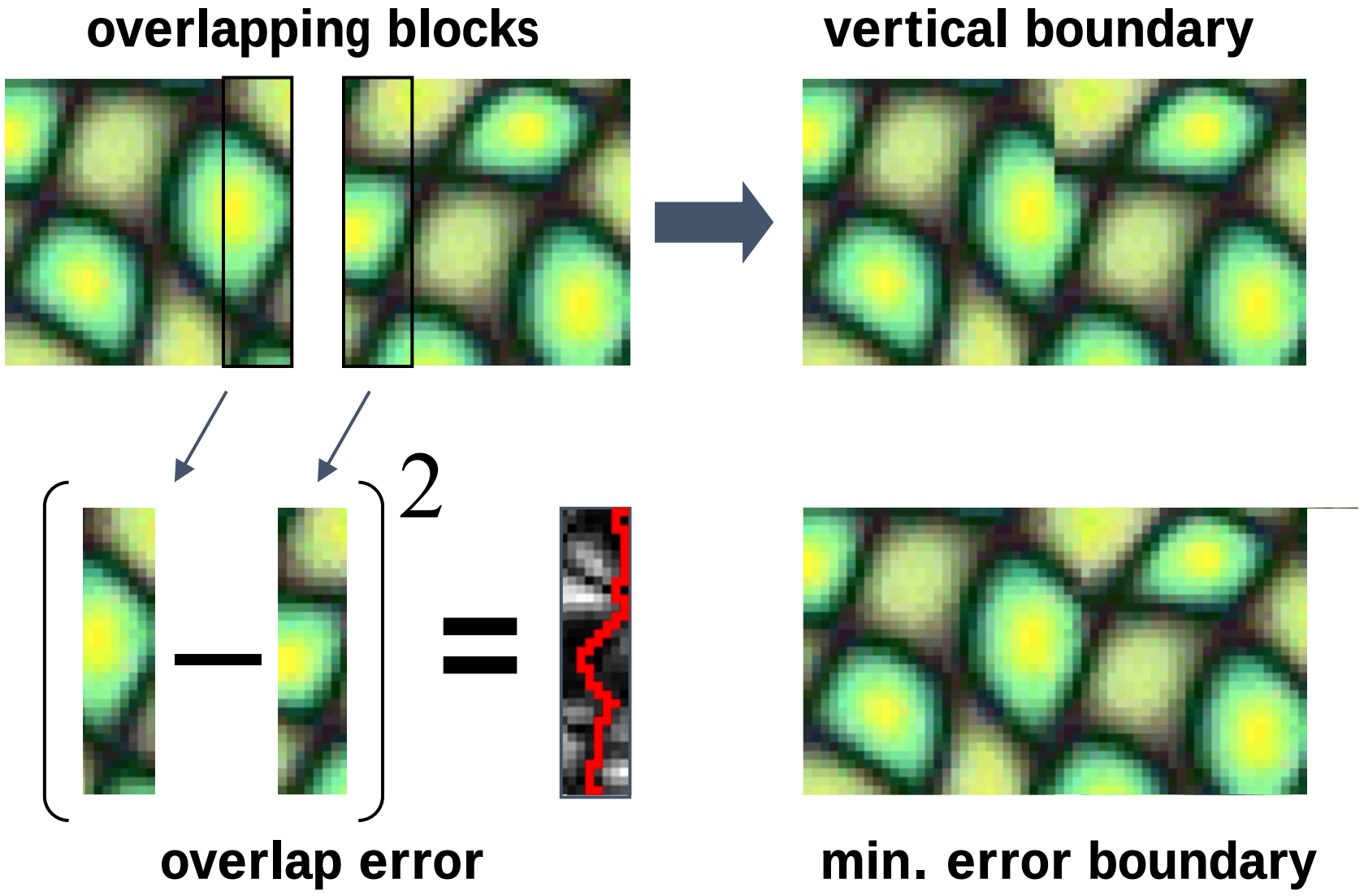

**Minimal error boundary cut**



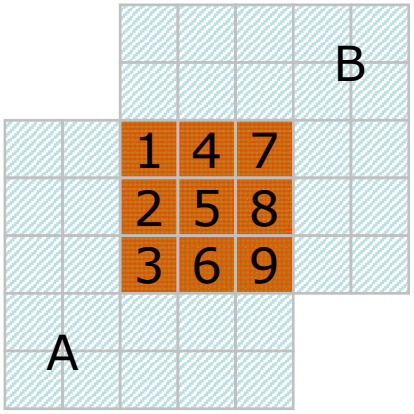




# Minimal error boundary



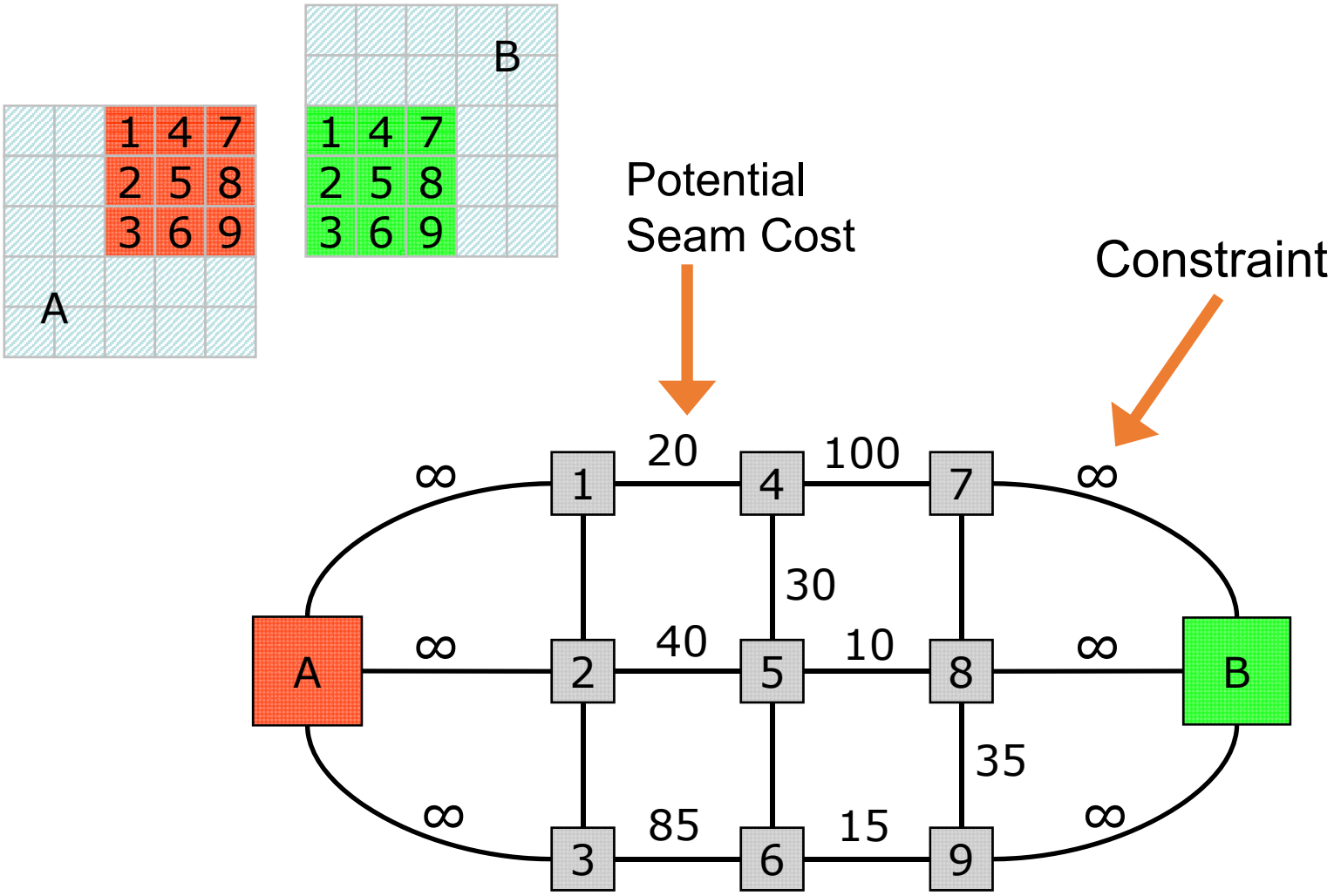
# Seam Optimization



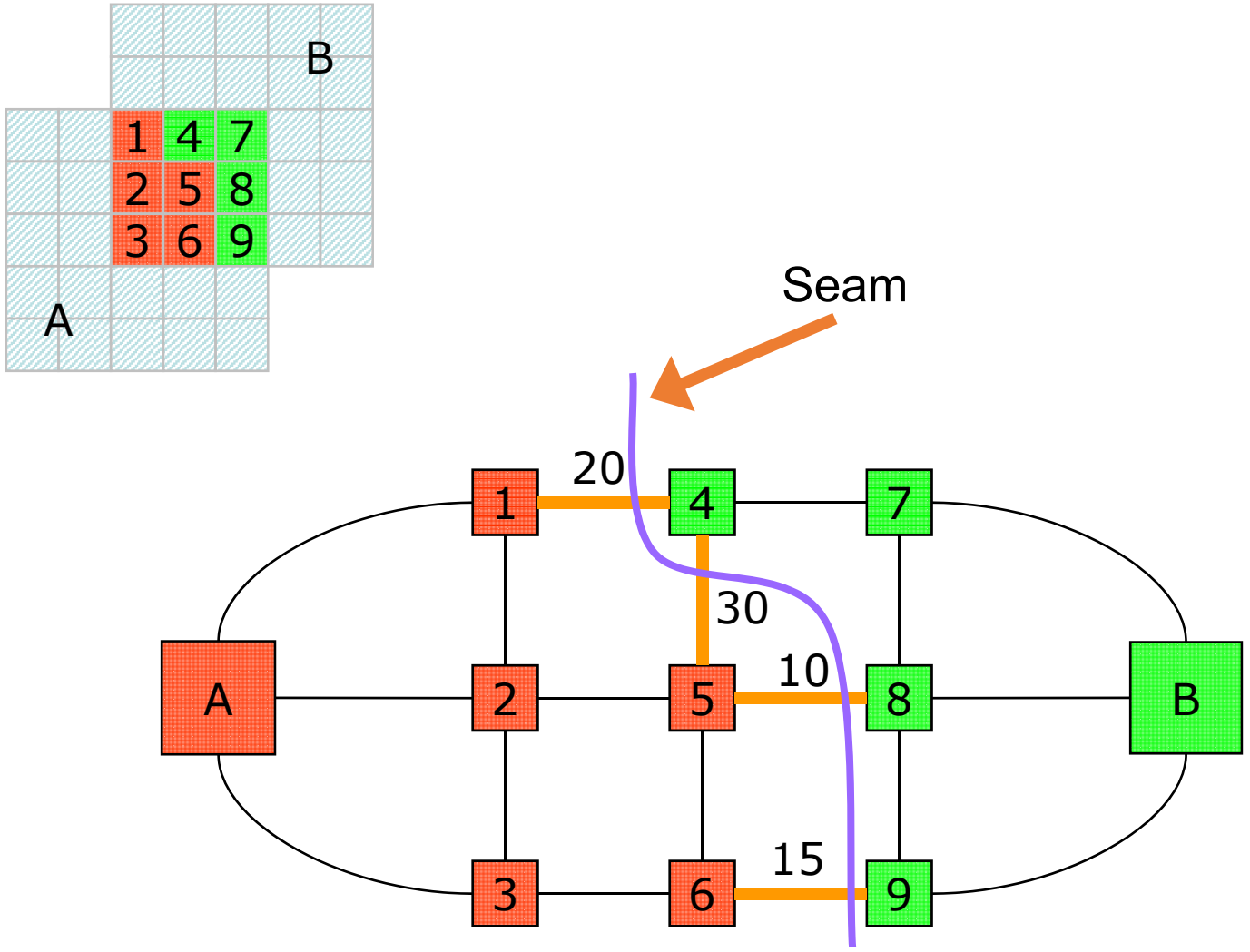

Construct graph such that:

Graph Mincut ⇔ Best Seam

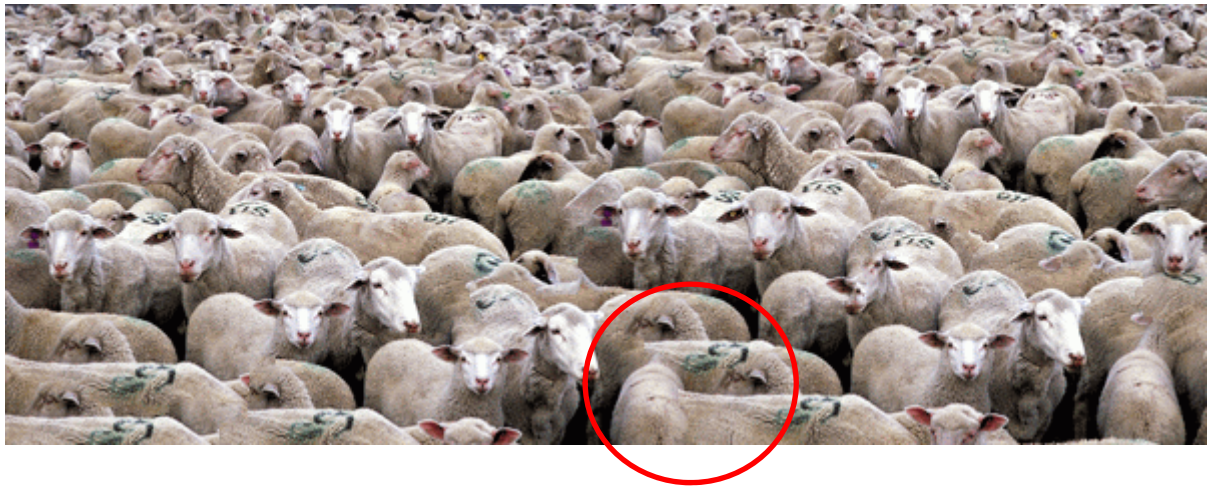
# Seam Optimization



# Seam Optimization



# Results: Natural Images



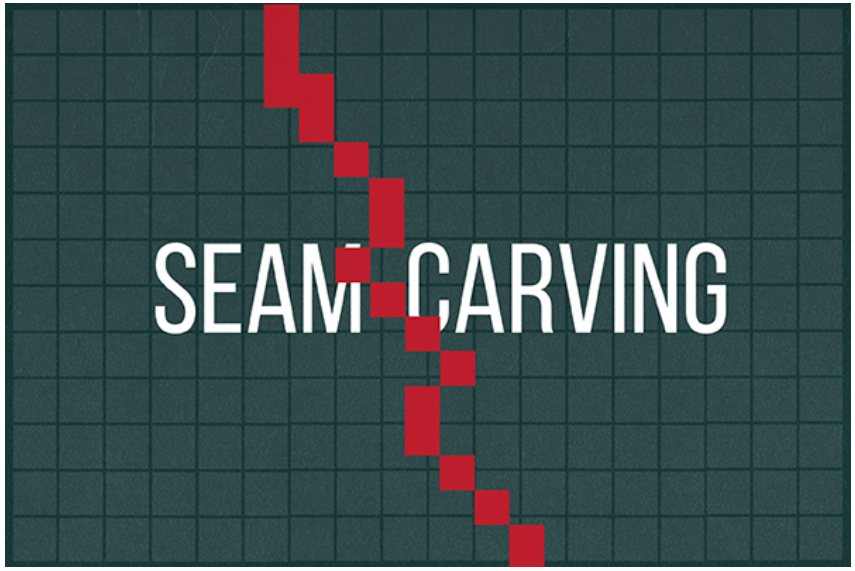
# Interactive Merging

# Results: Interactive Merging

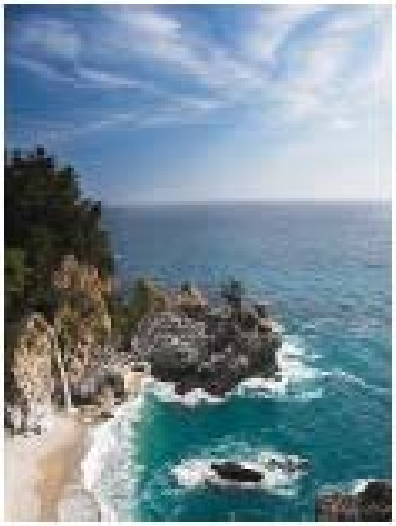
# Application: Seam Carving

[Avidan and Shamir, Siggraph 2007]

- Removing/adding unimportant seams
- Content-aware image resizing/retargetting




# Seam Carving

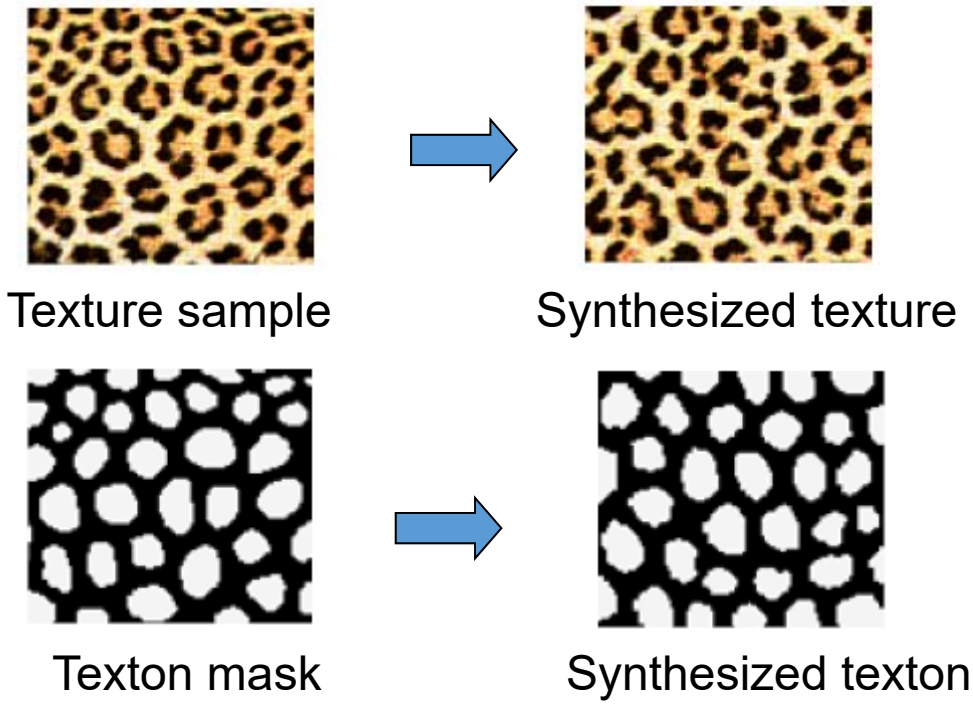


Seam carving Scaling Cropping

# Texton-based Synthesis

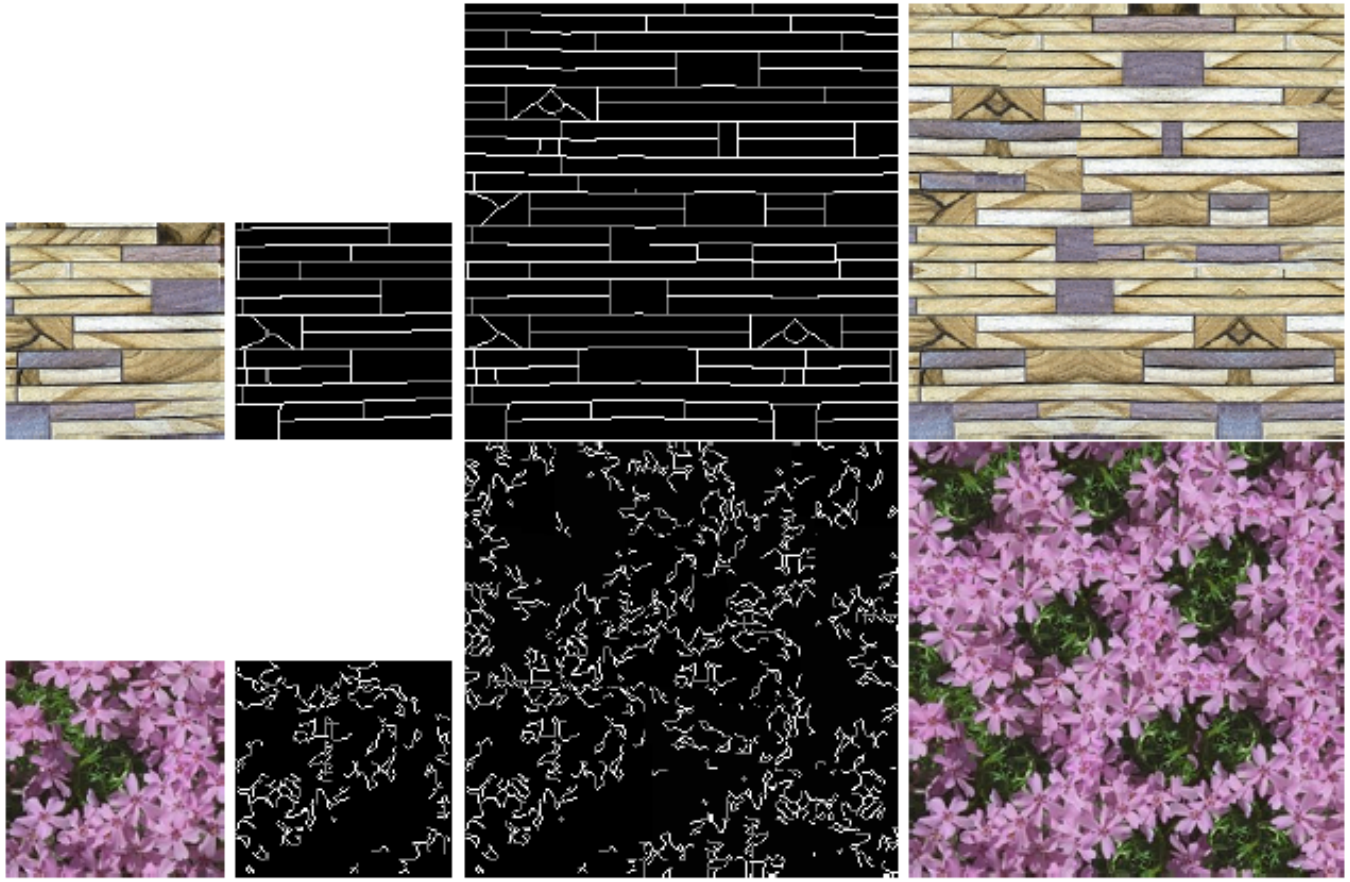
[Zhang et al., Siggraph 2003]

- Texton: texture element
- Texture elements don't break apart using texton synthesis



# Feature Map + Texture Map

[Wu&Yu, Siggraph 2004]



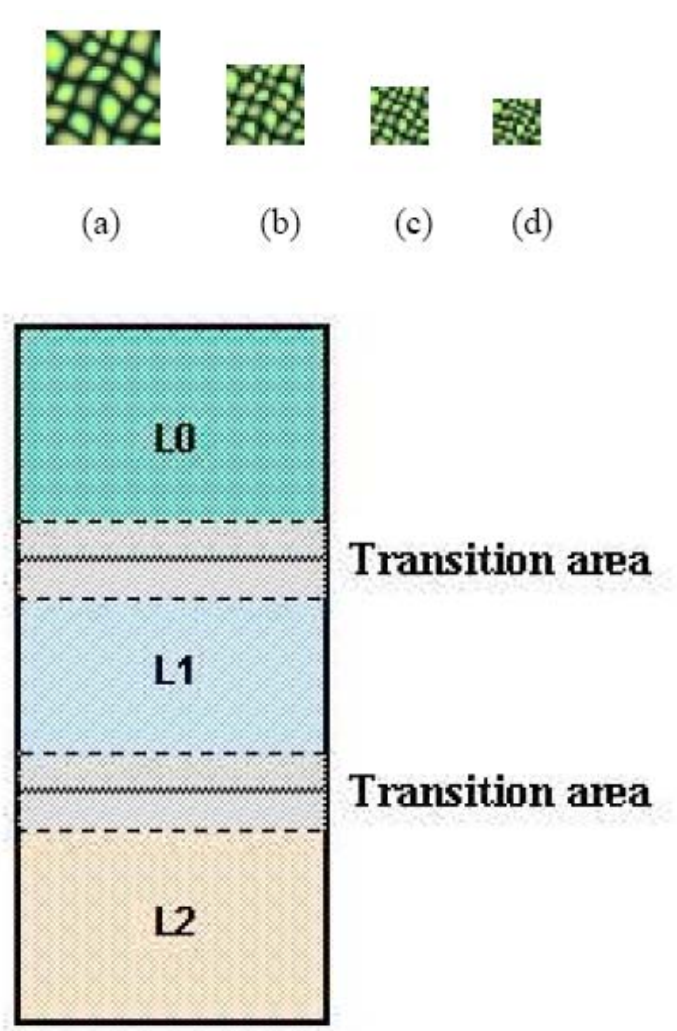
sample textures

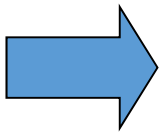
feature maps

synthesized feature maps

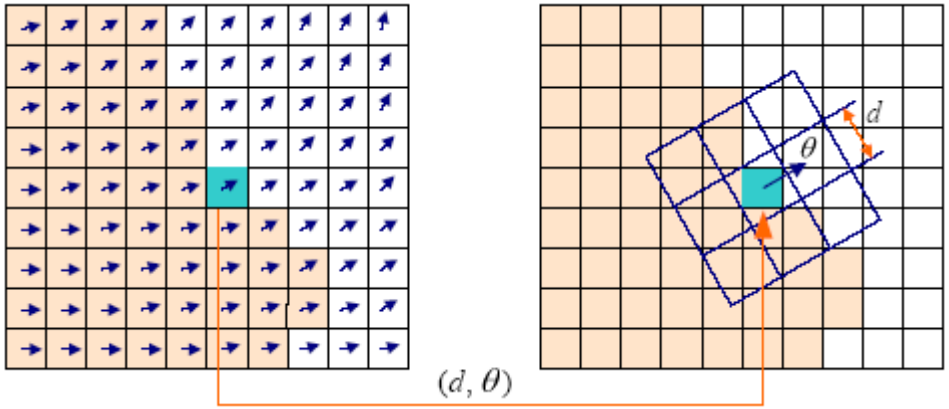
output textures

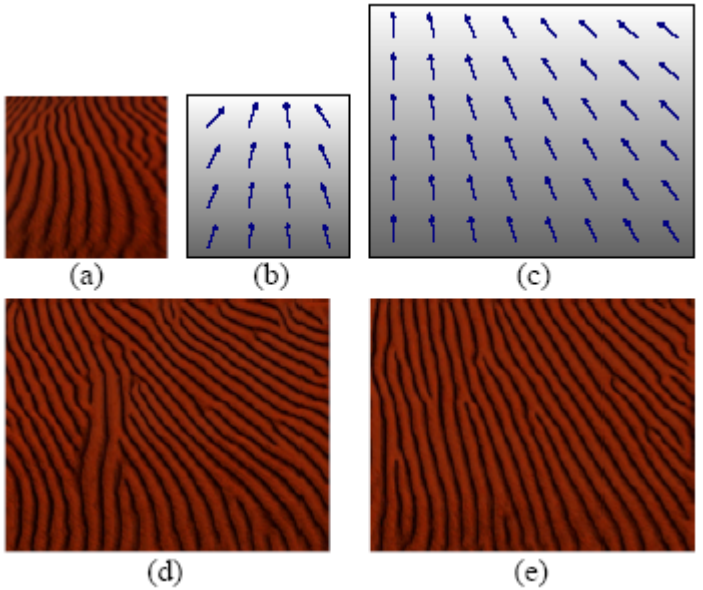
# Synthesis with Local Size Control

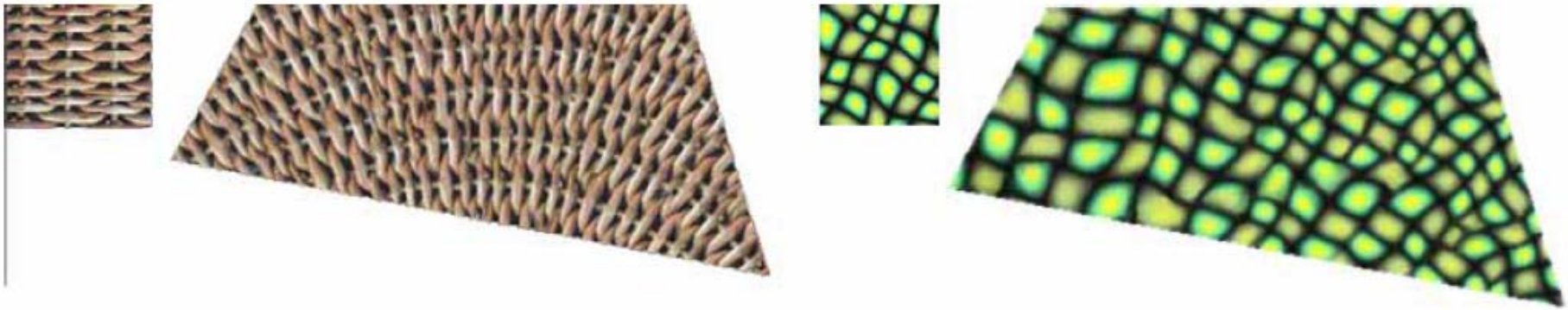


# Synthesis with Vector Field Control

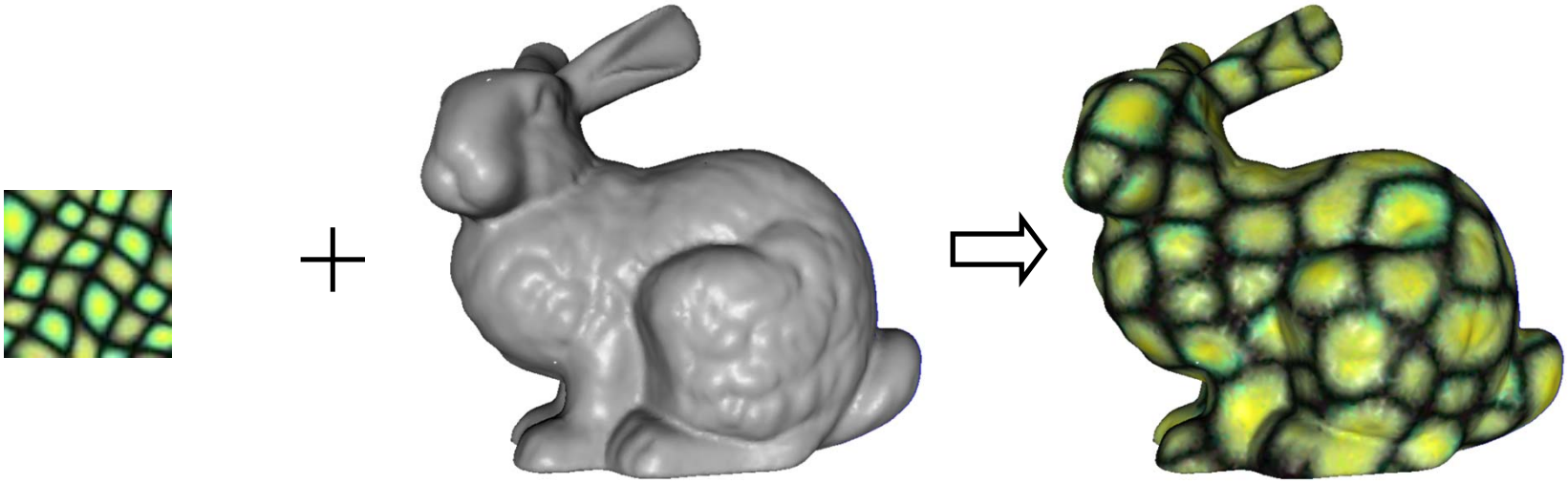


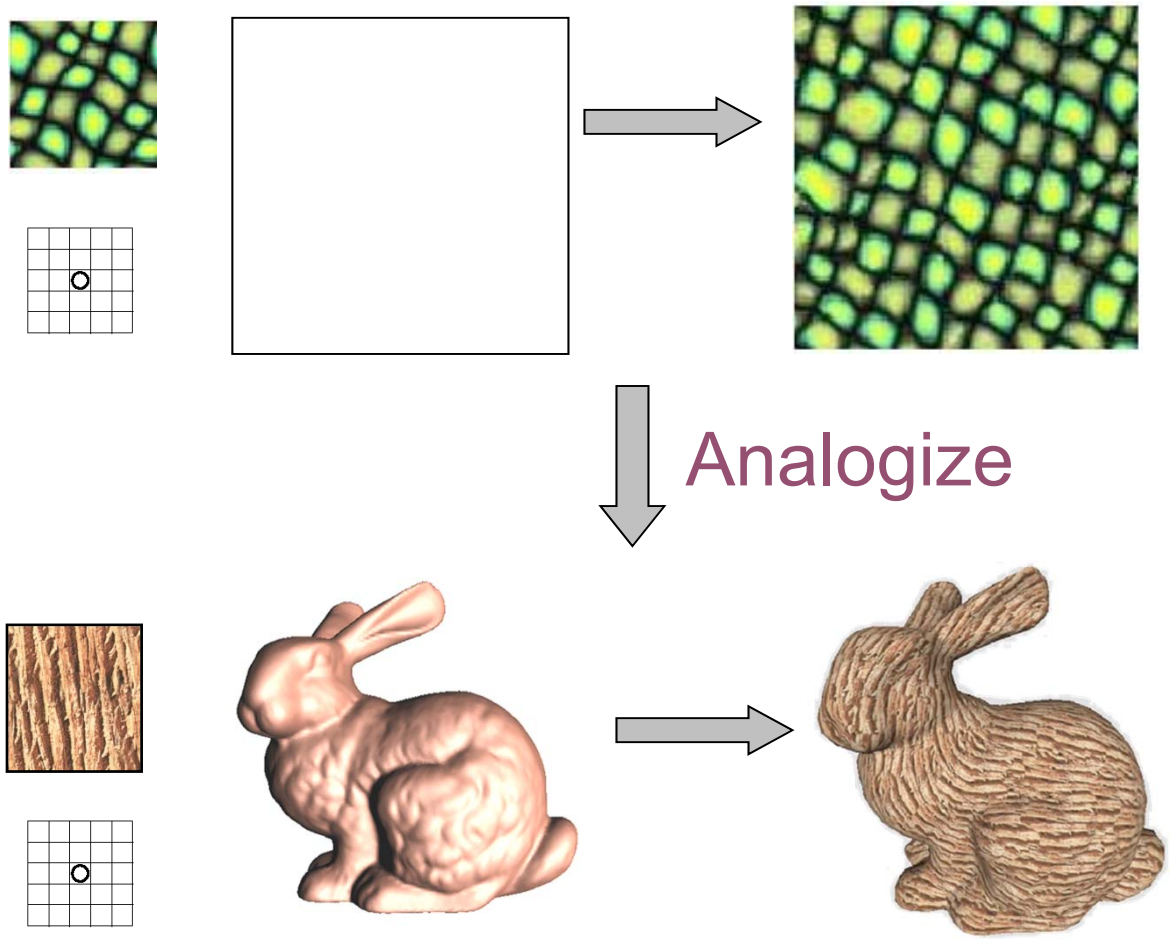



# 3. 3D Texture Synthesis

# Texture Synthesis on Surfaces

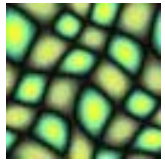


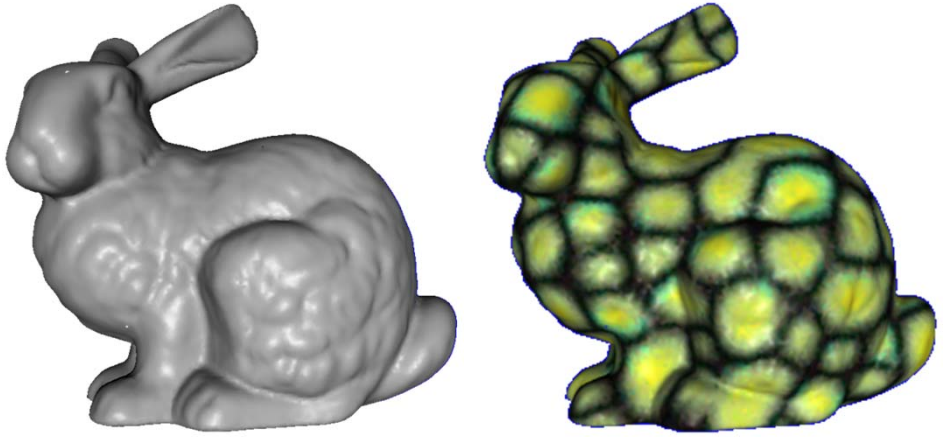
# Analogy



# Desired Properties

- Share advantages of 2D algorithm
  - Quality
  - Efficient
  - General
  - Easy to use
- Minimum distortion
- Minimum discontinuity





# 3D Texture Synthesis

- Procedural texture synthesis
- Sample-based texture synthesis
- Geometry synthesis

# 3D Texture Synthesis

- Procedural texture synthesis
- Sample-based texture synthesis
- Geometry synthesis

# Procedural Textures

- Use 3D functions to define texture properties of objects
  - Non-trivial programming
  - Flexibility
  - Parametric control
  - Unlimited resolution, antialiasing possible
  - Low memory requirements
  - Low-cost visual complexity
  - Adapts to arbitrary geometry

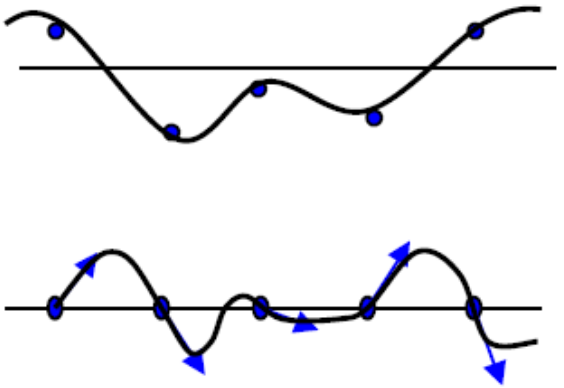
# Procedural Textures

- **Analytic scalar function of world coordinates** $(x,y,z)$
- **Texturing: evaluation of function on object surface**
  - Ray tracing: 3D intersection point with surface
- **Textures of natural objects**
  - Similarity between different patches
    - Repetitiveness, coherence
  - Similarity on different resolution scales
    - Self-similarity
  - But never completely identical
    - Additional disturbances, turbulence, noise
- **Procedural texture function**
  - Mimics statistical properties of natural textures
  - Purely empirical approach
    - Looks convincing, but has nothing to do with material's physics

# Perlin's Noise

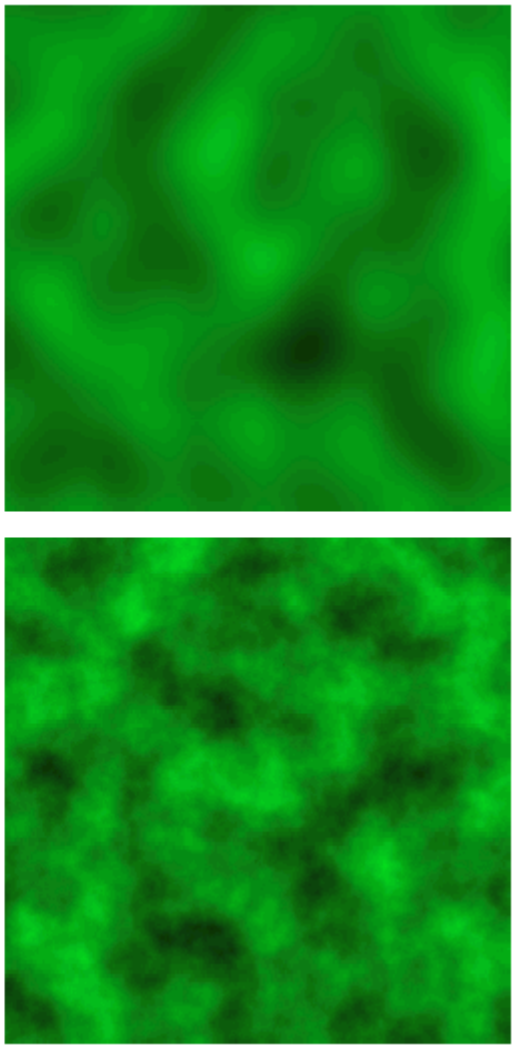
[Perlin, Siggraph 1985]

- **Noise(x,y,z)**
  - Statistical invariance under rotation
  - Statistical invariance under translation
  - Narrow bandpass limit in frequency
- **Integer lattice (i,j,k)**
  - Random number at each lattice point (i,j,k)
    - Look-up table or hashing function
  - Gradient lattice noise
    - Random gradient vectors
- **Evaluation at (x,y,z)**
  - Tri-linear interpolation
  - Cubic interpolation (Hermite spline → later)
- **Unlimited domain**
  - Lattice replicated to fill entire space
- **Fixed fundamental frequency of ~1 Hz over lattice**
- **Smooth interpolation of interim values**

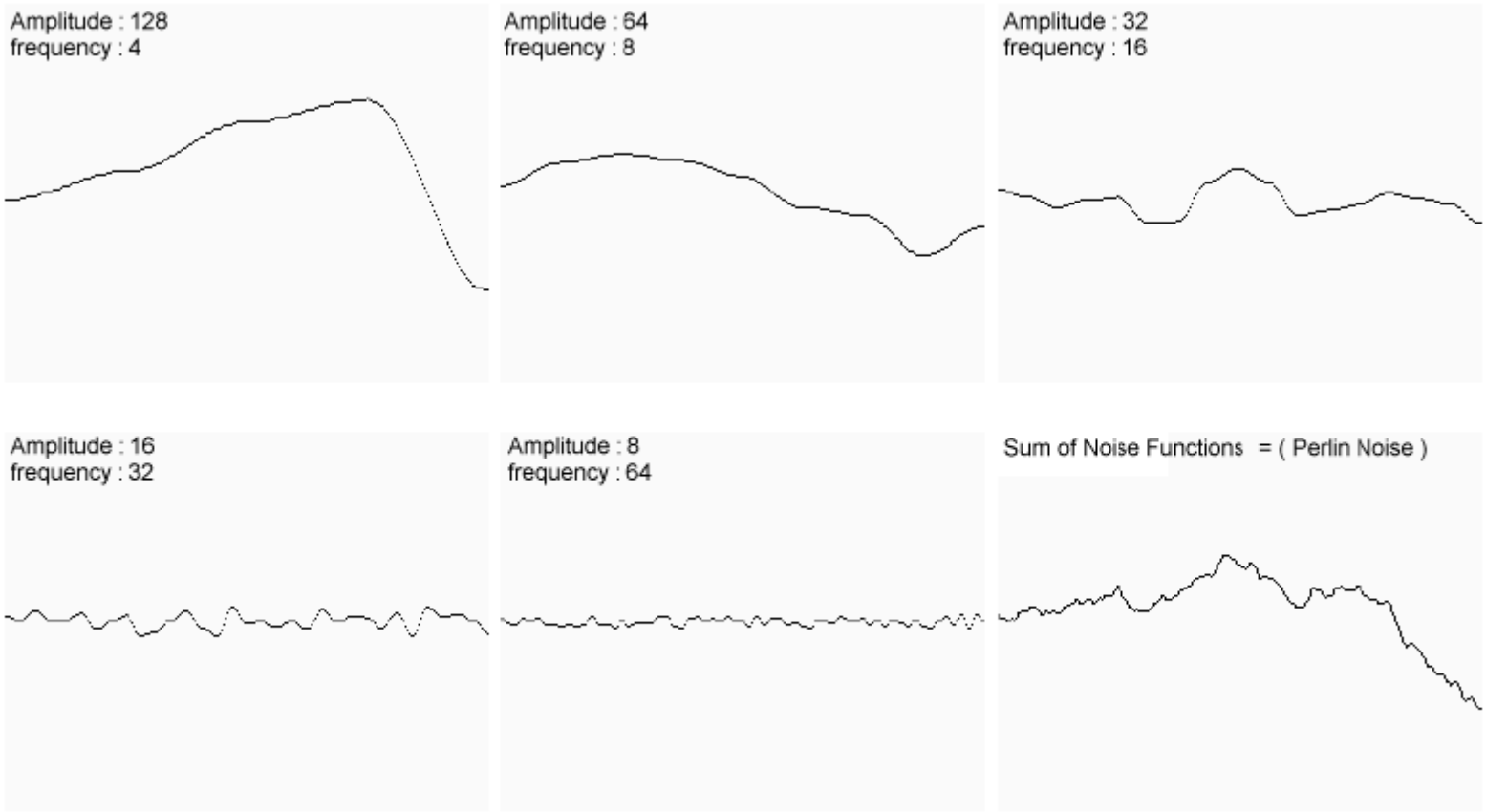


# Turbulence Function

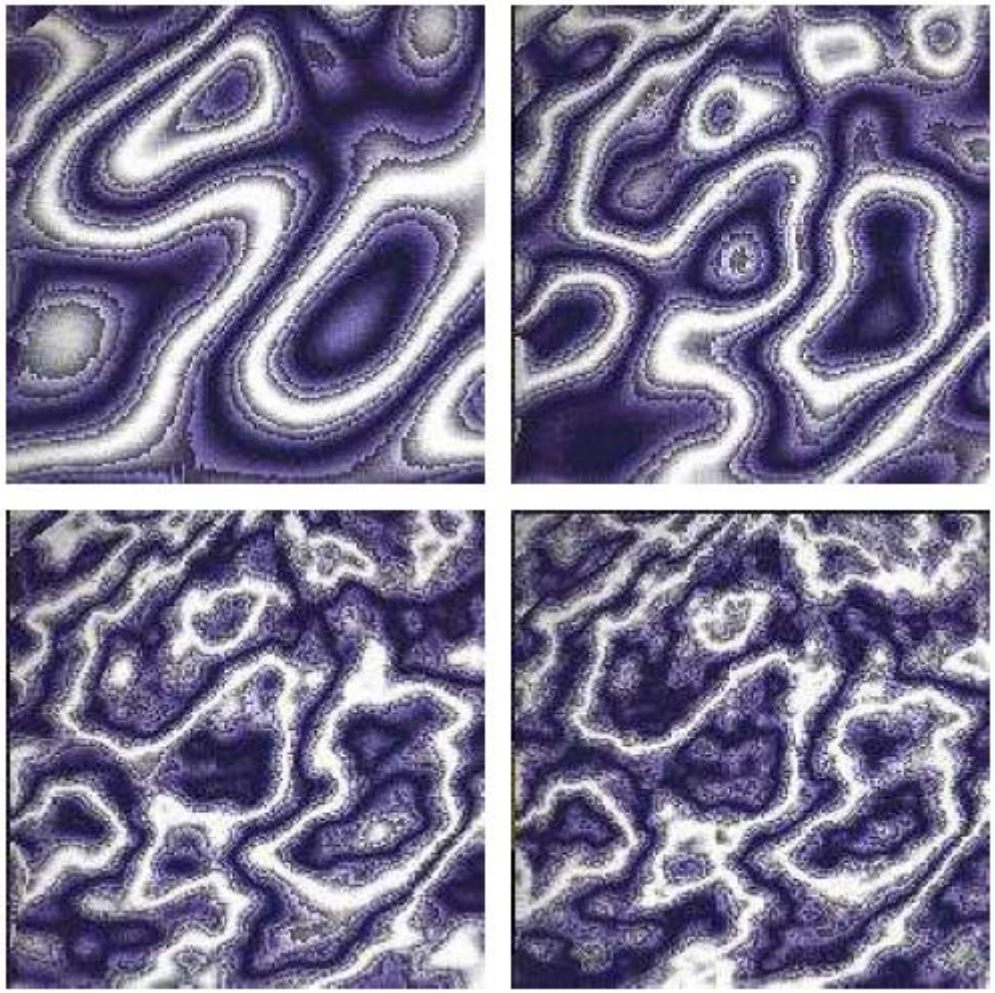
- **Noise function**
  - "White" frequency spectrum
- **Natural textures**
  - Decreasing power spectrum towards high frequencies
- **Turbulence from noise**
  - Turbulence(x) = $\sum^{k}_{i=0}$ abs(noise($2^i$ x) / $2^i$ )
  - Summation truncation
    - $1/2^{k+1}$ < size of one pixel (band limit)
  - 1. Term: noise(x)
  - 2. Term: noise(2x)/2
  - …
  - Power spectrum: 1/f
  - (Brownian motion: $1/f^2$)



# Synthesis of Turbulence (1D)

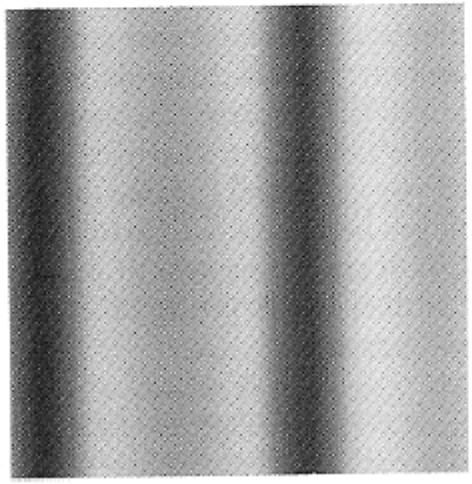


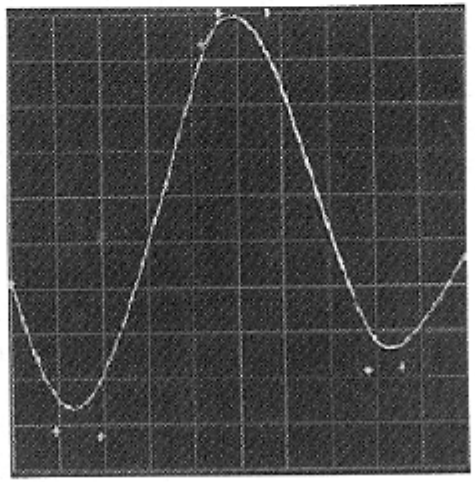
# Synthesis of Turbulence (2D)



# Example: Marble Texture Function

- **Overall structure: alternating layers of white and colored marble**
  - $f_{marble}(x,y,z)$ :=marble_color(sin(x))
  - marble_color : transfer function
- **Realistic appearance: simulated turbulence**
  - $f_{marble}(x,y,z)$ :=marble_color(sin(x+turbulence(x,y,z)))
- **Moving object: turbulence function also transformed**





# Reaction Diffusion Based Method

[Turk, Siggraph 1991]

- Two-chemical reaction-diffusion system

$$\frac{\partial a}{\partial t} = F(a,b) + D_a \nabla^2 a$$

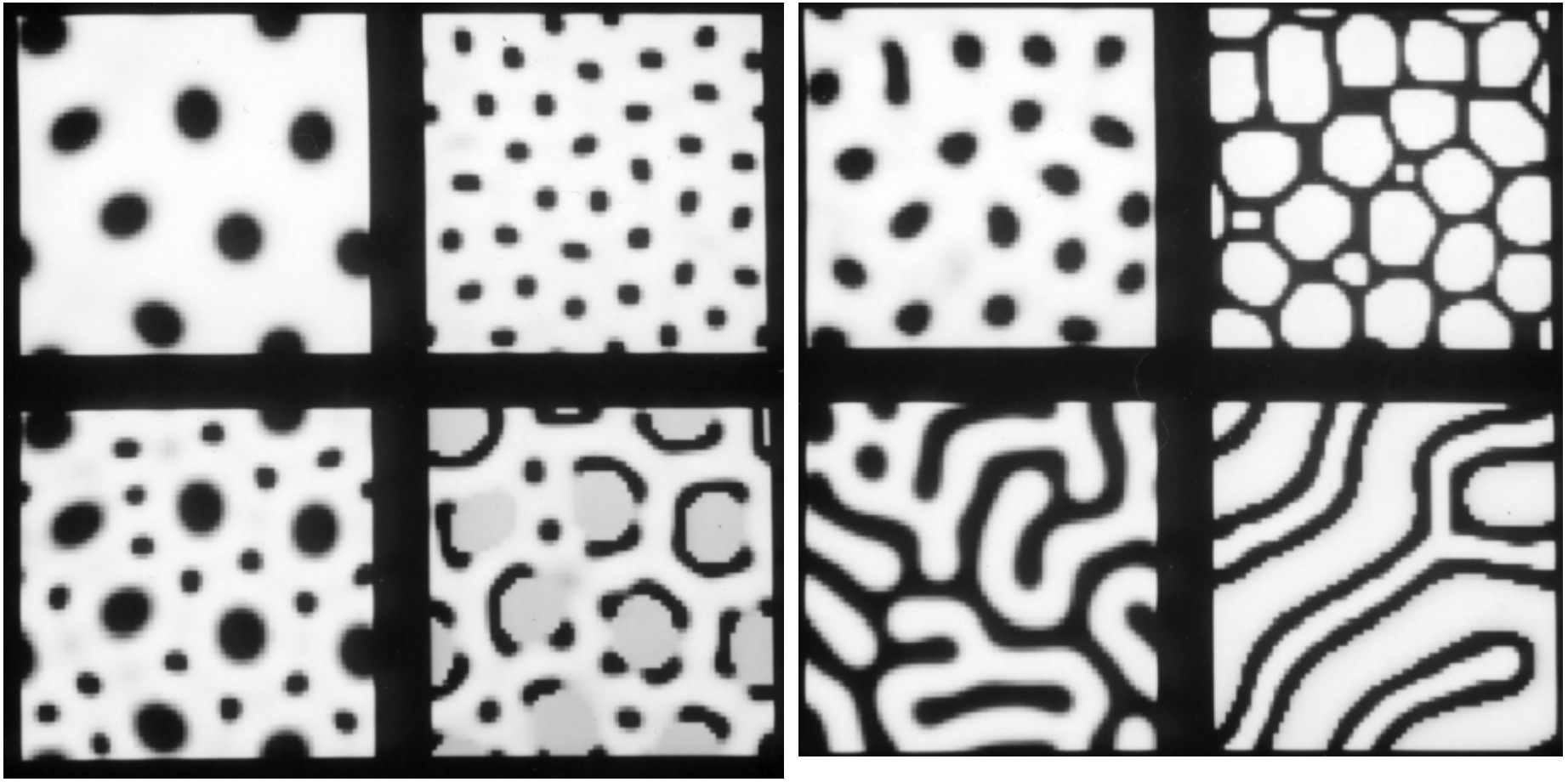
$$\frac{\partial b}{\partial t} = G(a,b) + D_b \nabla^2 b$$

- Discrete computation

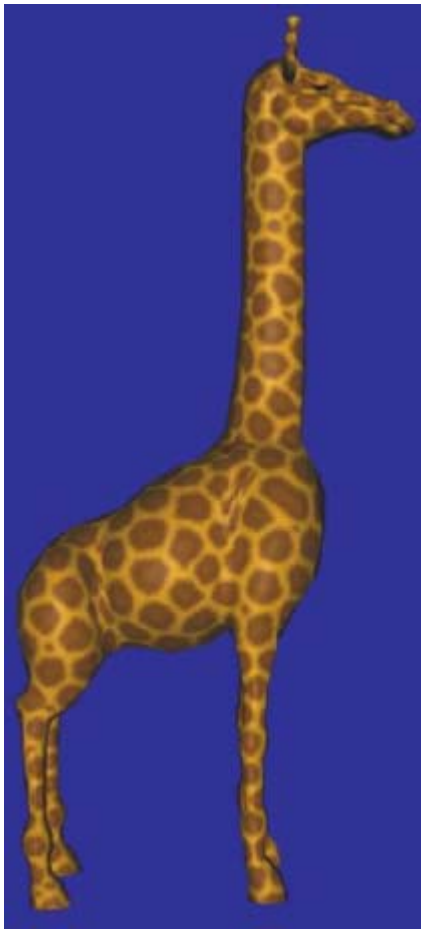
$$\Delta a_{ij} = s\,(16 - a_{ij}\, b_{ij}) + D_a\,(a_{i+1j} + a_{i-1j} + a_{ij+1} + a_{ij-1} - 4a_{ij})$$

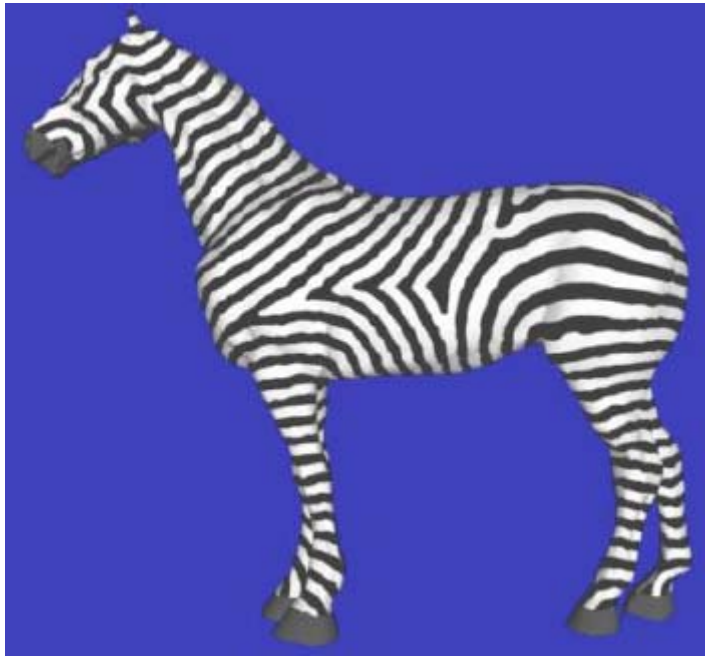
$$\Delta b_{ij} = s\,(a_{ij}\, b_{ij} - b_{ij} - \beta_{ij}) + D_b\,(b_{i+1j} + b_{i-1j} + b_{ij+1} + b_{ij-1} - 4b_{ij})$$

# Examples of Reaction-Diffusion



# Results

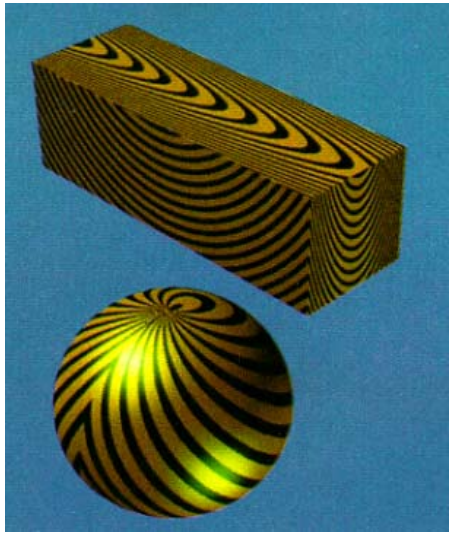


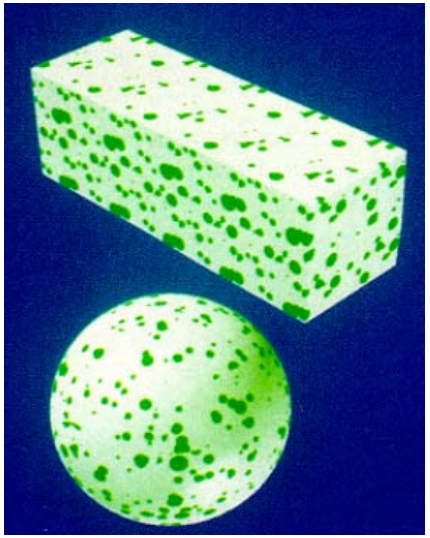


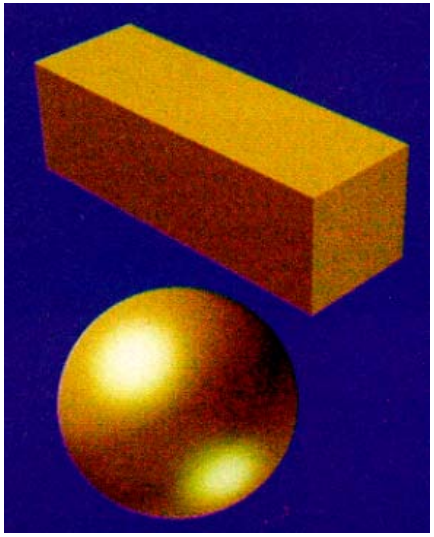
# Solid Textures

[Peachey, Siggraph 1985]

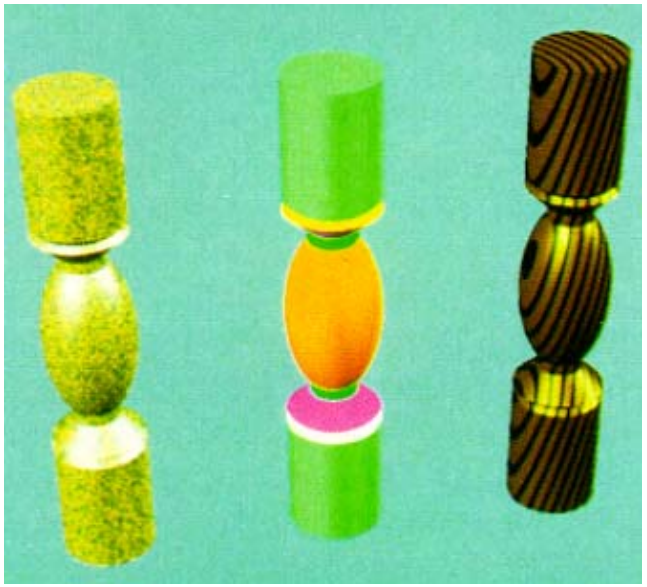
- Solid texture functions in 3D space
- Nonhomogeneous materials
  - wood and stone

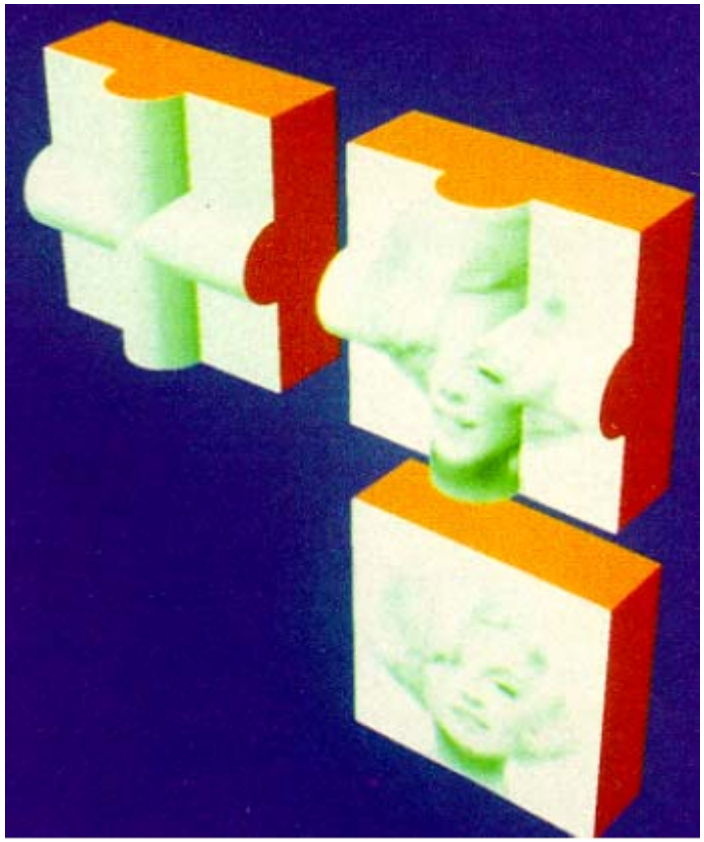


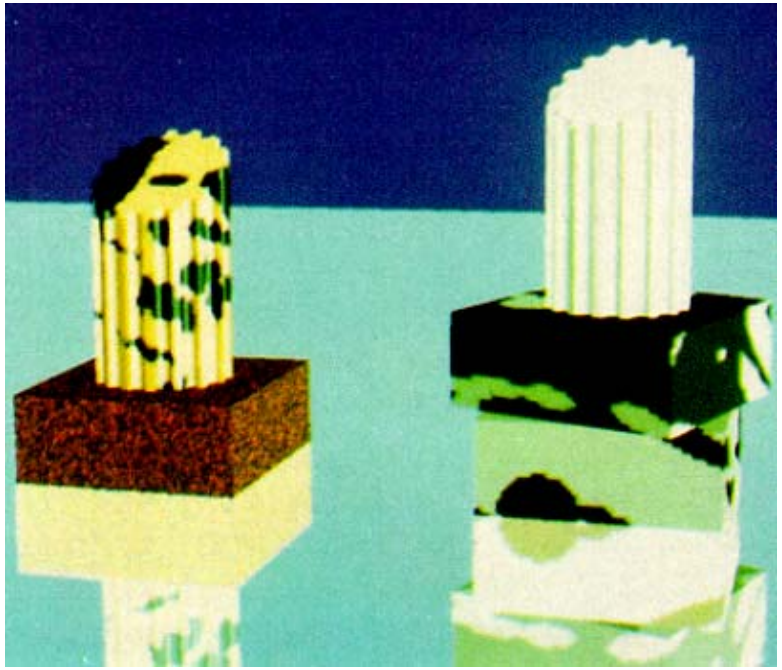




# Examples







# 3D Texture Synthesis

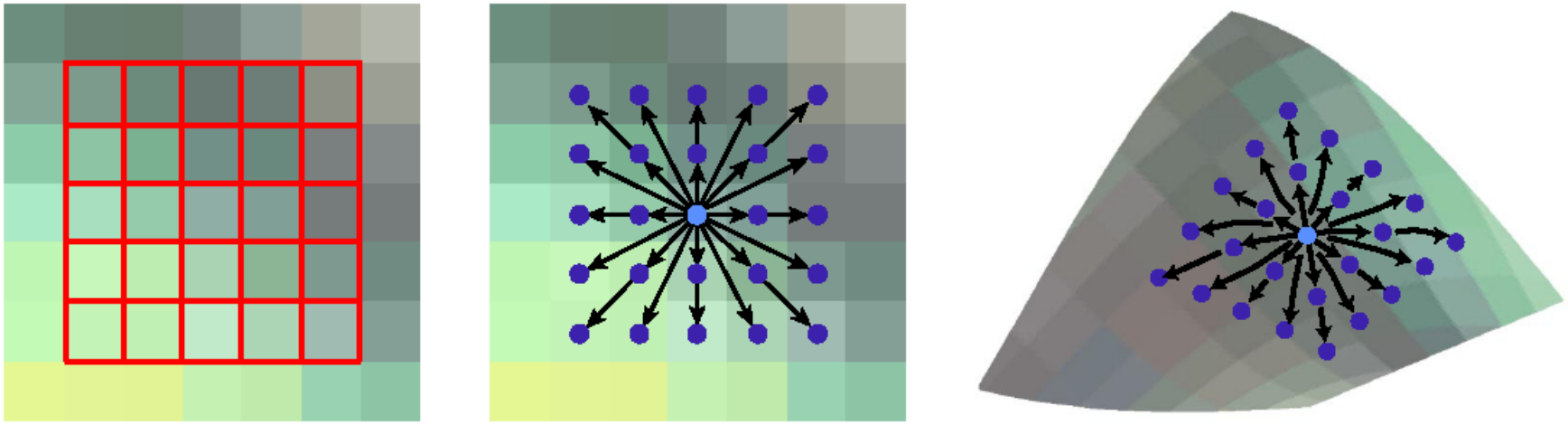
- Procedural texture synthesis
- Sample-based texture synthesis
- Geometry synthesis

# Methodology

- Synthesize a surface texture by coloring mesh vertices
- Extensions from 2D texture synthesis
- Key issues
  - Resampling
  - Local flattening
    - Size
    - Orientation

Vertex colors on three levels in the mesh hierarchy

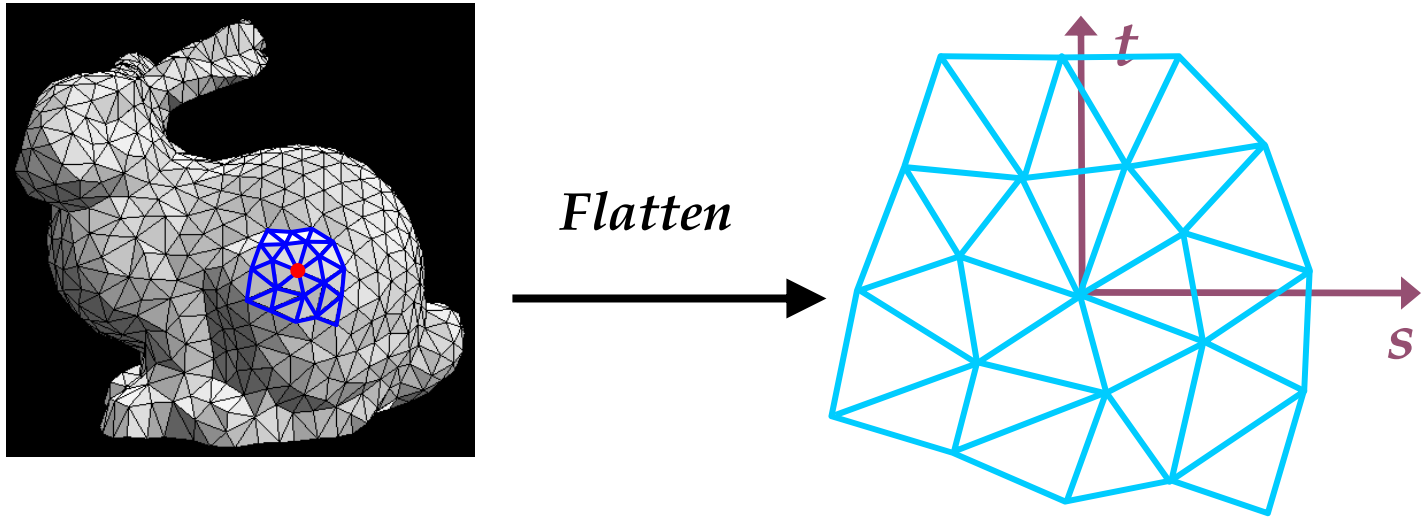
# Neighborhood Sampling on Surface



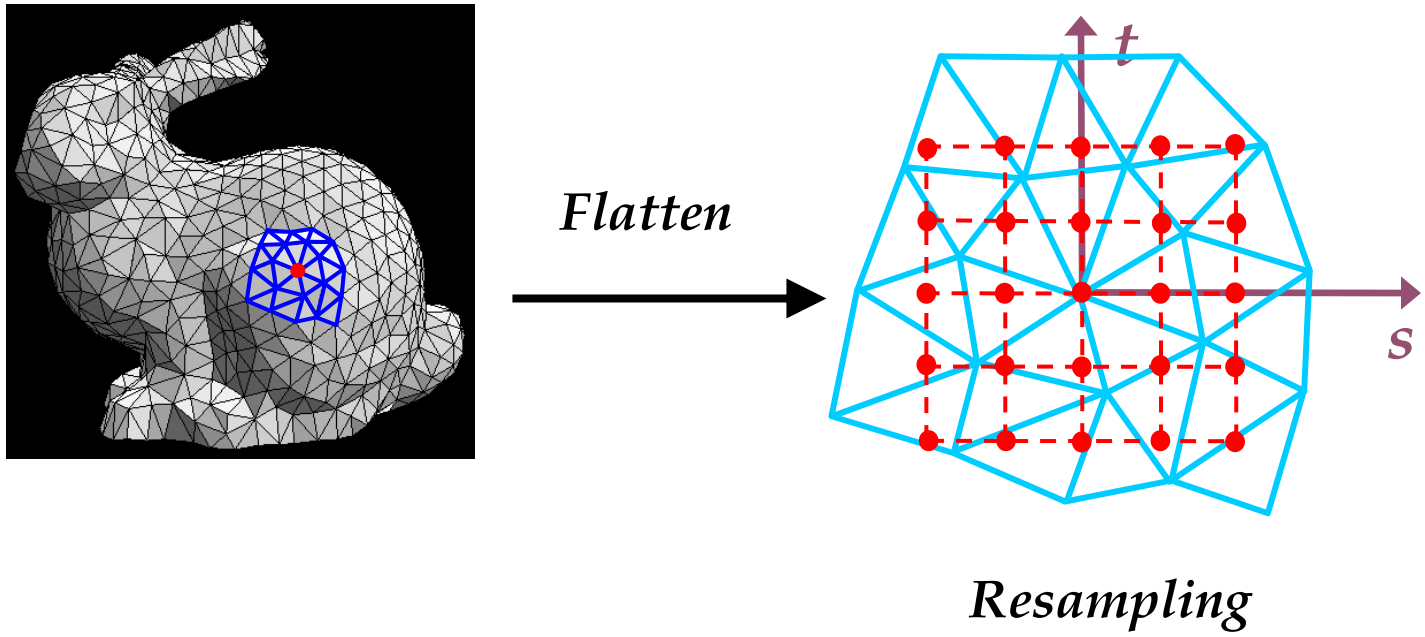

A pixel in image

A vertex on surface

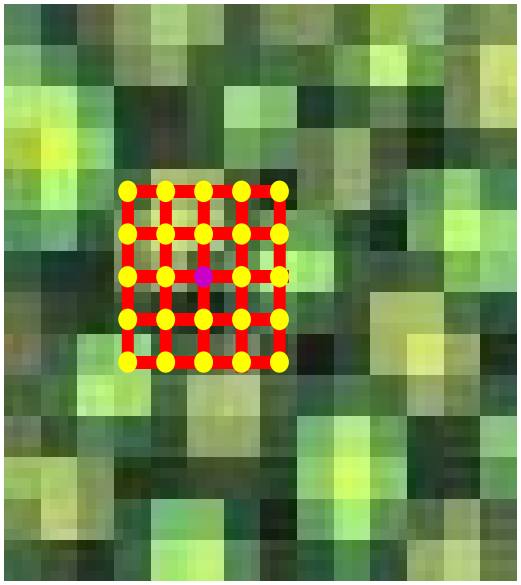
# Resampling

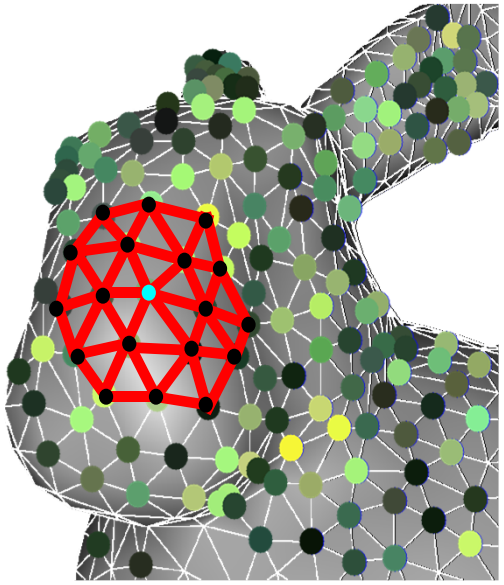


# Resampling

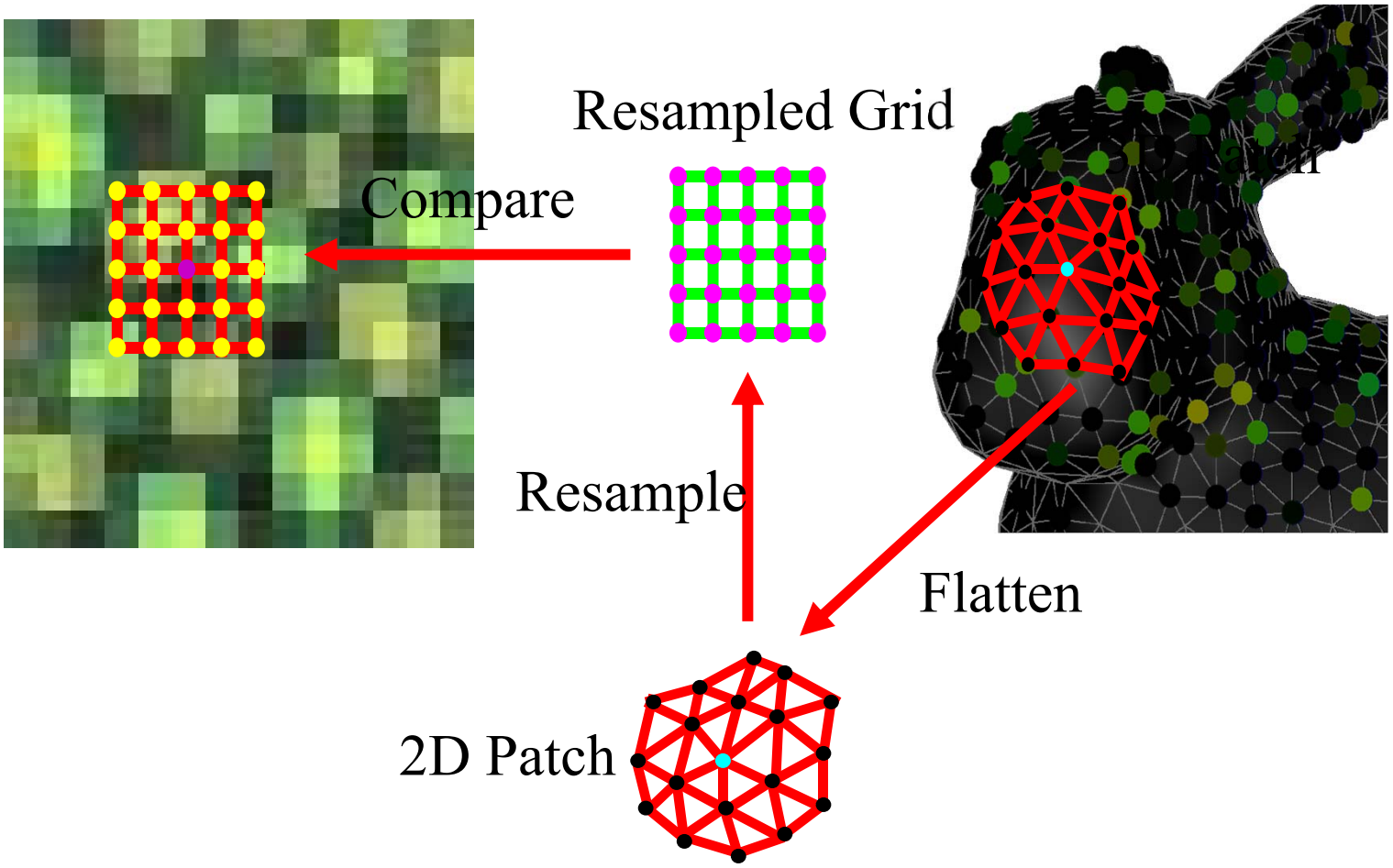


# Neighborhood Comparison

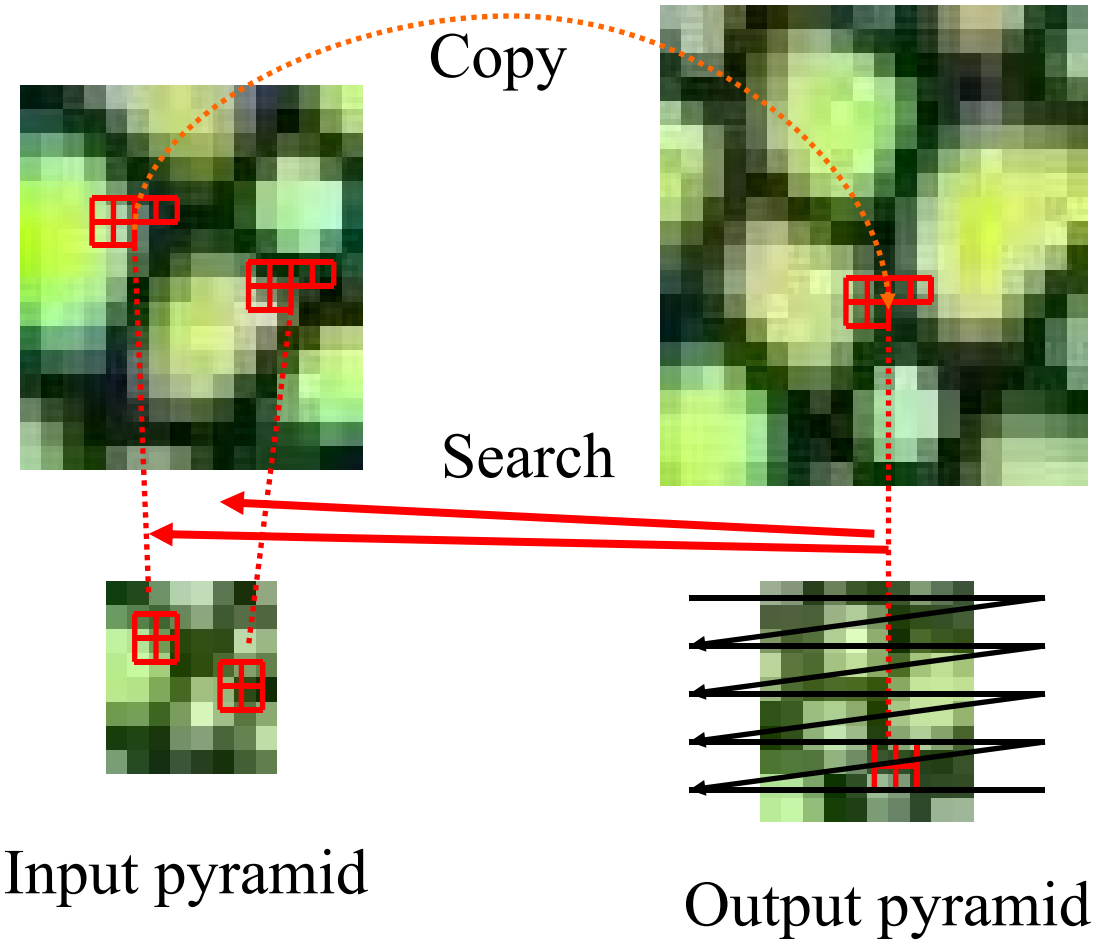




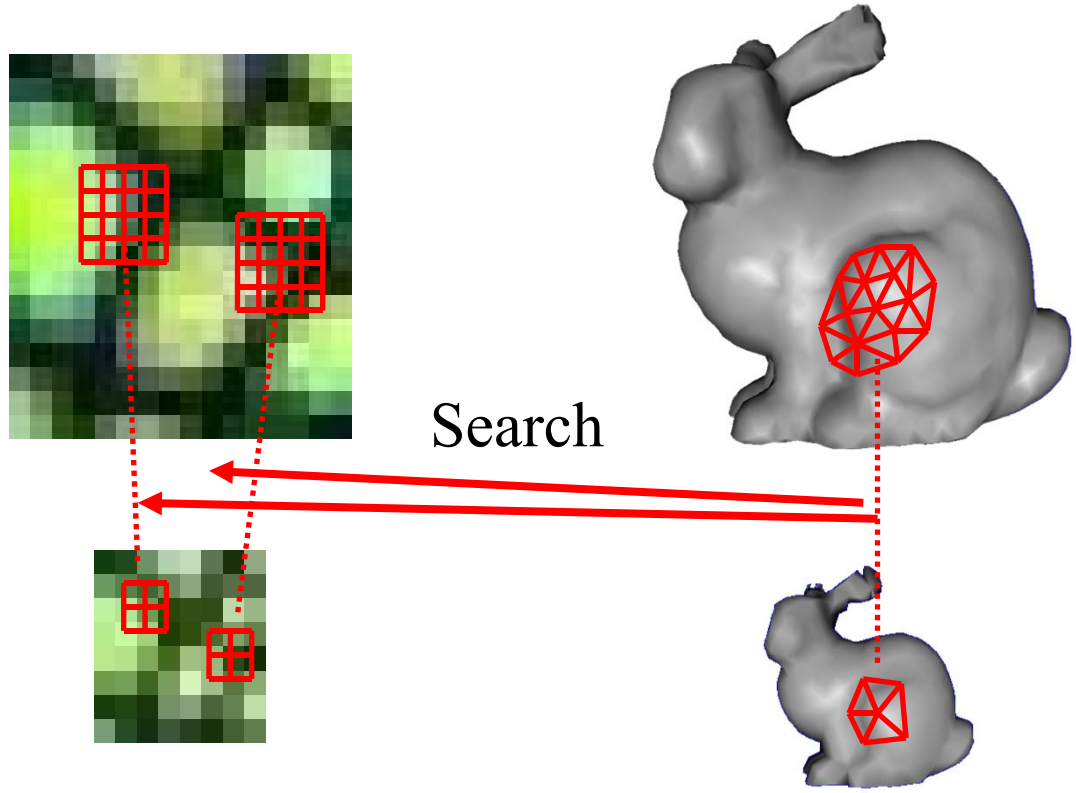
# Mesh Neighborhood



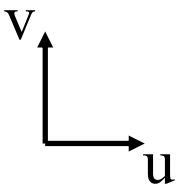
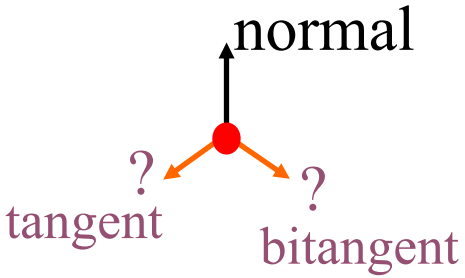
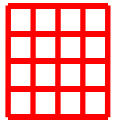
# Recap: Texture Synthesis by Neighborhood Search



# Surface Texture Synthesis by Neighborhood Search



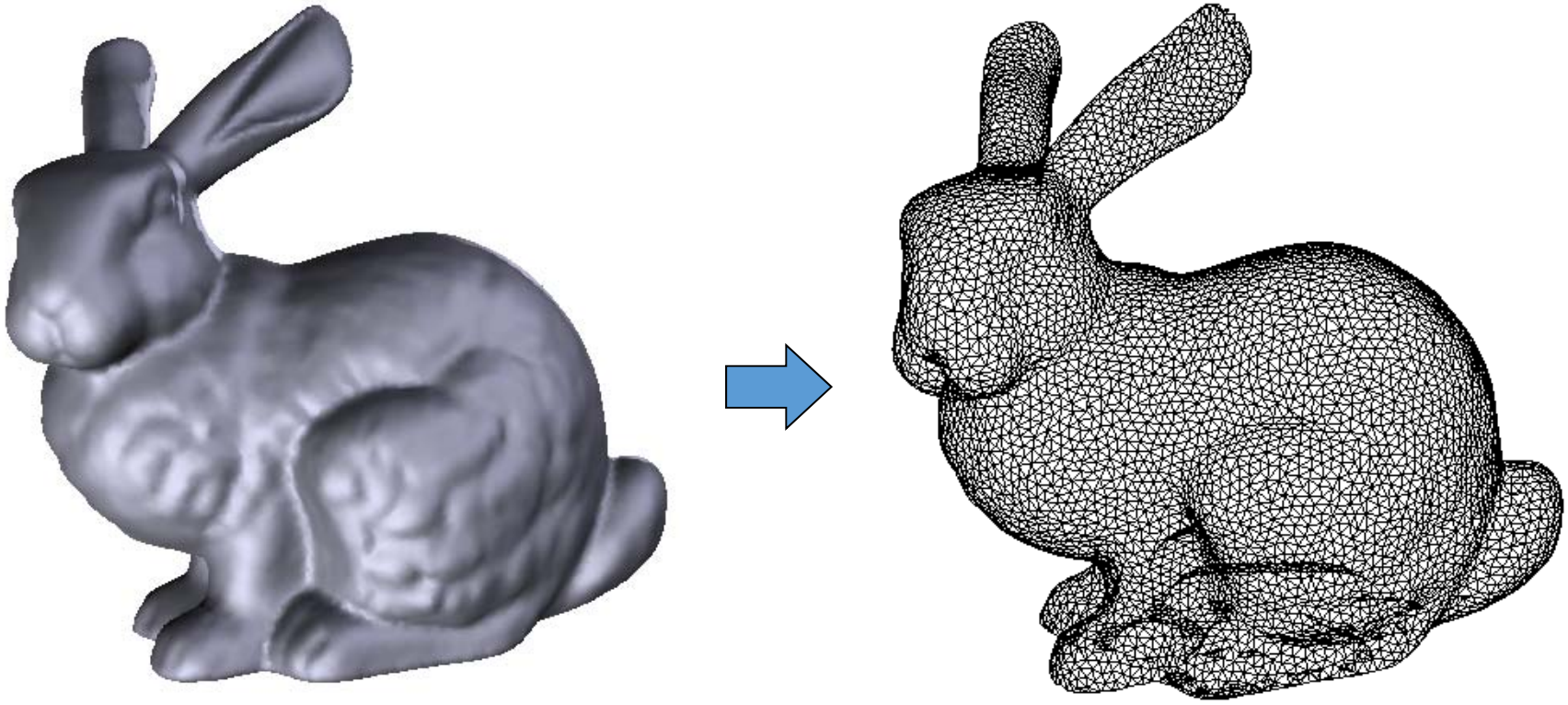
# Differences between 2D and 3D

| | image | mesh | Solution |
|---|---|---|---|
| Pixels/Vertices | grid | ? | mesh re-tiling [Turk'92] |
| Local Orientation (Vector field) |   |   | user-specified relaxation random |
| Synthesis Order | scanline | ? | random |
| Neighborhood |  |  | flattening/resampling |

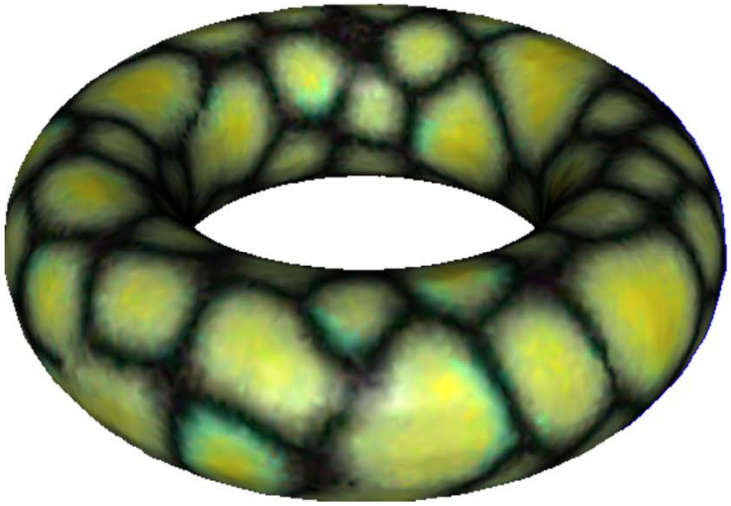
# Sampling on Surface: Retiling

[Turk, Siggraph 1992]

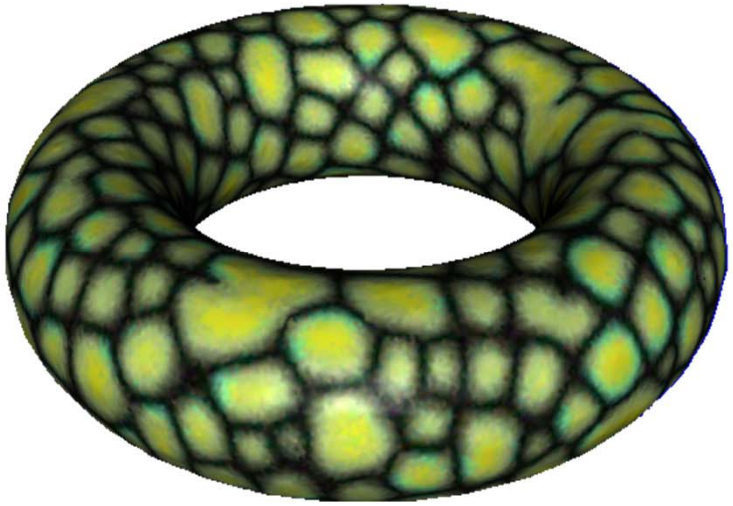
- Distribute a set of vertices over surface as uniformly as possible



# Retiling Density



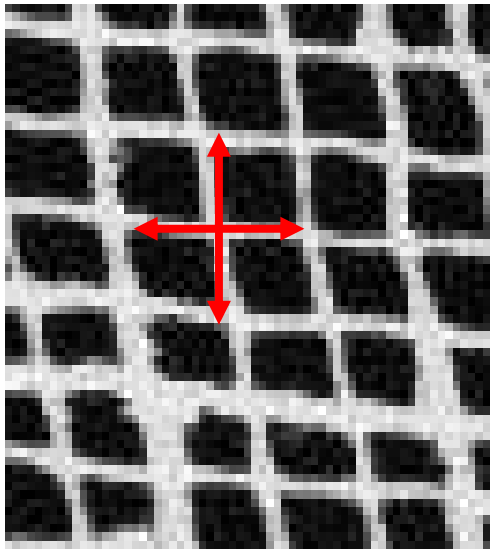
24576 vertices



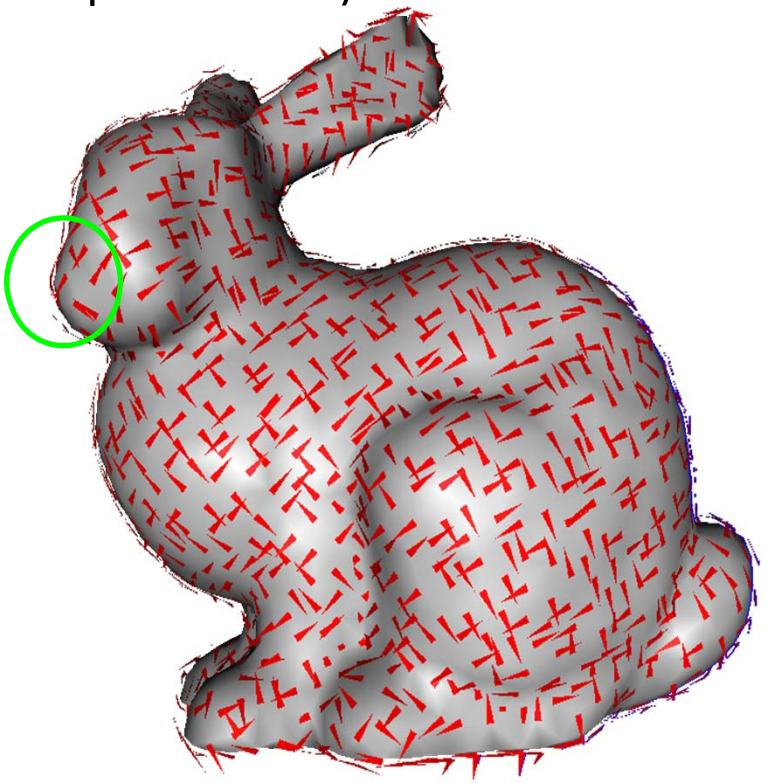
73728 vertices

# Texture Orientation

- Methods for orienting textures
  - user-specified
  - random (for isotropic textures)
  - smooth or symmetric (for anisotropic textures)
    - by relaxation

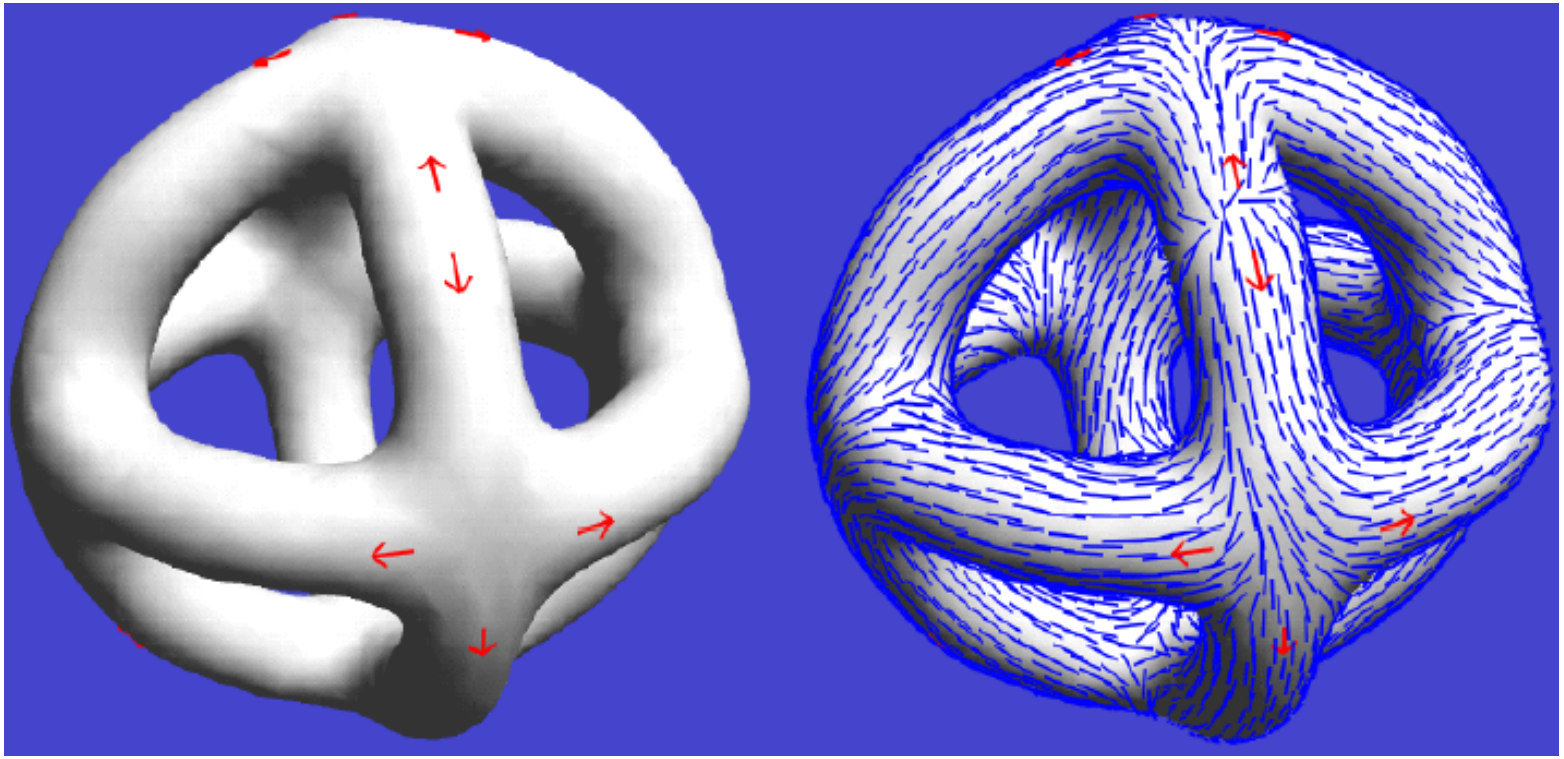


4-way symmetric texture



4-way symmetric vector field

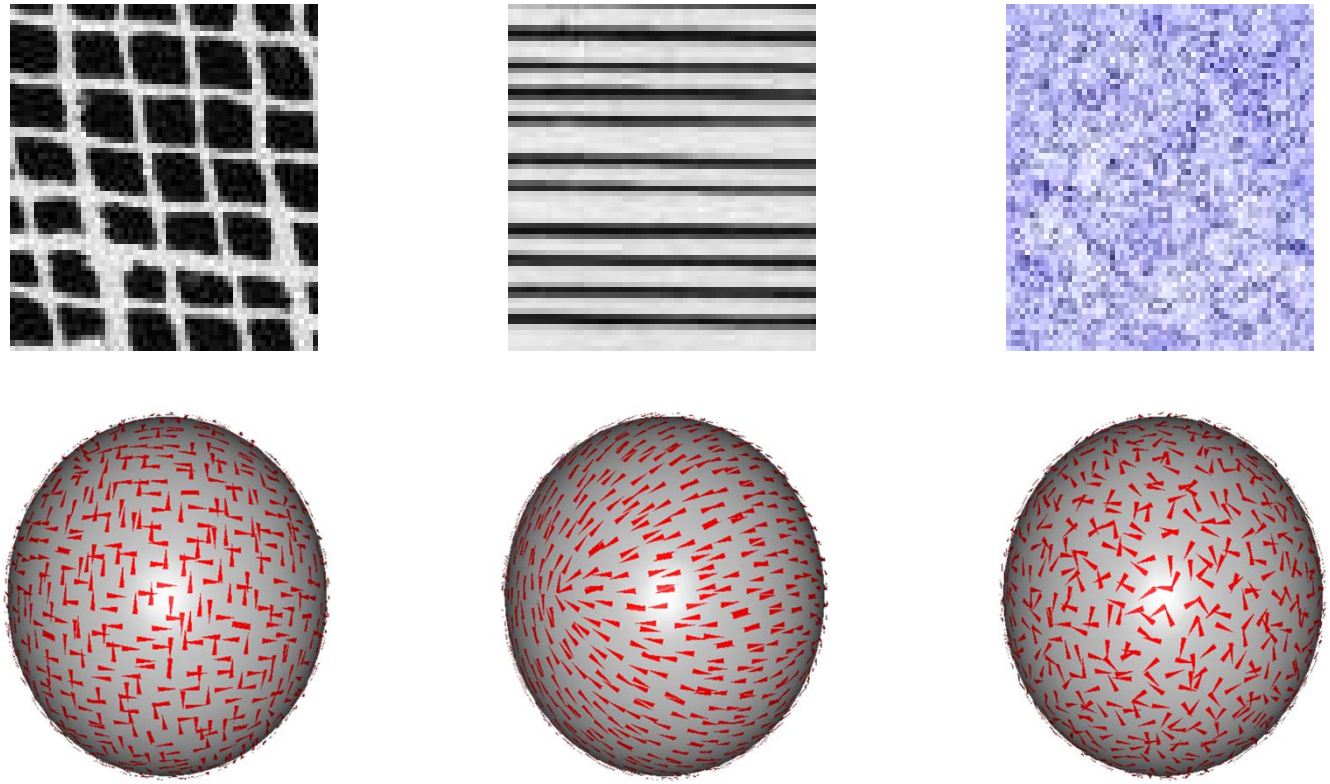
# User Specified Vector Field



User specified vectors
on some vertices

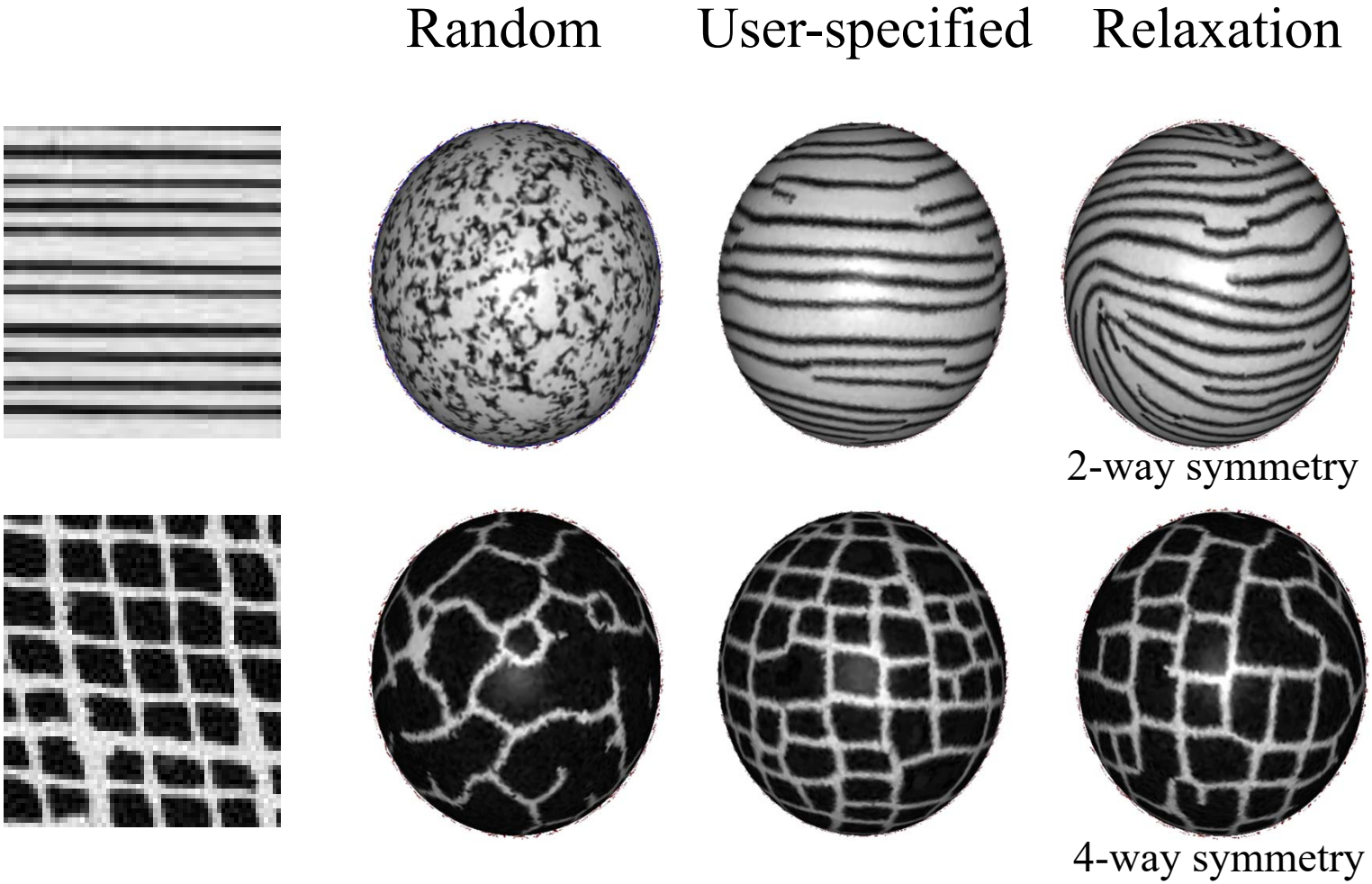
Interpolated vector field
on all vertices

# Texture Orientation Examples



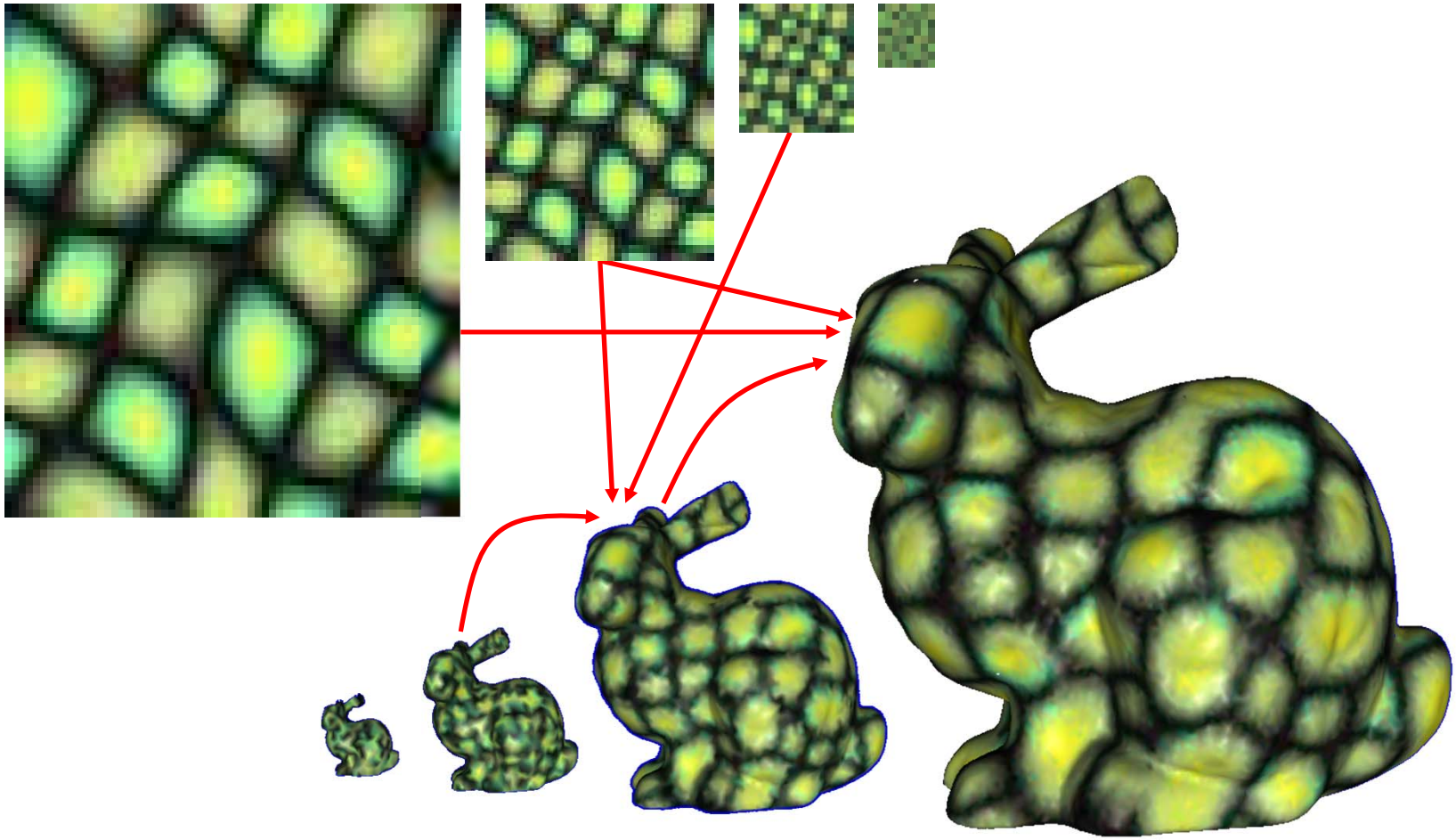
4-way symmetric  2-way symmetric  Random

# Results: Other Orientations

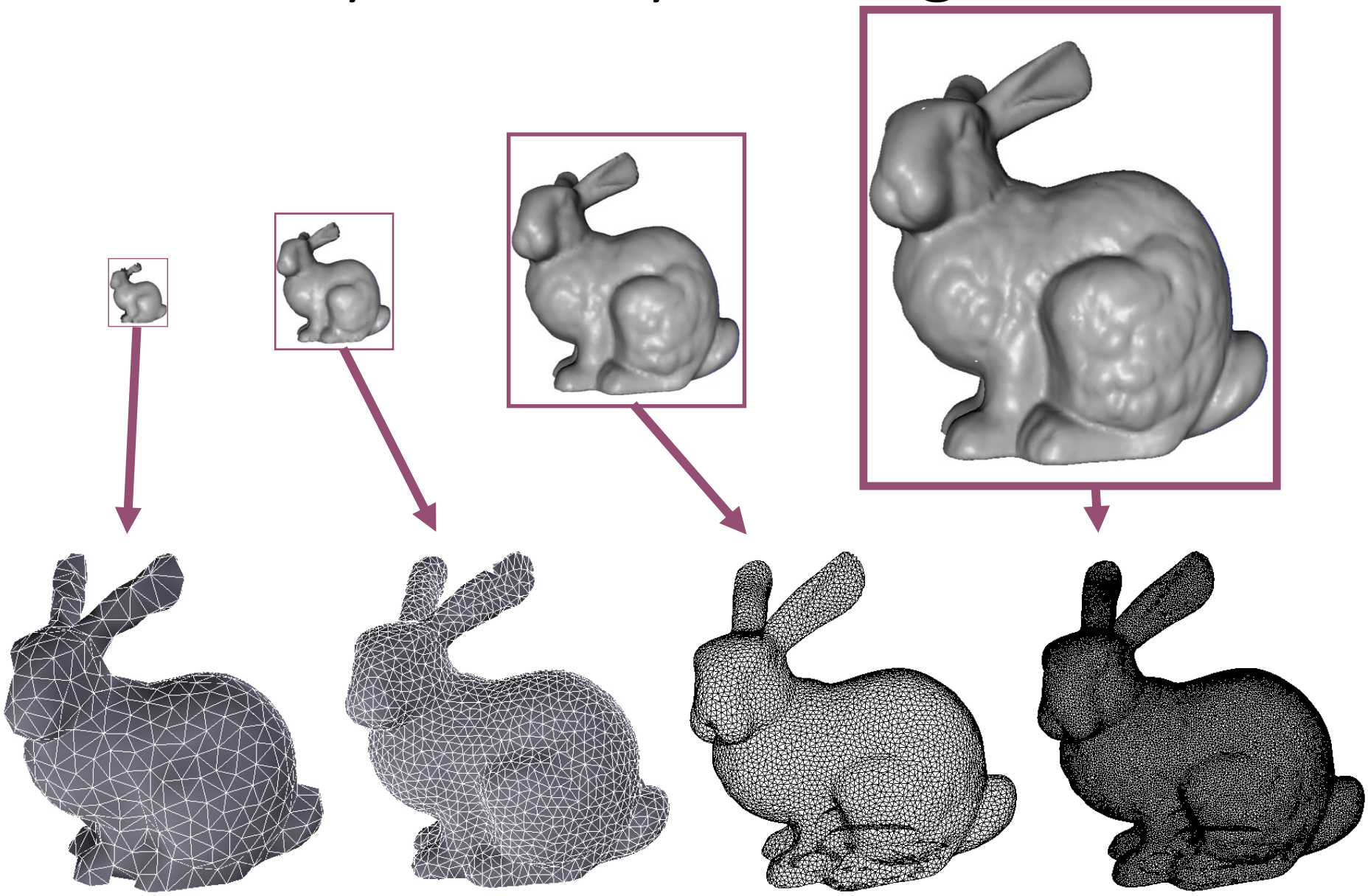


# Multiresolution Synthesis

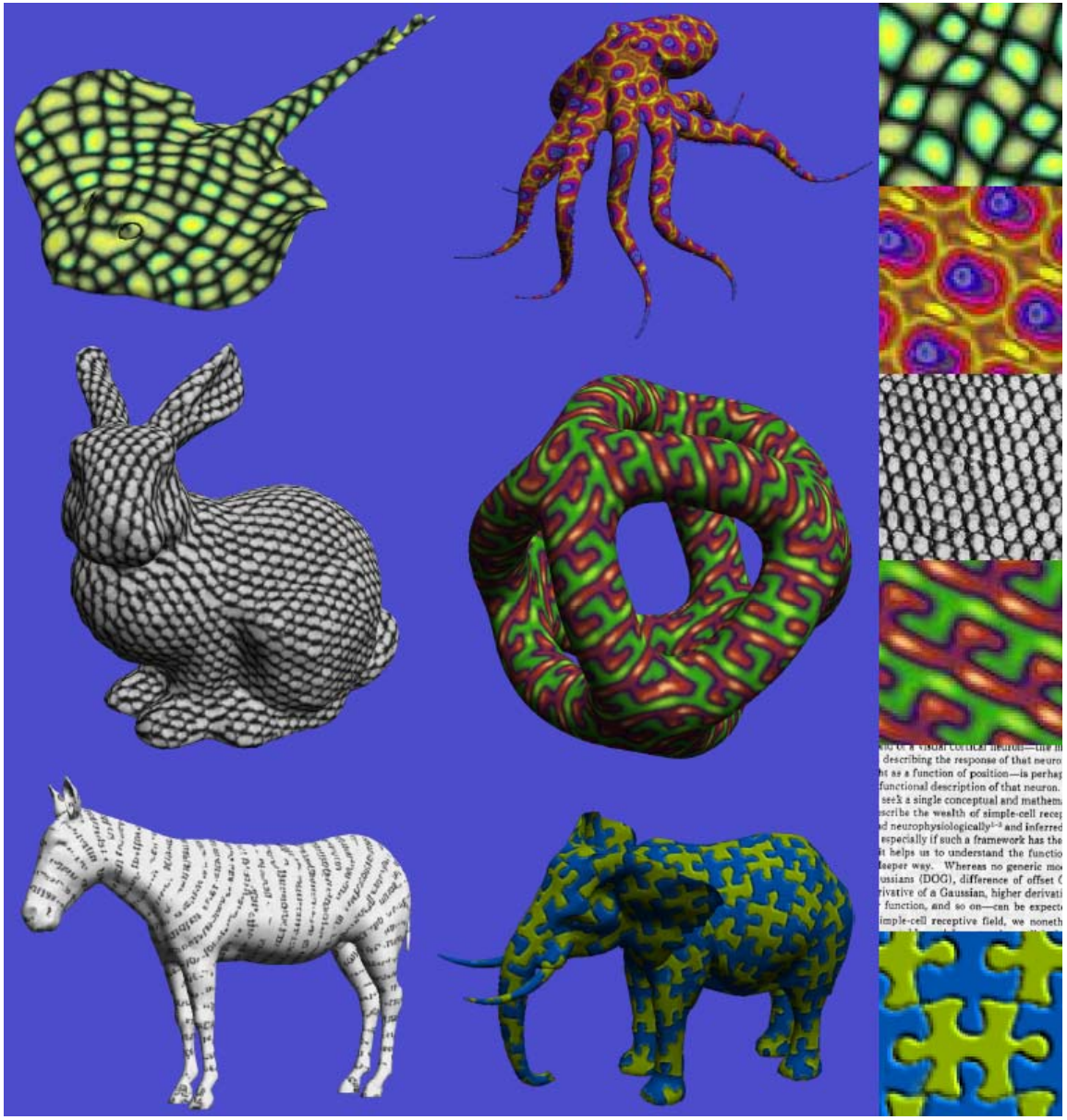
[Wei&Levoy, Siggraph 2001]



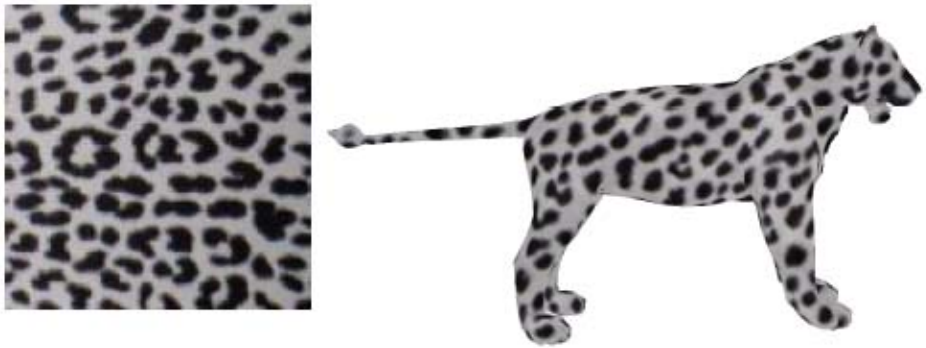
# Mesh Pyramid by Retiling

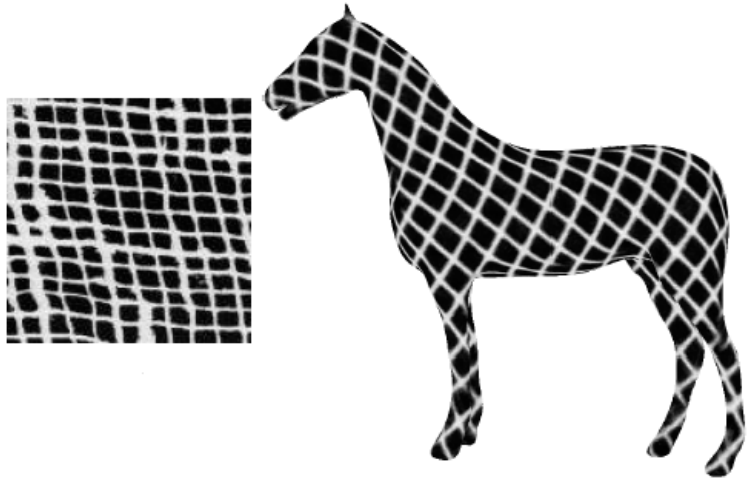


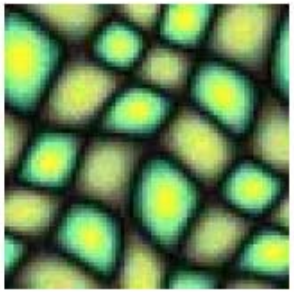
# Examples

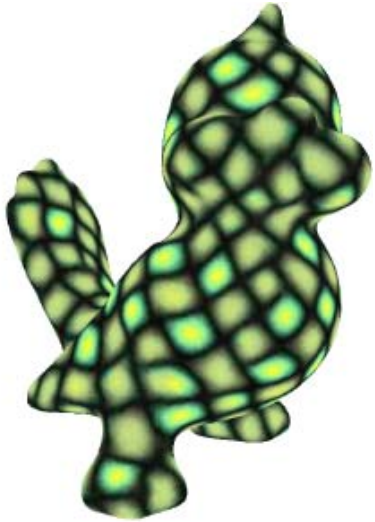


# Results





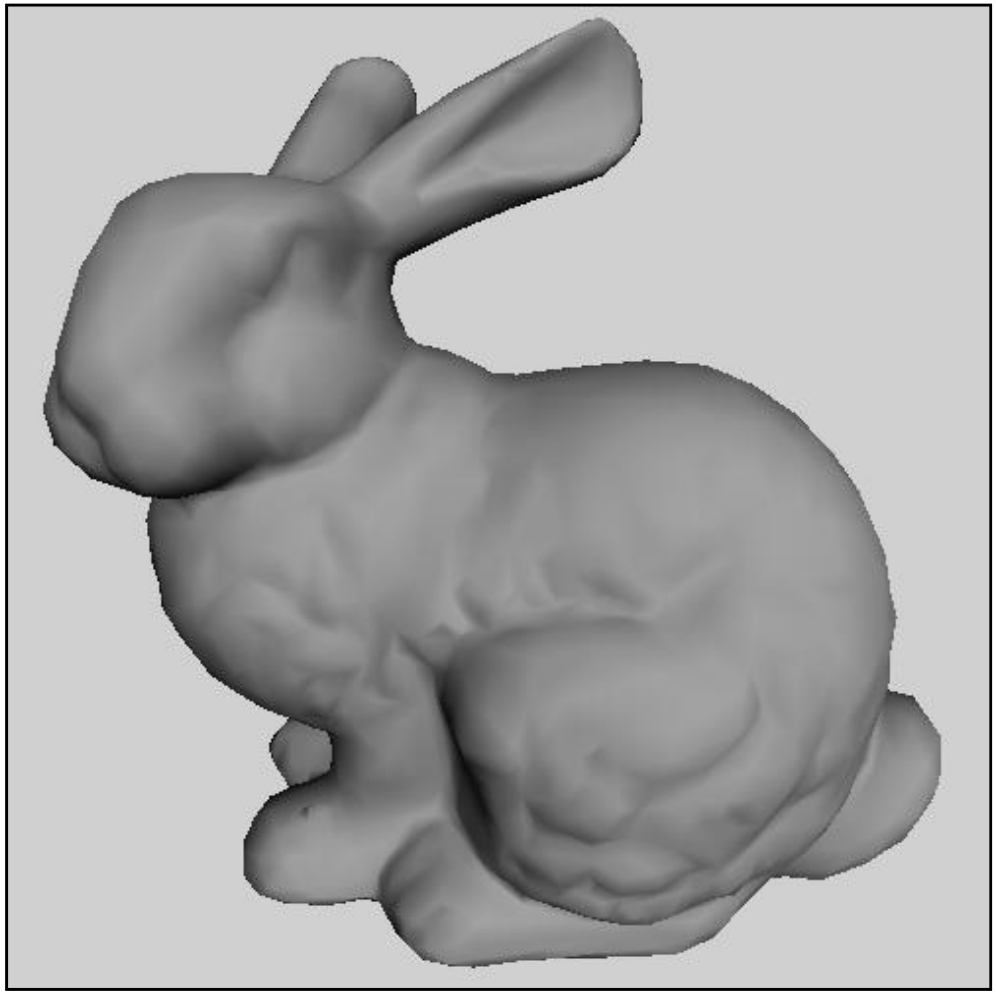




# Patch-based Synthesis

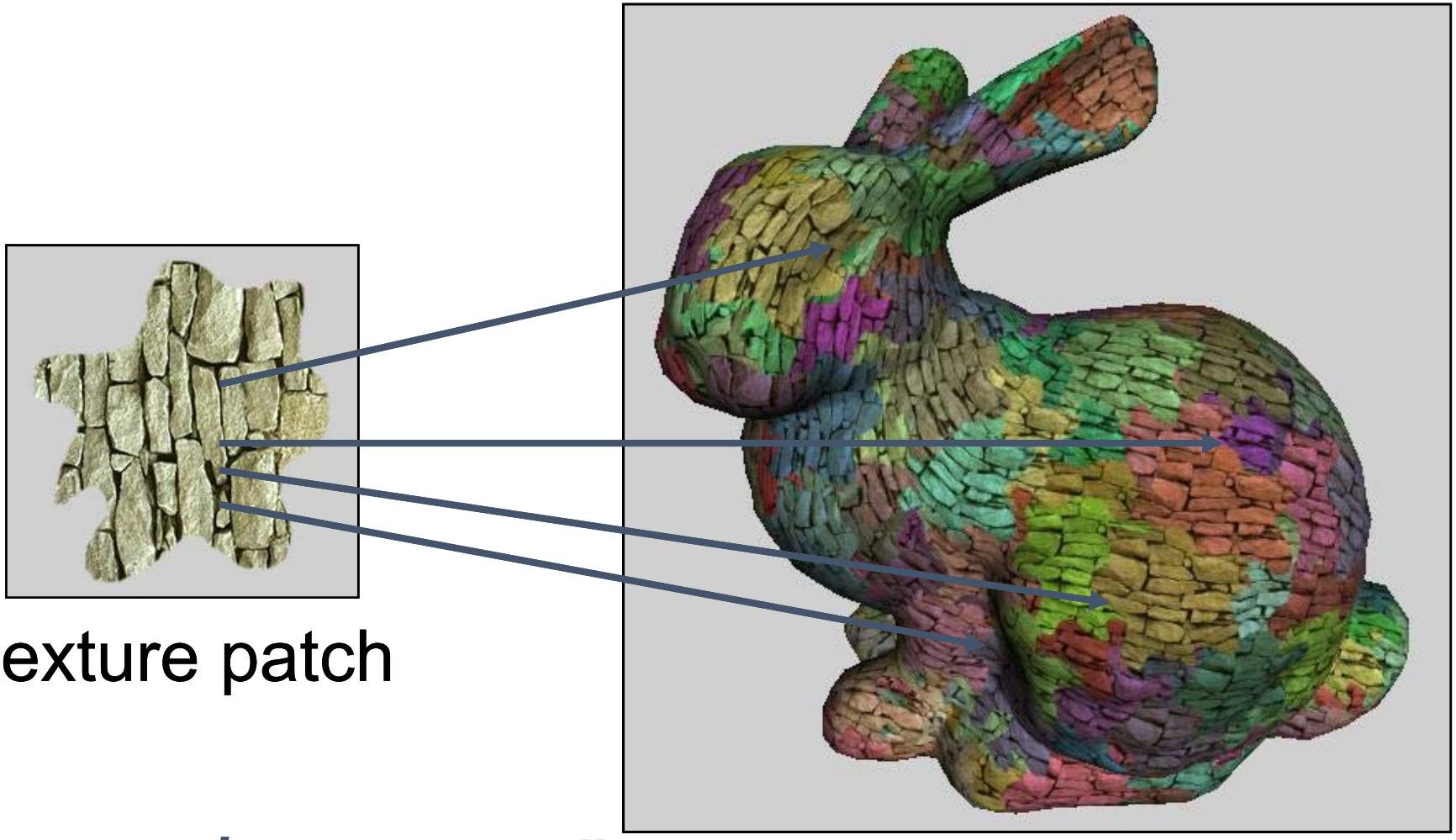
[Praun et al., Siggraph 2001]

texture patch



surface

# Key Idea: Patch Pasting



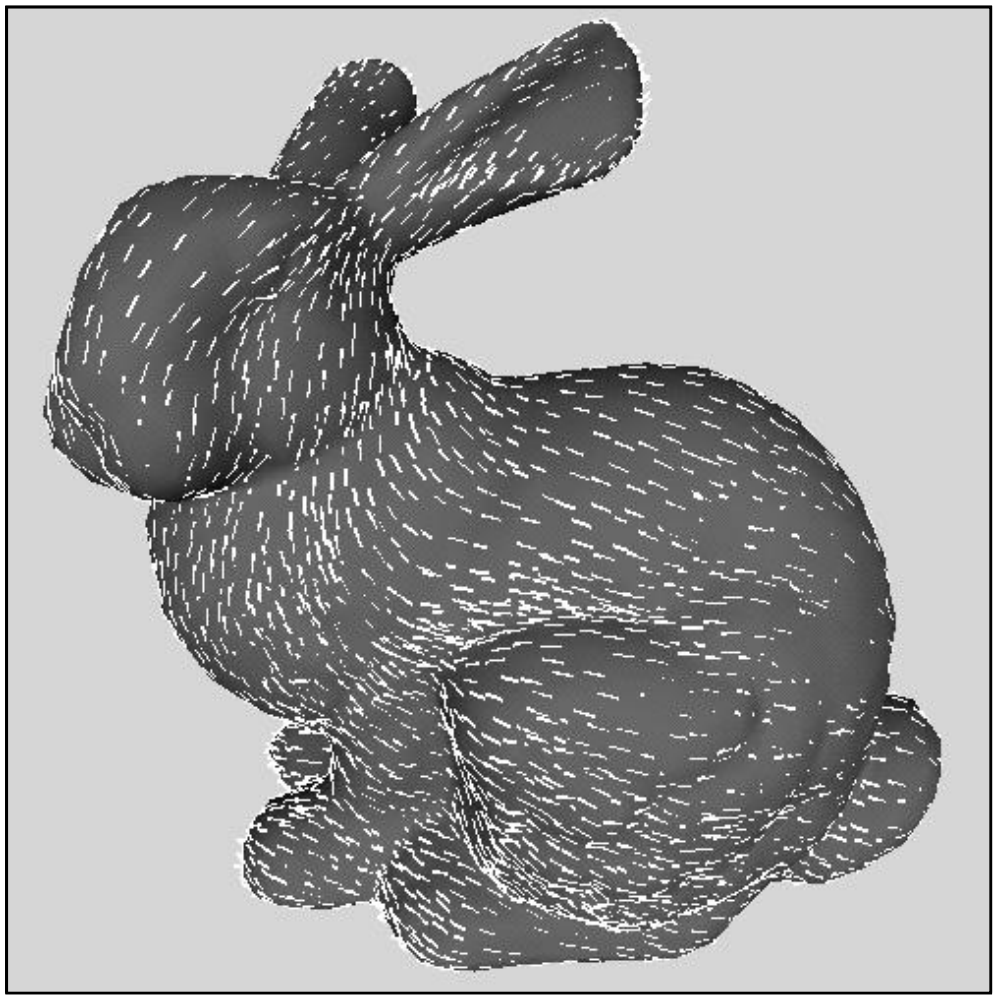
texture patch

surface

*"lapped textures"*

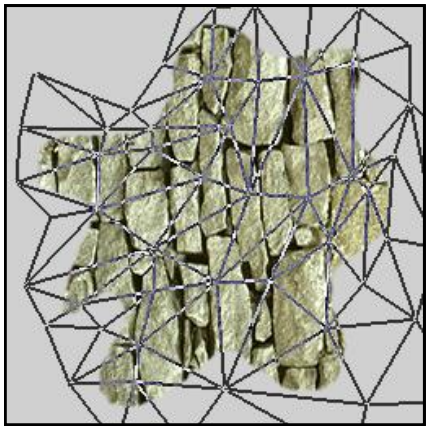
# Algorithm

texture patch

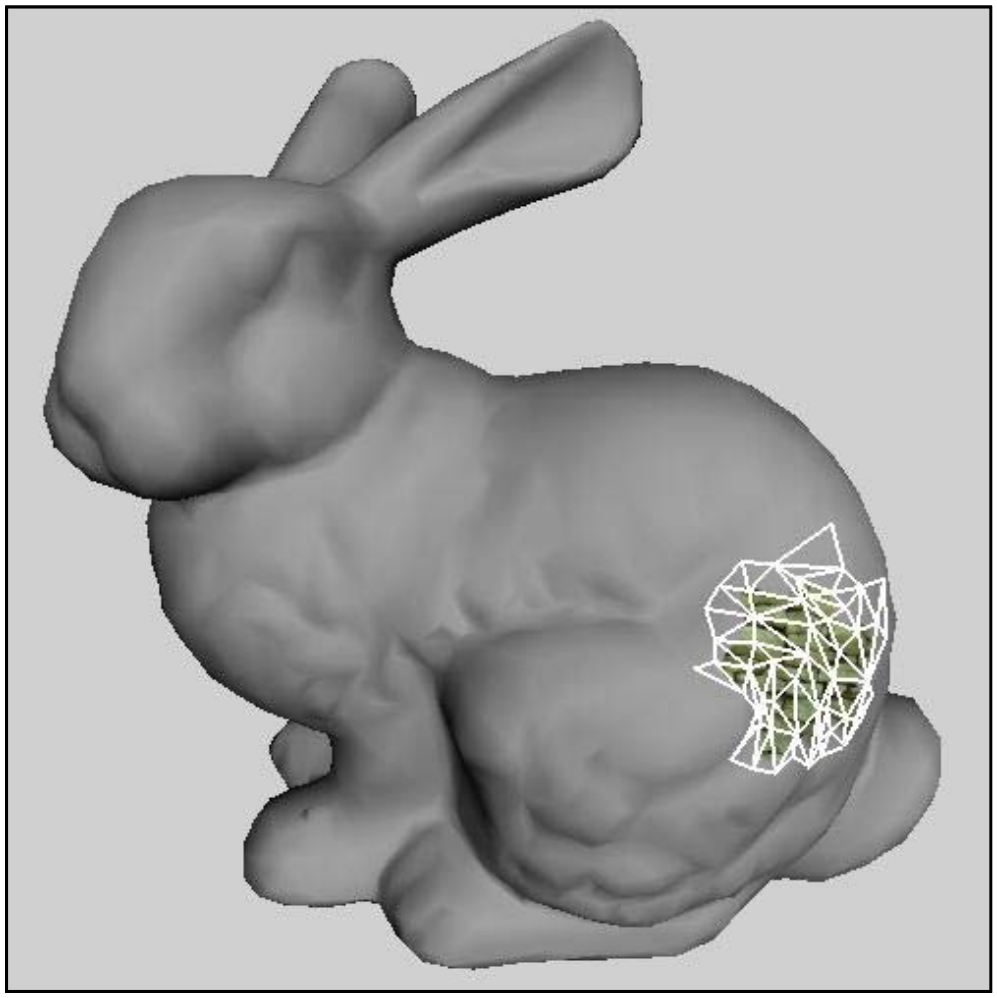


surface

# Algorithm

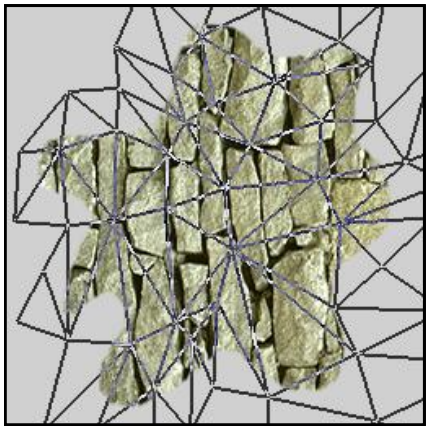


texture patch

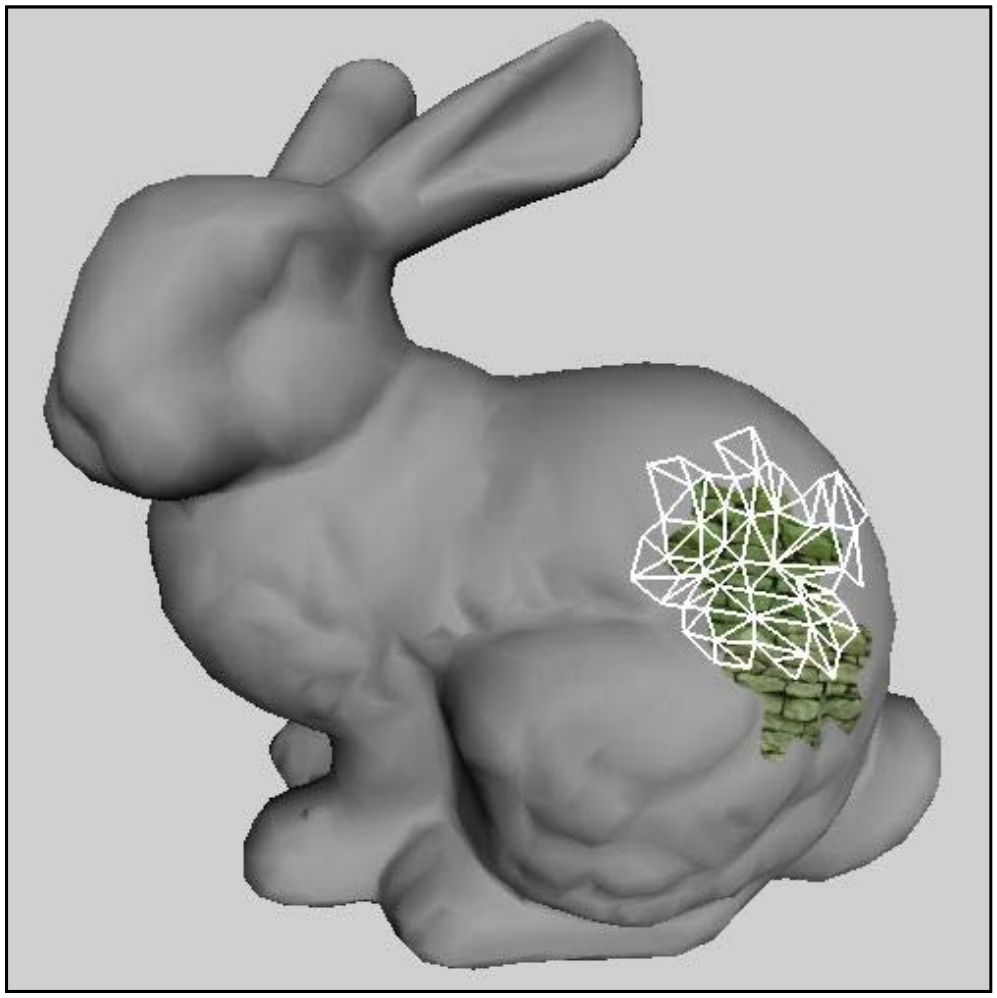


surface

# Algorithm



texture patch



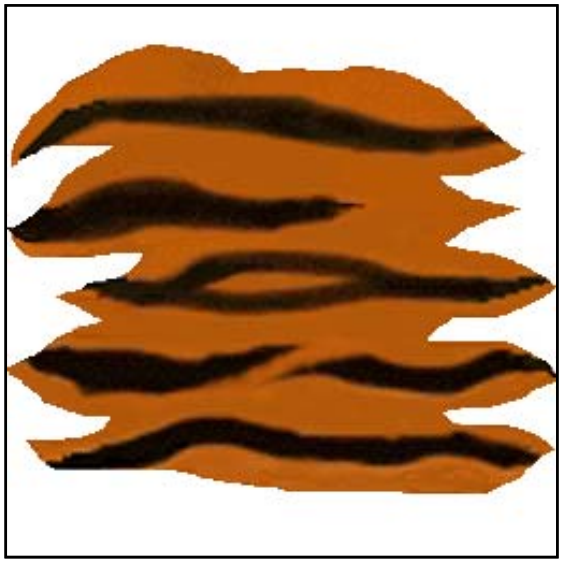
surface

# Algorithm

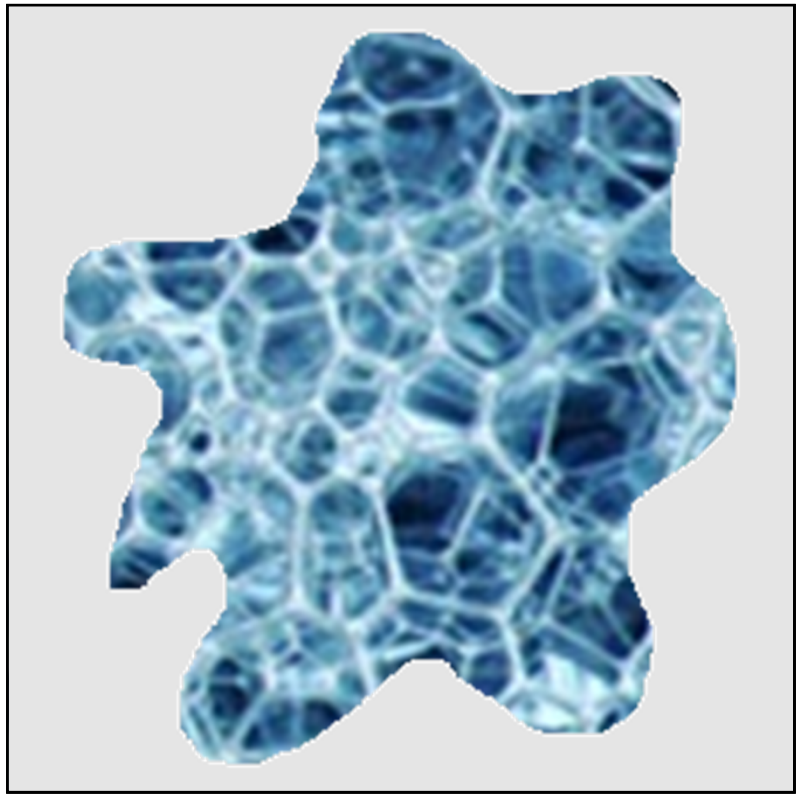
texture patch

surface

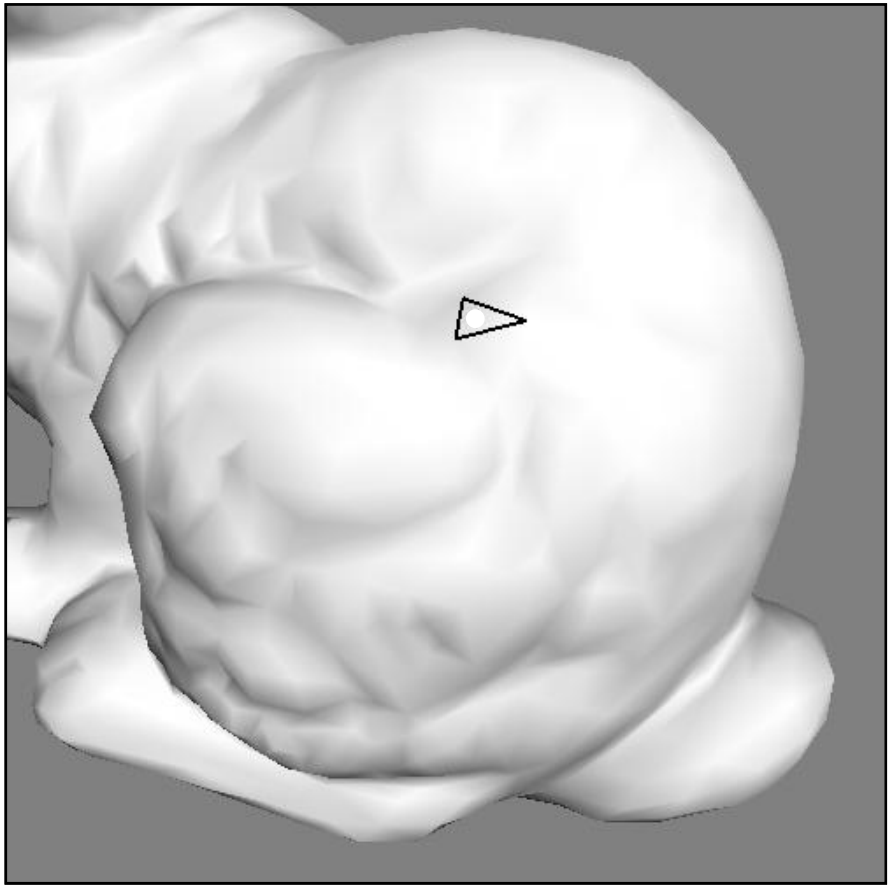
# Texture Patch Creation

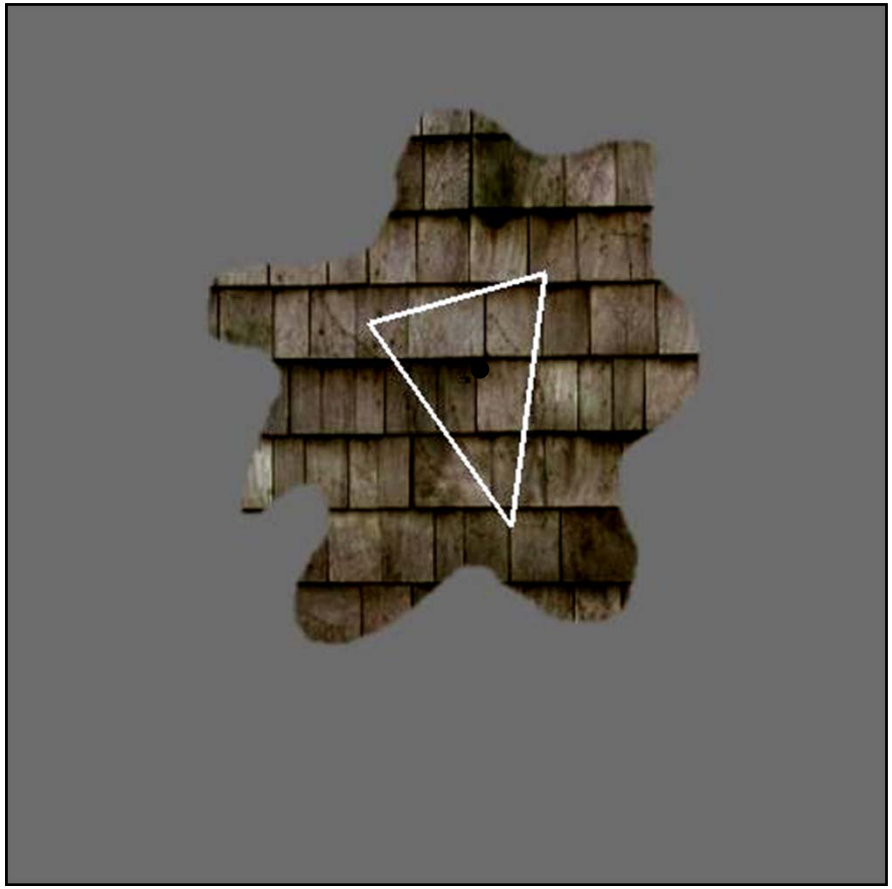


# Less Structure → Splotch

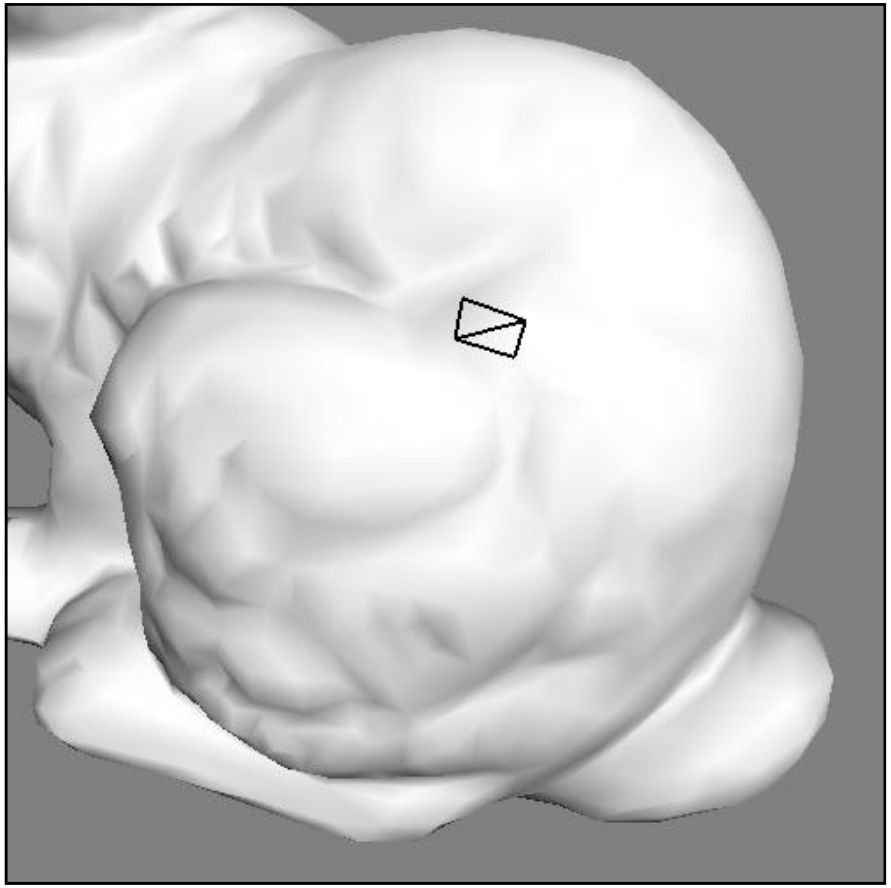


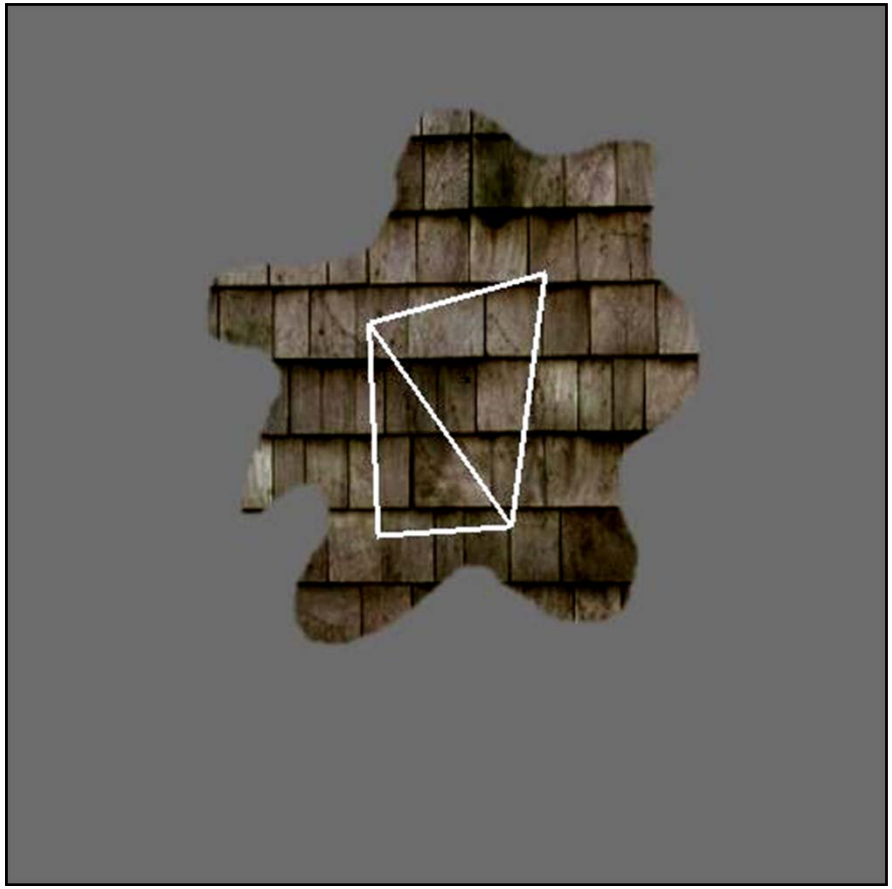
# Patch Growth



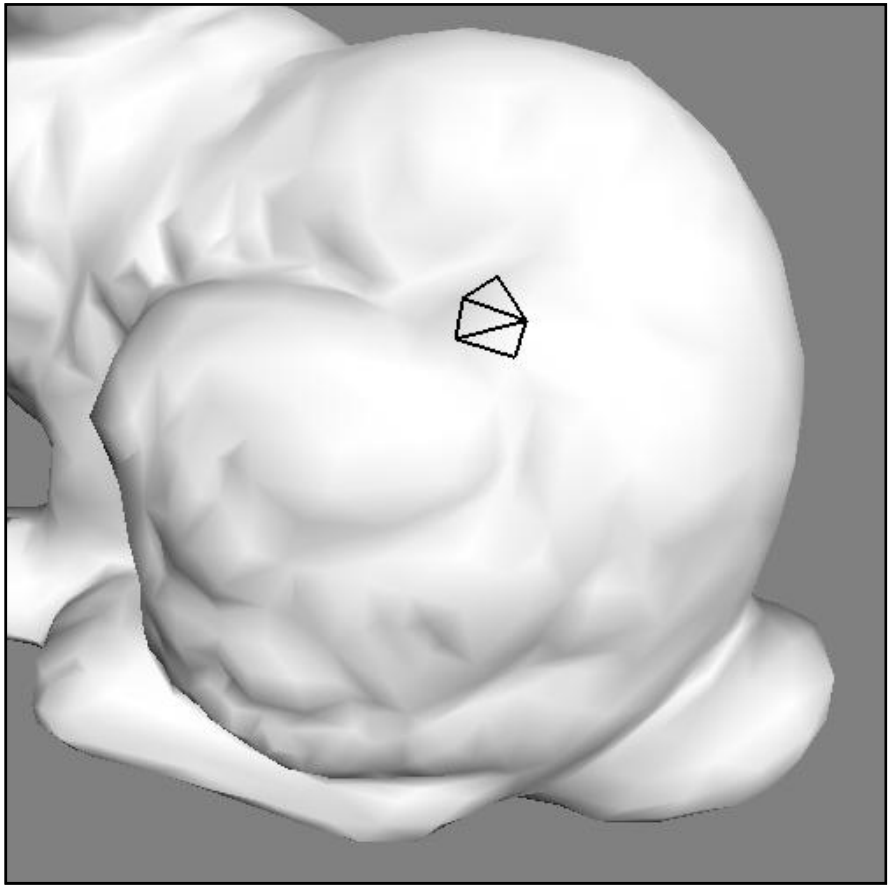


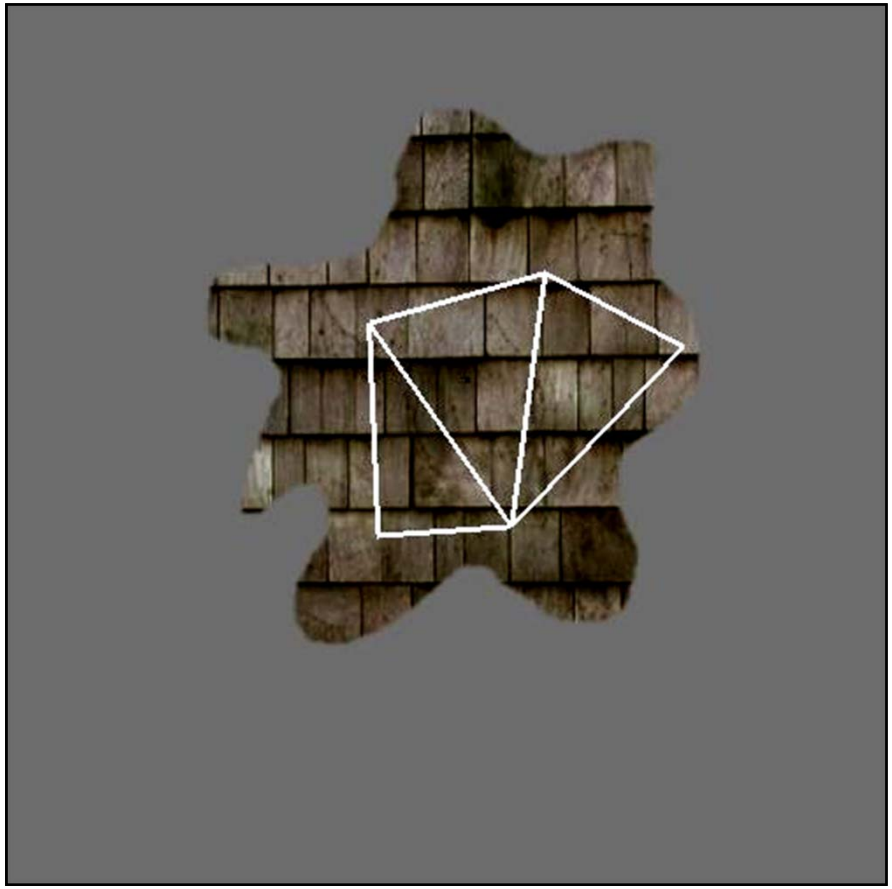
# Patch Growth



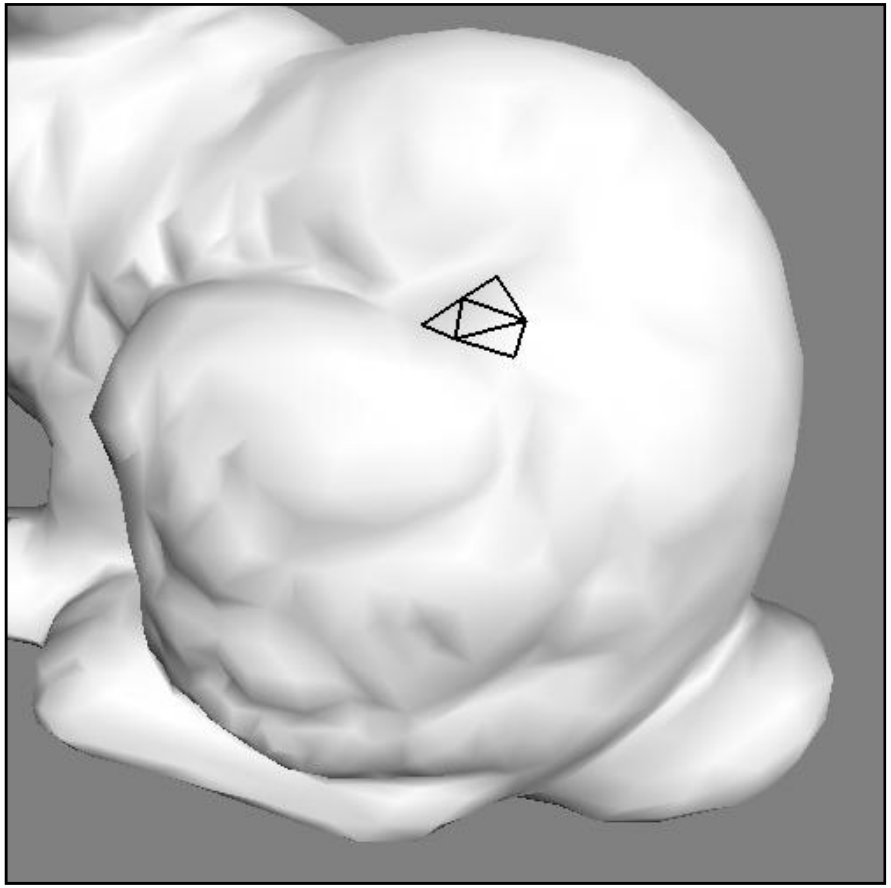


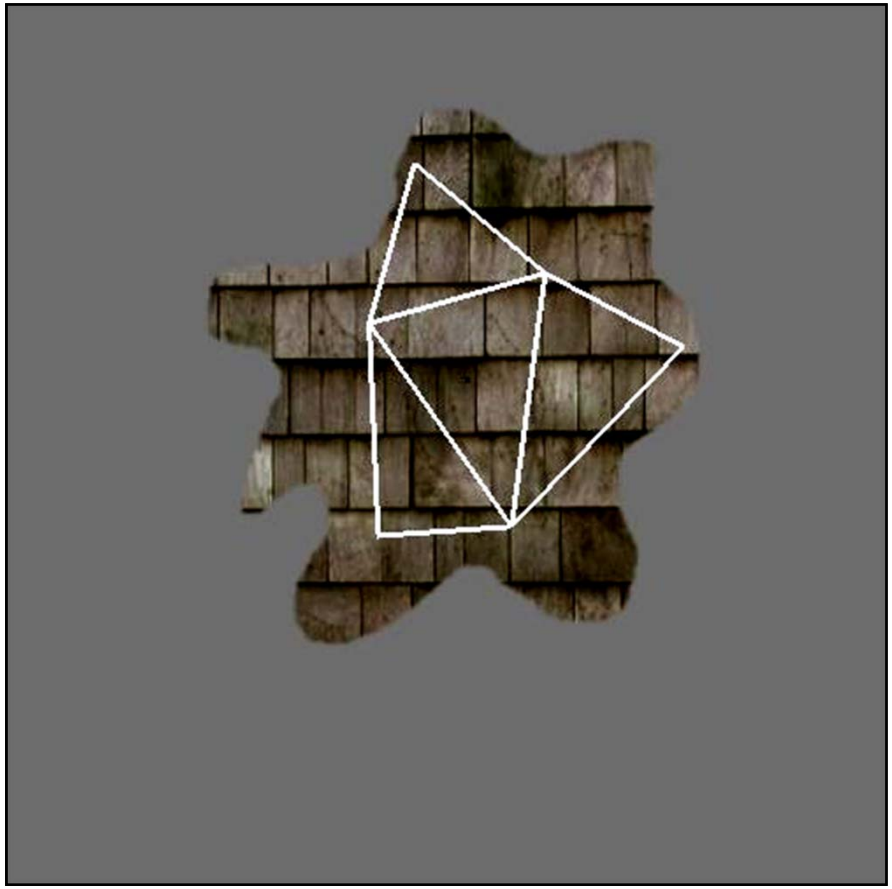
# Patch Growth



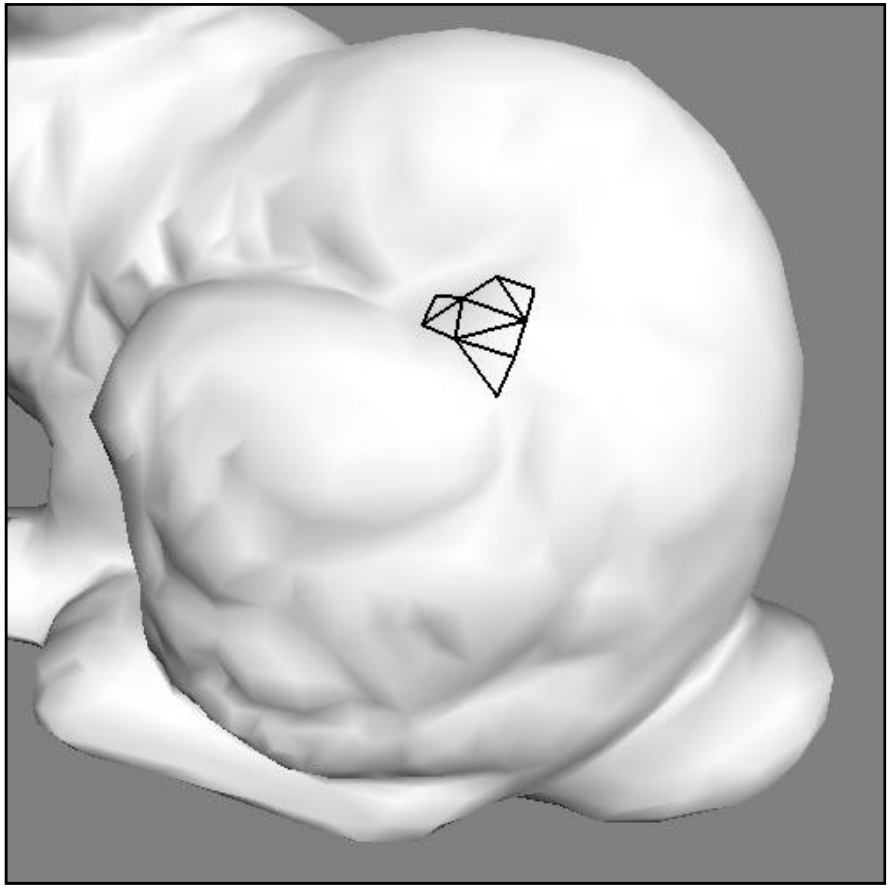


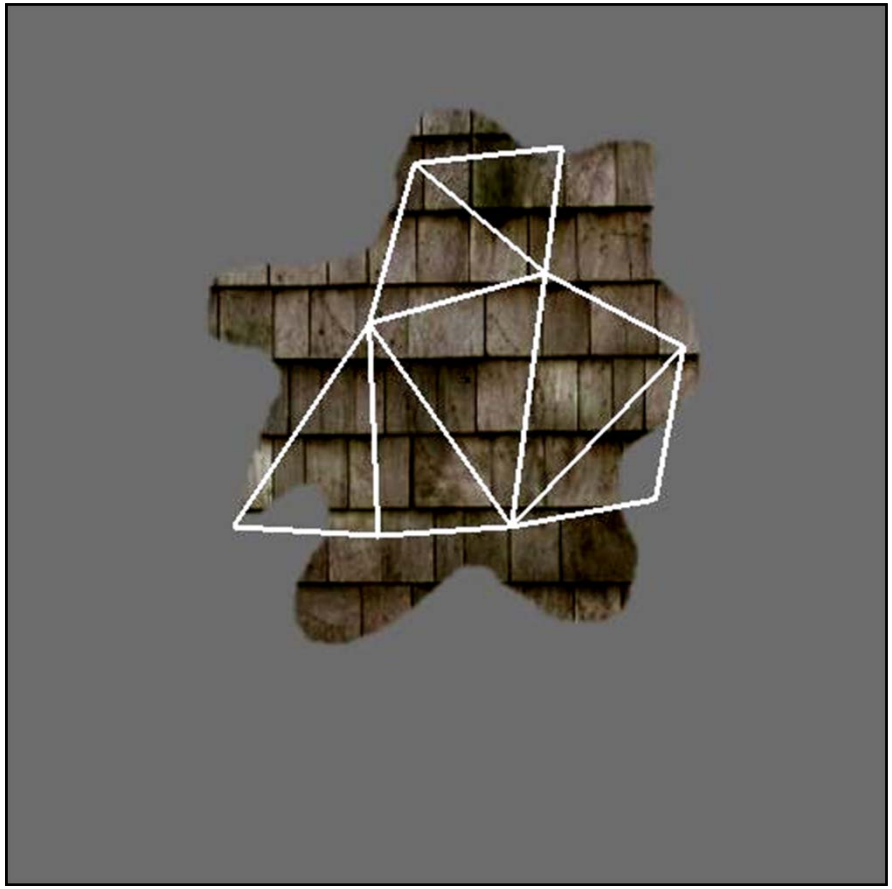
# Patch Growth



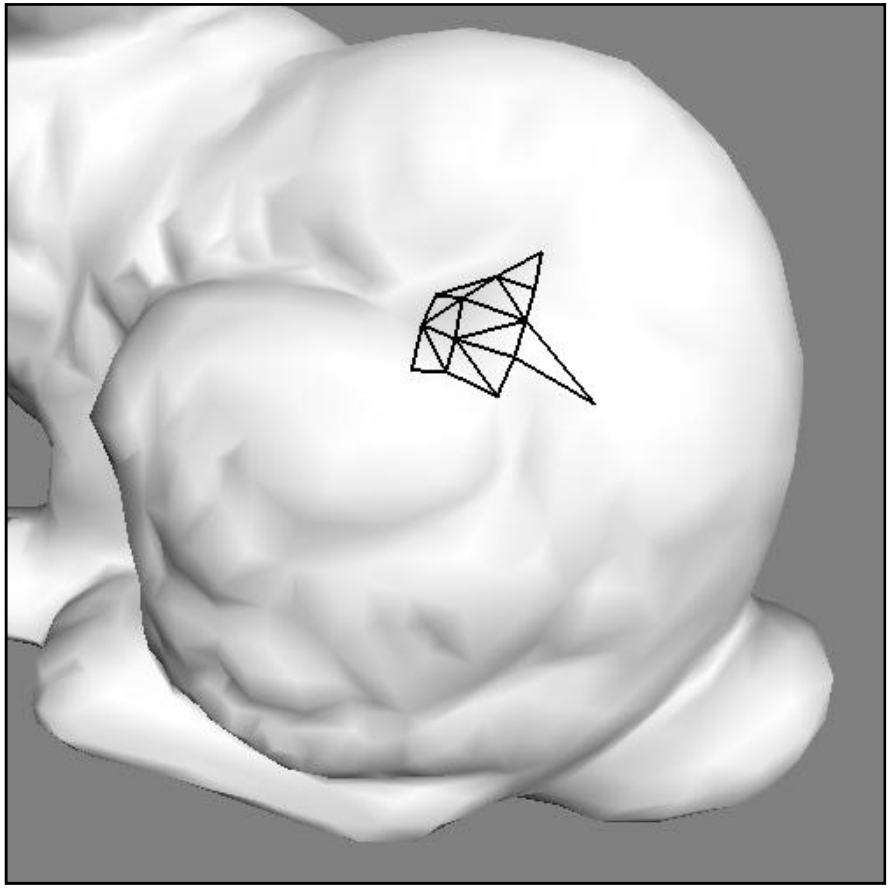


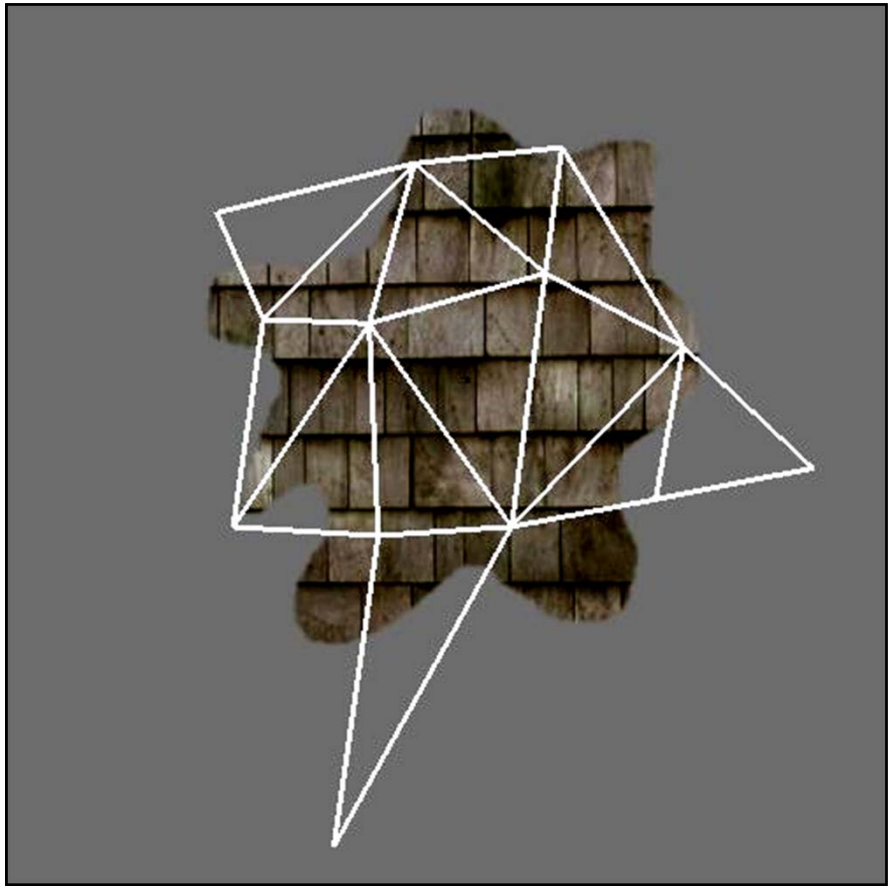
# Patch Growth



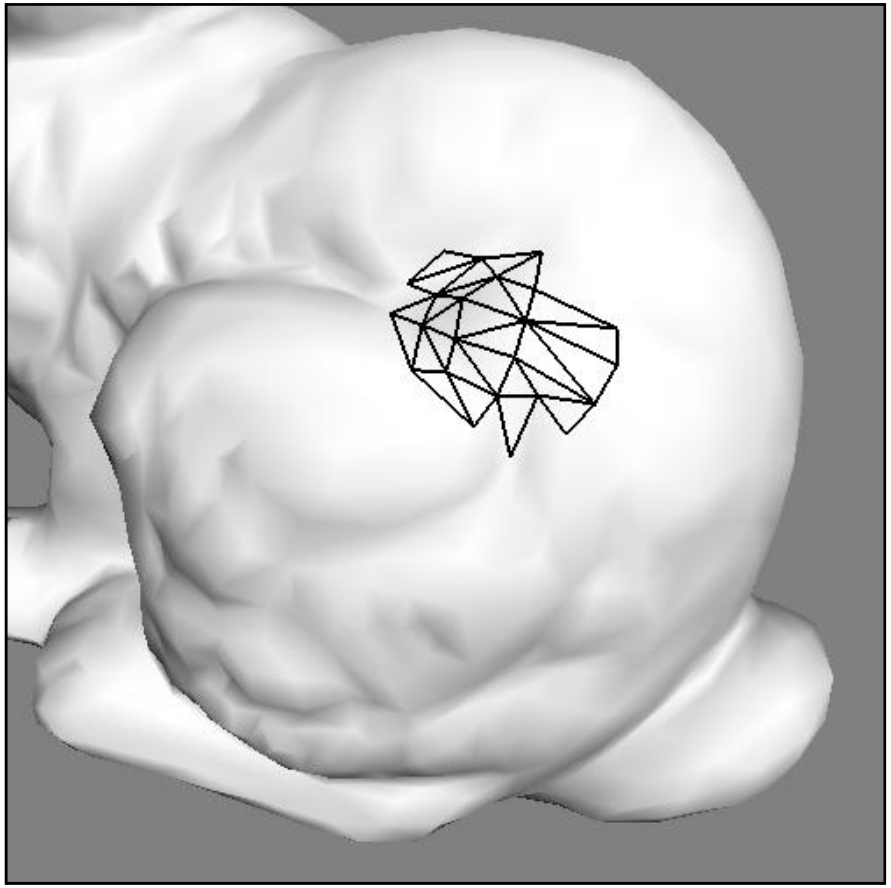


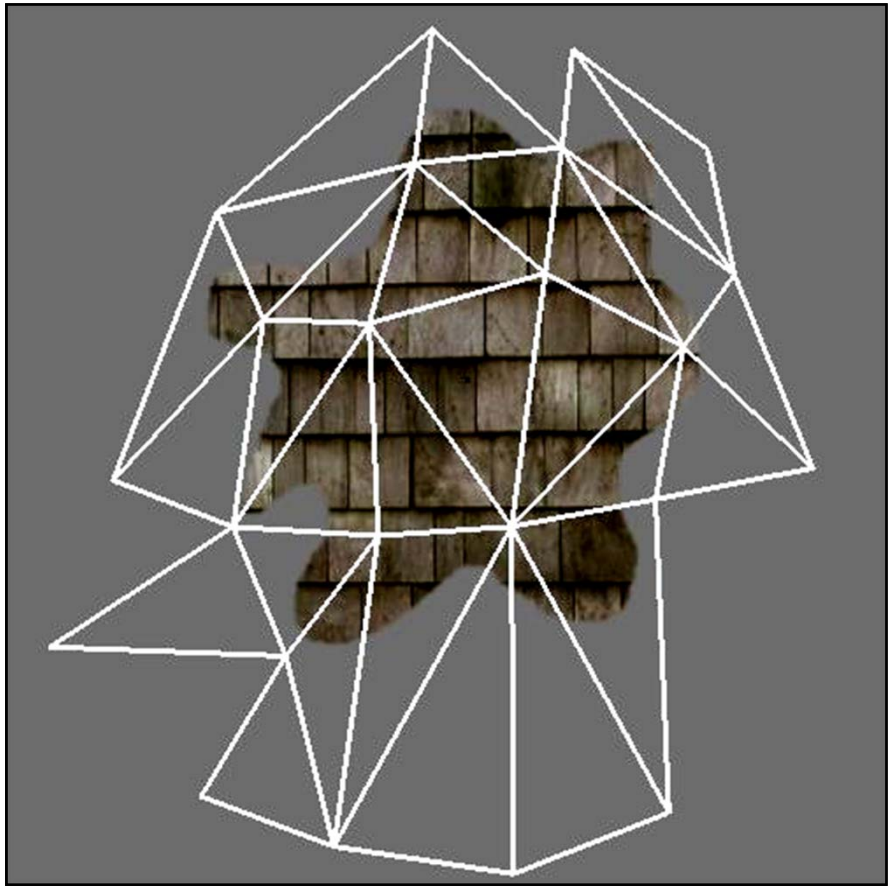
# Patch Growth



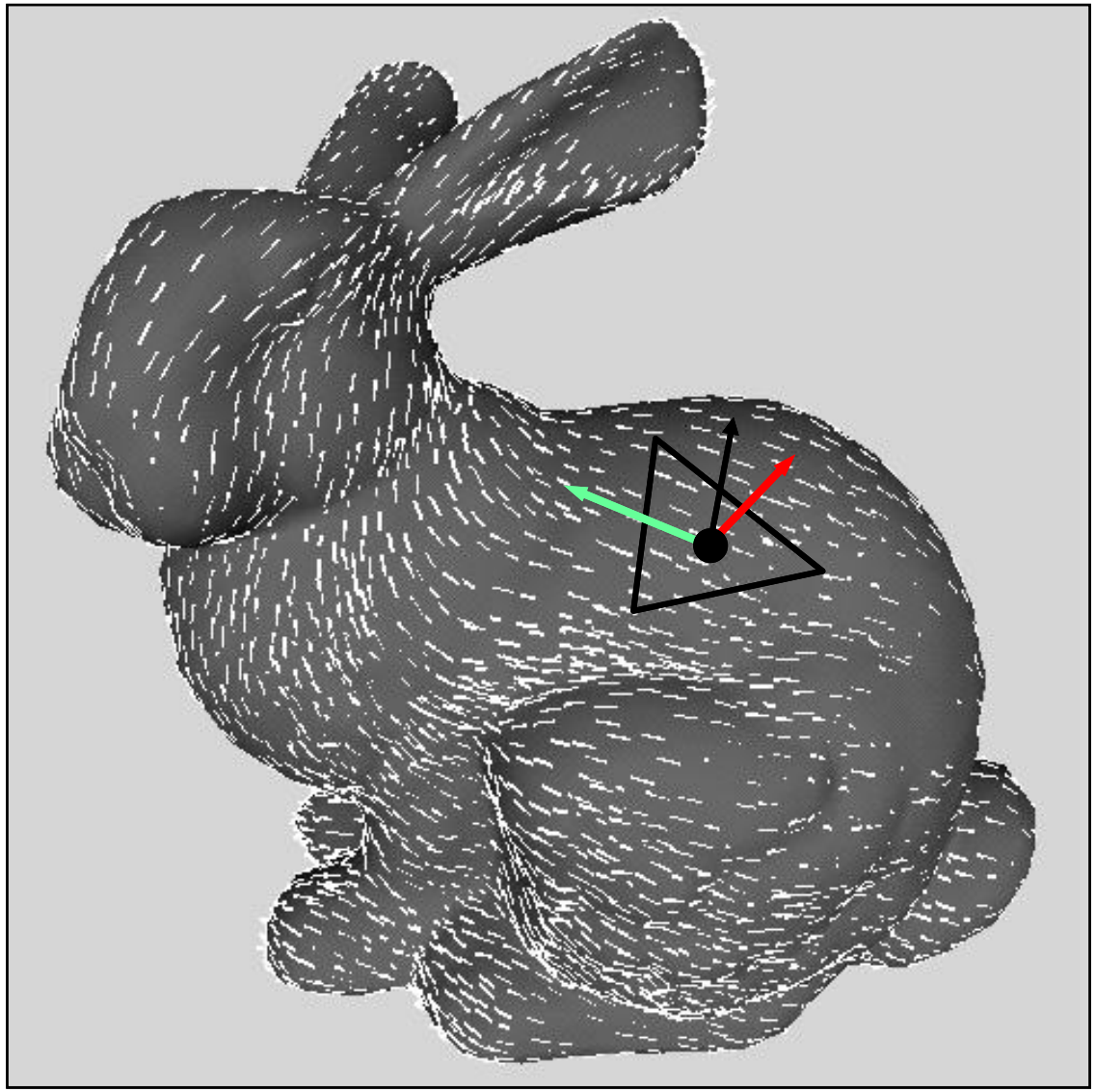


# Patch Growth

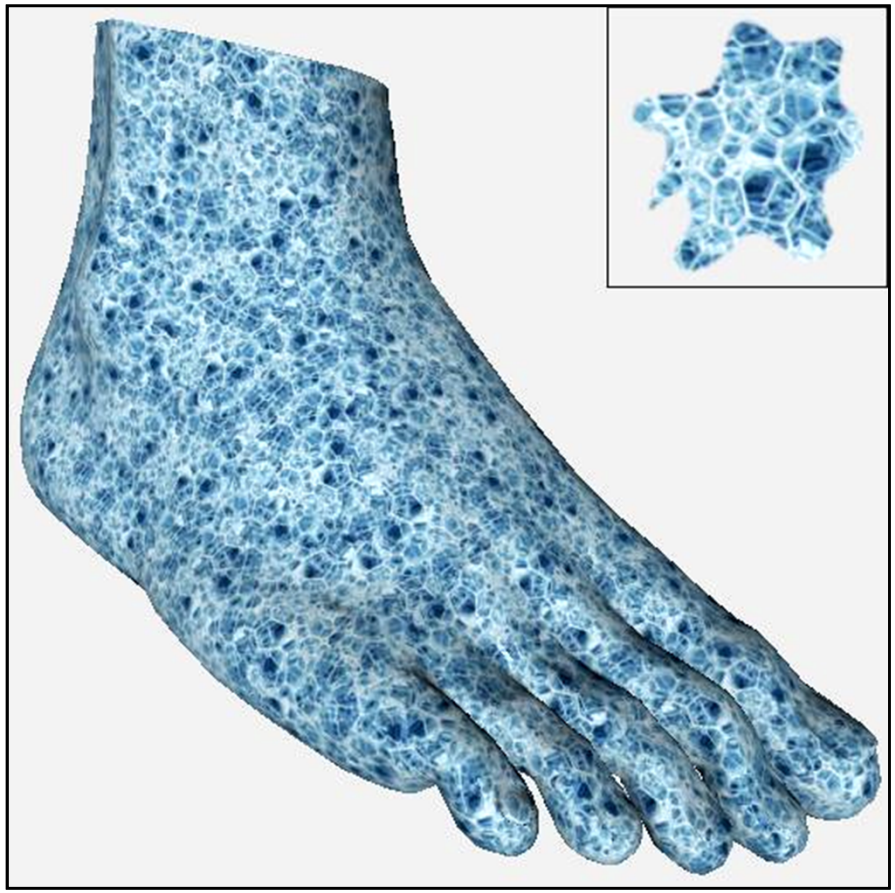


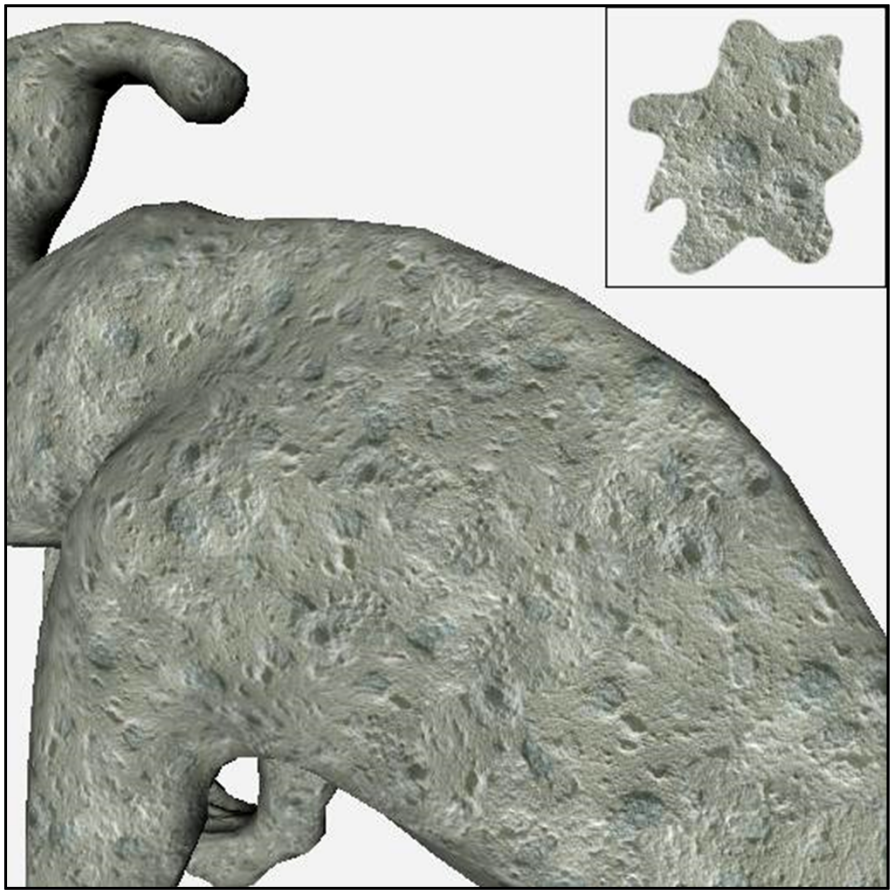


# Tangential Vector Field



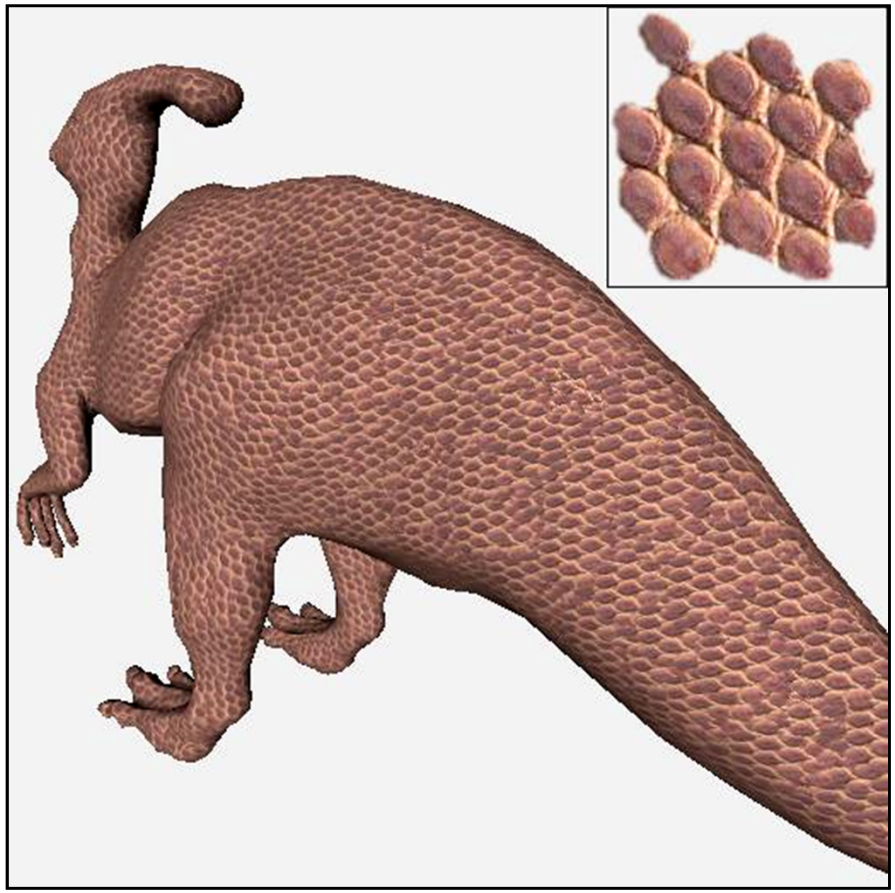
# Results: Splotches

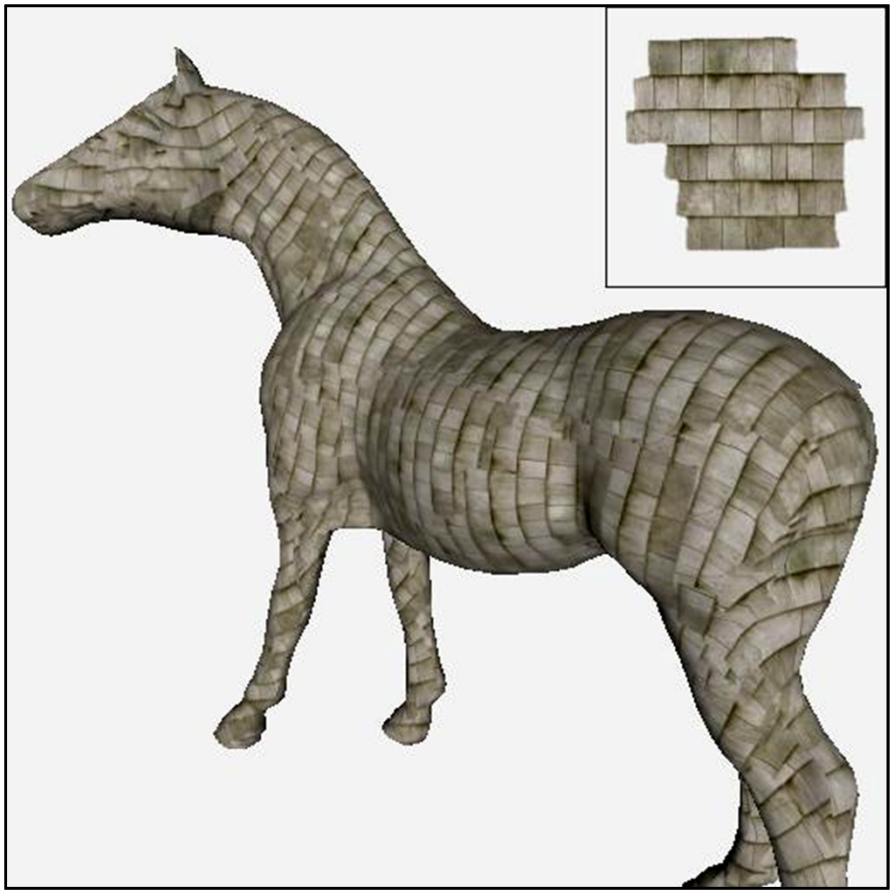




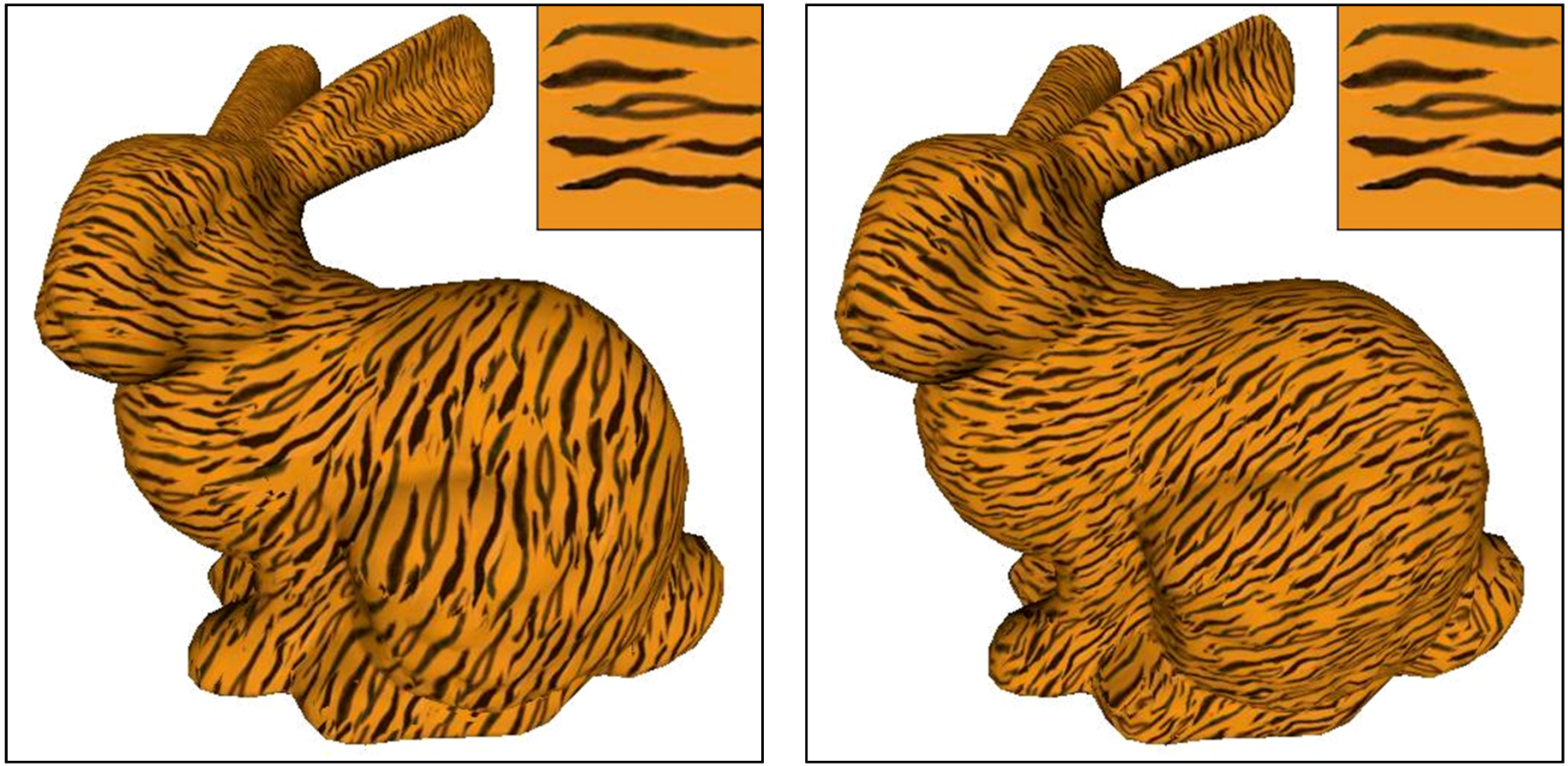
*(completely automatic: no direction field)*

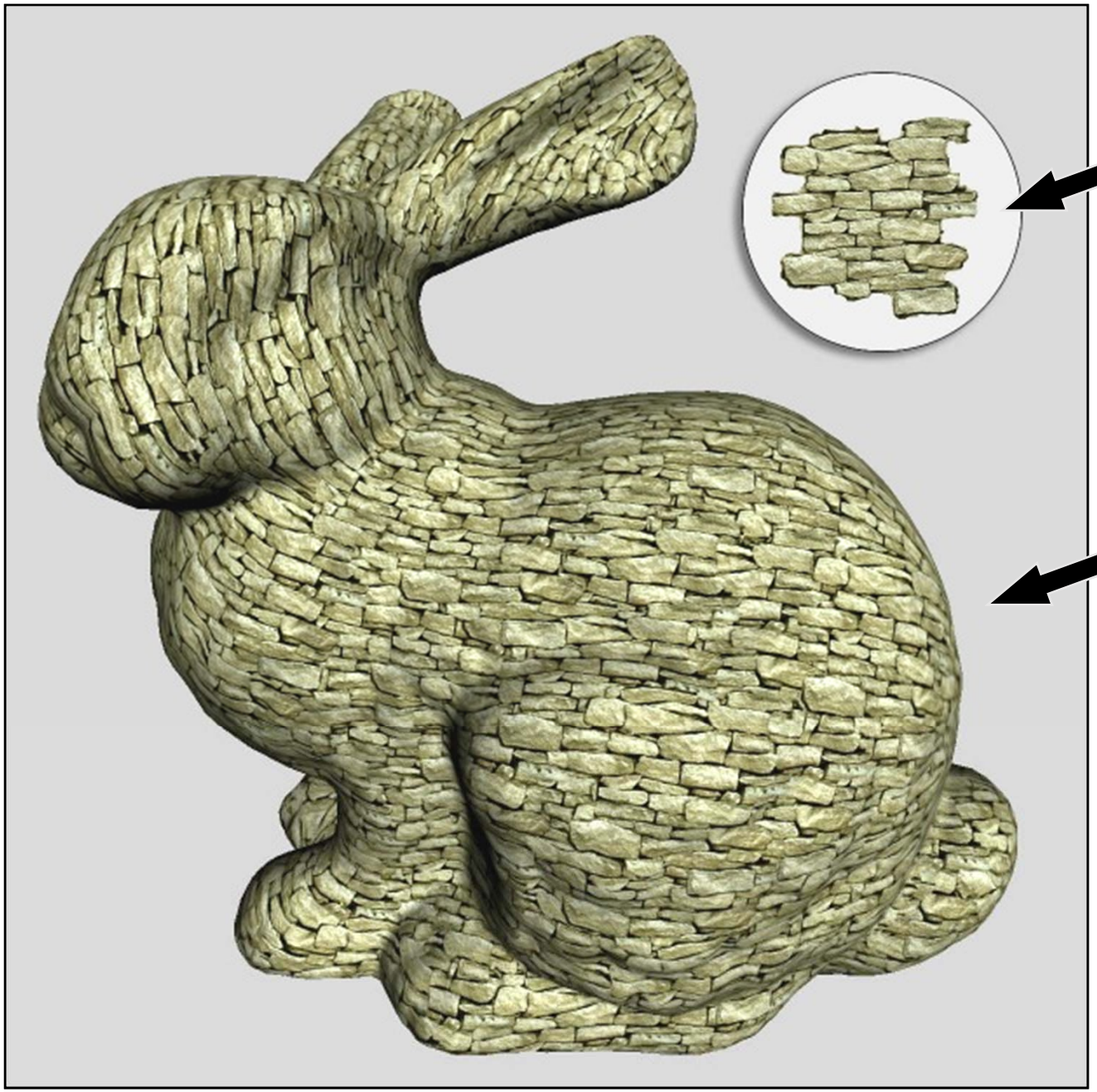
# Results: Anisotropic





# Controlling Direction and Scale





256 x 256
texture

(282 times)

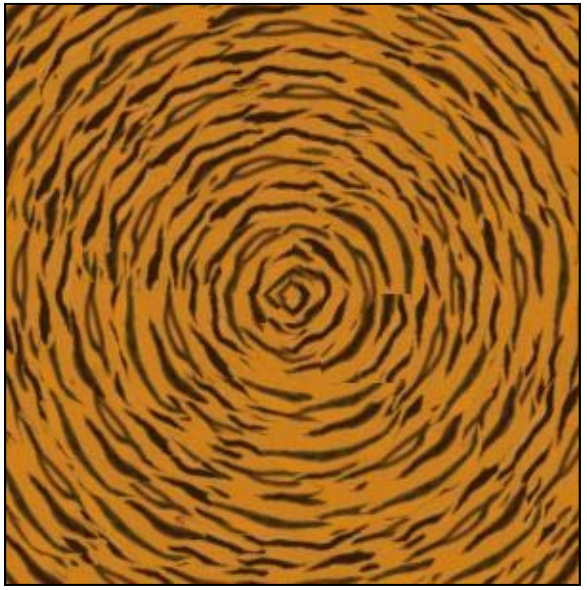
15,000 faces

25 frames
per sec!

# Limitations

low-frequency components

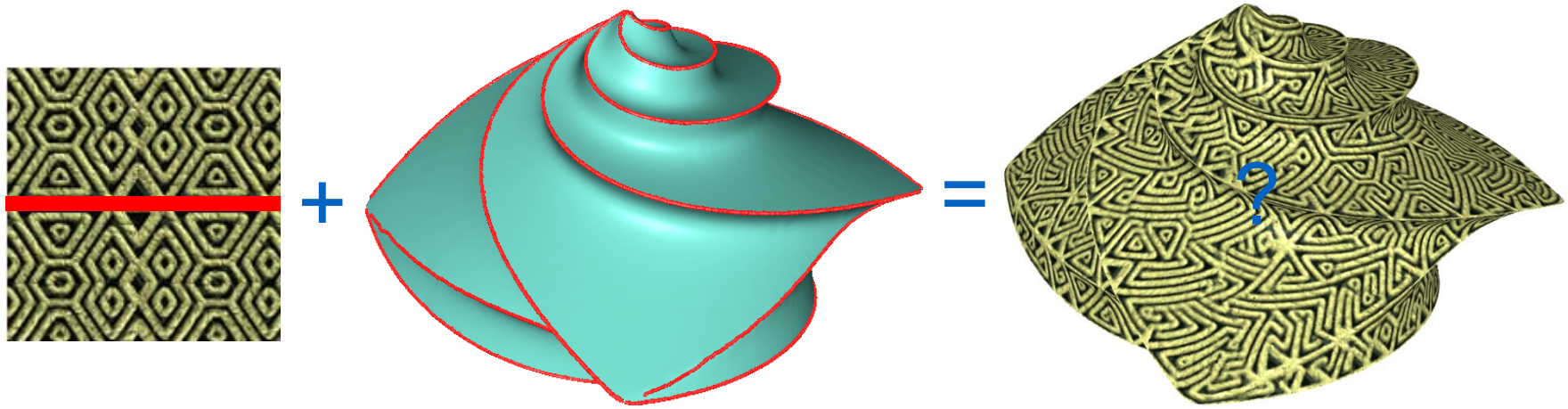
boundary mismatches



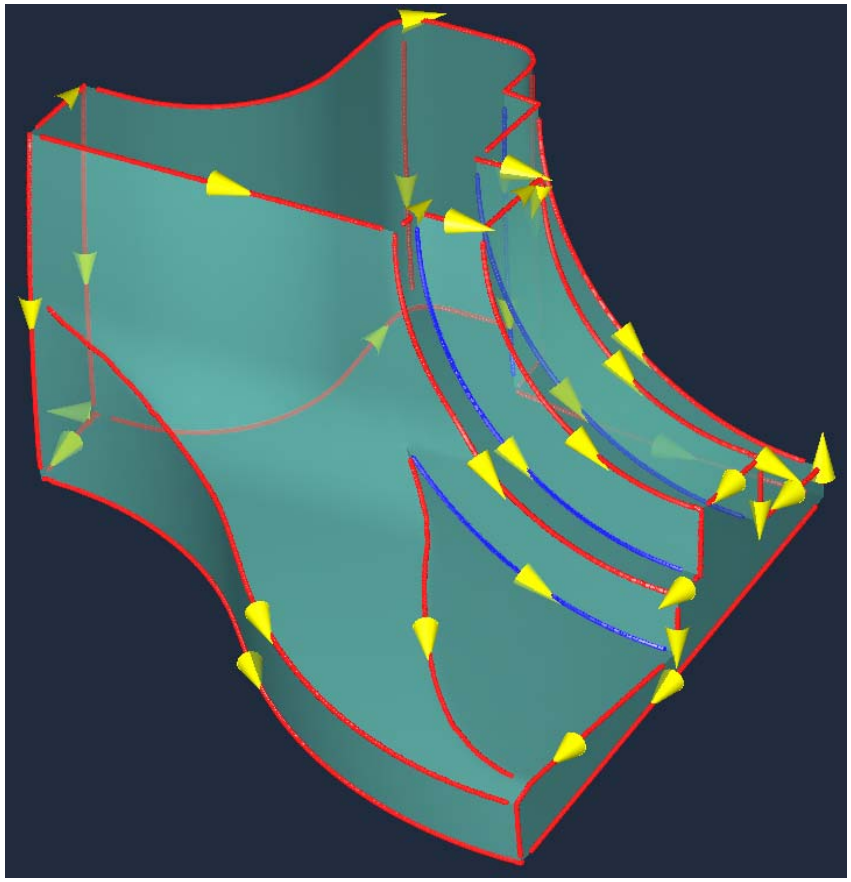
direction field singularities

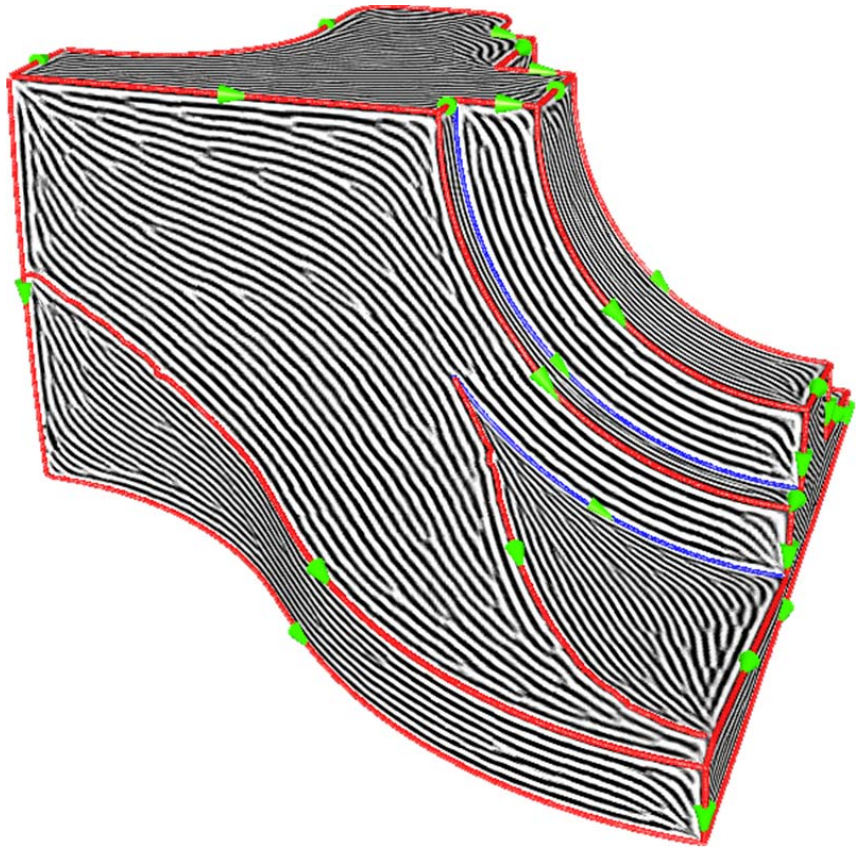
# Feature-aligned Texture Synthesis

[Xu et al., Siggraph Asia 2009]

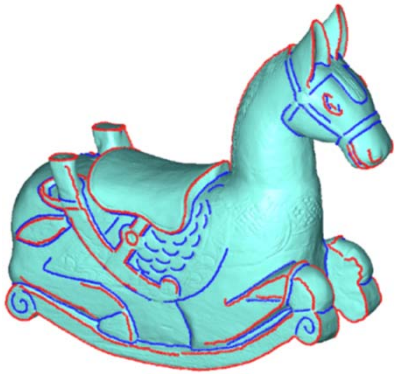


# Curve Orientation and Vector Field



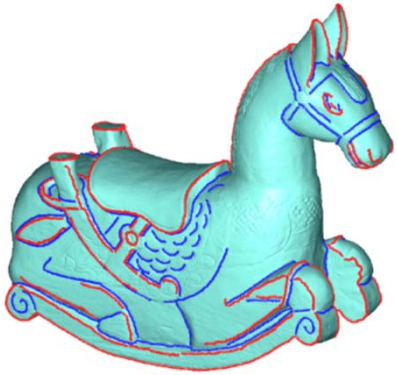


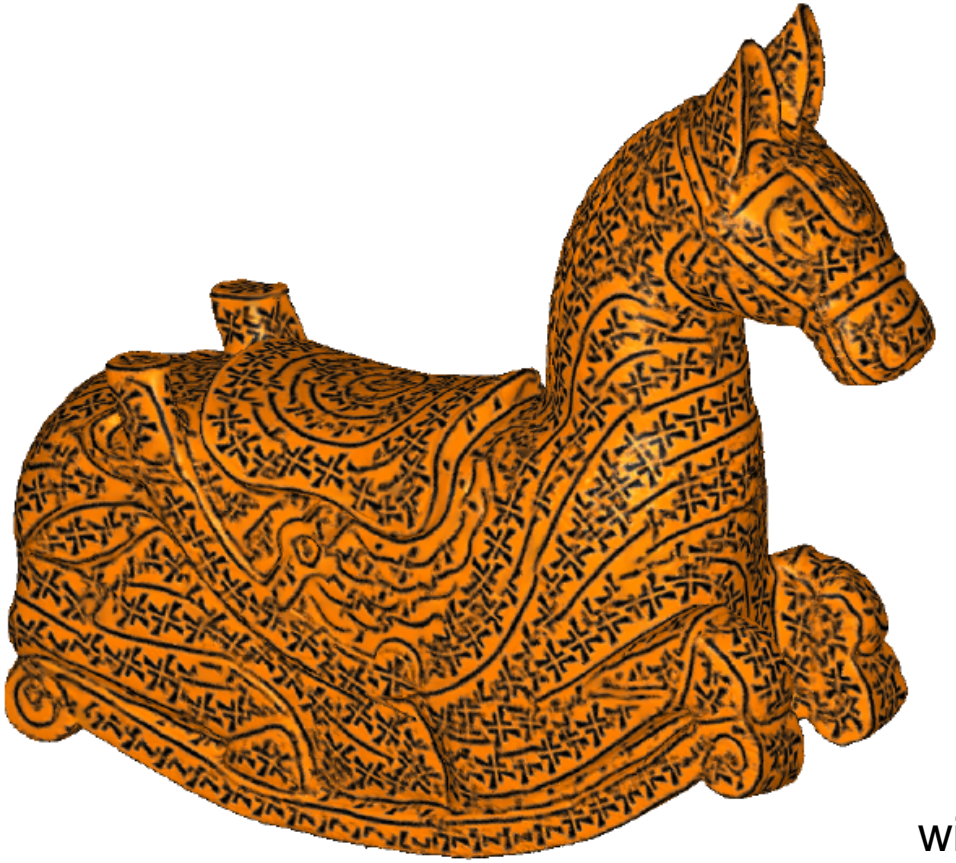
# Results



w/o alignment

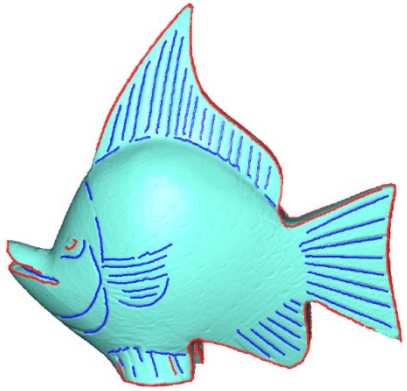
# Results

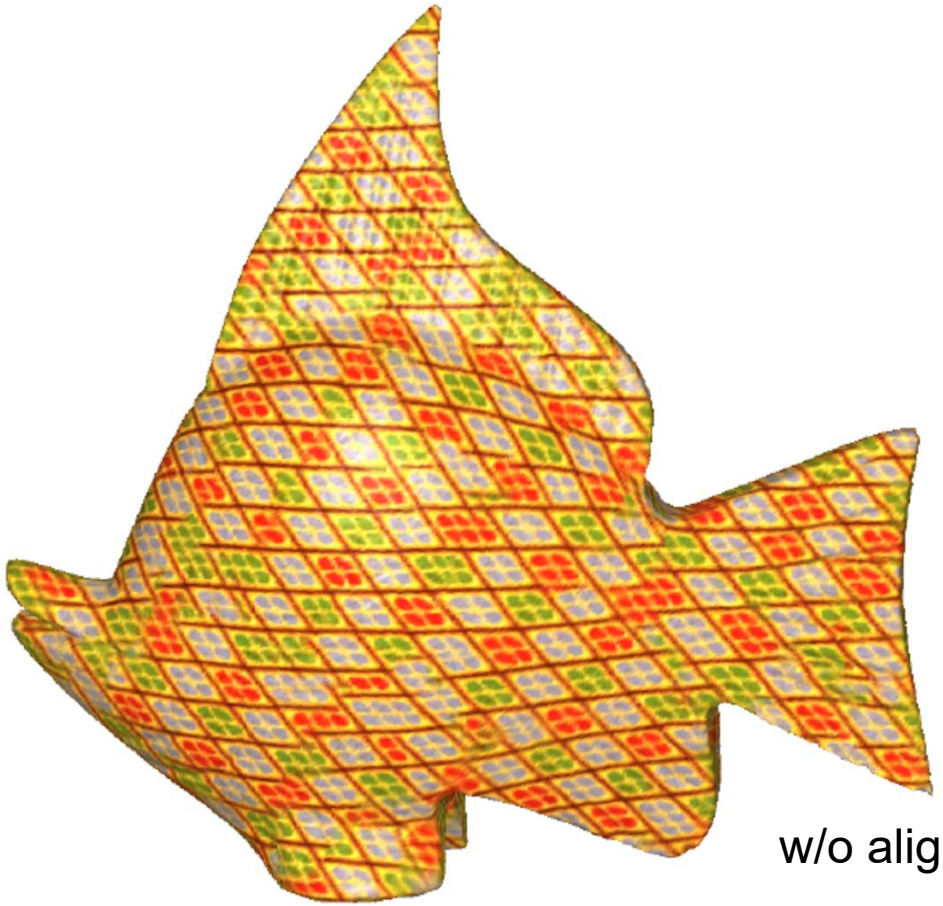




with alignment

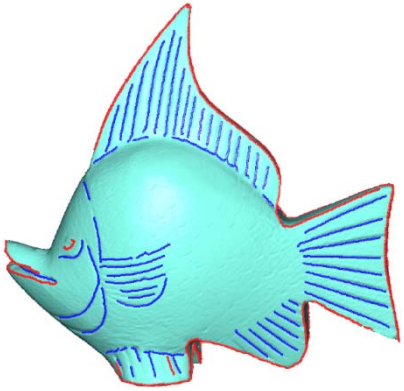
# Results

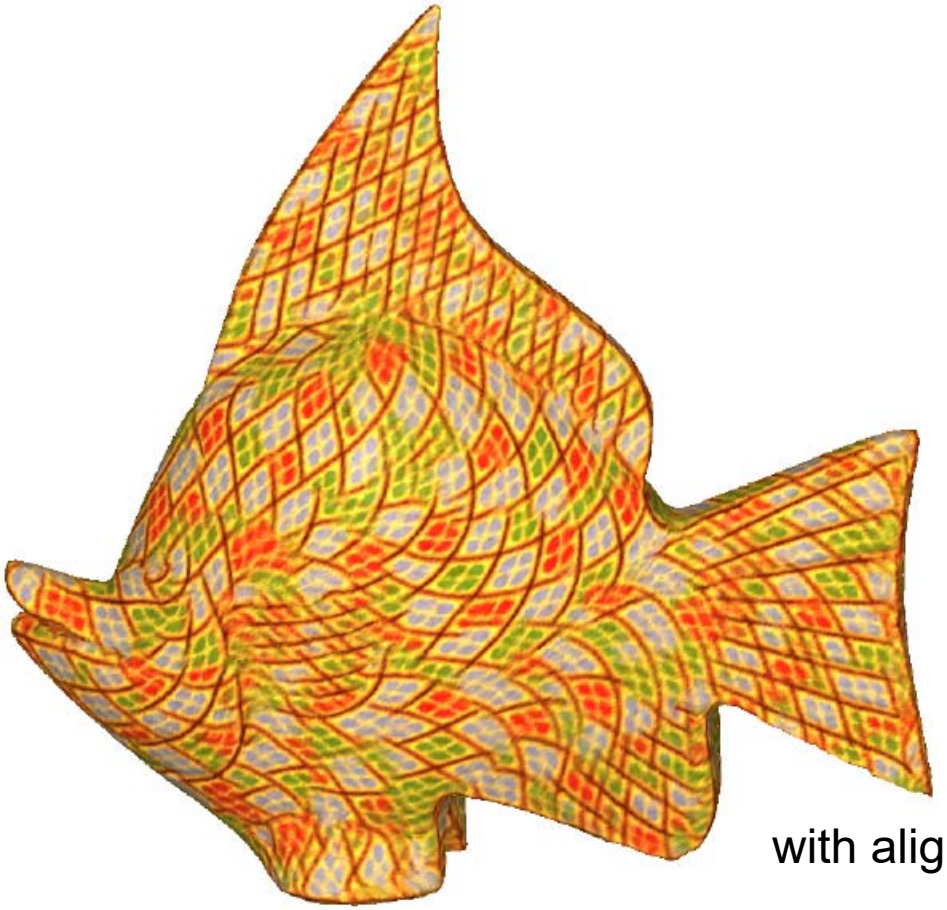




w/o alignment

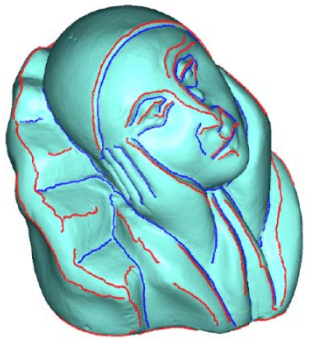
# Results





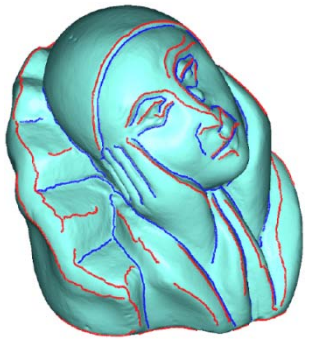
with alignment

# Results



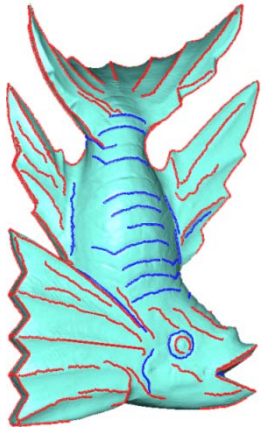
w/o alignment

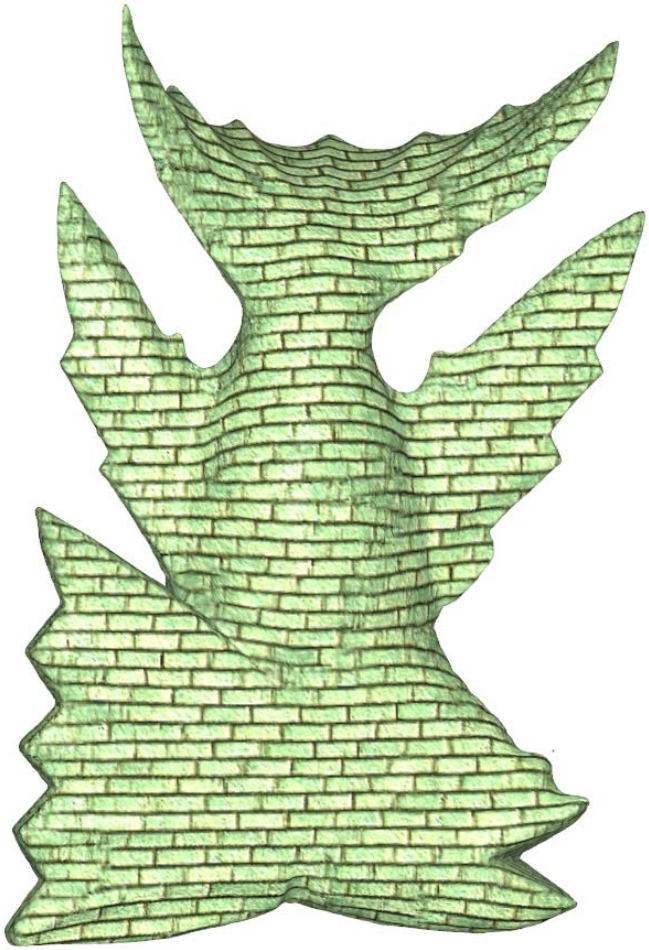
# Results



with alignment

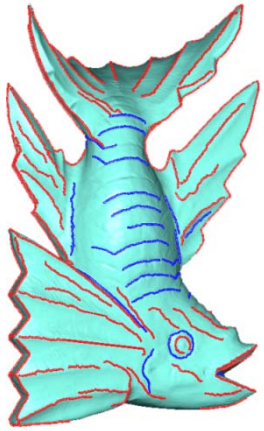
# Results

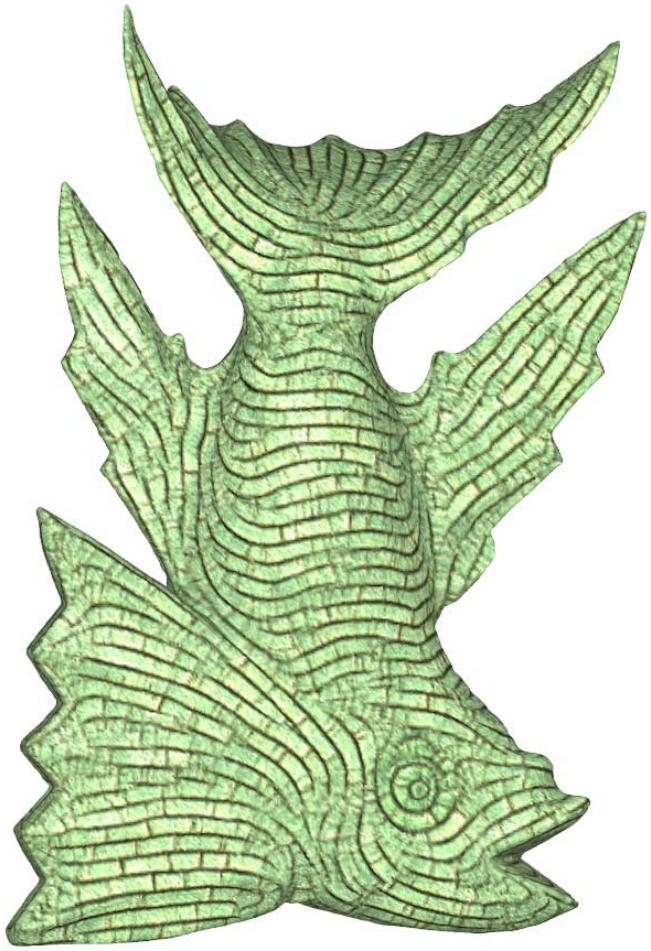




w/o alignment

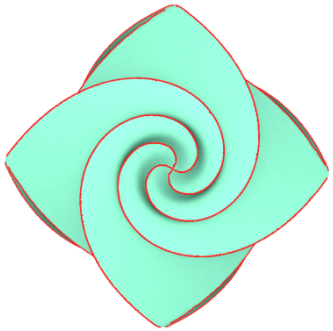
# Results

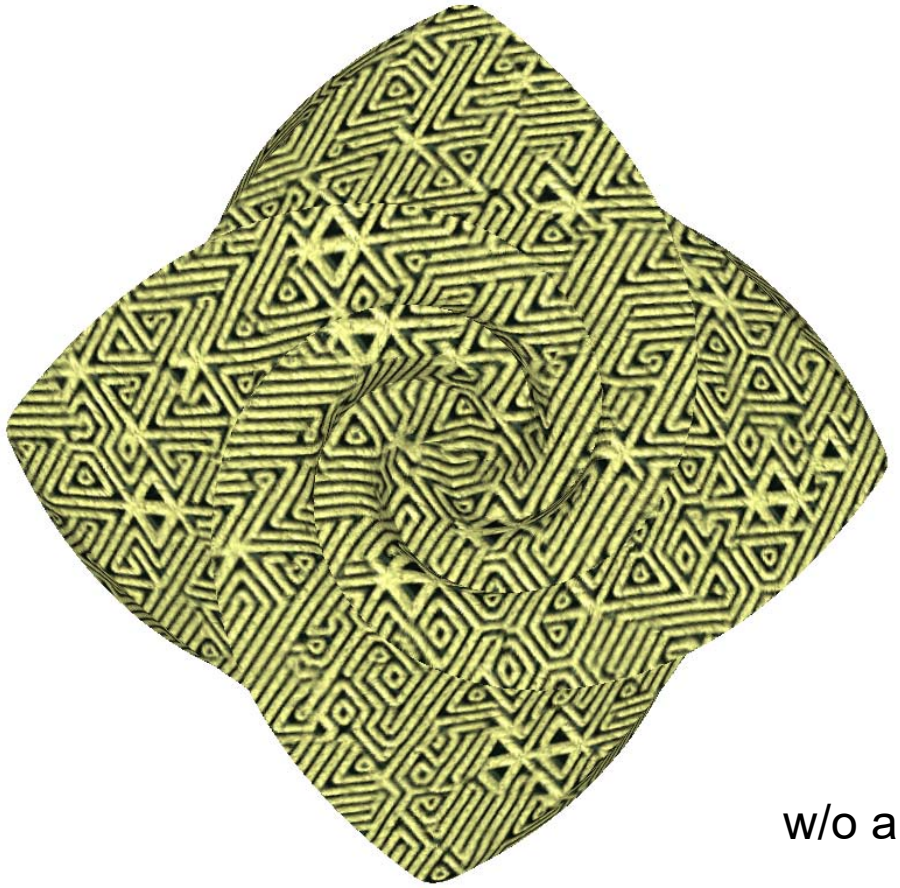




with alignment

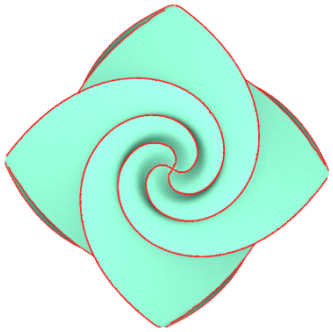
# Results

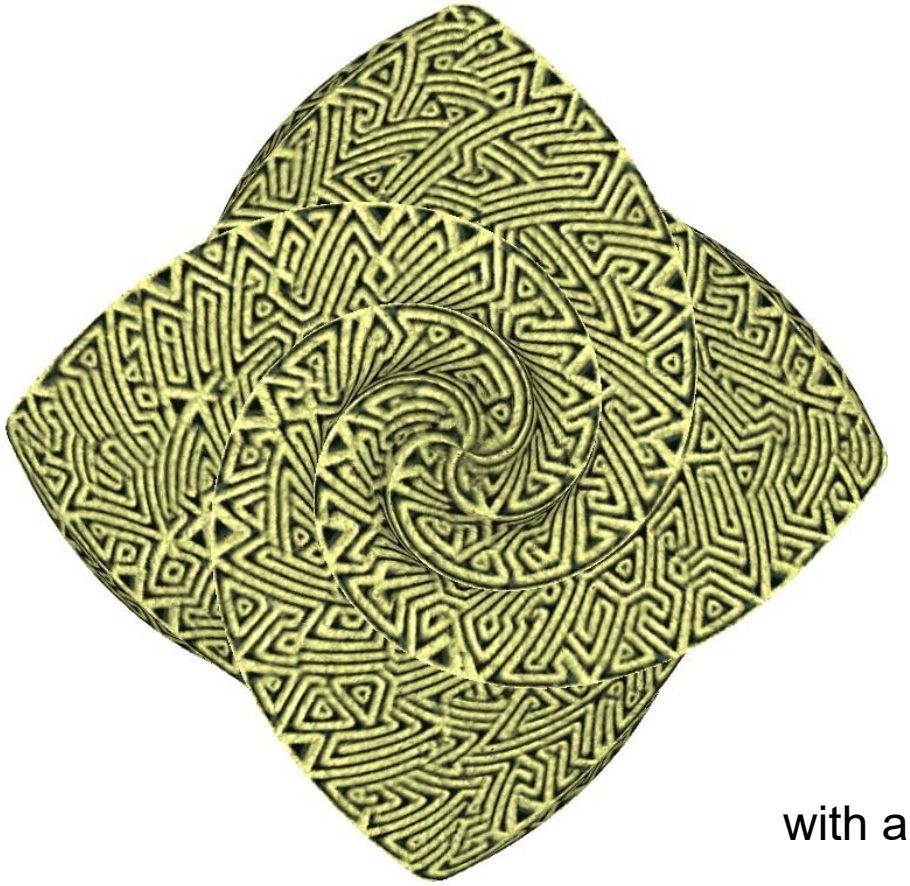




w/o alignment

# Results





with alignment

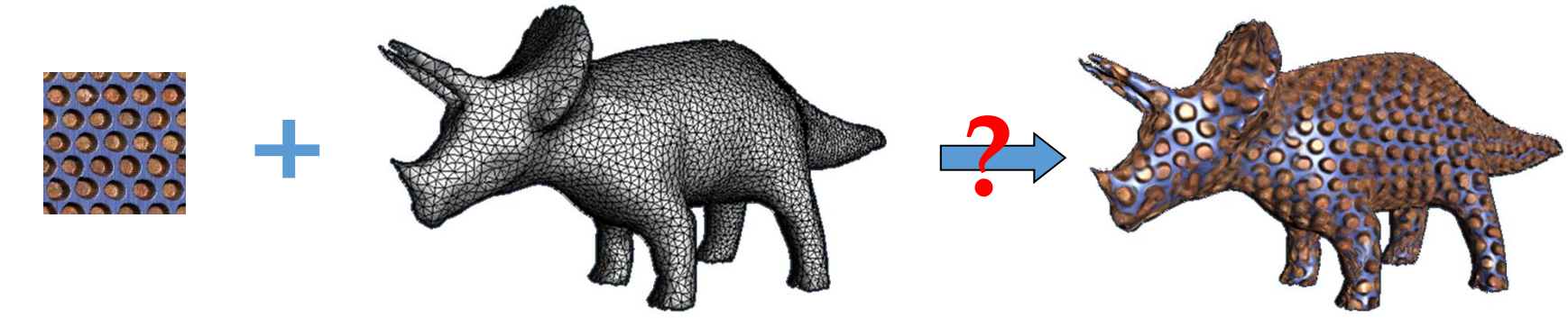
# Progressively-Variant Texture Synthesis

[Zhang et al., Siggraph 2003]

# BTF Synthesis

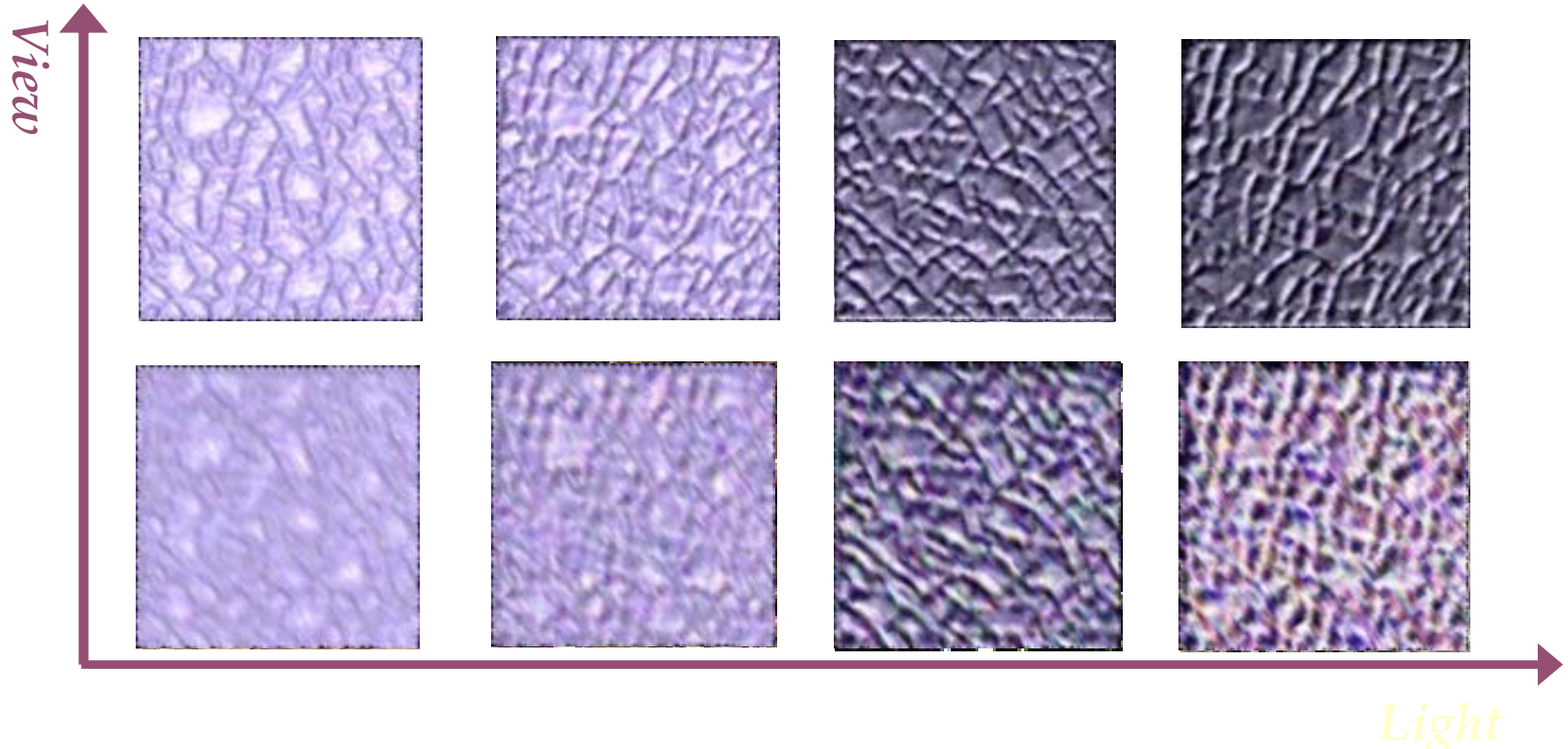
[Tong et al., Siggraph 2002]

- Bidirectional Texture Functions (BTF): A collection of images of the same surface under different lighting and viewing directions.
  - **6D Function** $(x, y, l_\theta, l_\varphi, v_\theta, v_\varphi)$
  - **Dense Sampling in Viewing/Lighting Directions**
  - **Capturing Appearance of Real World Surface**



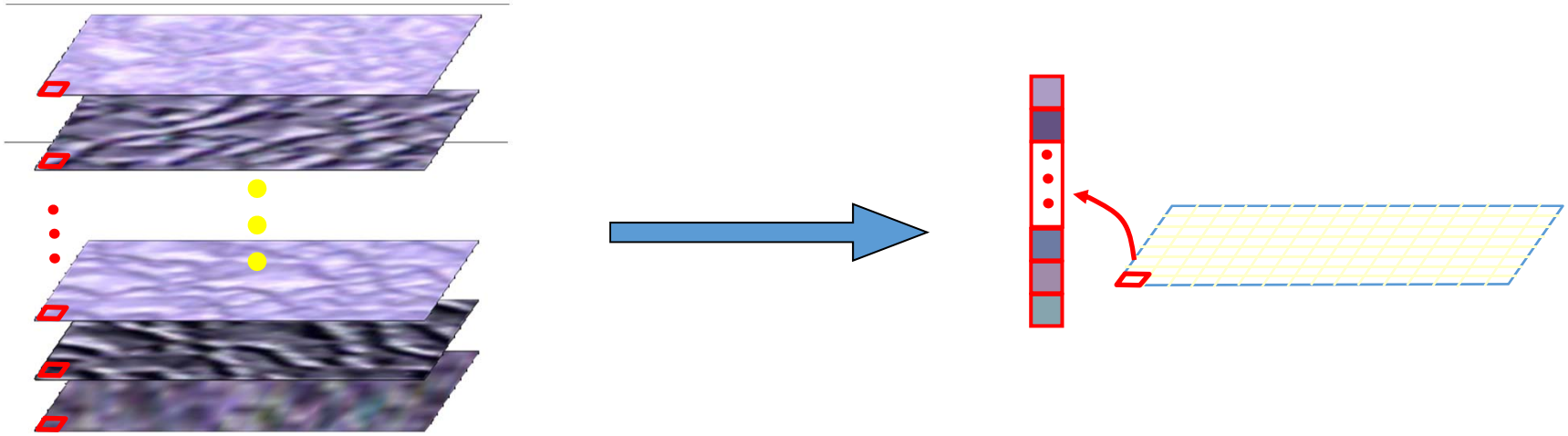

*Input BTF* *Input Mesh* *Result*

# *Real World Texture from CuRet*




- Geometry Details (Mesostructure) on Surface
- Self-Occlusion, Self-Shadow, and Specularity

# *Treating BTF as a 2D Texture Map*

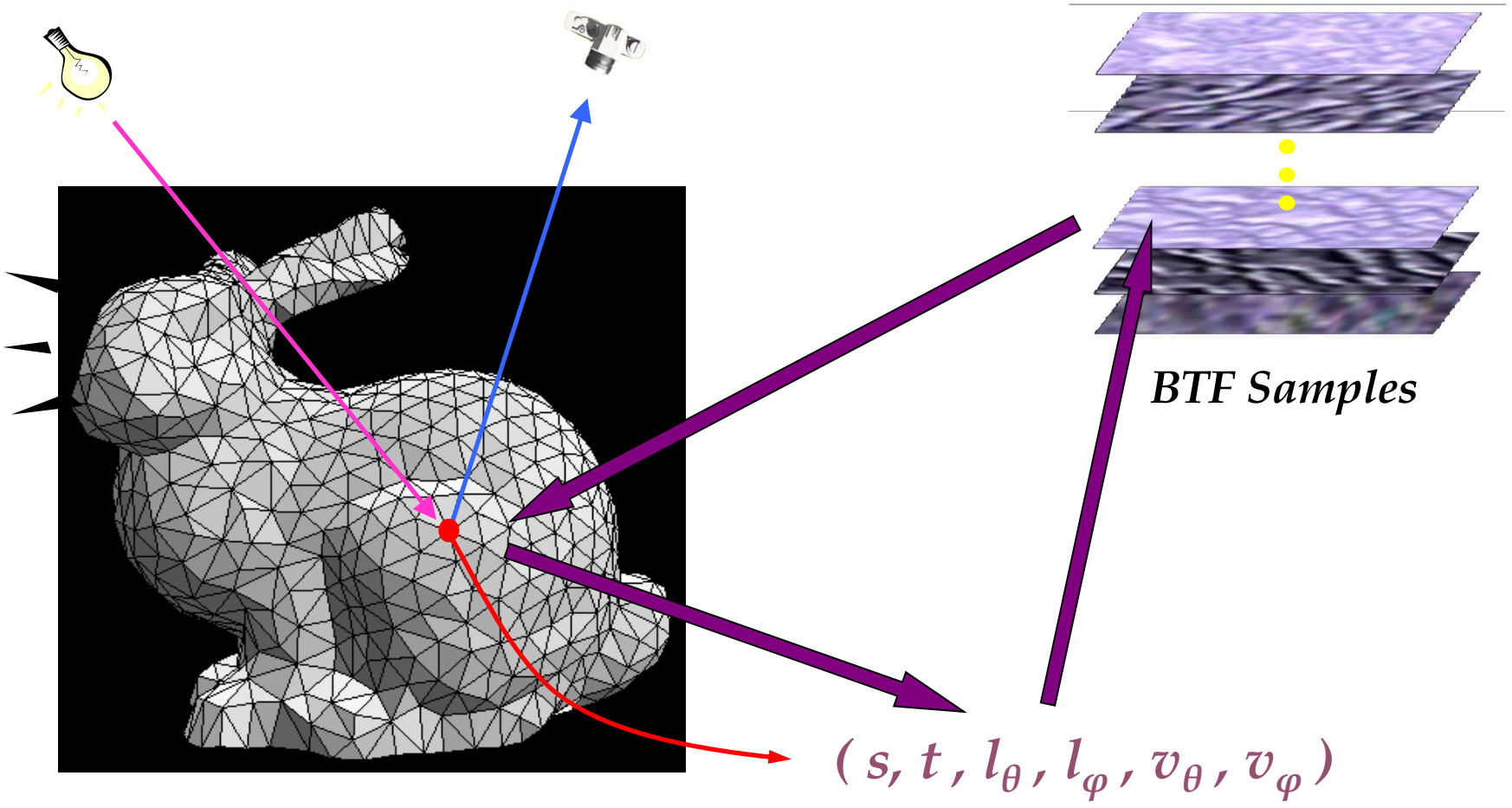


*12 X5 X 12 X5 = 3600*

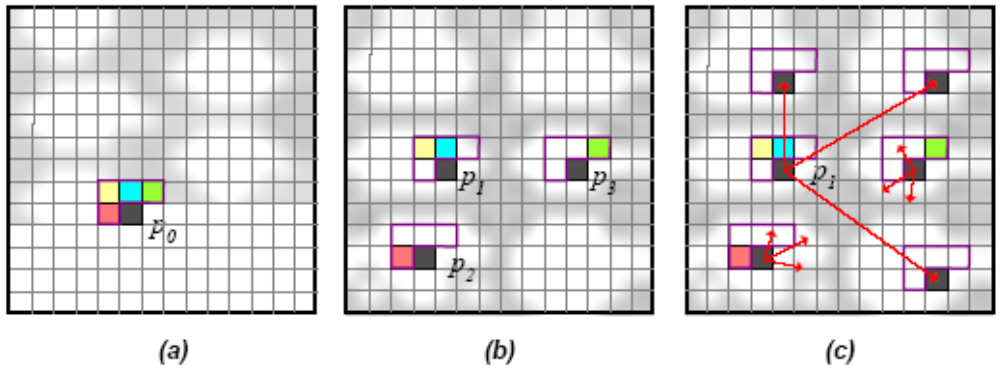
*10800-dimensional vector*

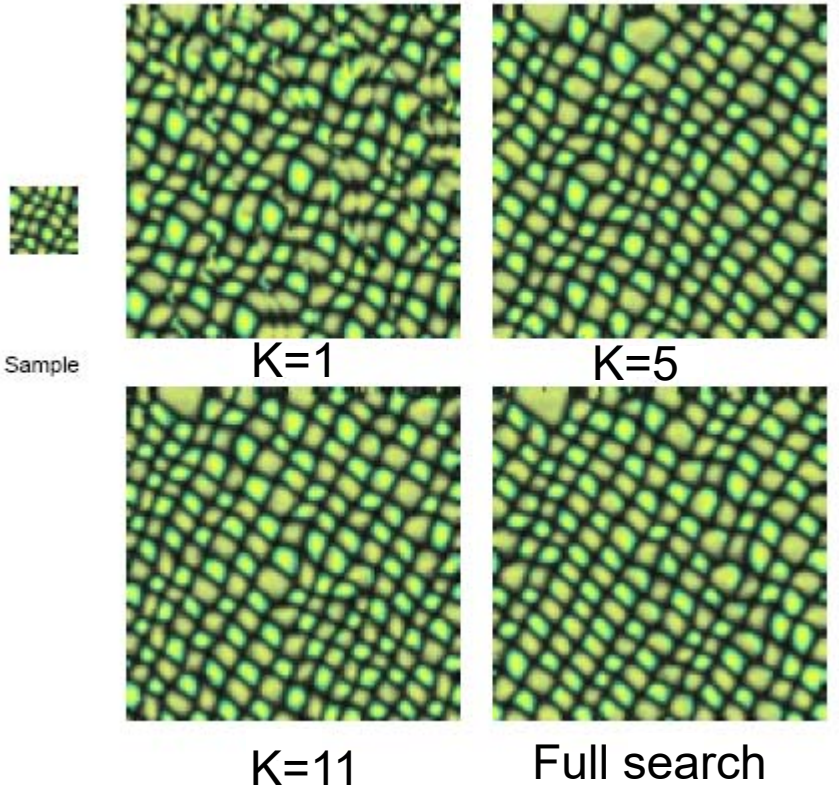
**Surface Texton**

# *Surface Texton Map & Rendering*

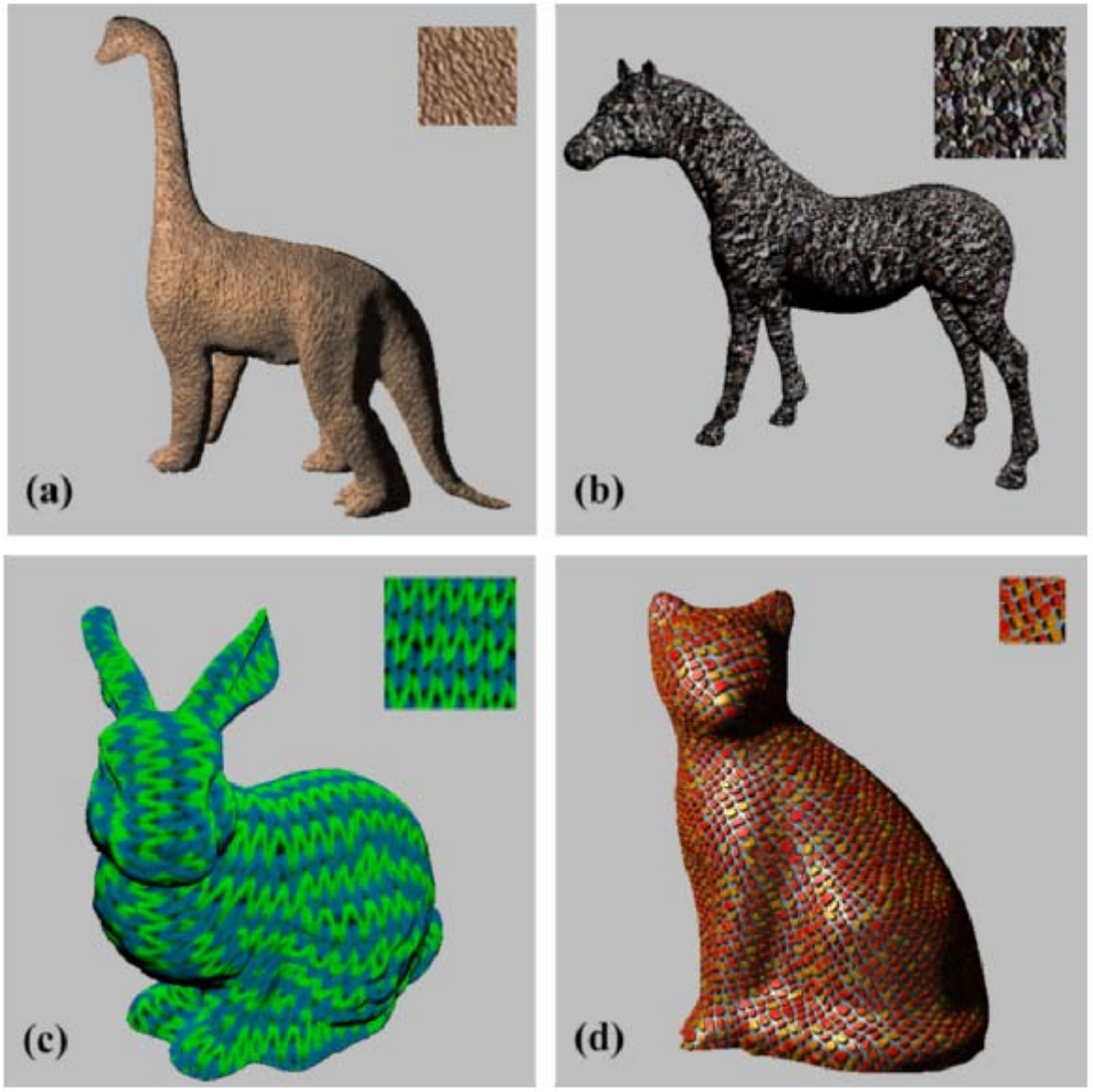


# K-Coherent Search

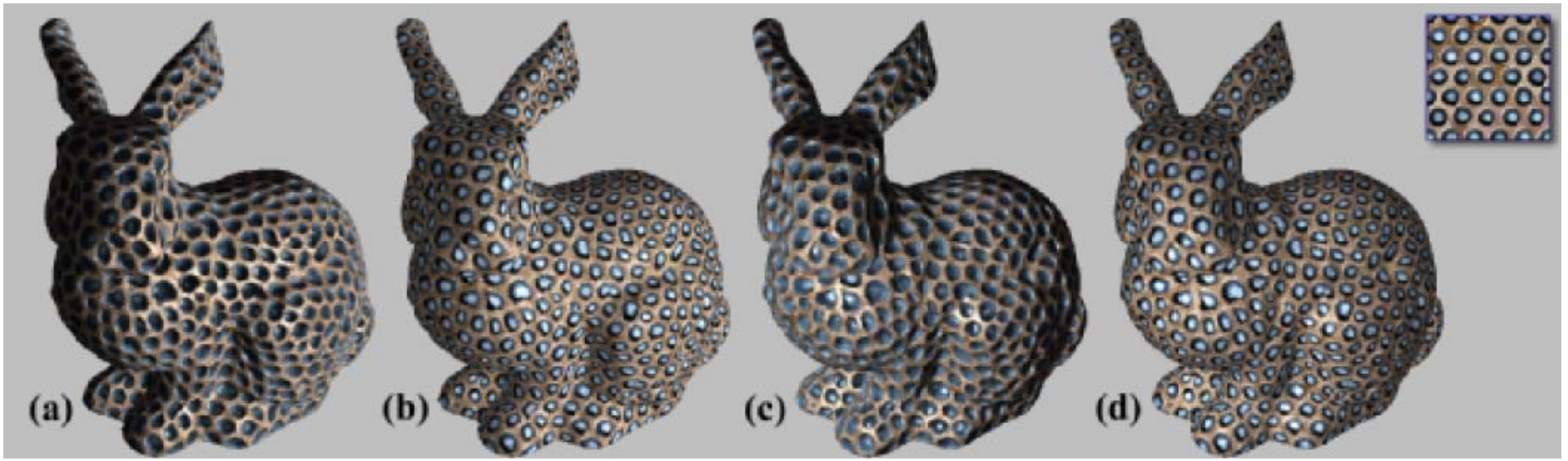


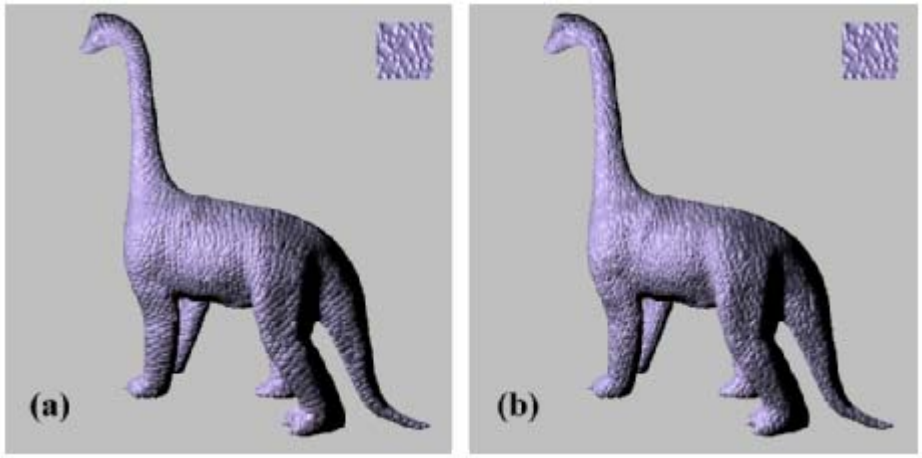



# Examples



# Comparison



# 3D Texture Synthesis

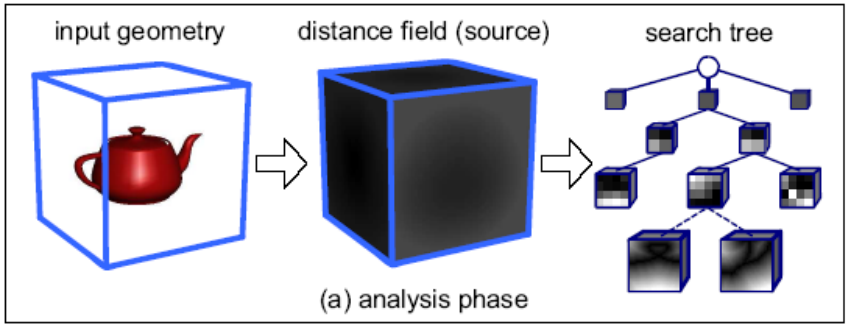
- Procedural texture synthesis
- Sample-based texture synthesis
- Geometry synthesis

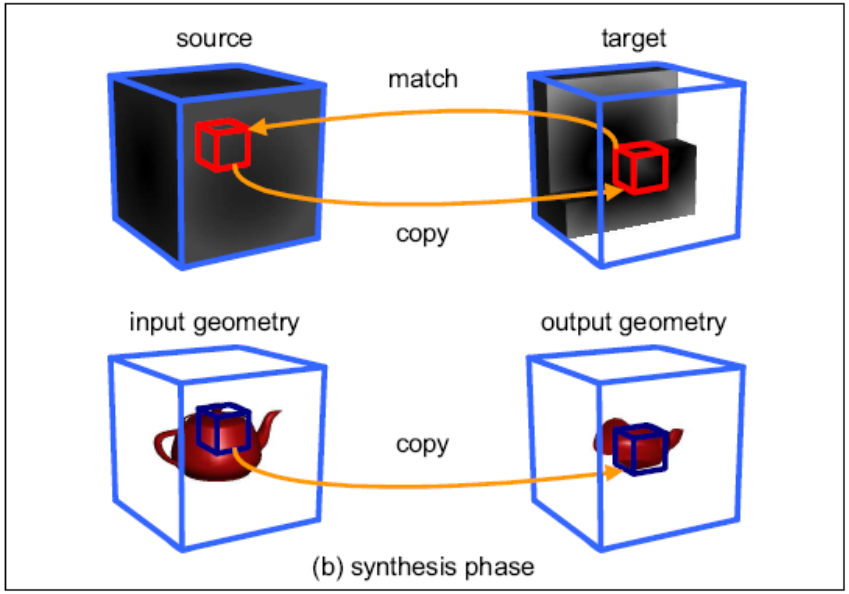
# Geometry Synthesis

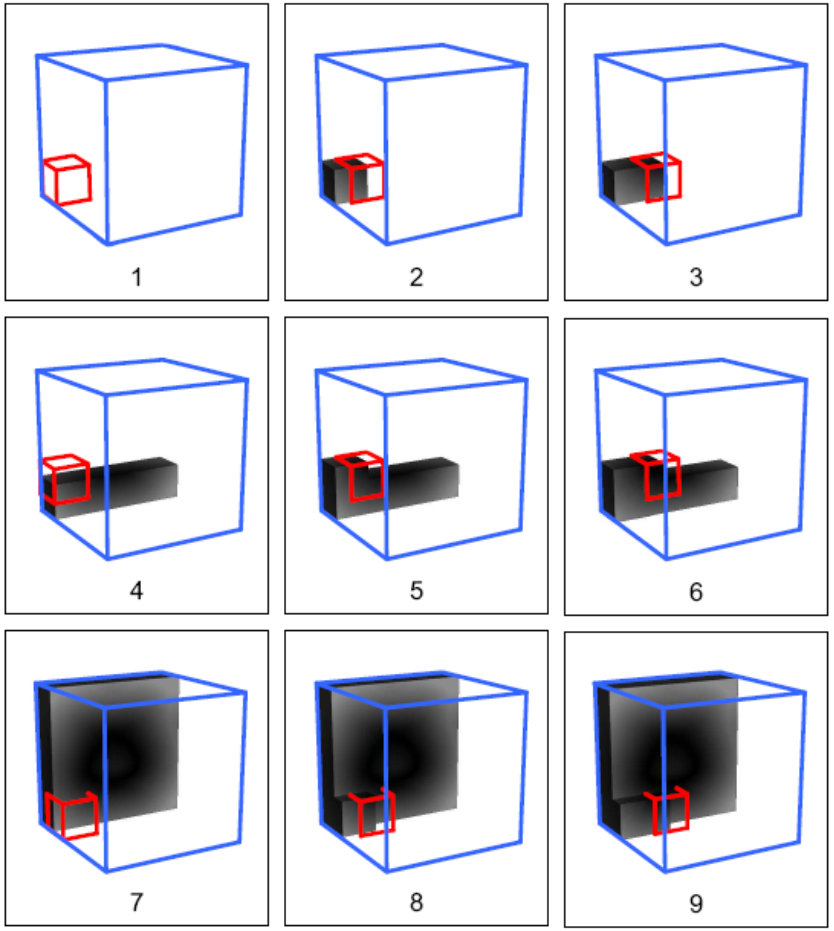
- Generating geometry over surfaces by texture (geometry) samples
- Methods:
  - 3D distance field based method
  - Image analogies extended to volumes
  - Mesh-based geometric texture synthesis technique

# 3D Distance Field based Method

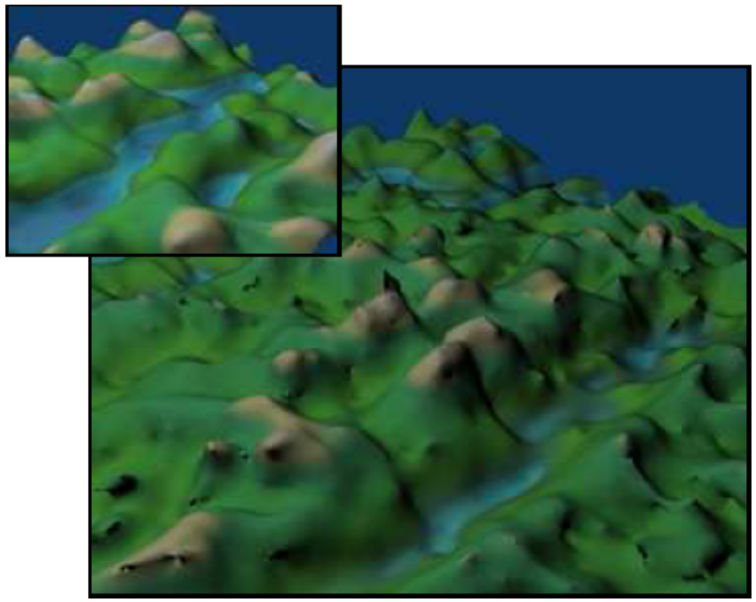
[Lagae et al., TR 2004]

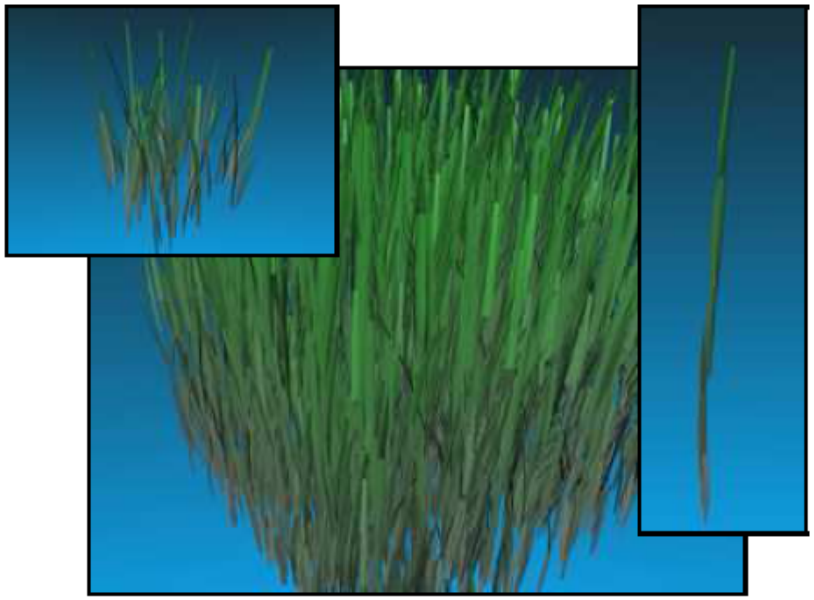


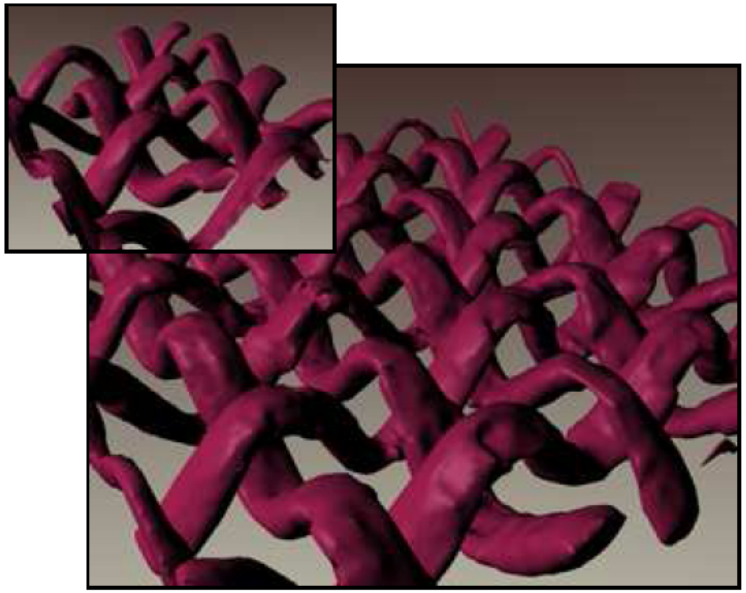



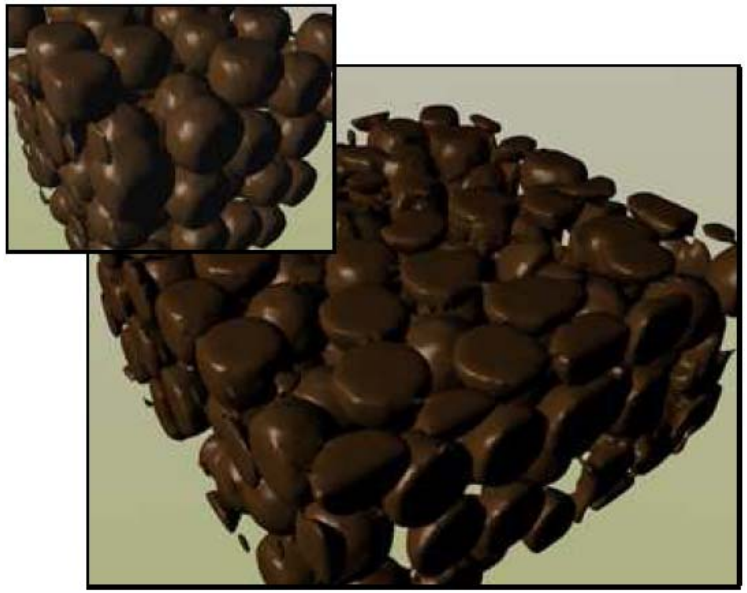



# Results



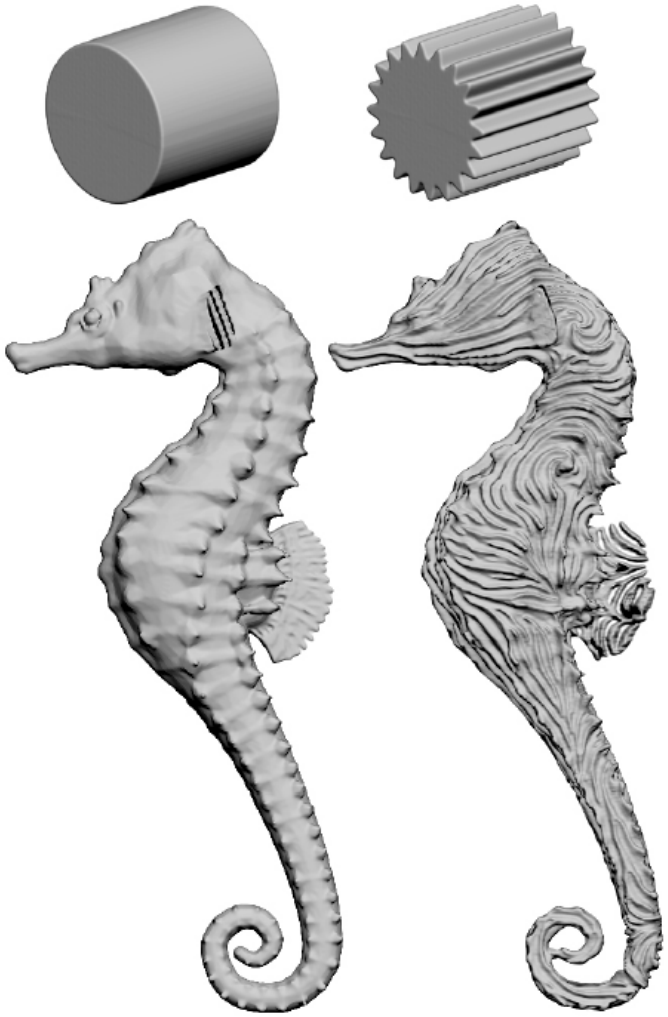


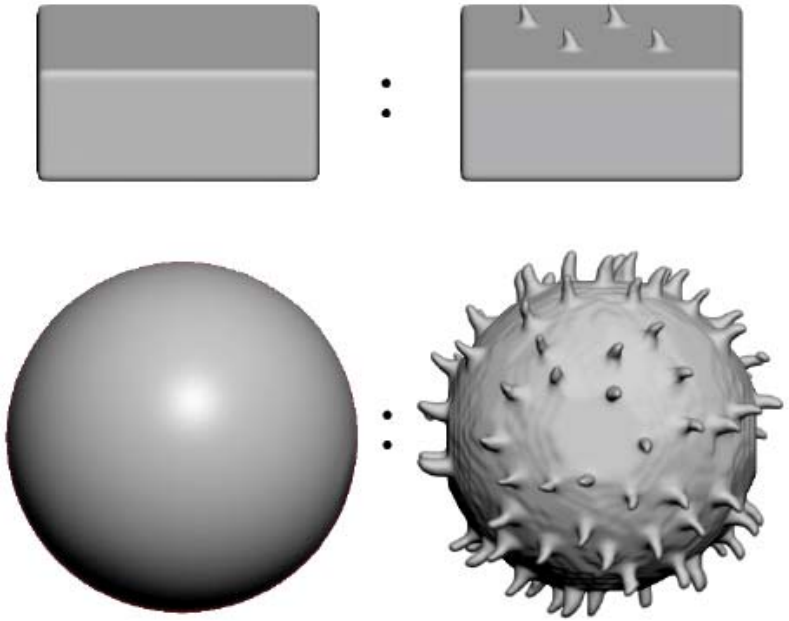




# Geometry Analogy

[Bhat et al., SGP 2004]

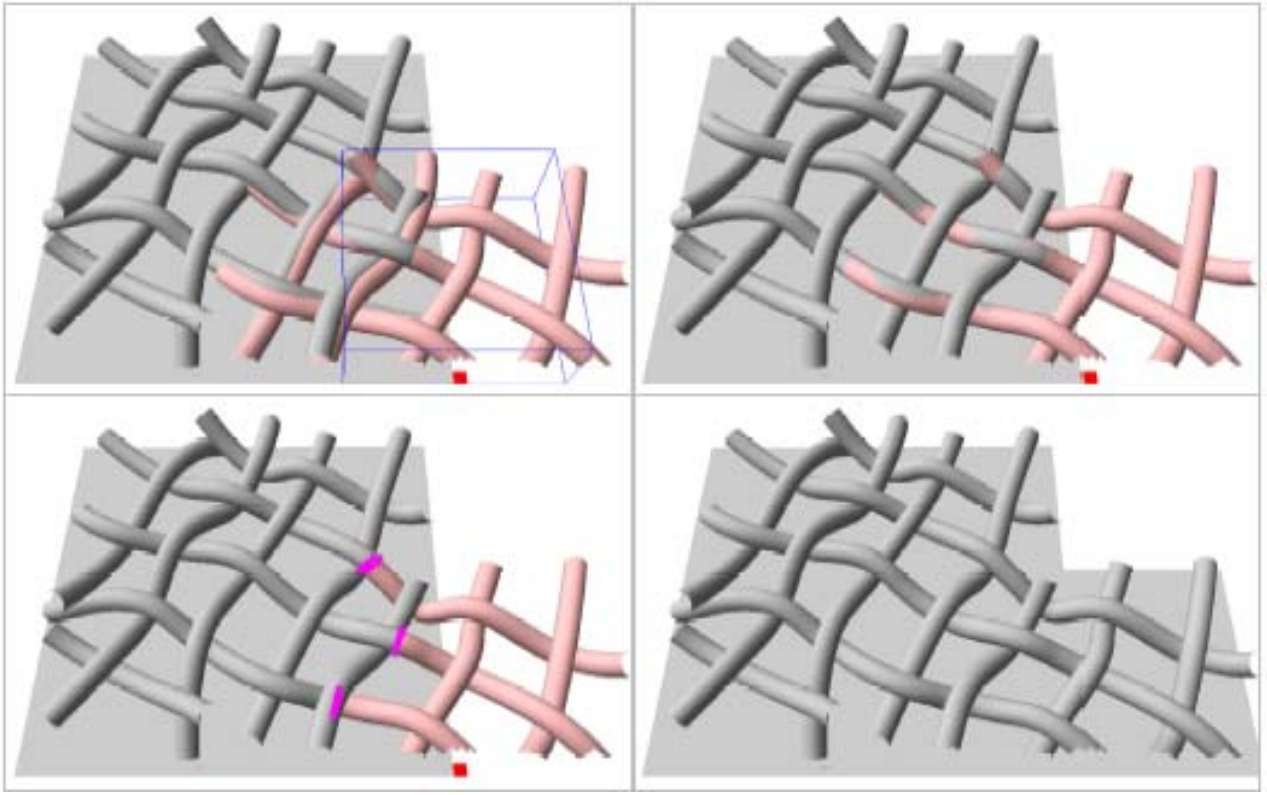




# Mesh Quilting

[Zhou et al., Siggraph 2006]

Geometry matching

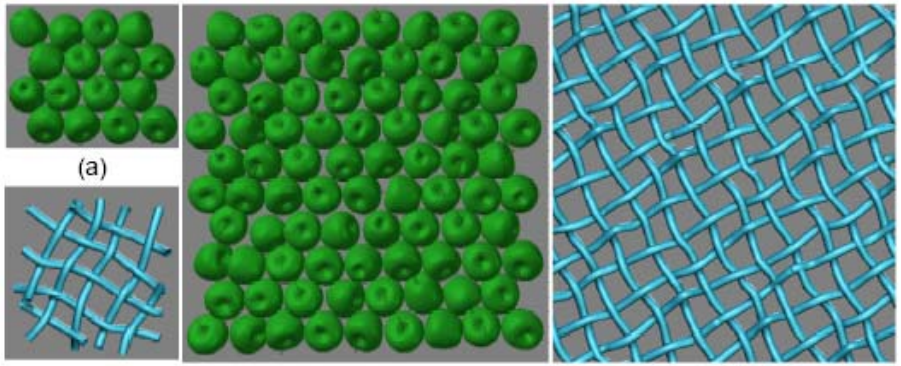


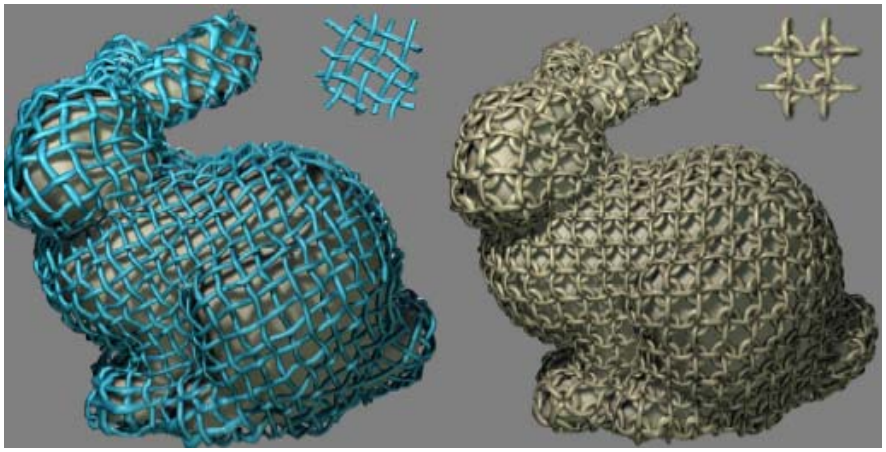
Element deformation

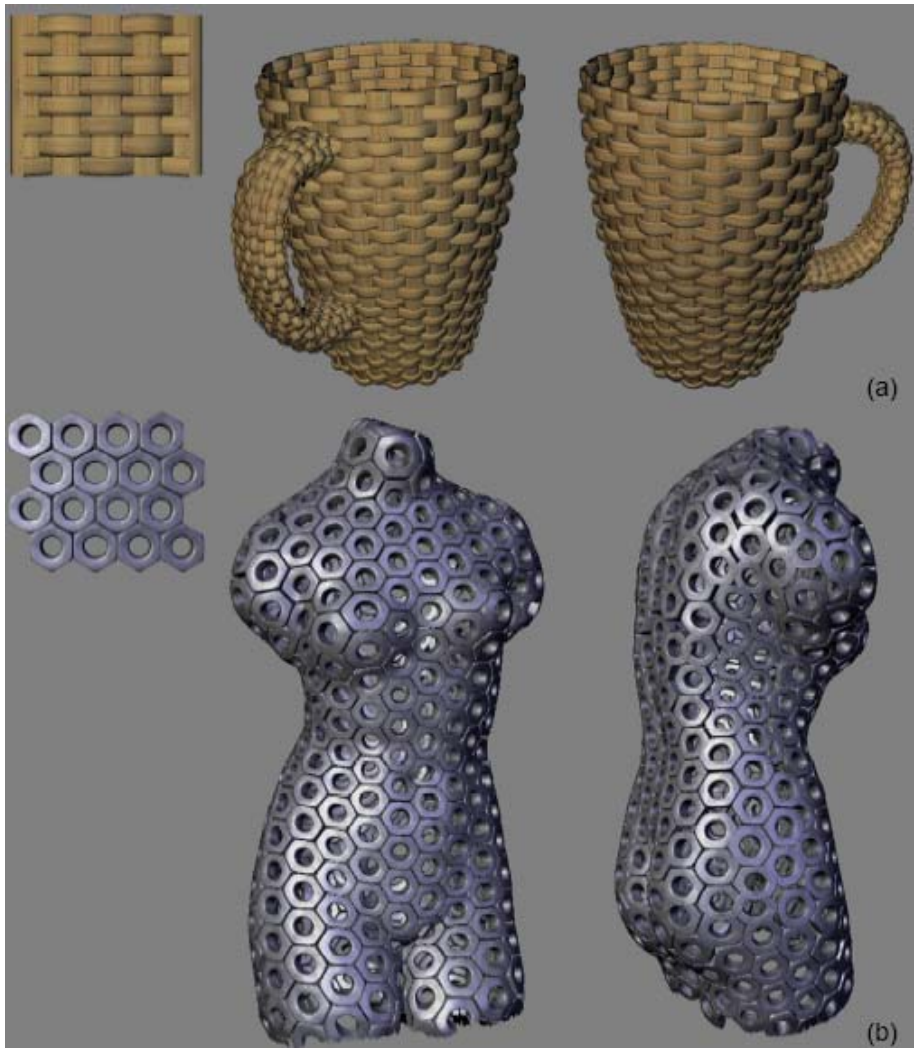
Element merging

Result

# Results

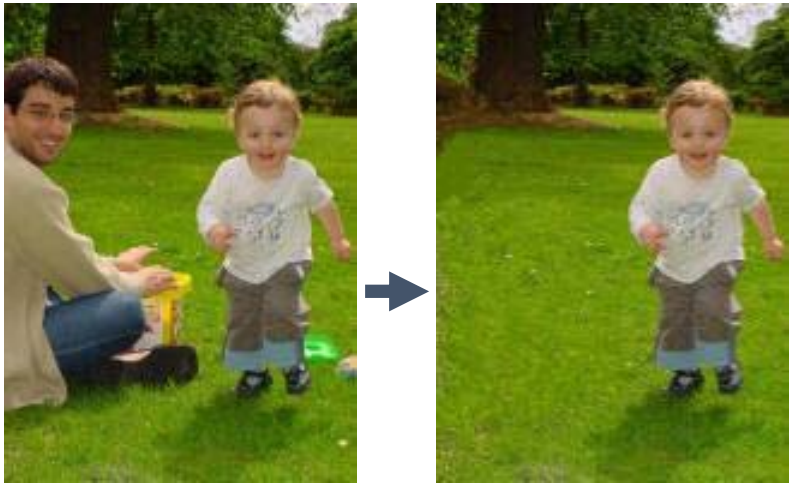


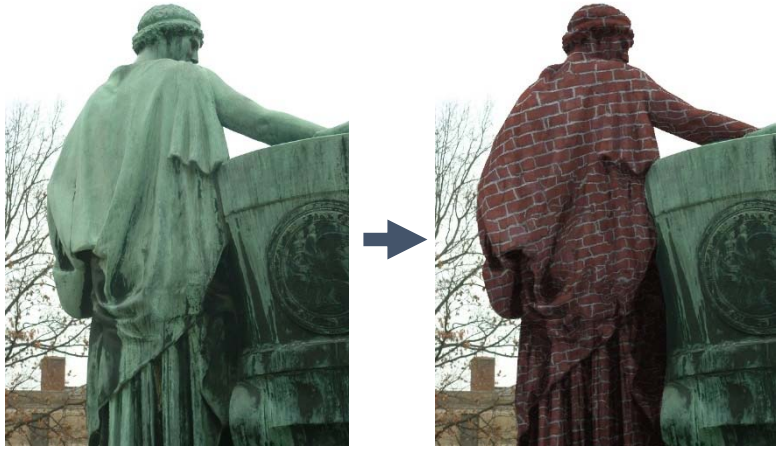





# Summary: Texture Synthesis

- An important topic on content generation (2D/3D) textures or repeated geometries
- A well-studied topic
- Many applications





中国科学技术大学

University of Science and Technology of China

# 谢　谢　！