

# Introduction to Computer Graphics

## – Image Processing (2) –

June 16, 2016

Kenshi Takayama

# Texture Synthesis

# Scenario 1: Removal of objects in images

Original



Mask



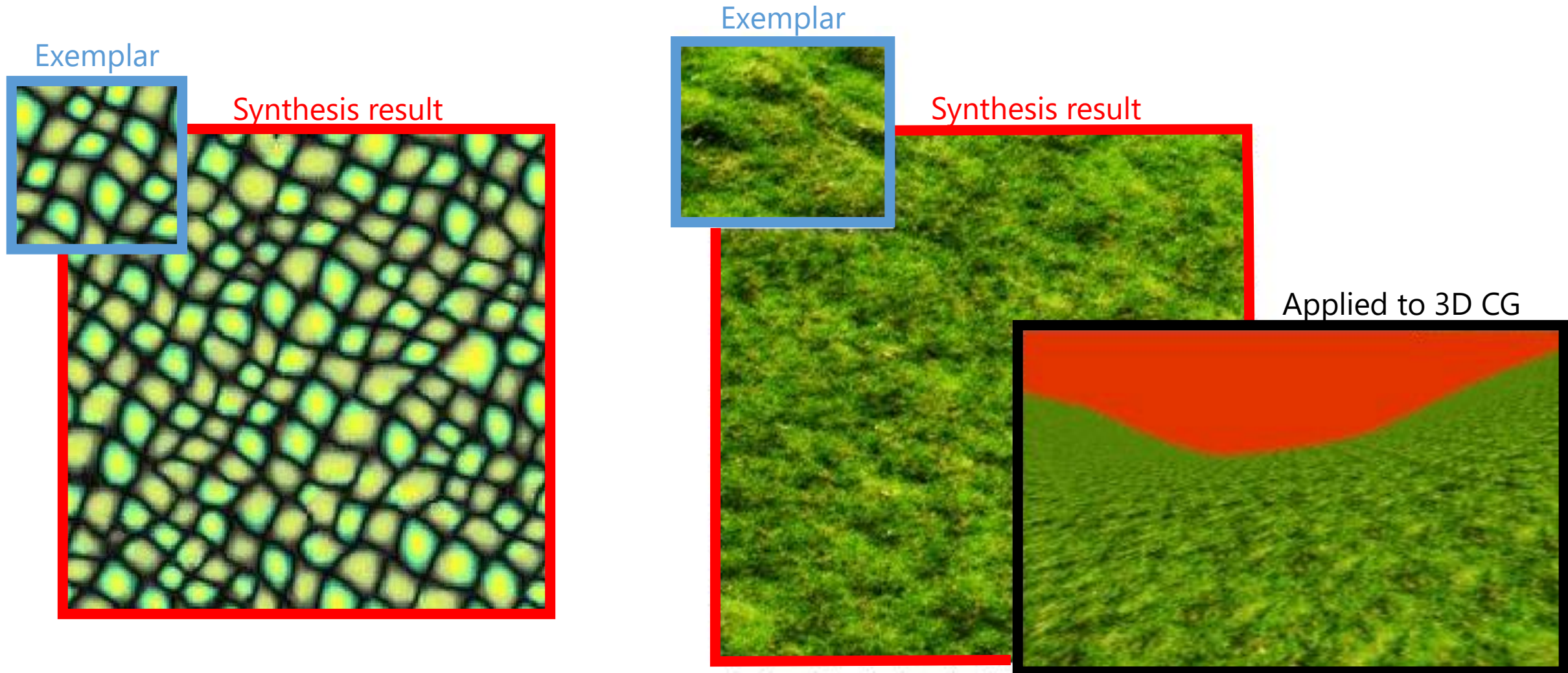
Synthesis result



- Bit different problem setting than “image cloning”

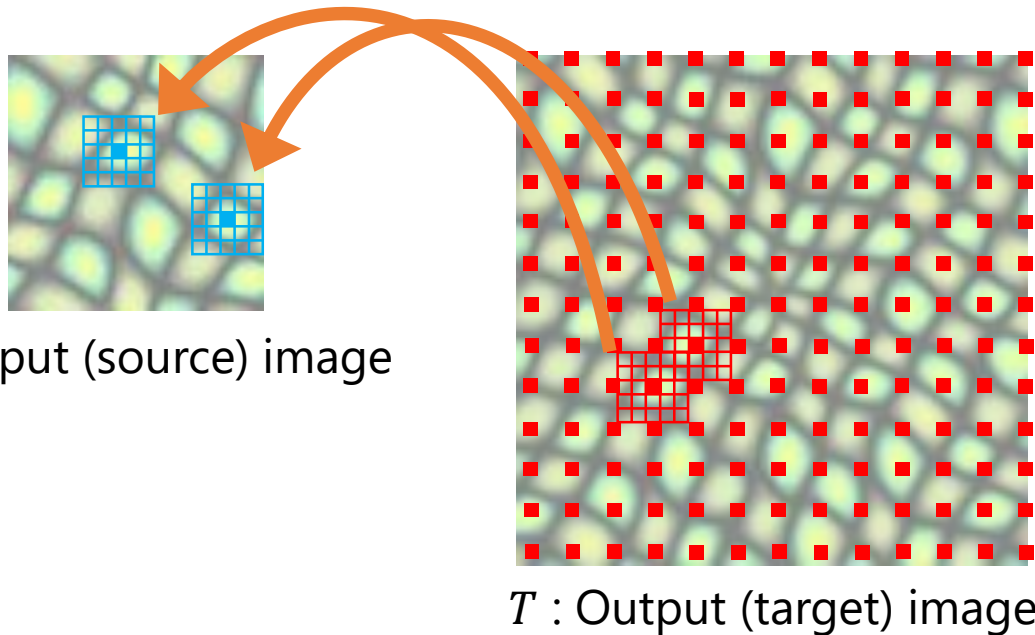


# Scenario 2: Synthesis of large texture image



# Similarity between input & output images [Kwatra05]

Look for the most similar patch



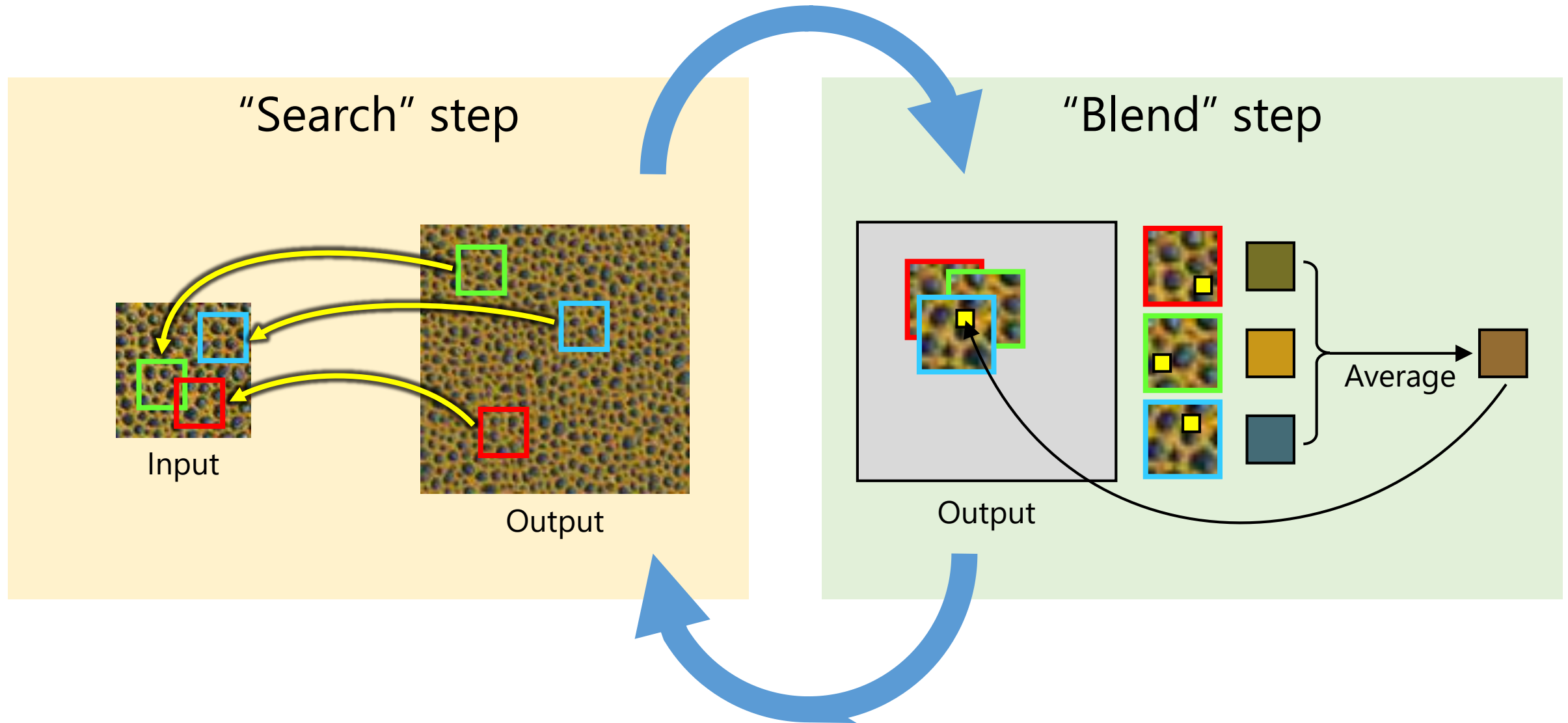
$$D(S, T) = \sum_{t \in T} \min_{s \in S} \|s - t\|^2$$

Sum of per-pixel differences squared

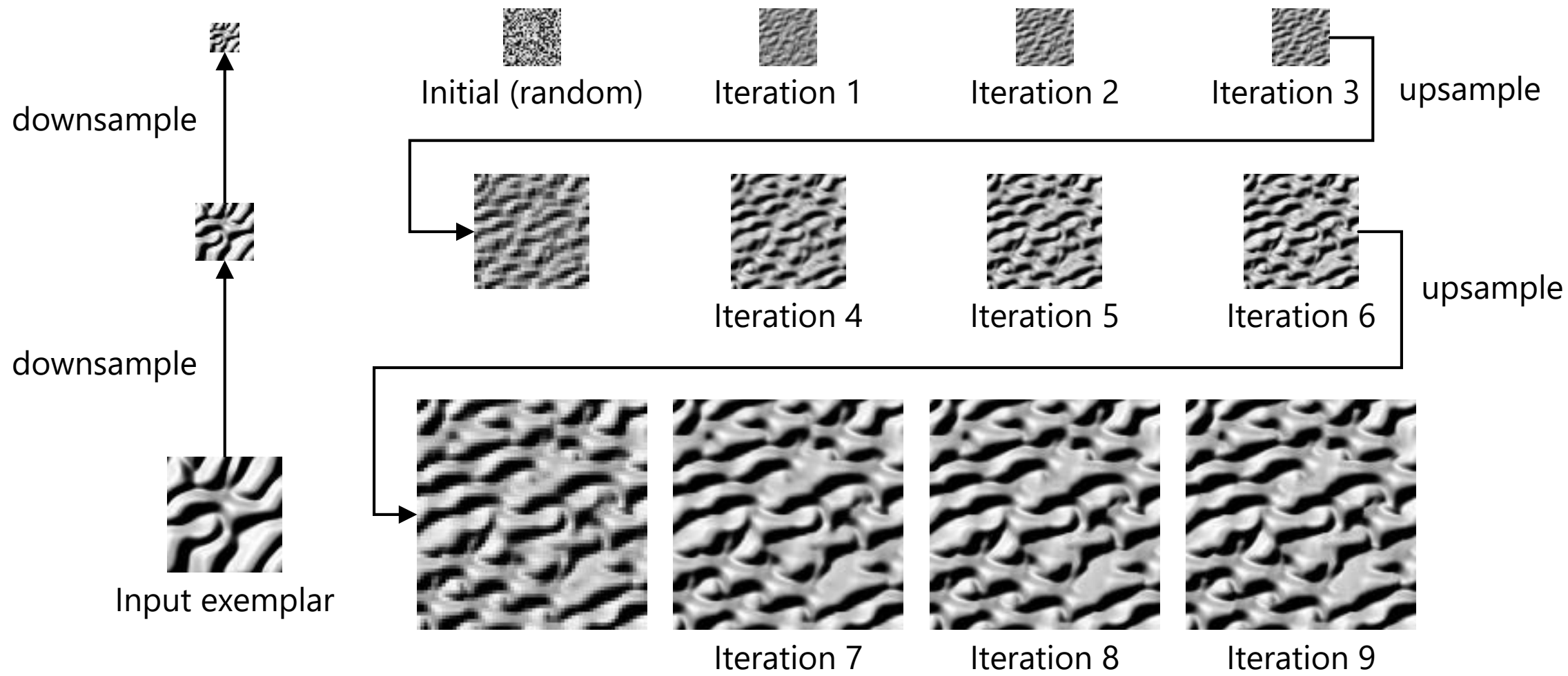
patches

- Want to find  $T$  which minimizes  $D$
- Direct solution seems infeasible  
→ iterative computation

# Optimization by iterative computation [Kwatra05]



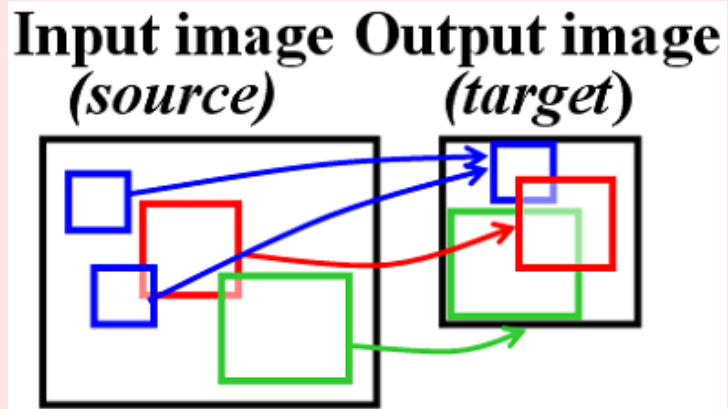
# Multiresolution synthesis



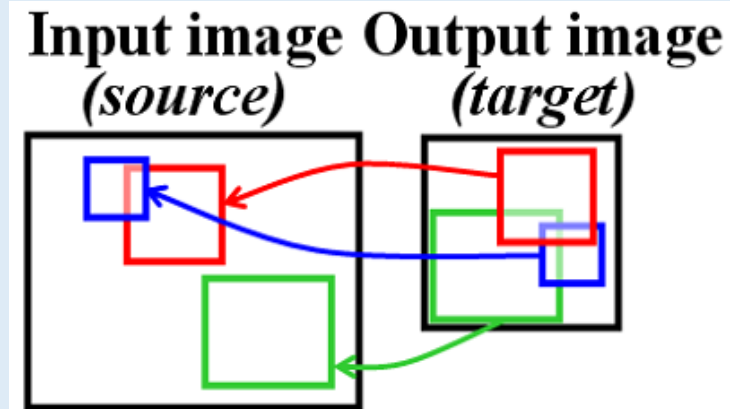
# Bidirectional similarity [Simakov08; Wei08]

$$D(S, T) = \sum_{s \in S} \min_{t \in T} \|s - t\|^2 + \lambda \sum_{t \in T} \min_{s \in S} \|s - t\|^2$$

Completeness term



Coherence term

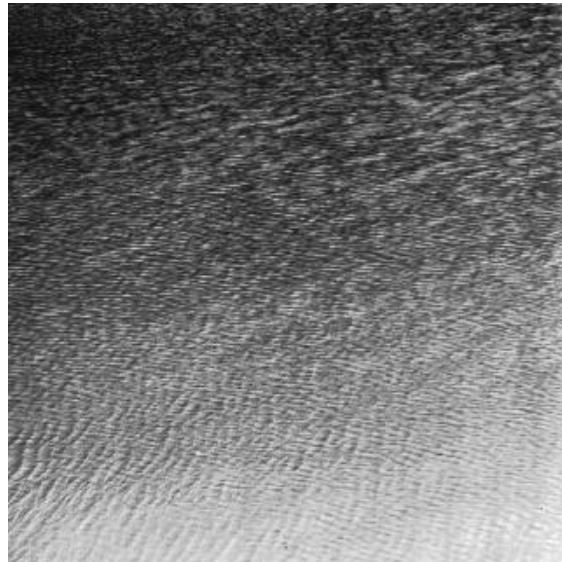




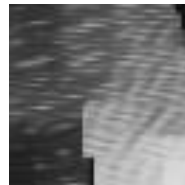
# Effect of Completeness/Coherence terms

problem a.k.a.  
"Image Summarization"

Input image



Output image

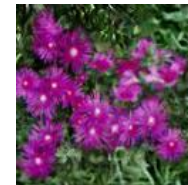


Completeness only

$$\sum_{s \in S} \min_{t \in T} \|s - t\|^2$$



Bidirectional



Coherence only

$$\sum_{t \in T} \min_{s \in S} \|s - t\|^2$$

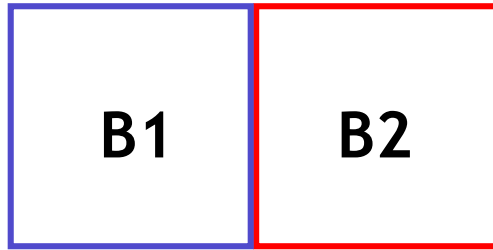
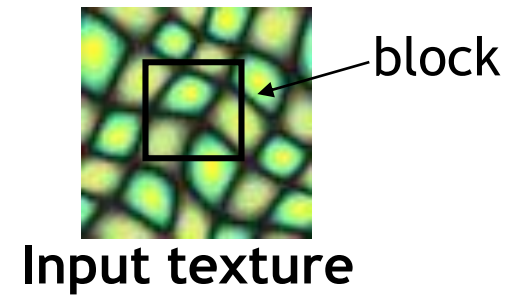


Bidirectional

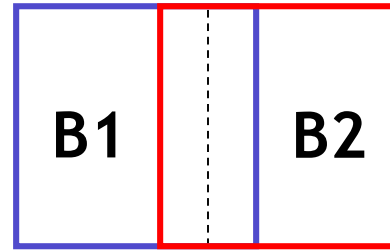
# Texture synthesis via stitching of patches

(briefly)

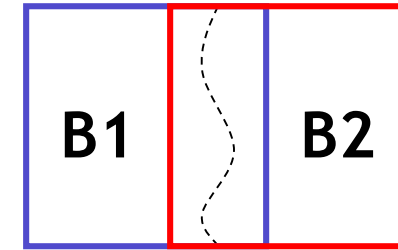
# Image Quilting [Efros01]



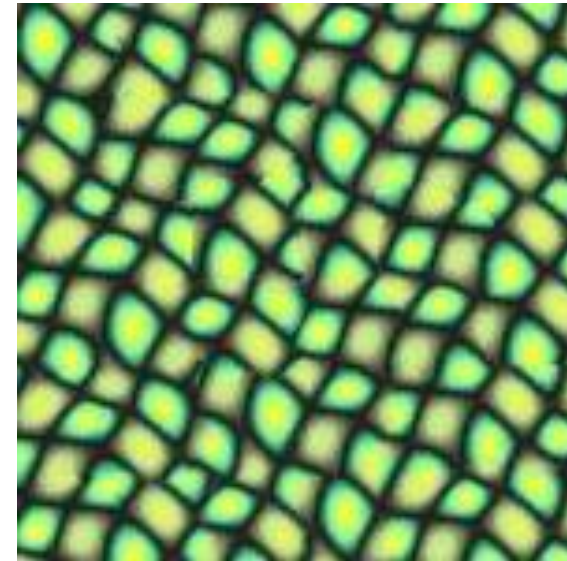
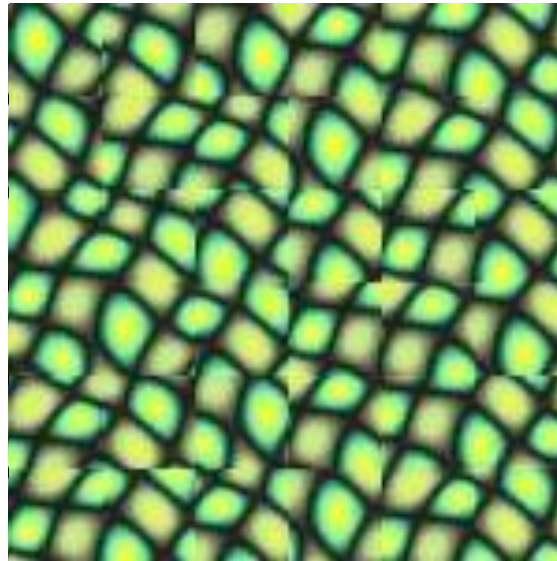
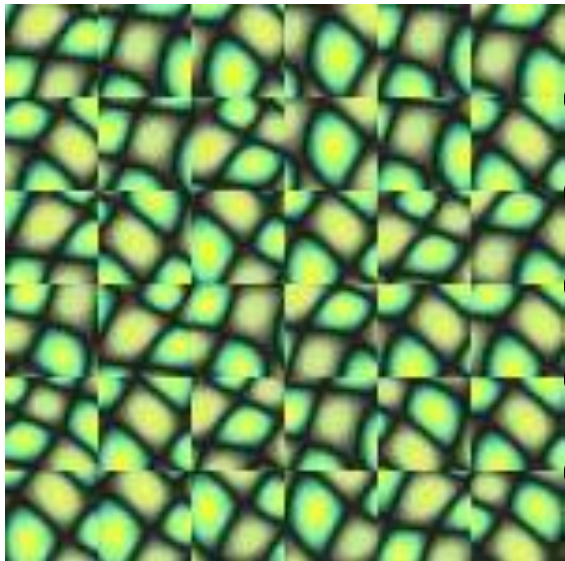
Random placement  
of blocks



Neighboring blocks  
constrained by overlap

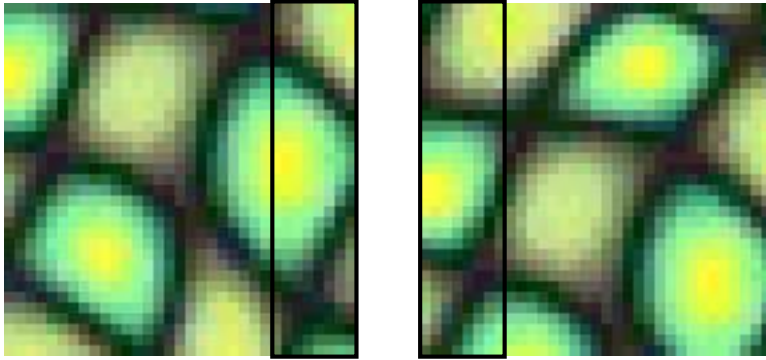


Minimal error  
boundary cut

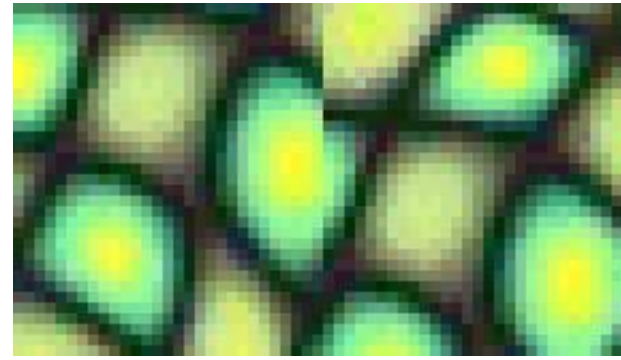


# Image Quilting [Efros01]

overlapping blocks

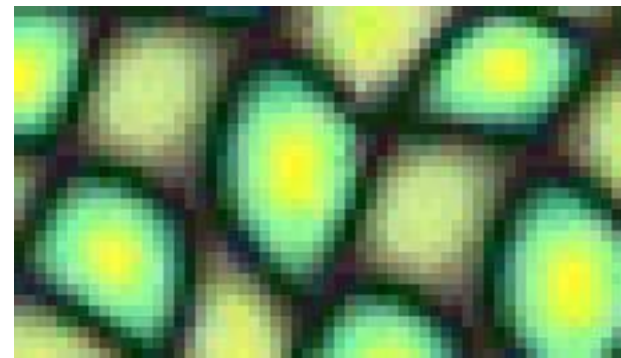


vertical boundary



A diagram illustrating the calculation of overlap error. It shows two vertical image strips, each containing the same pattern of green and yellow circular spots. A minus sign is placed between the two strips, and the entire expression is enclosed in large square brackets. To the right of the brackets is a superscript '2', followed by an equals sign and a vertical strip showing the resulting error. The error strip is black and white, with a red line tracing the boundary where the error is highest.

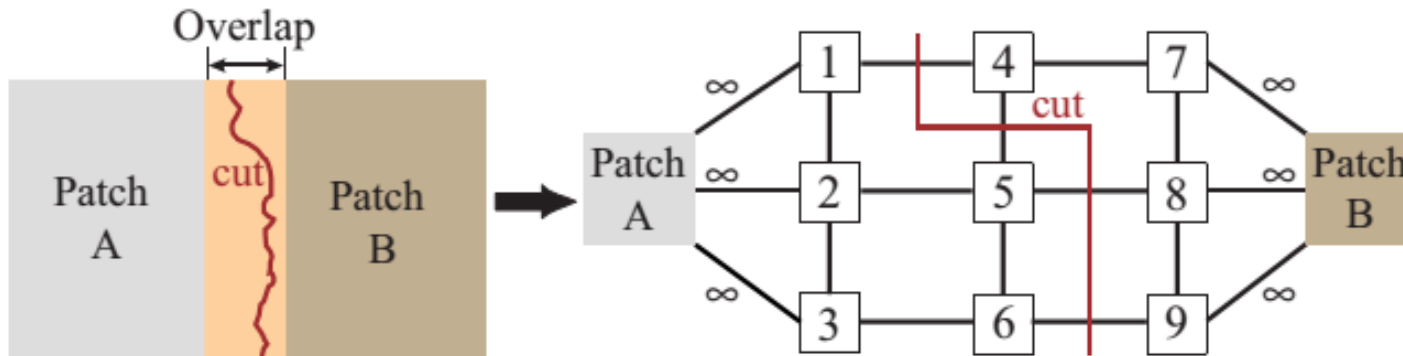
overlap error



min. error boundary



# Graphcut Textures [Kwatra03]

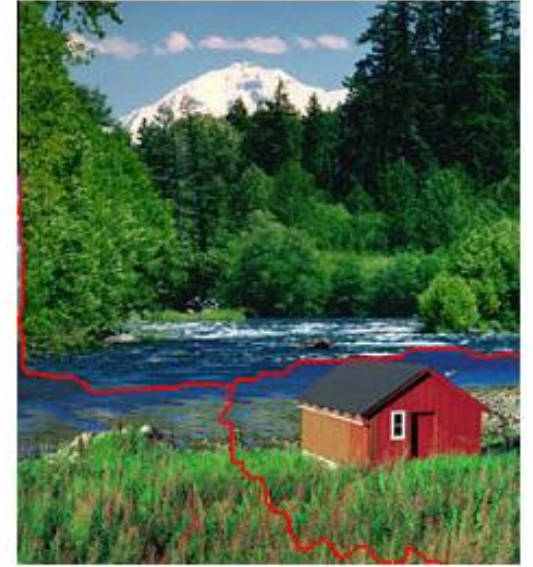
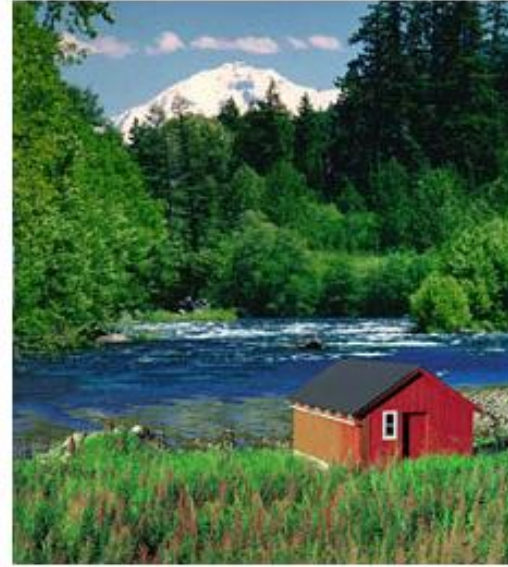


<https://www.youtube.com/watch?v=Ya6BshBH6G4>

- Formulate the best seam between patches as a minimum-cost cut in a graph




# Graphcut Textures [Kwatra03]

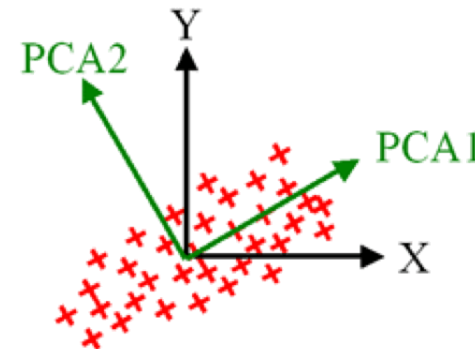
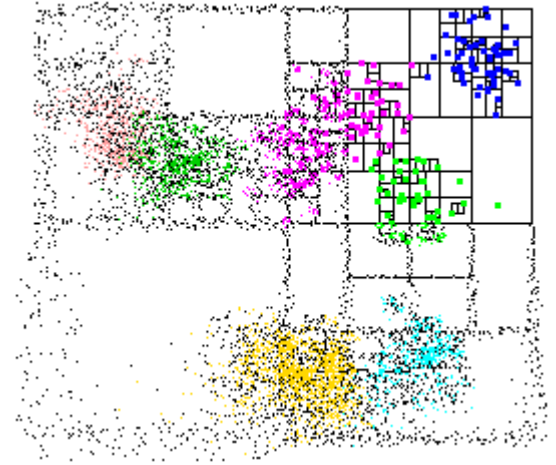


# Acceleration techniques for nearest neighborhood search

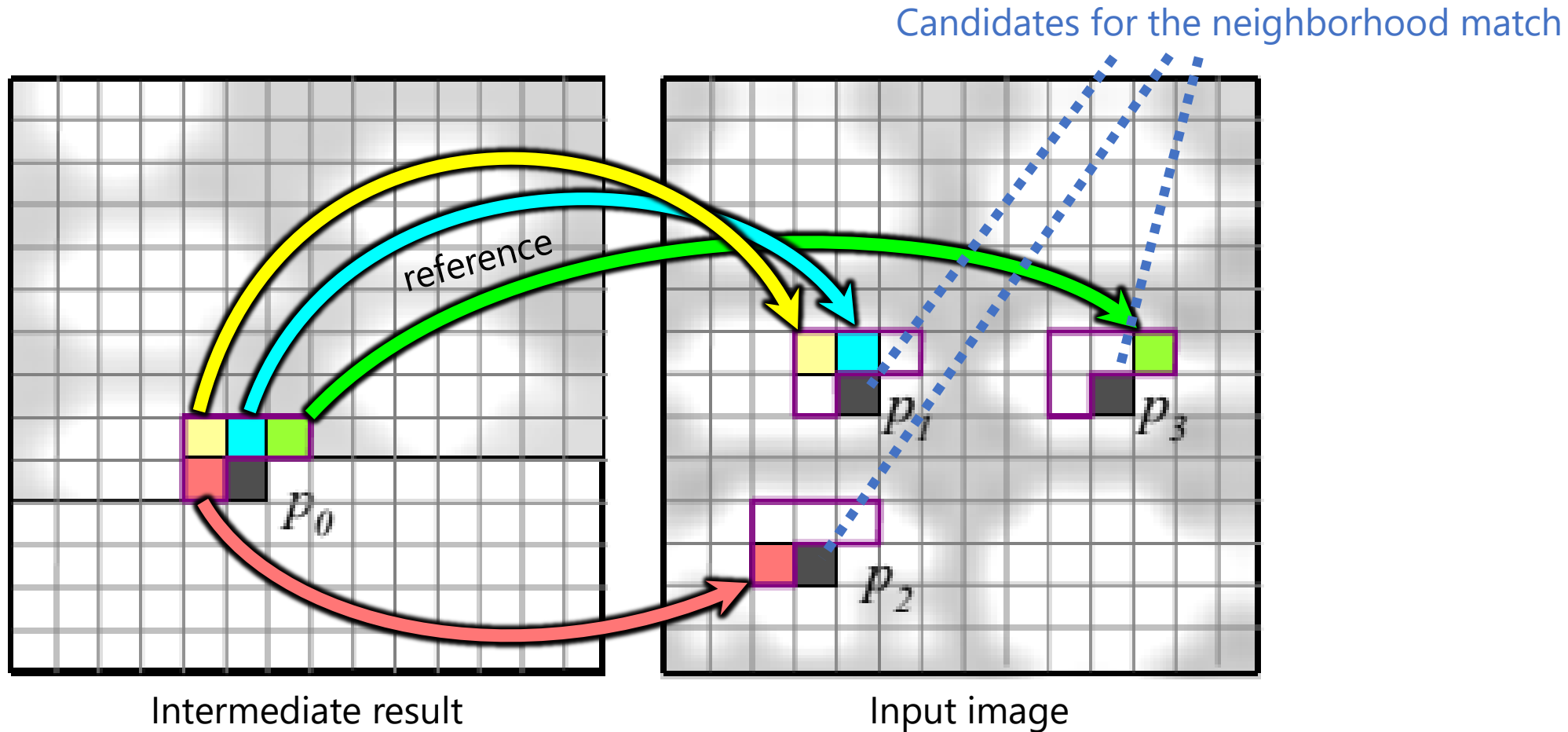
# Technique #1:

## Spatial data structure + dimensionality reduction

- 5x5 neighbor pixels each with RGB channels   
→ 75D vector
- Nearest neighbor search in high dimensional space  
→ Acceleration using k-d tree
- k-d tree performs poorly when dimensionality is too high  
→ Dimensionality reduction using **Principal Component Analysis**

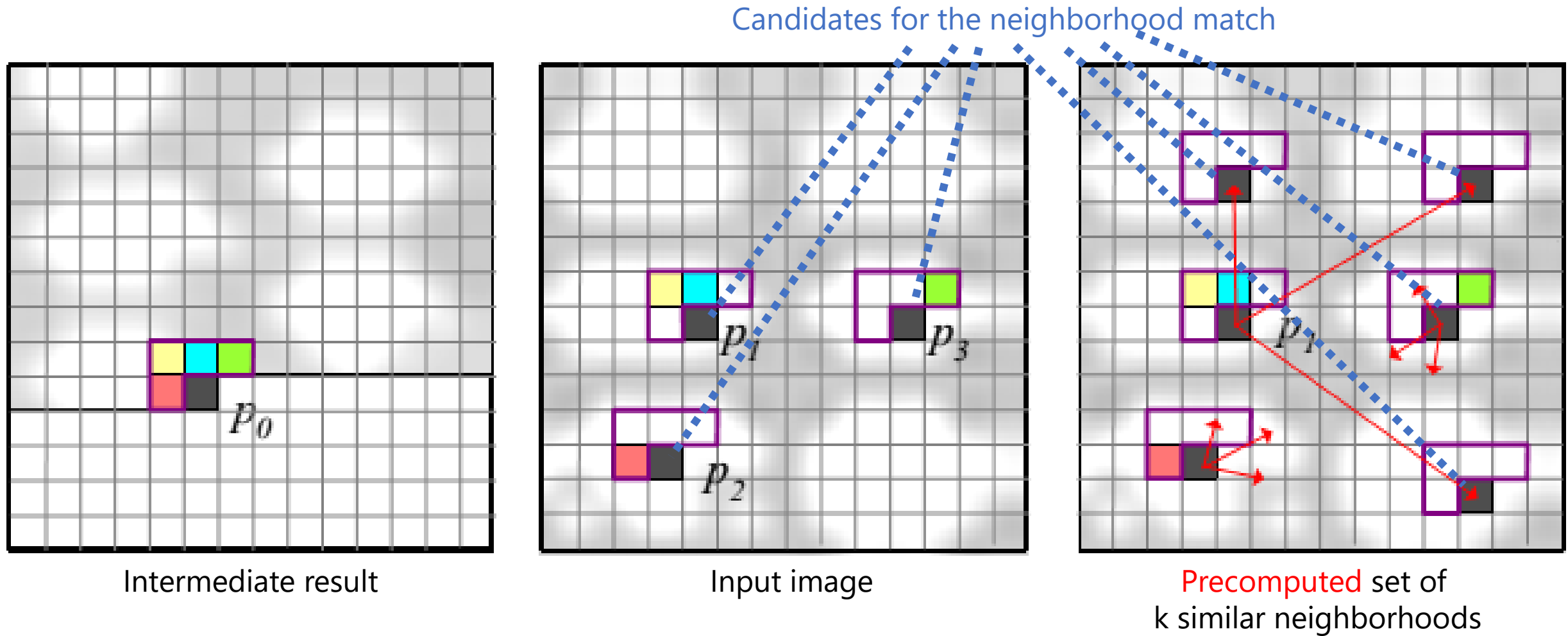


# Technique #2: k-coherence [Tong02]





# Technique #2: k-coherence [Tong02]





# Best bet: PatchMatch [Barnes09]

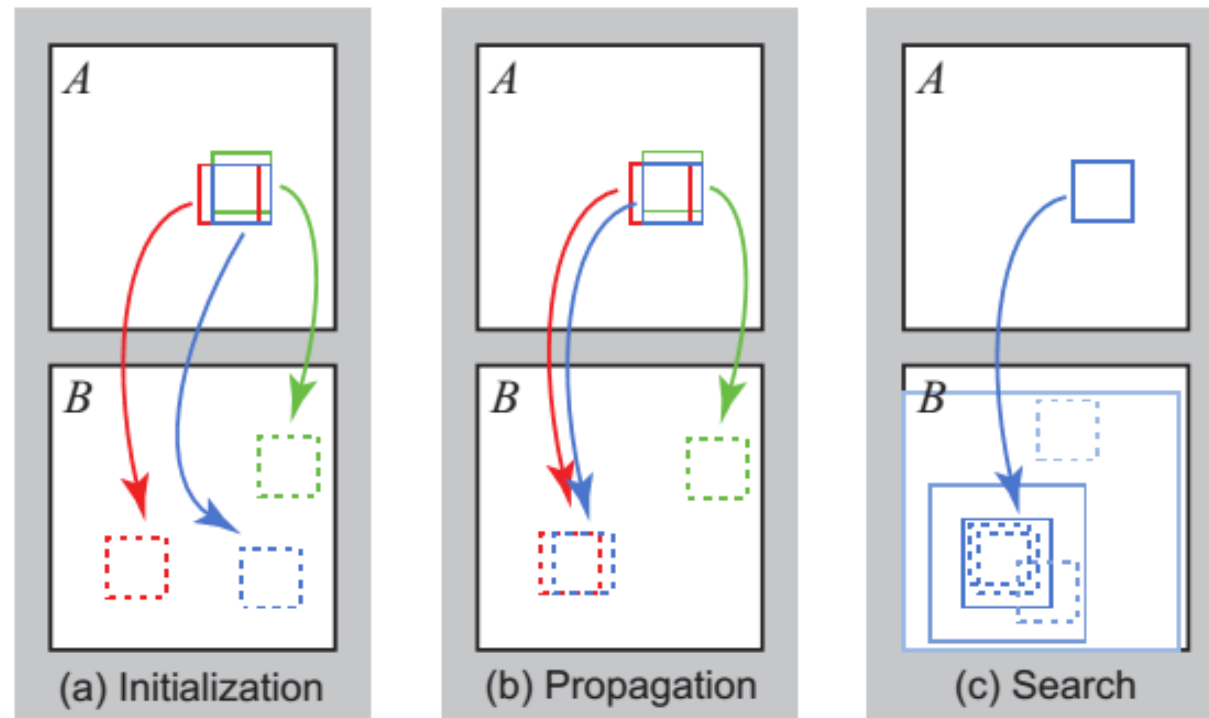


<https://www.youtube.com/watch?v=dgKjs8ZjQNg>

PatchMatch: a randomized correspondence algorithm for structural image editing [Barnes SIGGRAPH09]

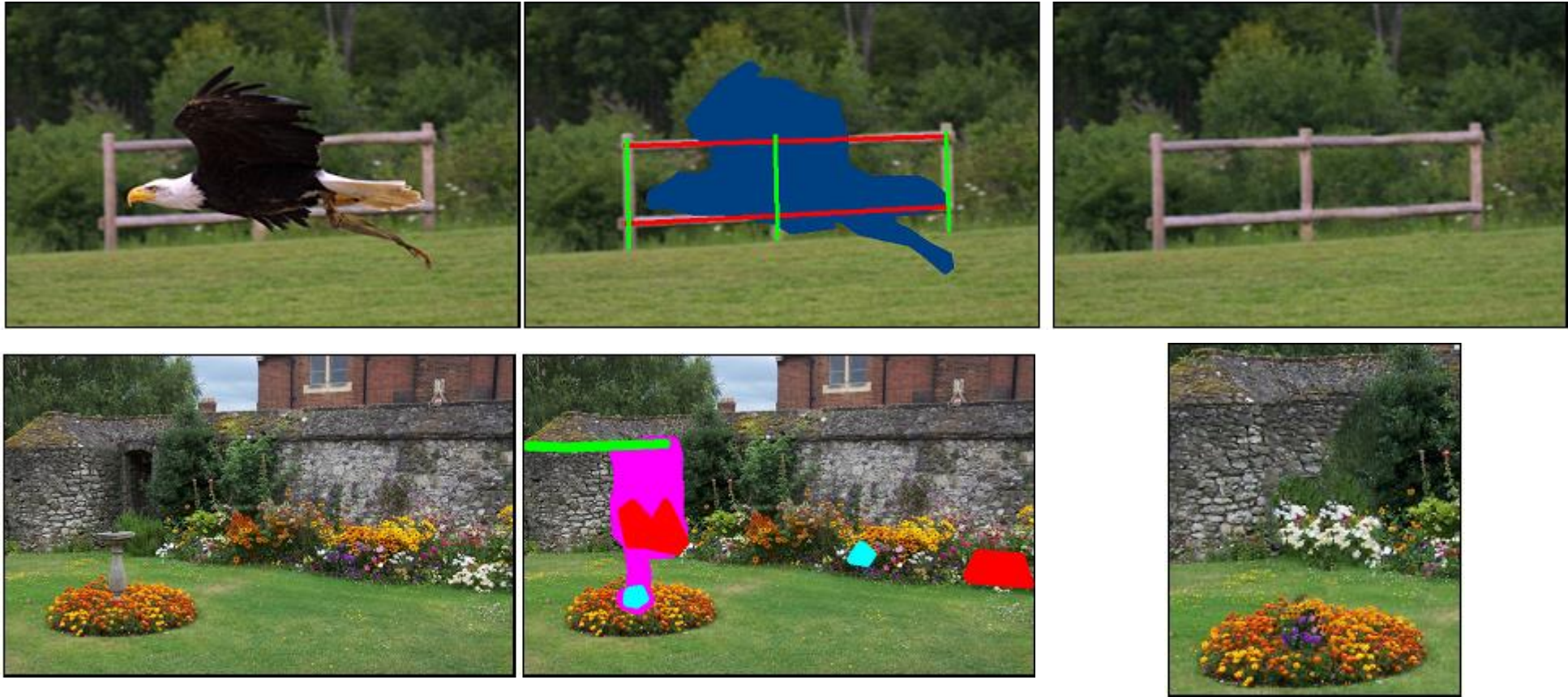
# Best bet: PatchMatch [Barnes09]

- Randomly initialize matches
- Update matches in scanline order
  - **Propagation:**  
Accept either left or above match if it's better than the current match
  - **Random Search:**  
Try a few random matches; accept if it's better than the current match
- Demo



# Extensions & applications

# Synthesis control by limiting the search space



- Output pixels with markers only match with input pixels with the same markers



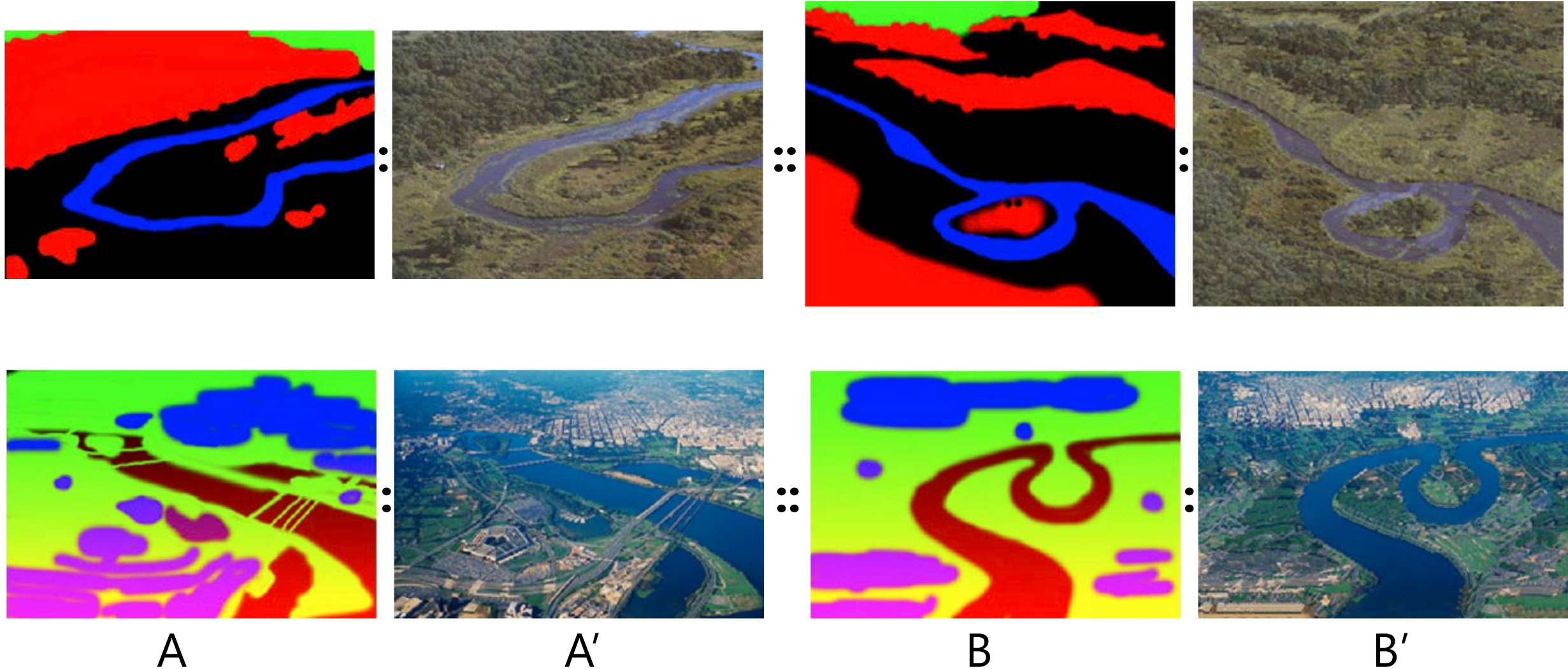
# Image Analogies [Hertzmann01]



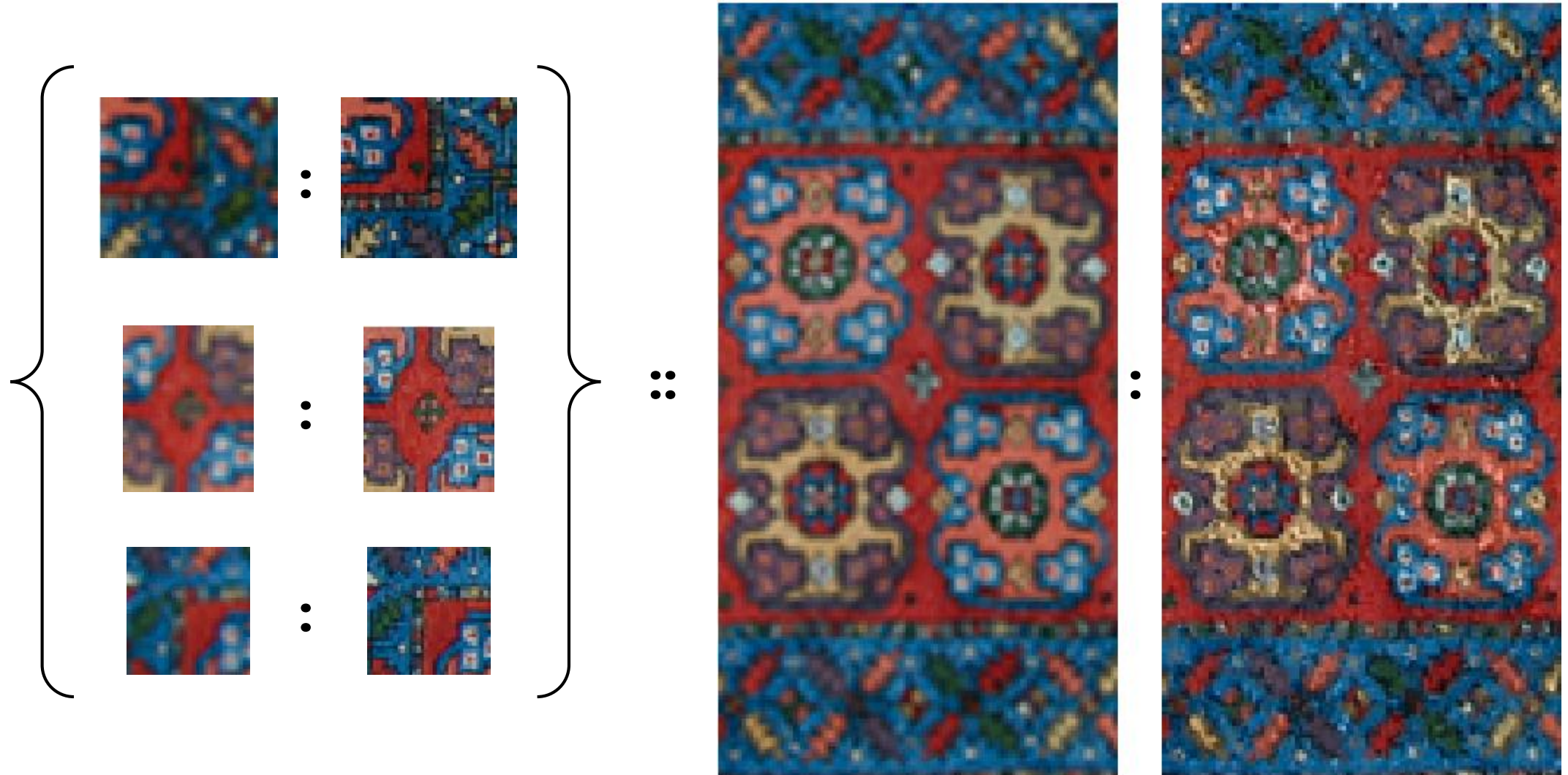
- Simulate arbitrary image filters using texture synthesis
- Variety of applications possible with this formulation



# Image Analogies – Texture by Numbers

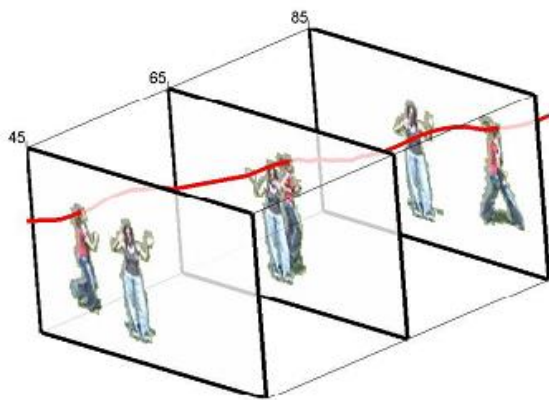


# Image Analogies – Super Resolution





# Removal of objects in videos



Frame 8



Frame 22



Frame 29



Frame 36



Frame 43



Frame 57

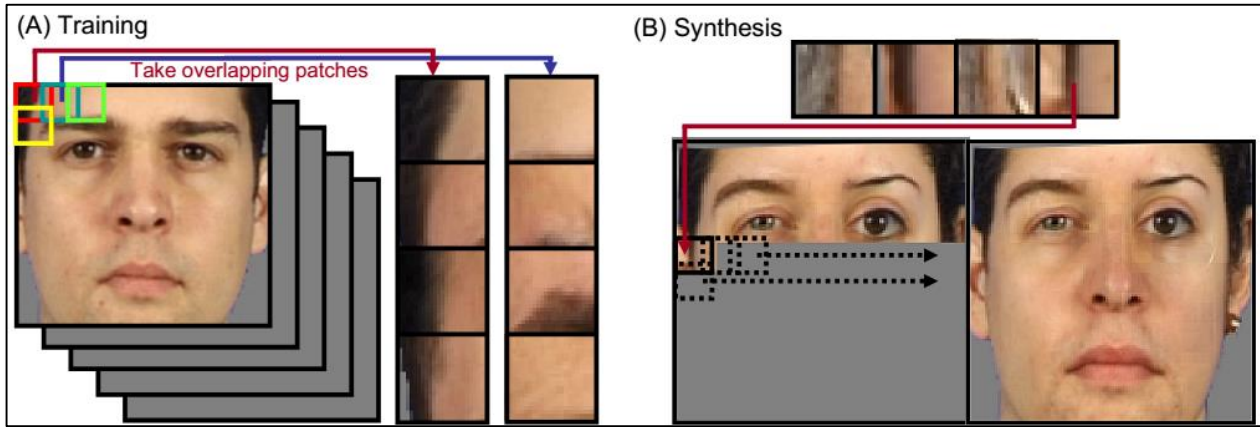


(a) Input sequence

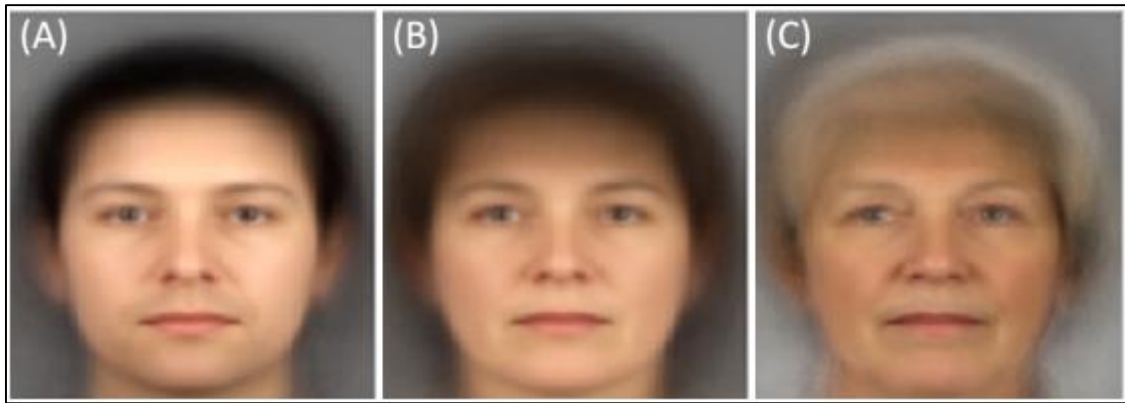
(b) Erasing the occluding person

(c) Spatio-temporal completion

# Random synthesis of face images [Mohammed09]

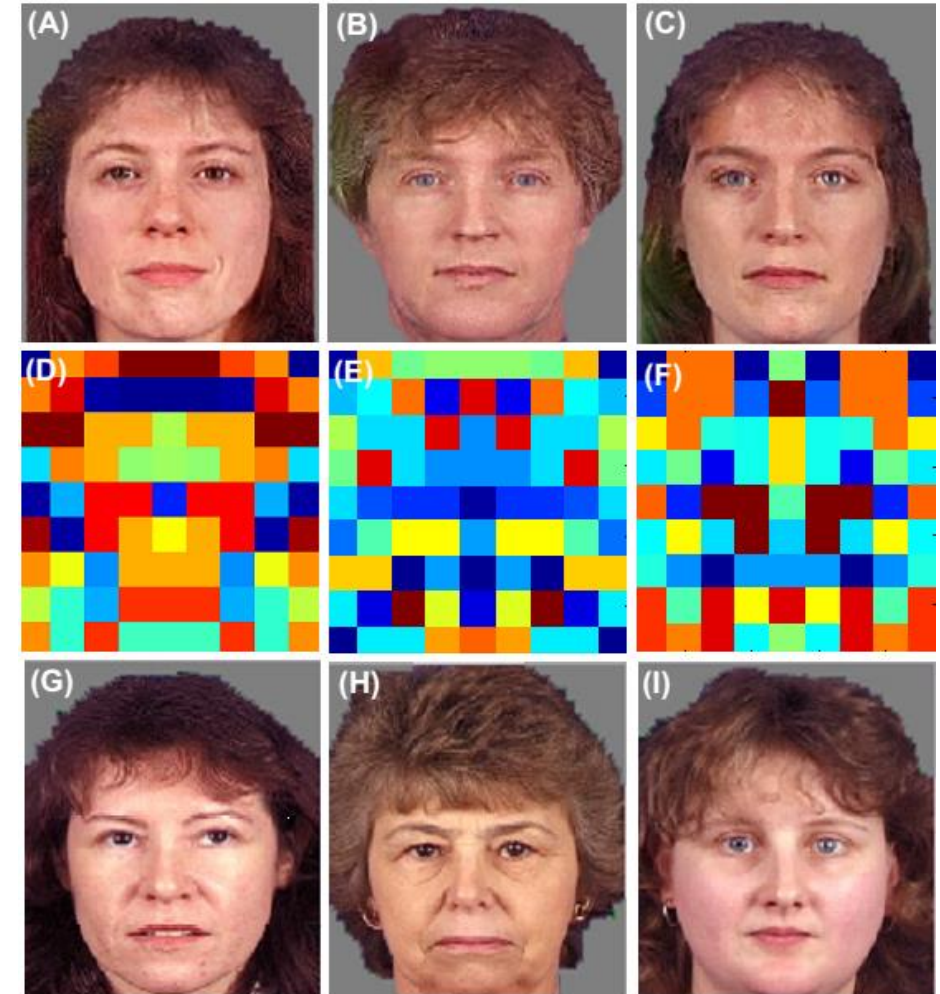


Naïve synthesis from face images with positional alignment



Parametric model for "average faces"

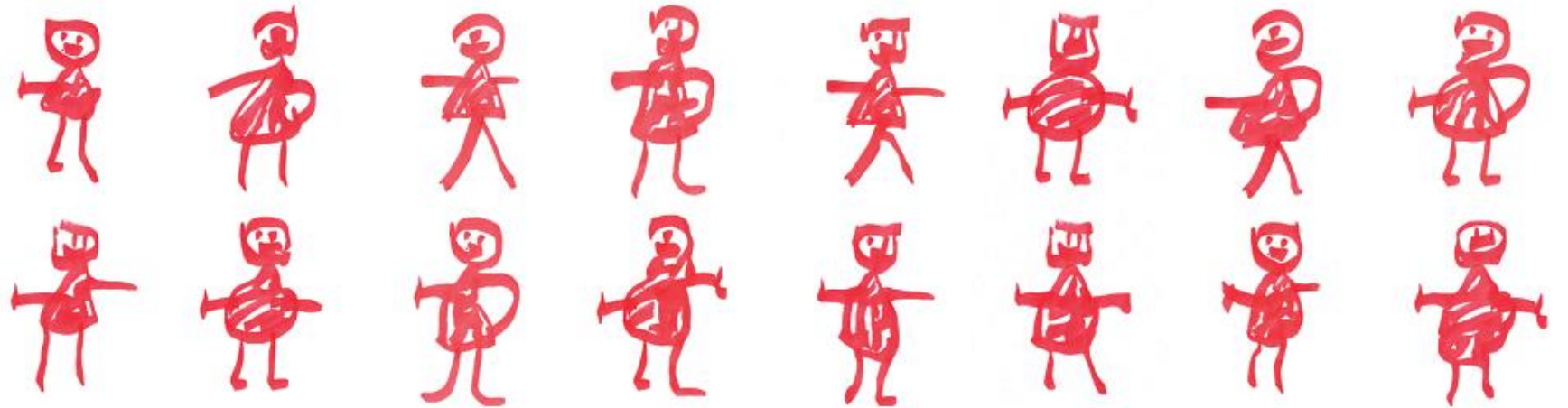
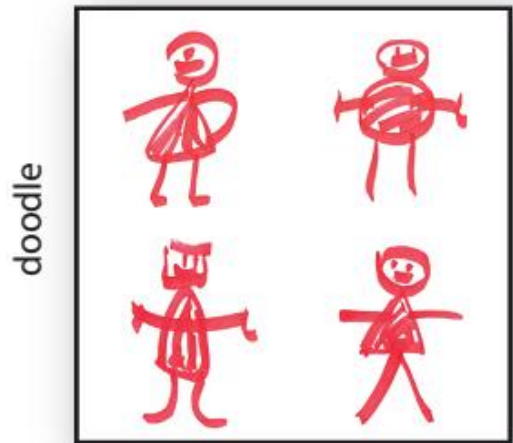
Synthesis result



Training images closest to synthesis results



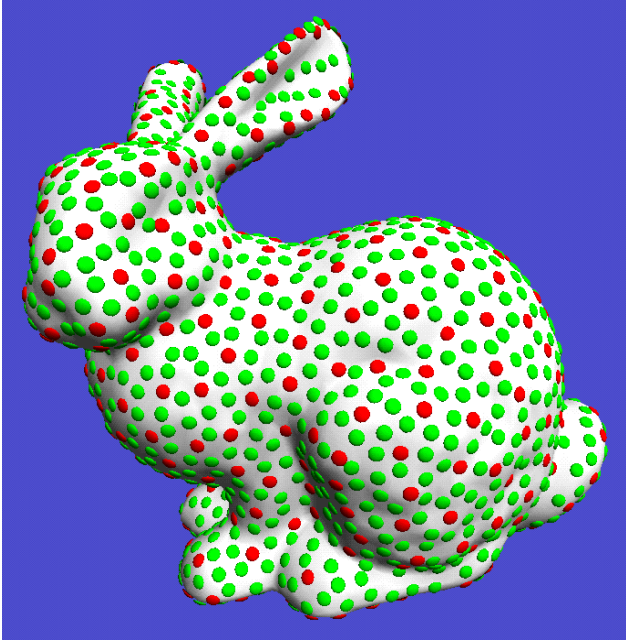
# Random synthesis of structured images [Risser10]



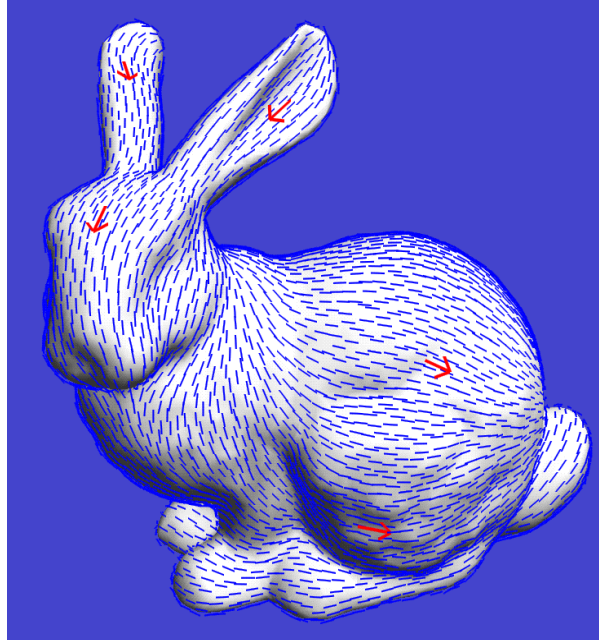


# Texture synthesis for 3D graphics

# On-surface texture synthesis [Wei01; Turk01]



Uniform sample points

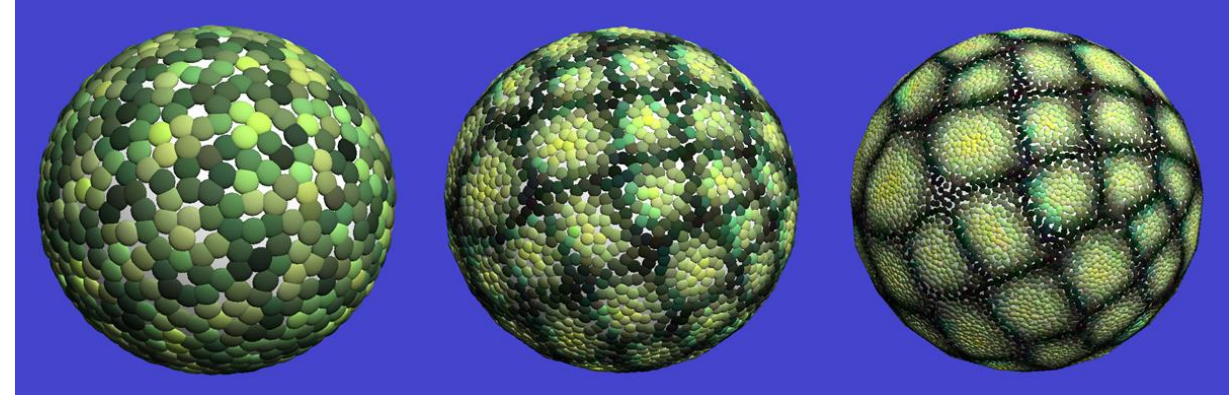


Vector field

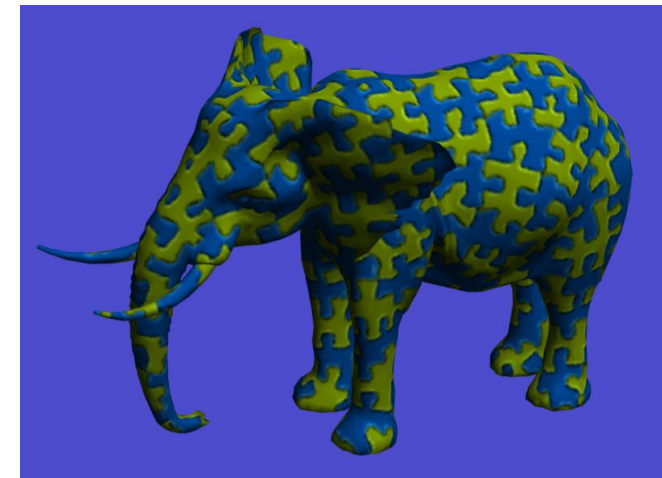
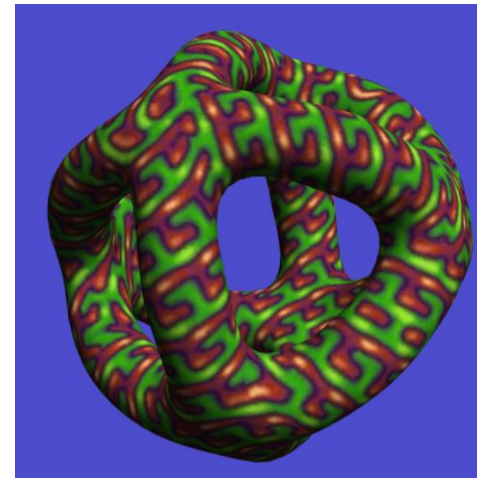
- Fundamentally equivalent to synthesizing texture images over UV parameter space

Texture synthesis over arbitrary manifold surfaces [Wei SIGGRAPH01]

Texture synthesis on surfaces [Turk SIGGRAPH01]

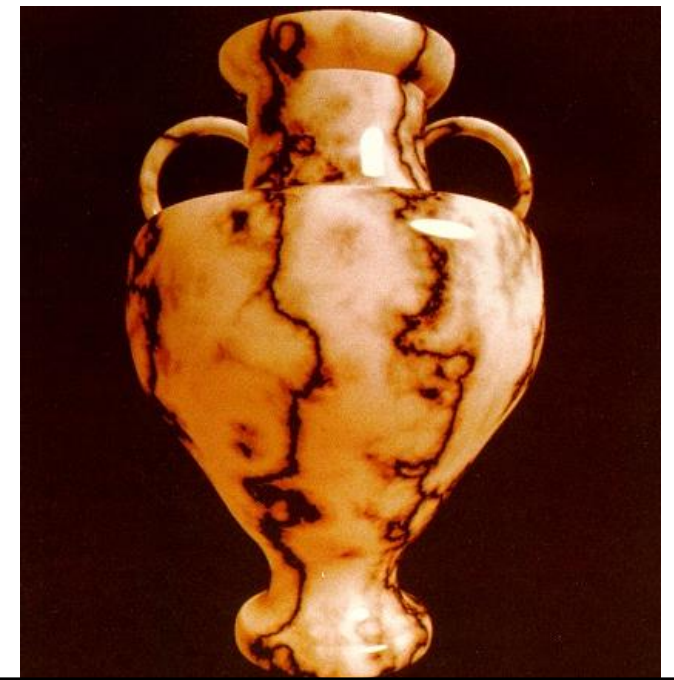


Multiresolution synthesis

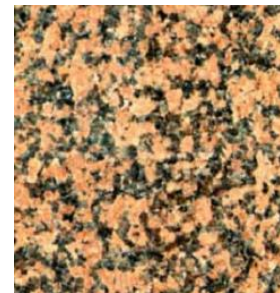


# Solid textures

- Represent texture as 3D volume (e.g. voxel) of RGB
  - RGB color directly obtained from XYZ coord  
→ easy to use!
- Early methods
  - Combine noise functions, tweak parameters
  - Automatic example-based synthesis using statistical approaches
    - Limited to noise-like textures



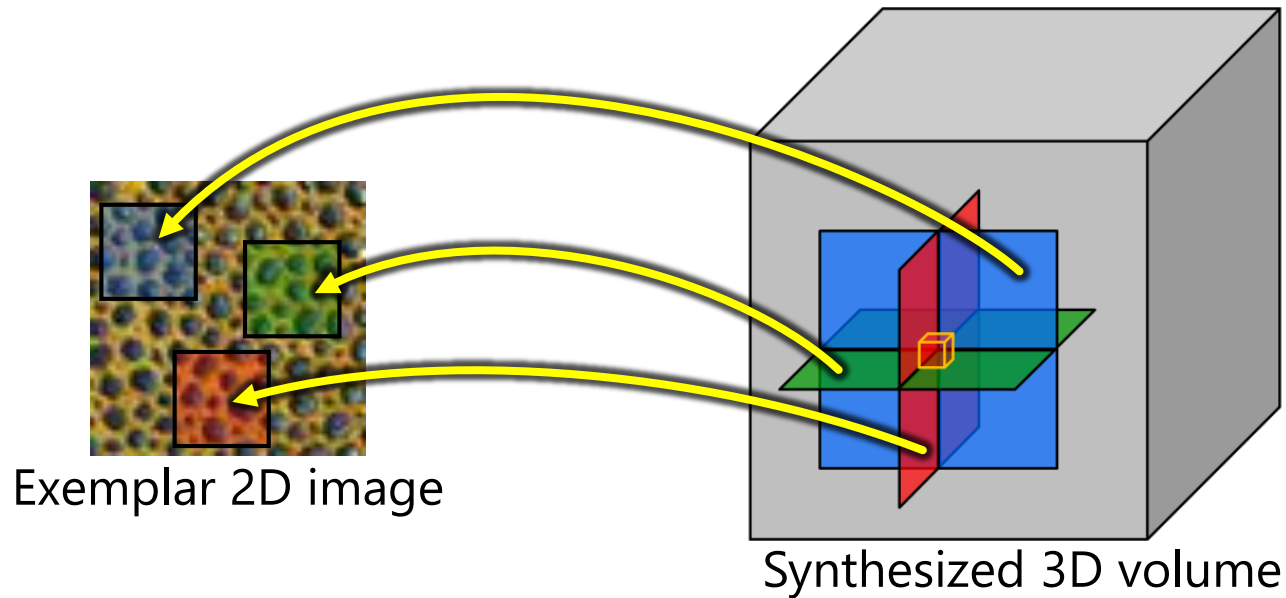
$marble(x, y, z)$   
 $= colormap(\sin(x + \text{noise}(x, y, z)))$





# Solid texture synthesis by optimization

- Almost straightforward generalization of 2D version [Kwatra05] to 3D



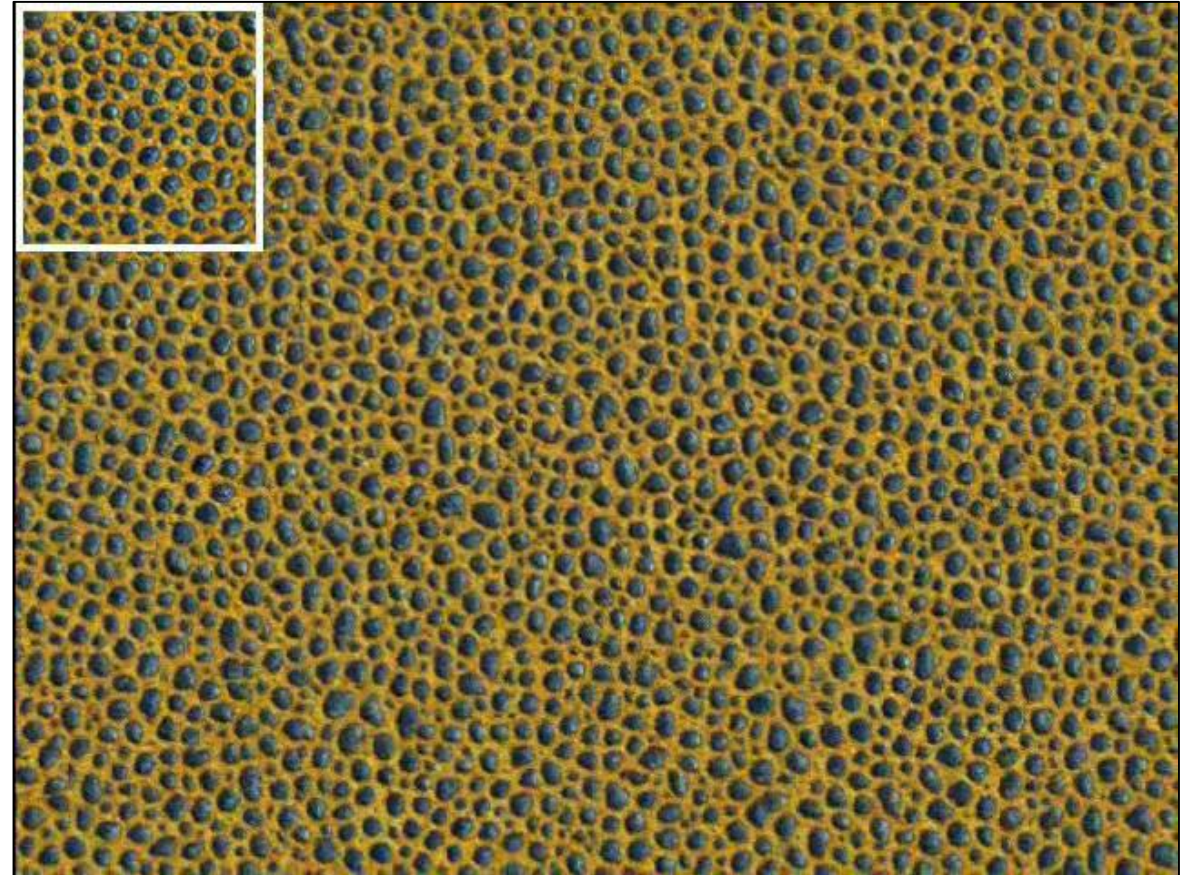
(Some tricks needed for better quality)





# Fast on-demand synthesis using GPU parallelism [Lefebvre05]

- Basic idea similar to [Kwatra05]
  - Key technique: precomputation + parallel independent processing
- Synthesize only when drawing  
= Reduced memory consumption  
→ suited for games



# On-demand synthesis specific to façade images [Lefebvre10]

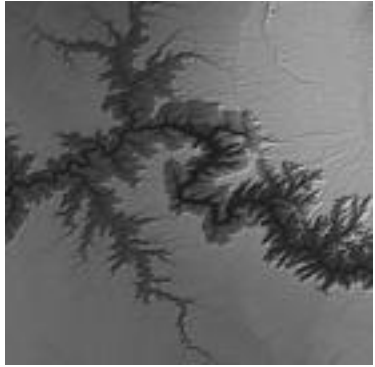
- Precompute horizontal/vertical seams → combine at runtime on GPU



# Applications of texture synthesis outside image processing



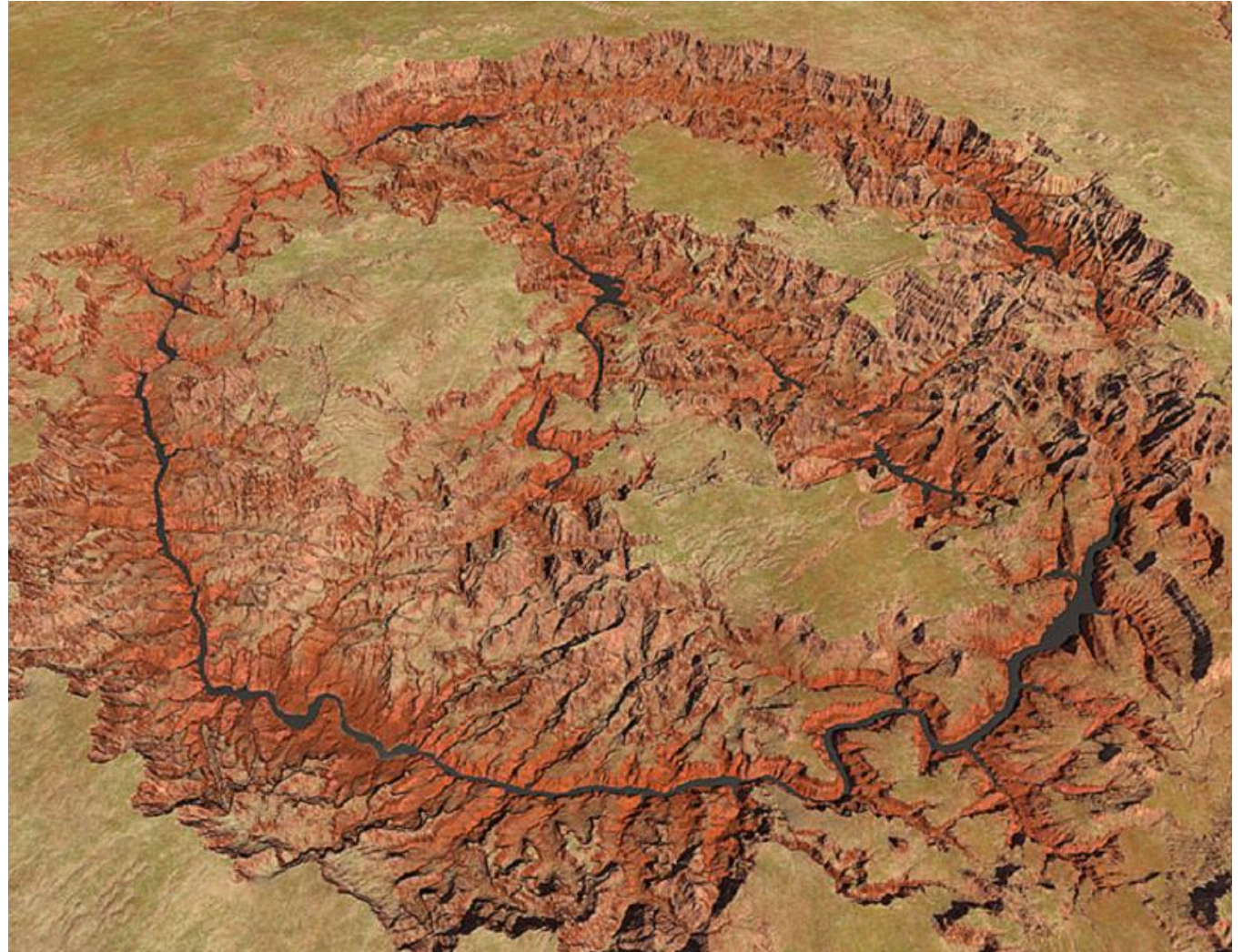
# Terrain (height field) synthesis [Zhou07]



Geographical data



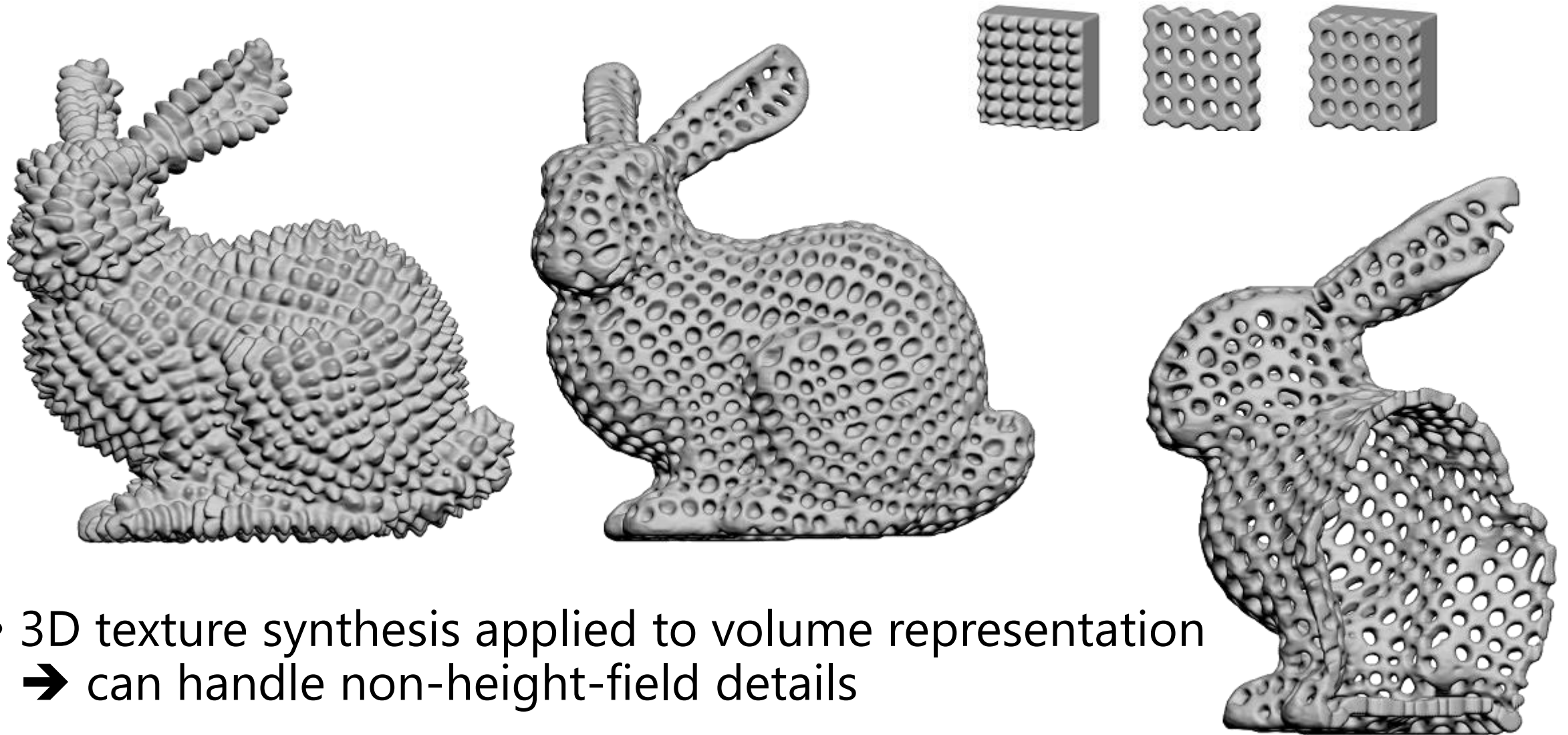
User's sketch



Synthesis result

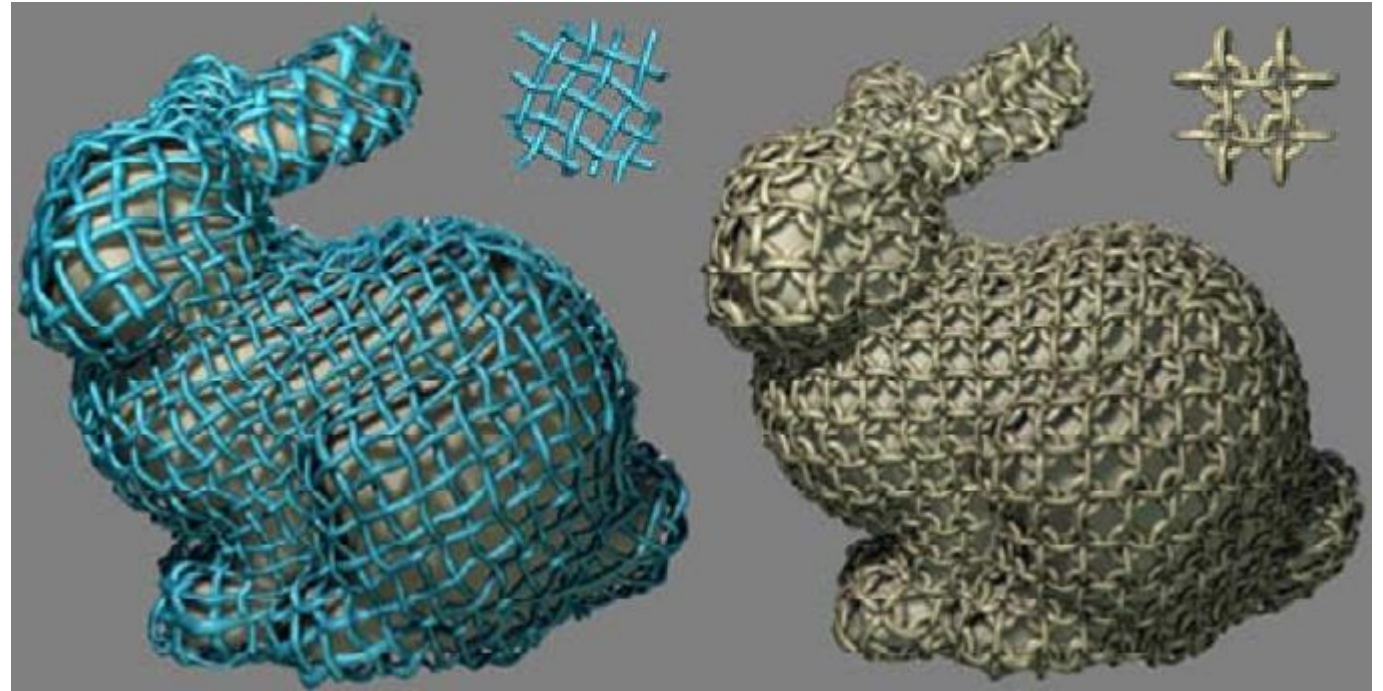
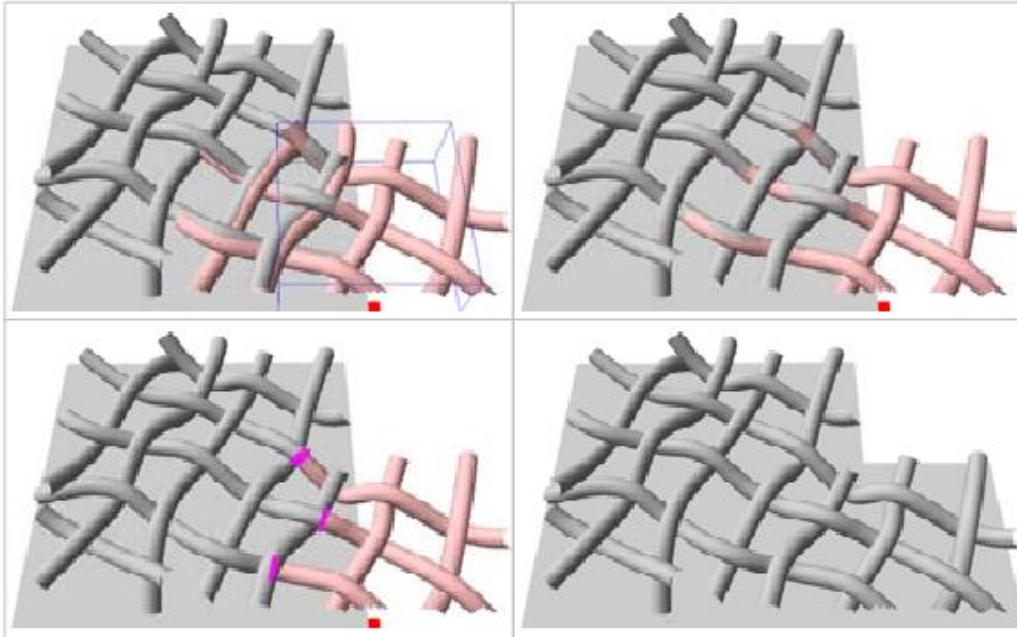


# Synthesis of surface details [Bhat04]



- 3D texture synthesis applied to volume representation  
→ can handle non-height-field details

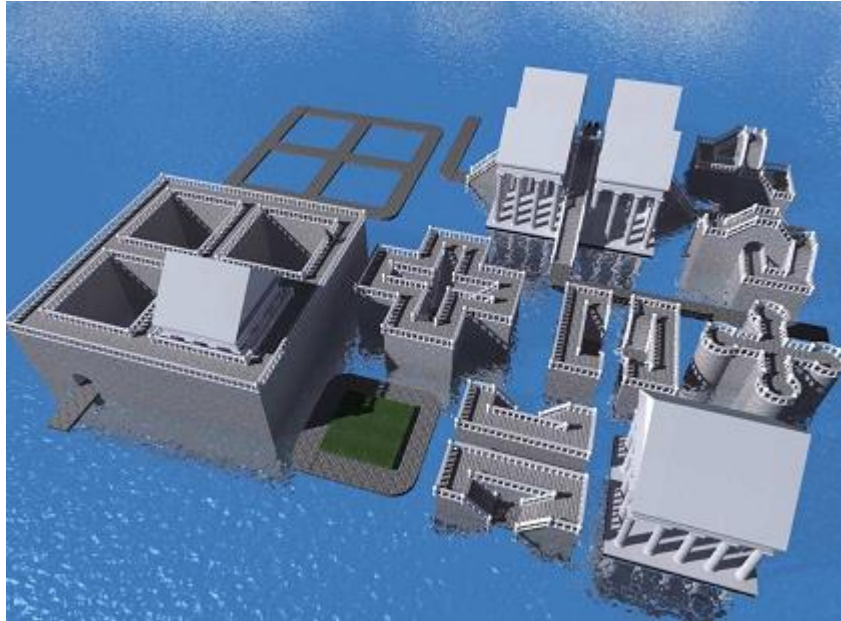
# Mesh Quilting [Zhou06]



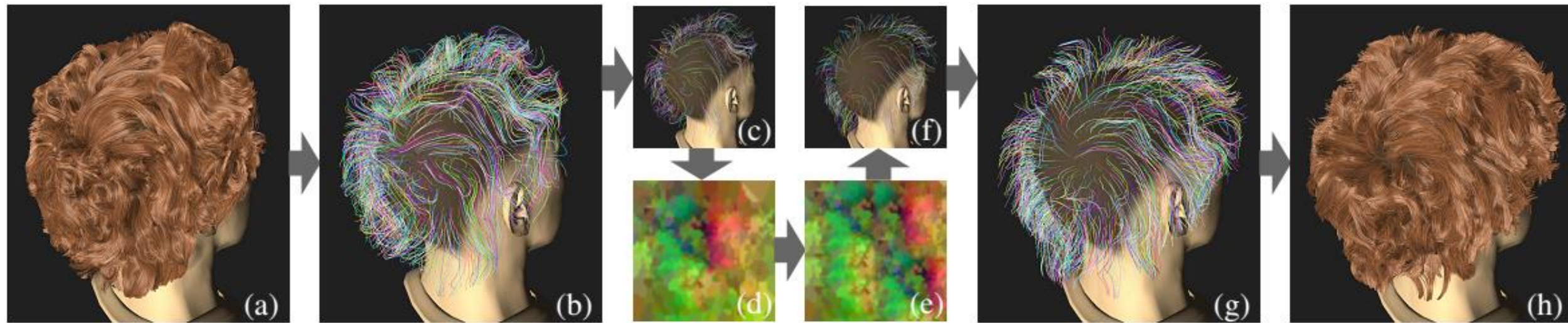
- Careful stitching of neighboring triangle meshes



# Synthesis of architectural models [Merrell07]



# Hair synthesis [Wang09]

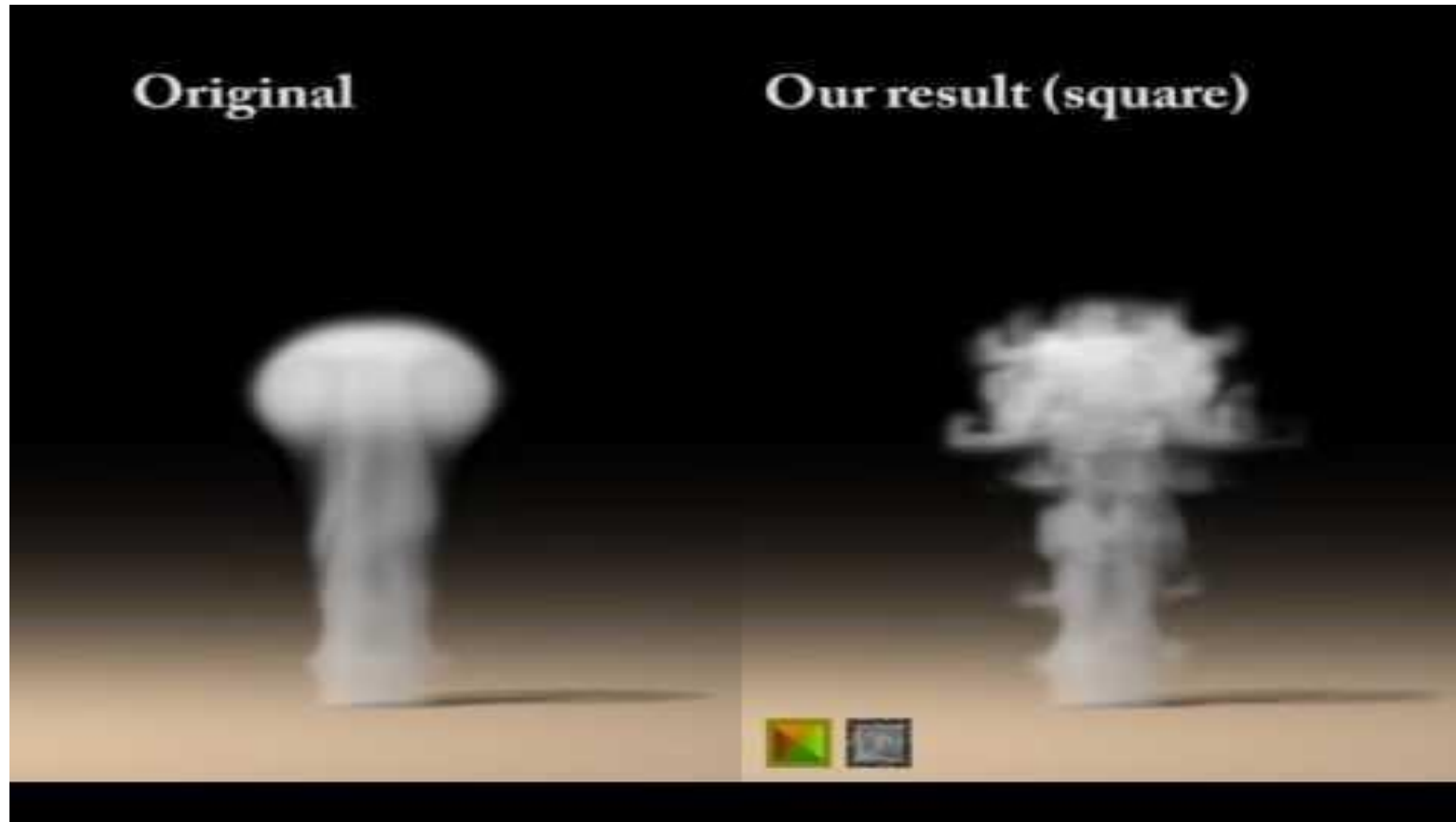


- Single hair strand = 3D polyline with  $N$  vertices =  $3N$  dim vector  
➔ Regarding this as color, apply texture synthesis

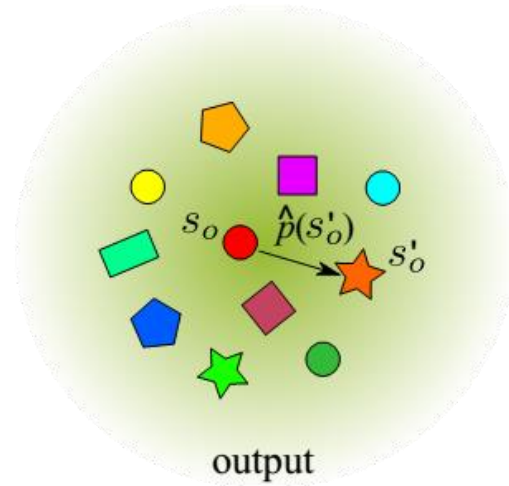
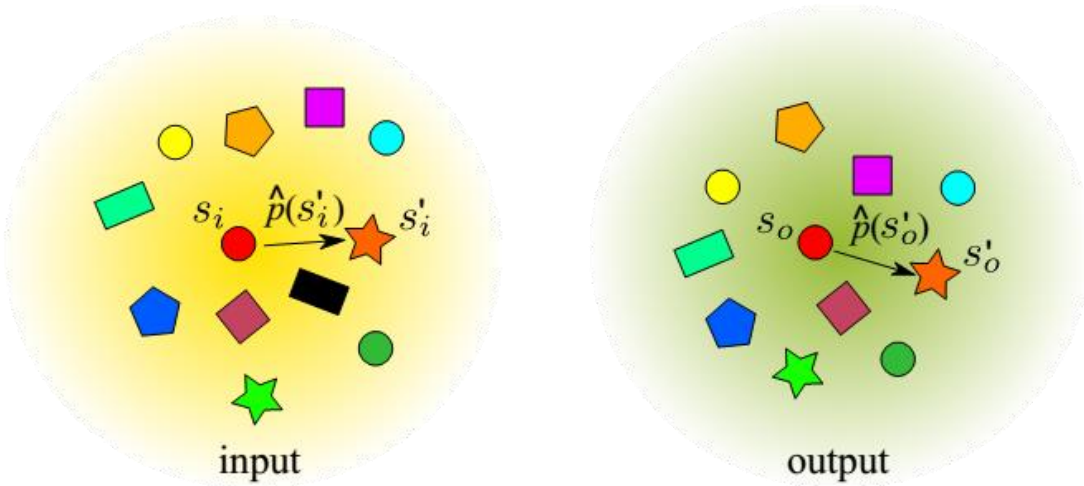


# Synthesis of artistic vortices for fluids [Ma09]

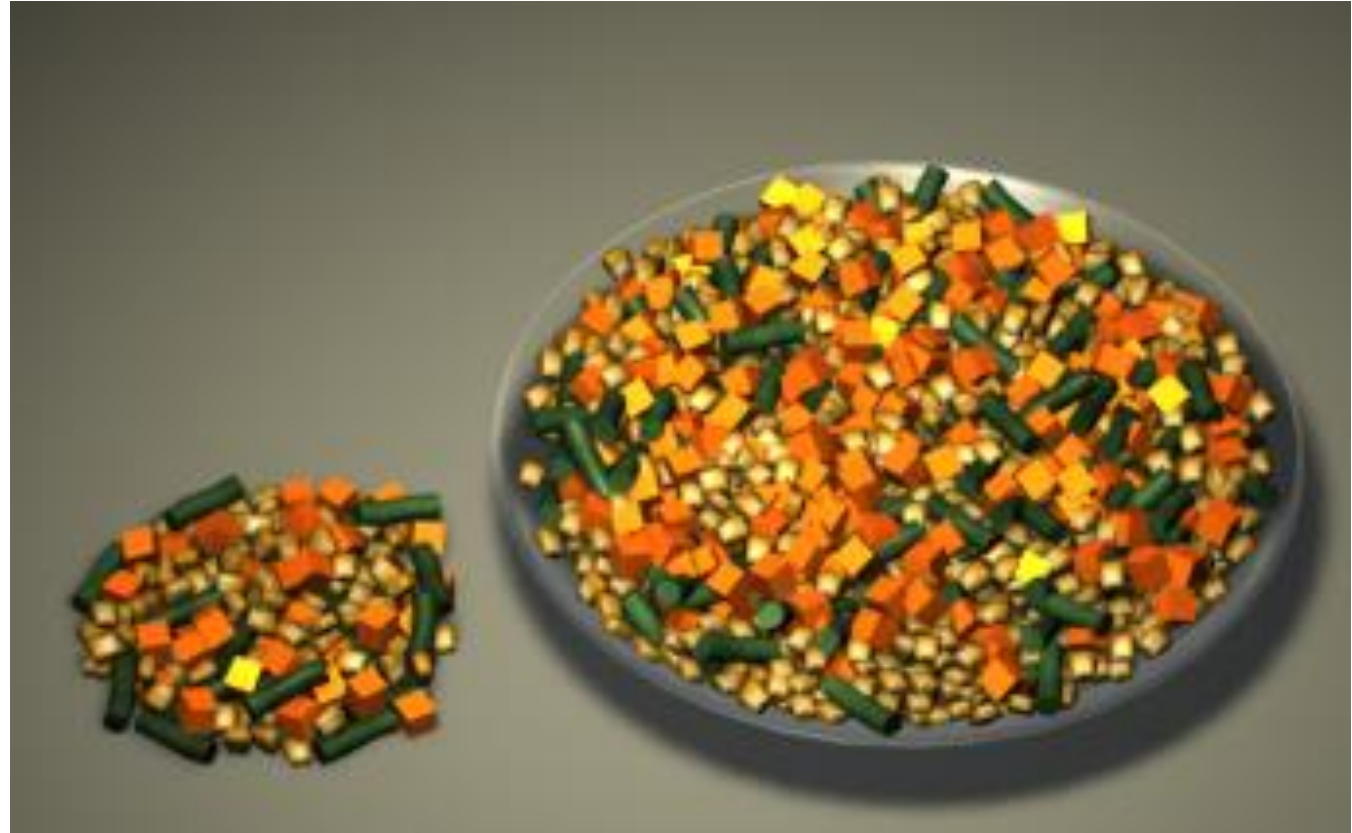
- Synthesize detailed vortex velocity field along input low-res velocity field
  - Regarding 2D/3D velocity vector as colors, apply texture synthesis



# Synthesis of element arrangement [Ma11]



- Define similarity between distributions of sample points
- Optimization algorithm similar to [Kwatra05]



# Pointers

- Existing implementations
  - <http://www2.mta.ac.il/~tal/ImageCompletion/ImageCompletion1.01.zip>
  - [http://www.cs.princeton.edu/gfx/pubs/Barnes\\_2009\\_PAR/patchmatch-2.1.zip](http://www.cs.princeton.edu/gfx/pubs/Barnes_2009_PAR/patchmatch-2.1.zip)
  - <http://research.nii.ac.jp/~takayama/cggems12/cggems12.zip>
- Surveys
  - State of the art in example-based texture synthesis [Wei EG09STAR]
  - Solid-Texture Synthesis; A Survey [Pietroni CGA10]