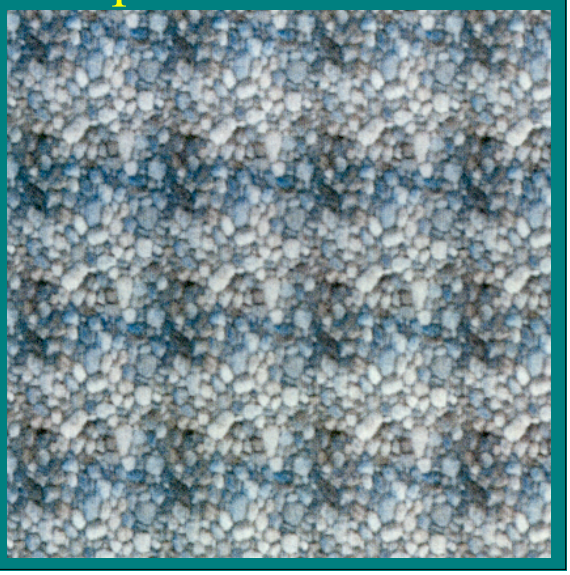
CMSC 435
Texture 2

Penny Rheingans
UMBC

Texture Synthesis

- Basic problem
  - You have a sample of some texture
  - You want to be able to generate more of the same
- Applications
  - Create new textures for objects
  - Infill missing regions

Sample Replication



Multiresolution Synthesis

- Jeremy De Bonet, SIGGRAPH '97,
- Analyze sample to generate global conditional distribution function
  - joint features
  - multiple resolutions

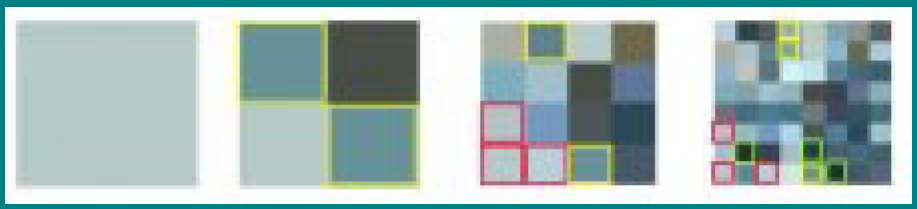
Synthesize texture with similar joint features

- Uniform sampling

- Frequency sampling

- De Bonet 97

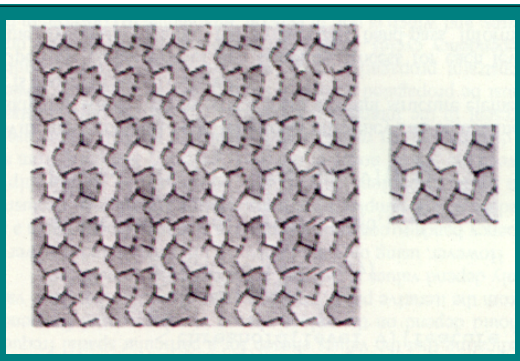
Multiresolution Synthesis

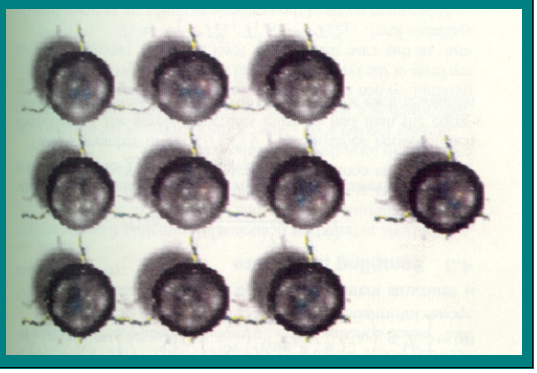


De Bonet '97

- De Bonet S97

Results



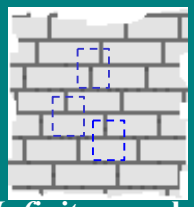


- De Bonet S97

Texture Sampling

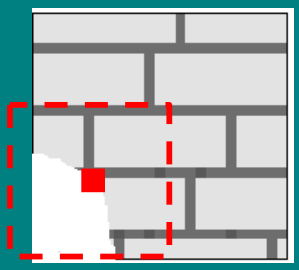
- Alexei Efros and Thomas Leung, ICCV '99, Texture Synthesis by Non-parametric Sampling
- Their goals:
  - preserve local structure
  - model wide range of real textures
  - ability to do constrained synthesis
- Their method:
  - Texture is "grown" one pixel at a time
  - conditional pdf of pixel given its neighbors synthesized thus far is computed directly from the sample image

Synthesizing One Pixel



***Infinite* sample image**

SAMPLE

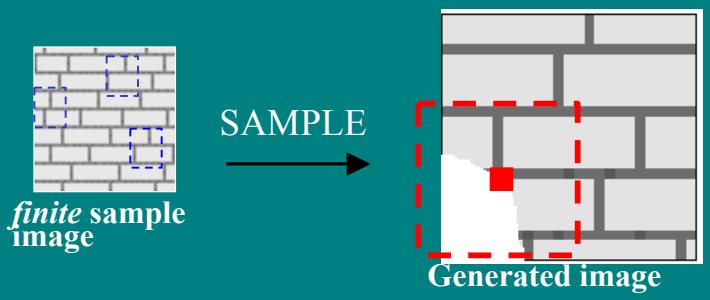


Generated image

- 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

Efros and Leung '99

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

Efros and Leung '99

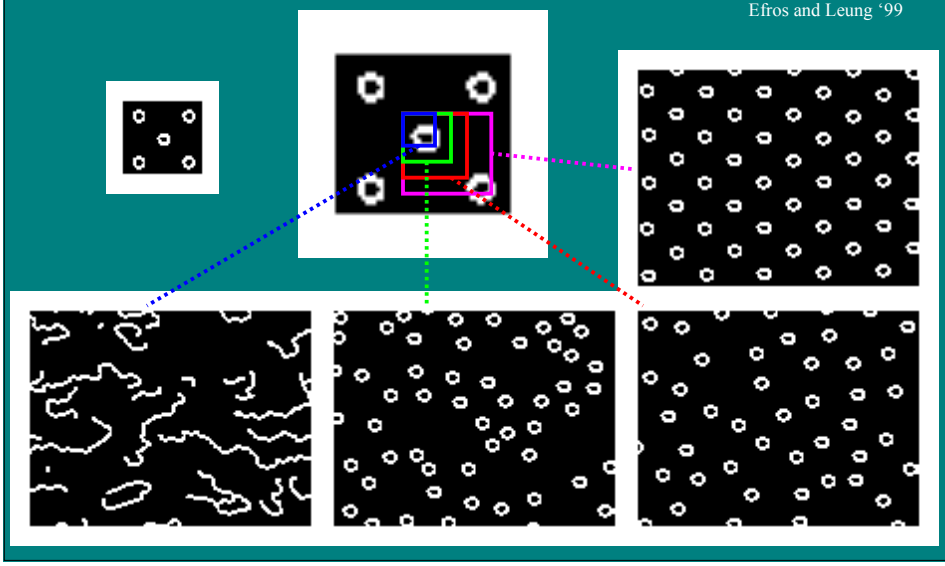
Growing Texture

- Starting from the initial configuration, we "grow" the texture one pixel at a time
- The size of the neighbourhood window is a parameter that specifies how stochastic the user believes this texture to be
- To grow from scratch, we use a random 3x3 patch from input image as seed

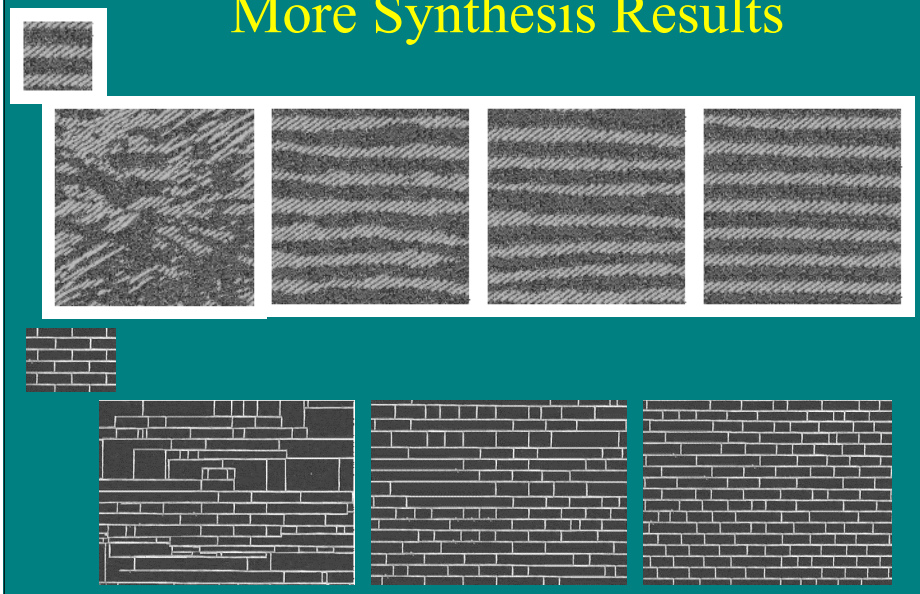
Efros and Leung '99

Randomness Parameter

Efros and Leung '99



More Synthesis Results

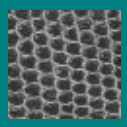


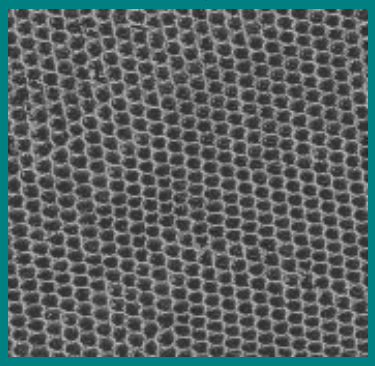
Increasing window size

Efros and Leung '99

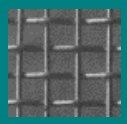
Brodatz Results

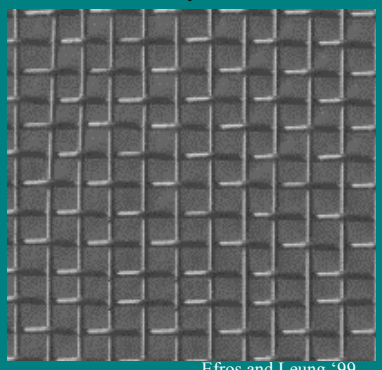
reptile skin





aluminum wire

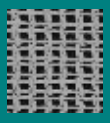


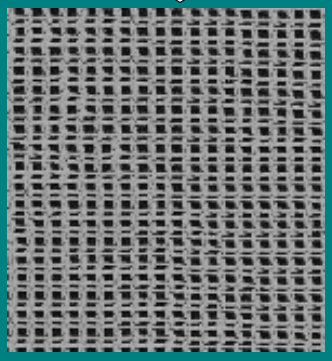


Efros and Leung '99

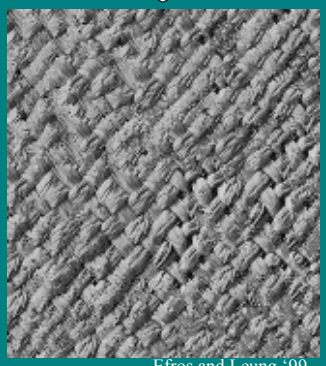
More Brodatz Results

french canvas





rafia weave



Efros and Leung '99

More Results

wood

granite

Efros and Leung '99

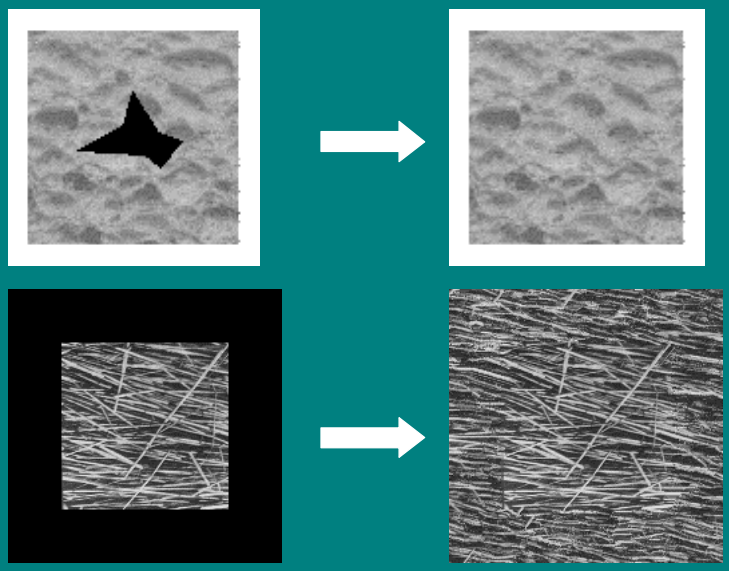
More Results

white bread

brick wall

Efros and Leung '99

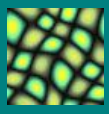
Constrained Synthesis

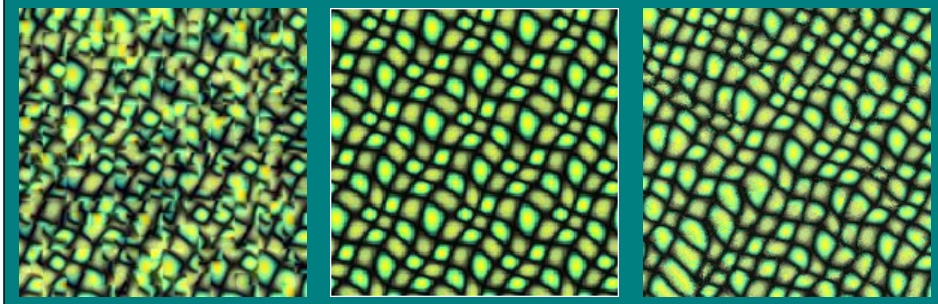


Efros and Leung '99

Visual Comparison

Synthetic tilable texture

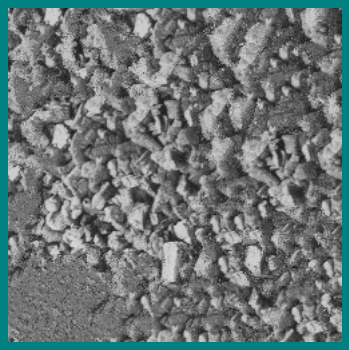




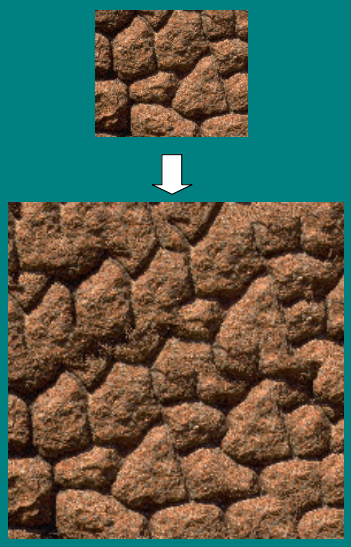
[DeBonet, '97] **Simple tiling** **Our approach**

Efros and Leung '99

Failure Cases



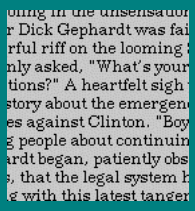
Growing garbage



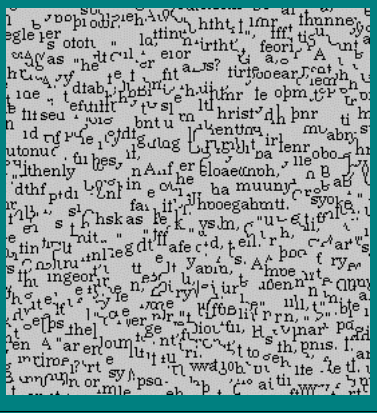
Verbatim copying

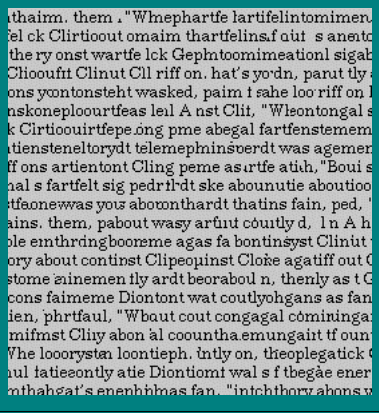
Efros and Leung '99

Homage to Shannon



Efros and Leung '99





Characteristics

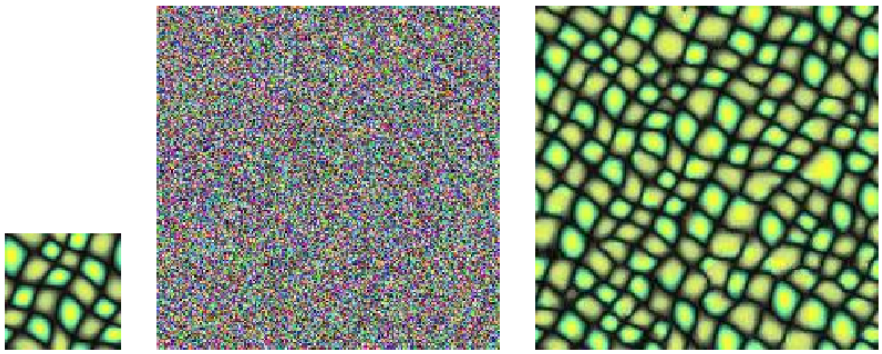
- High quality results which capture much of structure of input image
- Really slow
  - 192x192 image from 64x64 sample == 1941 sec

TVG Texture Synthesis

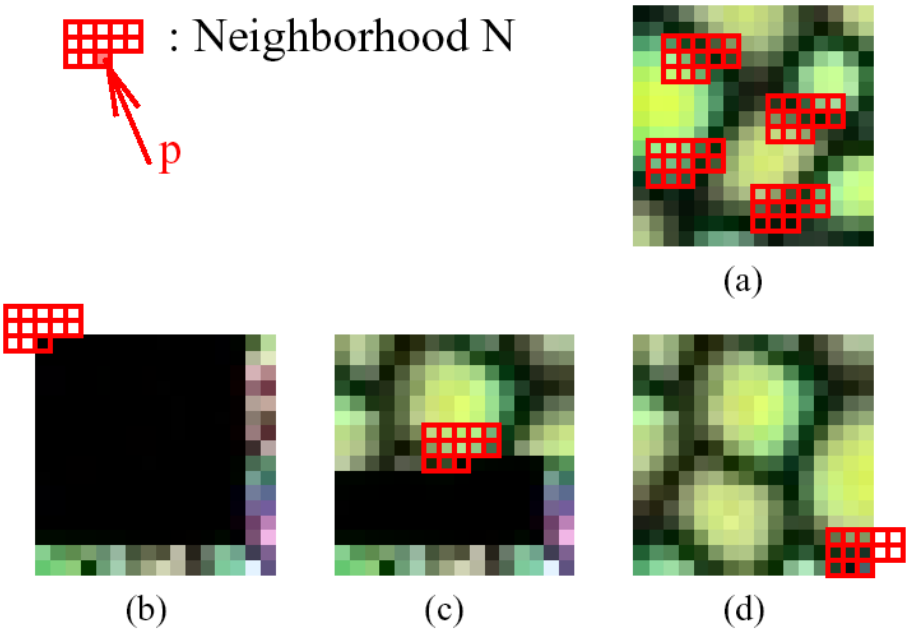
- Li-Yi Wei and Marc Levoy, SIGGRAPH '00,
  - Fast Texture Synthesis using Tree-structured Vector Quantization,
- Characteristics
  - Multiresolution search approach
  - Accelerate with tree-structured vector quantization

Basic Approach

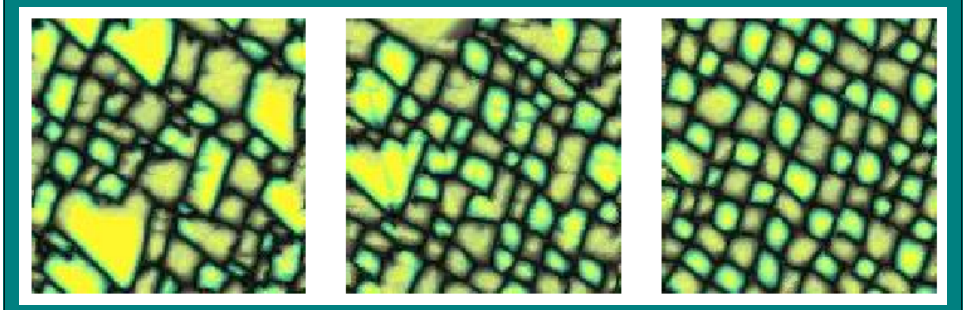
- Transform random input texture to output



Wei and Levoy '00



Effect of Neighborhood Size



5x5 7x7 9x9

Wei and Levoy '00

Multiresolution Neighborhood

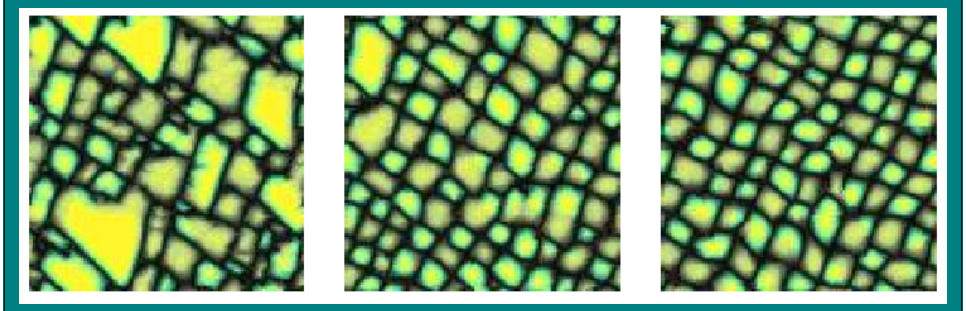
	O	O	O	O	O		
	O	O	O	O	O		
	O	O	X				

Q	Q	Q	
Q	Y	Q	
Q	Q	Q	

Wei and Levoy '00

Effect of # of Pyramid Levels

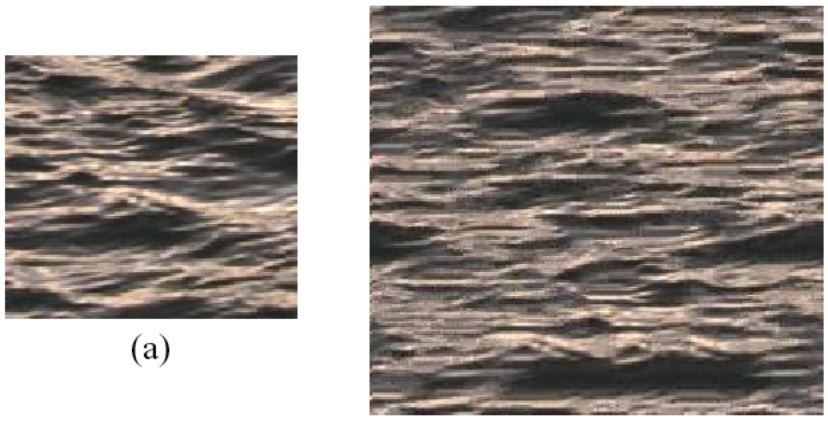
- Quality improves as number of levels increases



1 2 3

Wei and Levoy '00

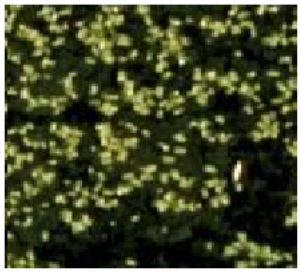
Results



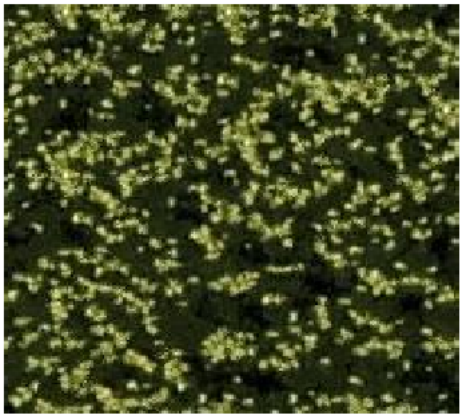
(a)

Wei and Levoy '00

Results



(b)

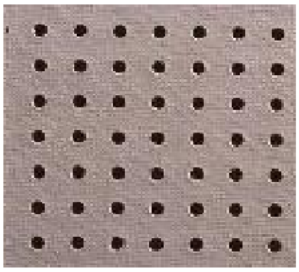


Wei and Levoy '00

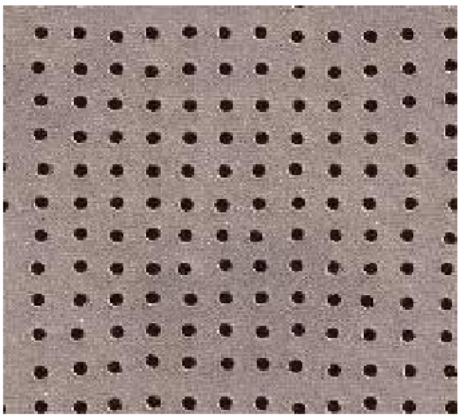
Results

(f)

Wei and Levoy '00



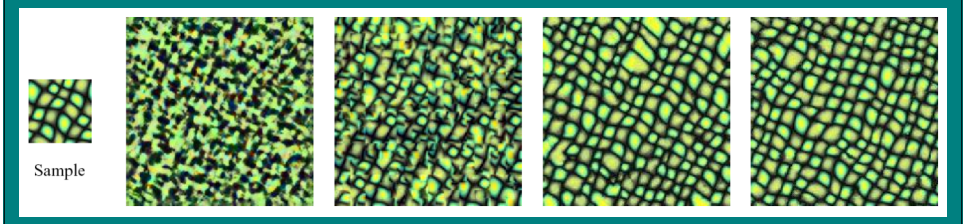
(g)



Wei and Levoy '00

(i)

Wei and Levoy '00

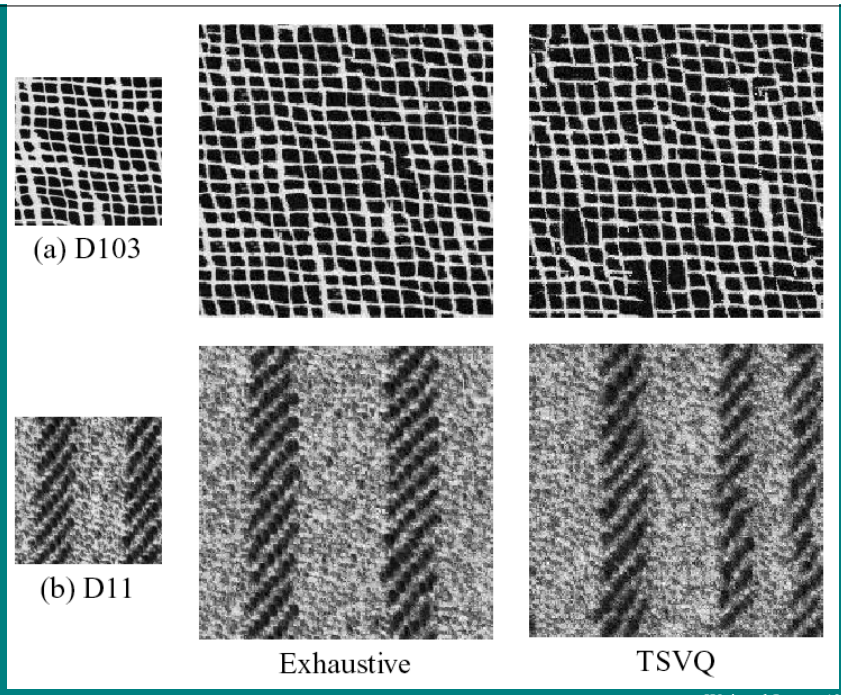


Heeger & Bergen De Bonet Efros & Leung Wei & Levoy

Wei and Levoy '00

Acceleration

- Limit neighborhood searched exhaustively
  - Causal neighborhood
  - Represent as vector
  - 192x192 from 64x64 sample == 503 sec
- Recast problem as nearest-neighbor search
- Accelerate with TSVQ
  - Generate binary tree-structured codebook from training vectors
  - For each output pixel, search tree for nearest match
  - 192x192 from 64x64 sample == 24 sec



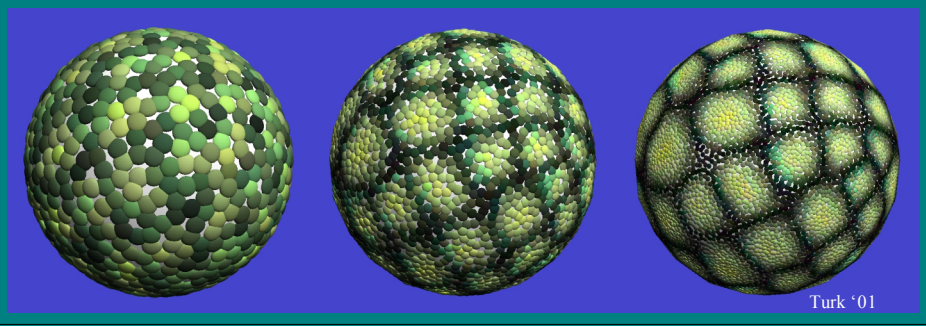
Wei and Levoy '00

Synthesis on Arbitrary Surfaces

- Greg Turk, SIGGRAPH '01
  - Texture Synthesis on Surfaces
- Basic Approach
  - Modify hierarchical sampling approach to replay image pyramid with mesh pyramid
  - Select color at a point by matching neighborhood in mesh
  - Resulting texture naturally follows shape of object

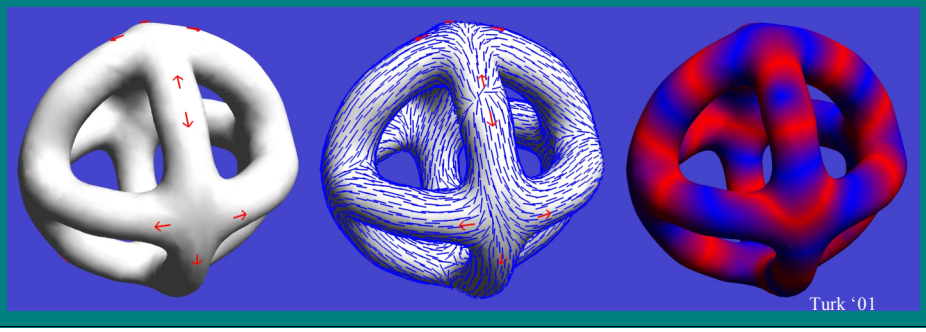
Mesh Hierarchy

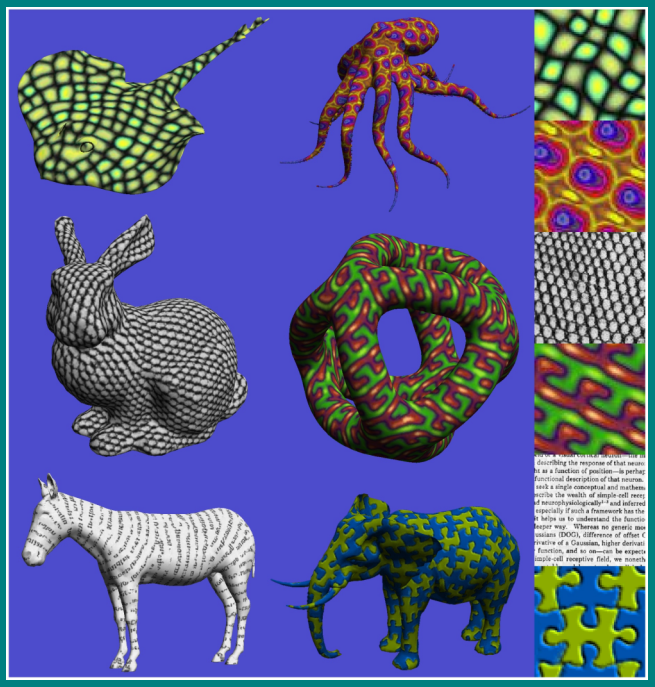
- Generate hierarchy of points from low to high density
- Connect points to create hierarchy of meshes
- Texture generated in coarse-to-fine manner



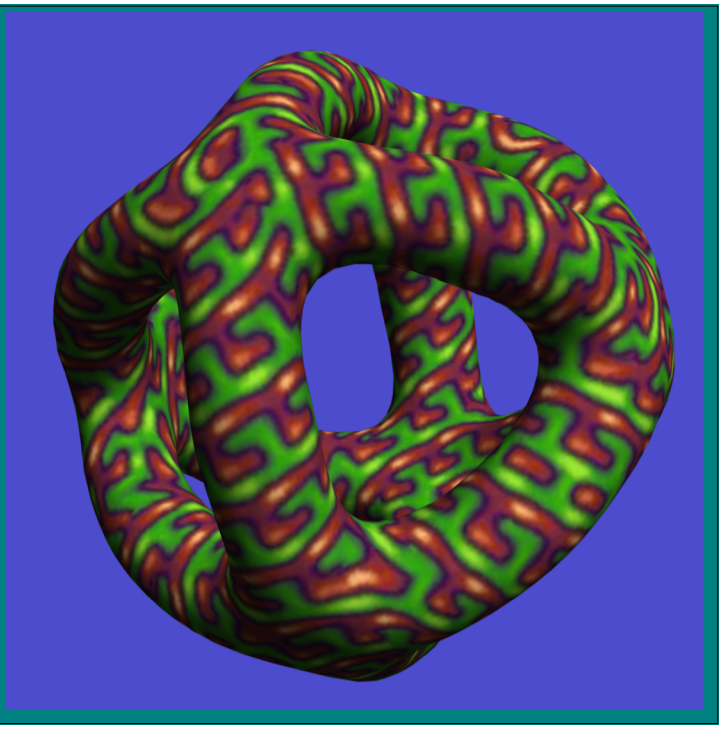
User-specified Vector Field

- Vector field indicates orientation of texture (order of point visitation)
- Texture grows along vector field





Turk '01



Turk '01


Turk '01


Turk '01

Image Analogies

- Aaron Hertzmann, Charles Jacobs, Nuria Oliver, Brian Curless, and David Salesin, SIGGRAPH 2001.
- Apply texture synthesis techniques to reproduce transformation operation from example

The Problem

- Given a pair of images A and A' (the *unfiltered* and *filtered source images*, respectively), along with some additional *unfiltered target image* B, synthesize a new *filtered target image* B' such that

  A : A' :: B : B'

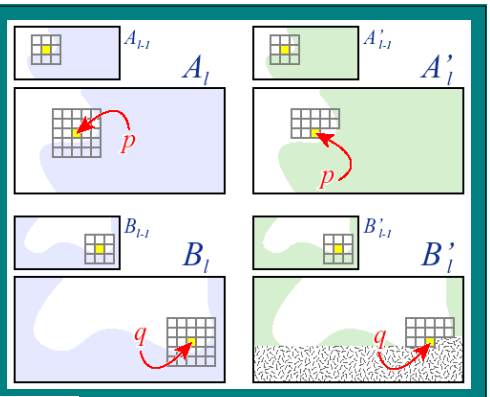
- Questions
  - How to measure similarity between source images (A, A')?
  - How to compare unfiltered images (A, B)?

Basic Process

- Training phase
  - Selection of source images
  - Annotation as necessary
  - Filter created
- Application phase
  - Filter applied to target

The Method

function CREATEIMAGEANALOGY(A, A', B):
  Compute Gaussian pyramids for A, A', and B
  Compute features for A, A', and B
  Initialize the search structures (e.g., for ANN)
  for each level ℓ, from coarsest to finest, **do**:
    for each pixel $q \in B'_\ell$, in scan-line order, **do**:
      $p \leftarrow$ BESTMATCH(A, A', B, B', s, ℓ, q)
      $B'_\ell(q) \leftarrow A'_\ell(p)$
      $s_\ell(q) \leftarrow p$
  return B'_L



function BESTMATCH(A, A', B, B', s, ℓ, q):
  $p_{\text{app}} \leftarrow$ BESTAPPROXIMATEMATCH(A, A', B, B', ℓ, q)
  $p_{\text{coh}} \leftarrow$ BESTCOHERENCEMATCH(A, A', B, B', s, ℓ, q)
  $d_{\text{app}} \leftarrow \|F_\ell(p_{\text{app}}) - F_\ell(q)\|^2$
  $d_{\text{coh}} \leftarrow \|F_\ell(p_{\text{coh}}) - F_\ell(q)\|^2$
  if $d_{\text{coh}} \leq d_{\text{app}}(1 + 2^{\ell - L}\kappa)$ **then**
    return p_{coh}
  else
    return p_{app}

Matching

- BestApproximateMatch
  - Approximate nearest neighbor (ANN)
  - Tree-structured vector quantization (TSVQ)
- BestCoherenceMatch
  - Pick match that best continues pixels already generated

Features

- RGB didn't work well
  - Not enough good matches
- Alternatives
  - Luminance (chromatic info just copied from B)
    - Optional luminance remapping
  - Orientable filters

- Blur

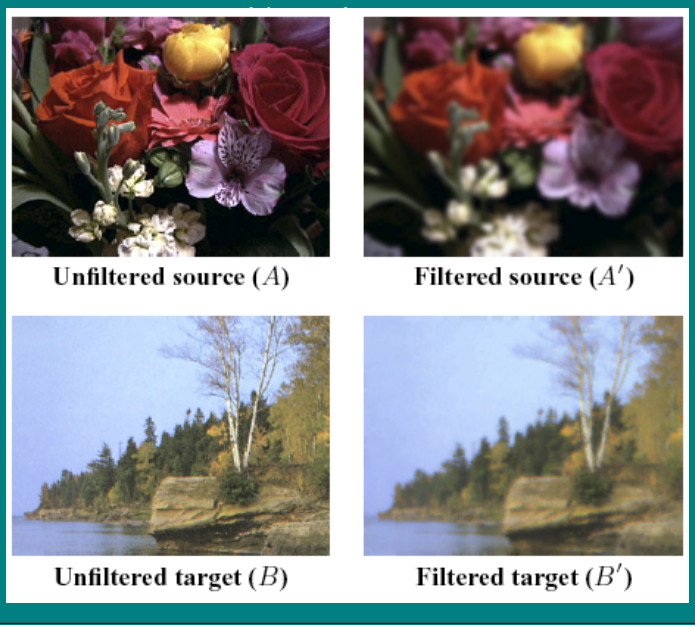
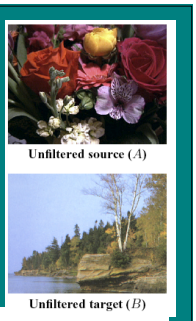
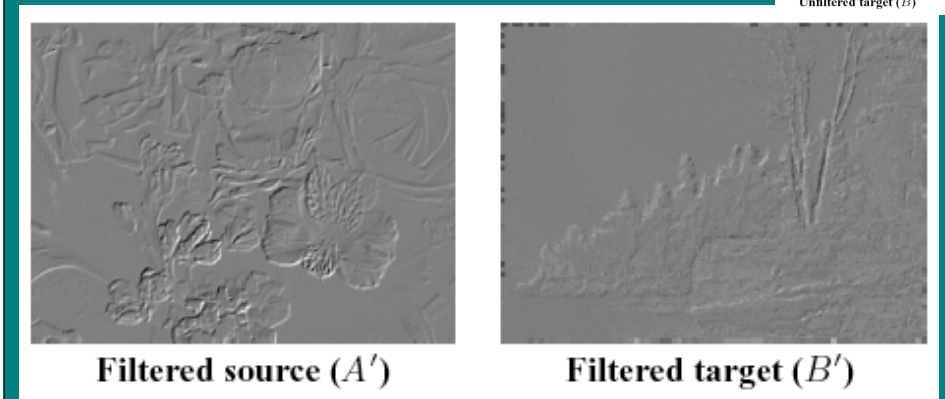


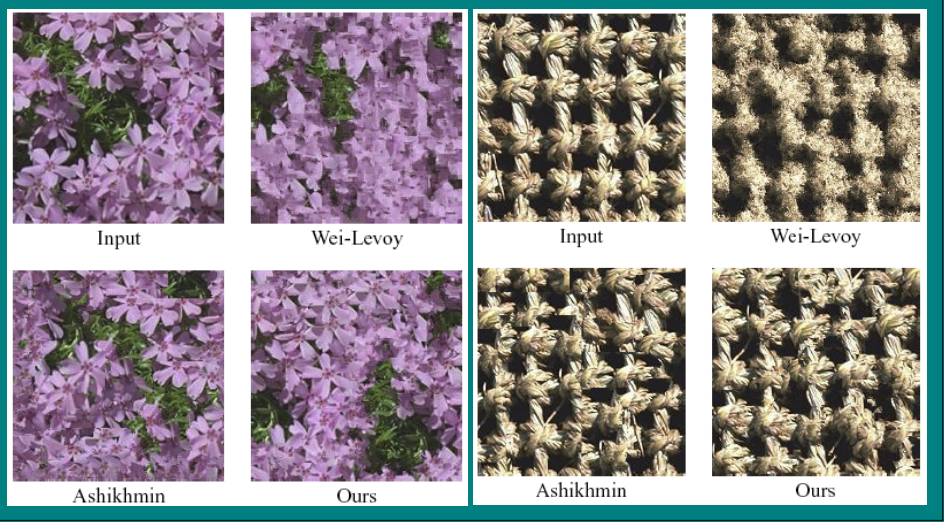
Image Filtering

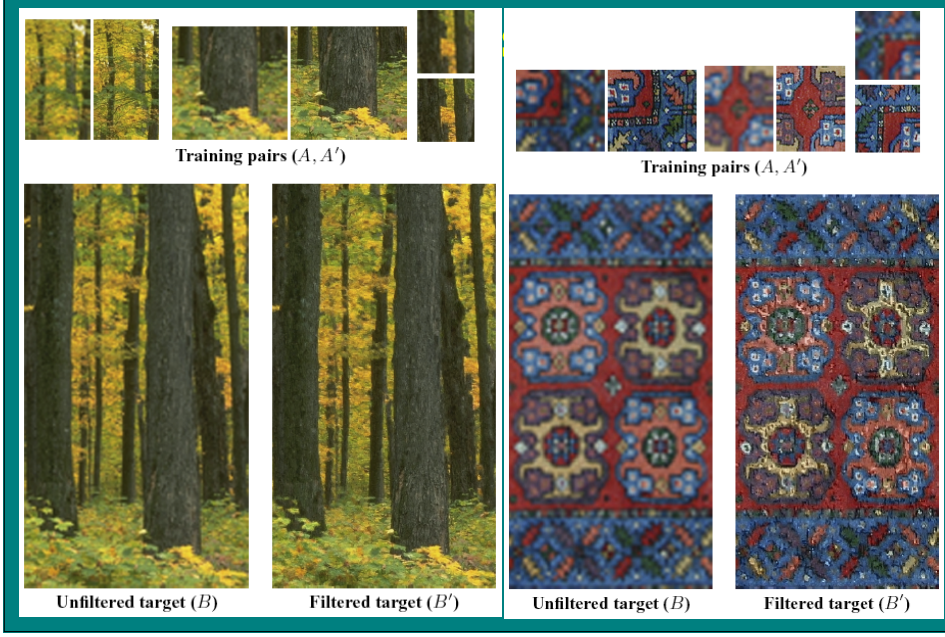


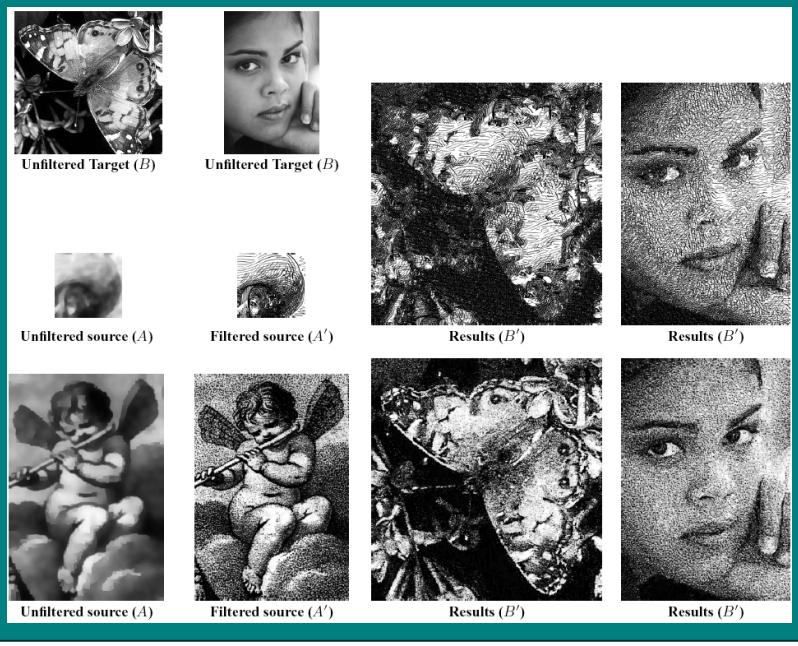
- Emboss

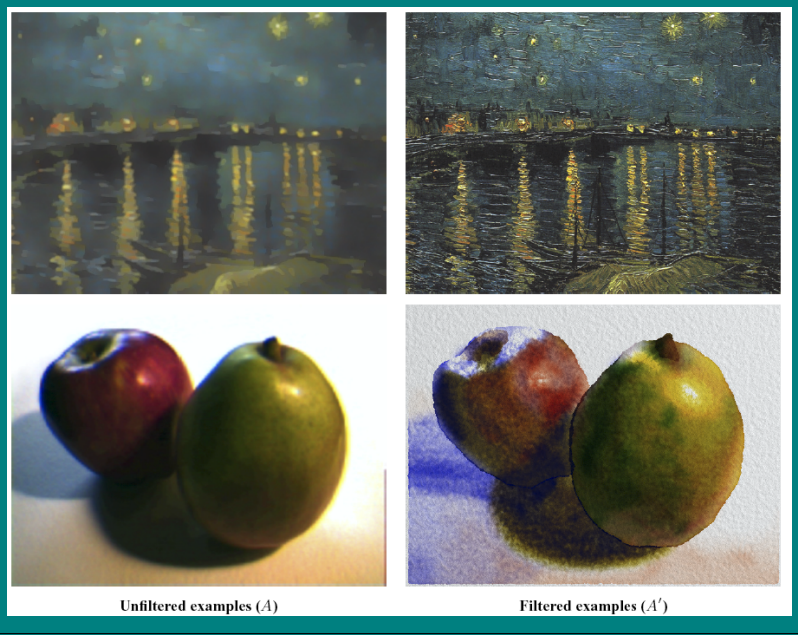


Texture Synthesis Results

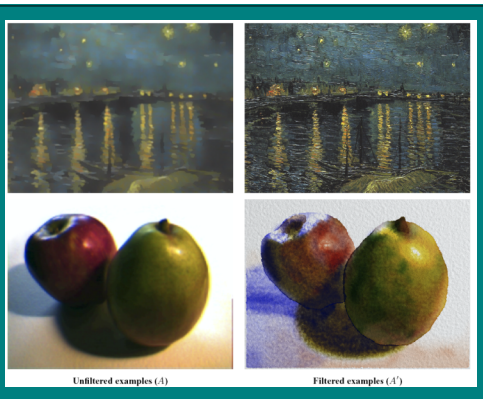


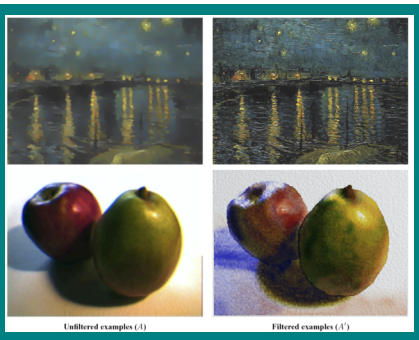


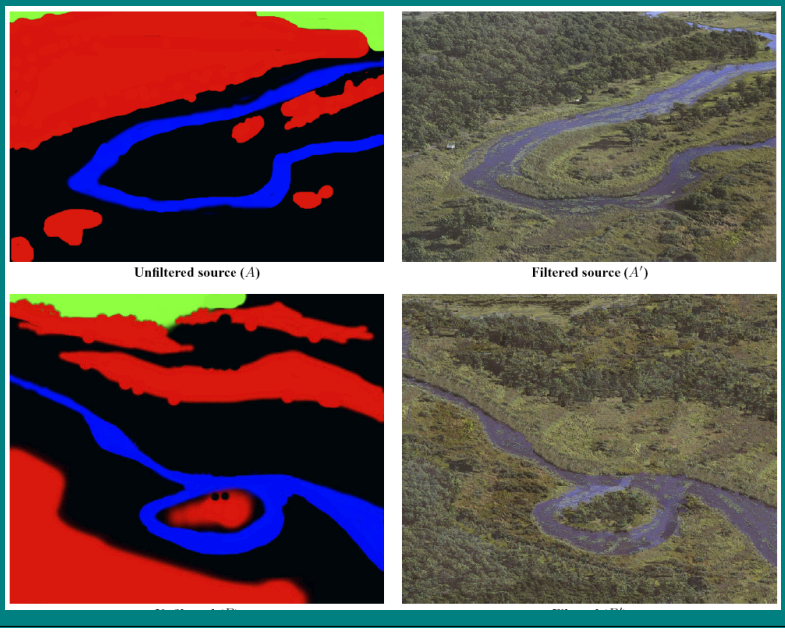

Unfiltered Target (B)
Unfiltered Target (B)
Unfiltered source (A)
Filtered source (A')
Results (B')
Results (B')
Unfiltered source (A)
Filtered source (A')
Results (B')
Results (B')

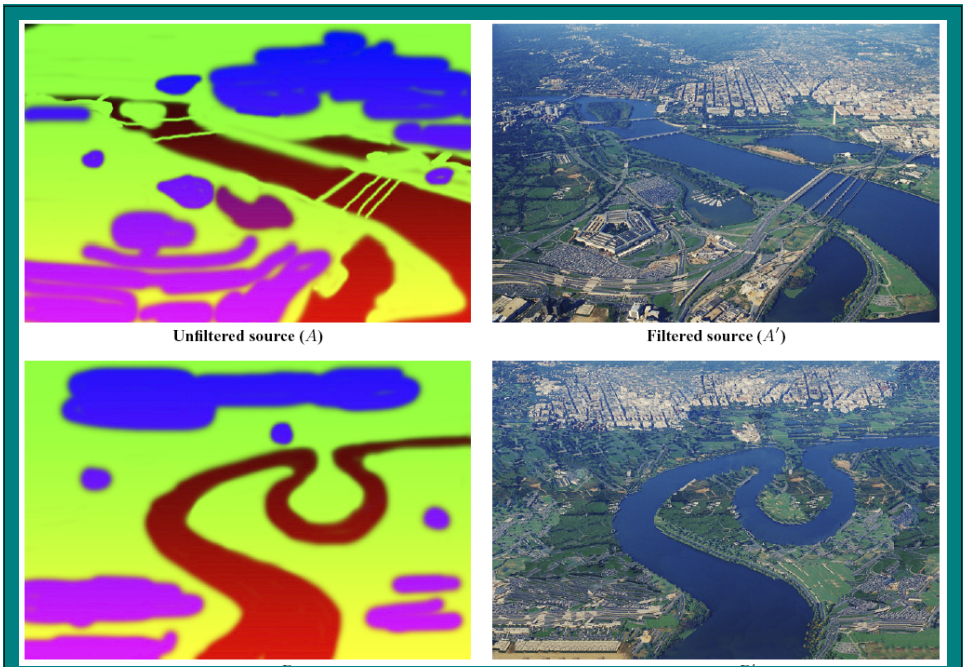

Unfiltered examples (A)
Filtered examples (A')

Painting-by-Example






Unfiltered source (A)
Filtered source (A')


Unfiltered source (A)
Filtered source (A')

Limitations

- Performance range: seconds to hours
- Only captures low-level statistical image features
- Similarity metric doesn't match perception
- Requires registered source images

Project Website

- URL
  - www.grail.cs.washington.edu/projects/image-analogies
- Contains
  - More examples
  - Software