Textures and Inpainting

Digital Visual Effects, Spring 2008

Yung-Yu Chuang

2008/6/10

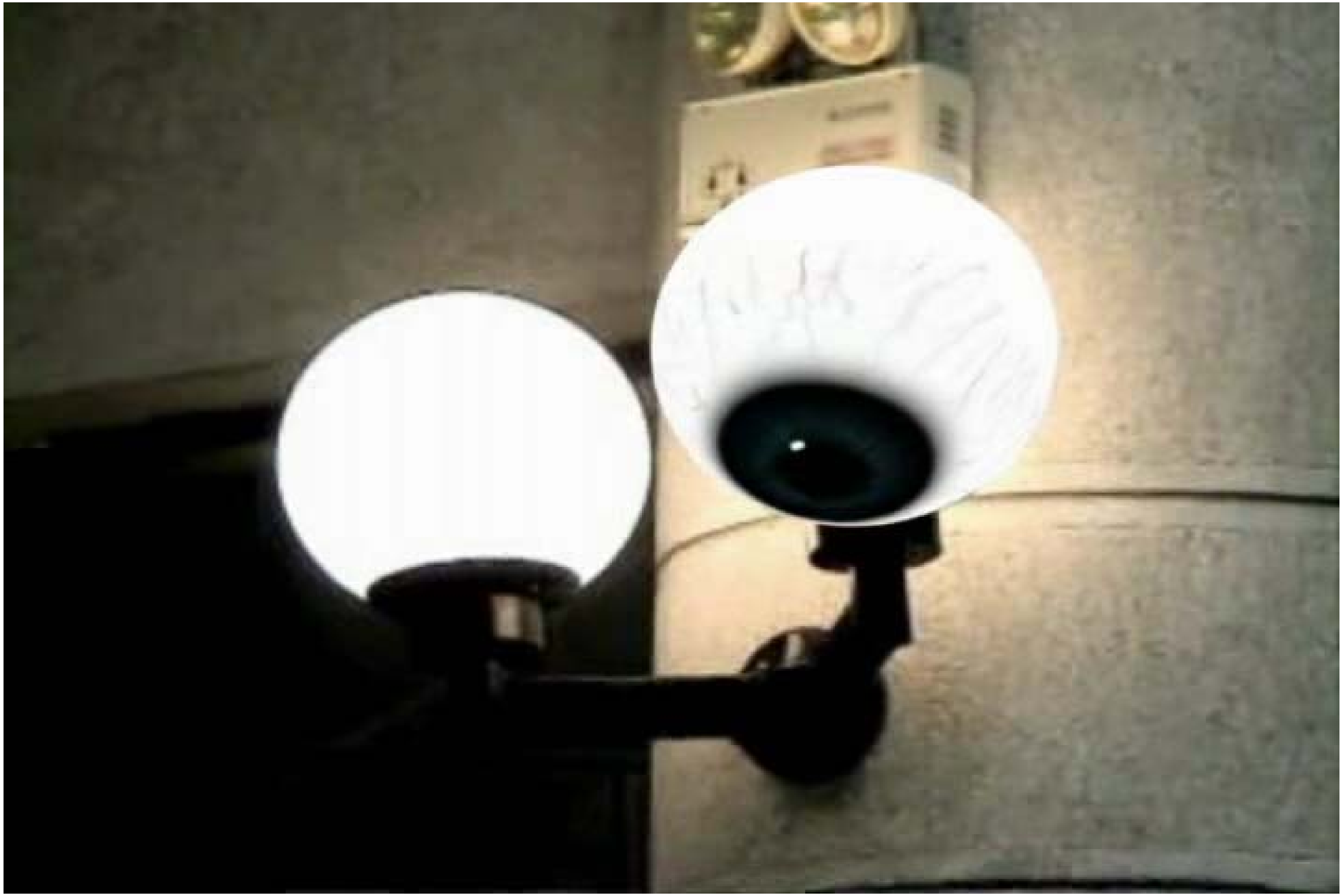
with slides by Alex Efros, Li-Yi Wei, Arno Schedl and Paul Debevec

Announcements

- Winners for project #3
- Final project:
  - demo on 6/25 (Wednesday) 1:30pm in this room
  - Report due on 6/26 (Thursday) 11:59pm

Honorable mention (13): 羅聖傑 劉俊良

Honorable mention (14): 陳鴻銘 張炳傑



Third place (18): 梁 彧 吳孟松

Third place (20): 陳宜豪 古卡茲

Second place (21): 周建男 張家翰

First place (29): 梁立衡 張秉榆

Outline

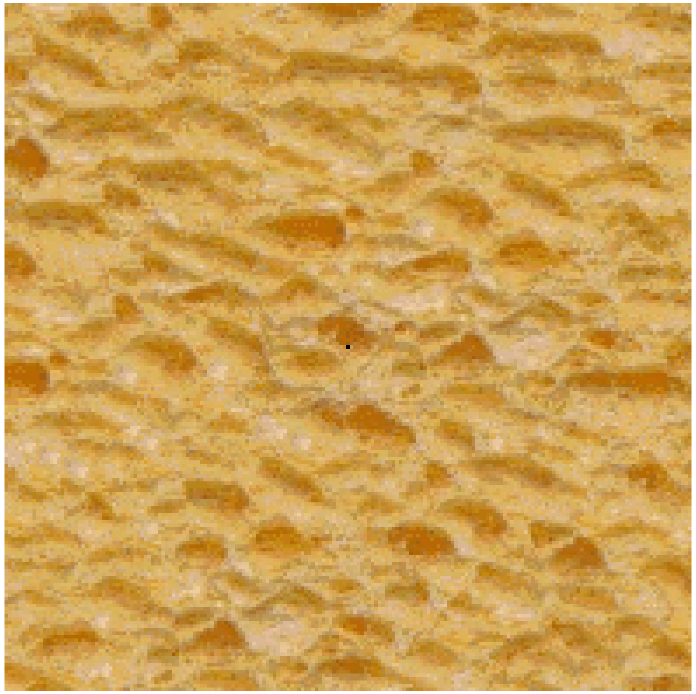
- Texture synthesis
- Acceleration by multi-resolution and TSVQ
- Patch-based texture synthesis
- Image analogies

Texture synthesis

Texture synthesis

input image

synthesis

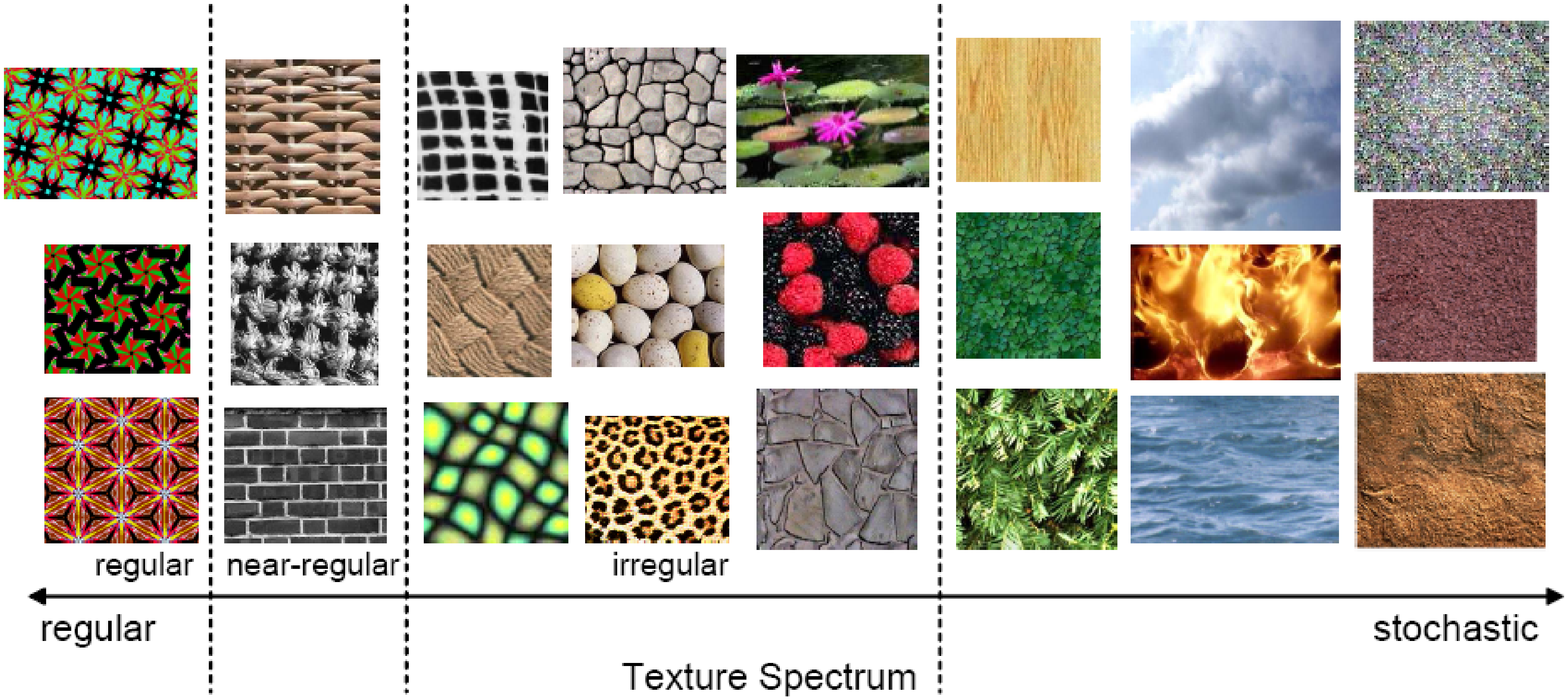


generated image

- Given a finite sample of some texture, the goal is to synthesize other samples from that same texture.
  - The sample needs to be "large enough"

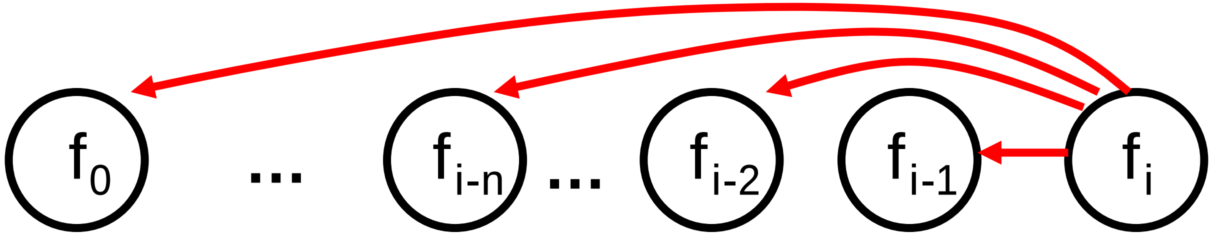
The challenge

- How to capture the essence of texture?
- Need to model the whole spectrum: from repeated to stochastic texture

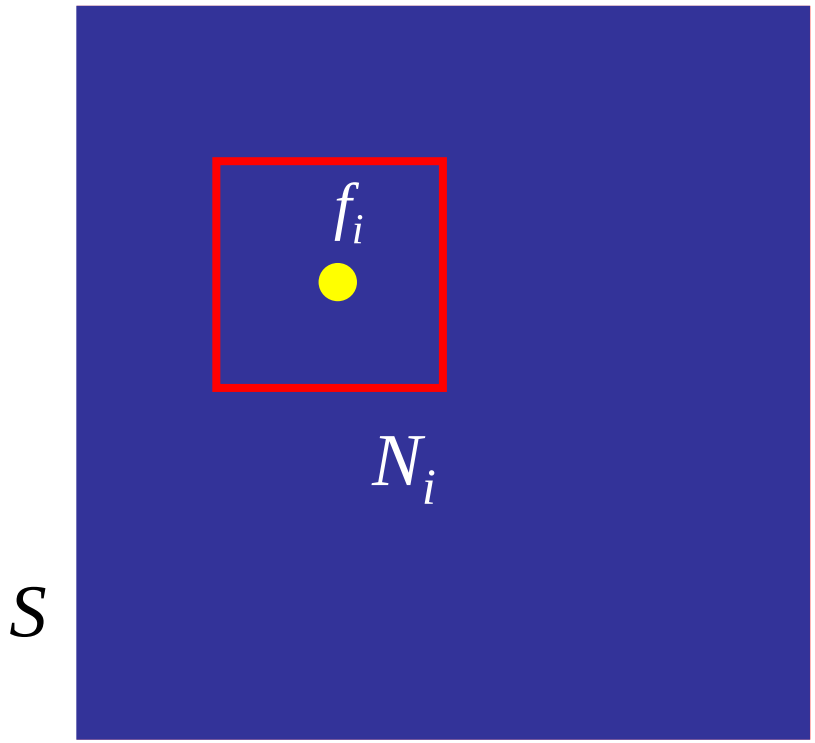


Markov property

- $P(f_i | f_{i-1}, f_{i-2}, f_{i-3}, \ldots f_0) = P(f_i | f_{i-1}, f_{i-2}, \ldots f_{i-n})$



- $P(f_i | f_{S-\{i\}}) = P(f_i | f_{N_i})$



Motivation from language

- [Shannon'48] proposed a way to generate English-looking text using N-grams:
  - Assume a generalized Markov model
  - Use a large text to compute probability distributions of each letter given N-1 previous letters
    - precompute or sample randomly
  - Starting from a seed repeatedly sample this Markov chain to generate new letters
  - One can use whole words instead of letters too.

Mark V. Shaney (Bell Labs)

- Results (using alt.singles corpus):
  - *"One morning I shot an elephant in my arms and kissed him."*
  - *"I spent an interesting evening recently with a grain of salt"*
- Notice how well local structure is preserved!
  - Now let's try this for video and in 2D…

Video textures

- SIGGRAPH 2000 paper by Arno Schedl, Riachard Szeliski, David Salesin and Irfan Essa.

Still photos

Video clips

Video textures

Problem statement

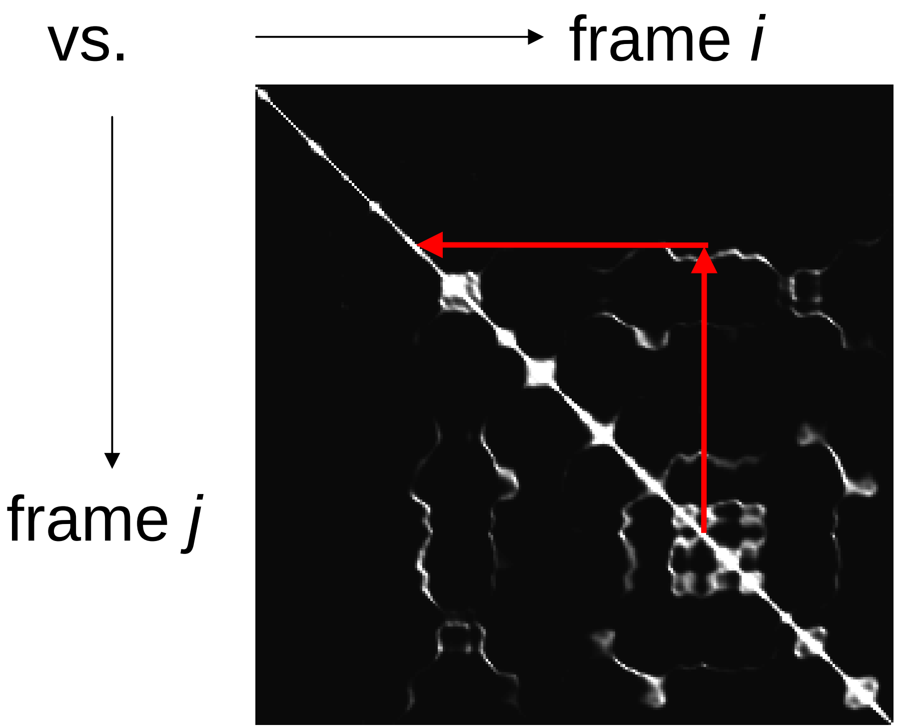
video clip　　　　video texture

Approach

How do we find good transitions?

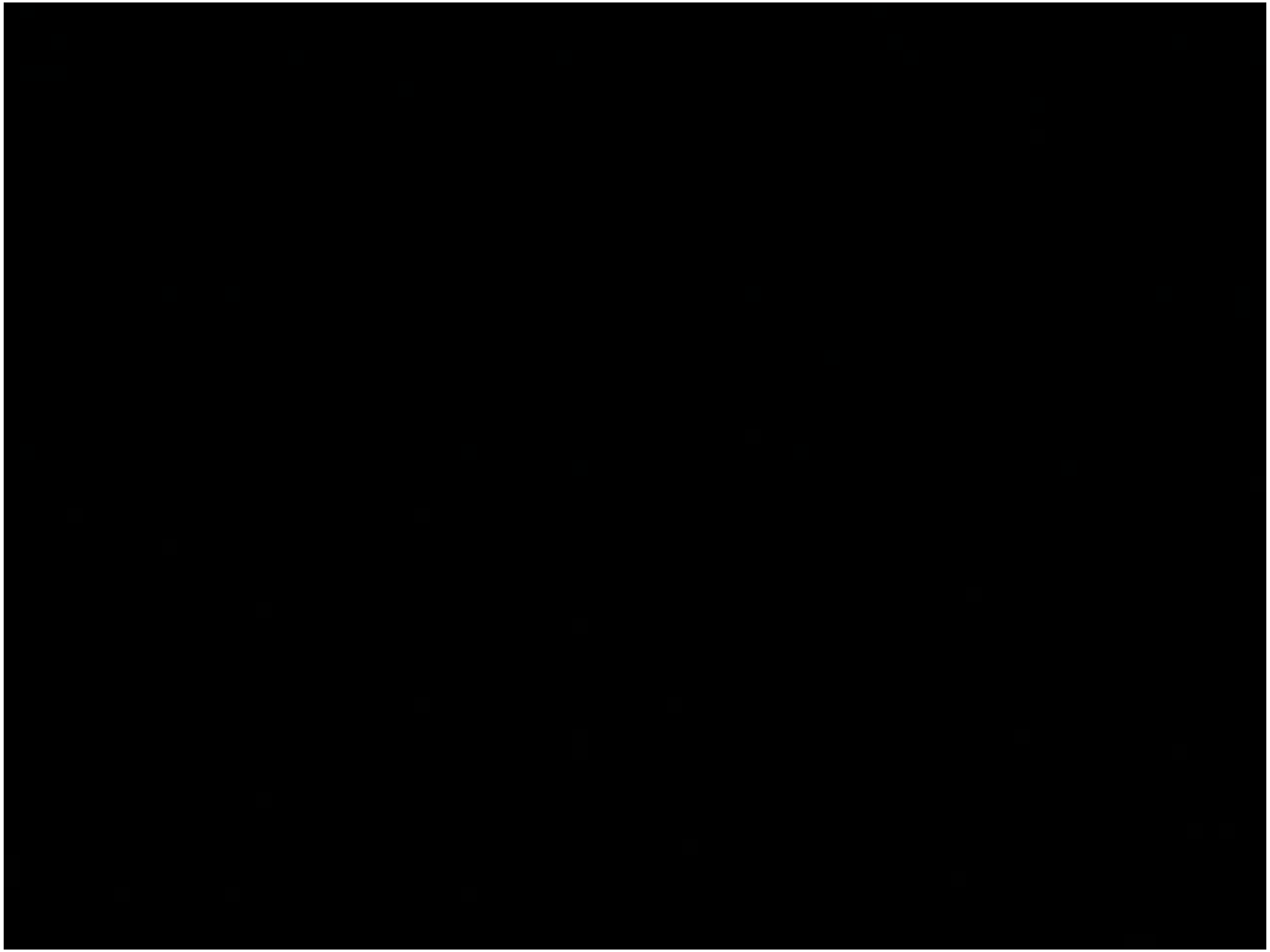
Finding good transitions

Compute L_2 distance $D_{i,\,j}$ between all frames

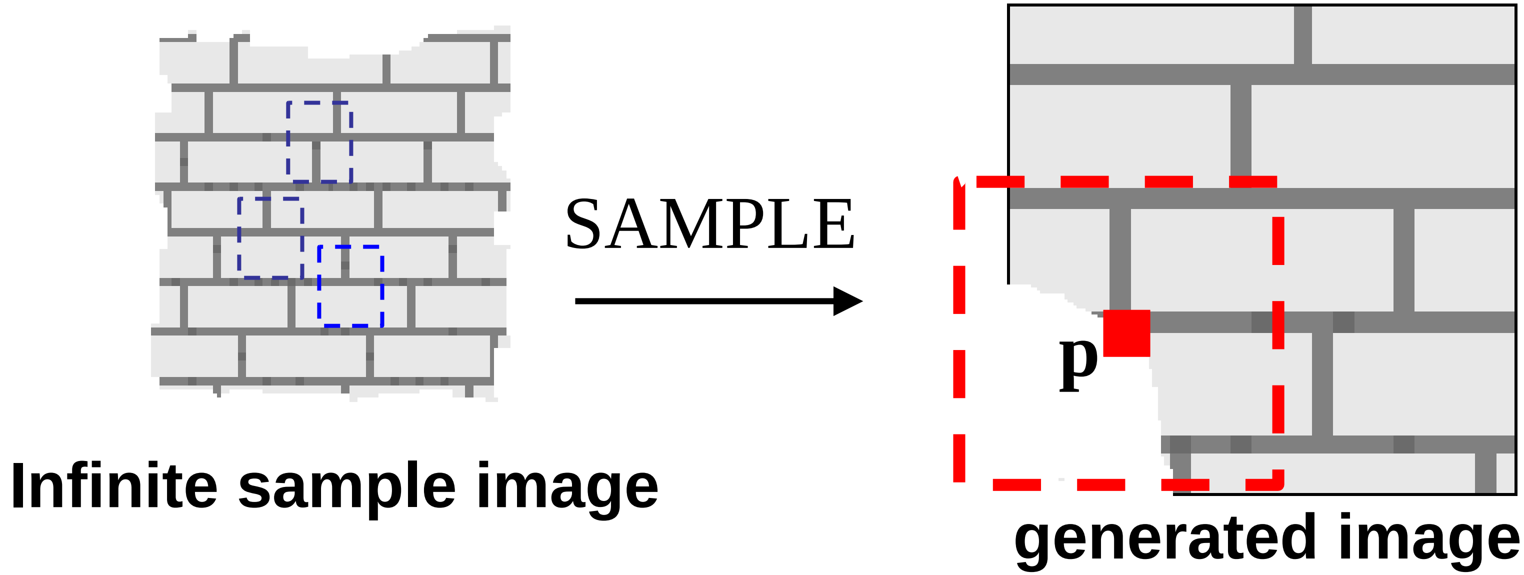


Similar frames make good transitions

Video textures

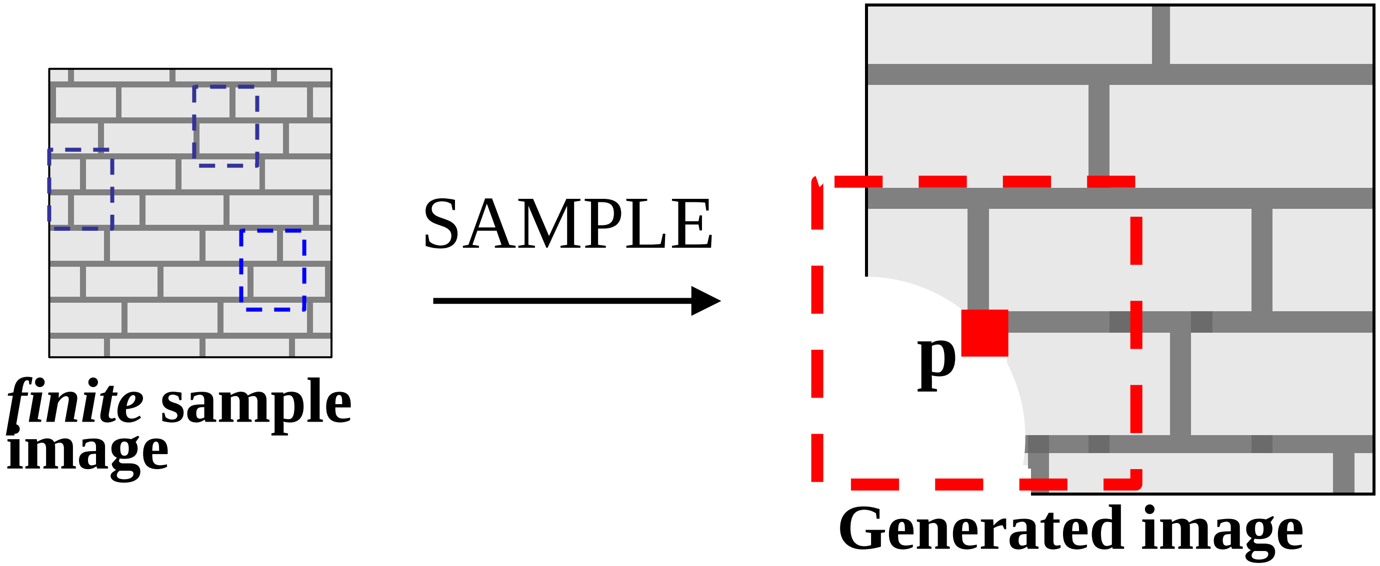


Ideally



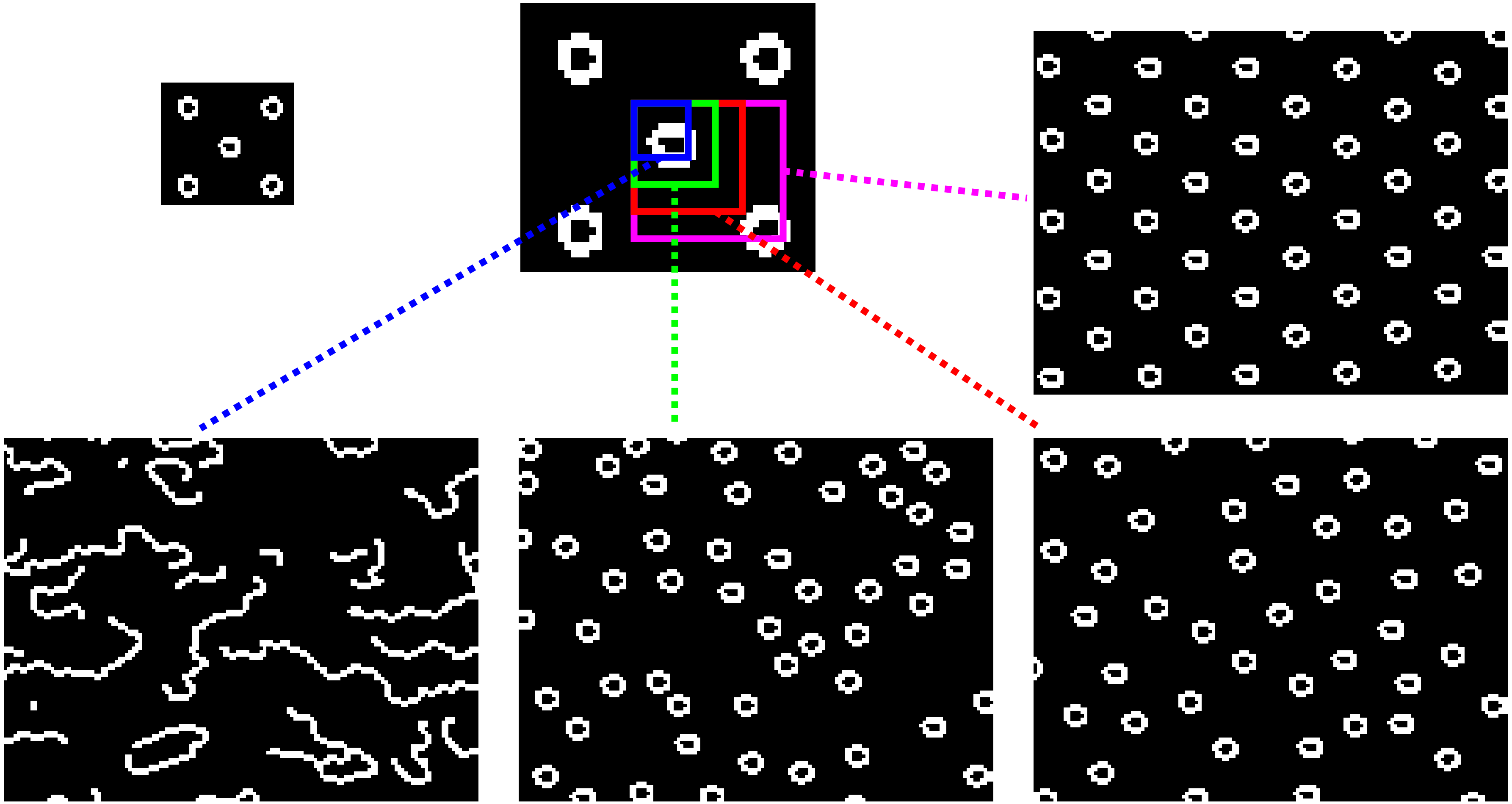
- Assuming Markov property, what is conditional probability distribution of p, given the neighbourhood window?
- Instead of constructing a model, let's directly search the input image for all such neighbourhoods to produce a histogram for p
- To synthesize p, just pick one match at random

In reality

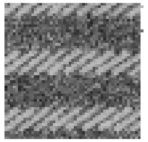


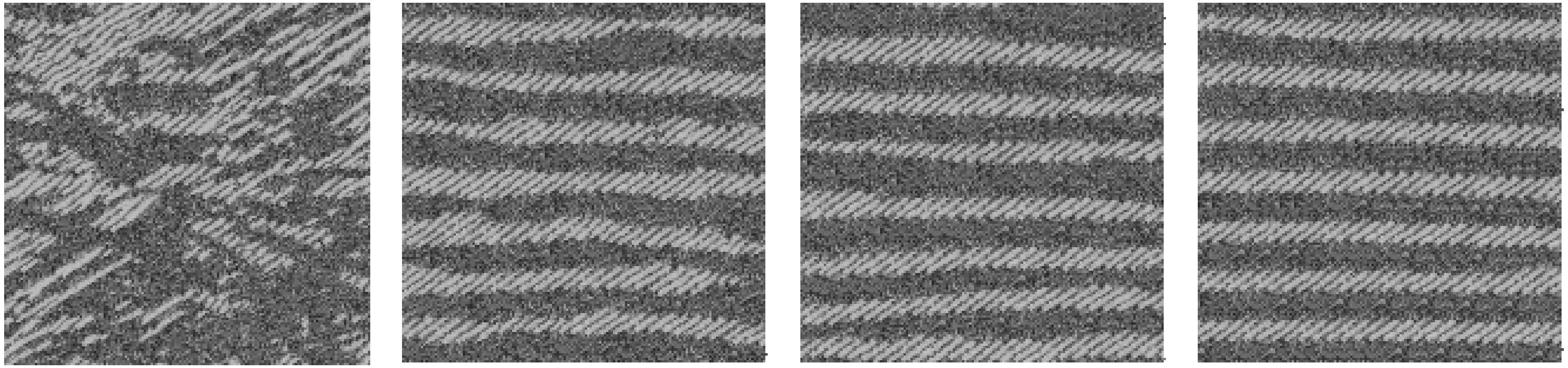
- However, since our sample image is finite, an exact neighbourhood match might not be present
- So we find the best match using SSD error (weighted by a Gaussian to emphasize local structure), and take all samples within some distance from that match
- Using *Gaussian-weighted* SSD is very important

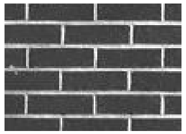
Neighborhood size matters

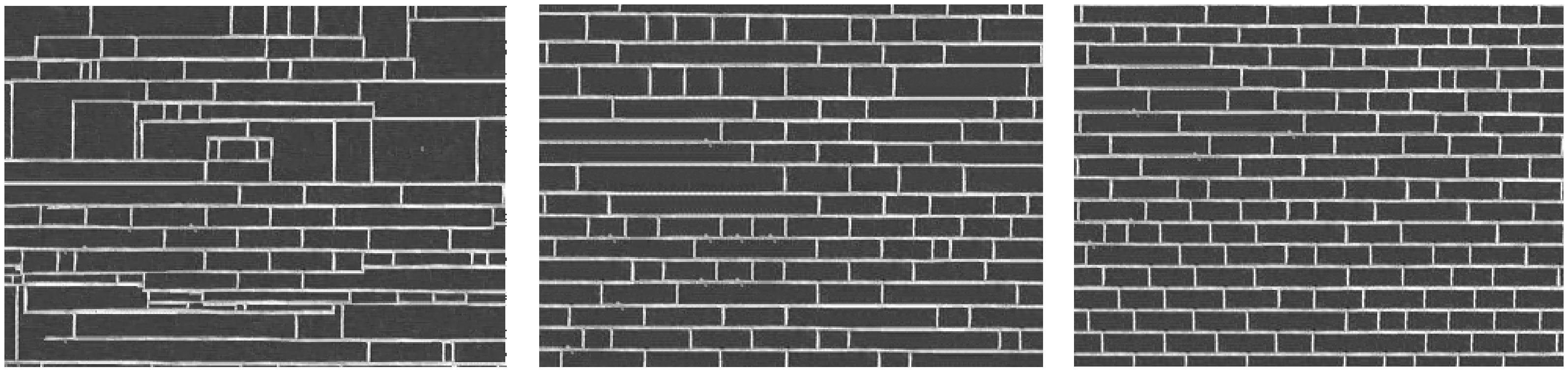


More results





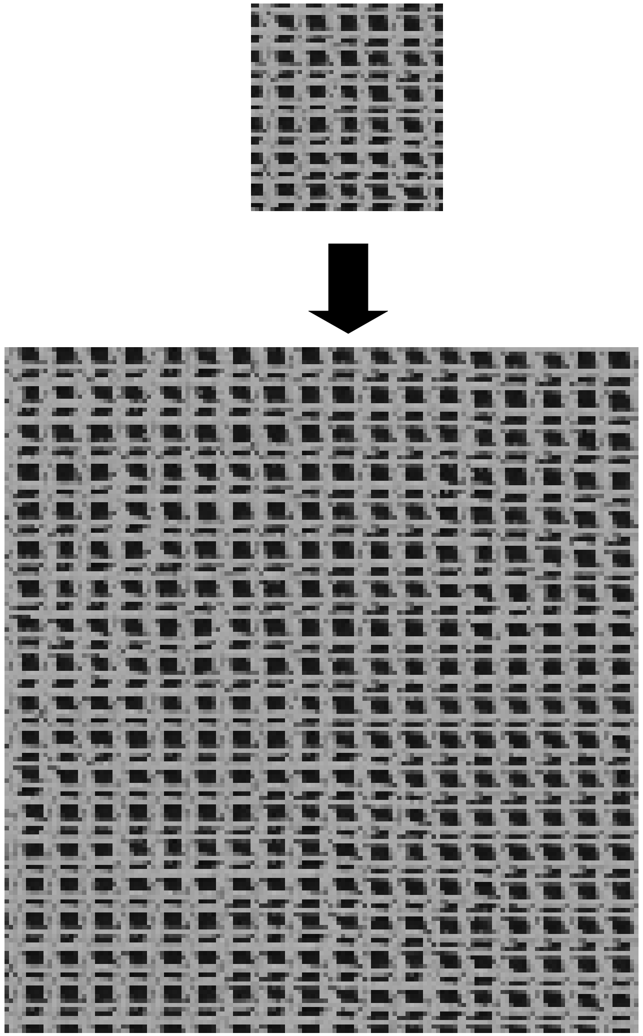




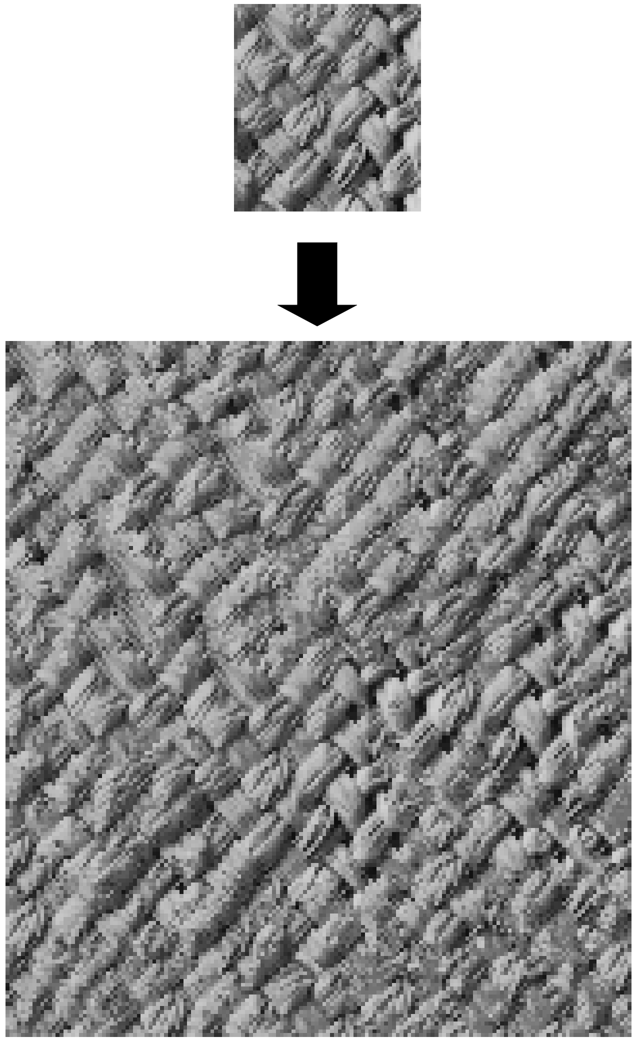
Increasing window size

More results

french canvas



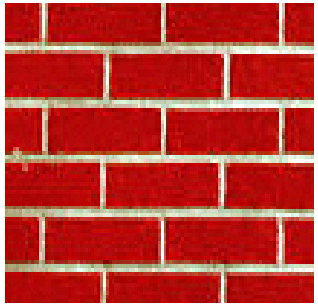
rafia weave

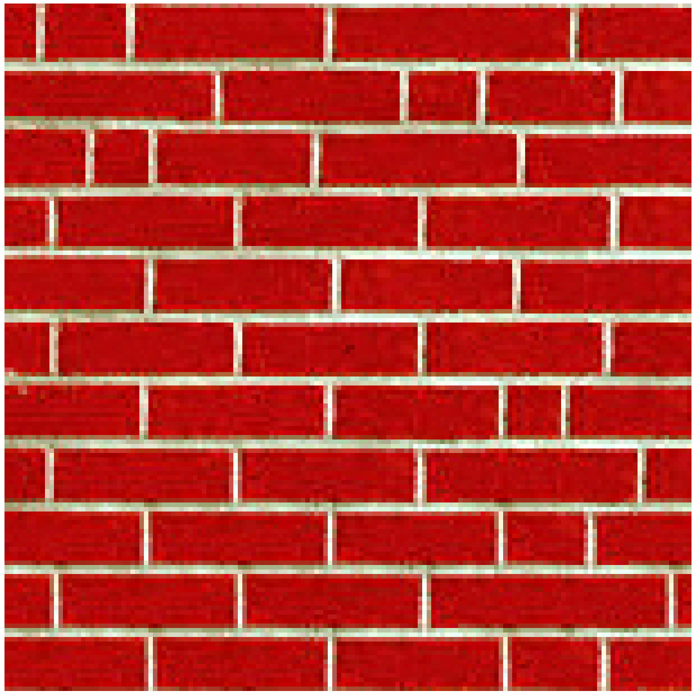


More results

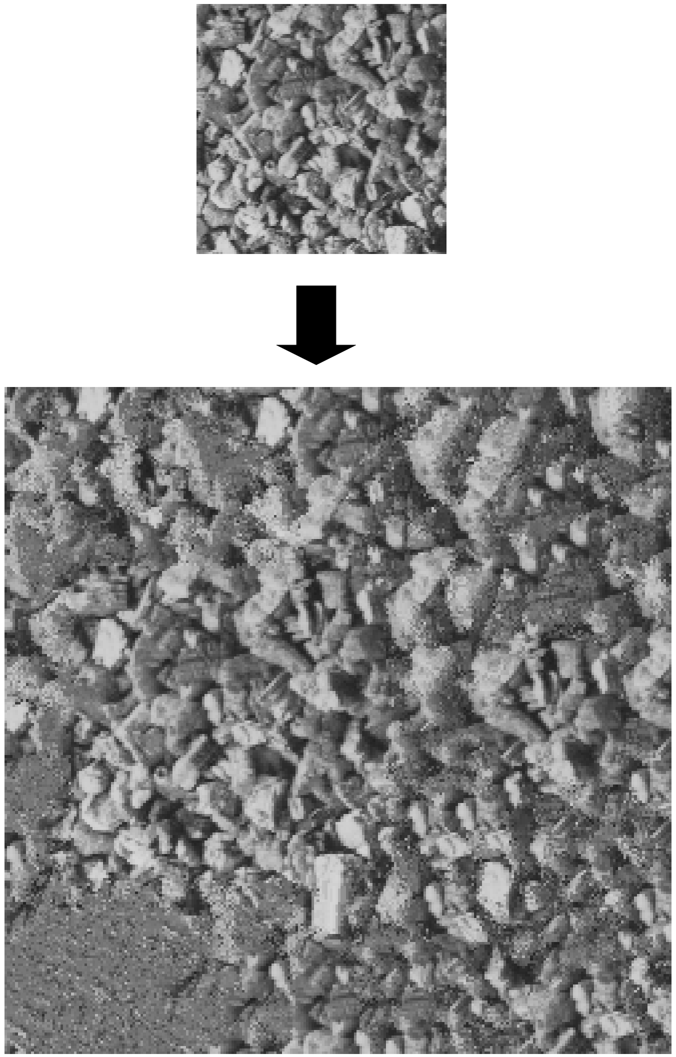
wood

brick wall

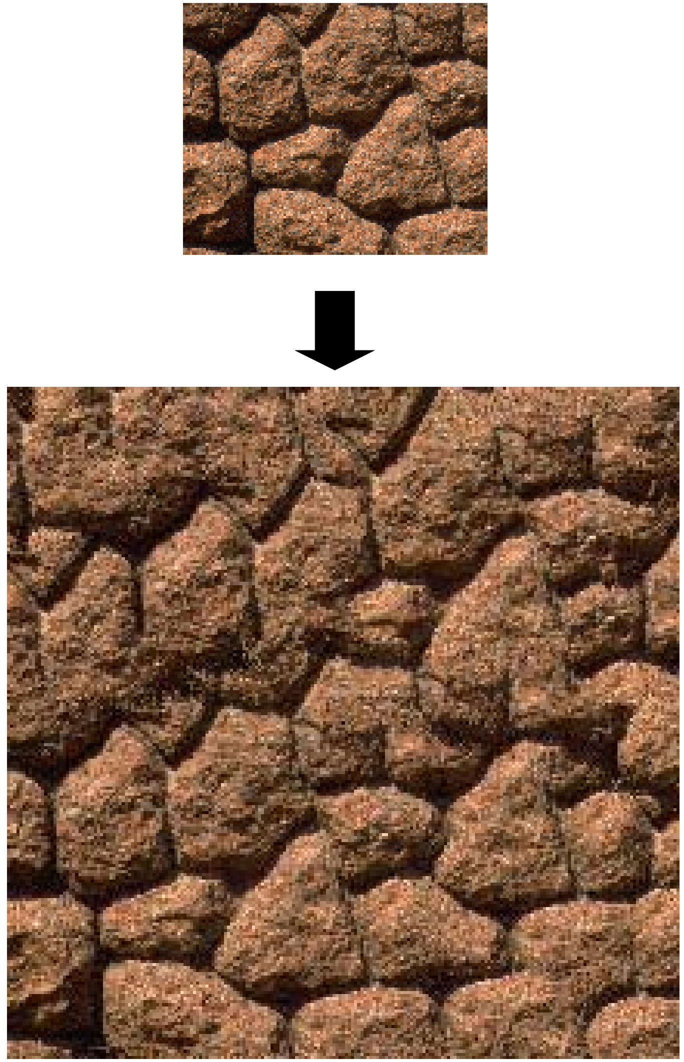




Failure cases



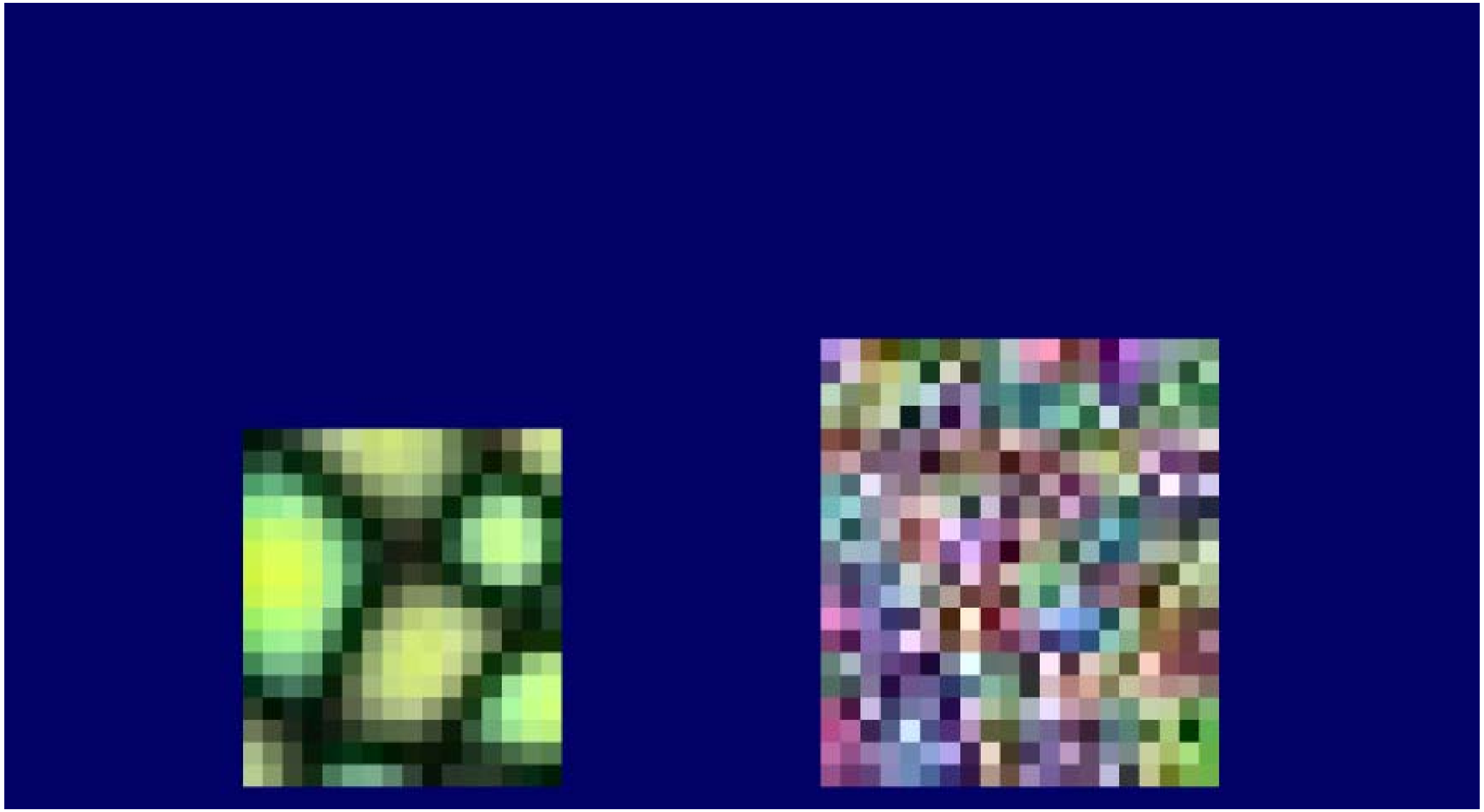
Growing garbage



Verbatim copying

Summary of the basic algorithm

- Exhaustively search neighborhoods



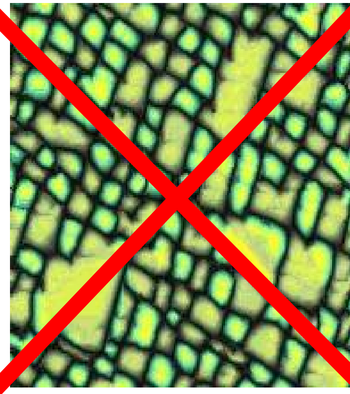
Neighborhood

- Neighborhood size determines the quality & cost

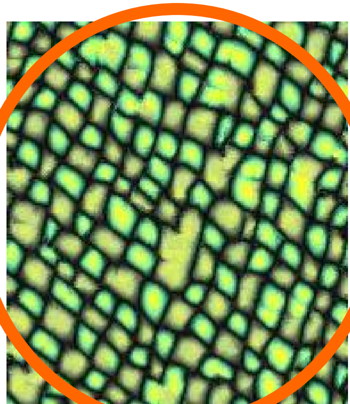
3×3
423 s

5×5
528 s

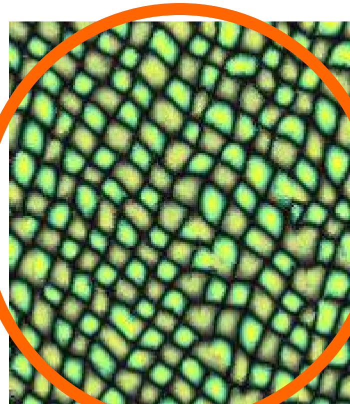
7×7
739 s



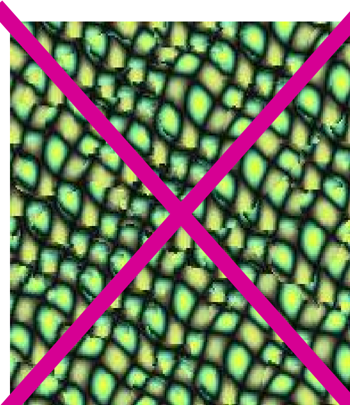
9×9
1020 s



11×11
1445 s



41×41
24350 s



Summary

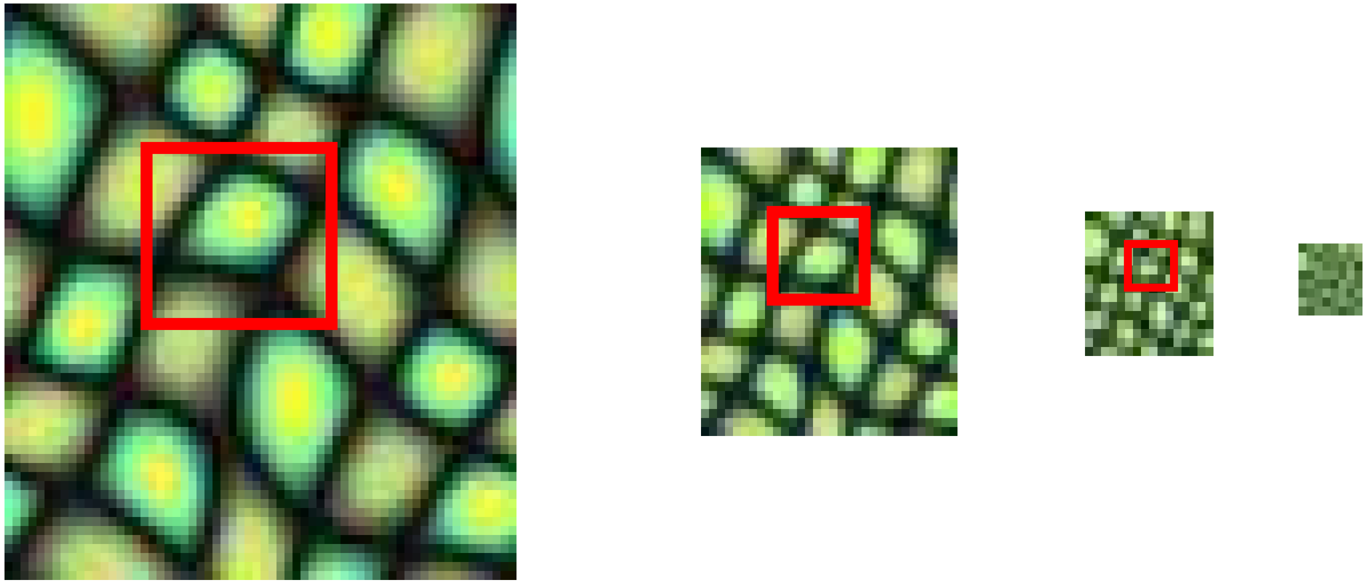
- Advantages:
  - conceptually simple
  - models a wide range of real-world textures
  - naturally does hole-filling
- Disadvantages:
  - it's slow
  - it's a heuristic

Acceleration by Wei & Levoy

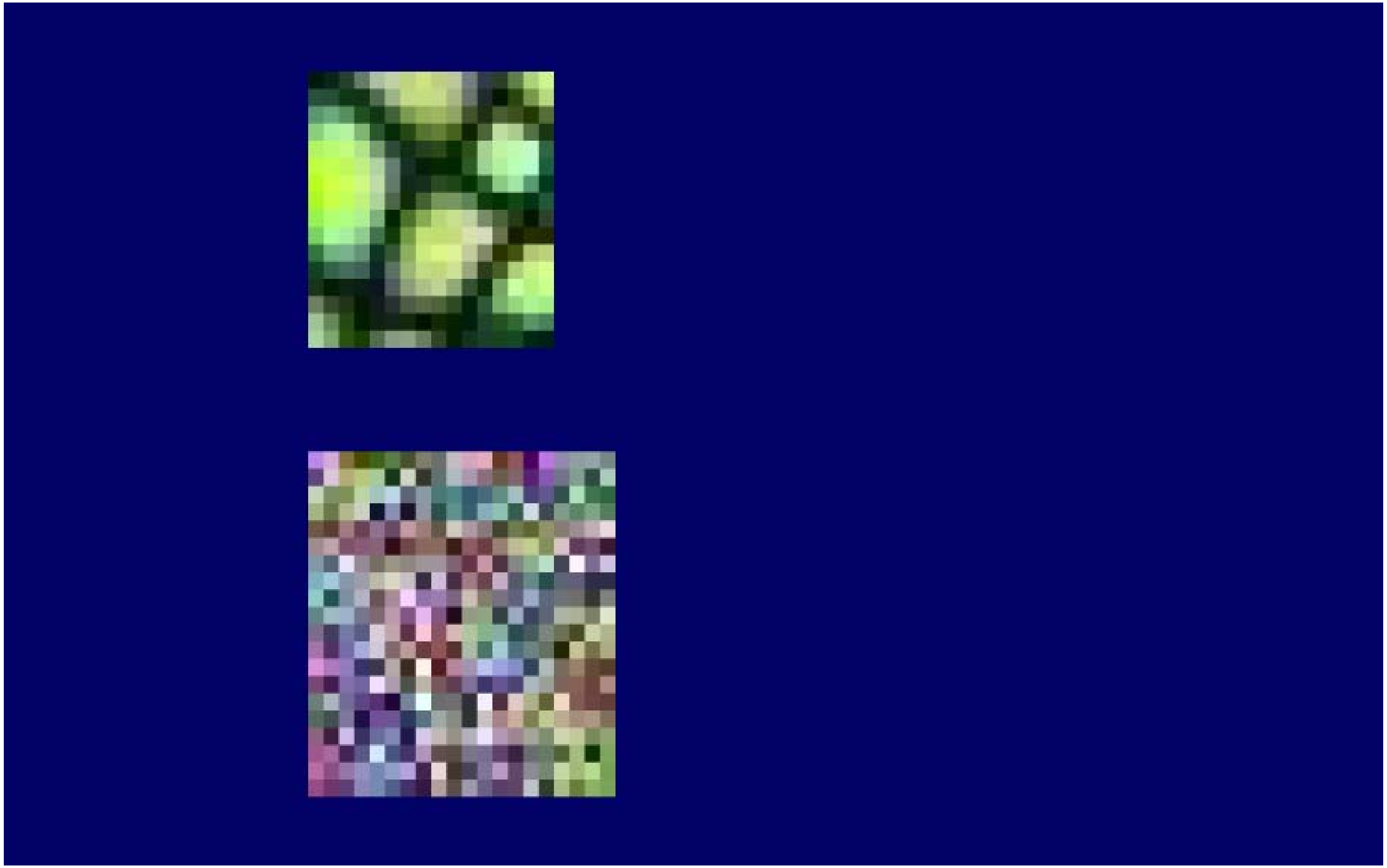
- Multi-resolution
- Tree-structure

Multi-resolution pyramid

High resolution | Low resolution

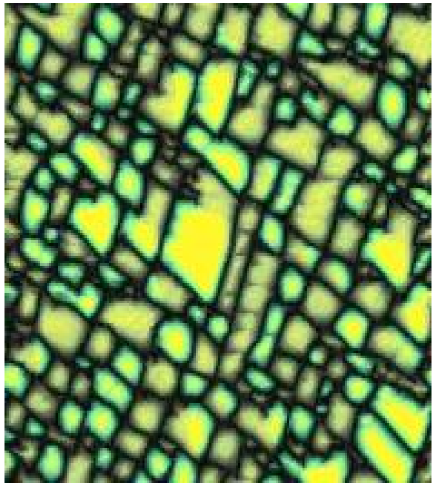


Multi-resolution algorithm

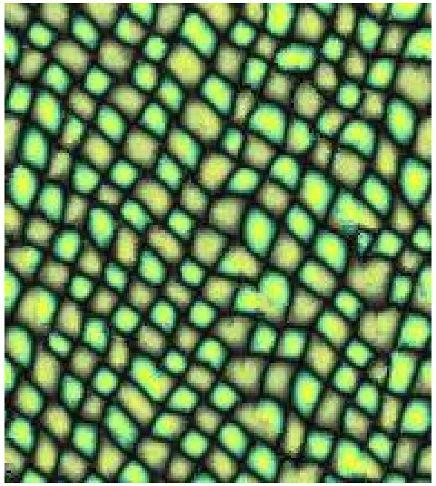


Benefits

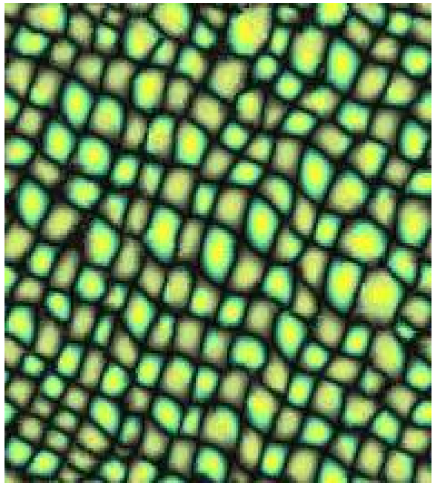
- Better image quality & faster computation (by using smaller windows)



1 level
5×5

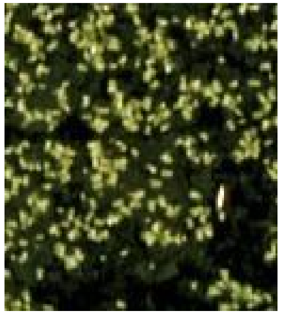


1 level
11×11

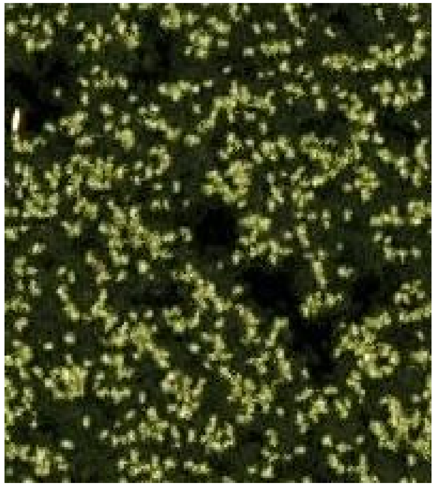


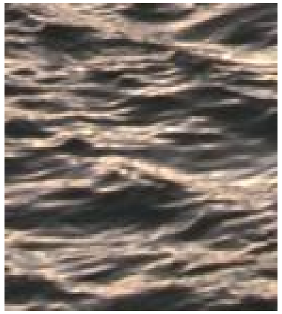
3 levels
5×5

Results

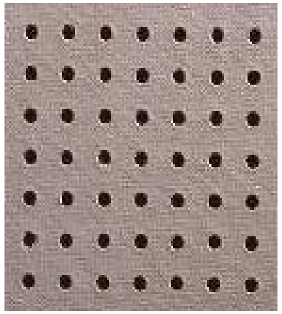


Random

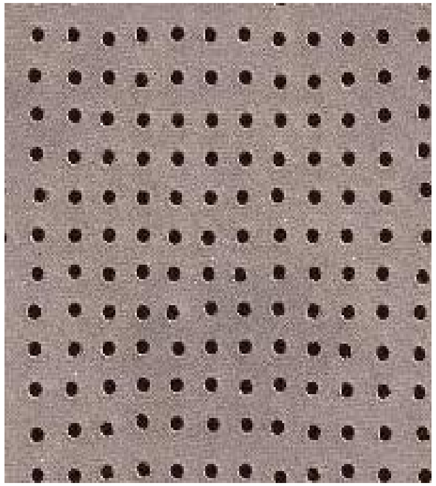




Oriented



Regular

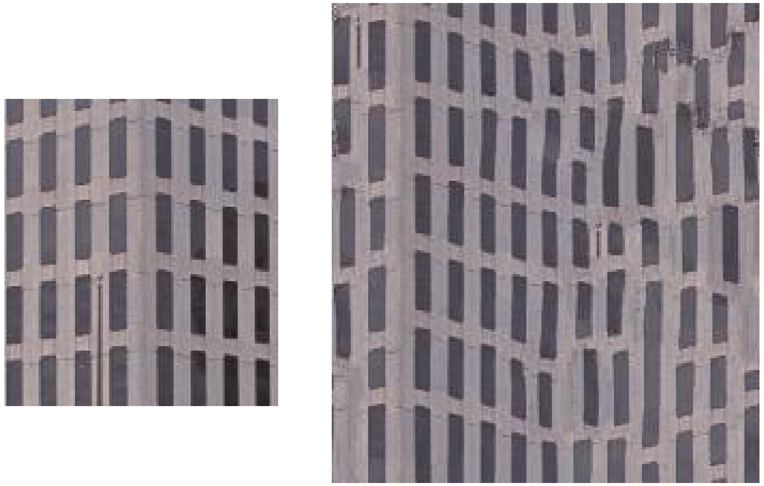


Semi-regular

Failures

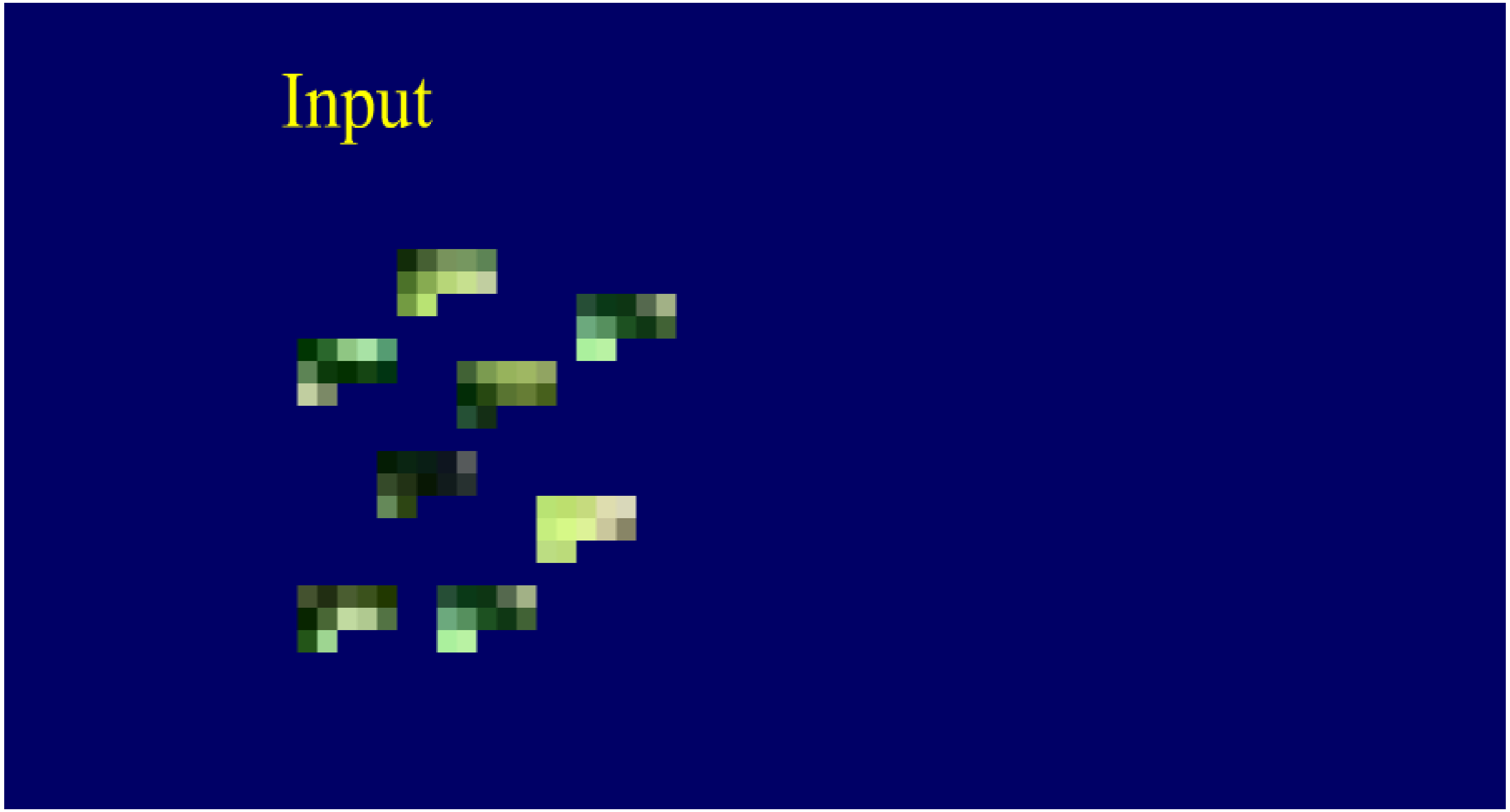
- Non-planar structures

- Global information



Acceleration

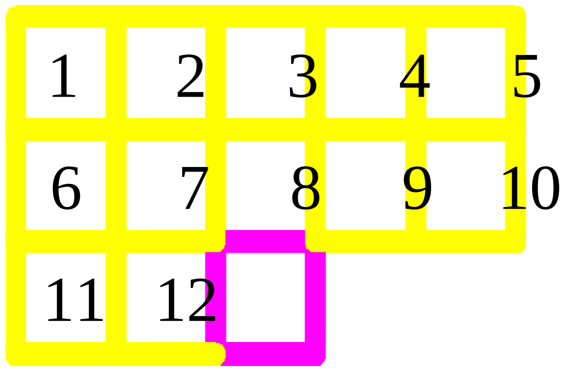
- Computation bottleneck: neighborhood search



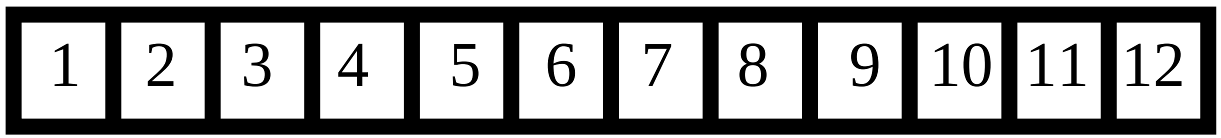
Nearest point search

- Treat neighborhoods as high dimensional points

Neighborhood



High dimensional point/vector

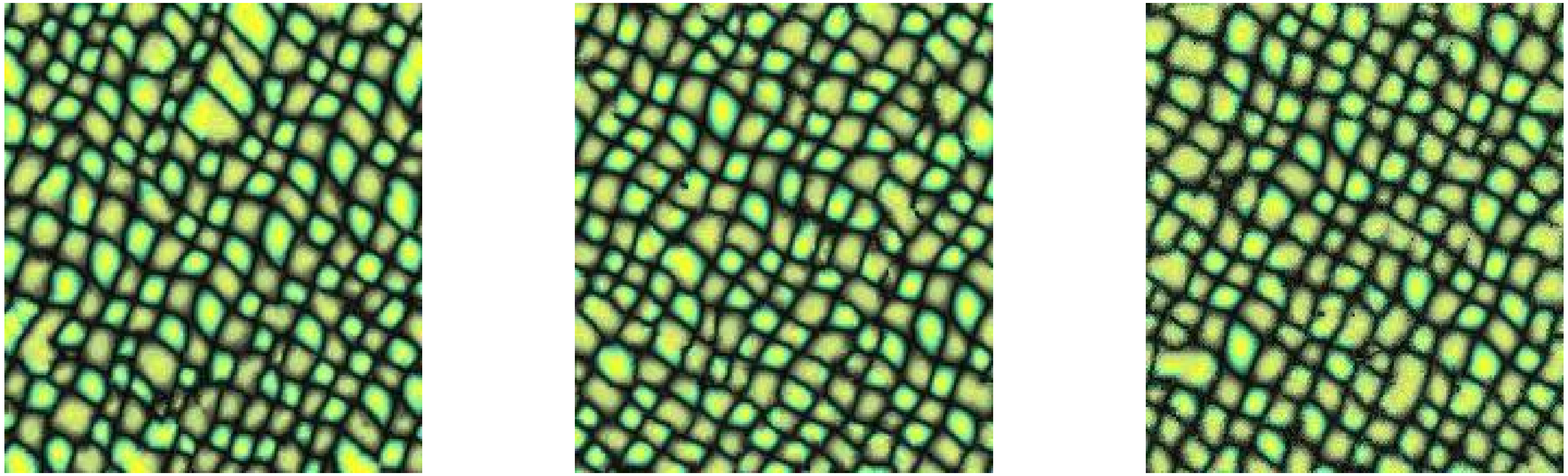


Tree-Structured Vector Quantization

Timing

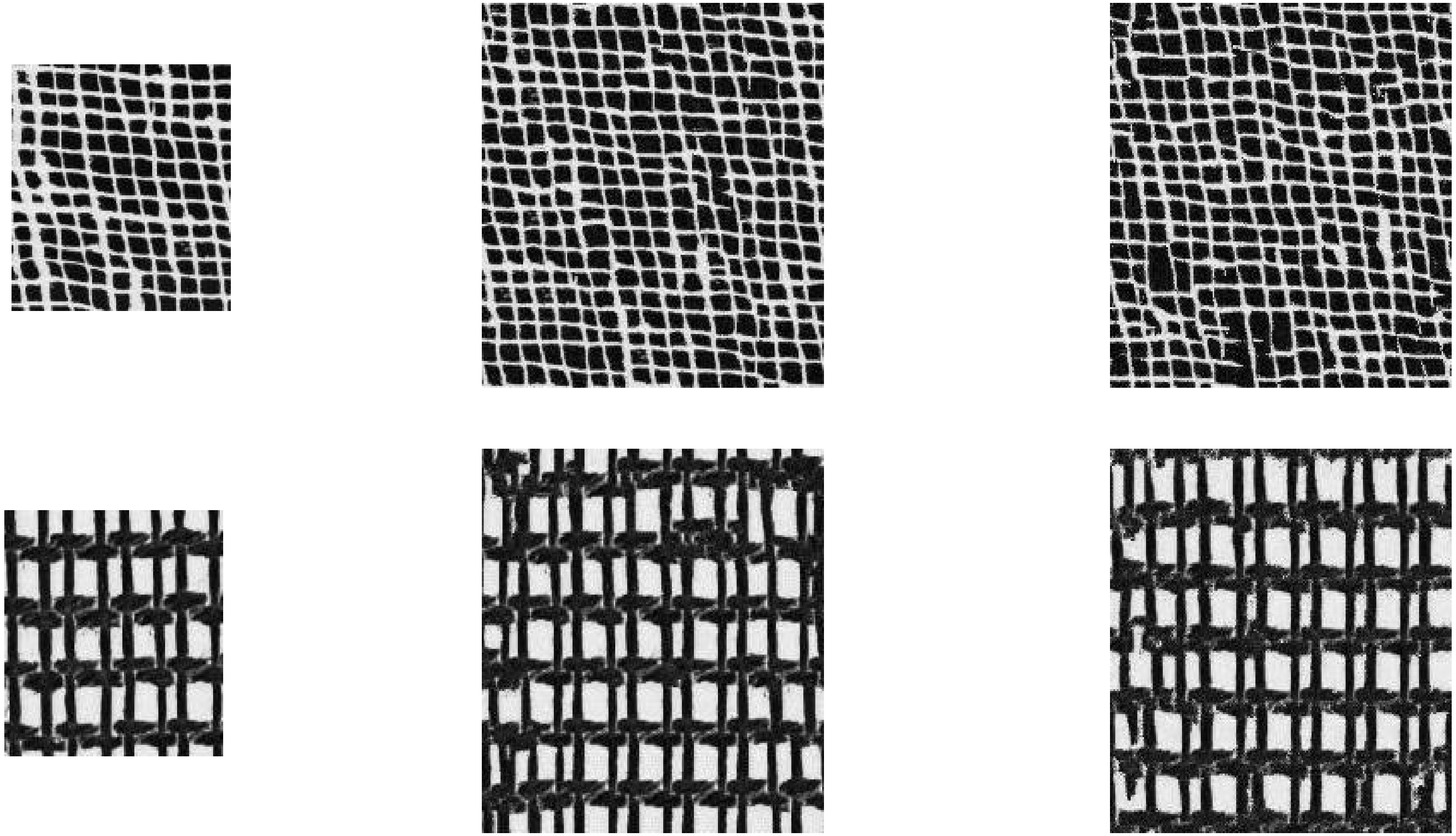
- Time complexity : O(log N) instead of O(N)

Efros 99 | Full searching | TSVQ



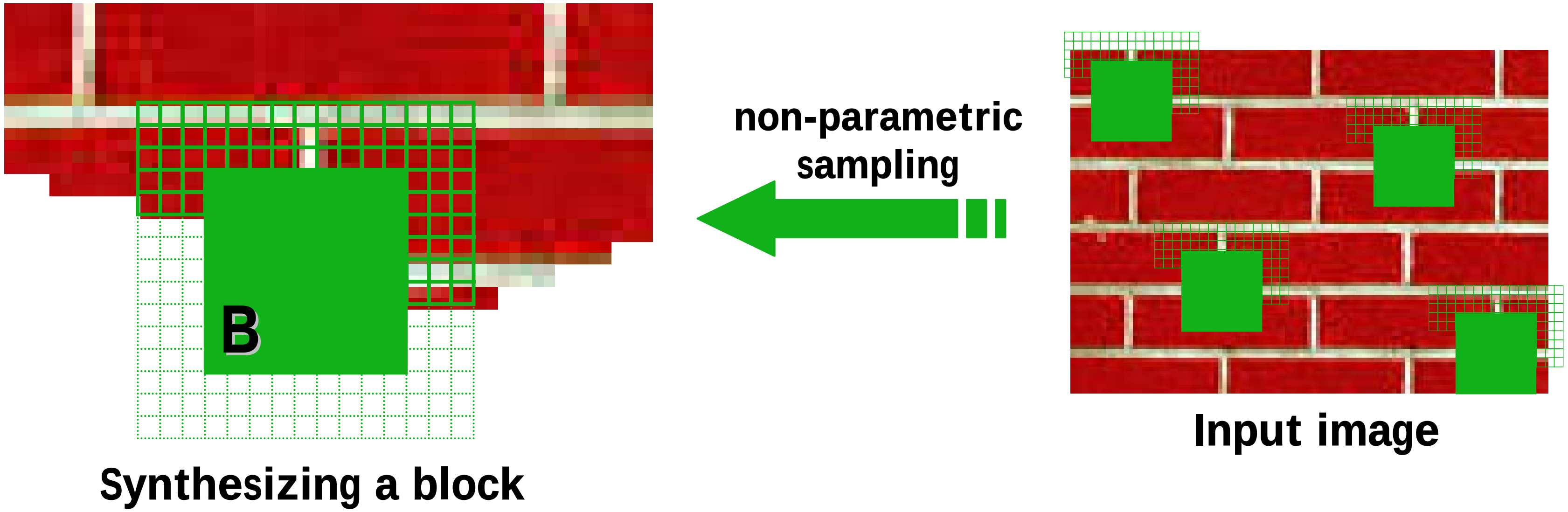
1941 secs | 503 secs | 12 secs

Results



Input　Exhaustive: 360 s　TSVQ: 7.5 s

Patch-based methods



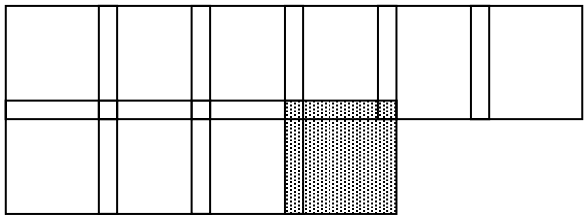
- Observation: neighbor pixels are highly correlated

Idea: unit of synthesis = block

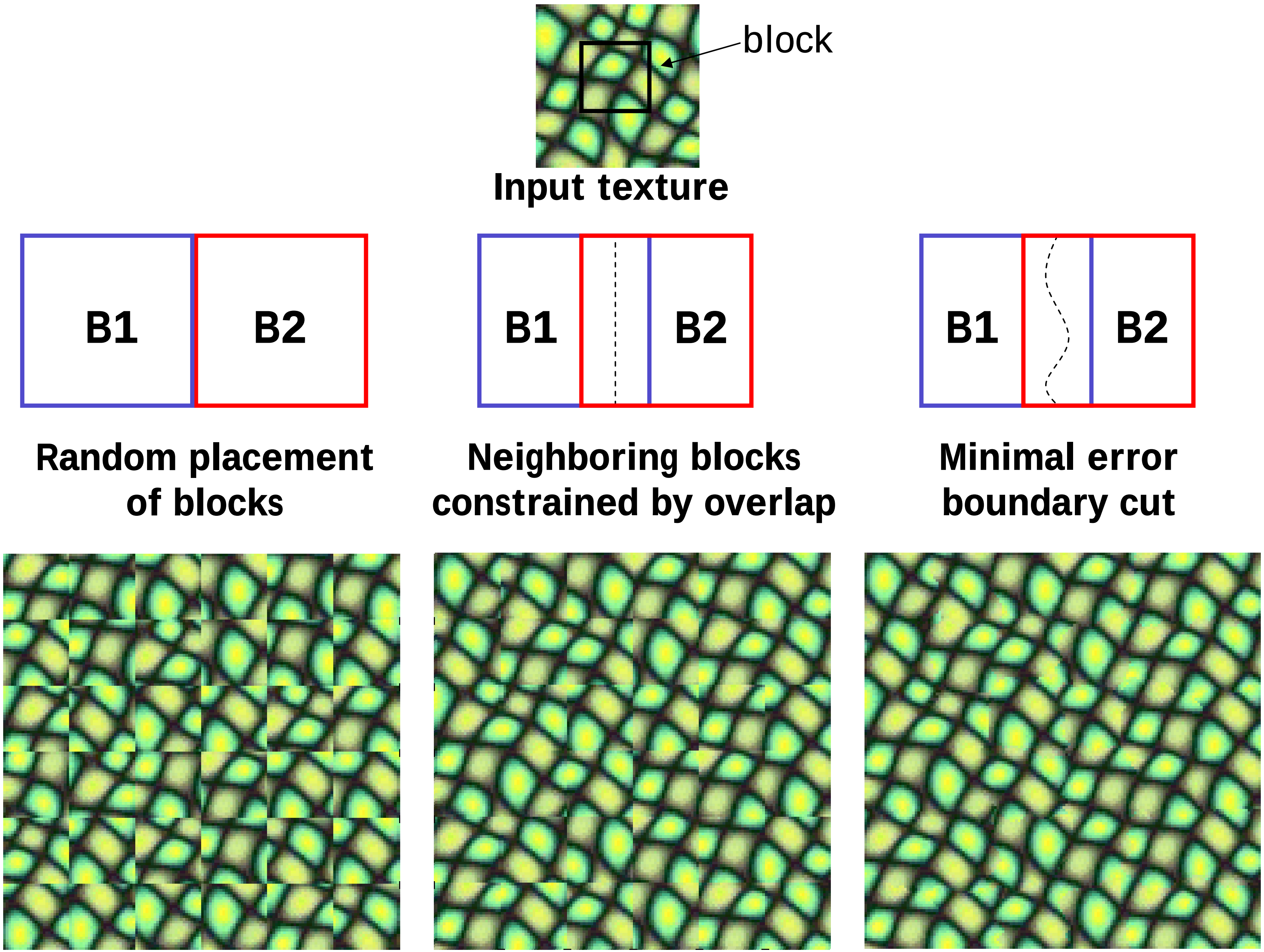
- Exactly the same but now we want P(**B**|N(**B**))
- Much faster: synthesize all pixels in a block at once

Algorithm

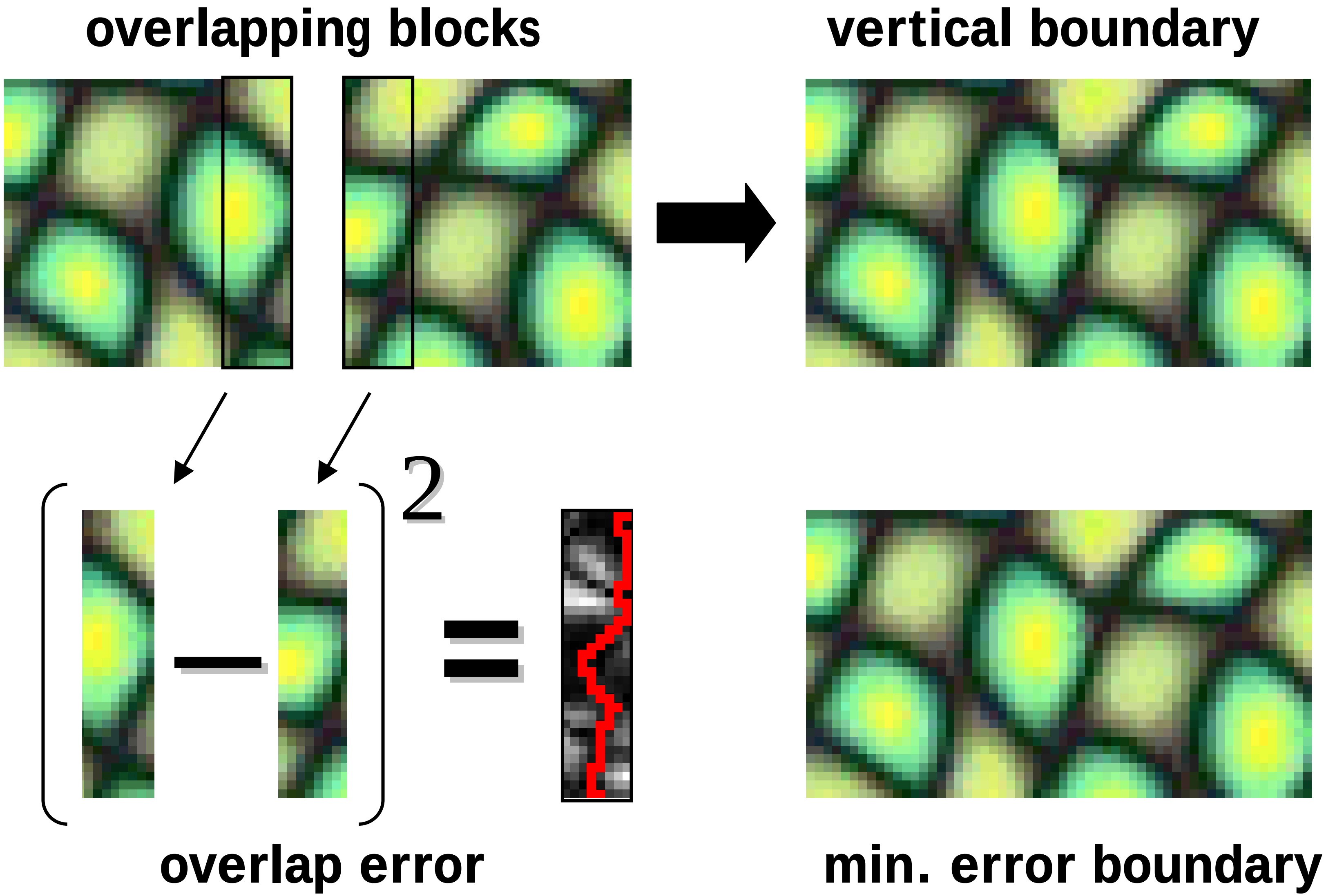
- Pick size of block and size of overlap
- Synthesize blocks in raster order



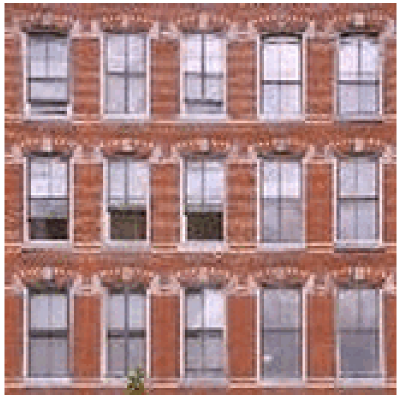
- Search input texture for block that satisfies overlap constraints (above and left)
- Paste new block into resulting texture
  - blending
  - use dynamic programming to compute minimal error boundary cut


block
Input texture
B1
B2
Random placement of blocks
B1
B2
Neighboring blocks constrained by overlap
B1
B2
Minimal error boundary cut

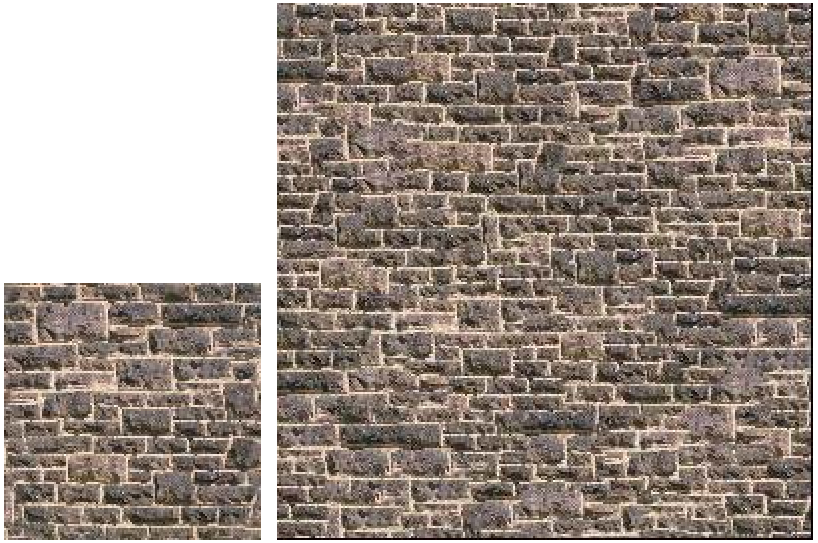
Minimal error boundary

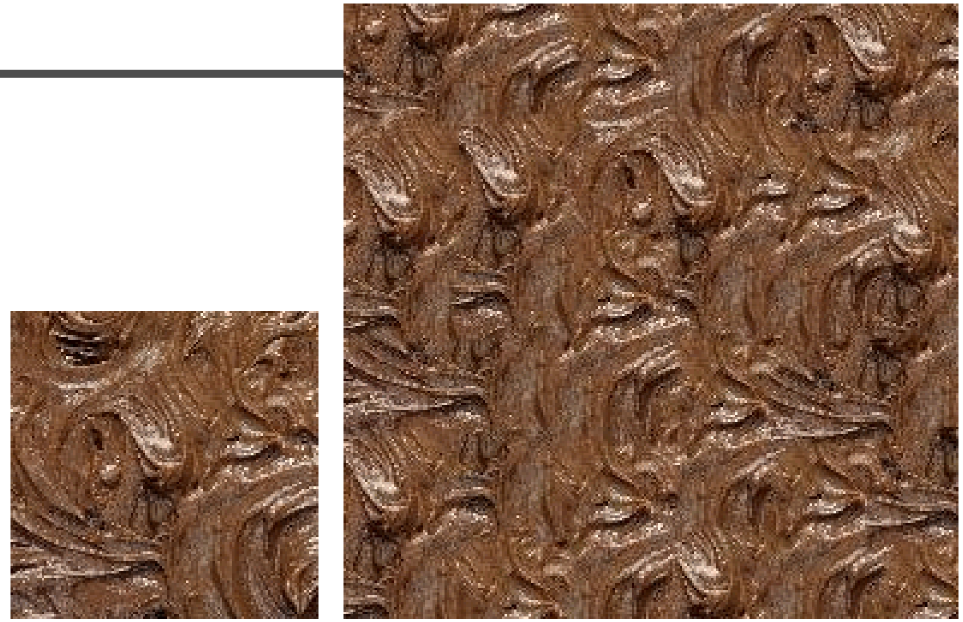


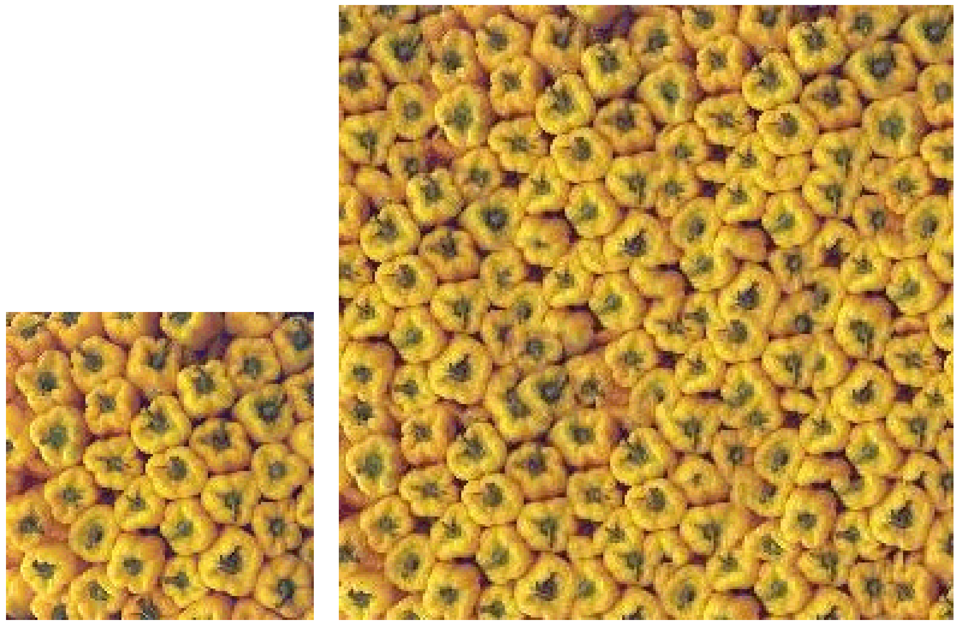
Results



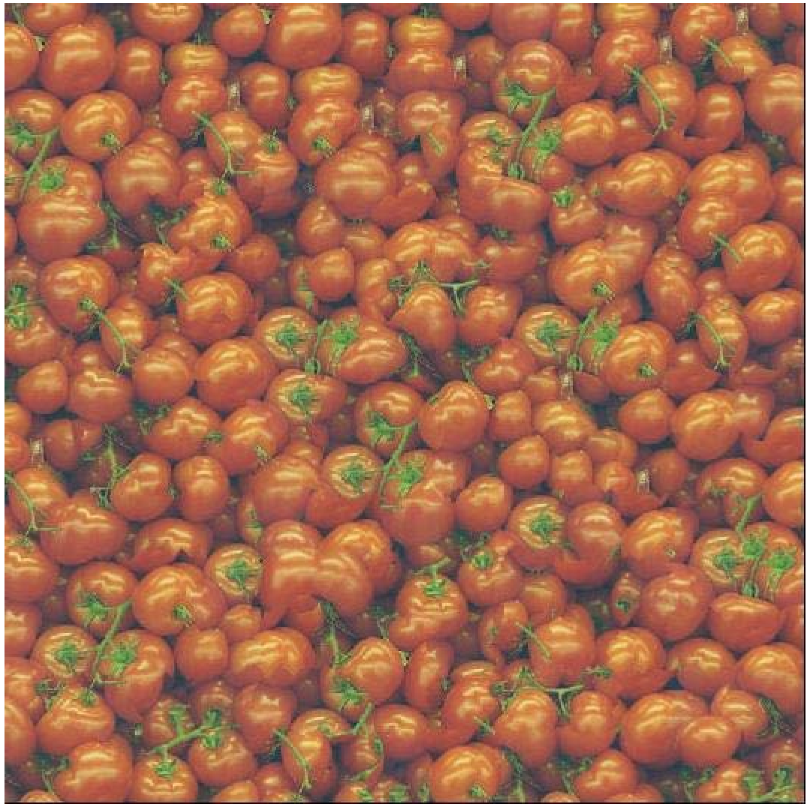
Results

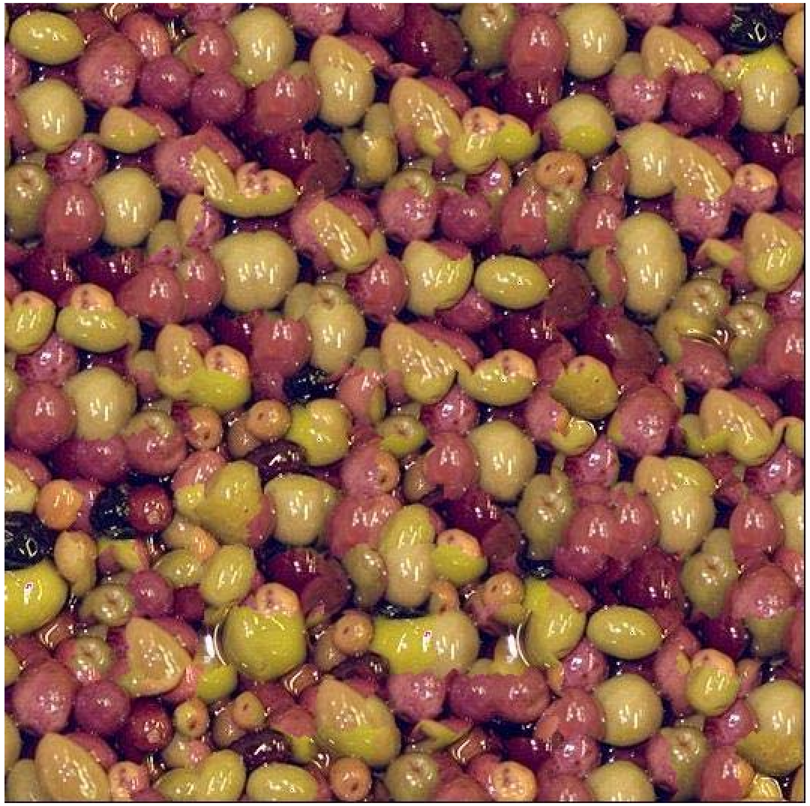






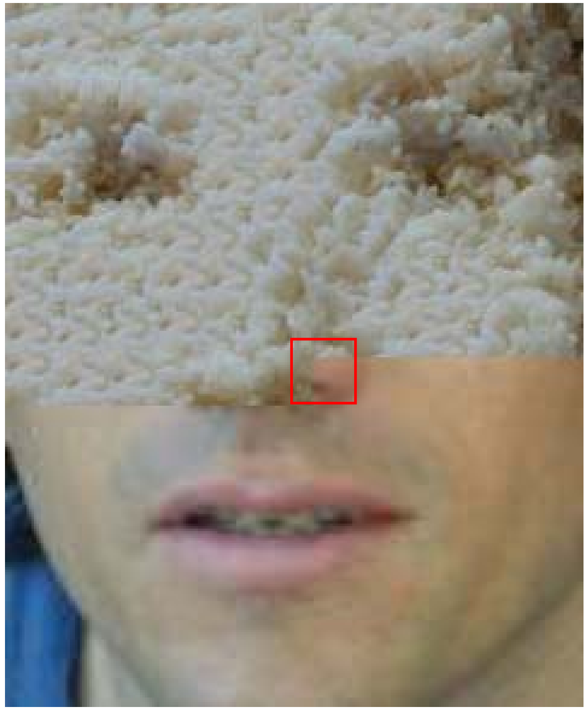
Failure cases



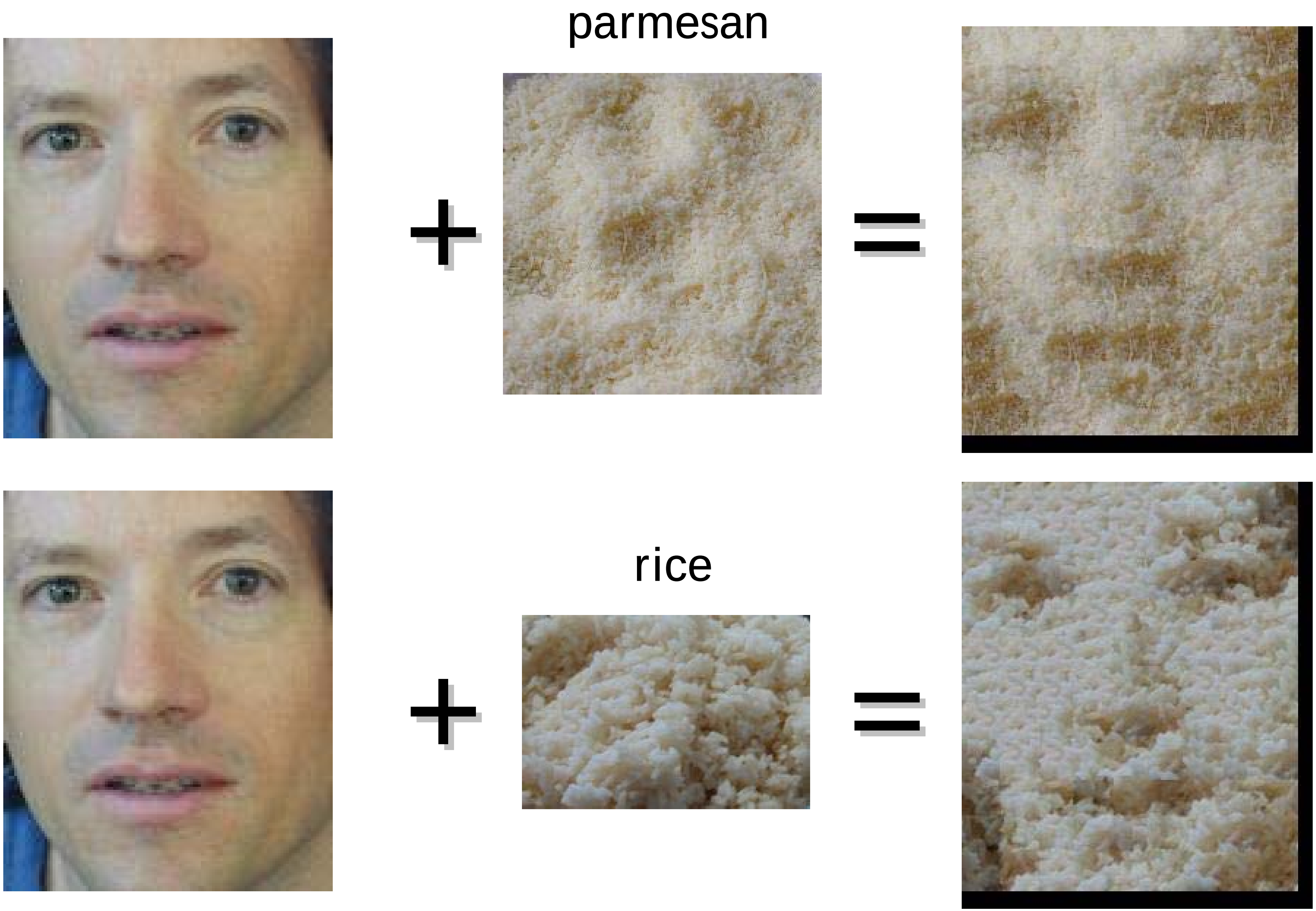


Texture transfer

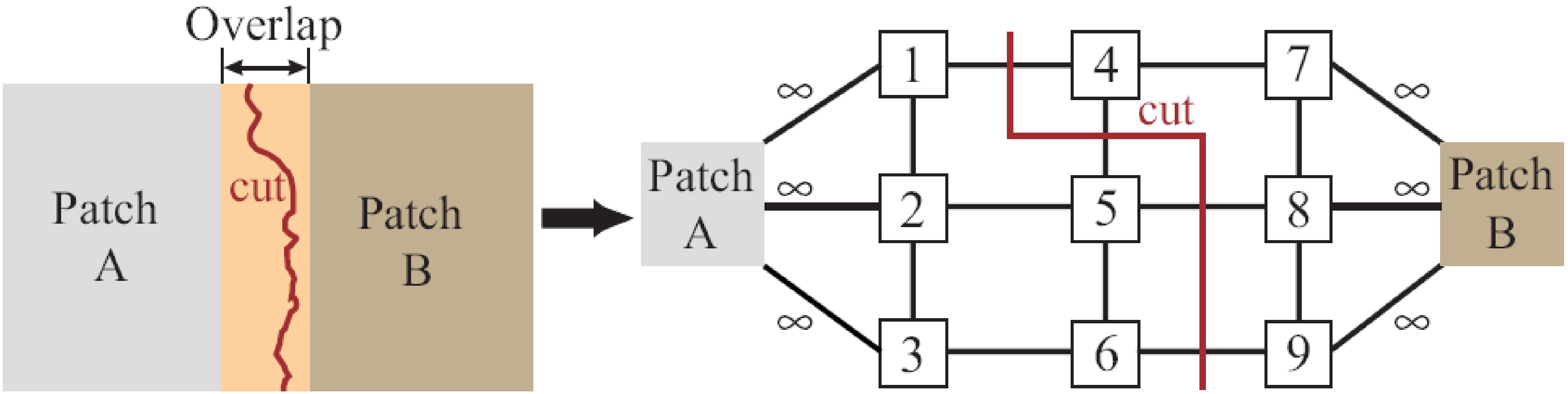
- Take the texture from one object and “paint” it onto another object



Then, just add another constraint when sampling: similarity to underlying image at that spot

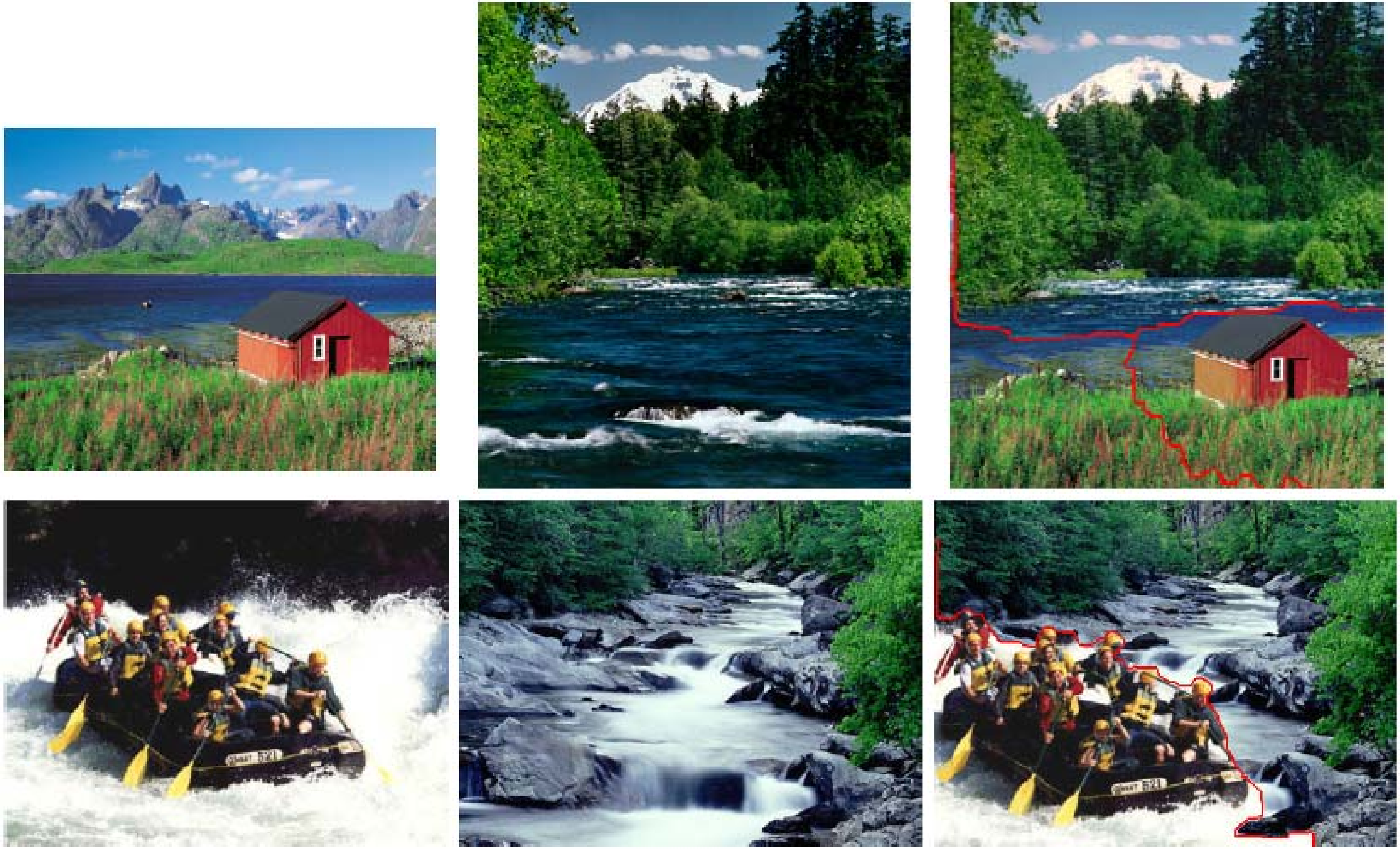

parmesan
rice

GraphCut textures

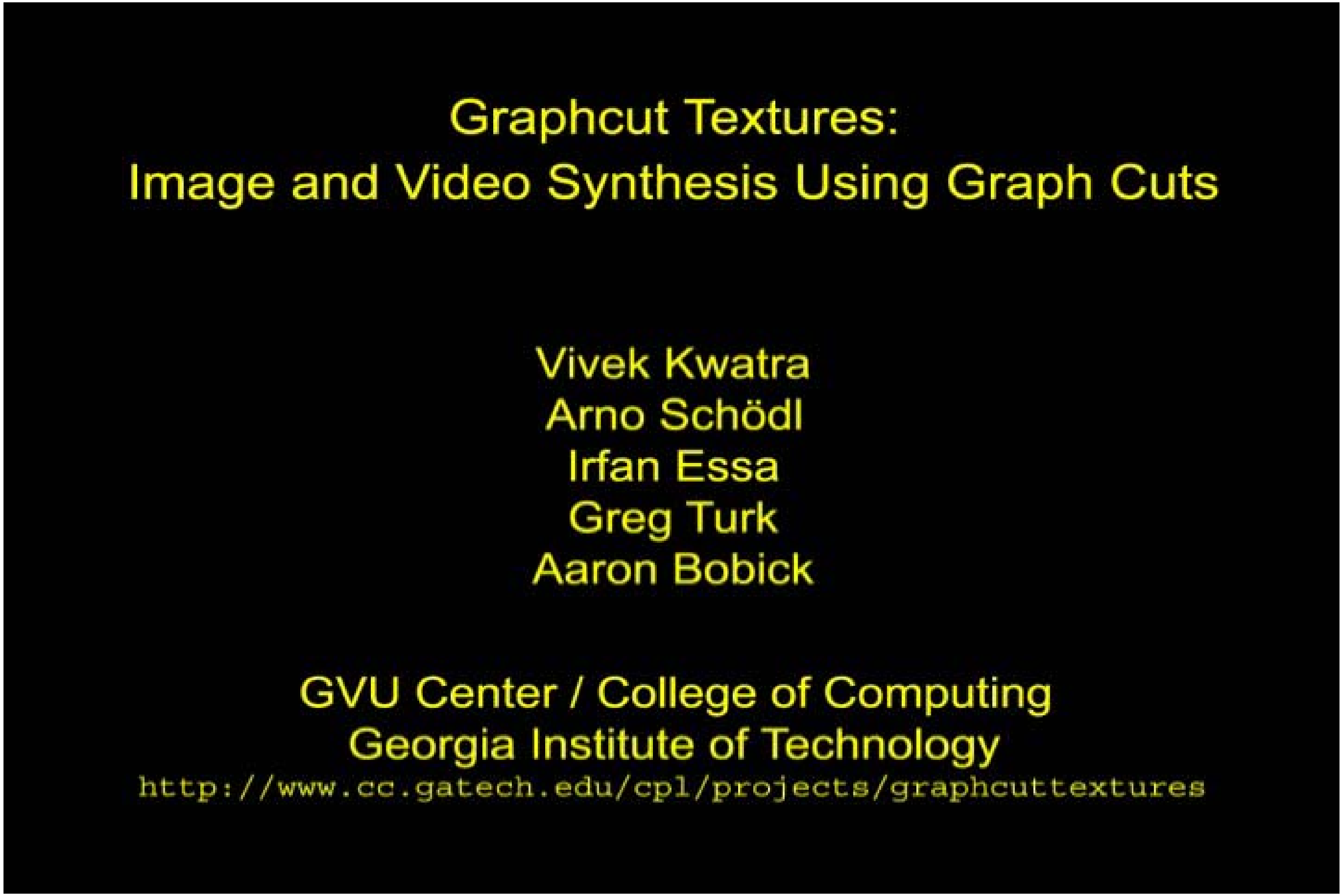


Input Image Quilting Graph cut

GraphCut textures



GraphCut textures

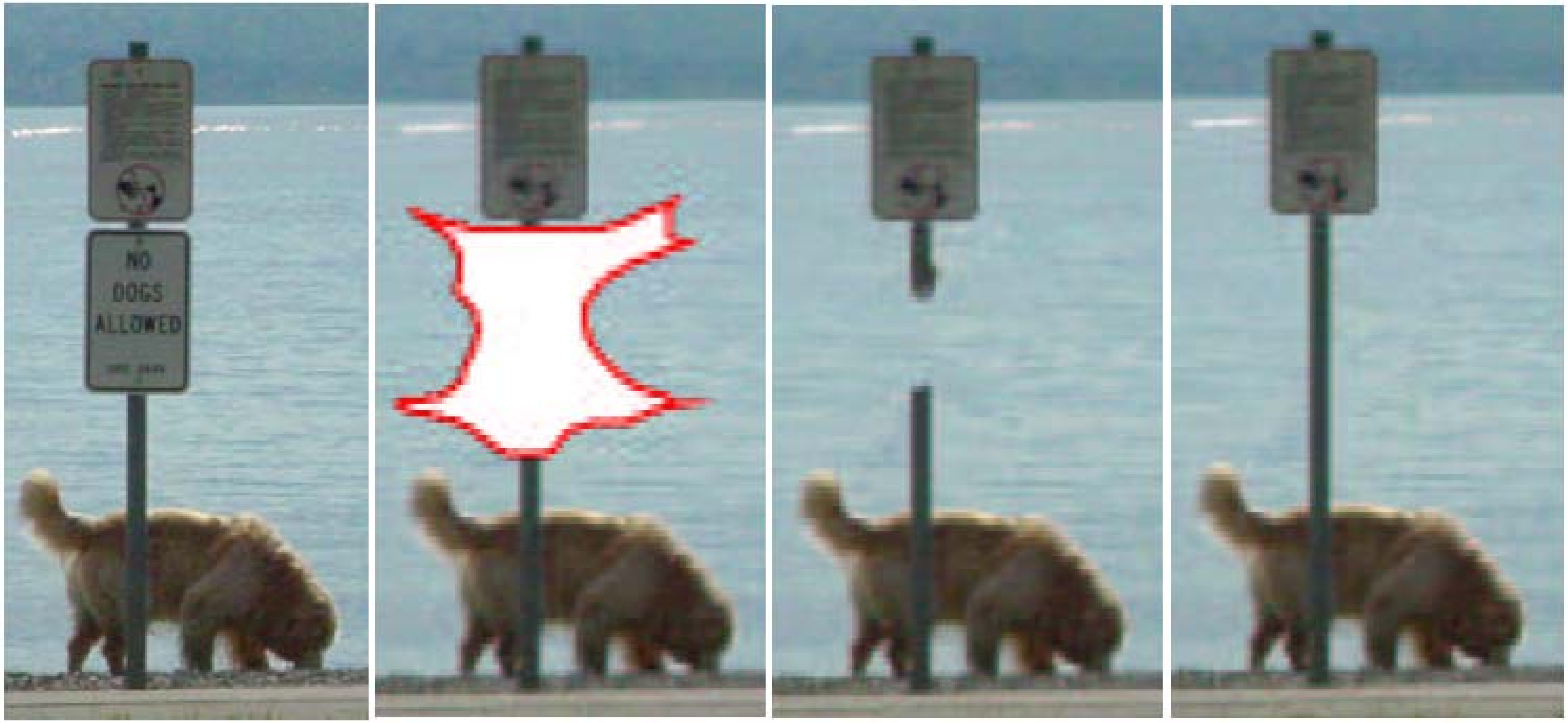


Inpainting

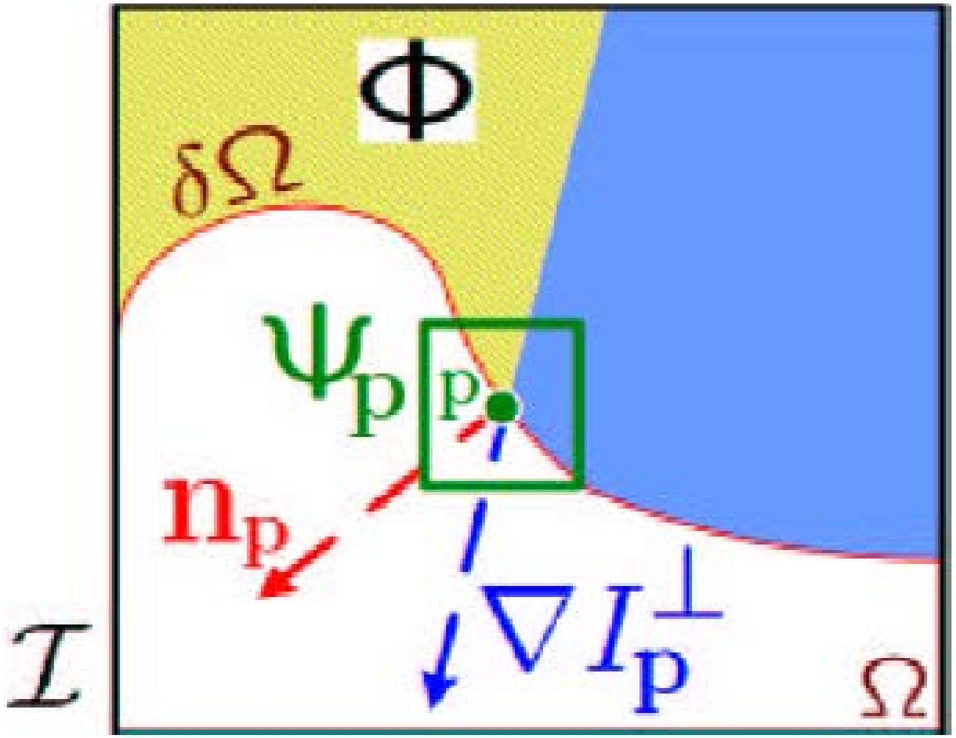
- Growing is in "onion peeling" order
  - within each "layer", pixels with most neighbors are synthesized first

Image extrapolation

Inpainting



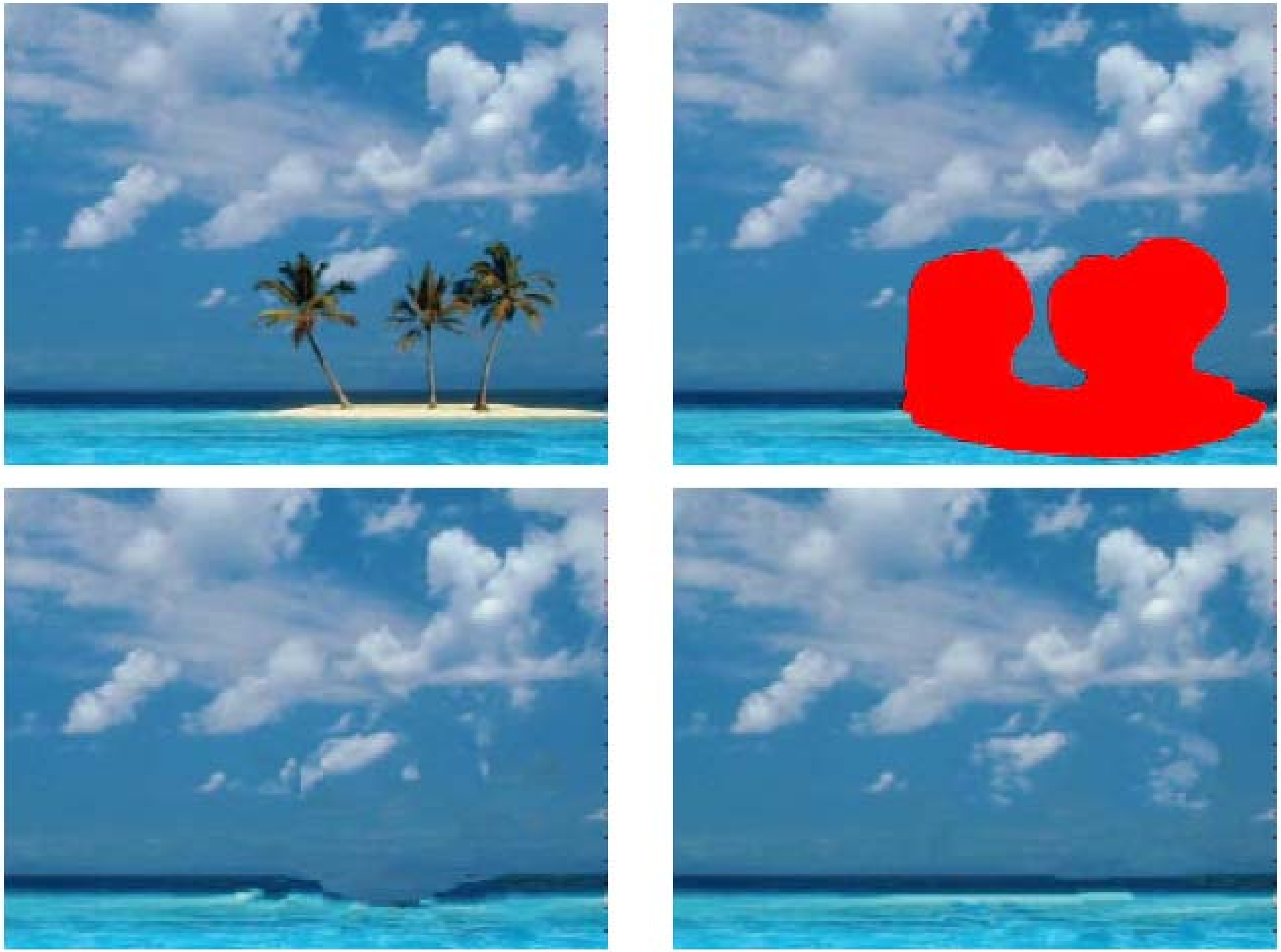
Inpainting



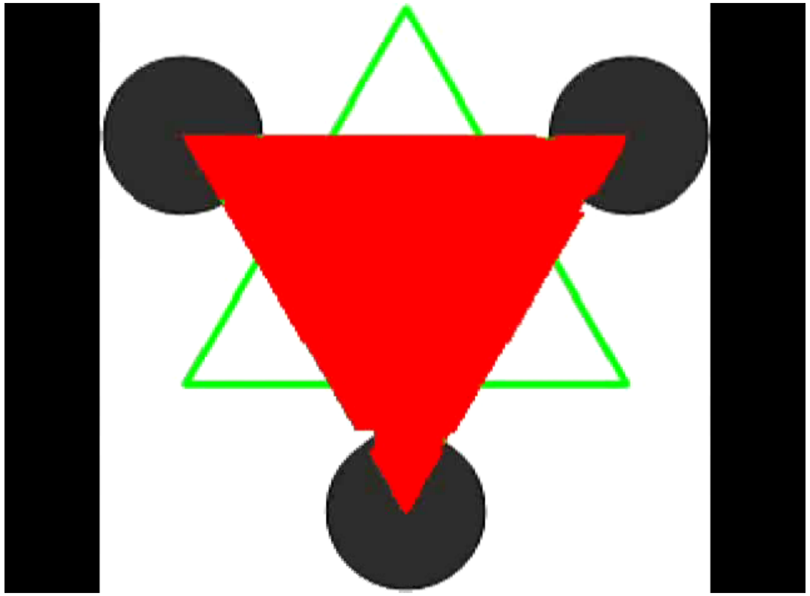
$$P(\mathbf{p}) = C(\mathbf{p})D(\mathbf{p})$$

$$C(\mathbf{p}) = \frac{\sum_{\mathbf{q}\in\Psi_{\mathbf{p}}\cap(\mathcal{I}-\Omega)} C(\mathbf{q})}{|\Psi_{\mathbf{p}}|}, \qquad D(\mathbf{p}) = \frac{|\nabla I_{\mathbf{p}}^{\perp}\cdot\mathbf{n}_{\mathbf{p}}|}{\alpha}$$

Results



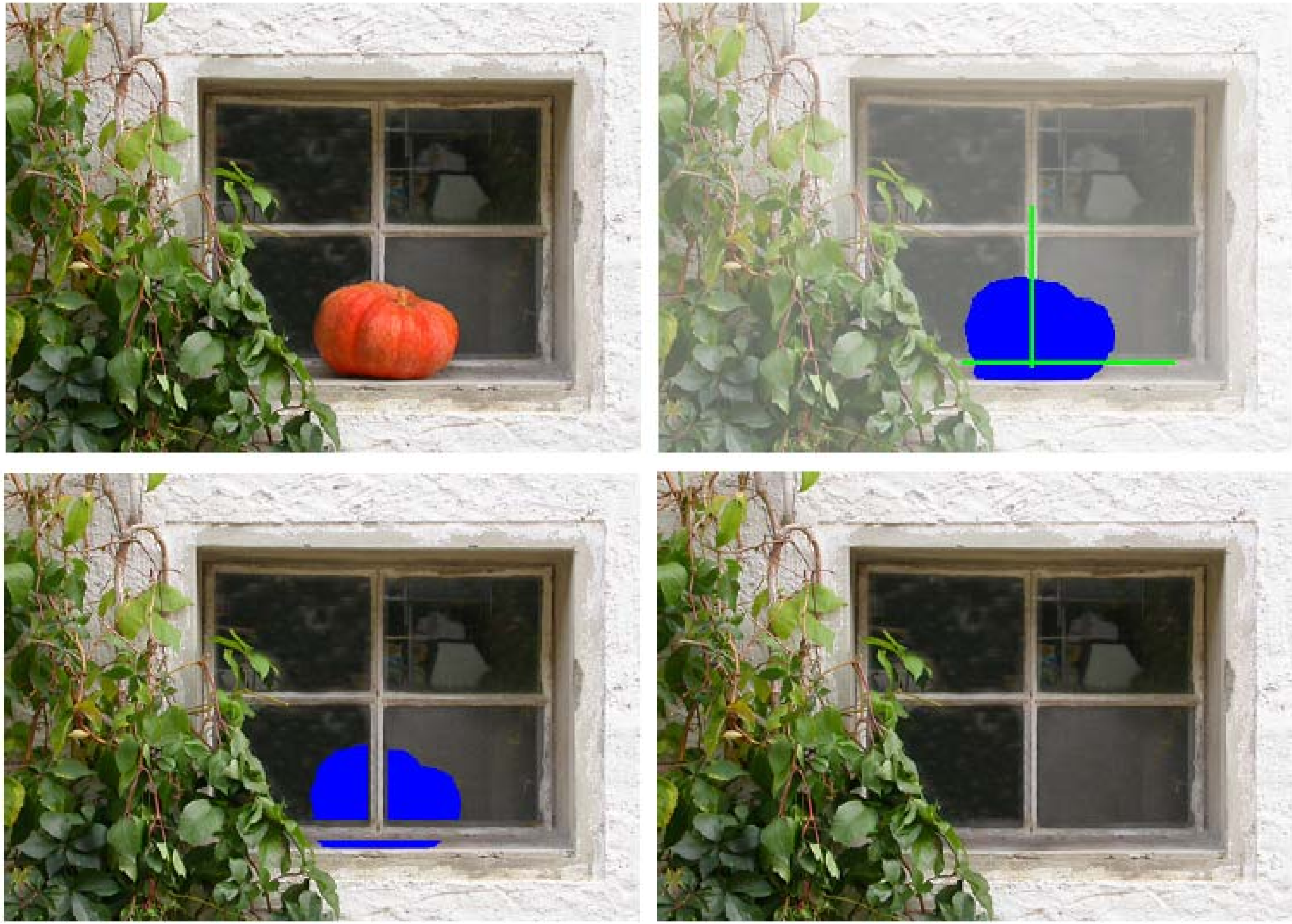
Results



Results

http://research.microsoft.com/vision/cambridge/i3l/patchworks.htm

Structure propagation



Structure propagation

Image Analogies

A

:

A'

::

B

:

?

Image Analogies Implementation

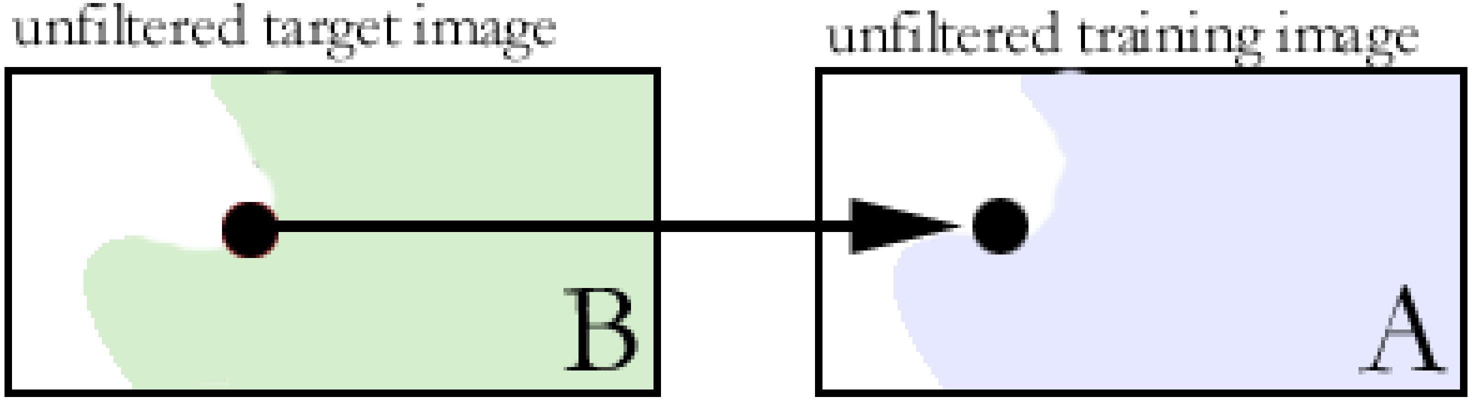


Image Analogies Implementation

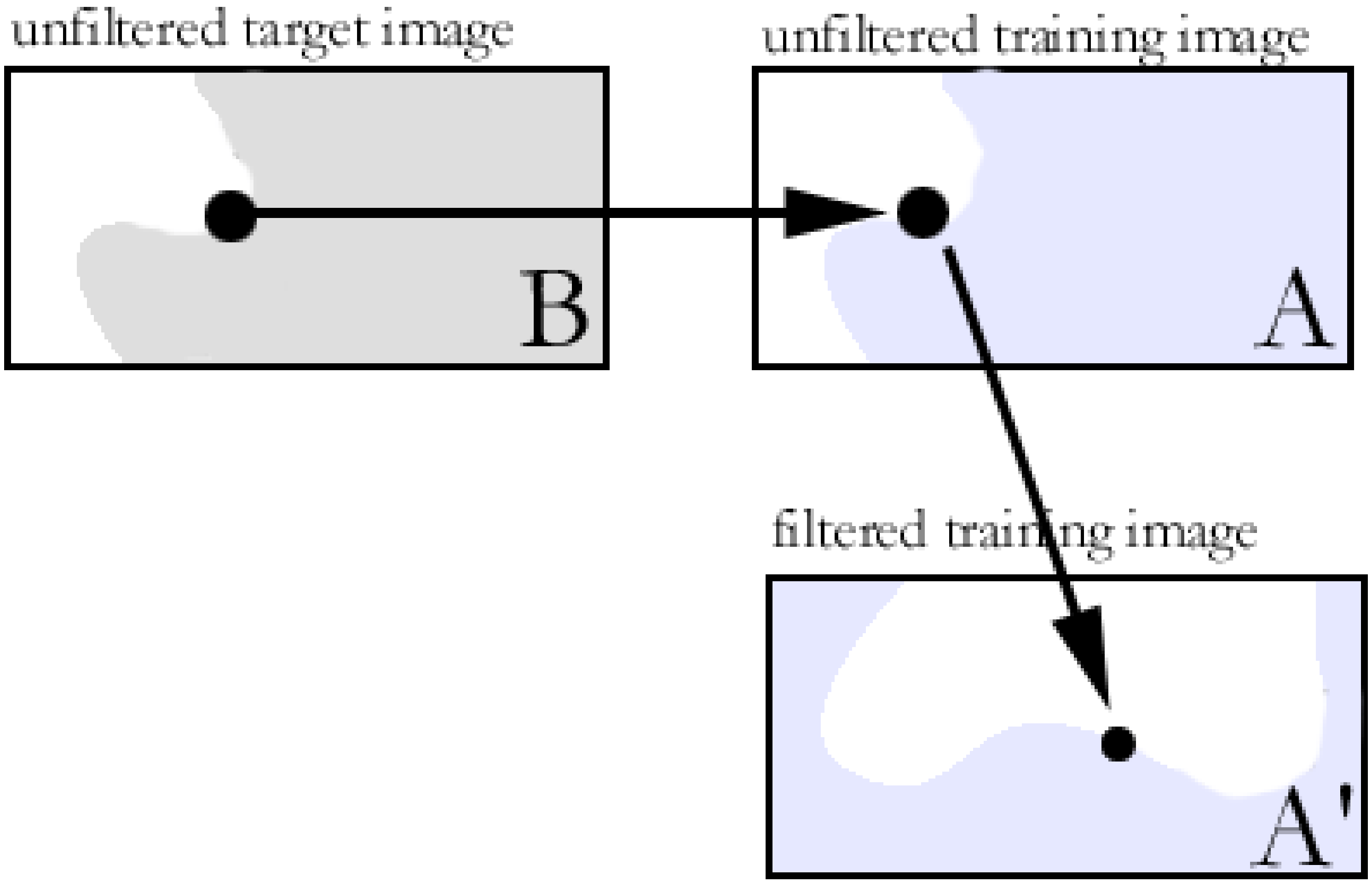
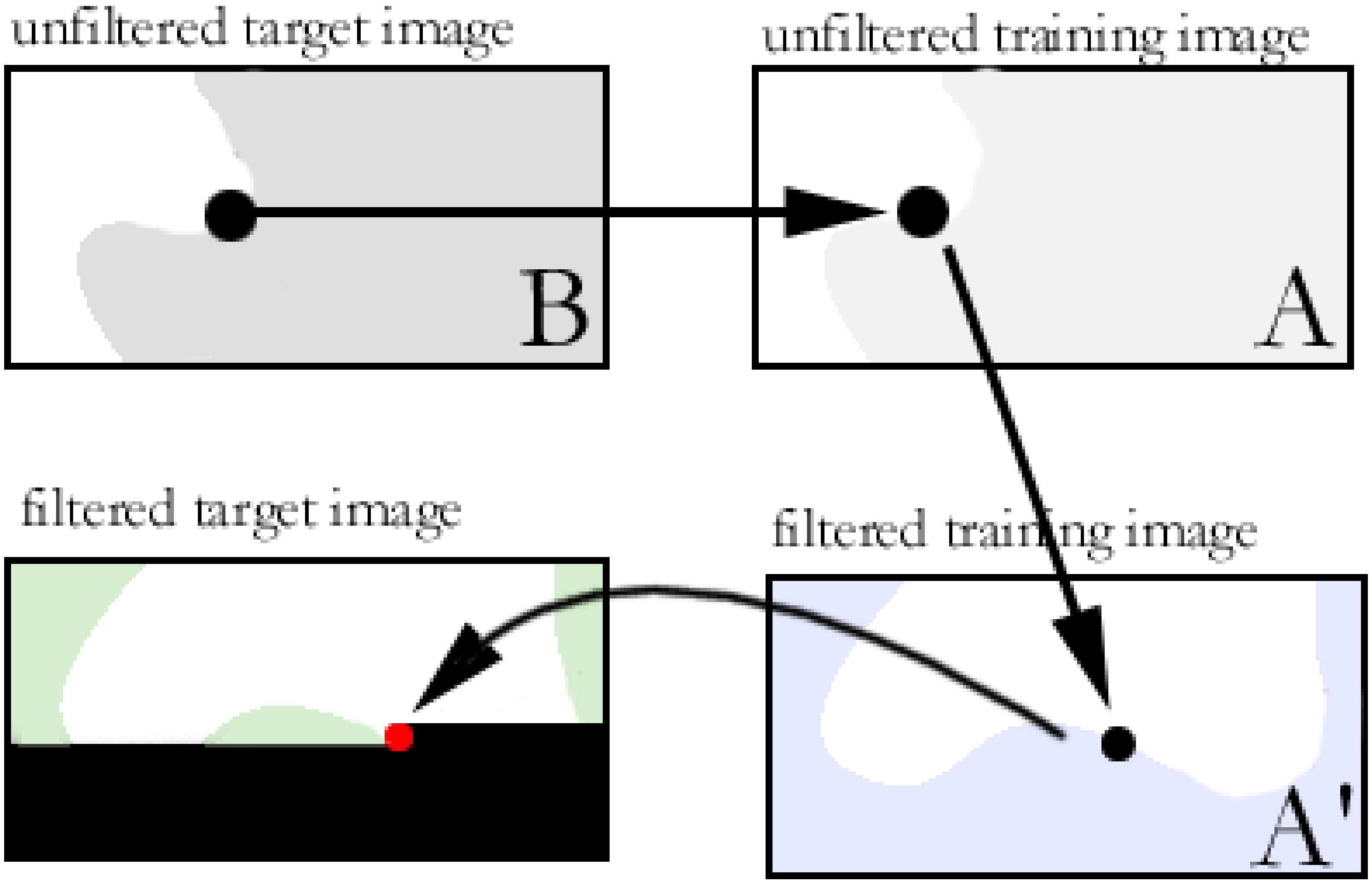


Image Analogies Implementation



Balance between approximate and coherence searches

function $\textsc{BestMatch}(A, A', B, B', s, \ell, q)$:

$p_{\text{app}} \leftarrow \textsc{BestApproximateMatch}(A, A', B, B', \ell, q)$

$p_{\text{coh}} \leftarrow \textsc{BestCoherenceMatch}(A, A', B, B', s, \ell, q)$

$d_{\text{app}} \leftarrow \|F_\ell(p_{\text{app}}) - F_\ell(q)\|^2$

$d_{\text{coh}} \leftarrow \|F_\ell(p_{\text{coh}}) - F_\ell(q)\|^2$

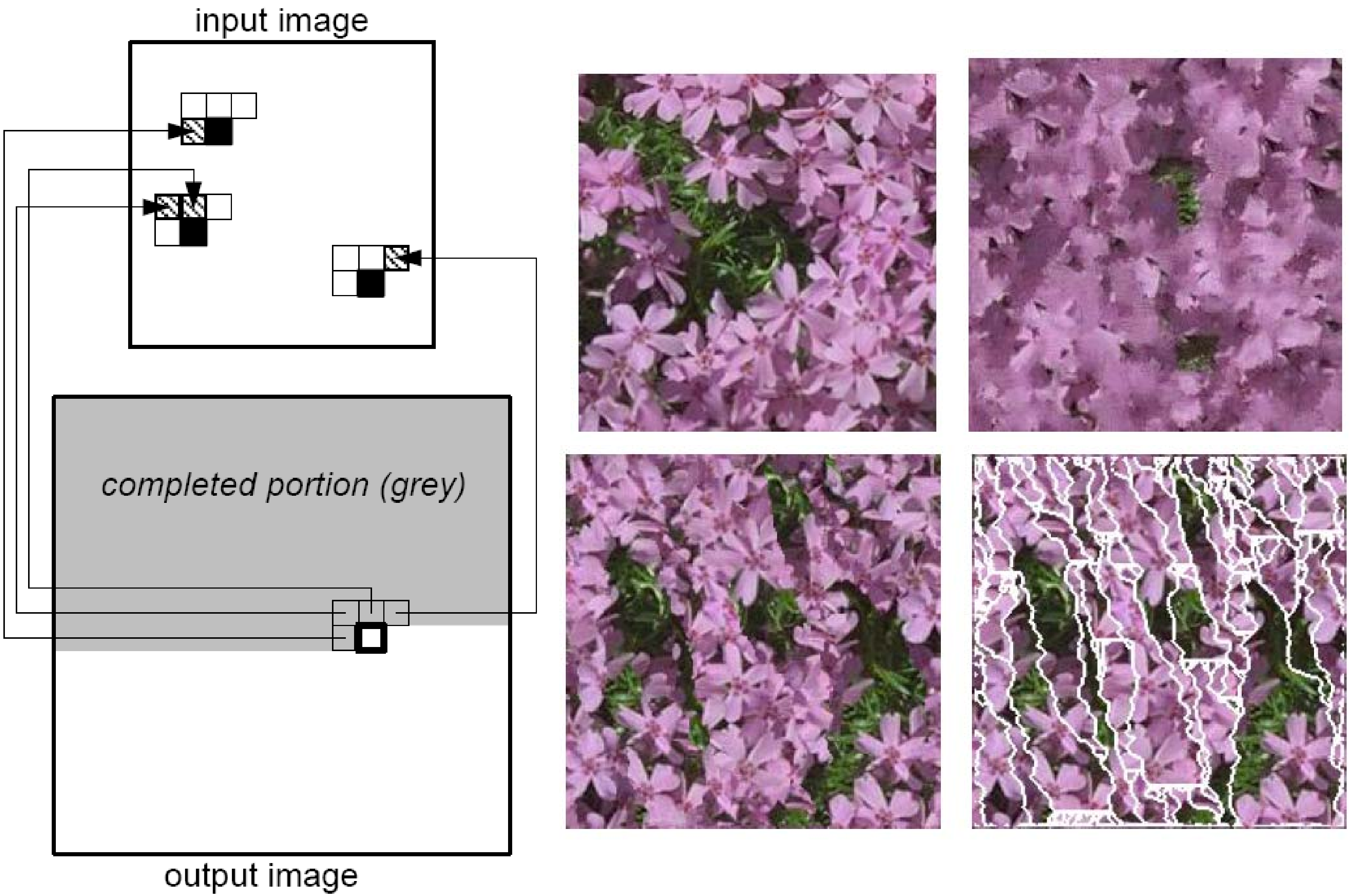
if $d_{\text{coh}} \leq d_{\text{app}}(1 + 2^{\ell - L}\kappa)$ **then**

  return p_{coh}

else

  return p_{app}

Coherence search



Learn to blur

Unfiltered source (A)

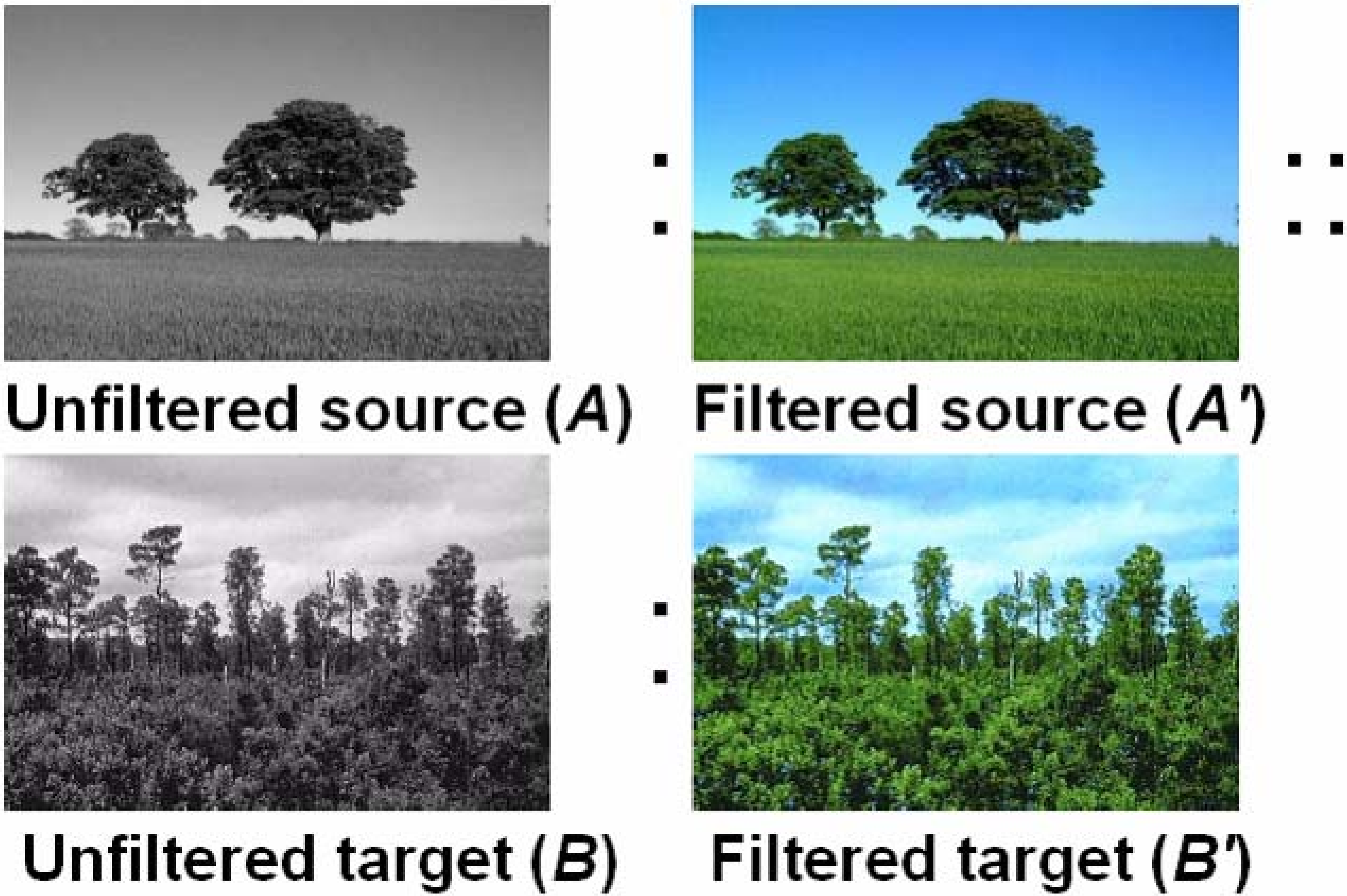
Filtered source (A')

Unfiltered target (B)

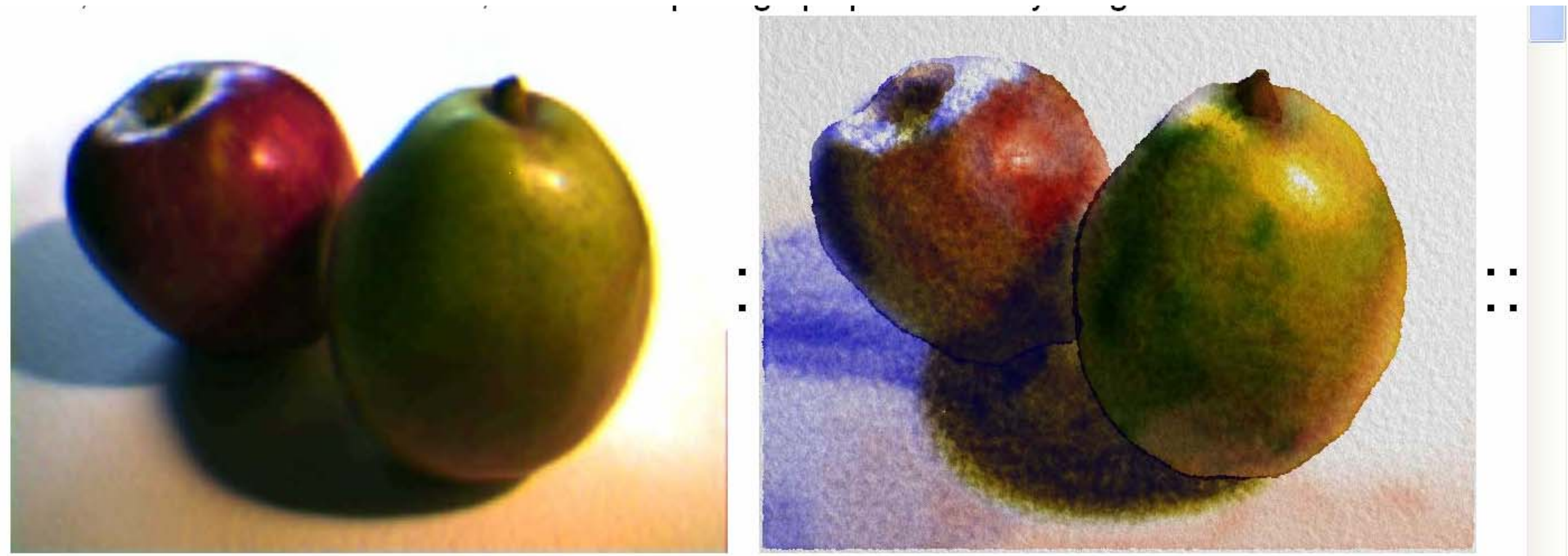
Filtered target (B')

Super-resolution

Colorization



Artistic filters



Unfiltered source (A) Filtered source (A')

: :  : 

B **B'**

::

:

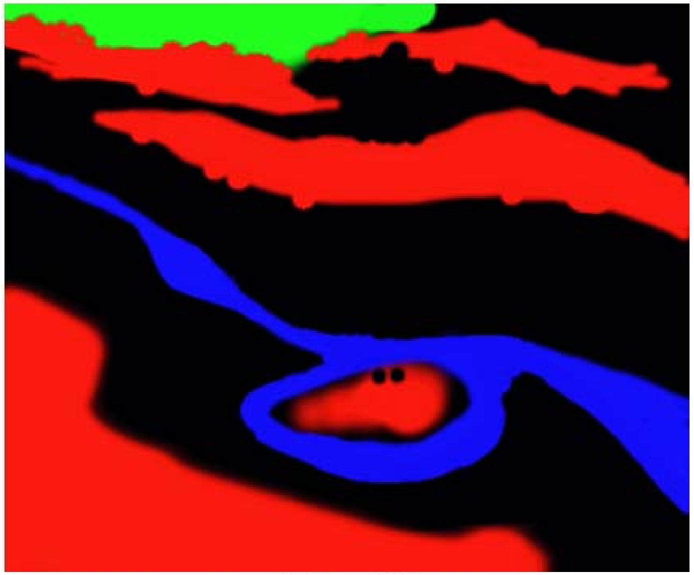
B **B'**

DigiVFX

Texture by numbers

Unfiltered source (A)

Filtered source (A')



Unfiltered (B)

Filtered (B')

Texture by numbers

Image Analogies

Aaron Hertzmann
Charles Jacobs
Nuria Oliver
Brian Curless
David Salesin

The Matrix Reloaded

The end!