

اصول پردازش تصویر

# *Principles of Image Processing*

مصطفی کمالی تبریزی

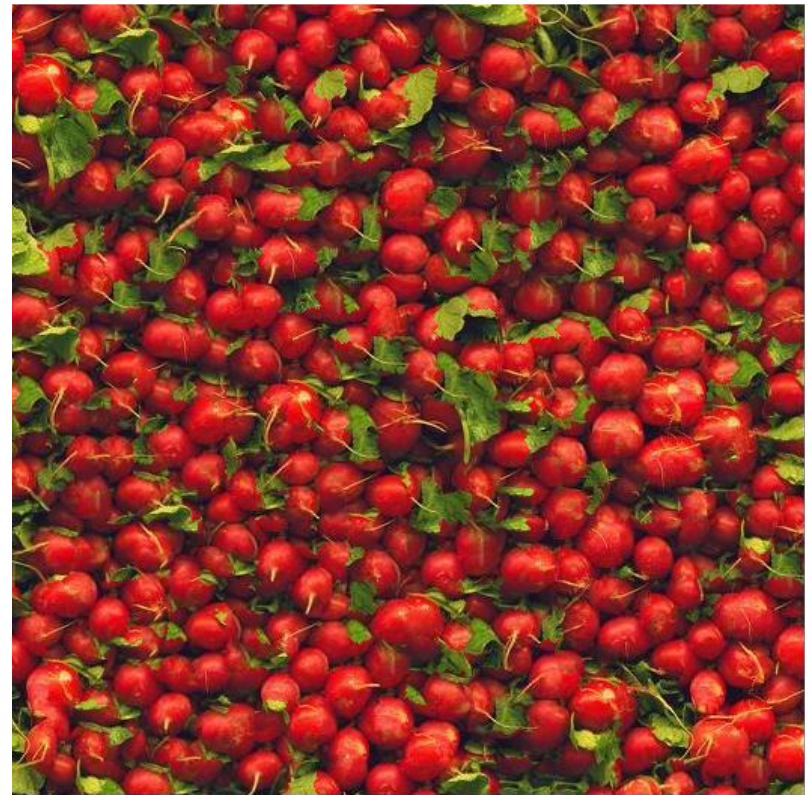
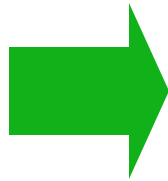
۱۰ و ۱۵ آذر ۱۳۹۹

جلسه بیست یکم و دوم

# Texture Synthesis

# Texture Synthesis

- Goal: create new samples of a given texture
- Many applications: virtual environments, hole-filling, texturing surfaces, ...





resize

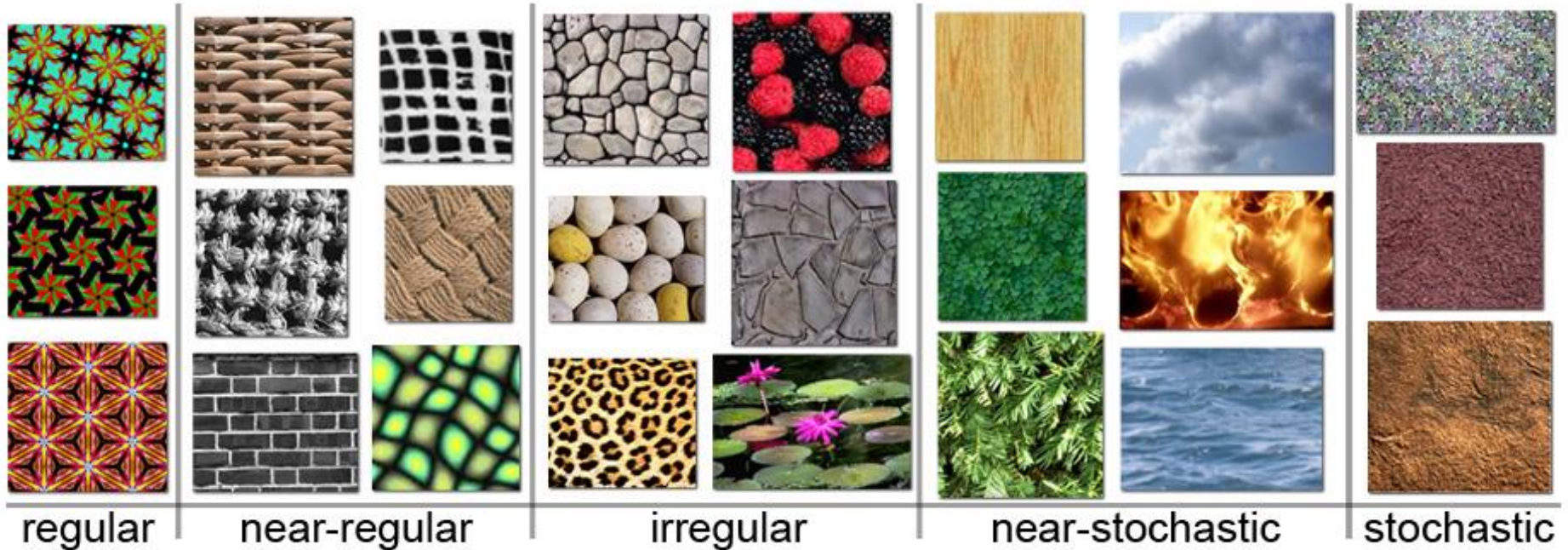


texture synthesis





# The Challenge



Need to model the whole spectrum:  
from repeated to stochastic texture

# One Idea: Build Probability Distributions

## Basic idea

1. Compute statistics of input texture  
(e.g., histogram of edge filter responses)
2. Generate a new texture that keeps those same statistics



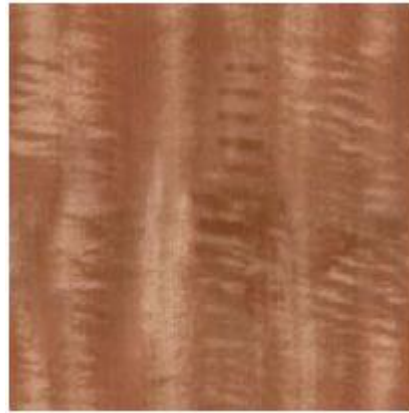
- D. J. Heeger and J. R. Bergen. (*SIGGRAPH 1995*)  
Pyramid-based texture analysis/synthesis.
- E. P. Simoncelli and J. Portilla. (*ICIP 1998*)  
Texture characterization via joint statistics of wavelet coefficient magnitudes.

# One Idea: Build Probability Distributions

But it (usually) doesn't work

- Probability distributions are hard to model well

Input

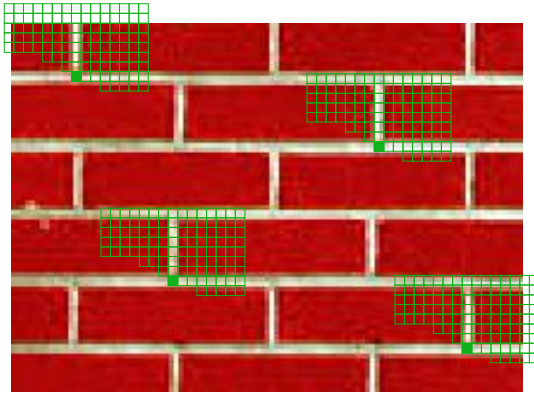


Synthesized

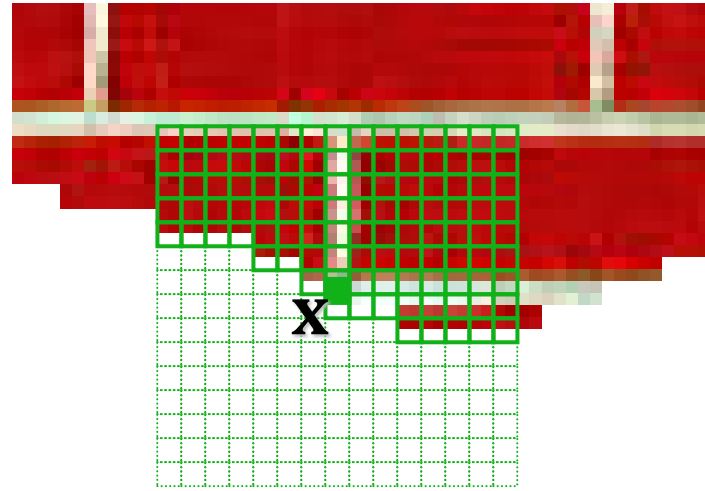




# Synthesizing One Pixel



input image



synthesized image

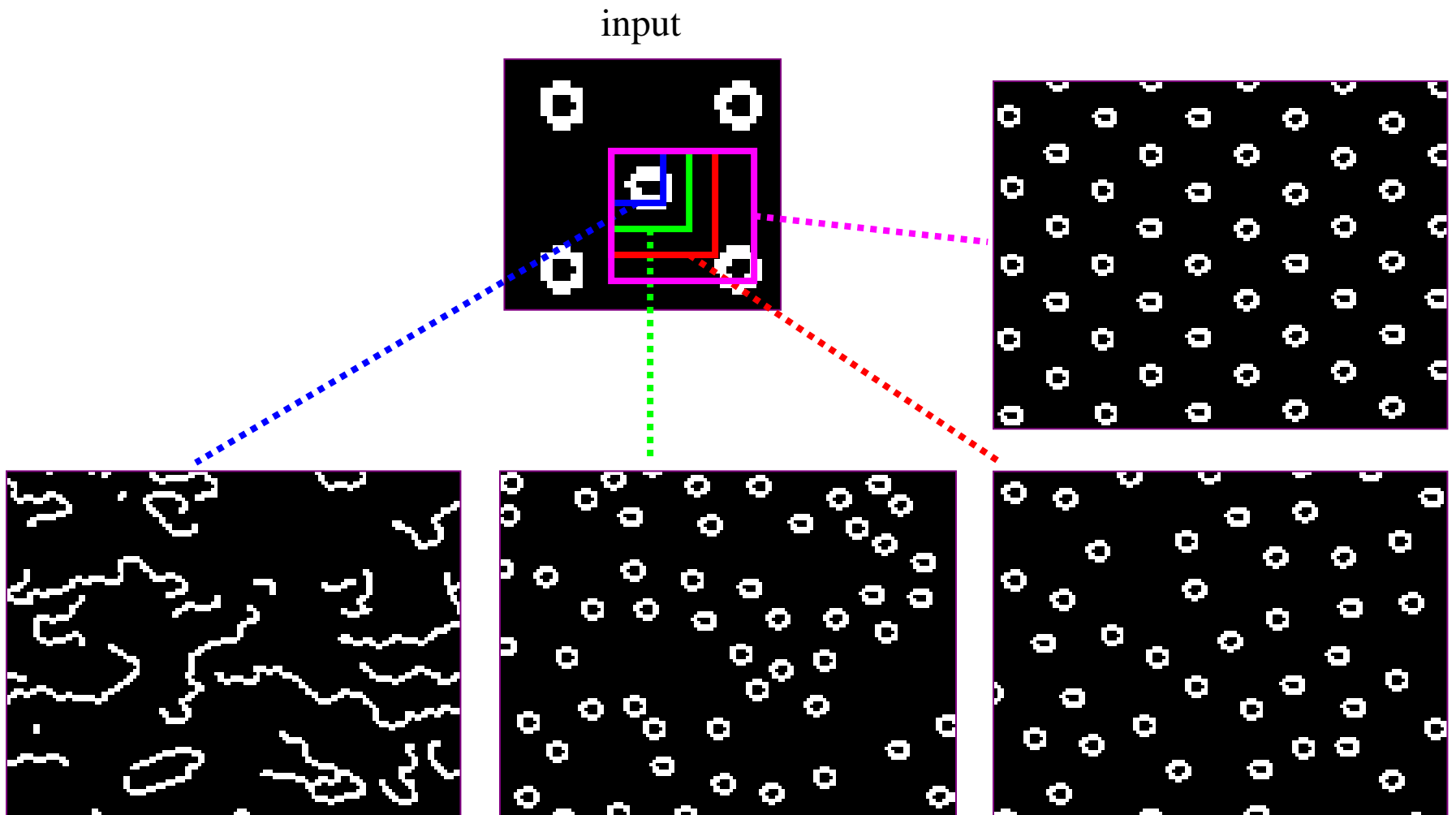
- What is  $p(\mathbf{x} \mid \text{neighborhood of pixels around } \mathbf{x})$ ?
- Find all the windows in the image that match the neighborhood
- To synthesize  $\mathbf{x}$ 
  - pick one matching window at random
  - assign  $\mathbf{x}$  to be the center pixel of that window
- An **exact** neighbourhood match might not be present, so find the **best** matches using **SSD error** and randomly choose between them, preferring better matches with higher probability



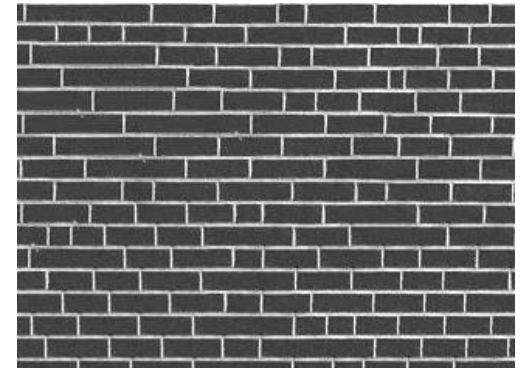
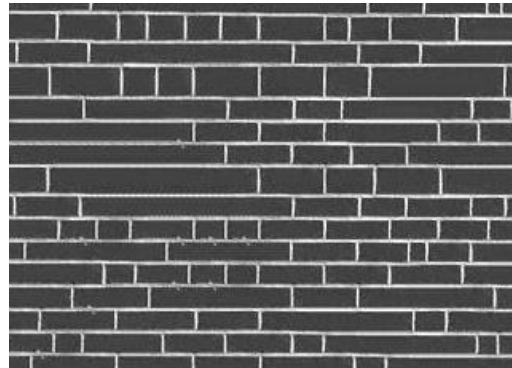
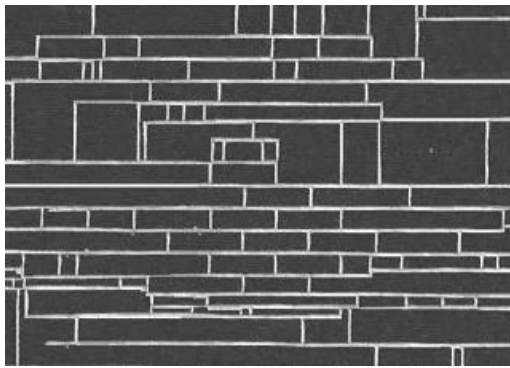
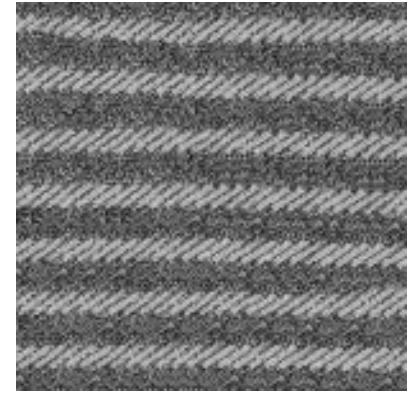
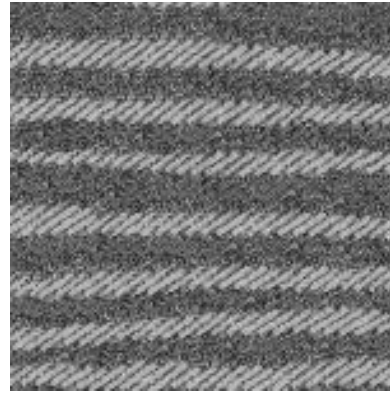
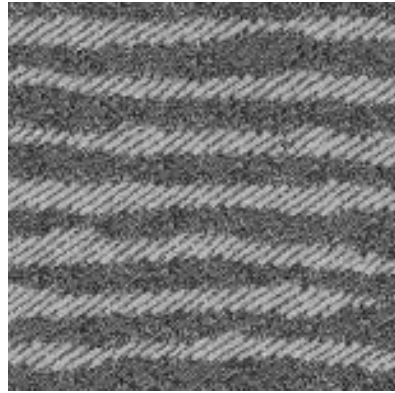
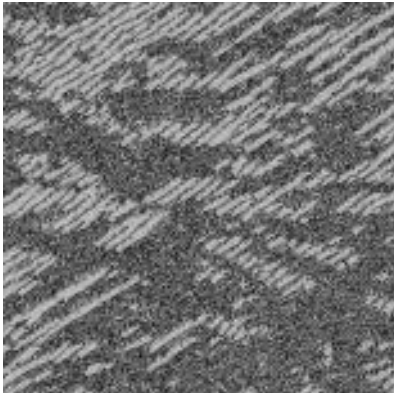
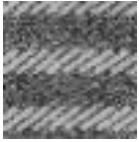
# Details

- How to match patches?
  - Gaussian-weighted SSD  
(more emphasis on nearby pixels)
- What order to fill in new pixels?
  - “Onion skin” order: pixels with most neighbors are synthesized first
  - To synthesize from scratch, start with a randomly selected small patch from the source texture
- How big should the patches be?

# Size of Neighborhood Window



# Varying Window Size



Increasing window size



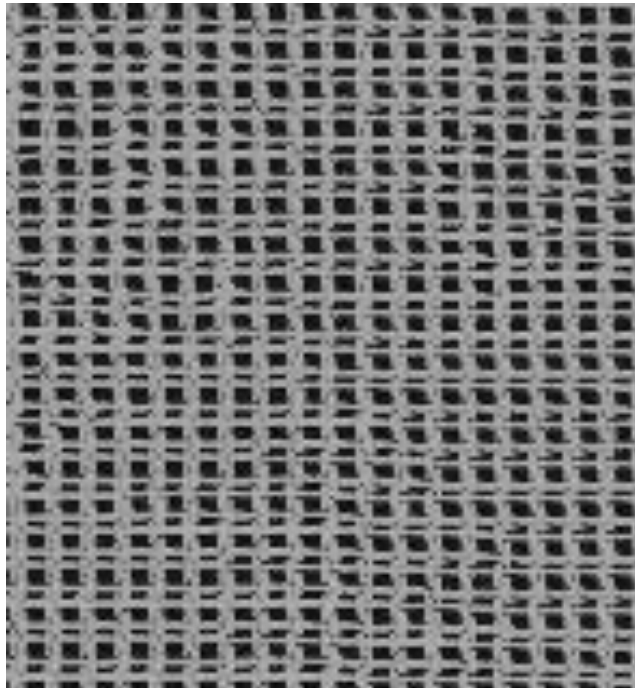
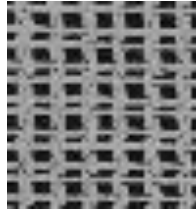
# Texture Synthesis Algorithm

- While image not filled
  1. Get unfilled pixels with filled neighbors, sorted by number of filled neighbors
  2. For each pixel, get top  $n$  matches based on visible neighbors
    - Patch Distance: Gaussian-weighted SSD
  3. Randomly select one of the matches and copy pixel from it

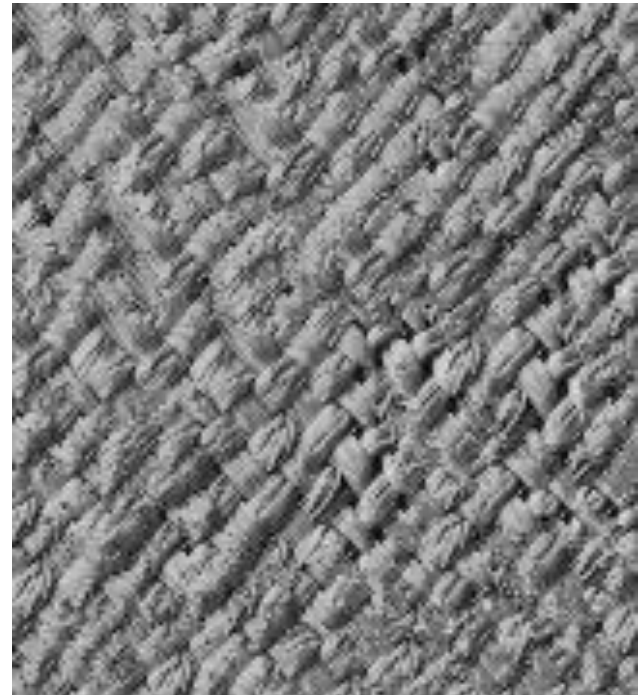


# Synthesis Results

French canvas



Raffia weave

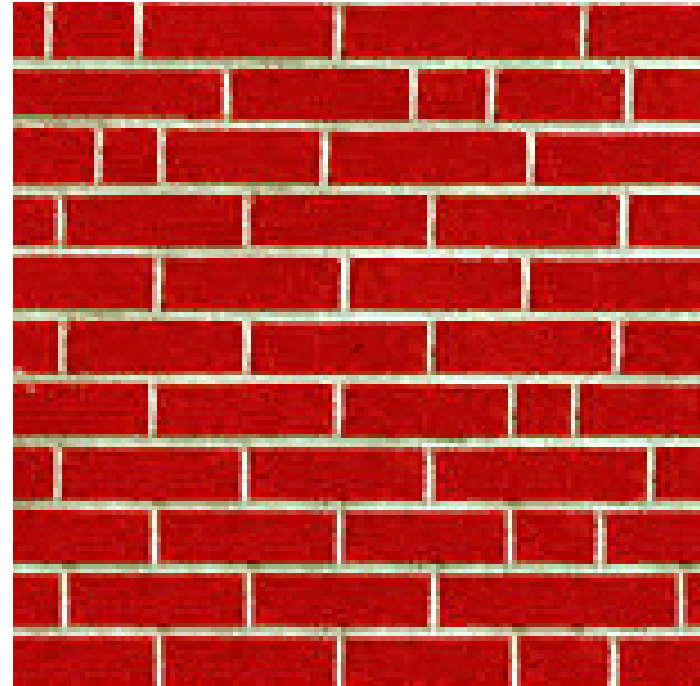
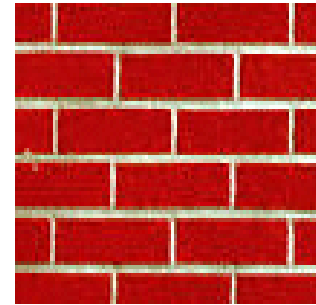


# More Results

White bread



Brick wall



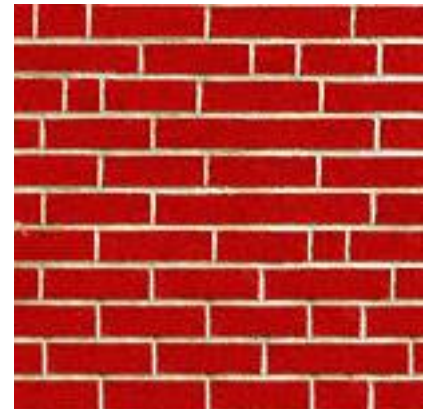
# Homage to Shannon

oming in the unsensational  
r Dick Gephardt was fair  
rful riff on the looming  
nly asked, "What's your  
tions?" A heartfelt sigh  
story about the emergen  
es against Clinton. "Boy  
g people about continuin  
ardt began, patiently obs  
s, that the legal system k  
g with this latest tanger



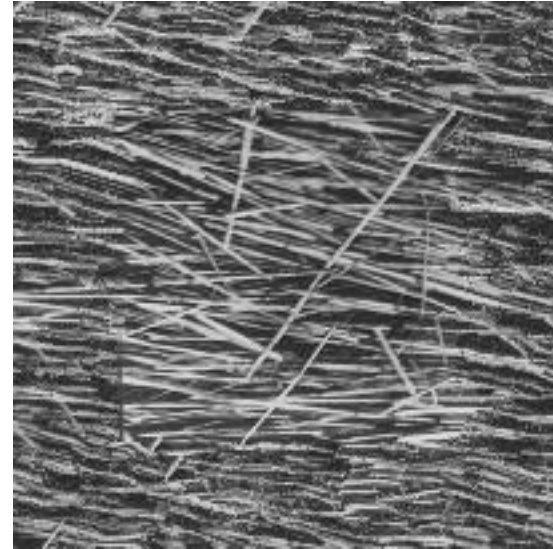
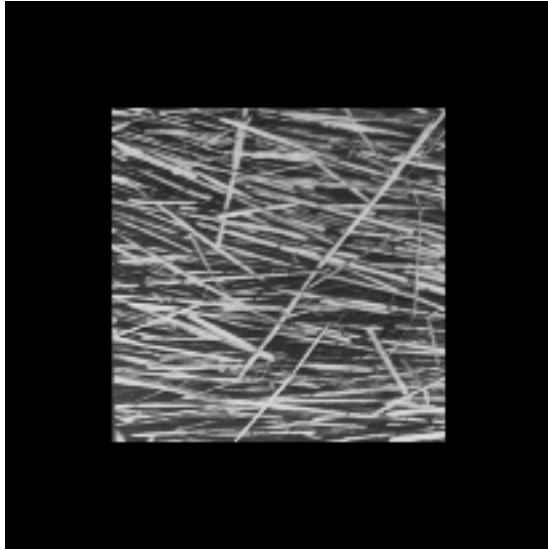
ithaim. them . "Whnephartfe lartifelintomimen  
fel ck Clirtioout omaim thartfelins.f out s anetc  
the ry onst wartfe lck Gephtoomimeationl sigak  
Chiooufit Clinut Cll riff on. hat's yodn, parut tly  
ons yootonsteht wasked, paim t sahe loo riff on l  
nskoneploourtfeas leil A nst Clit, "Wleontongal s  
k Clirtioouirtfepe.ong pme abegal fartfenstemem  
tiensteneltorydt telemephminsverdt was agemer  
ff ons artientont Cling peme as artfe atich, "Boui s  
aal s fartfelt sig pedrtldt ske abounutie aboutioo  
tfaonewwas yow abowonthardt thatins fain, ped, '  
ains. them, pabout wasy arfint couitly d, l n A h  
le emthrdngbooreme agas fa bontinsyst Clinut  
ory about continst Clipeopinst Cloke agatiff out C  
stome zinemen tly ardt beoraboul n, thenly as t C  
cons faimeme Diontont wat coutlyohgans as fan  
ien, phrtfaul, "Wbaut cout congagal comininga  
mifmst Cliiy abon al coountha.emungait tfoun  
Vhe loocrystan loontieph. intly on, tieoplegatick C  
aul fatiecontly atie Diontiomt wal s f tbegae ener  
nthahsat's enenhhhoas fan. "intchthorv abonsu

# Hole Filling





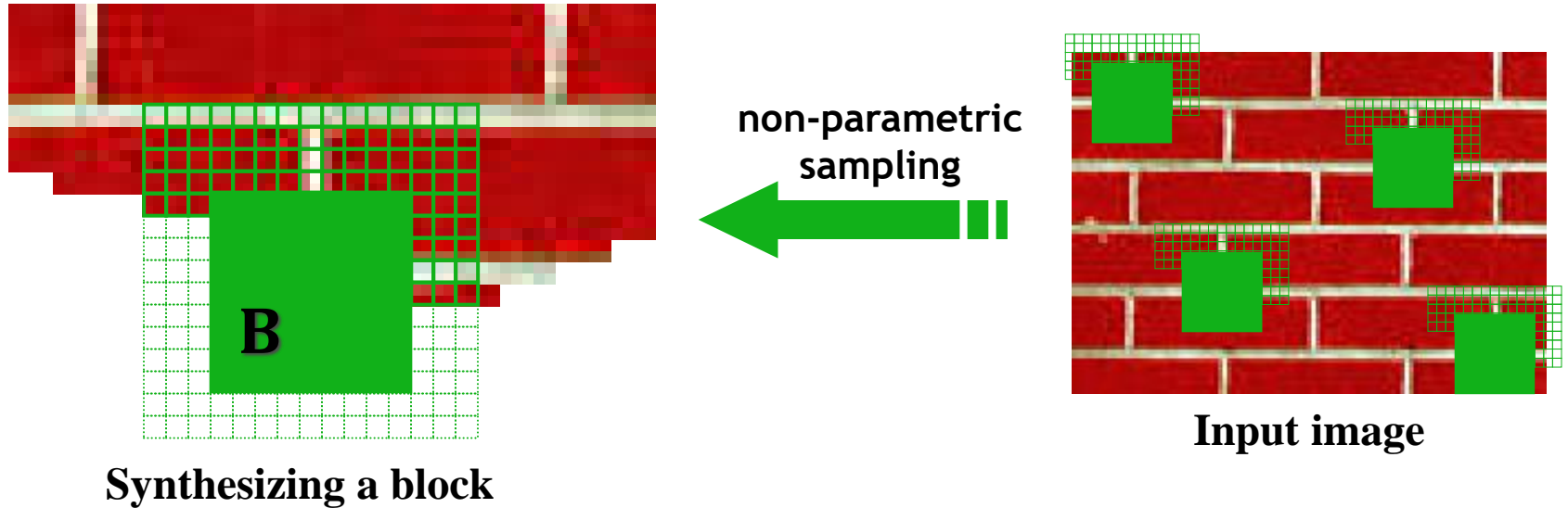
# Extrapolation



# Summary

- The Efros & Leung texture synthesis algorithm
  - Very simple
  - Surprisingly good results
  - Synthesis is easier than analysis!
  - ...but very slow

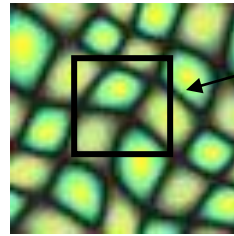
# Image Quilting [Efros & Freeman 2001]



- Observation: neighbor pixels are highly correlated

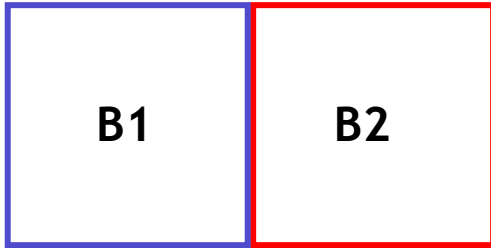
**Idea: unit of synthesis = block**

- Exactly the same but now we want  $p(\mathbf{B} \mid N(\mathbf{B}))$
- Much faster: synthesize all pixels in a block at once

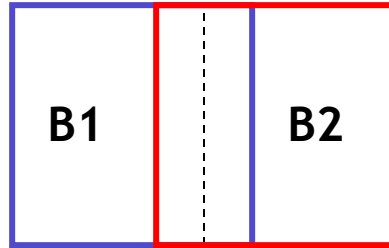


block

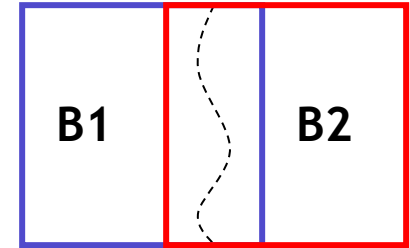
**Input texture**



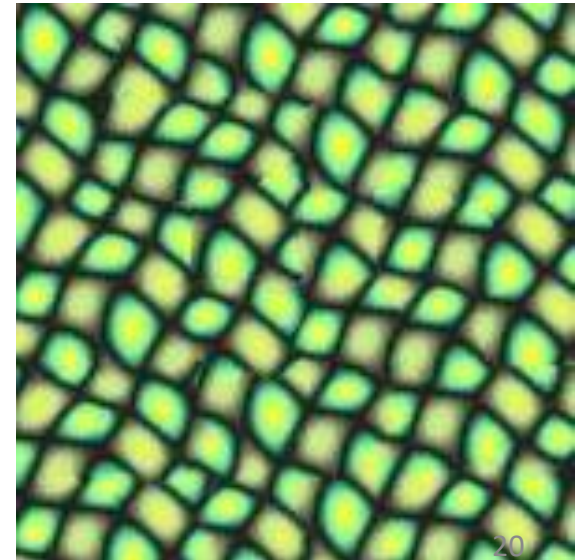
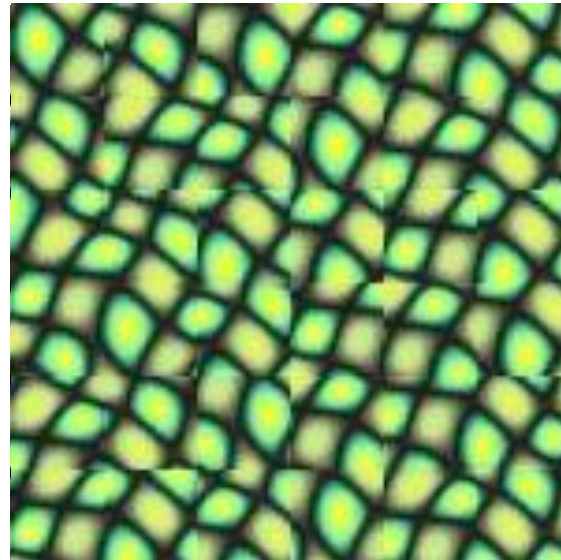
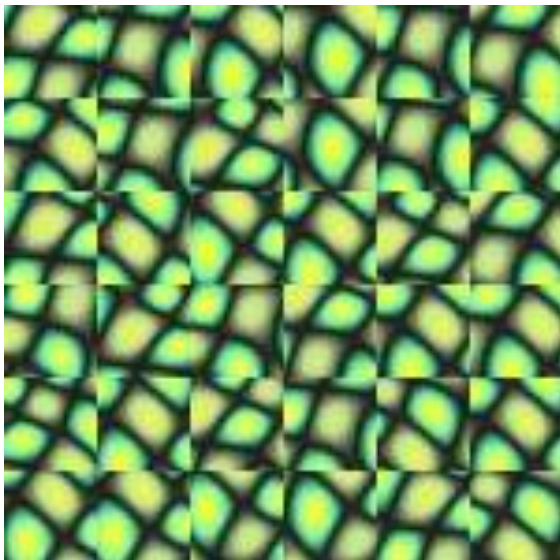
**Random placement  
of blocks**



**Neighboring blocks  
constrained by overlap**

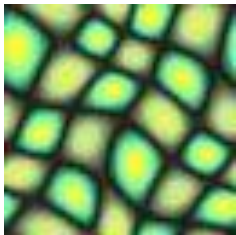


**Minimal error  
boundary cut**



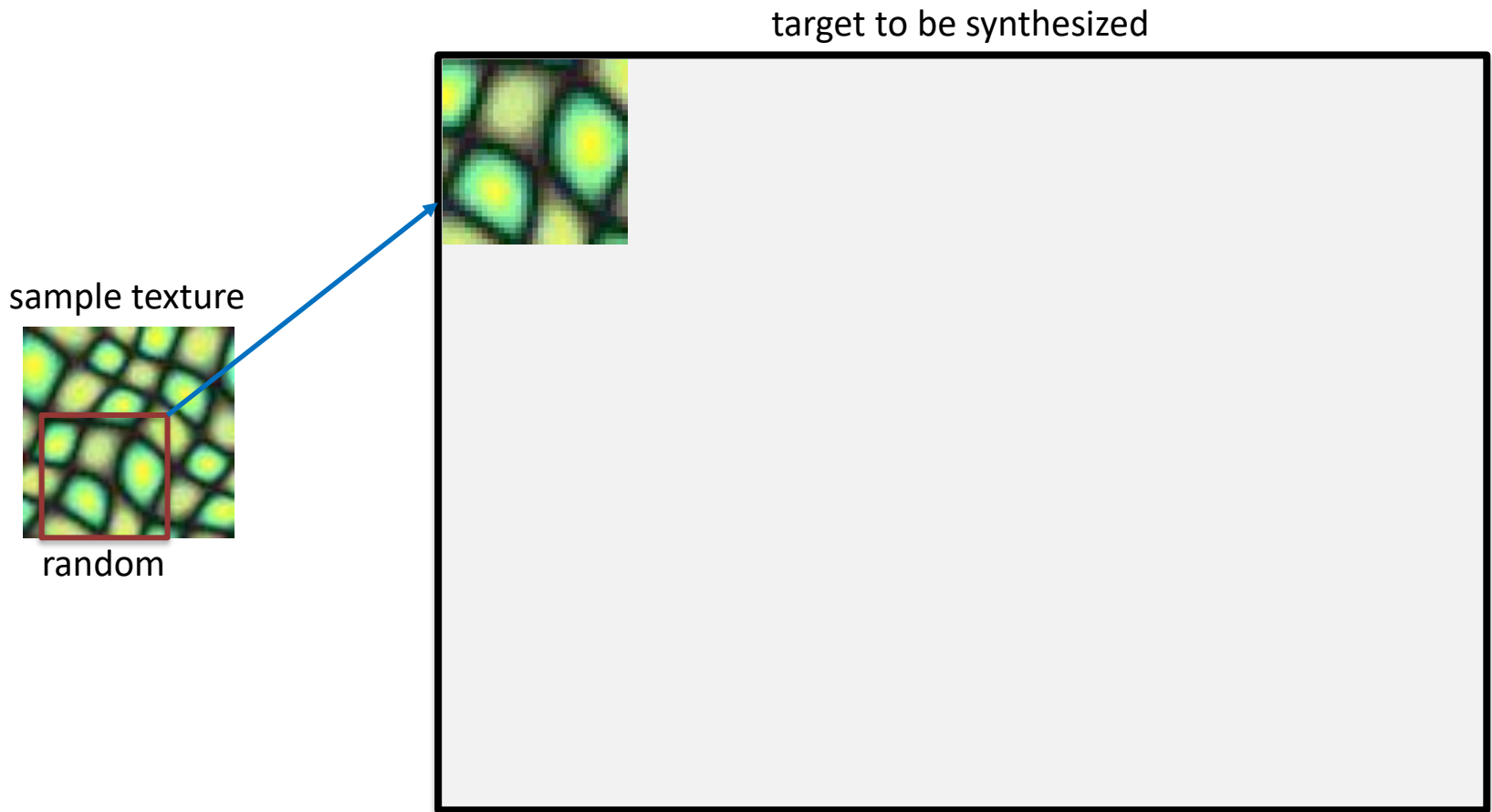


sample texture

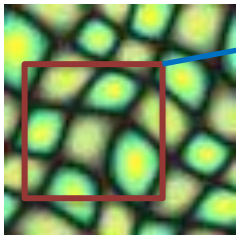


target to be synthesized

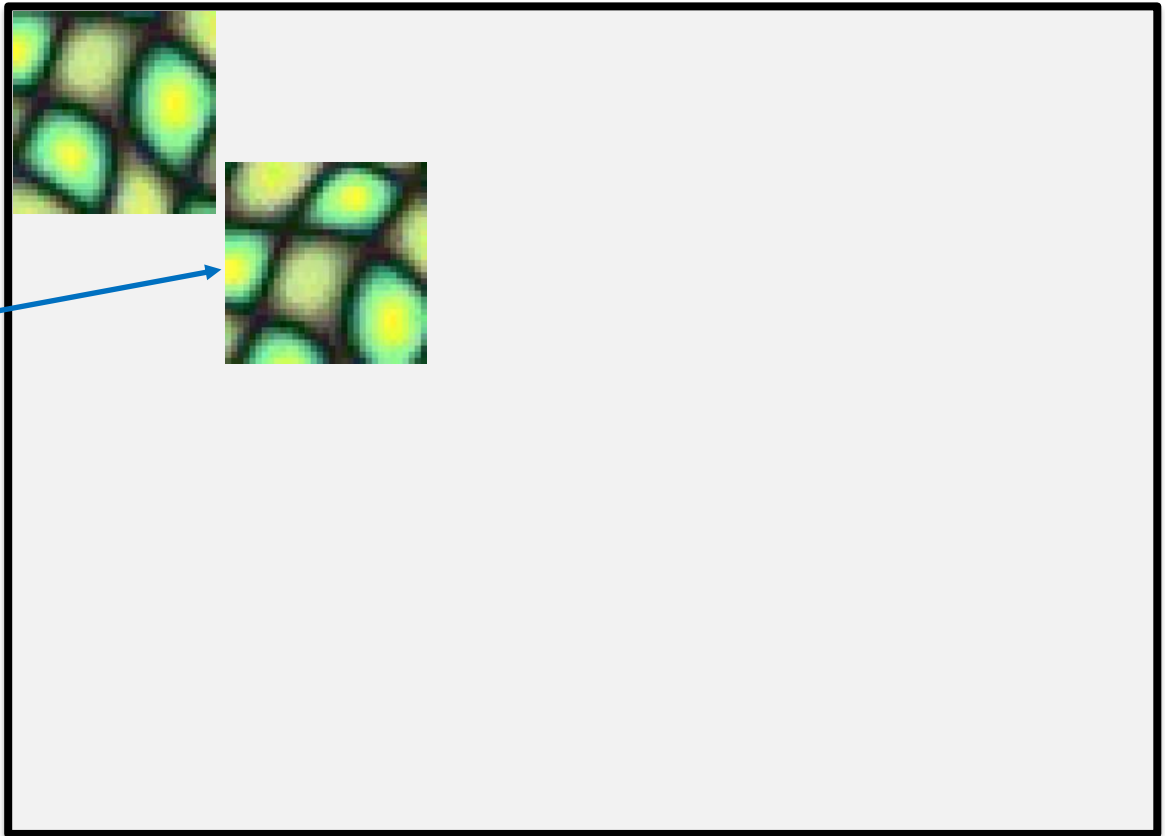




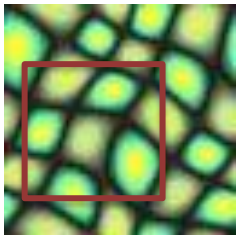
sample texture



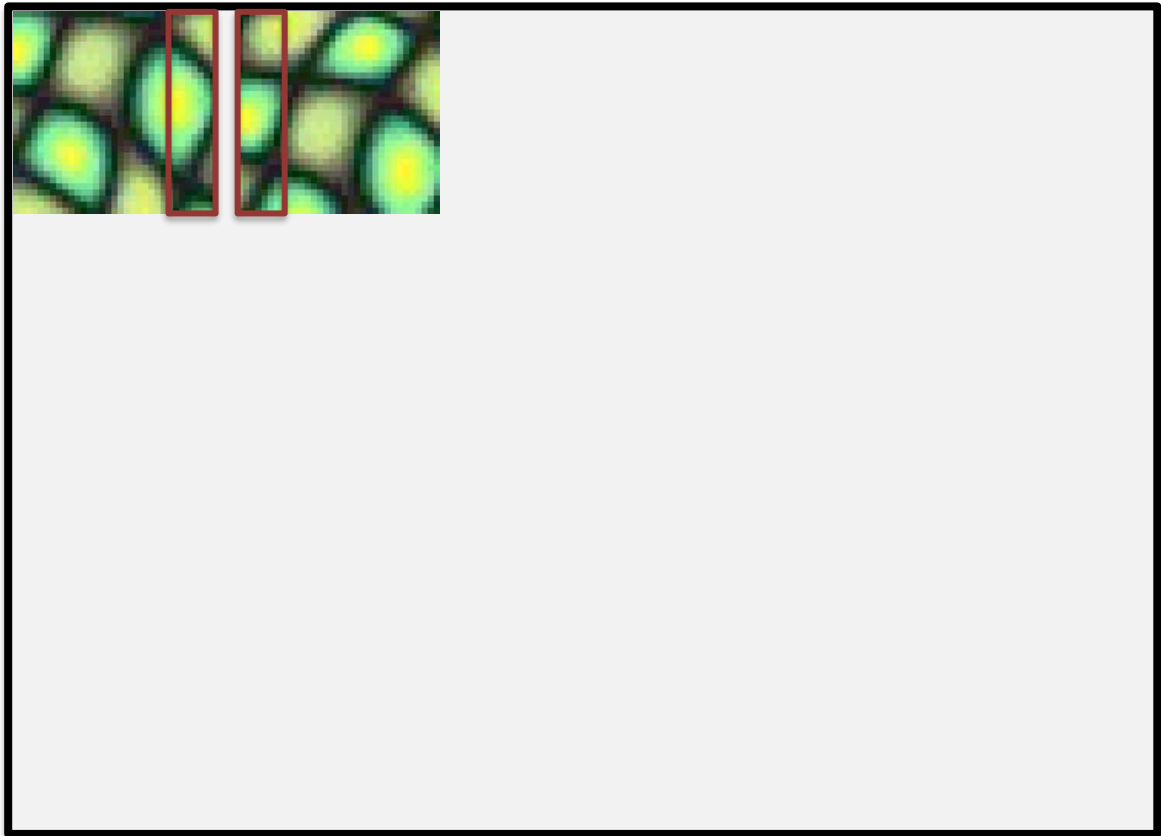
target to be synthesized



sample texture

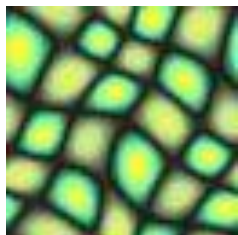


target to be synthesized

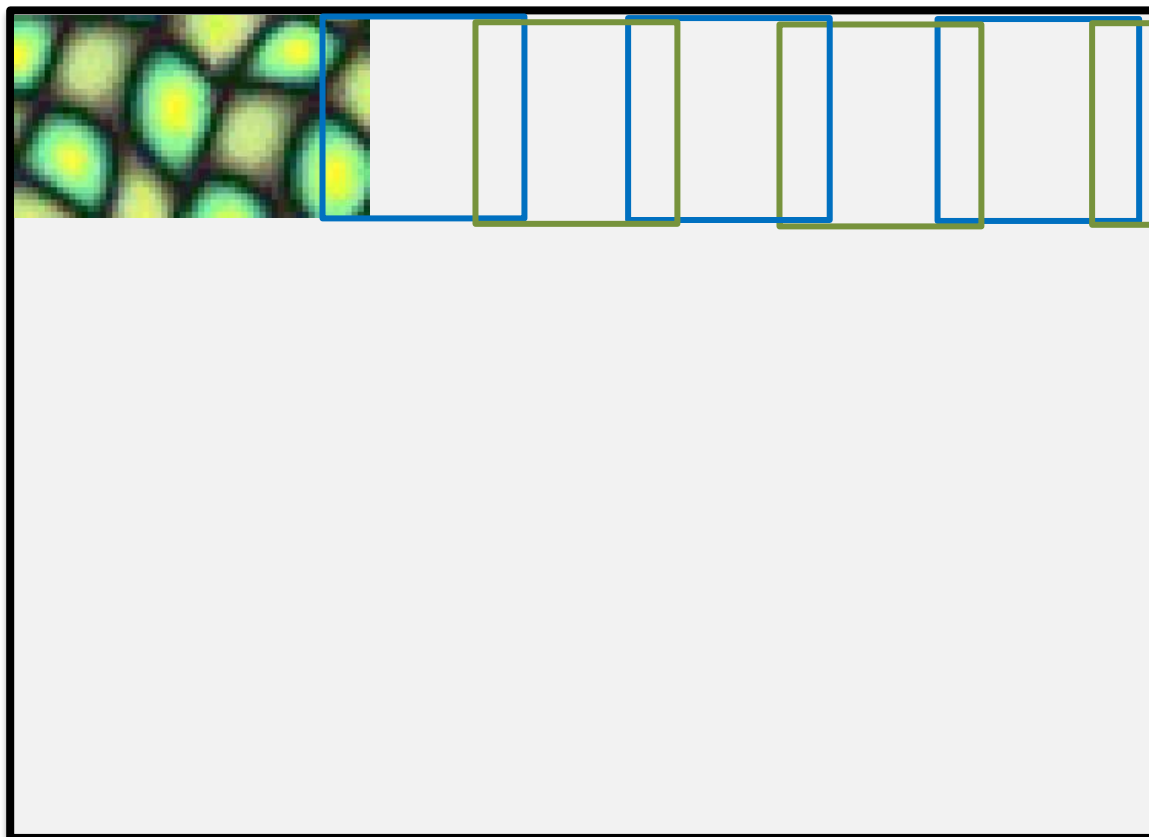




sample texture

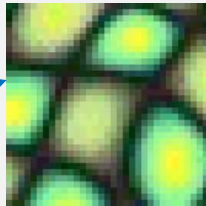
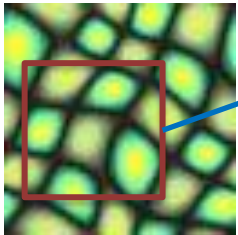


target to be synthesized

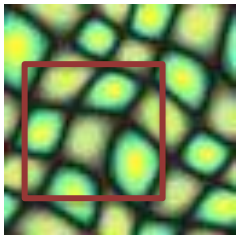


target to be synthesized

sample texture



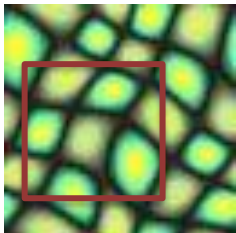
sample texture



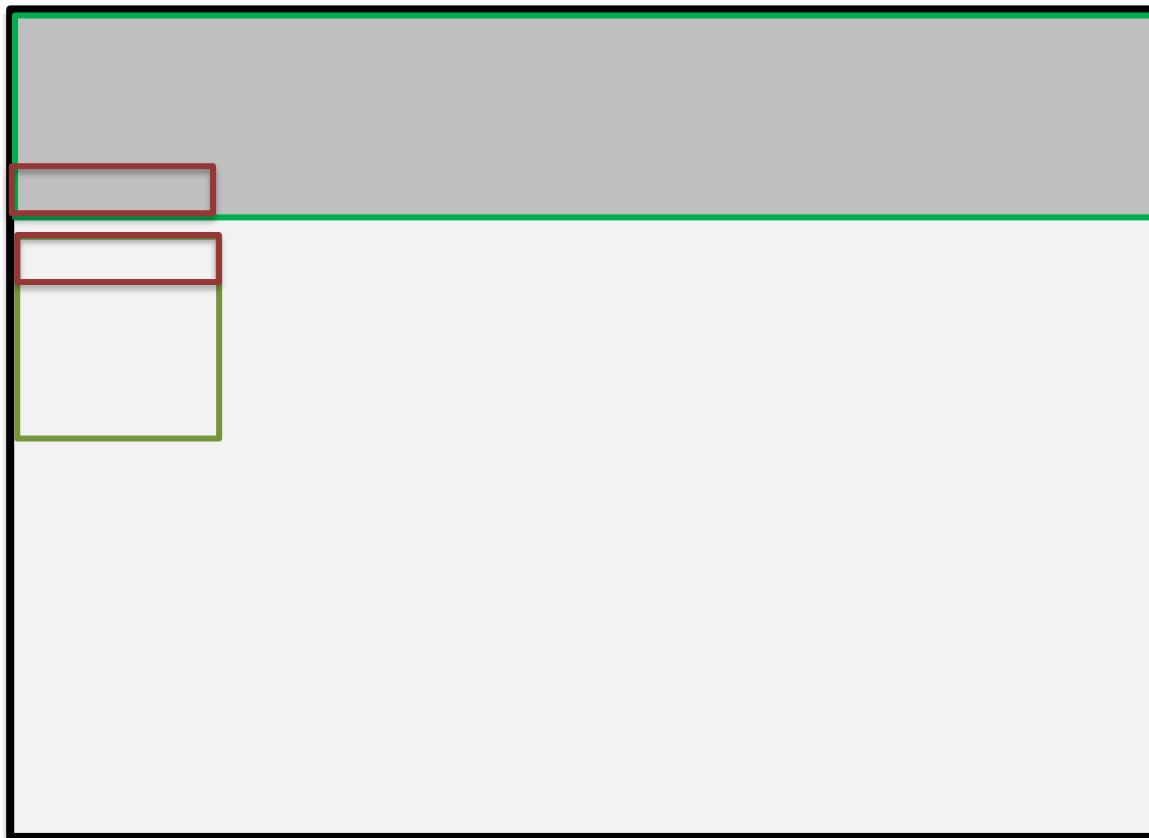
target to be synthesized



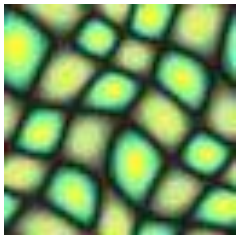
sample texture



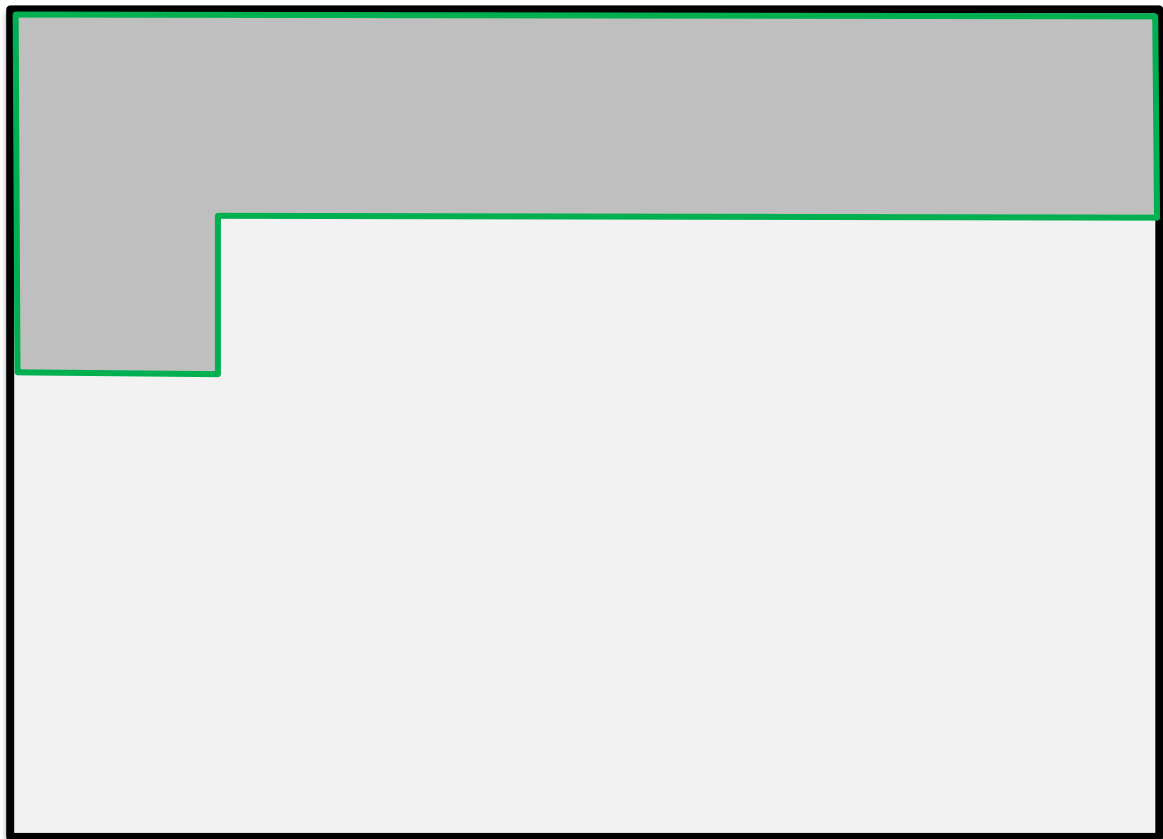
target to be synthesized



sample texture

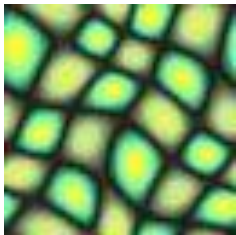


target to be synthesized

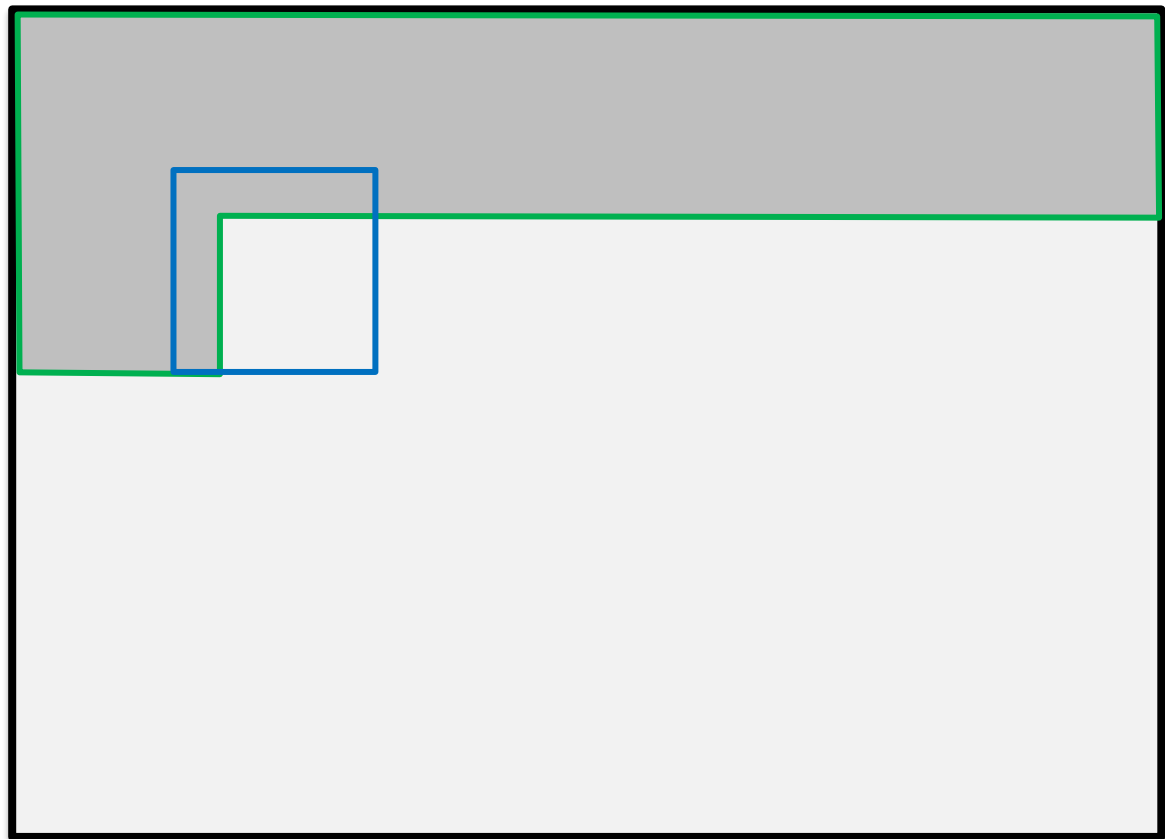




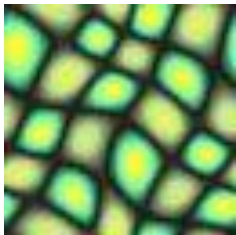
sample texture



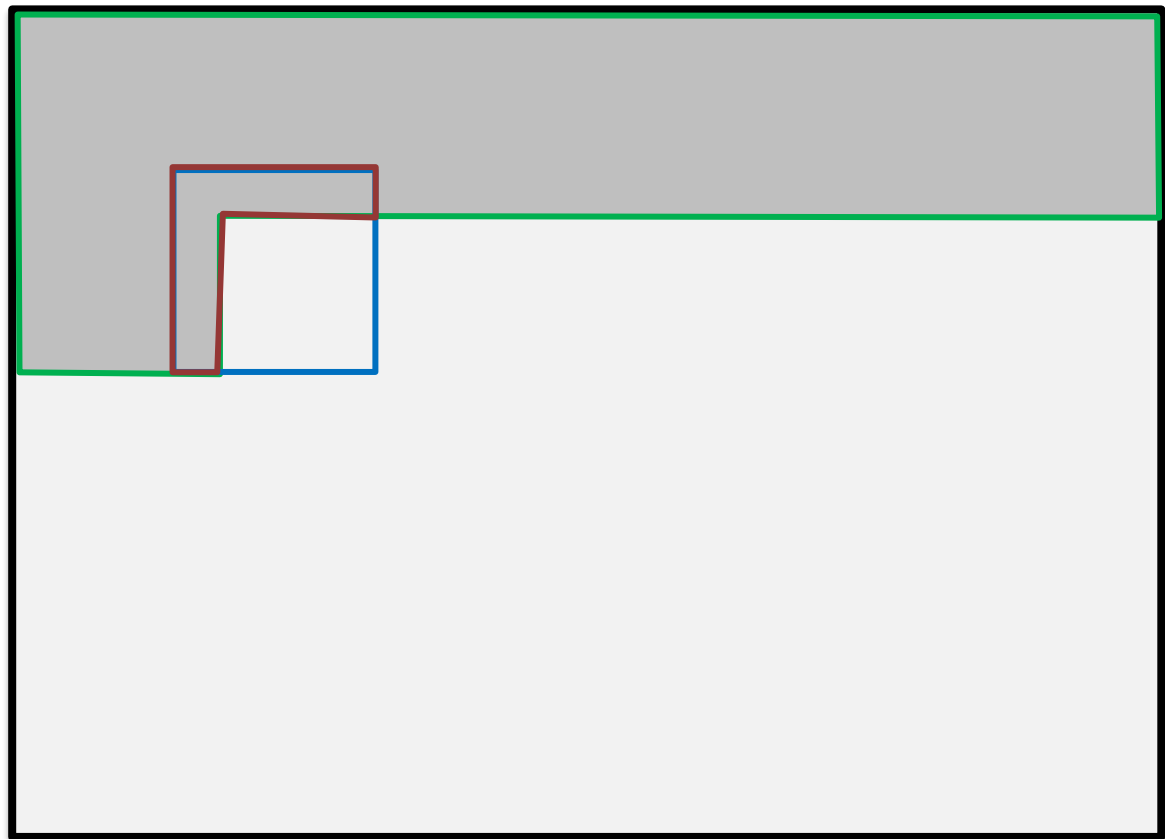
target to be synthesized



sample texture

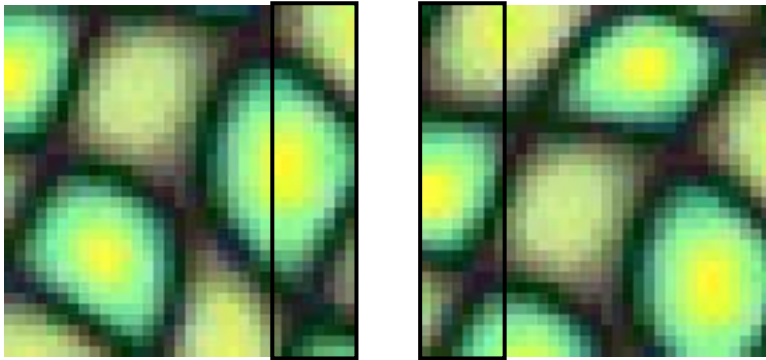


target to be synthesized

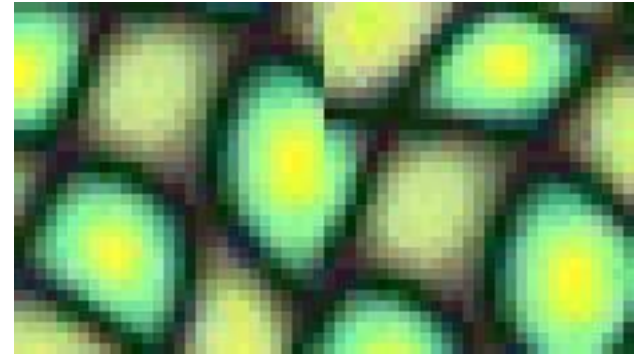


# Minimal Error Boundary

overlapping blocks

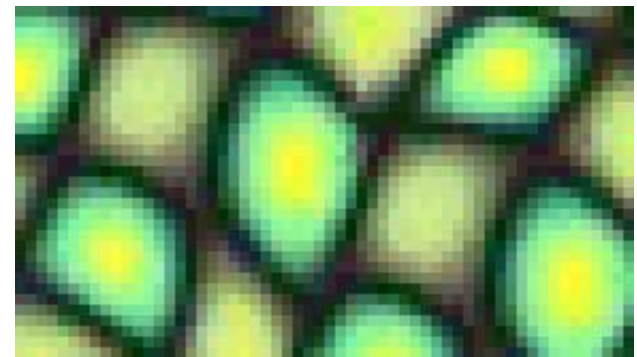


vertical boundary



A diagram illustrating the calculation of overlap error. It shows two vertical blocks of the cell image, one shifted to the left of the other. A large right square bracket groups the two blocks, with a minus sign between them. To the right of the bracket is a superscript '2'. This is followed by an equals sign and a vertical strip of the image showing the overlap area, with a red jagged line indicating the boundary.

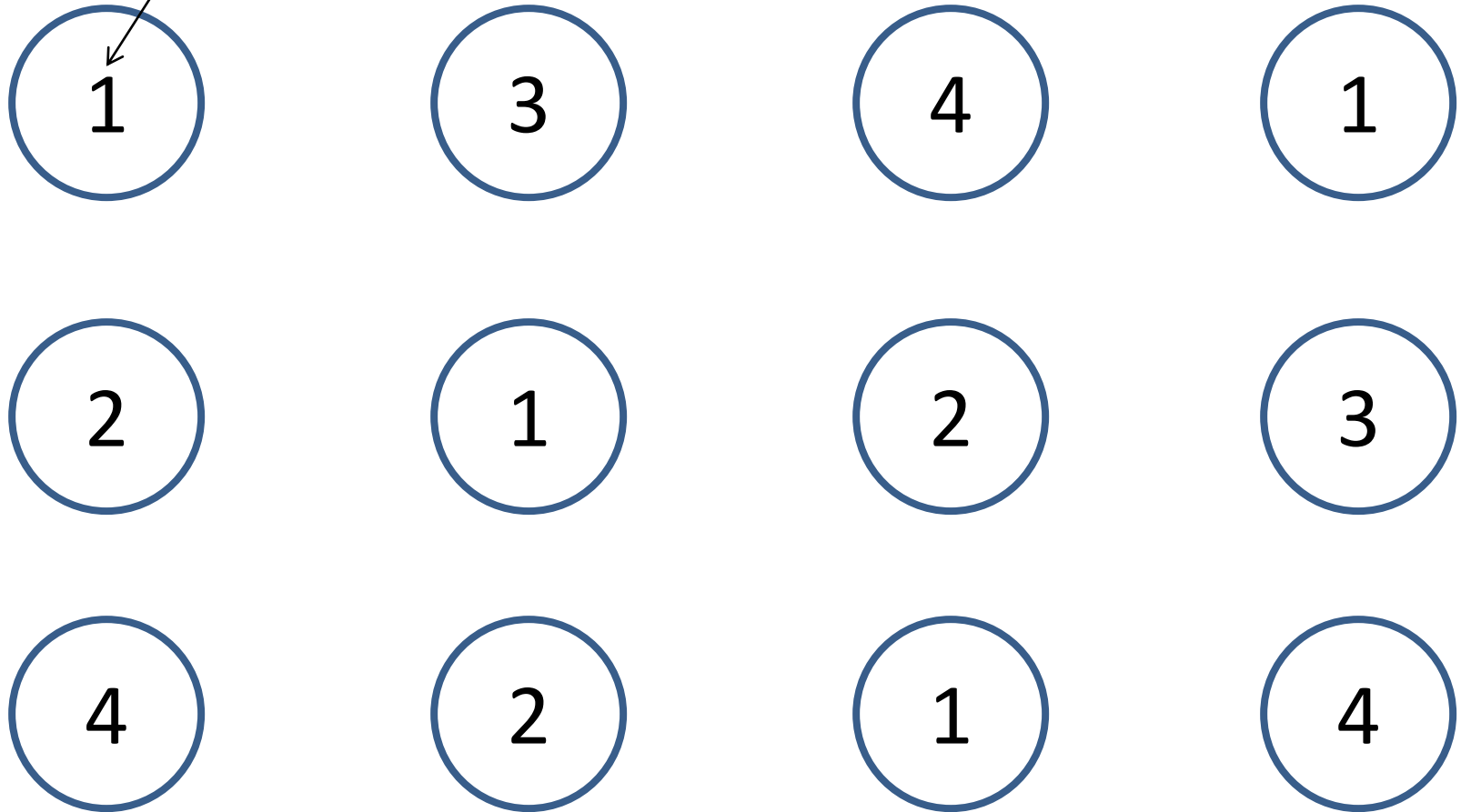
overlap error



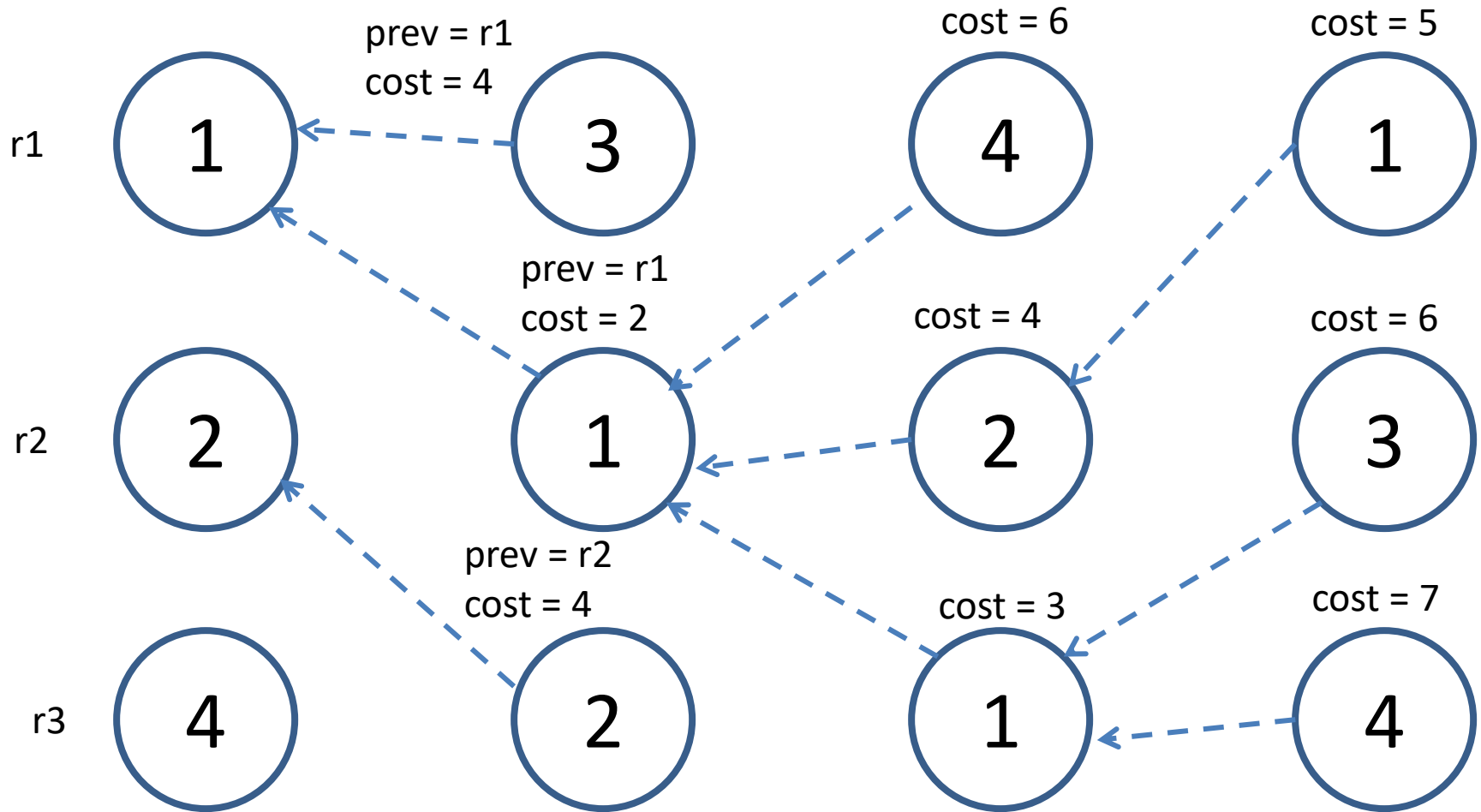
min. error boundary

# Solving for Minimum Cut Path

Cost of a cut through this pixel

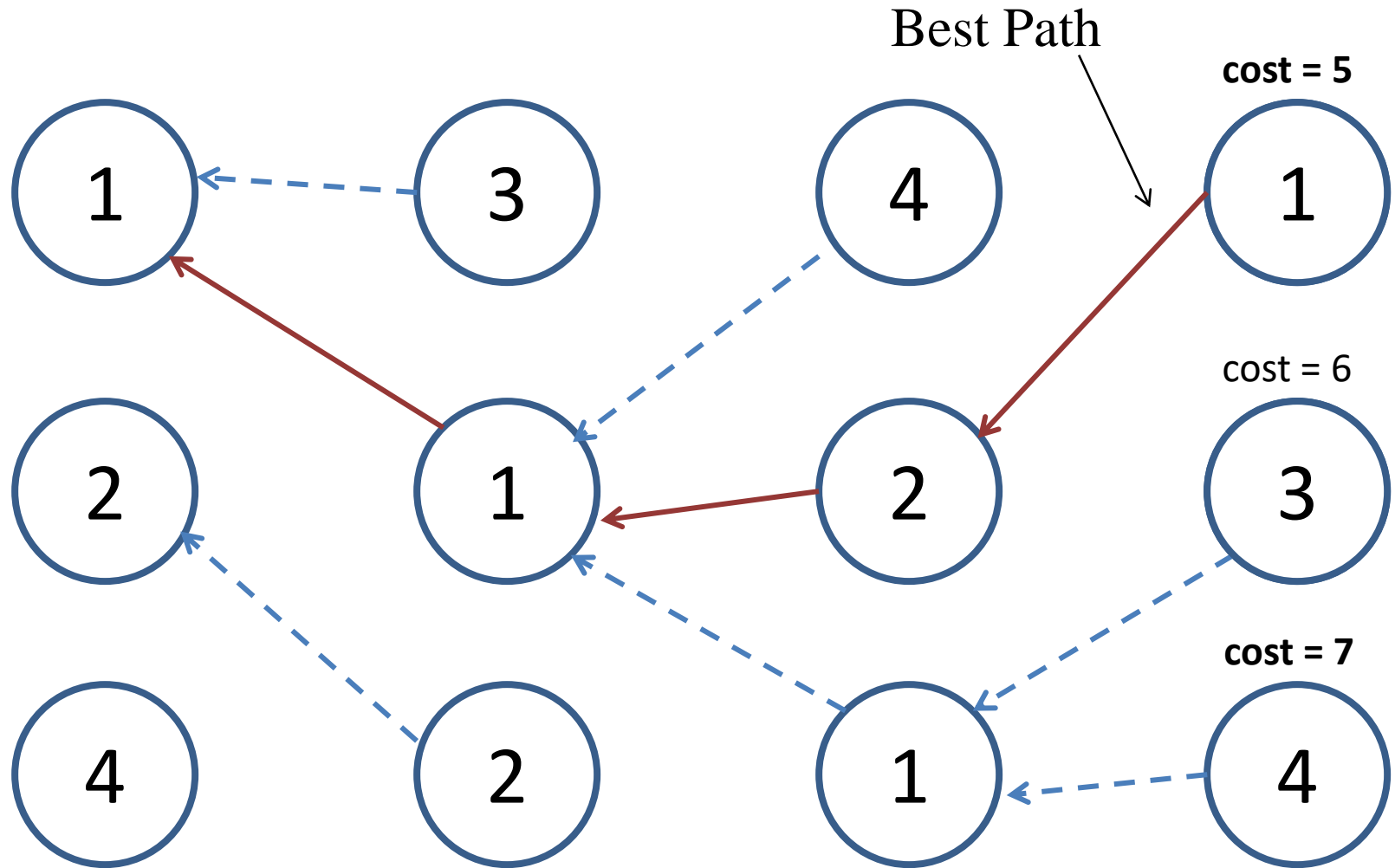


# Solving for Minimum Cut Path

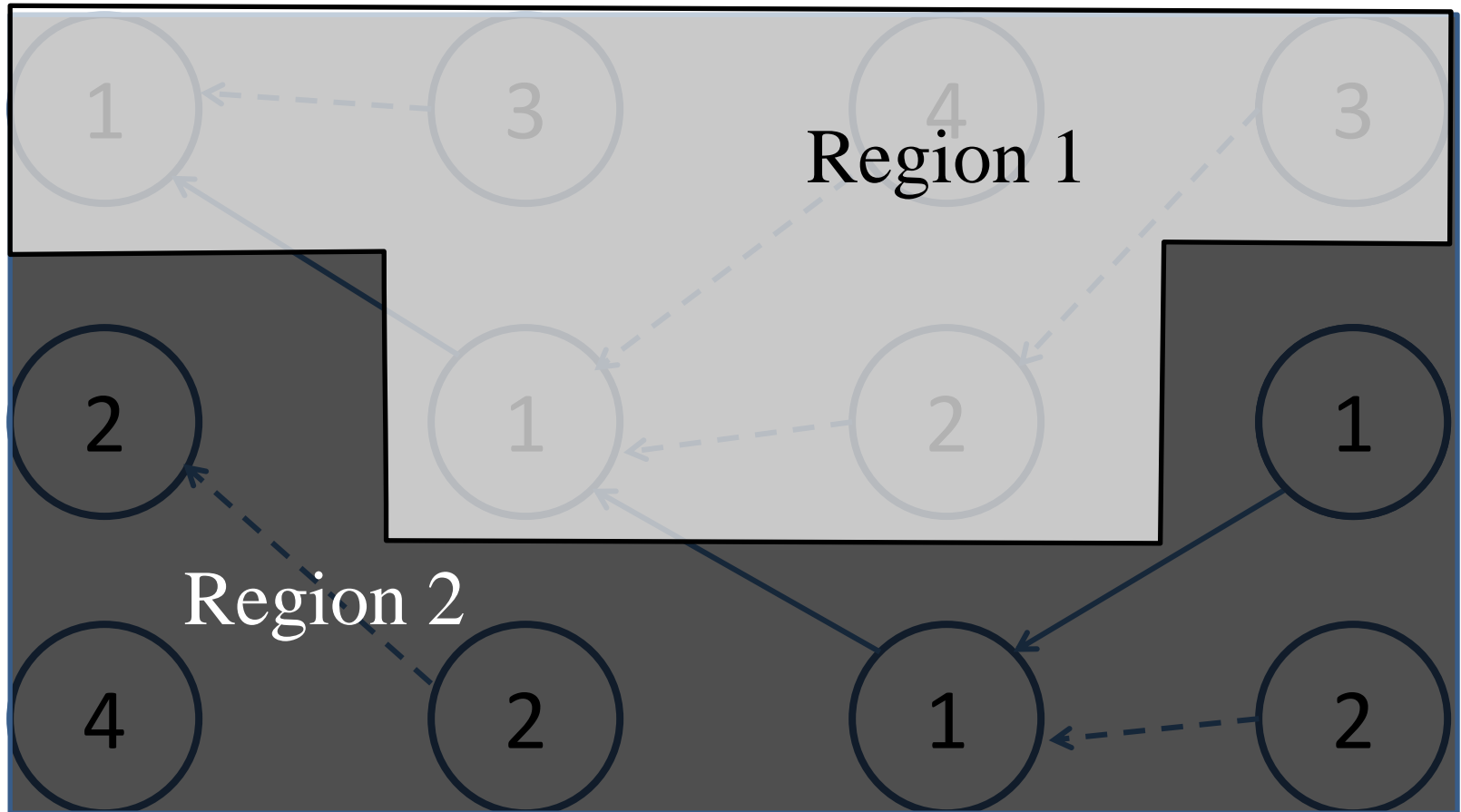




# Solving for Minimum Cut Path

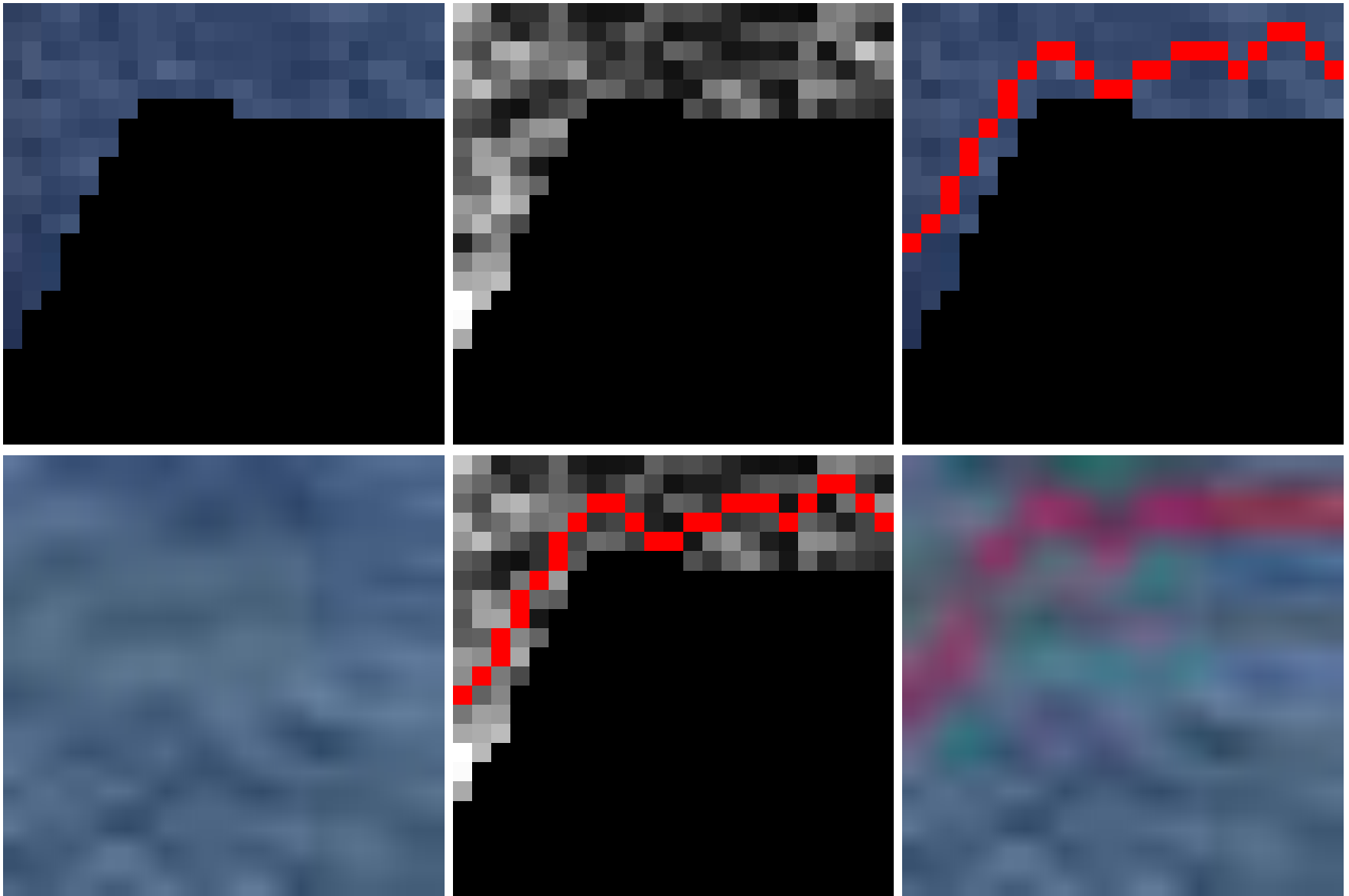


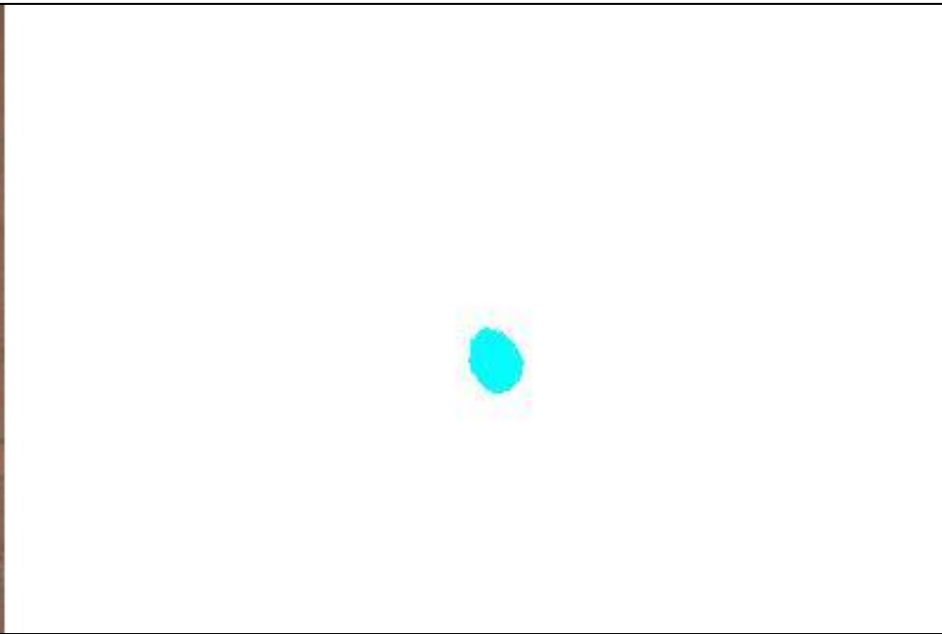
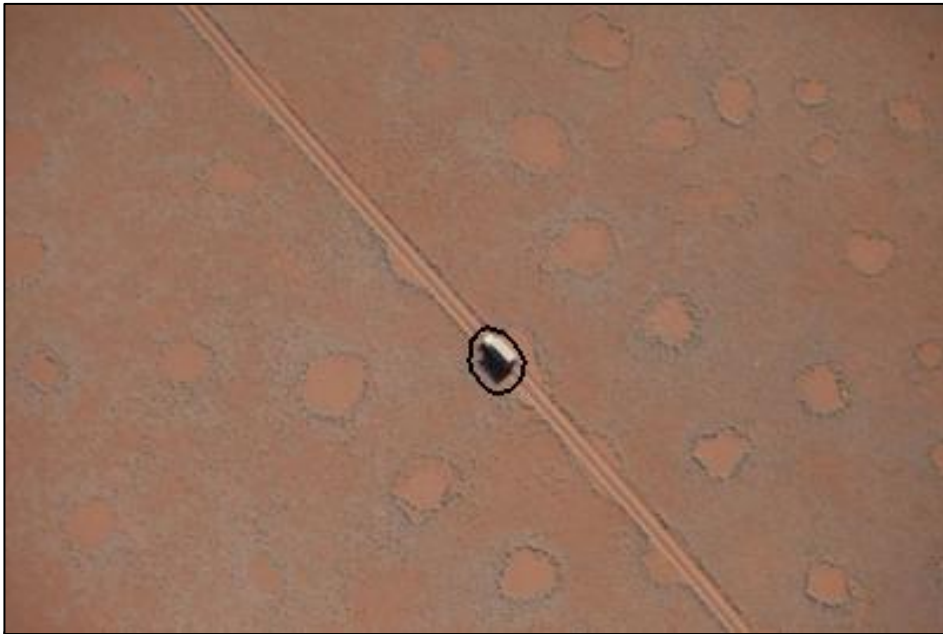
# Solving for Minimum Cut Path

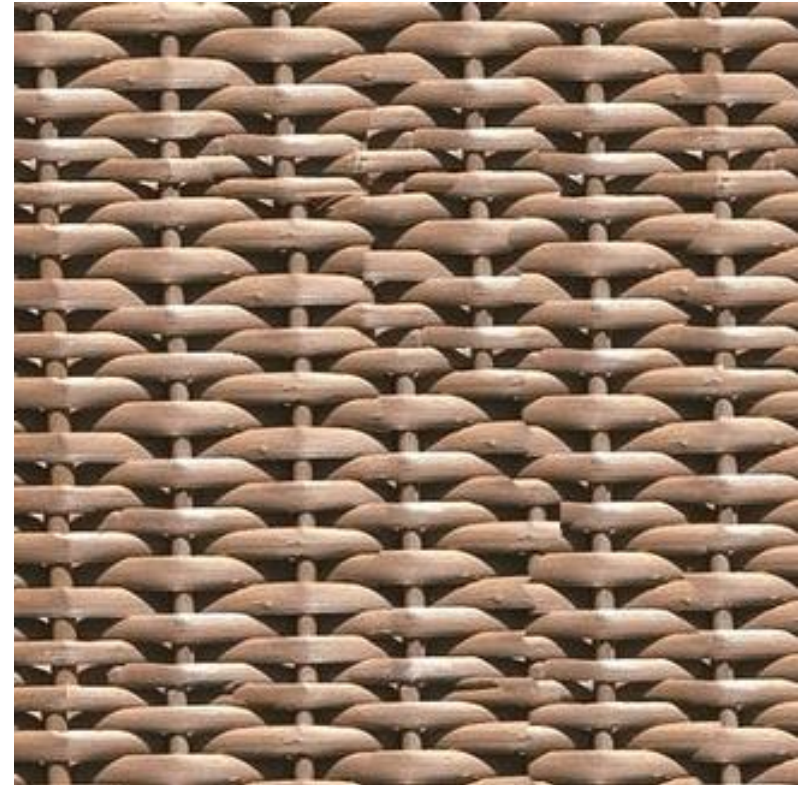
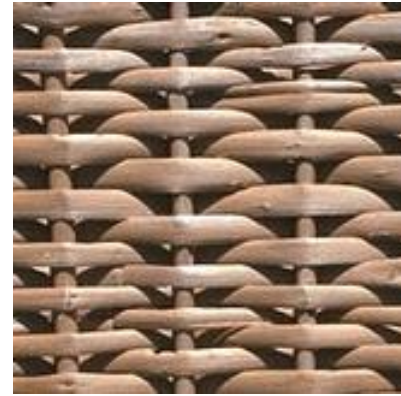


Mask Based on Best Path

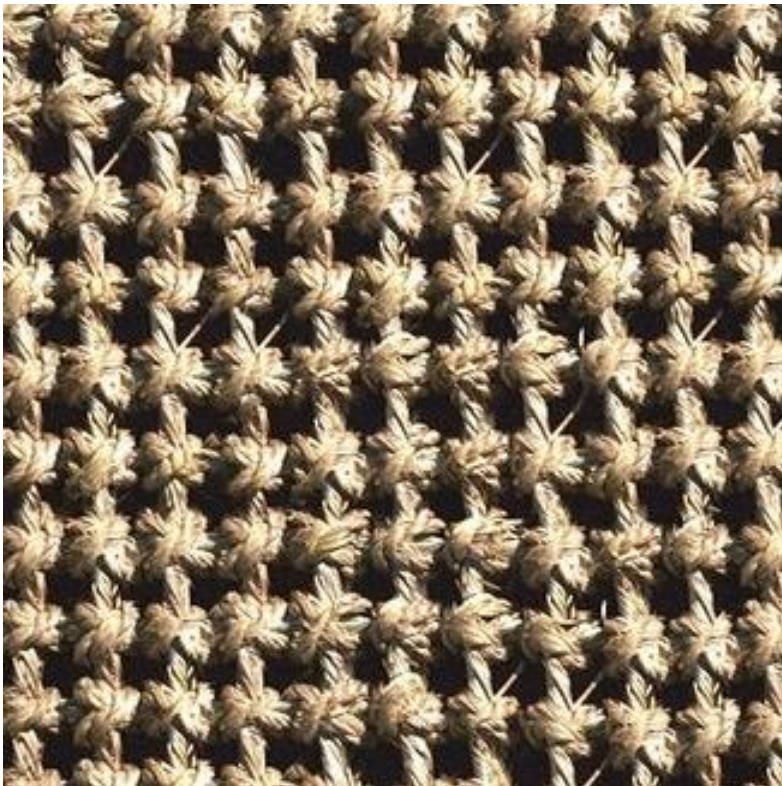
# Image Inpainting



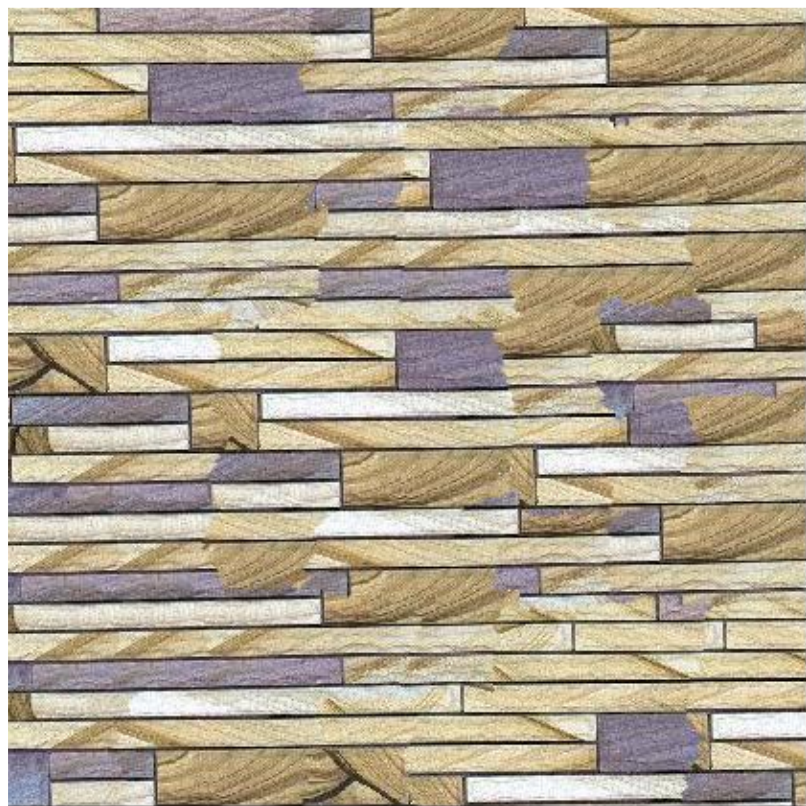










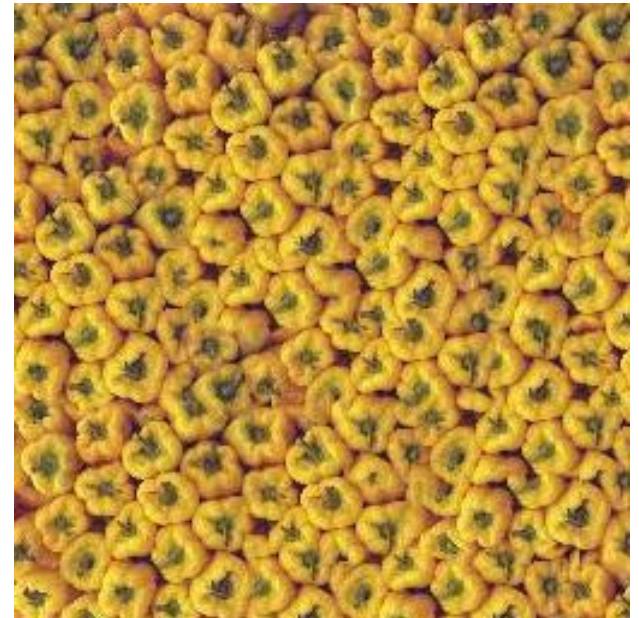
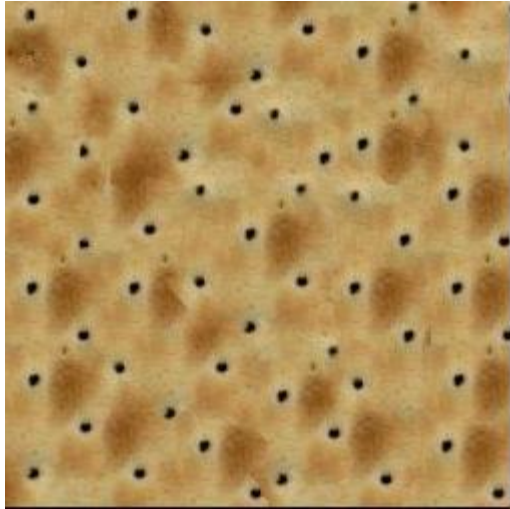
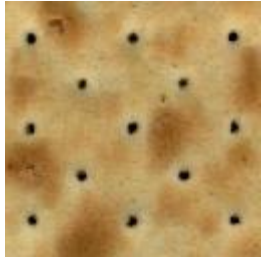




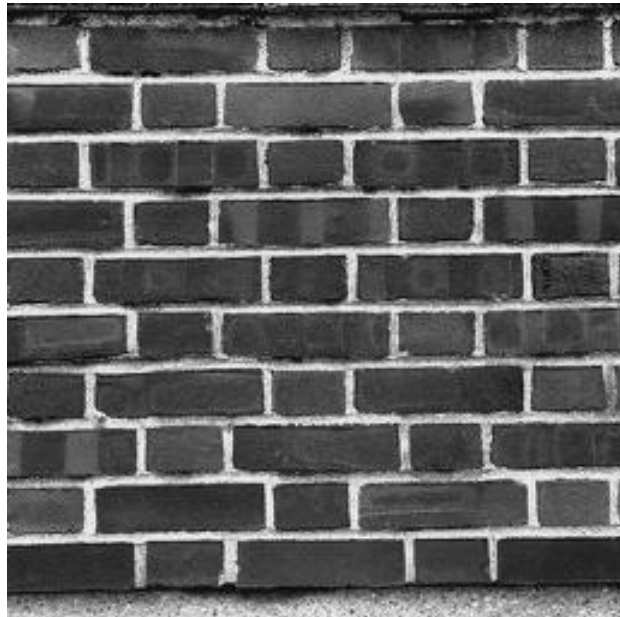








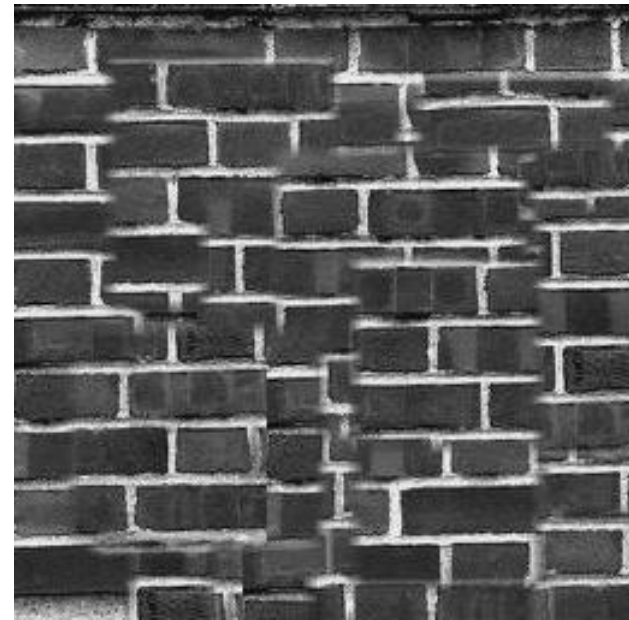




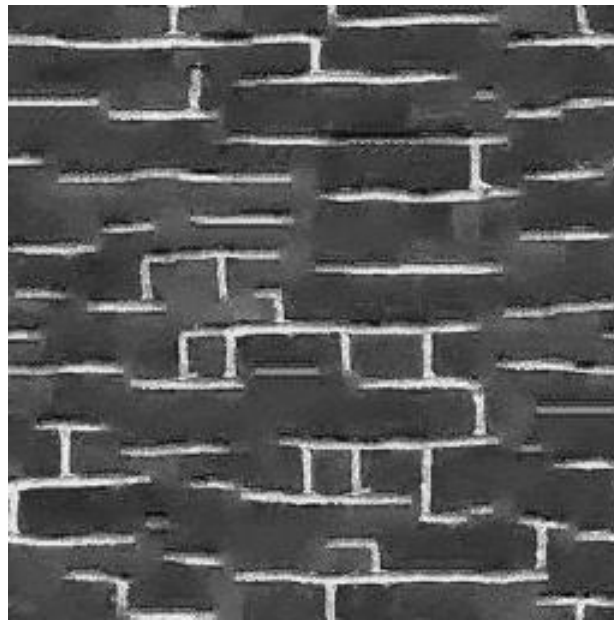
**Input Image**



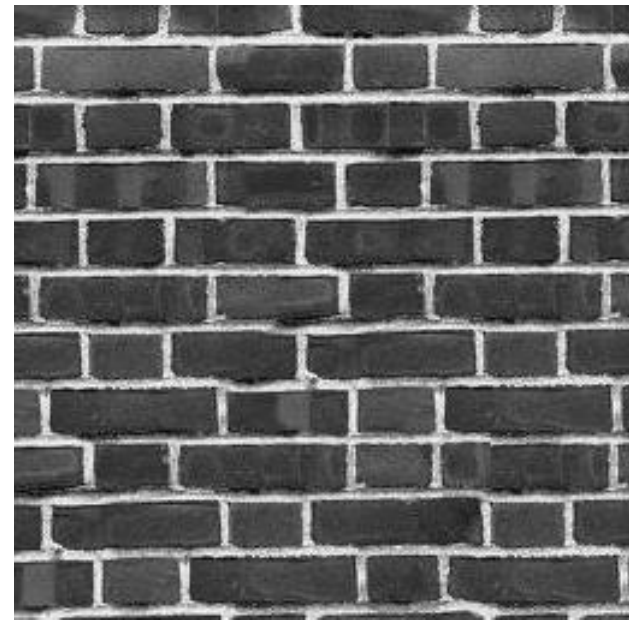
**Portilla & Simoncelli**



**Xu, Guo & Shum**



**Wei & Levoy**



**Quilting**

... of a visual cortical neuron—the in-  
 describing the response of that neuro-  
 ht as a function of position—is perhap  
 functional description of that neuron.  
 seek a single conceptual and mathem.  
 describe the wealth of simple-cell recep-  
 and neurophysiologically<sup>1-3</sup> and inferred  
 especially if such a framework has the  
 it helps us to understand the functio-  
 deeper way. Whereas no generic mo-  
 ussians (DOG), difference of offset C-  
 rivative of a Gaussian, higher derivati-  
 function, and so on—can be expecte-  
 simple-cell receptive field, we noneth-

**Input Image**

... of a visual cortical neuron—the in-  
 describing the response of that neuro-  
 ht as a function of position—is perhap  
 functional description of that neuron.  
 seek a single conceptual and mathem.  
 describe the wealth of simple-cell recep-  
 and neurophysiologically<sup>1-3</sup> and inferred  
 especially if such a framework has the  
 it helps us to understand the functio-  
 deeper way. Whereas no generic mo-  
 ussians (DOG), difference of offset C-  
 rivative of a Gaussian, higher derivati-  
 function, and so on—can be expecte-  
 simple-cell receptive field, we noneth-

**Portilla & Simoncelli**

... of a visual cortical neuron—the in-  
 describing the response of that neuro-  
 ht as a function of position—is perhap  
 functional description of that neuron.  
 seek a single conceptual and mathem.  
 describe the wealth of simple-cell recep-  
 and neurophysiologically<sup>1-3</sup> and inferred  
 especially if such a framework has the  
 it helps us to understand the functio-  
 deeper way. Whereas no generic mo-  
 ussians (DOG), difference of offset C-  
 rivative of a Gaussian, higher derivati-  
 function, and so on—can be expecte-  
 simple-cell receptive field, we noneth-

**Wei & Levoy**

... of a visual cortical neuron—the in-  
 describing the response of that neuro-  
 ht as a function of position—is perhap  
 functional description of that neuron.  
 seek a single conceptual and mathem.  
 describe the wealth of simple-cell recep-  
 and neurophysiologically<sup>1-3</sup> and inferred  
 especially if such a framework has the  
 it helps us to understand the functio-  
 deeper way. Whereas no generic mo-  
 ussians (DOG), difference of offset C-  
 rivative of a Gaussian, higher derivati-  
 function, and so on—can be expecte-  
 simple-cell receptive field, we noneth-

**Xu, Guo & Shum**

... of a visual cortical neuron—the in-  
 describing the response of that neuro-  
 ht as a function of position—is perhap  
 functional description of that neuron.  
 seek a single conceptual and mathem.  
 describe the wealth of simple-cell recep-  
 and neurophysiologically<sup>1-3</sup> and inferred  
 especially if such a framework has the  
 it helps us to understand the functio-  
 deeper way. Whereas no generic mo-  
 ussians (DOG), difference of offset C-  
 rivative of a Gaussian, higher derivati-  
 function, and so on—can be expecte-  
 simple-cell receptive field, we noneth-

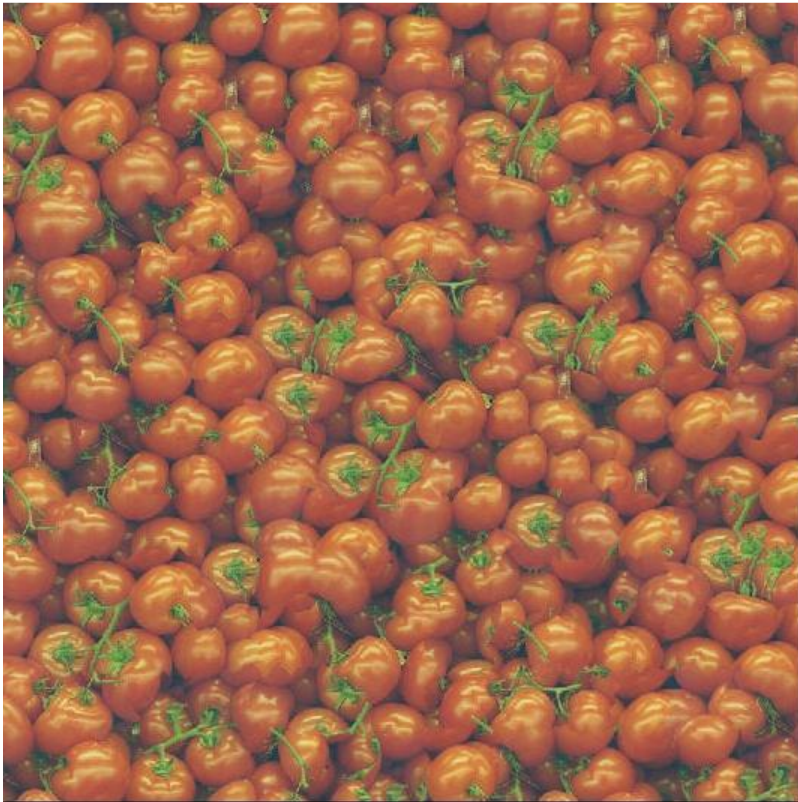
**Quilting**





# ***Failures***

**(Chernobyl Harvest)**



# In-Painting Natural Scenes



Criminisi, Perez, and Toyama. “[Object Removal by Exemplar-based Inpainting](#),” CVPR 2003

# Key Idea: Filling Order Matters

## In-painting Result



Image with Hole



Raster-Scan Order



Onion-Peel  
(Concentric Layers)



Gradient-Sensitive  
Order

Criminisi, Perez, and Toyama. “[Object Removal by Exemplar-based Inpainting](#),” CVPR 2003



# Filling Order

Fill a pixel that:

1. Is surrounded by other known pixels
2. Is a continuation of a strong gradient or edge



Criminisi, Perez, and Toyama. “[Object Removal by Exemplar-based Inpainting](#),” CVPR 2003



Original



With Hole



Onion-Ring Fill

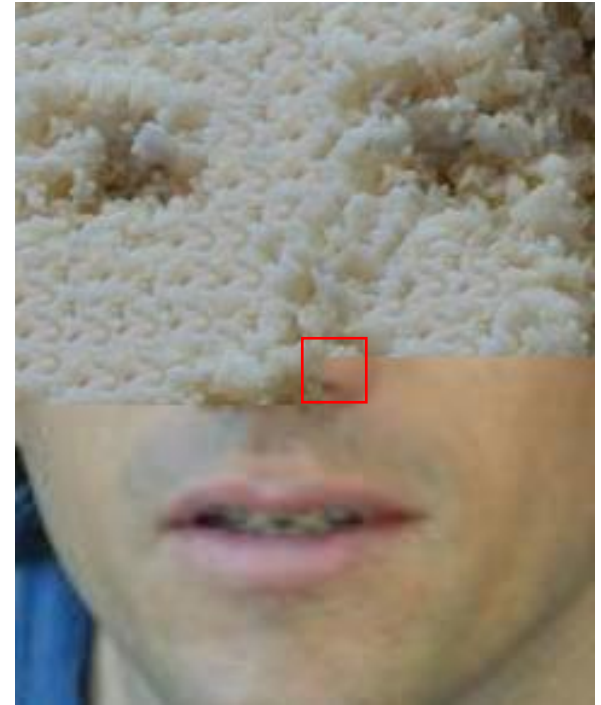


Criminisi

Criminisi, Perez, and Toyama. “[Object Removal by Exemplar-based Inpainting](#),” CVPR 2003

# Texture Transfer

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

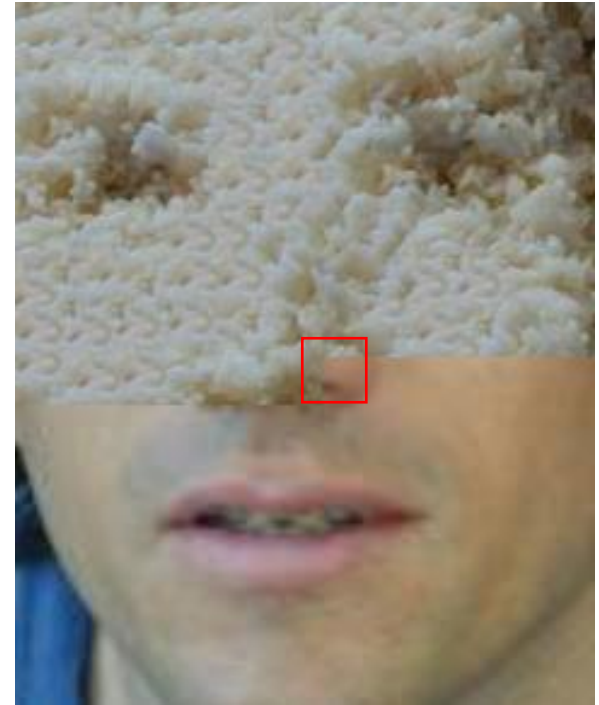


Same as texture synthesis, except an additional constraint:

1. Consistency of texture
2. Patches from texture should correspond to patches from constraint in some way. Typical example: blur luminance, use SSD for distance

# Texture Transfer

- Take the texture from one object and “paint” it onto another object
  - This requires separating texture and shape
  - That’s HARD, but we can cheat
  - Assume we can capture shape by boundary and rough shading



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



parmesan



+



=



rice

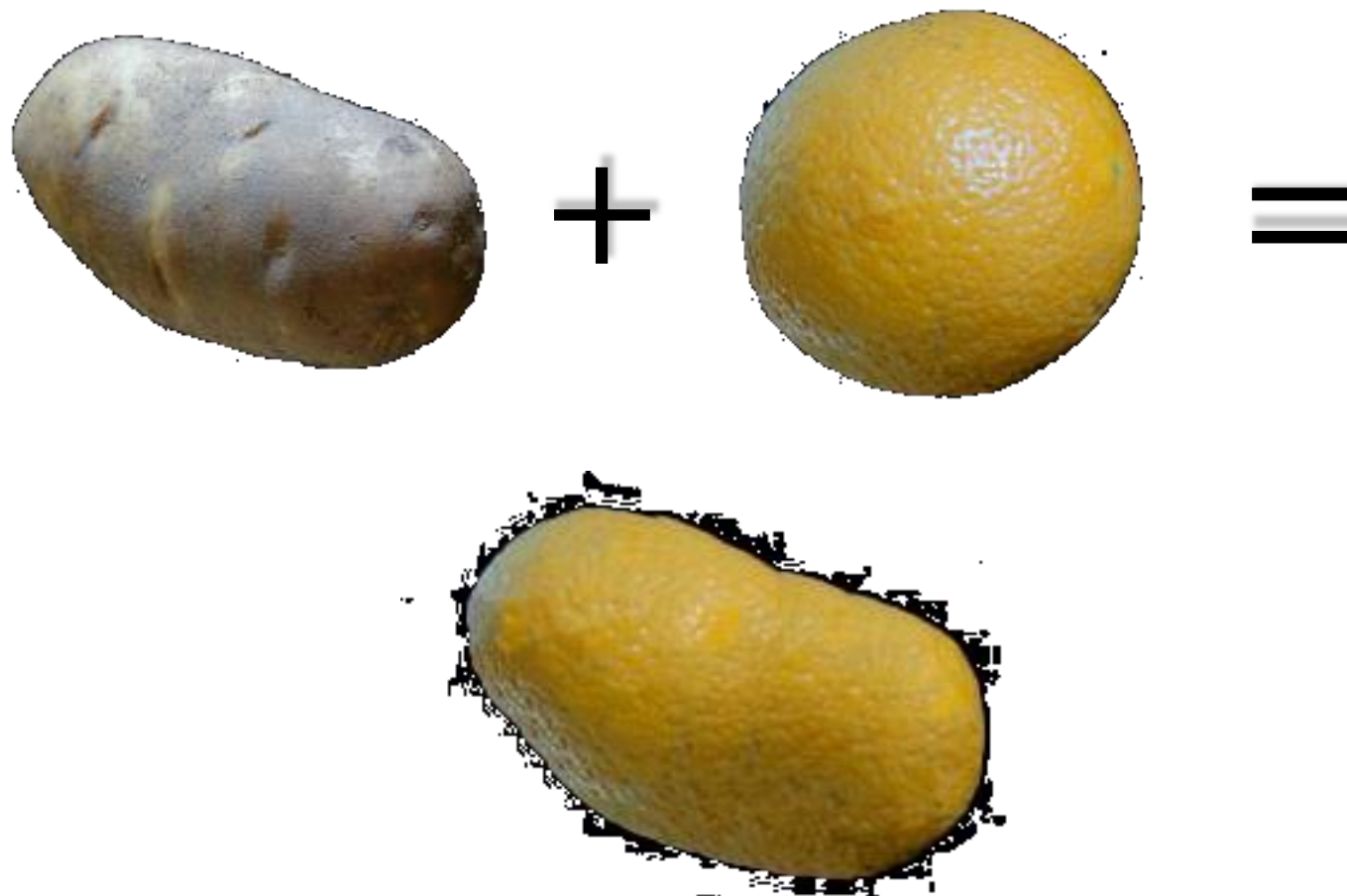


+



=





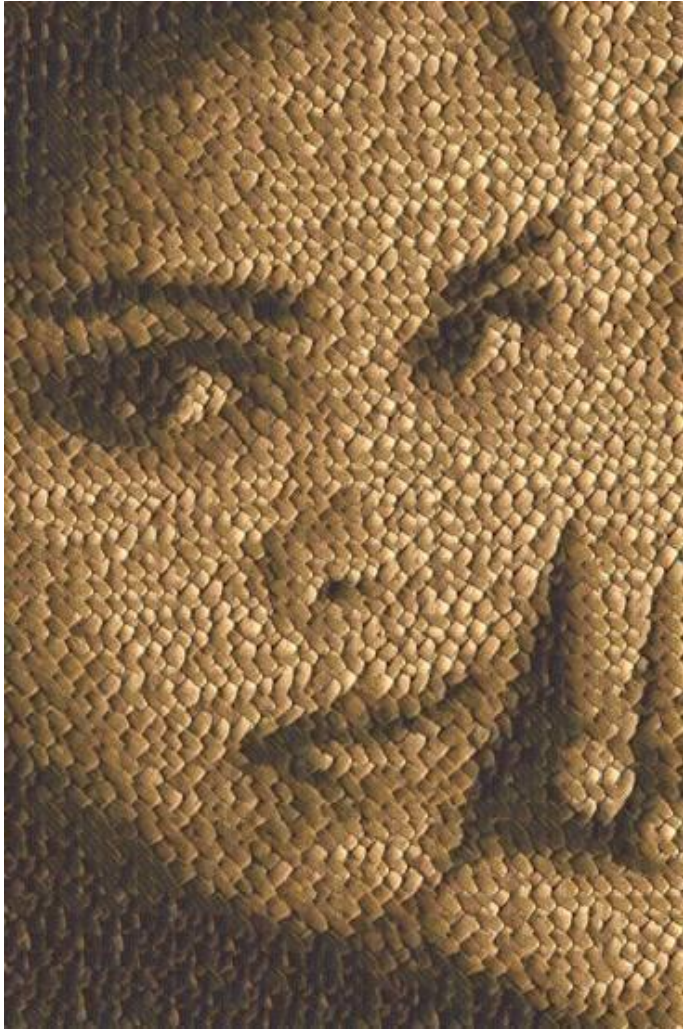
# Texture Synthesis and Transfer Recap



For each overlapping patch in the output image

1. Compute the cost to each patch in the sample
  - Texture synthesis: this cost is the SSD (sum of square difference) of pixel values in the overlapping portion of the existing output and sample
  - Texture transfer: cost is  $\alpha SSD_{overlap} + (1 - \alpha)SSD_{transfer}$ . The latter term enforces that **at the source and target correspondence patches should match**.
2. Select one sample patch that has a small cost
3. Find a cut through the left/top borders of the patch based on overlapping region with existing output
  - Use this cut to create a mask that specifies which pixels to copy from sample patch
4. Copy masked pixels from sample image to corresponding pixel locations in output image



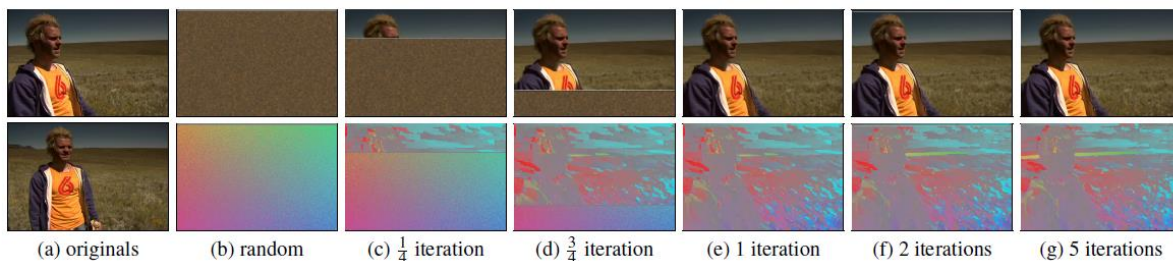
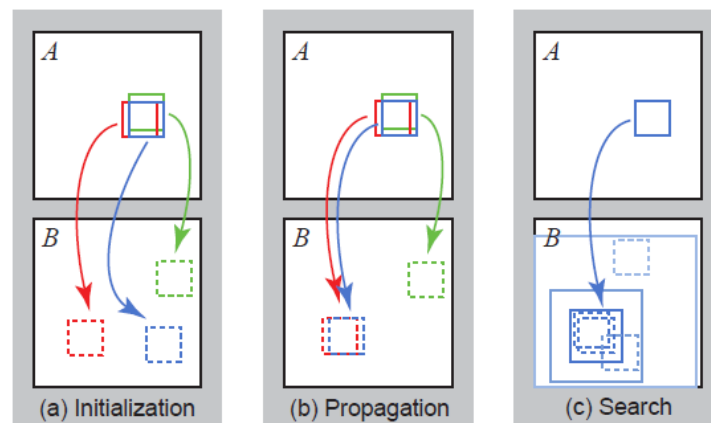


Recommended!

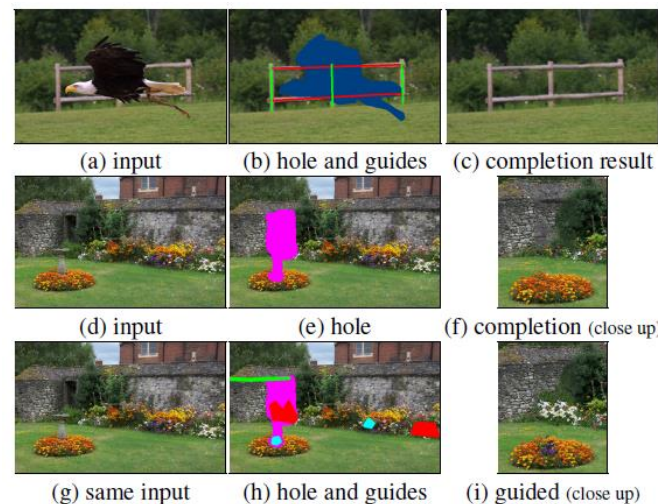
# PatchMatch

More efficient search:

1. Randomly initialize matches
2. See if neighbor's offsets are better
3. Randomly search a local window for better matches
4. Repeat 2, 3 across image several times



Reconstructing top-left image with patches from bottom-left image



Barnes et al. Patch Match, SIGGRAPH 2009

Barnes et al. Generalized Patch Match, ECCV 2010

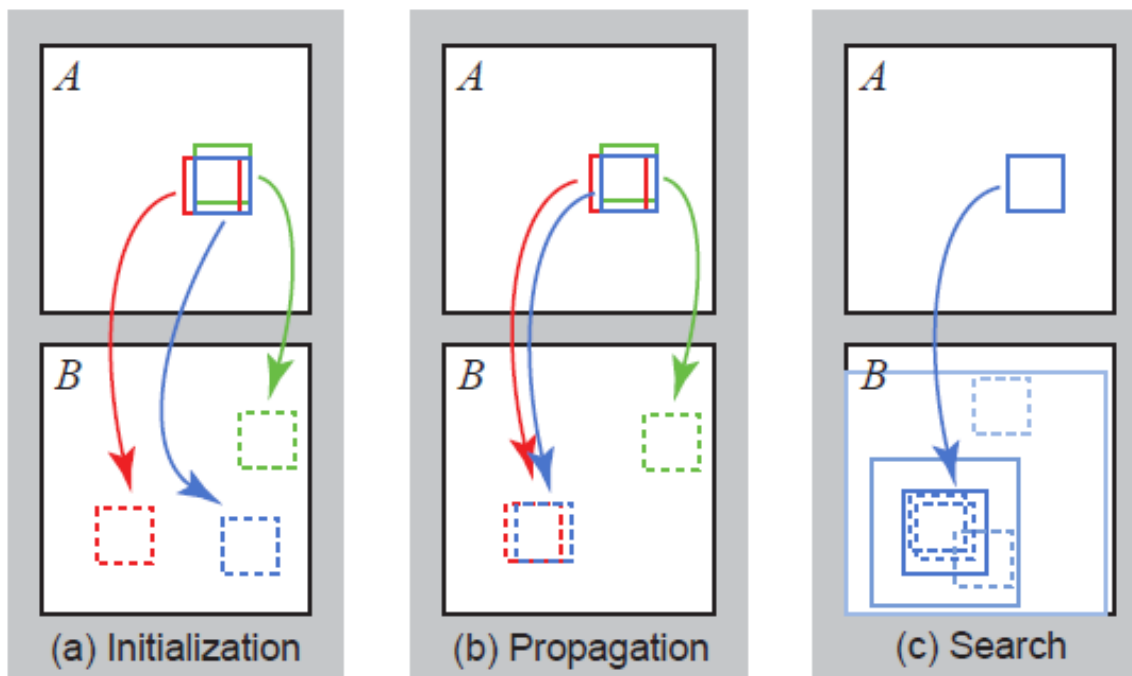
Ehret & Arias, On the convergence of PatchMatch and its variants, CVPR 2018

Applications to hole-filling, retargeting;  
constraints can guide search

# PatchMatch

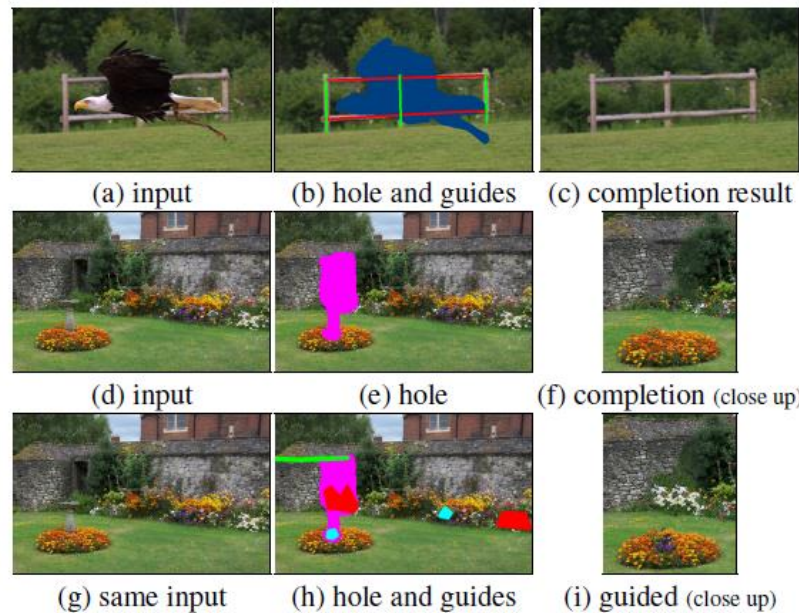
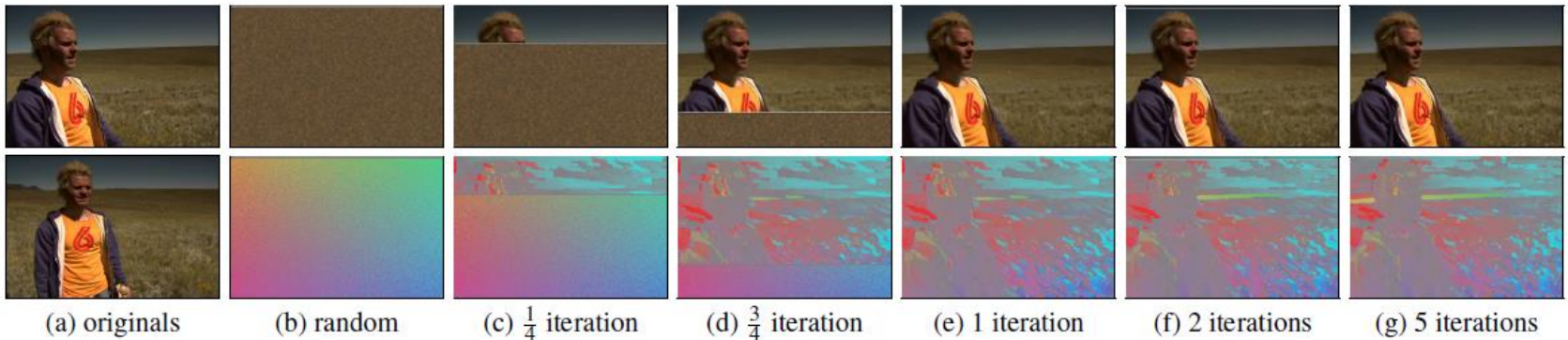
More efficient search:

1. Randomly initialize matches
2. See if neighbor's offsets are better
3. Randomly search a local window for better matches
4. Repeat 2, 3 across image several times





# PatchMatch





# Related idea: Image Analogies



*A*

∴



*A'*



*B*

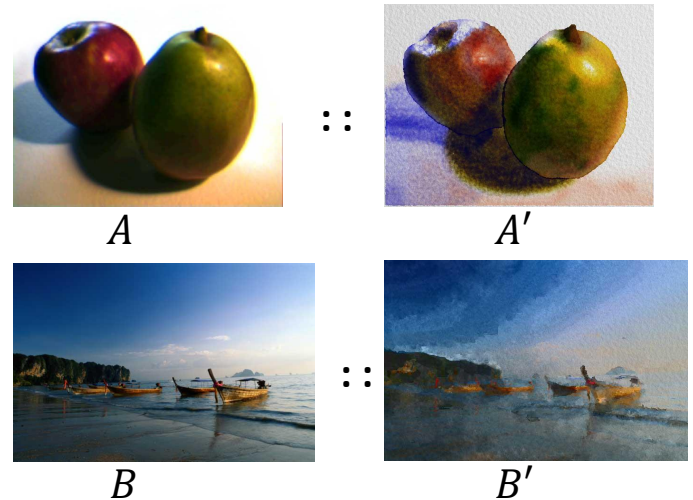
∴



*B'*



# Image Analogies



- Define a similarity between  $A$  and  $B$
- For each patch in  $B$  :
  - Find a matching patch in  $A$ , whose corresponding  $A'$  also fits in well with existing patches in  $B'$
  - Copy the patch in  $A'$  to  $B'$
- Algorithm is done iteratively, coarse-to-fine



# Blur Filter



**Unfiltered source ( $A$ )**



**Filtered source ( $A'$ )**



**Unfiltered target ( $B$ )**

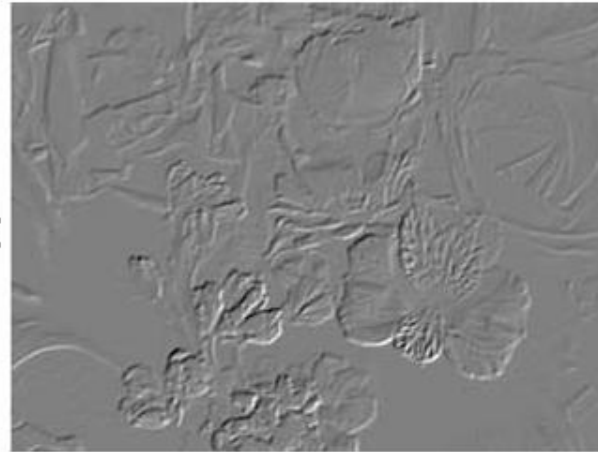


**Filtered target ( $B'$ )**

# Edge Filter



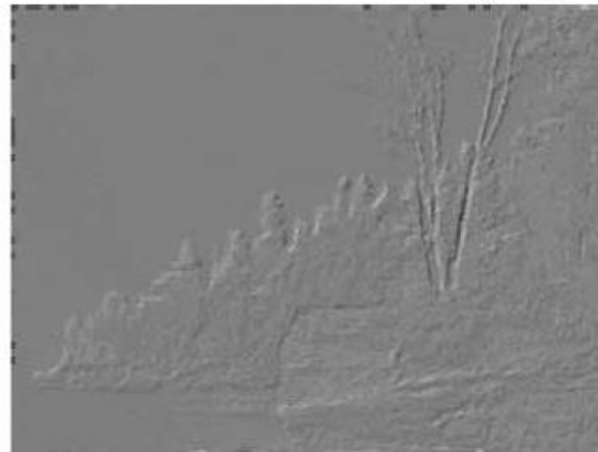
**Unfiltered source ( $A$ )**



**Filtered source ( $A'$ )**



**Unfiltered target ( $B$ )**



**Filtered target ( $B'$ )**



# Artistic Filters



*A*



*A'*



*B*



*B'*



# Colorization



**Unfiltered source ( $A$ )**



**Filtered source ( $A'$ )**



**Unfiltered target ( $B$ )**



**Filtered target ( $B'$ )**

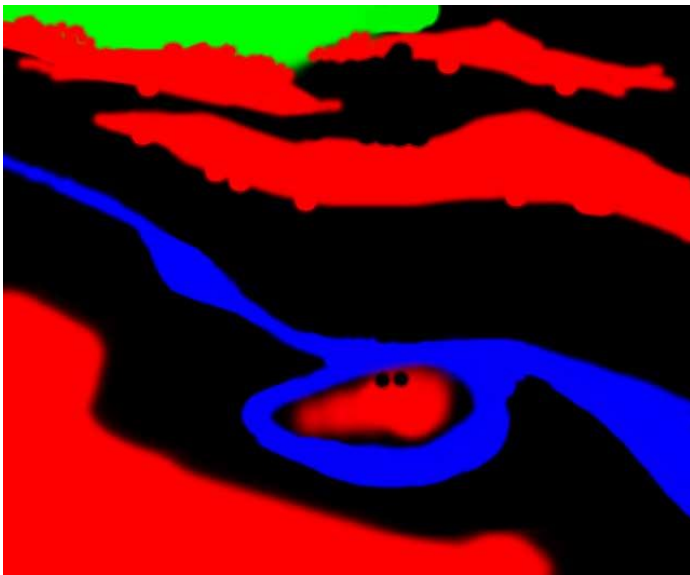
# Texture-by-Numbers



A



A'



B



B'

# Super-Resolution



$A$



$A'$





# Super-Resolution (result!)



$B$



$B'$



# Texture Synthesis



**a**



**b**



*(Image Inpainting & Hole Filling)*

# References

- Texture Analysis and Synthesis  
Szeliski, Section 10.5
- Efros & Leung, *Texture Synthesis by non-parametric sampling*, ICCV 1999
- Efros & Freeman, *Image Quilting for Texture Synthesis and Transfer*, SIGGRAPH 2001
- Criminisi et al, *Object Removal by Exemplar-based Inpainting*, CVPR 2003
- Barnes et al, PatchMatch, SIGGRAPH 2009
- Hertzman et al, Image Analogies, SIGGRAPH 2001