

Texture Synthesis and Hole-Filling



Computational Photography

Derek Hoiem, University of Illinois

Next section: The digital canvas



Cutting and pasting objects,
filling holes, and blending



Image warping and object
morphing

Today's Class

- Texture synthesis and hole-filling



Texture

- Texture depicts spatially repeating patterns
- Textures appear naturally and frequently



radishes



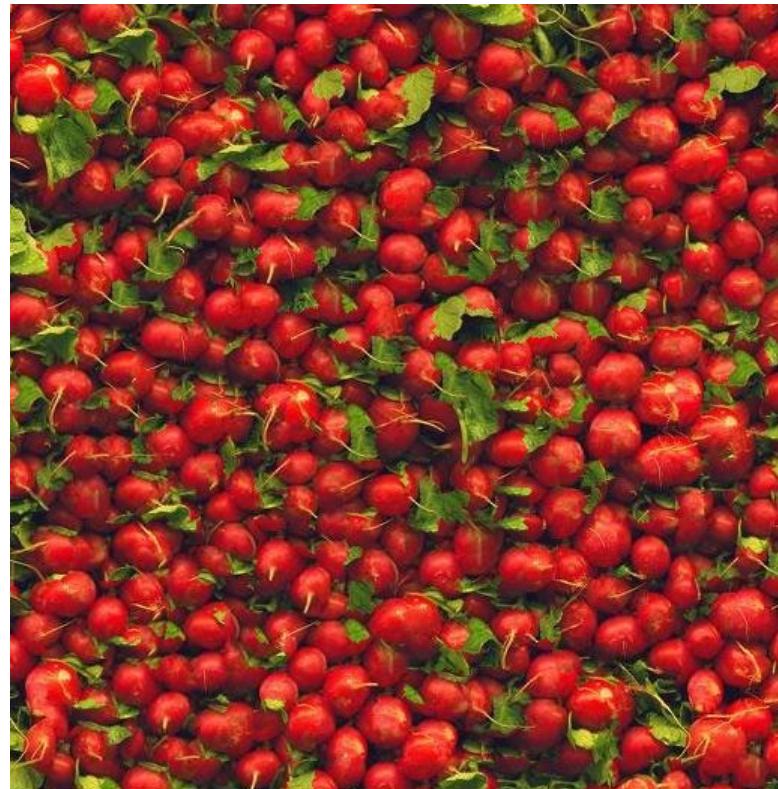
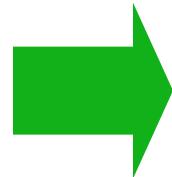
rocks



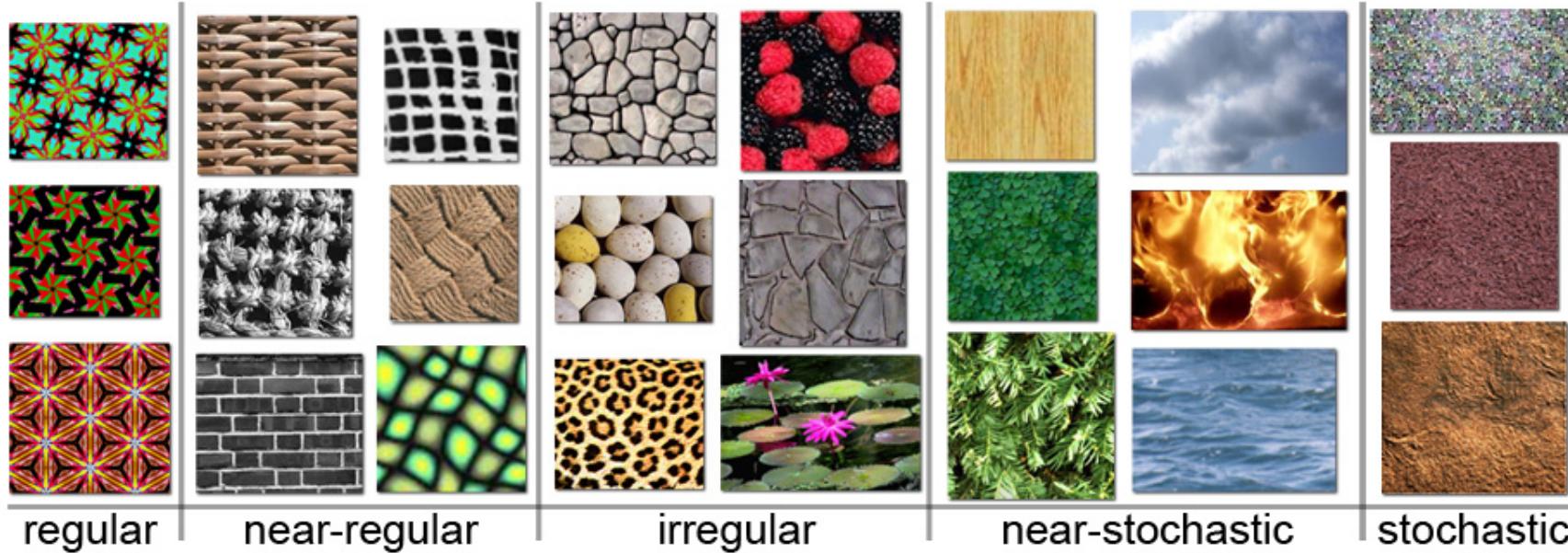
yogurt

Texture Synthesis

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



The Challenge

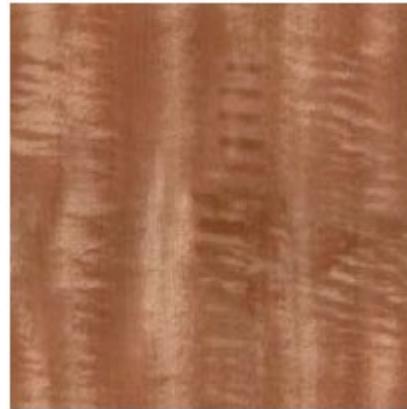


Need to model the whole spectrum: from repeated to stochastic texture

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. Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95*.
- E. P. Simoncelli and J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. In *ICIP 1998*.

One idea: Build Probability Distributions

But it (usually) doesn't work

- Probability distributions are hard to model well

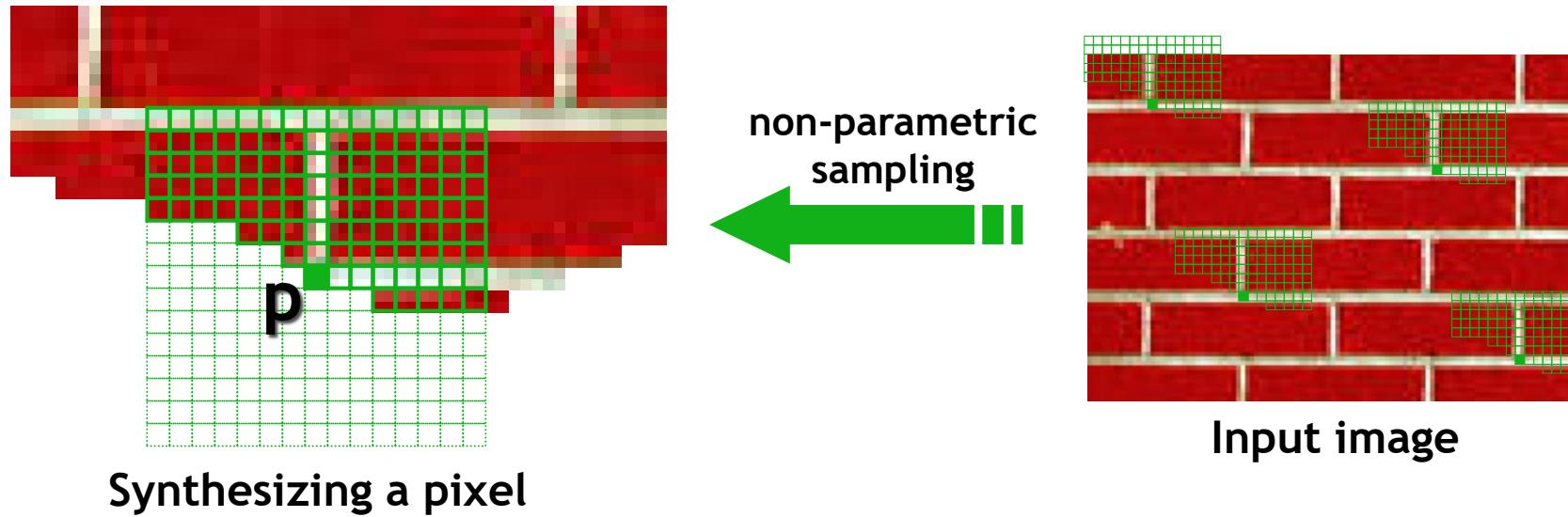
Input



Synthesized



Another idea: Sample from the image



- Assuming Markov property, compute $P(p | N(p))$
 - Building explicit probability tables infeasible
 - Instead, we *search the input image* for all similar neighborhoods — that's our pdf for p
 - To sample from this pdf, just pick one match at random

Idea from Shannon (Information Theory)

- Generate English-sounding sentences by modeling the probability of each word given the previous words (n-grams)
- Large “n” will give more structured sentences

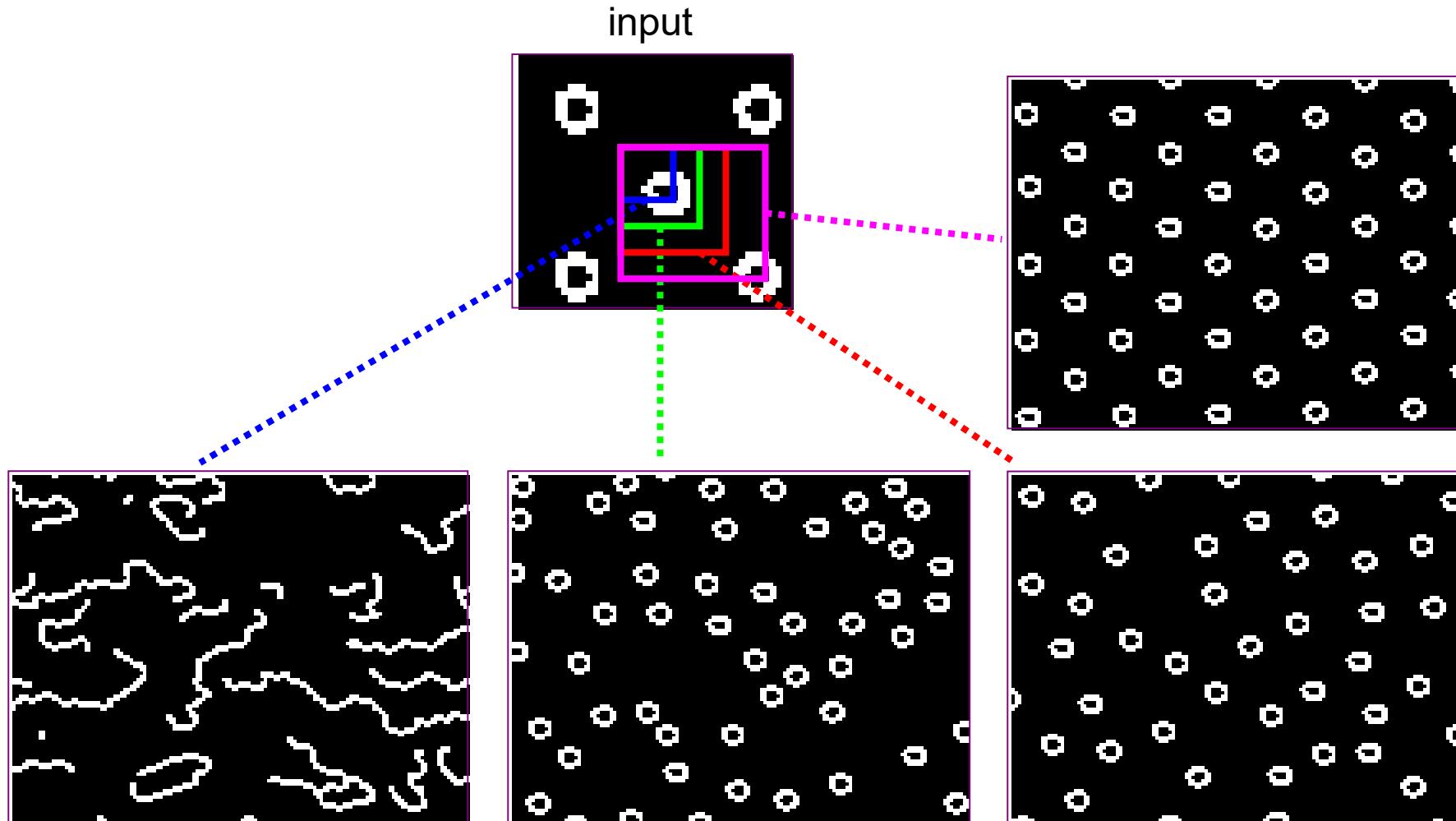
“I spent an interesting evening recently
with a grain of salt.”

(example from fake single.net user [Mark V Shaney](#))

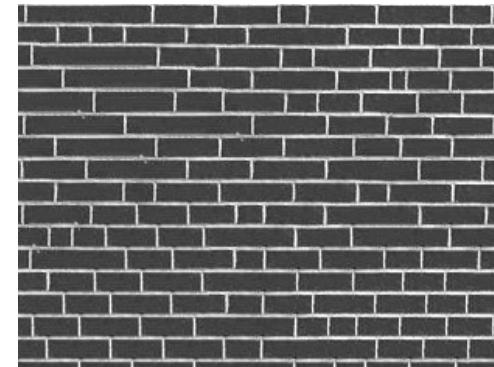
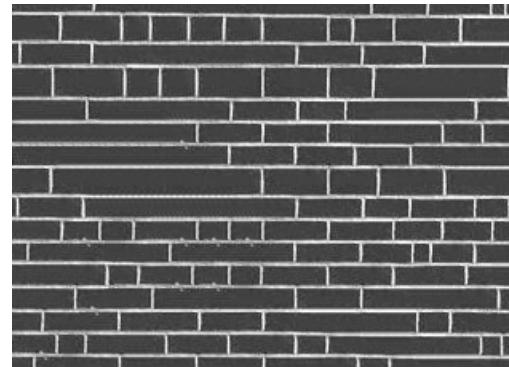
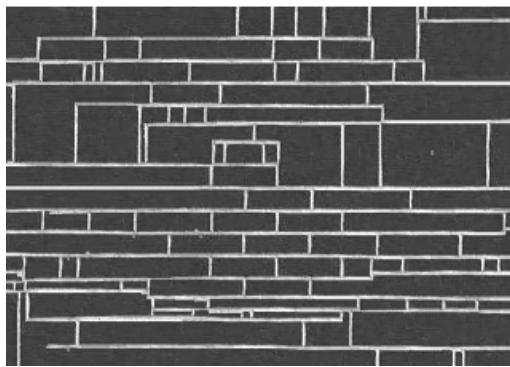
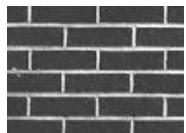
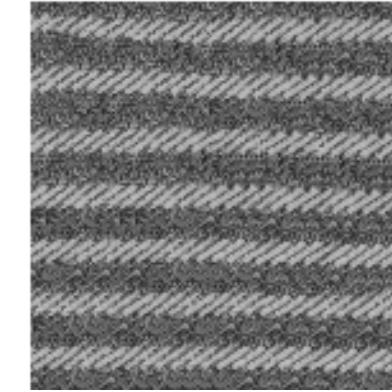
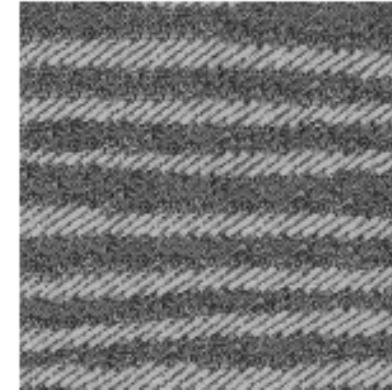
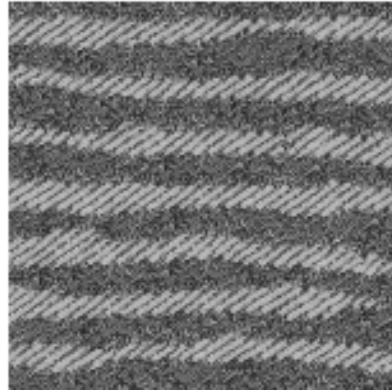
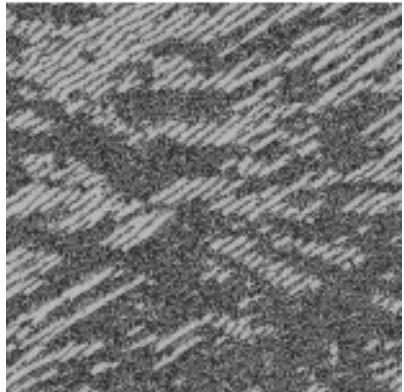
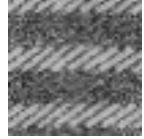
Details

- 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



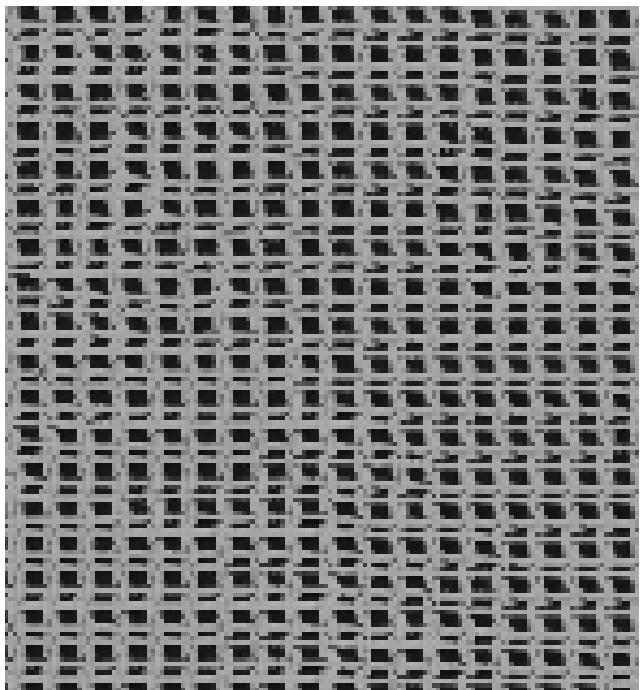
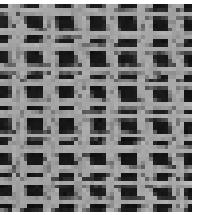
Texture synthesis algorithm

While image not filled

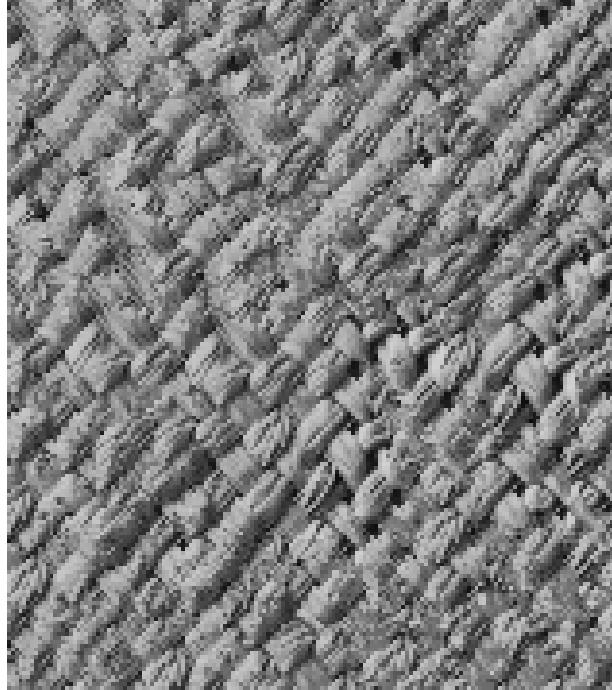
1. Get unfilled pixels with filled neighbors, sorted by number of filled neighbors
2. For each pixel, get top N matches based on visible neighbors
 - Patch Distance: Gaussian-weighted SSD
3. Randomly select one of the matches and copy pixel from it

Synthesis Results

french canvas



rafia weave

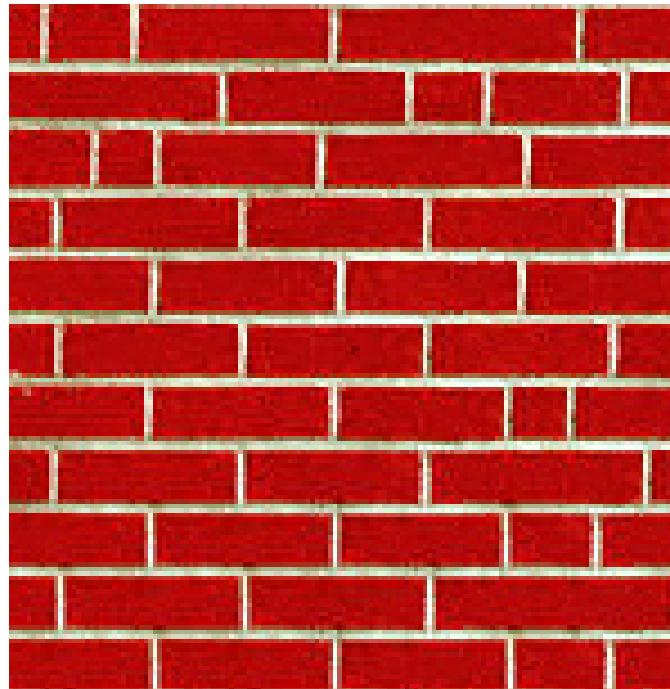
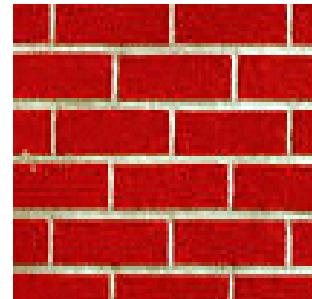


More Results

white bread



brick wall



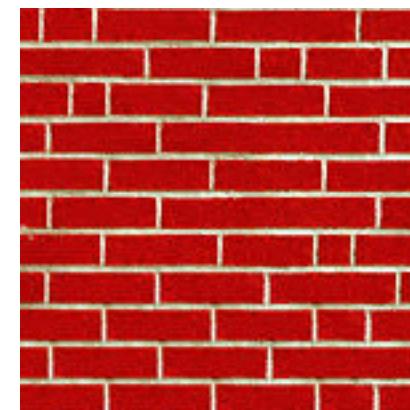
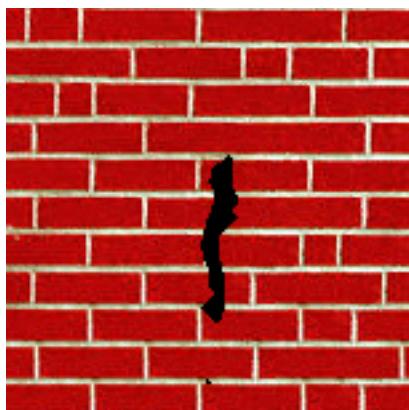
Homage to Shannon

uring in the unsensational
r Dick Gephardt was fai-
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-
, that the legal system k-
g with this latest tanger

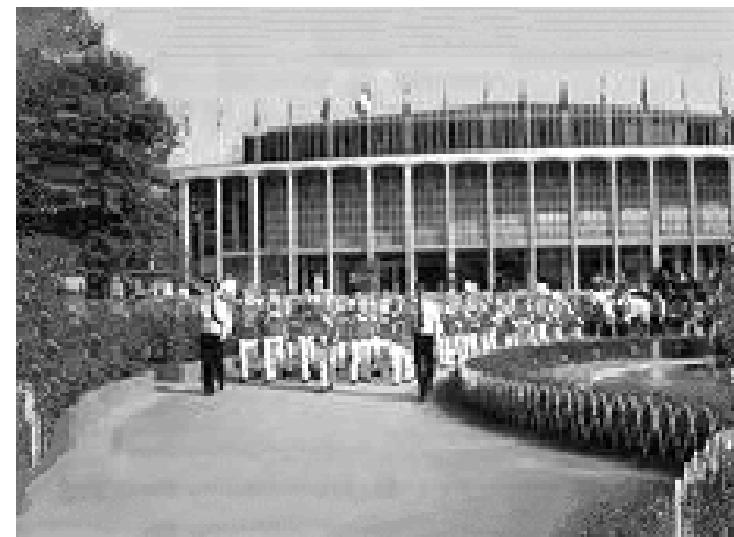
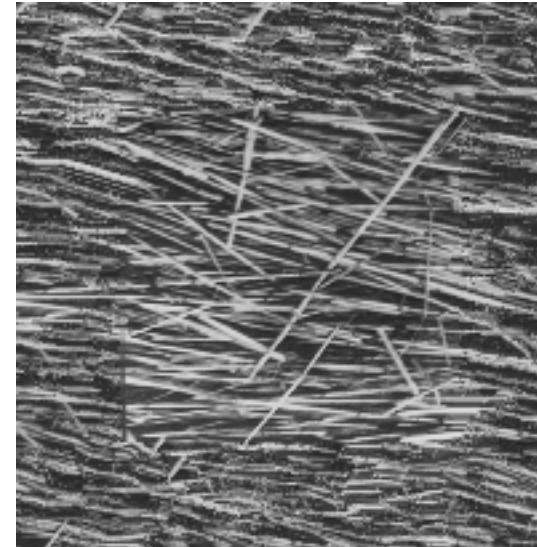
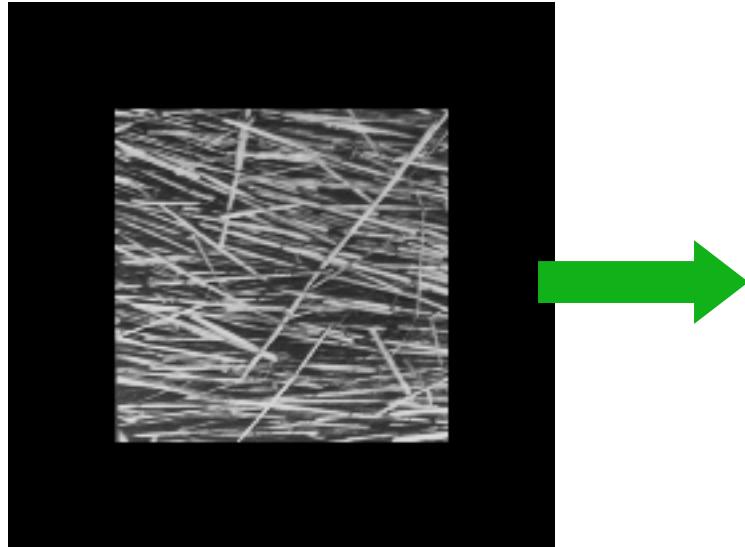


athaim, them . "Whnephartfe lartifelintomimen-
fel ok Clirtioout omaim thartfelins. faut s anentc
the ry onst wartfe lck Gephntoomimeationl sigak
Chiooufit Clinut Cll riff on, hat's yordn, parut tly:
ons yontonsteht wasked, paim t sahe loo' riff on I
nskoneploourtfeas leil A nst Clit, "Wleontongal s
k Cirtioouirtfepe ong pme abegal fartfenstemem
tienstenelitorydt telemphinsvoerdt was agemer
ff ons artientont Cling peme as urfe atih, "Boui s
nal s fartfelt sig pedrl'dt ske abounutie aboutoo
tfeonewwas yous abowonthardt thatins fain, ped,
ains, them, pabout wasy arfiut coutly d, ln A h
ole einthrdingbooneme agas fa bontinsyst Clinut
ory about continst Clipeopinst Cloke agatiff out C
stome zainemen tly ardt beoraboul n, thenly as t C
cons fairmem Diontont wat coutlyohgans as fan
ien, phrtfaul, "Wbaut cout congagal comininga:
mifmst Cliiy abon al coountha.emungaint tf oun
The looorystan loontieph, intly on, theoplegatick C
aul tatteeonly atie Dioniomi wal s f tbegae ener
nthahsat's enenhmas fan, "intchthorw ahons v

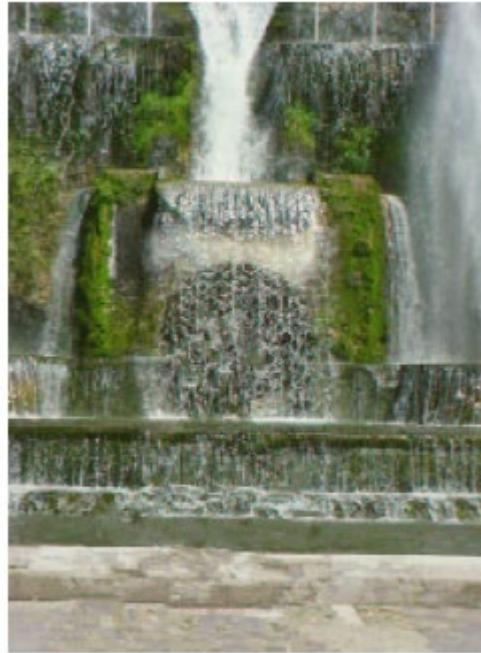
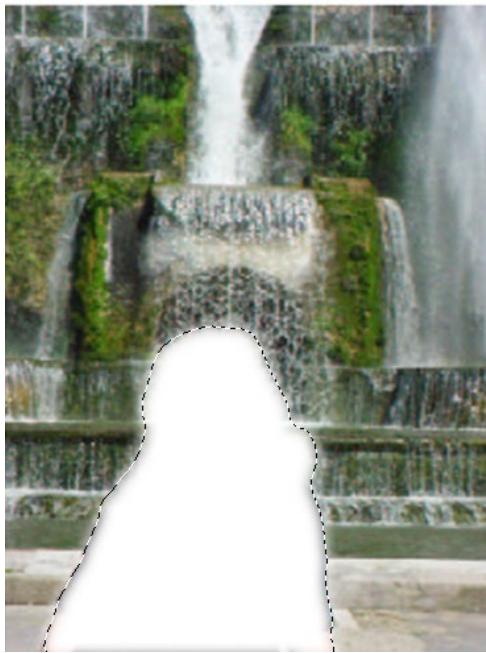
Hole Filling



Extrapolation



In-painting natural scenes



Key idea: Filling order matters

In-painting Result



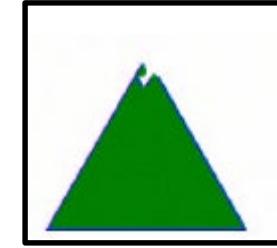
Image with Hole



Raster-Scan Order



Onion-Peel
(Concentric Layers)

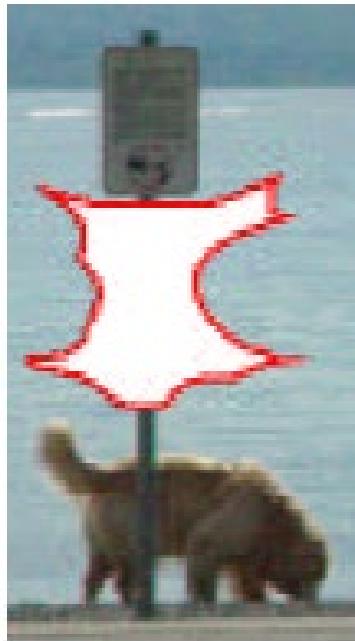


Gradient-Sensitive
Order

Filling order

Fill a pixel that:

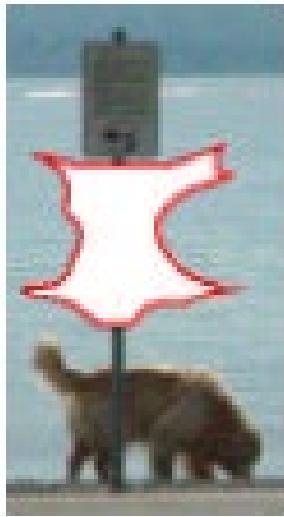
1. Is surrounded by other known pixels
2. Is a continuation of a strong gradient or edge



Comparison



Original



With Hole



Onion-Ring Fill



Criminisi

Comparison



a



b



Concentric Layers

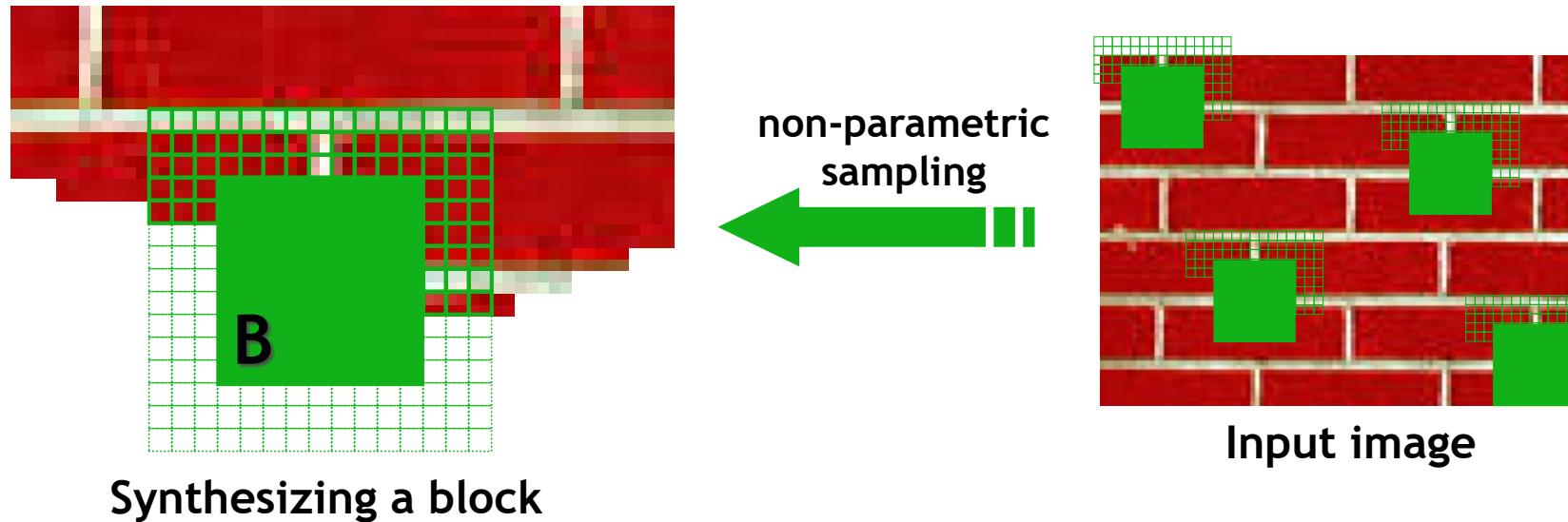


Gradient Sensitive

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]

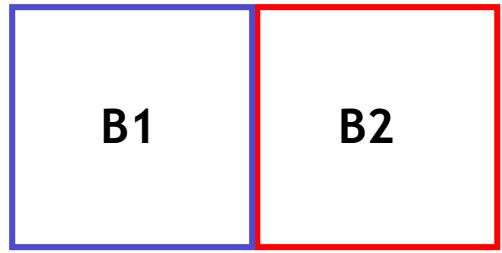
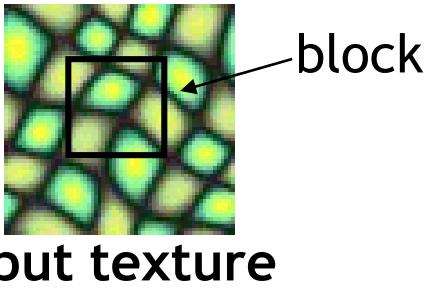


Synthesizing a block

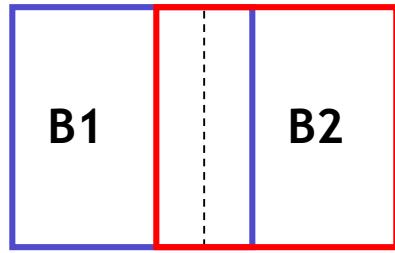
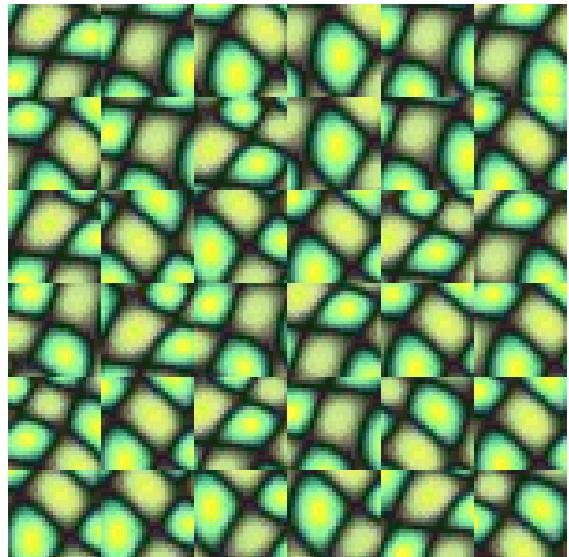
- Observation: neighbor pixels are highly correlated

Idea: unit of synthesis = block

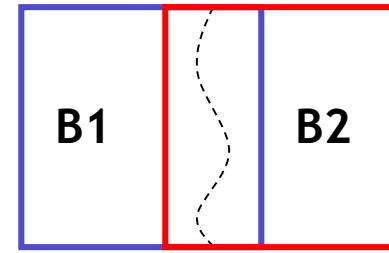
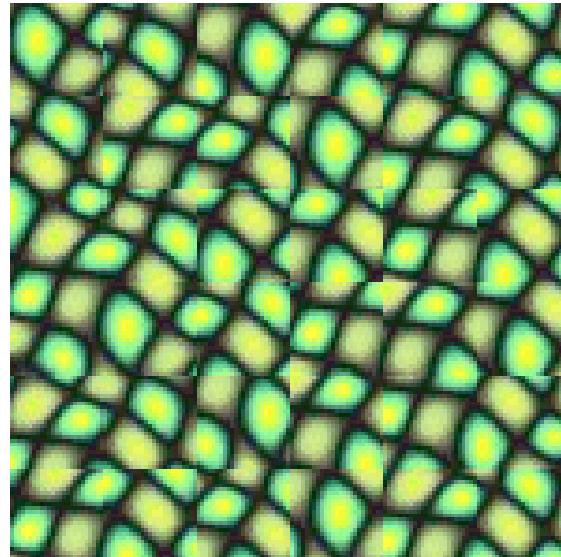
- Exactly the same but now we want $P(B|N(B))$
- Much faster: synthesize all pixels in a block at once



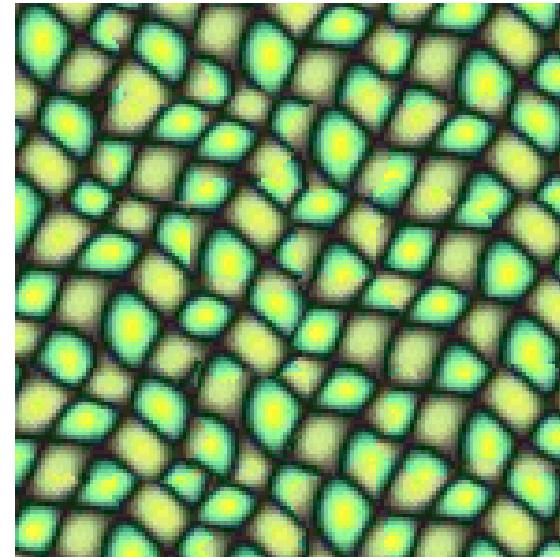
Random placement
of blocks



Neighboring blocks
constrained by overlap

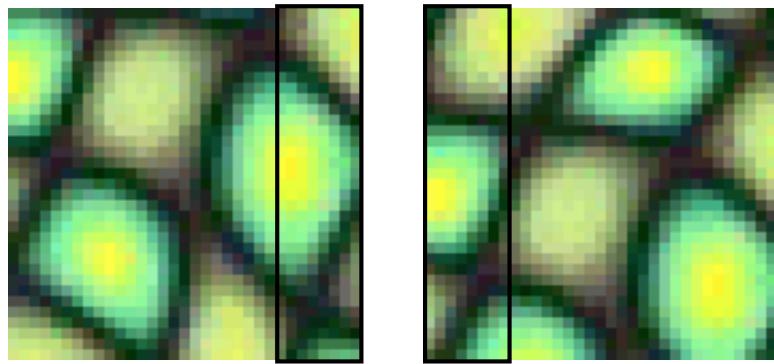


Minimal error
boundary cut

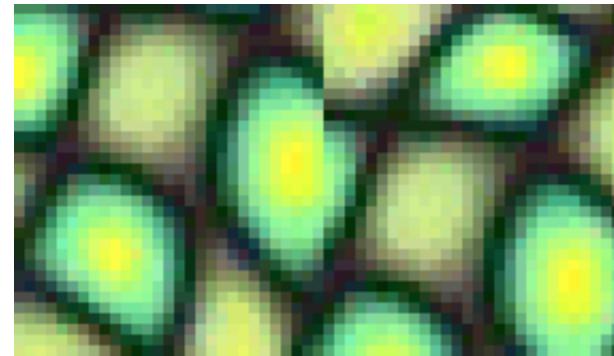


Minimal error boundary

overlapping blocks



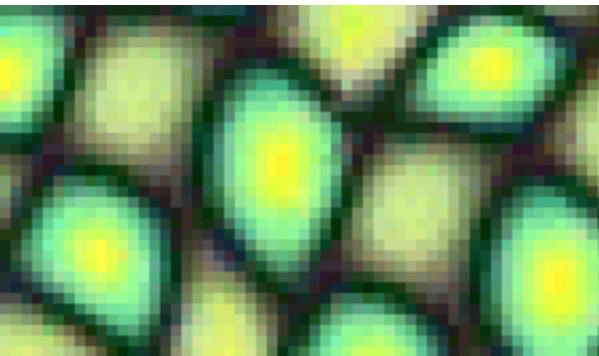
vertical boundary



$$\left(\begin{array}{c} \text{[Heatmap block]} \\ - \\ \text{[Heatmap block]} \end{array} \right)^2 = \text{[Red boundary map]}$$

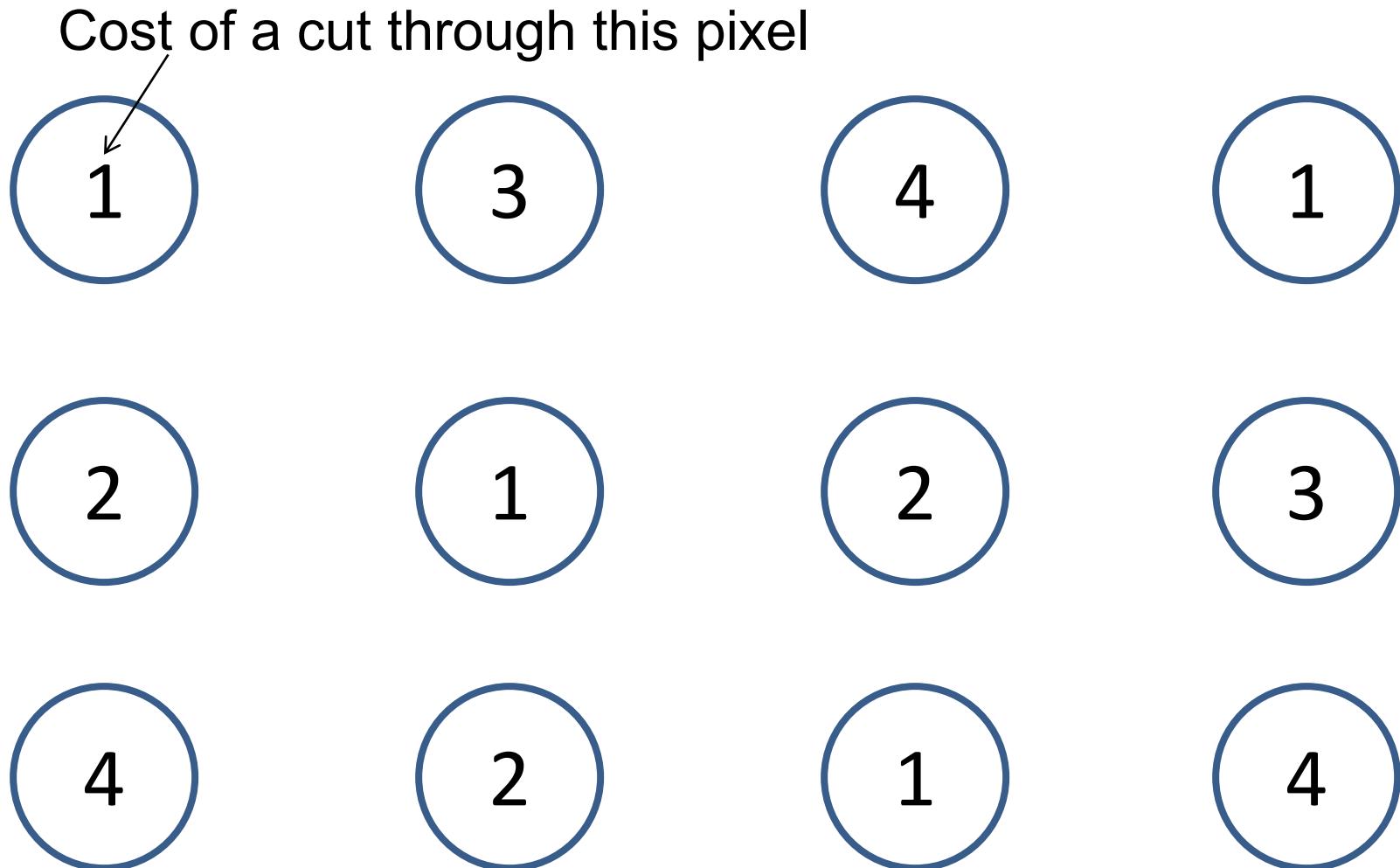
overlap error

The diagram shows two overlapping blocks of a heatmap. A bracket groups the two blocks with a minus sign between them, followed by a squared symbol (^2) and an equals sign (=). To the right of the equals sign is a small heatmap with a red jagged boundary, representing the overlap error.

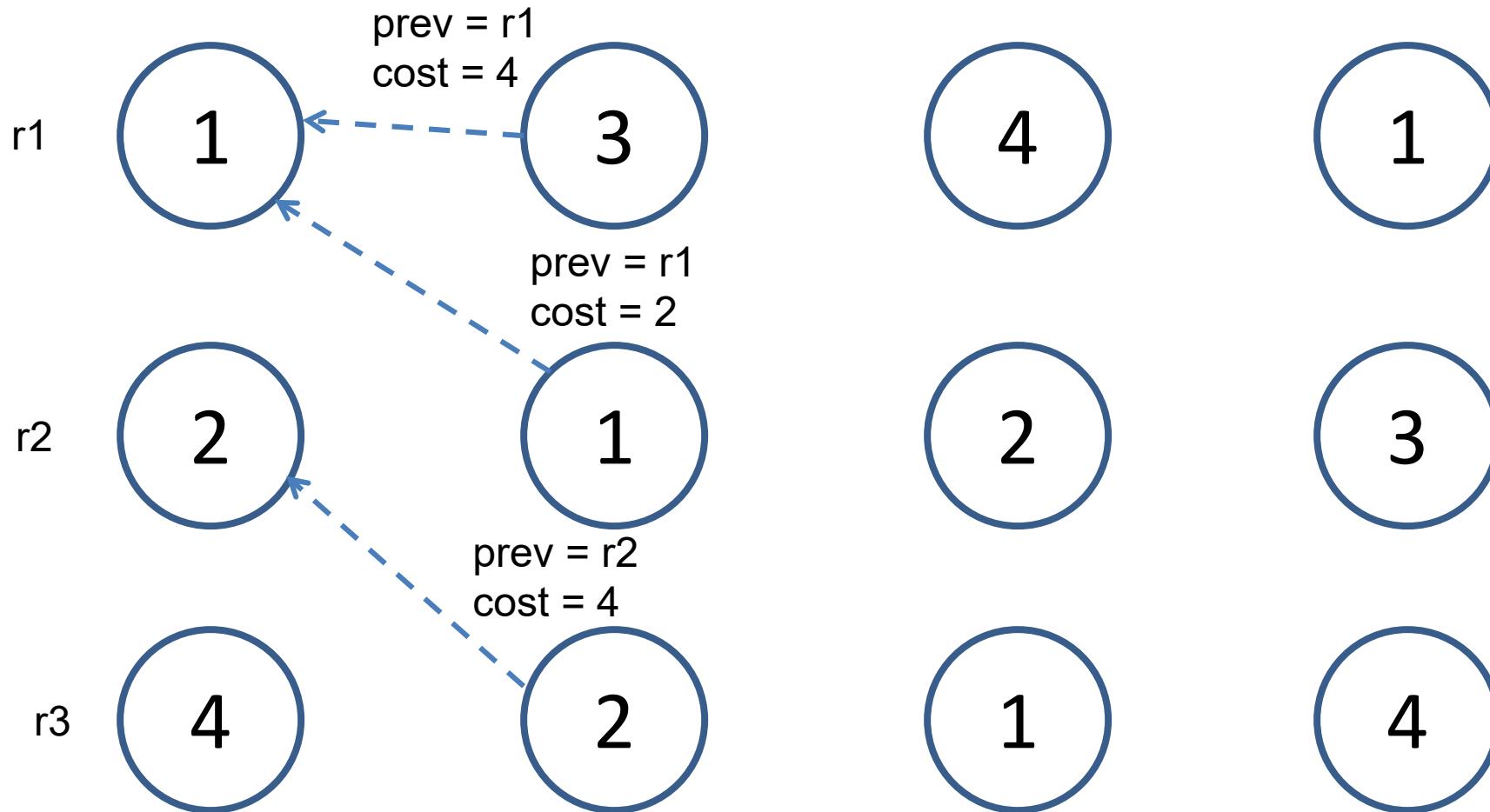


min. error boundary

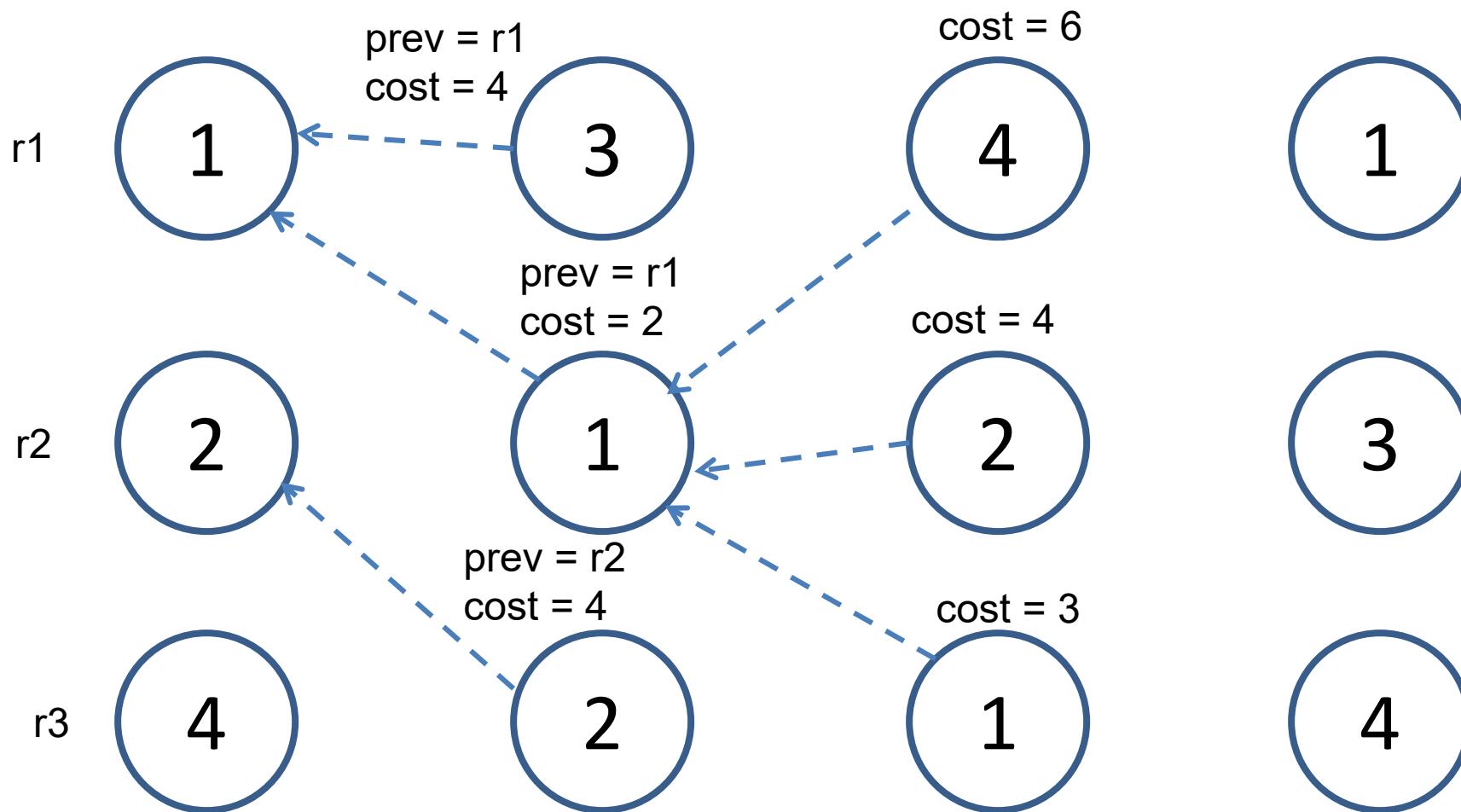
Solving for Minimum Cut Path



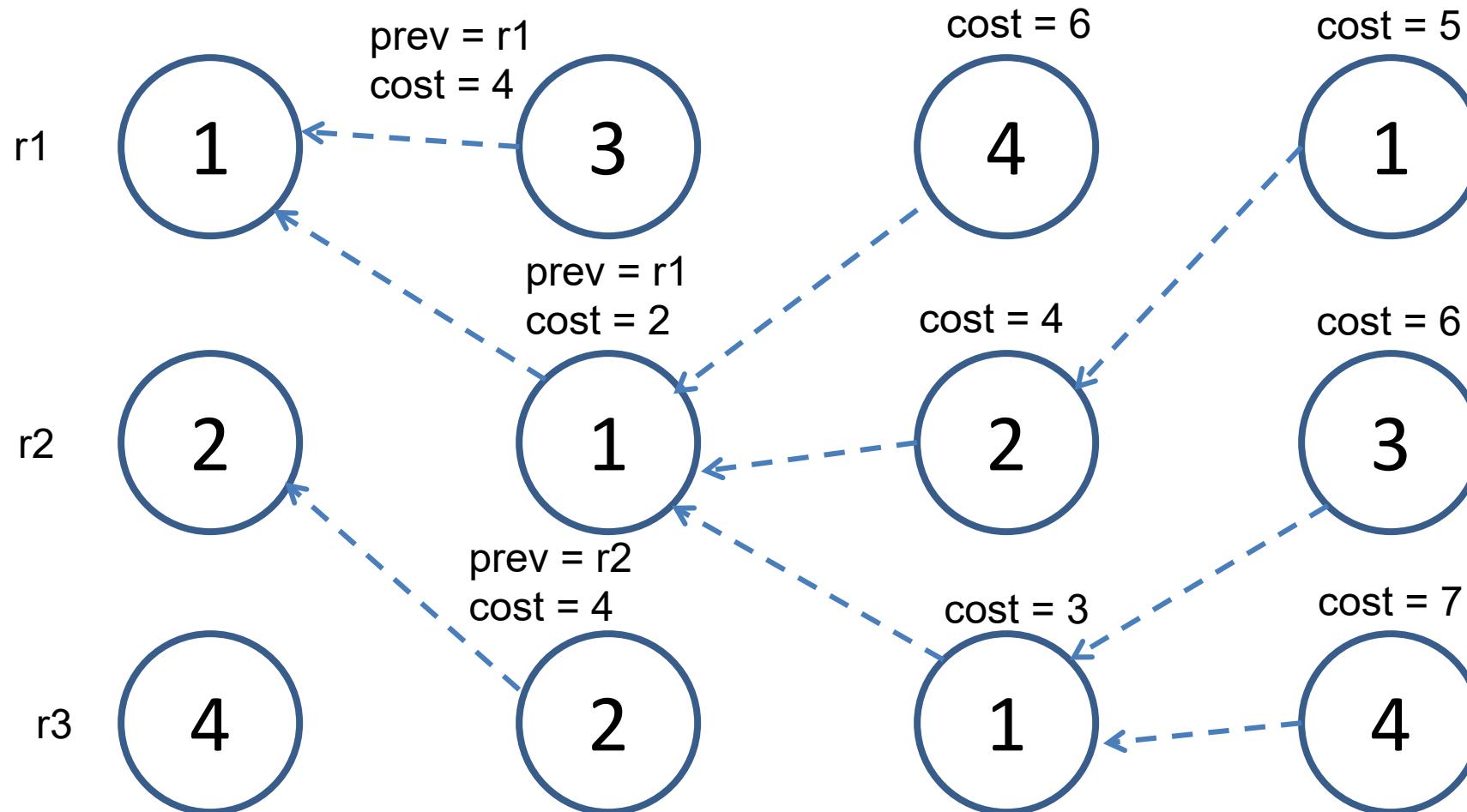
Solving for Minimum Cut Path



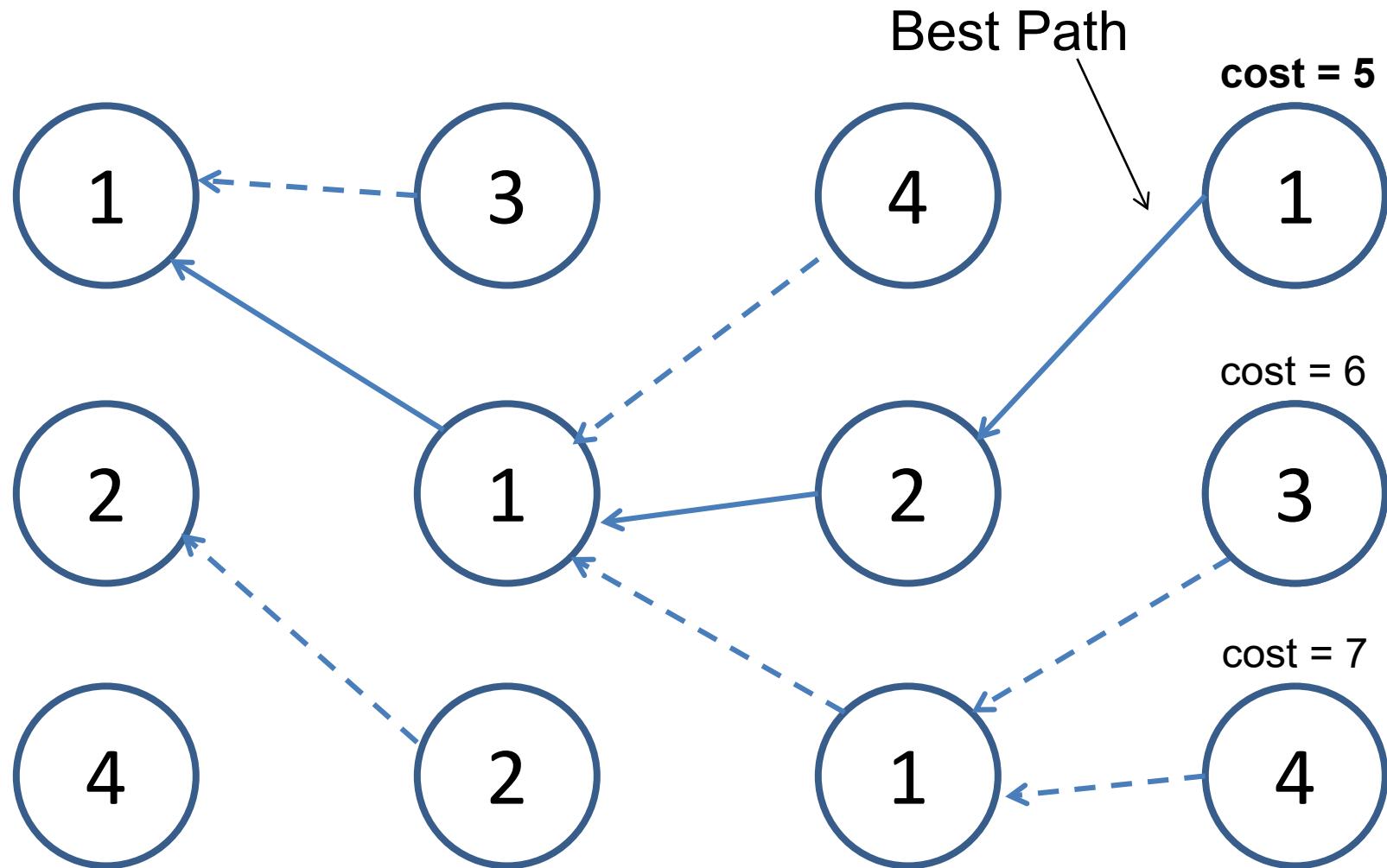
Solving for Minimum Cut Path



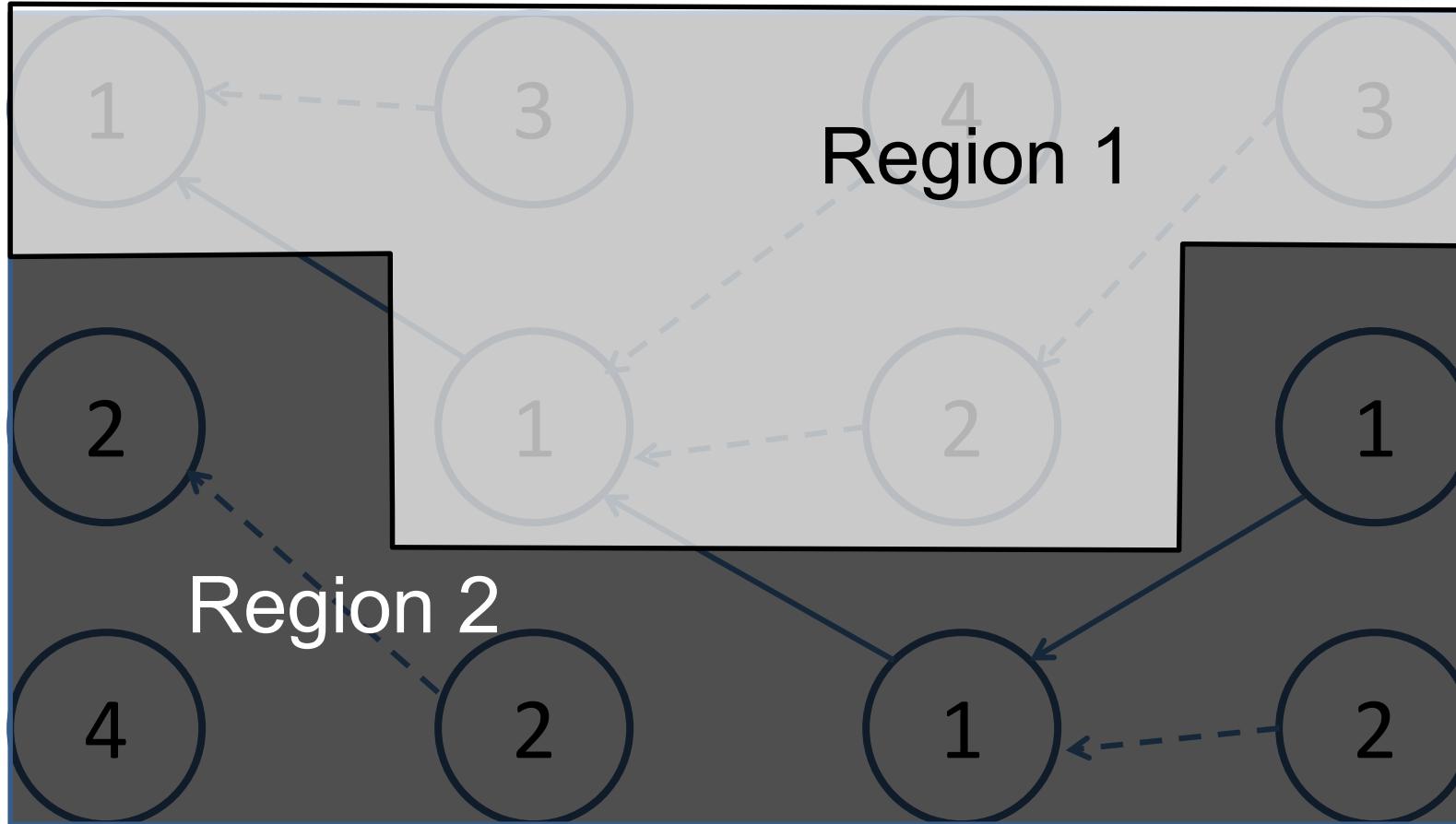
Solving for Minimum Cut Path



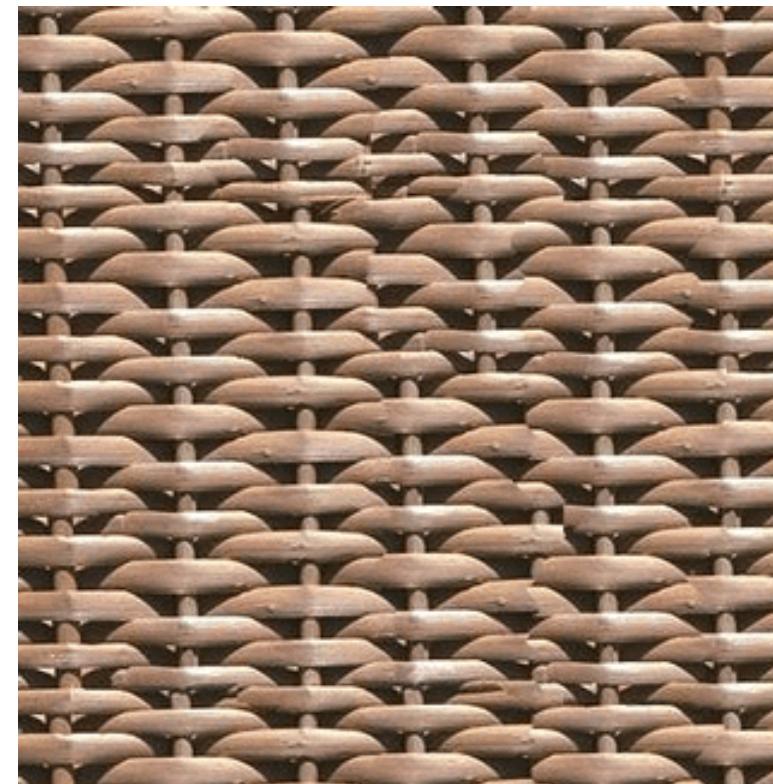
Solving for Minimum Cut Path

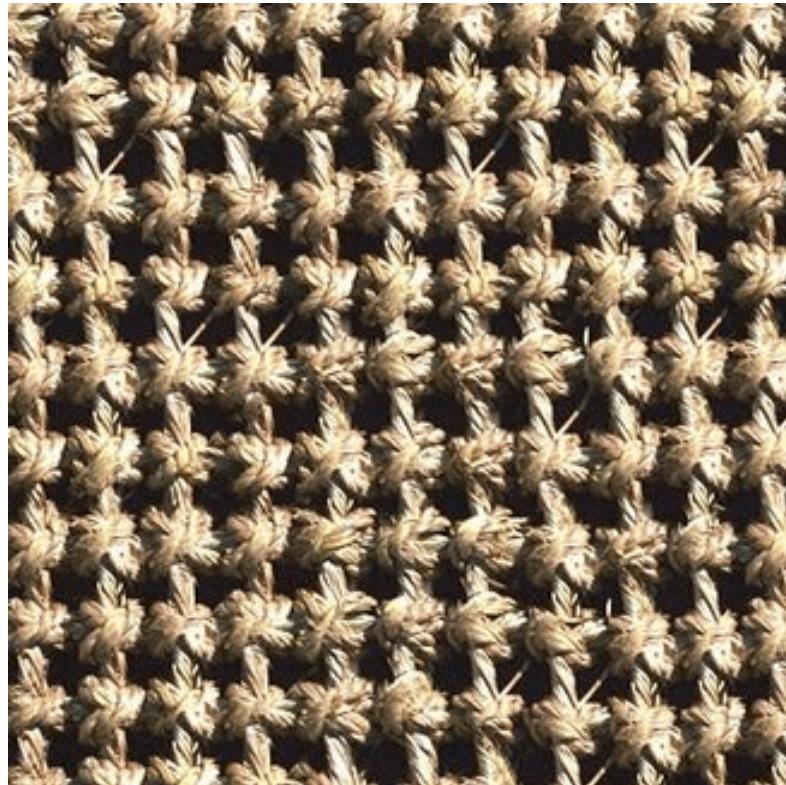


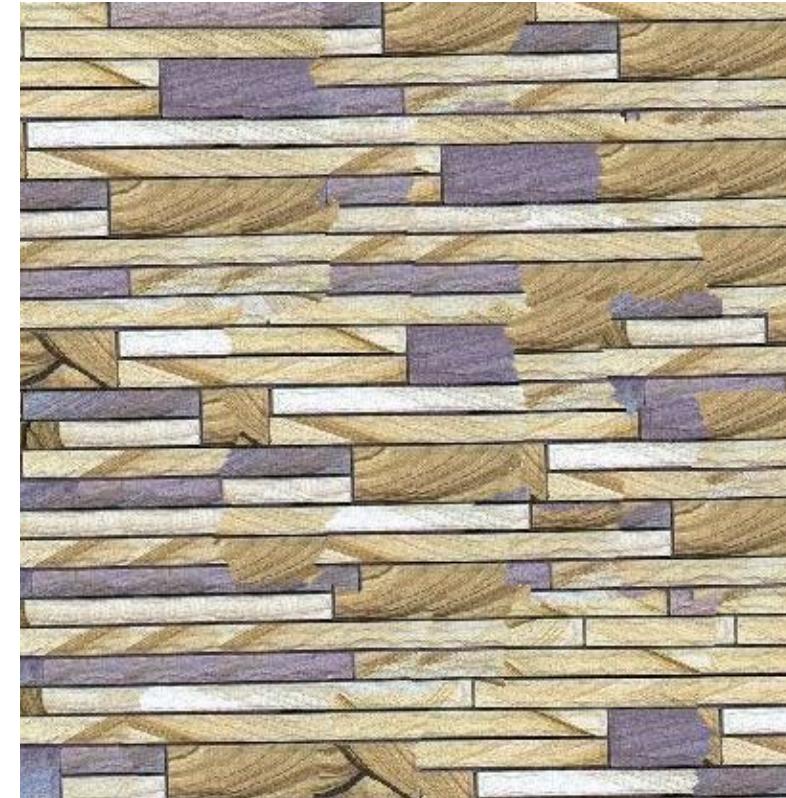
Solving for Minimum Cut Path



Mask Based on Best Path

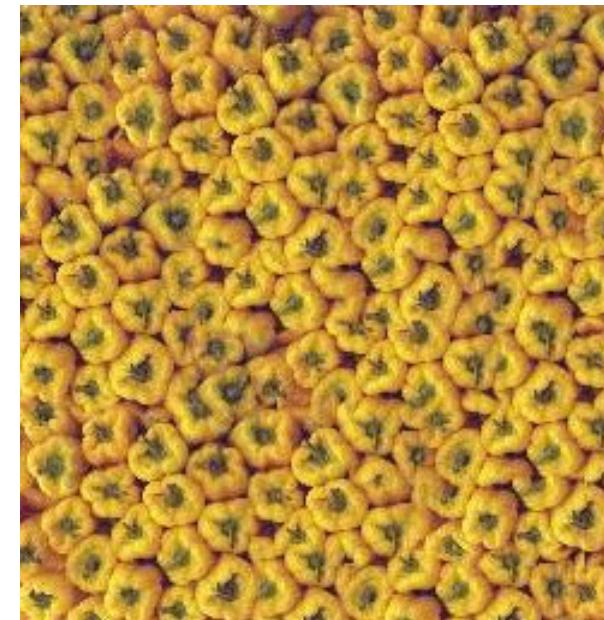
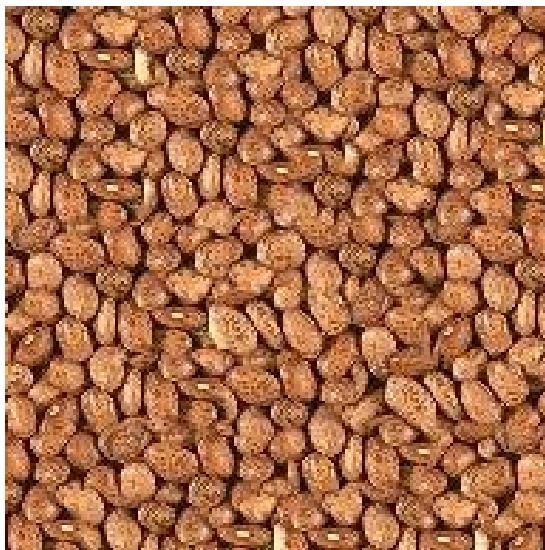
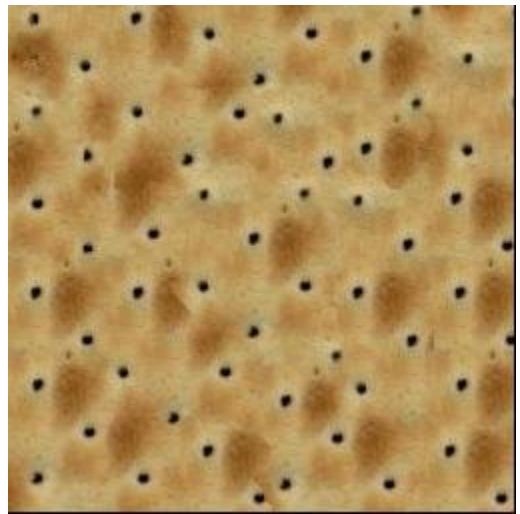
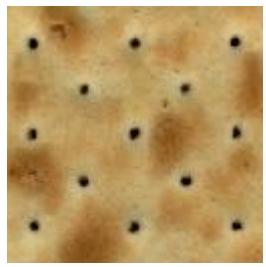




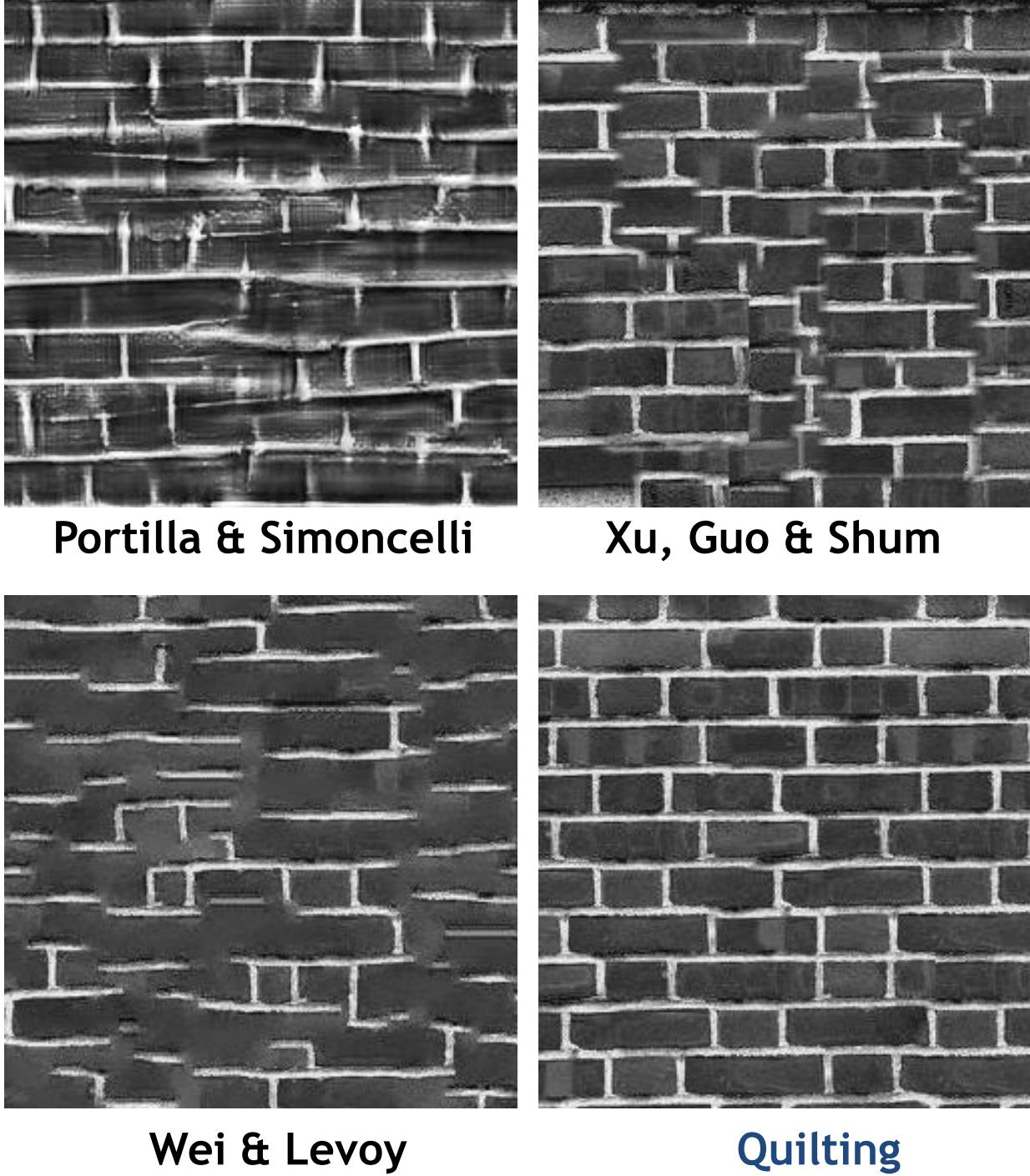








input image



eld of a visual cortical neuron—the model describing the response of that neuron as a function of position—is perhaps the most functional description of that neuron. We seek a single conceptual and mathematical framework to describe the wealth of simple-cell receptive fields neurophysiologically¹⁻³ and inferred especially if such a framework has the added benefit of helping us to understand the function in a deeper way. Whereas no generic model exists for all simple-cell receptive fields (DOG), difference of offset Gaussian derivative of a Gaussian, higher derivatives of a function, and so on—can be expected to provide a good approximation to simple-cell receptive field, we nonetheless need to consider other possibilities.

input image

Wei & Levoy

des and simple-cell receptors, the first neurons in the visual system. These neurons act as a filter, selecting certain types of information from the visual field. They are also involved in the detection of motion and depth perception. The second type of neuron is the complex cell, which receives input from multiple simple cells and integrates them to produce a more complex response. Complex cells are involved in tasks such as object recognition and scene analysis. The third type of neuron is the hypercomplex cell, which receives input from multiple complex cells and produces a highly specific response. Hypercomplex cells are involved in tasks such as reading and writing. The fourth type of neuron is the parvocellular neuron, which receives input from the macaque monkey's retina and processes color information. The fifth type of neuron is the magnocellular neuron, which receives input from the macaque monkey's retina and processes low-level visual features such as orientation and size.

Xu, Guo & Shum

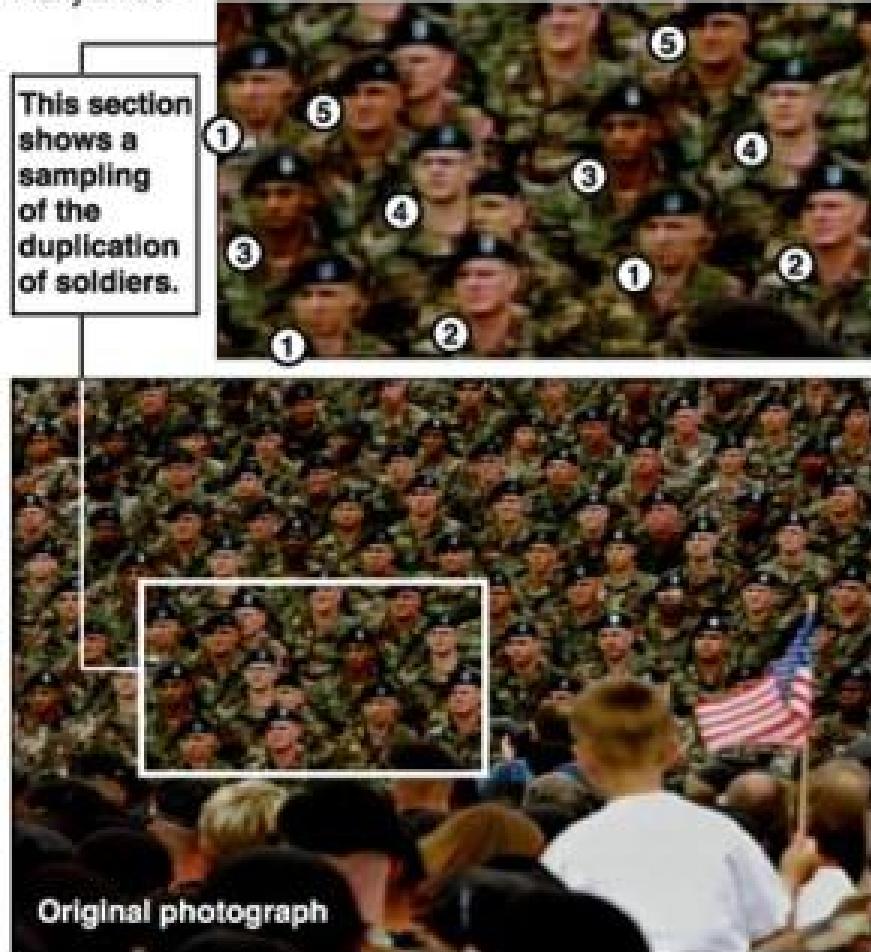
sition—is perk a single conceptual and of that neurube the wealth of simple- ual and matheurophysiologically¹⁻³ and simple-cell necially if such a framework^{y¹⁻³} and infer:lps us to understand the framework has perhay. Whereas no ge and the fumeuro:DOG), difference o is no generic a single conceptual and mrence of offse the wealth of simple-ce , higher deriescribing the response of —can be expes a function of position- helps us to understand truption of th per way. Whereas no gconceptual an sians (DOG), differencealth of simple

Quilting

Political Texture Synthesis

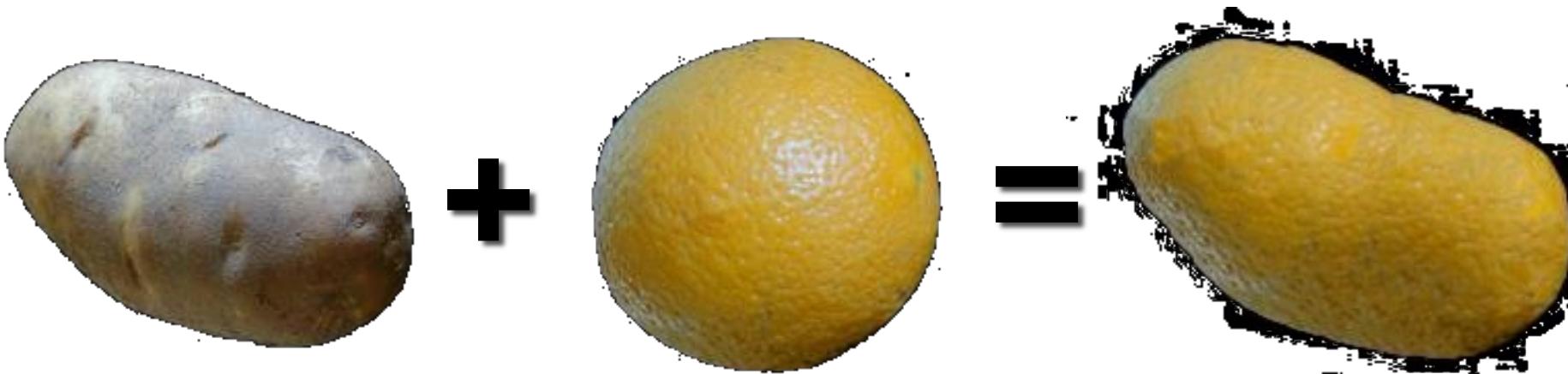
Bush campaign digitally altered TV ad

President Bush's campaign acknowledged Thursday that it had digitally altered a photo that appeared in a national cable television commercial. In the photo, a handful of soldiers were multiplied many times.



Texture Transfer

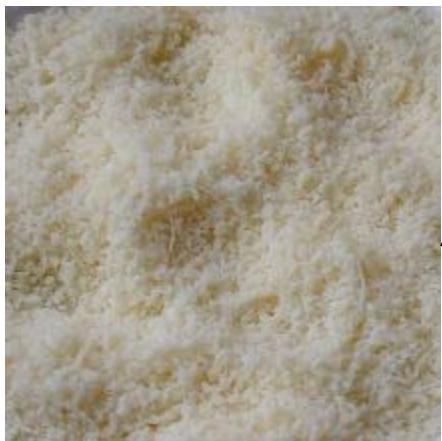
- Try to explain one object with bits and pieces of another object:



Texture Transfer



Constraint

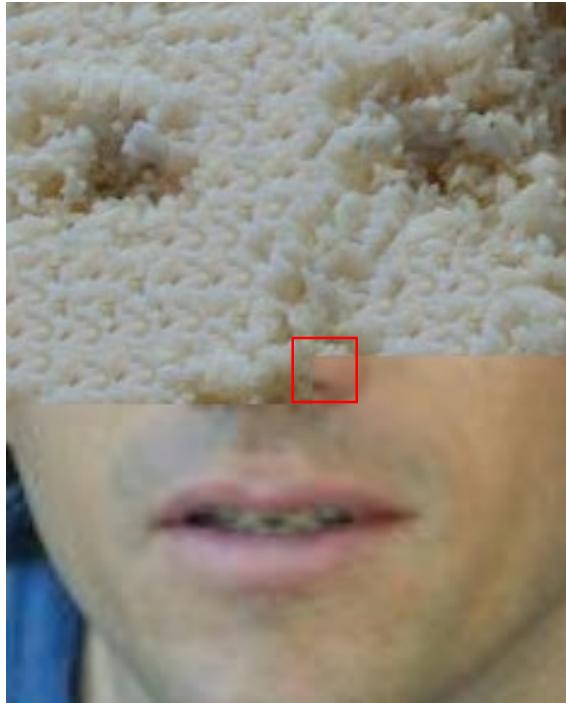


Texture sample



Texture Transfer

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

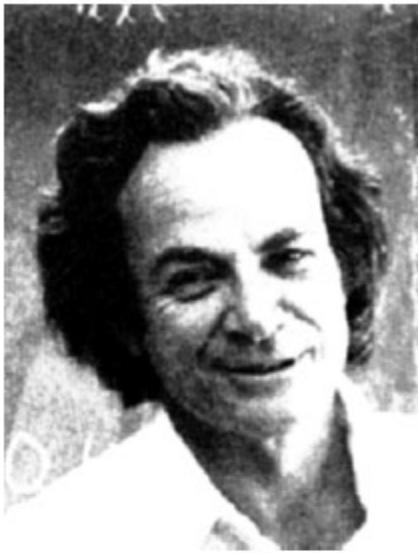


Same as texture synthesis, except an additional constraint:

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



source texture



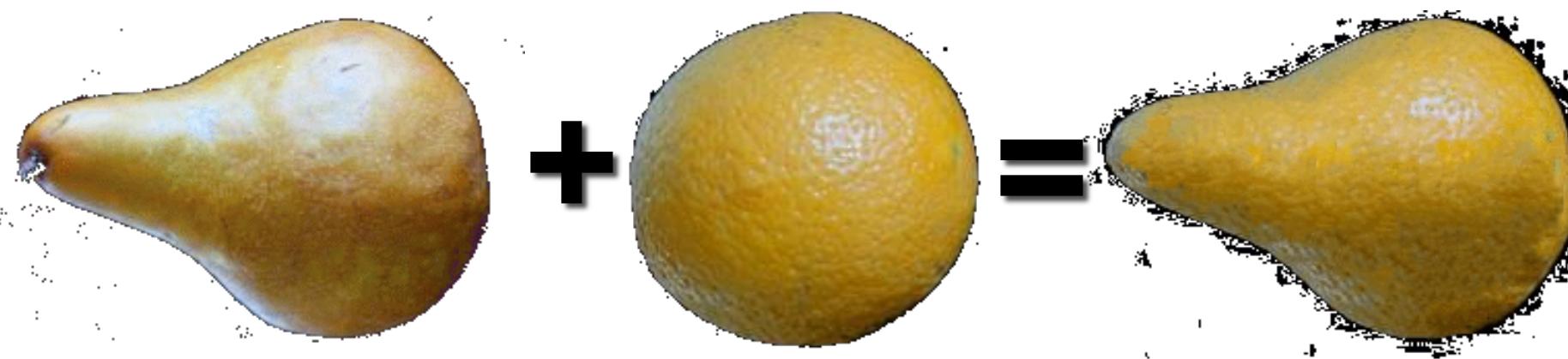
target image

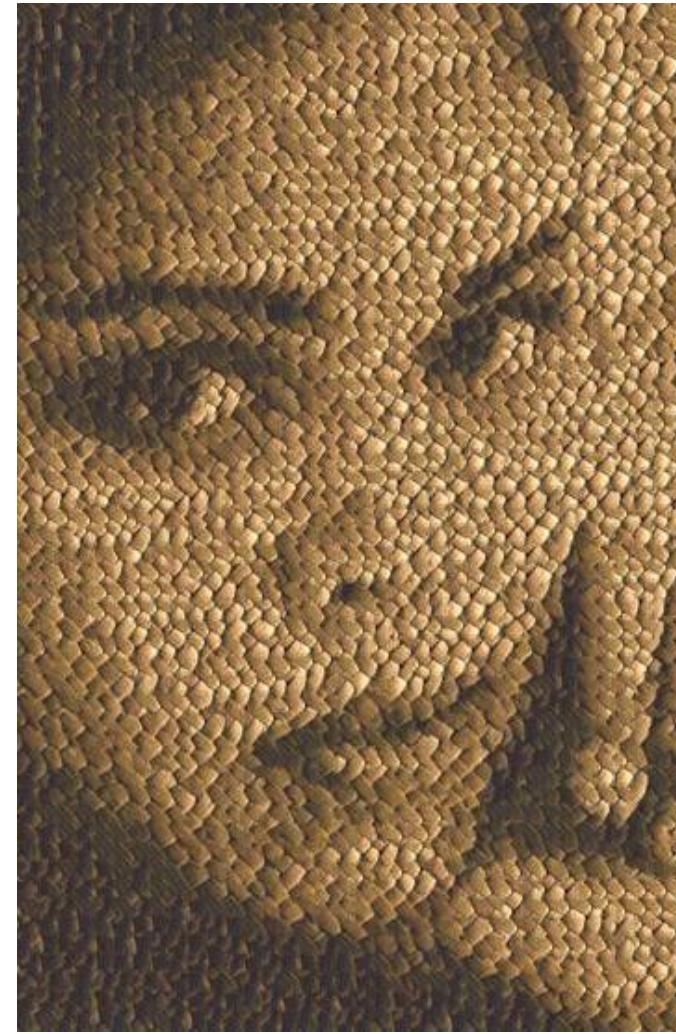
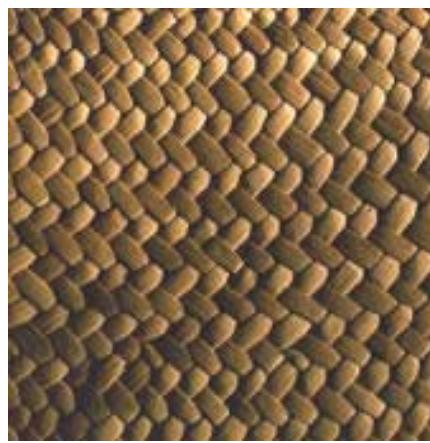


correspondence maps



texture transfer result





Making sacred toast



+

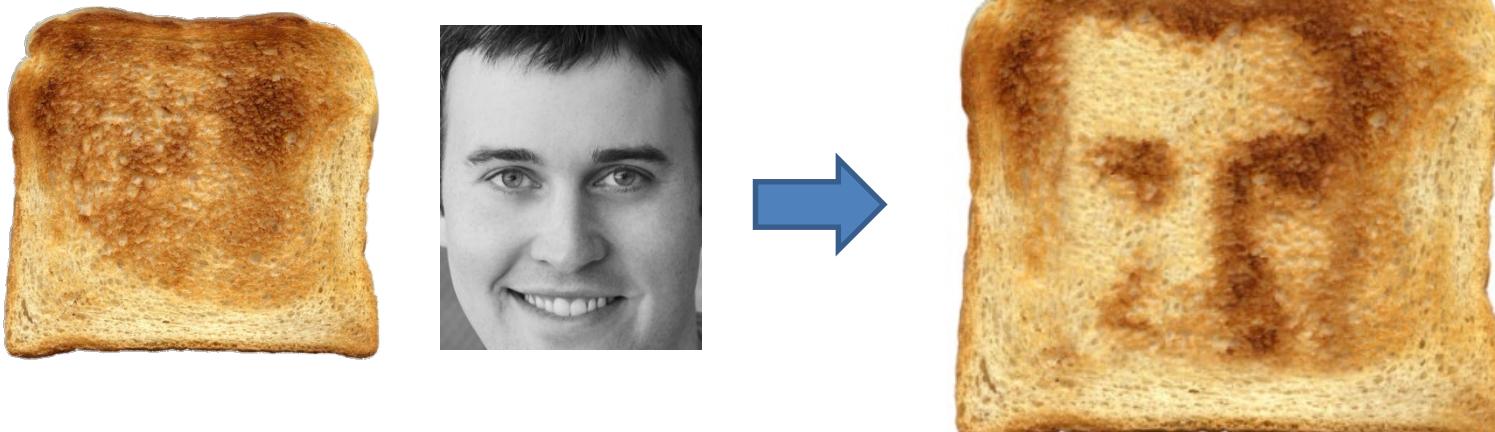
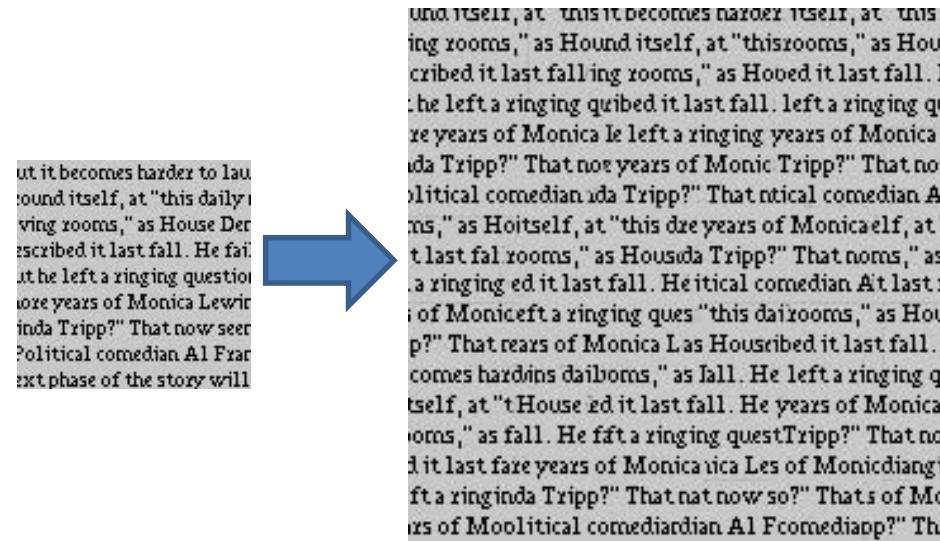


?
→

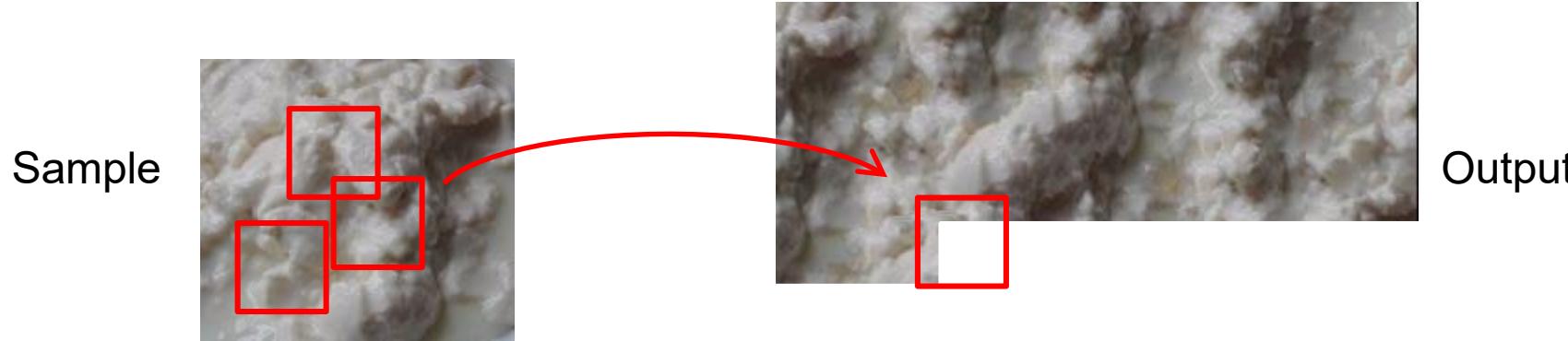


Project 2: texture synthesis and transfer

- https://courses.engr.illinois.edu/cs445/fa2019/projects/quilting/ComputationalPhotography_ProjectQuilting.html
- Note: this is significantly more challenging than the first project



Texture Synthesis and Transfer Recap



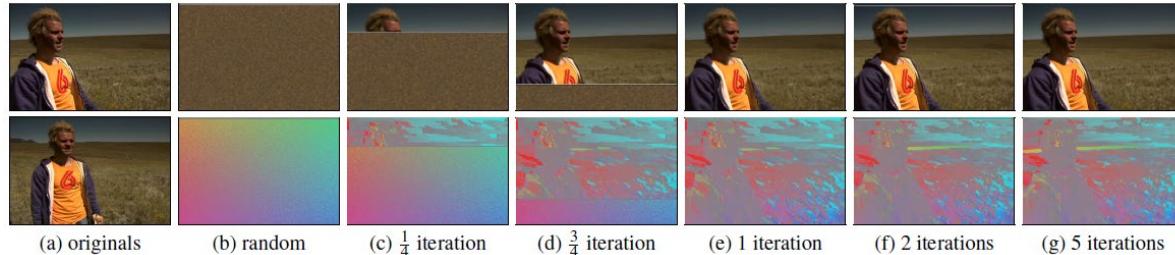
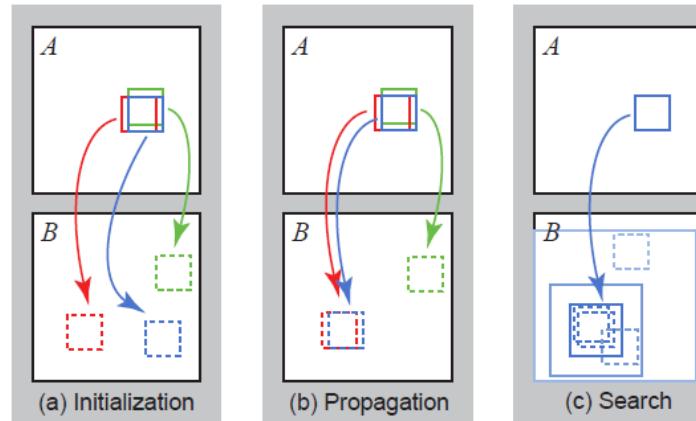
For each overlapping patch in the output image

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 the source and target correspondence patches should match.
2. Select one sample patch that has a small cost (e.g. randomly pick one of K candidates)
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

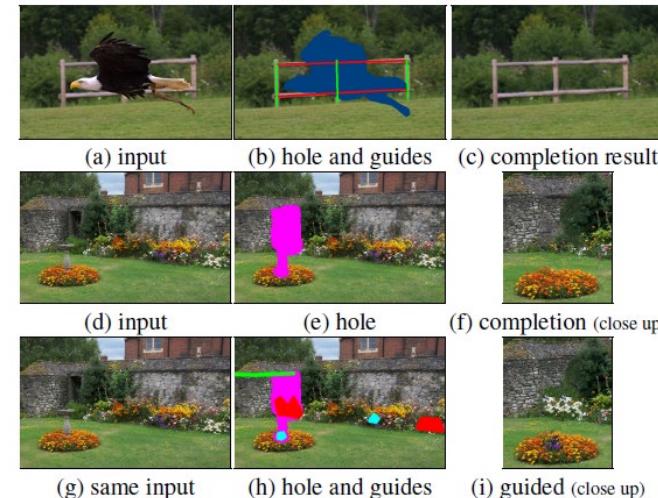
PatchMatch

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 3, 4 across image several times



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



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

Related idea: Image Analogies



A



A'



B

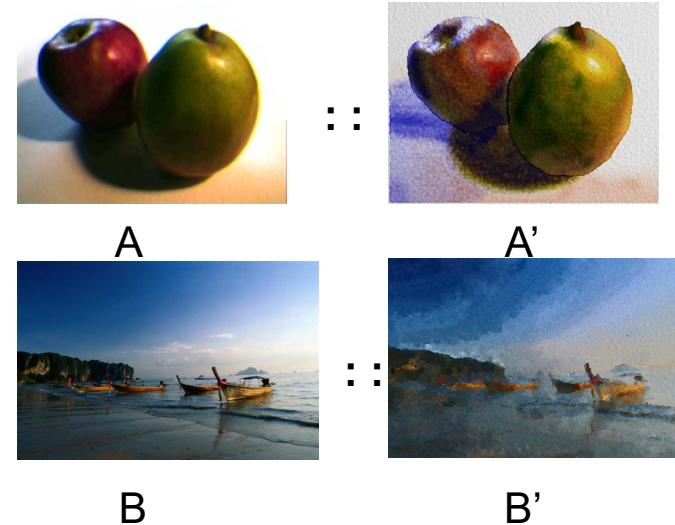


B'

Image Analogies, Hertzmann et al. SG 2001



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

Image-to-Image Translation with Conditional Adversarial Networks

<https://phillipi.github.io/pix2pix/>

Phillip Isola

Jun-Yan Zhu

Tinghui Zhou

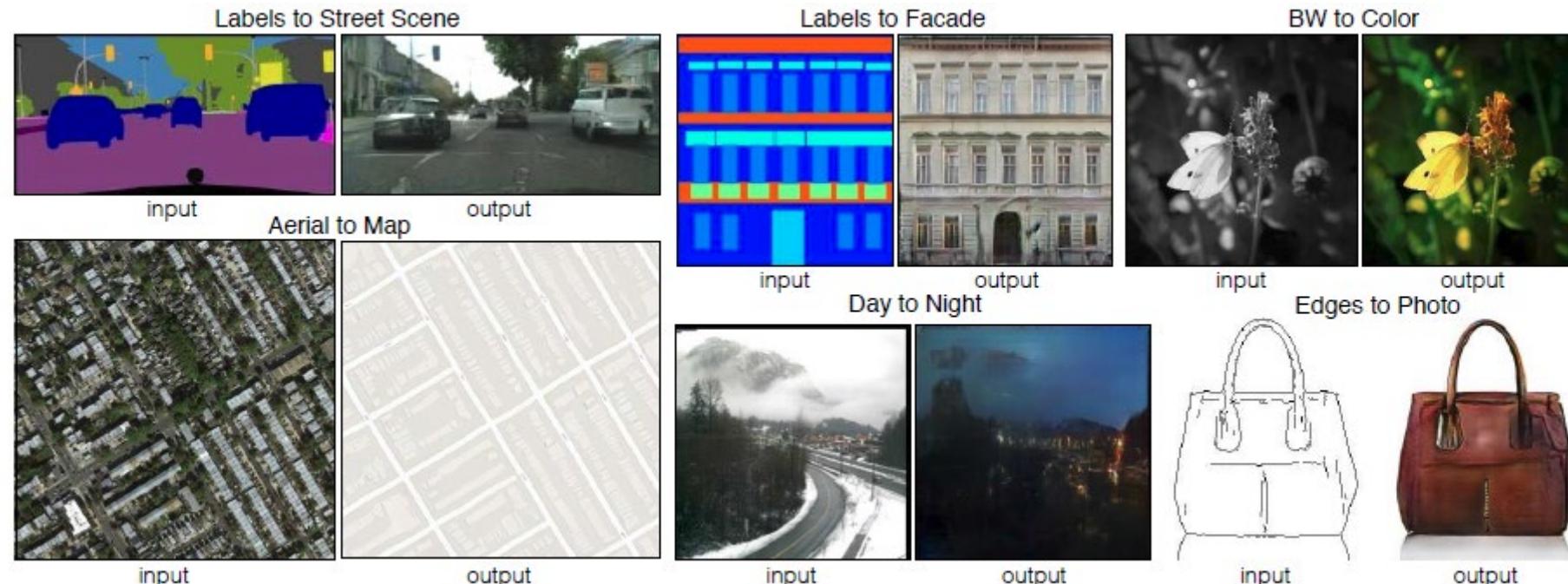
Alexei A. Efros

Berkeley AI Research (BAIR) Laboratory
University of California, Berkeley

{isola, junyanz, tinghuiz, efros}@eecs.berkeley.edu CVPR 2017

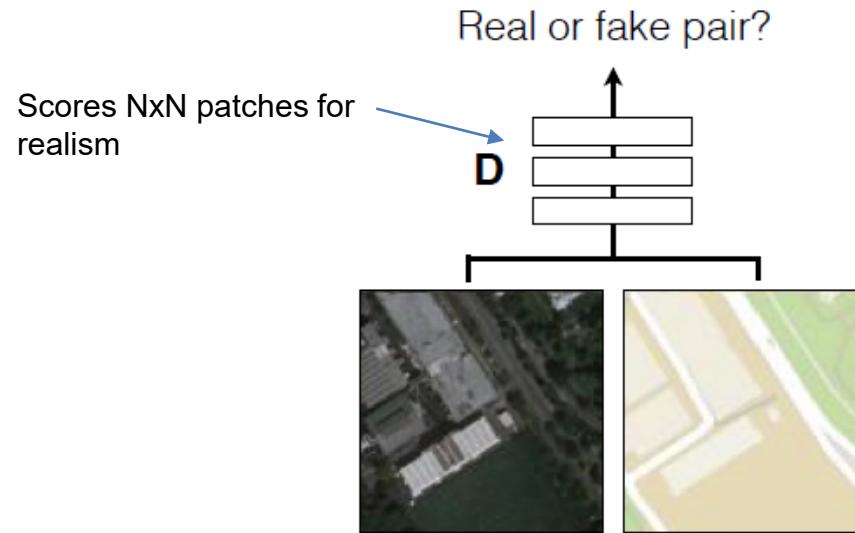
Learn to map from one image representation to another

- Trained from input/output pairs
- Patch memorization is implicit through learned representation



Learning to synthesize

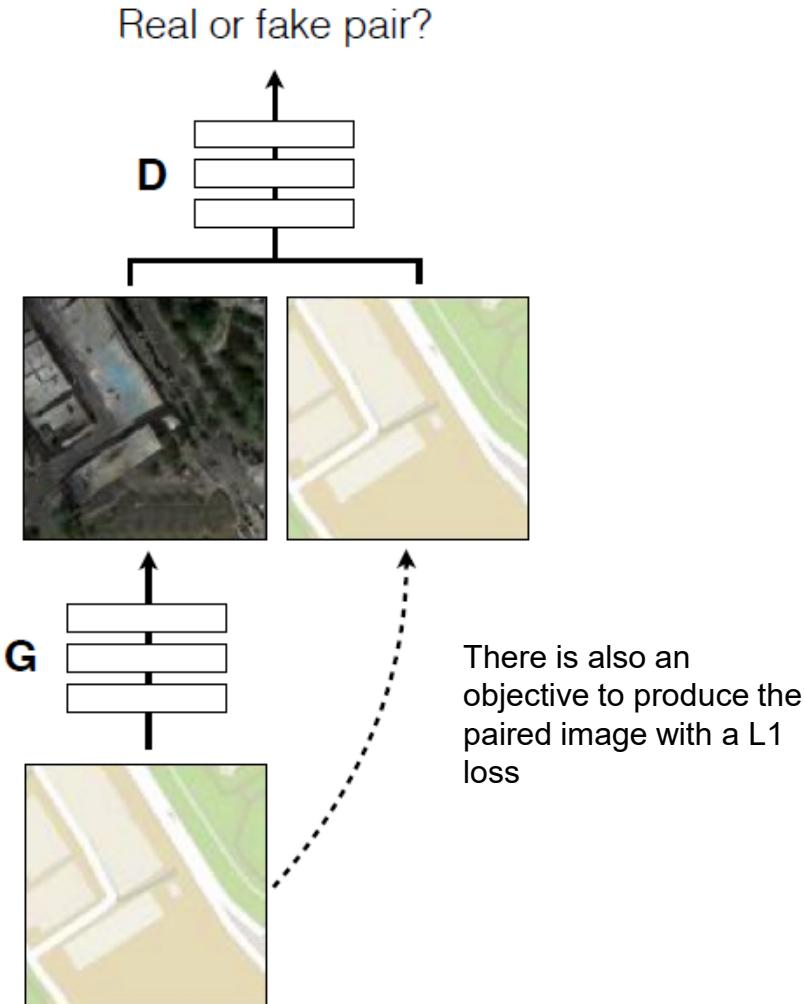
Positive examples



G tries to synthesize fake images that fool **D**

D tries to identify the fakes

Negative examples



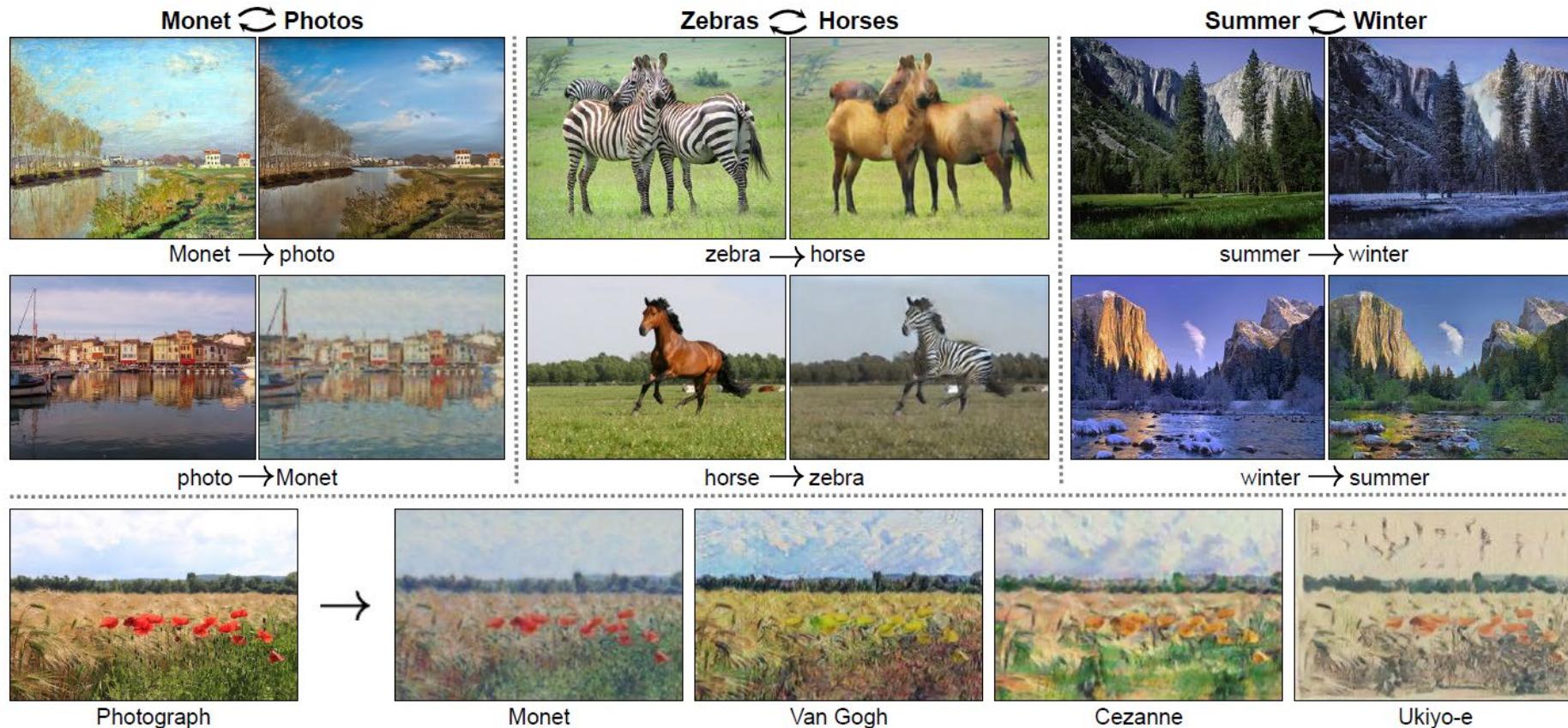
Demos

<https://affinelayer.com/pixsrv/>

Cycle GAN (ICCV 2017)

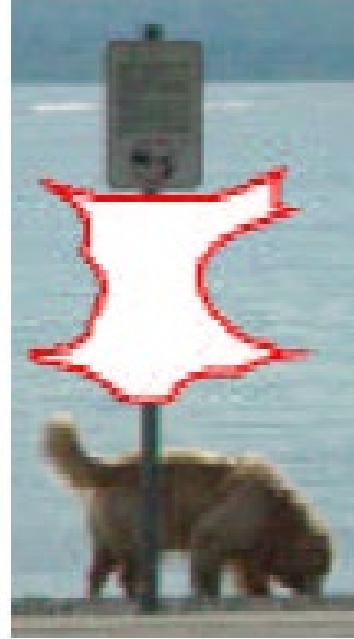
Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros
Berkeley AI Research (BAIR) laboratory, UC Berkeley



Things to remember

- Texture synthesis and hole-filling can be thought of as a form of probabilistic hallucination
- Simple, similarity-based matching is a powerful tool
 - Synthesis
 - Hole-filling
 - Transfer
 - Artistic filtering
 - Super-resolution
 - Recognition, etc.
- Key is how to define similarity and efficiently find neighbors
- New methods learn patch/image representations to create more flexible synthesis, so that similarity function and “neighbors” are implicit



Next class

- Cutting and seam finding