Texture Synthesis and Hole-Filling

Computational Photography

Yuxiong Wang, University of Illinois

Slides adopted from Derek Hoiem

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

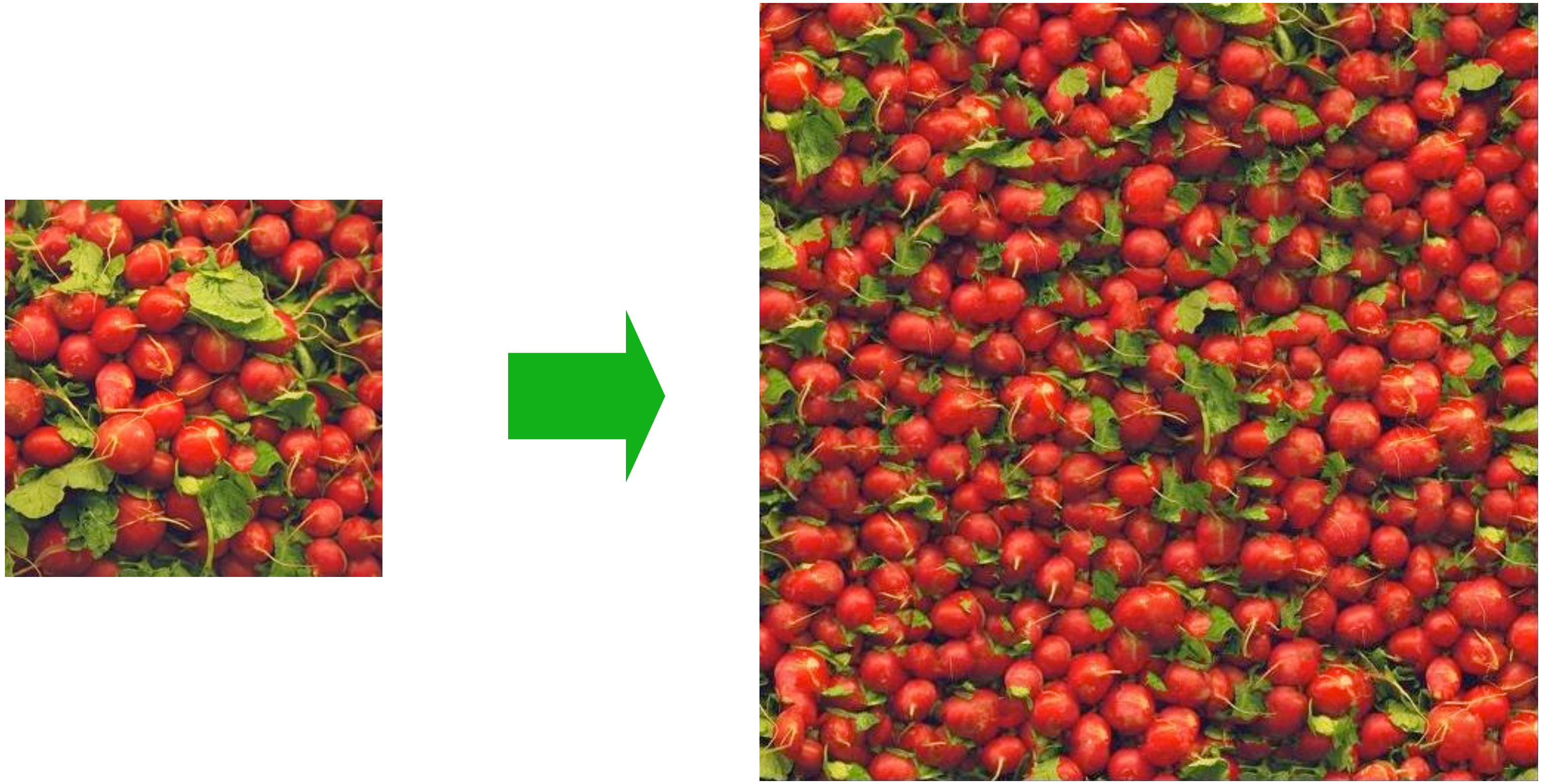
rocks

yogurt

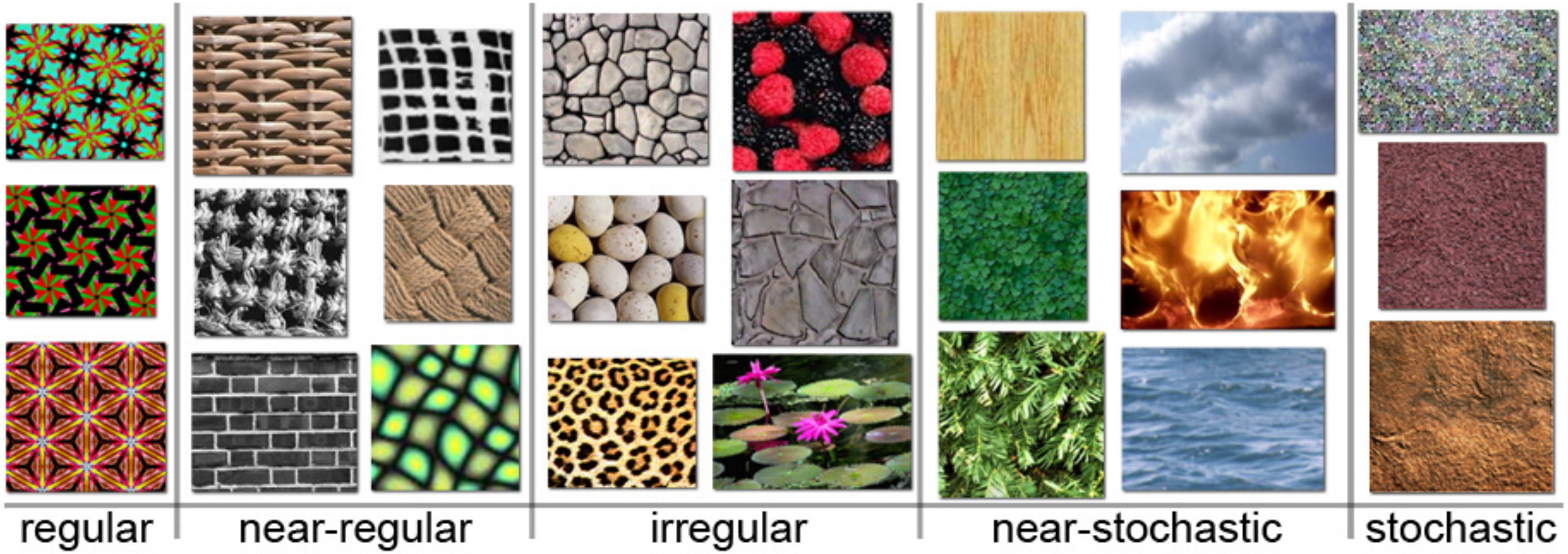
Many slides from James Hays

Texture Synthesis

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



The Challenge

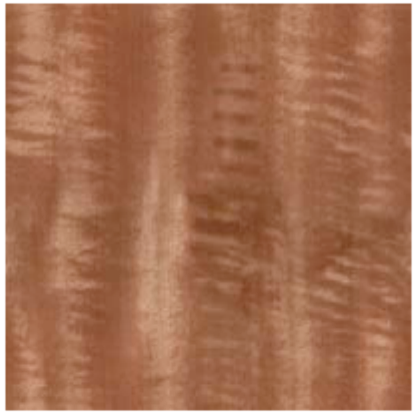


Need to model the whole spectrum: from repeated to stochastic texture

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

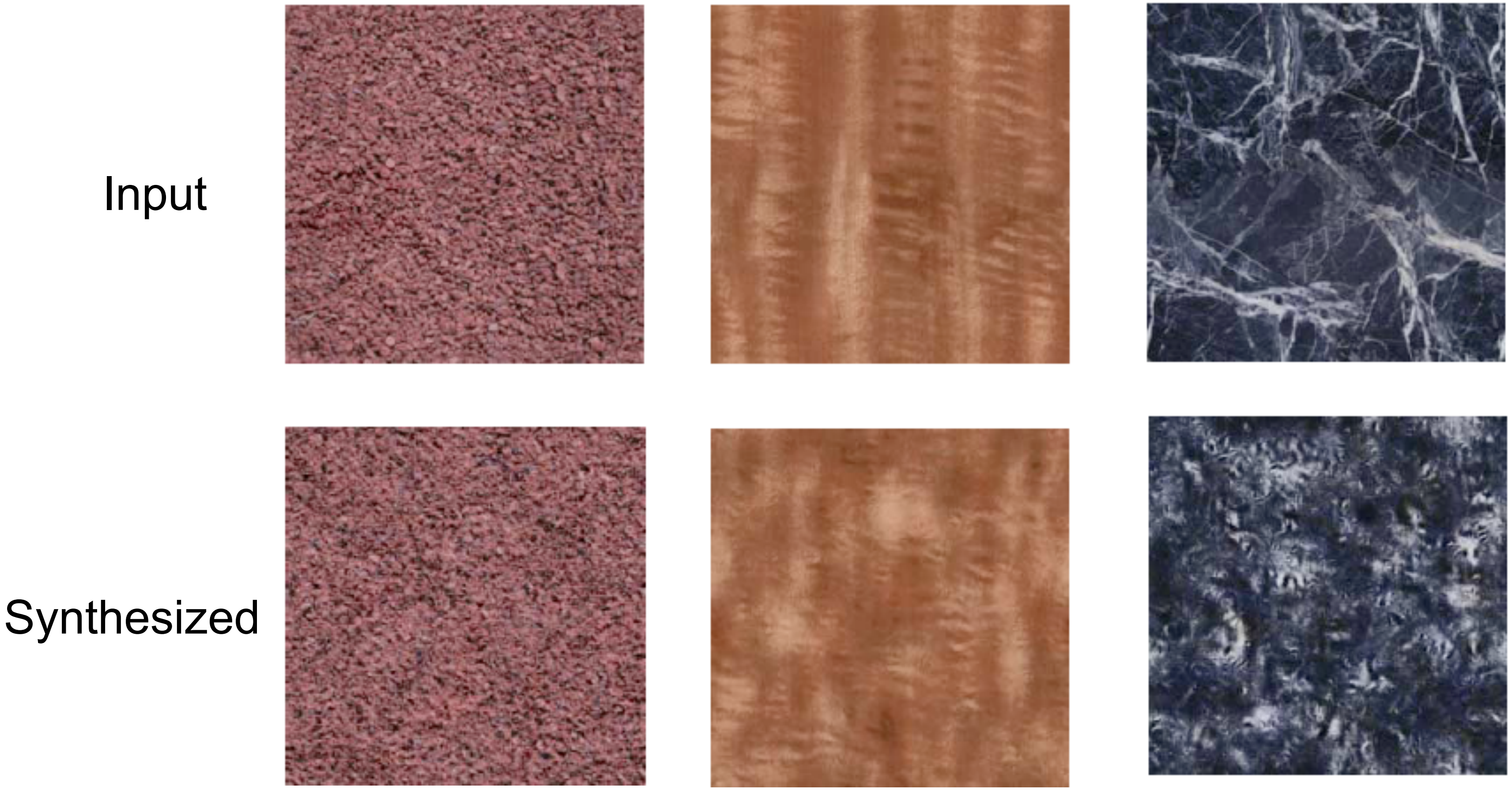




- D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95.*
- E. P. Simoncelli and J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. In *ICIP 1998.*

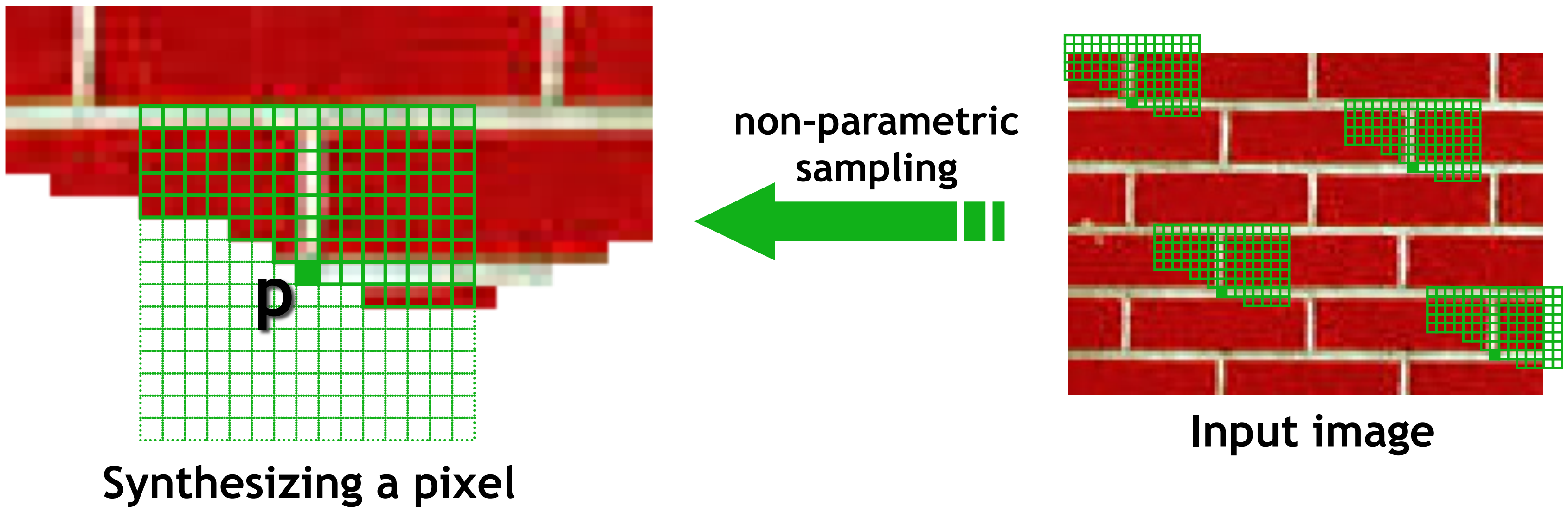
One idea: Build Probability Distributions

But it (usually) doesn't work

- Probability distributions are hard to model well



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

Efros and Leung 1999 SIGGRAPH

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

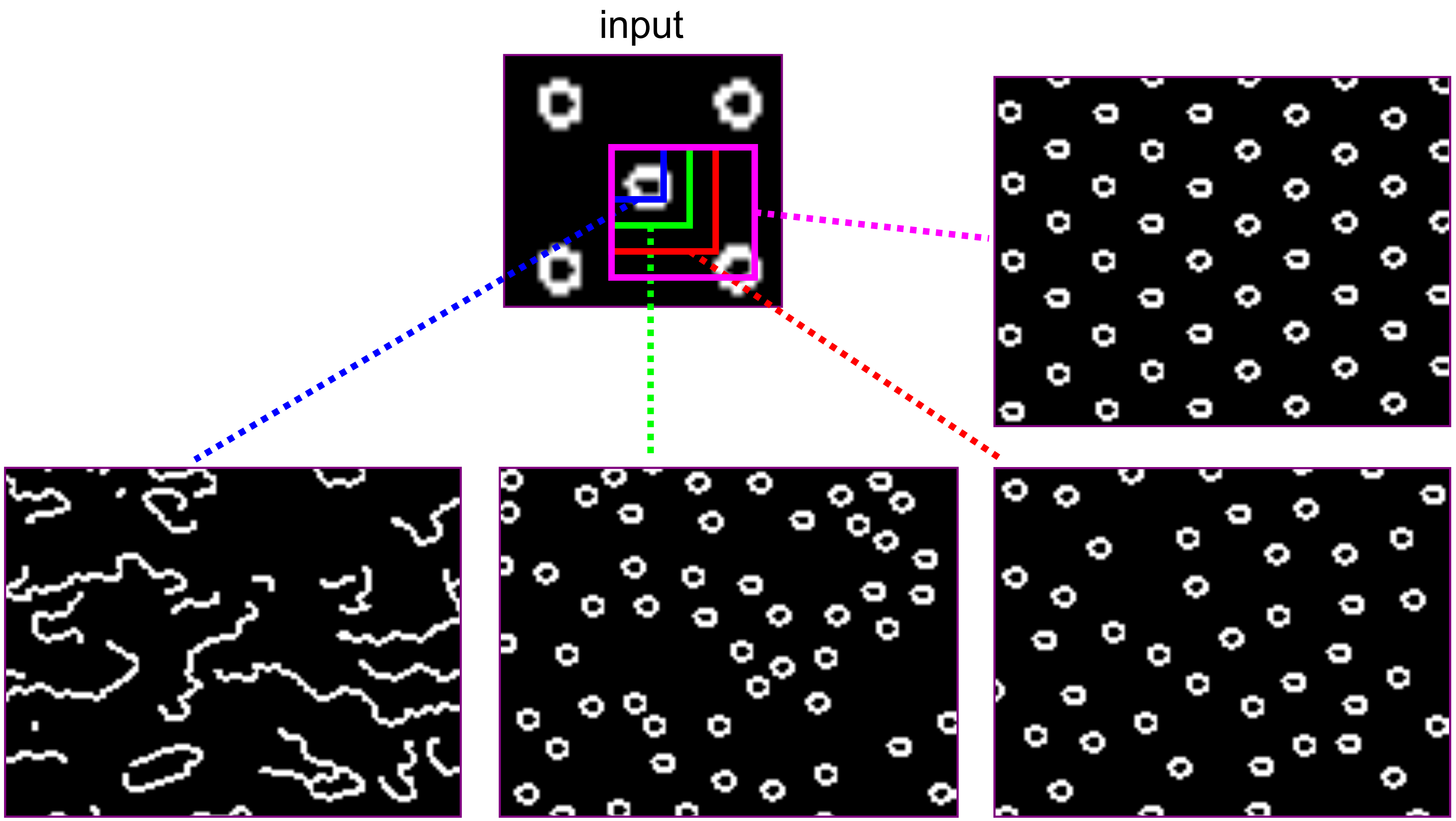
“I spent an interesting evening recently with a grain of salt.”

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

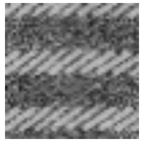
Details

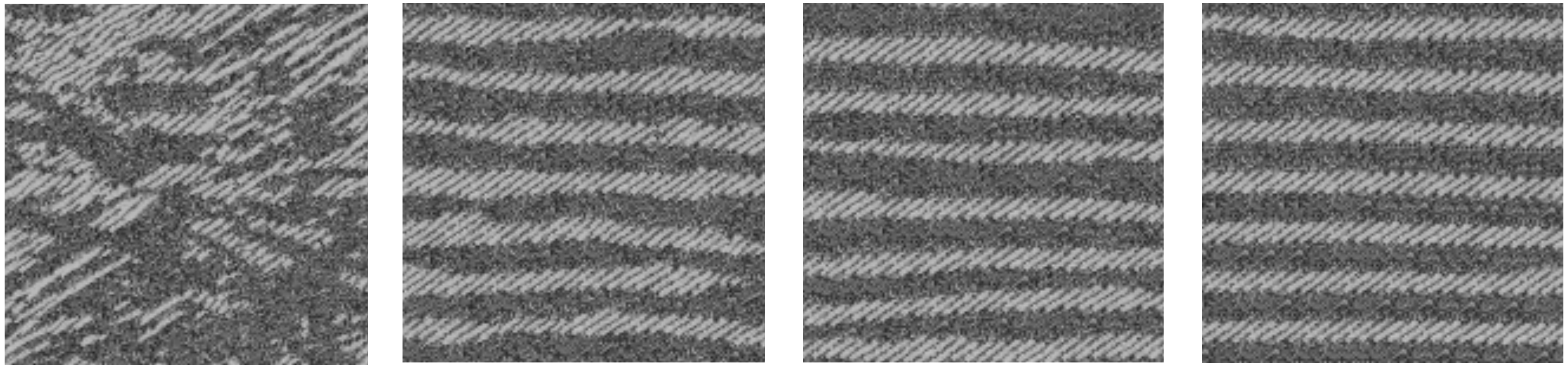
- How to match patches?
  - Gaussian-weighted SSD (more emphasis on nearby pixels)
- What order to fill in new pixels?
  - “Onion skin” order: pixels with most neighbors are synthesized first
  - To synthesize from scratch, start with a randomly selected small patch from the source texture
- How big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

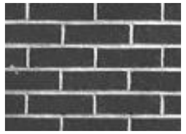
Size of Neighborhood Window

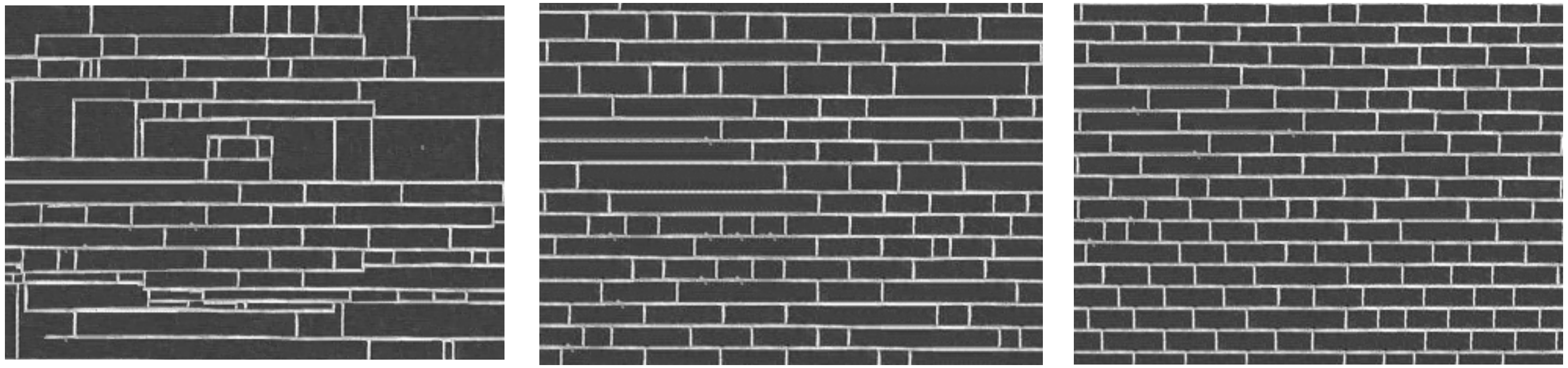


Varying Window Size









Increasing window size

Texture synthesis algorithm

While image not filled

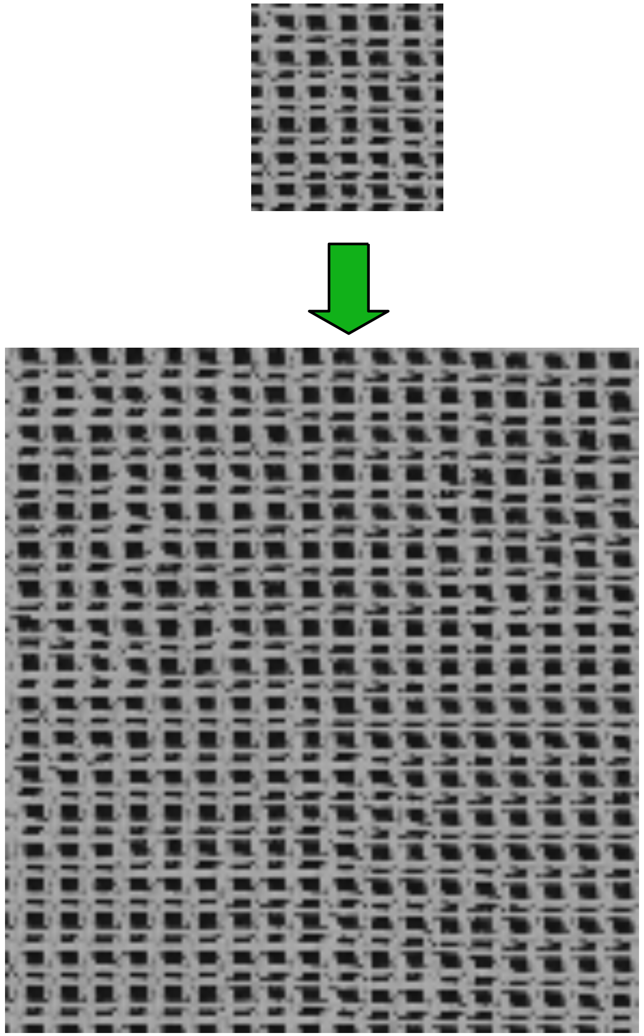
1. Get unfilled pixels with filled neighbors, sorted by number of filled neighbors

2. For each pixel, get top N matches based on visible neighbors
   - Patch Distance: Gaussian-weighted SSD

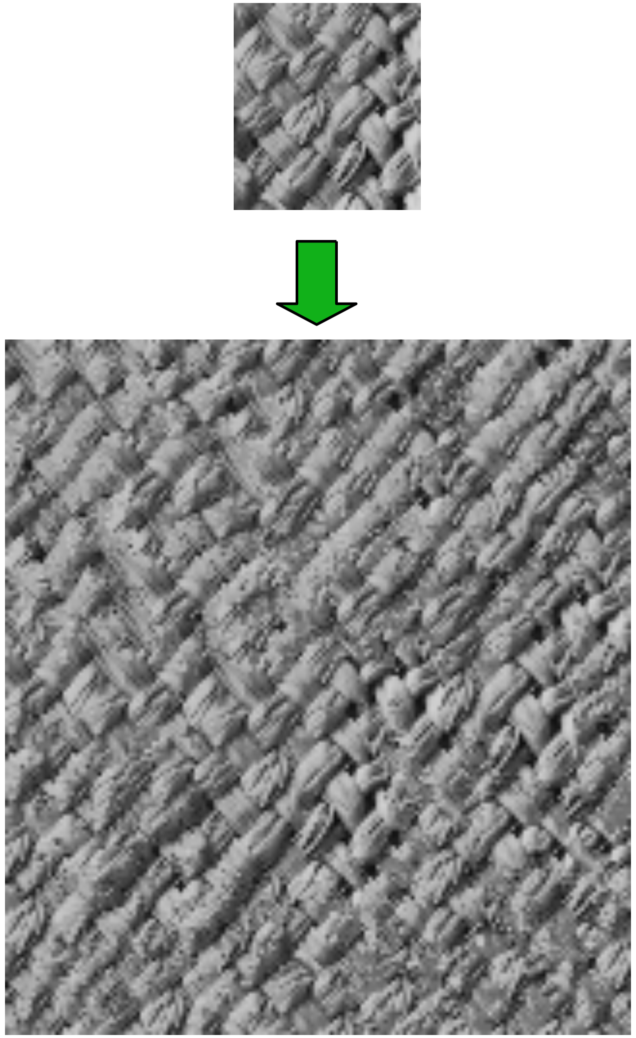
3. Randomly select one of the matches and copy pixel from it

Synthesis Results

french canvas

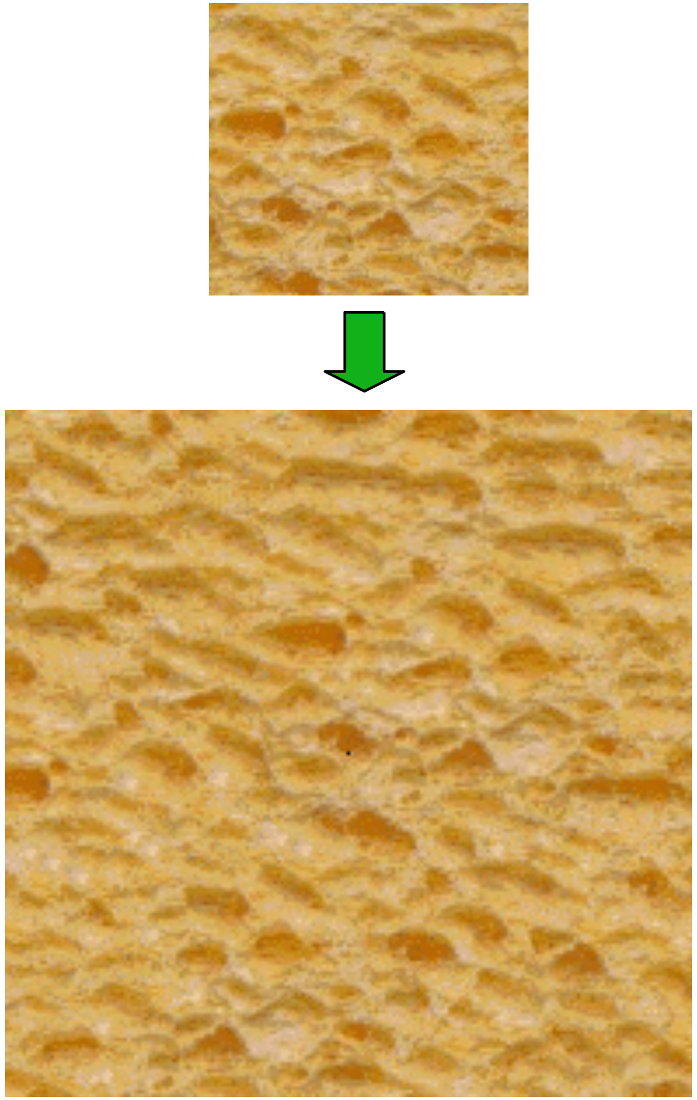


rafia weave

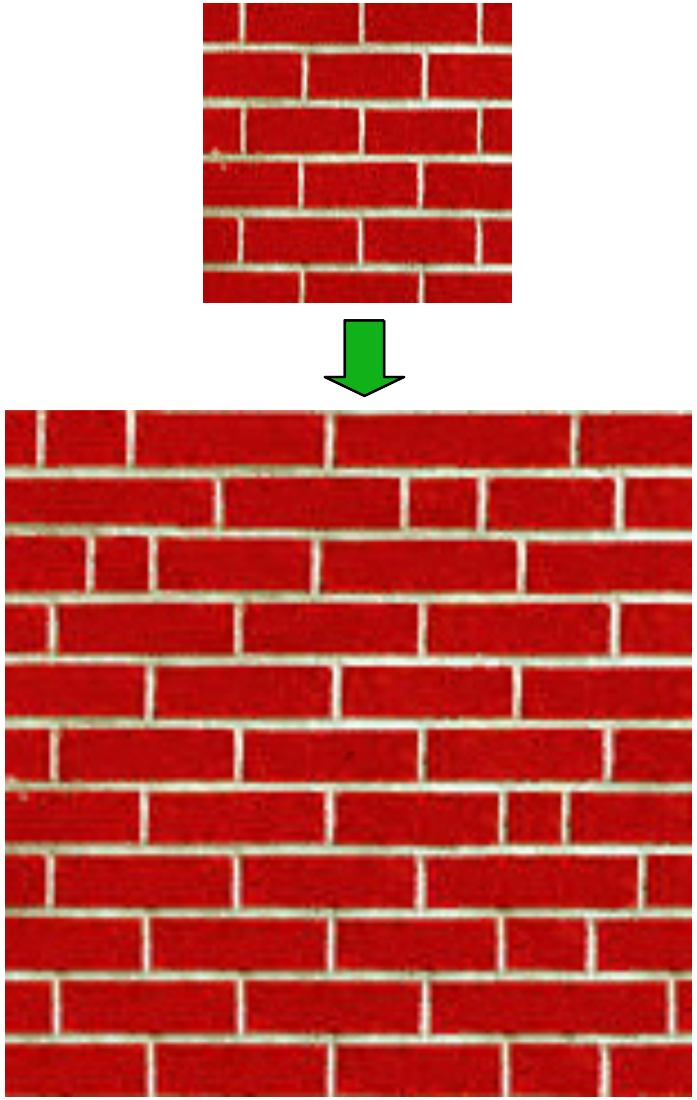


More Results

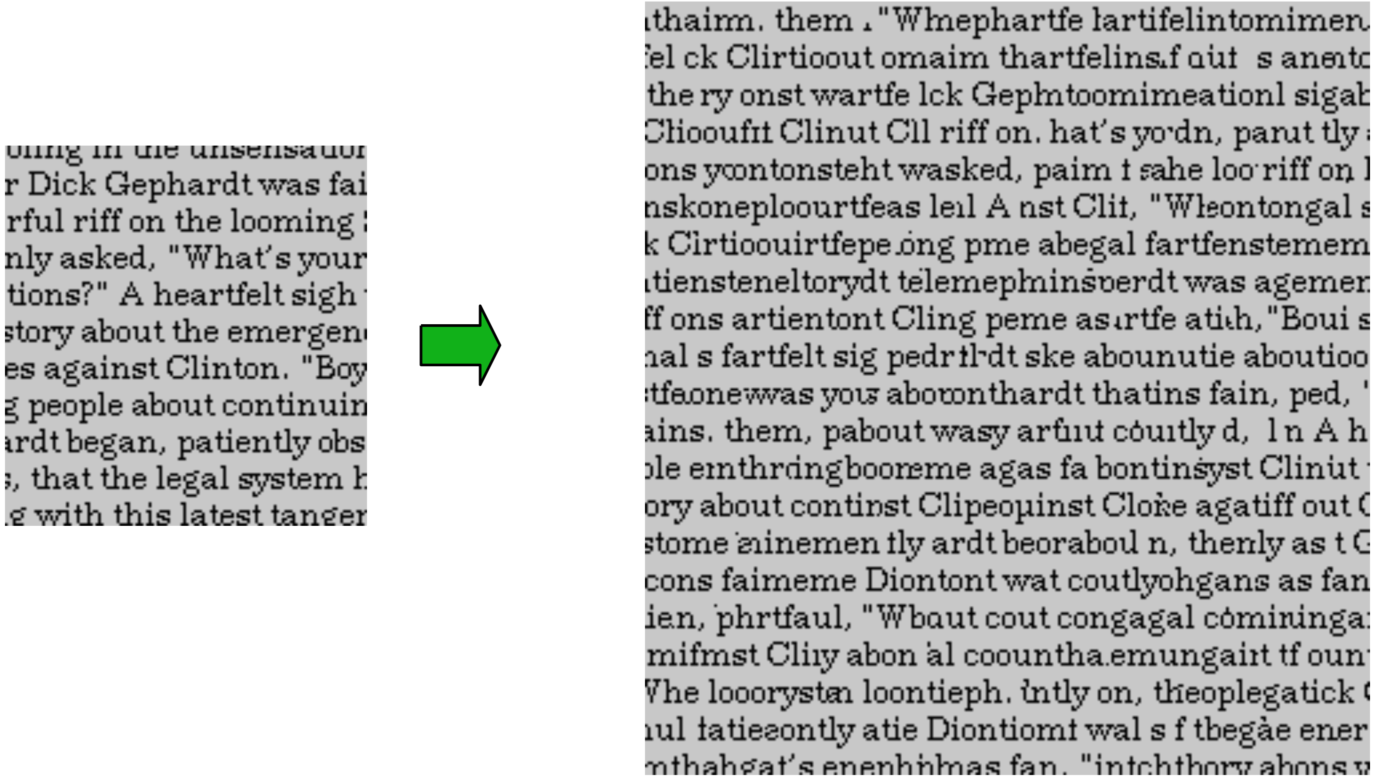
white bread



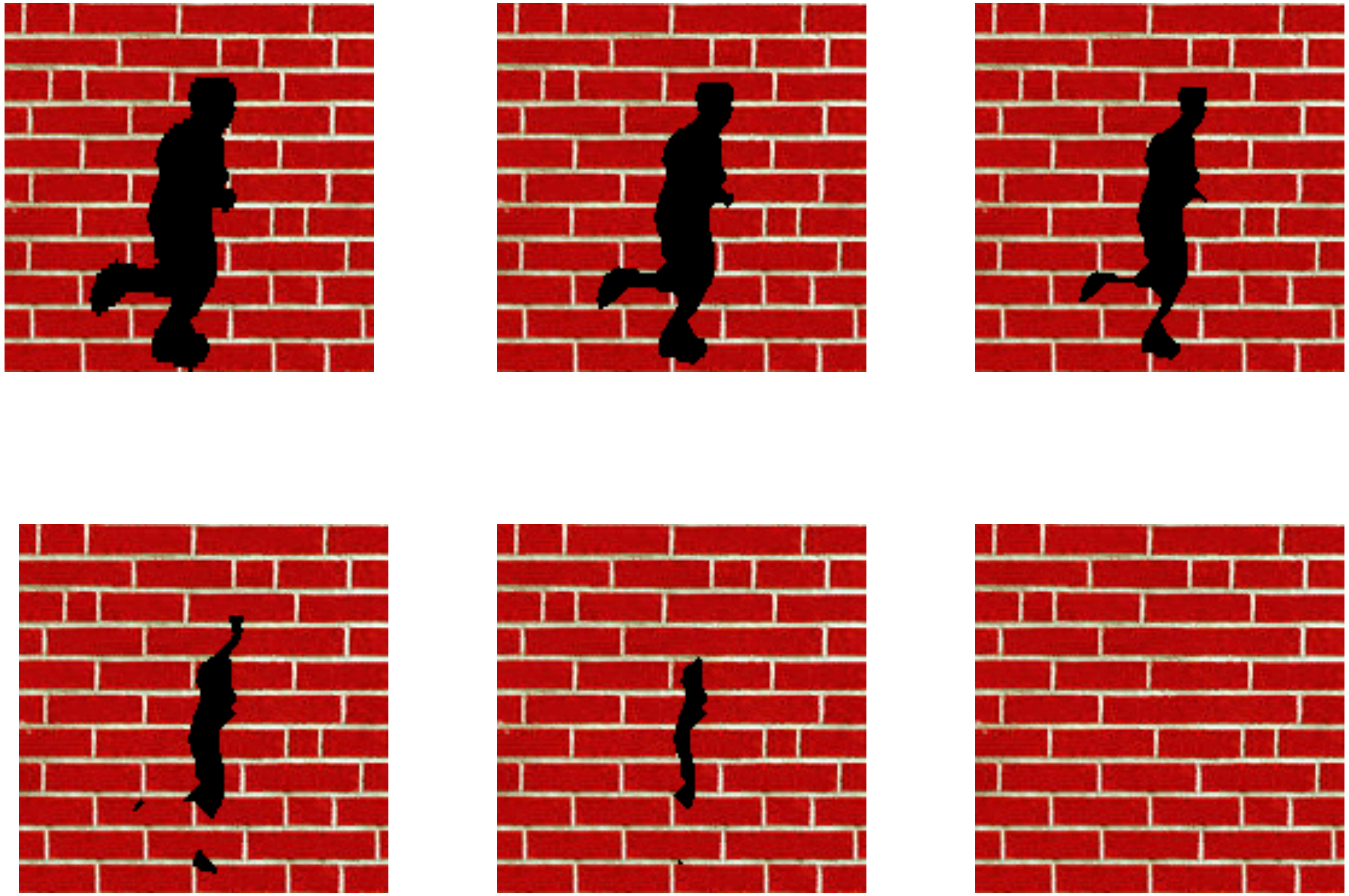
brick wall



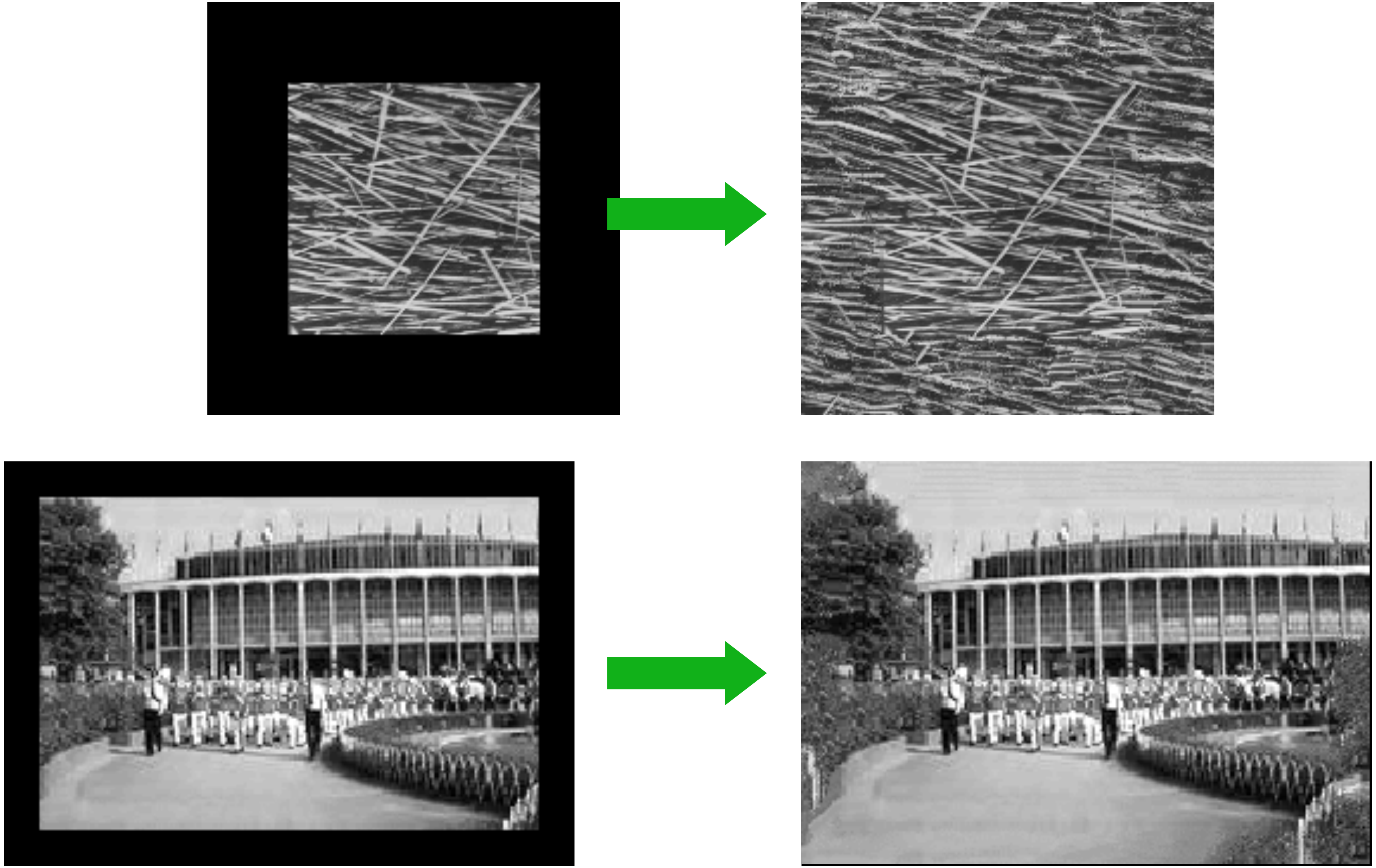
Homage to Shannon



Hole Filling



Extrapolation



In-painting natural scenes

Criminisi, Perez, and Toyama. "Object Removal by Exemplar-based Inpainting," Proc. CVPR, 2003.

Key idea: Filling order matters

In-painting Result

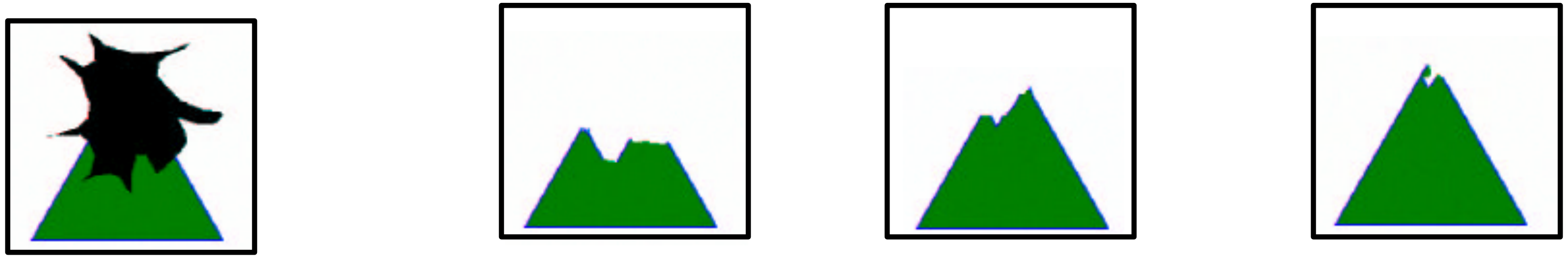


Image with Hole

Raster-Scan Order

Onion-Peel
(Concentric Layers)

Gradient-Sensitive
Order

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

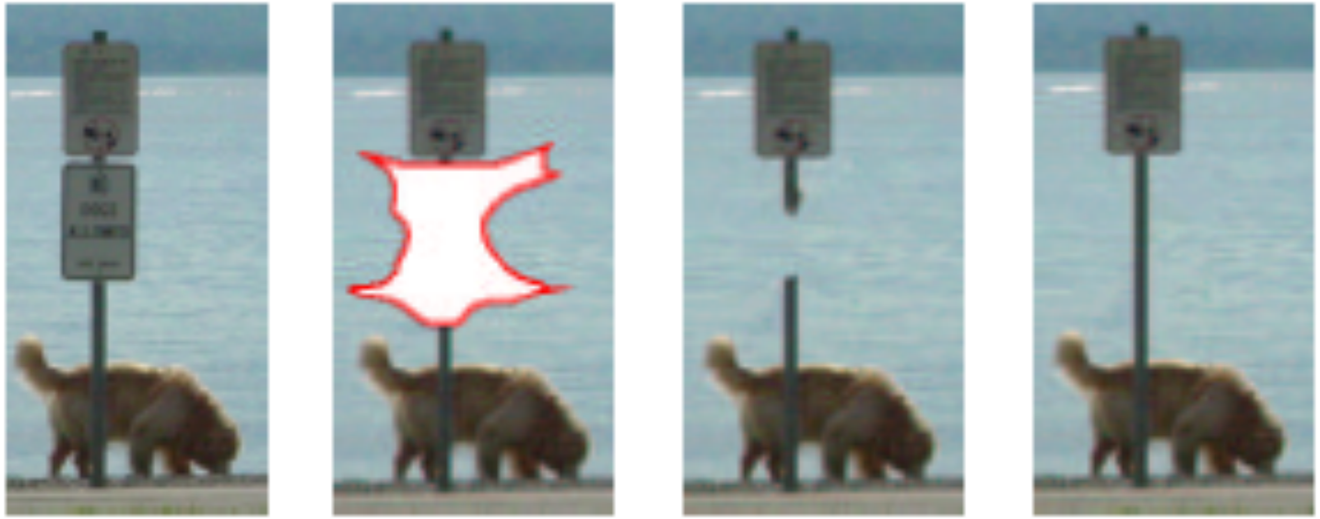
Filling order

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](#)," Proc. CVPR, 2003.

Comparison

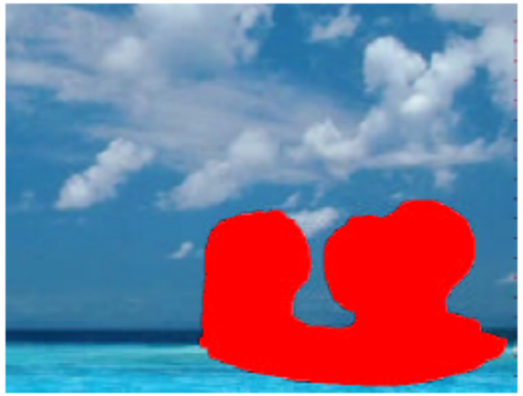


Original With Hole Onion-Ring Fill Criminisi

Criminisi, Perez, and Toyama. "[Object Removal by Exemplar-based Inpainting]," Proc. CVPR, 2003.

Comparison

a



b

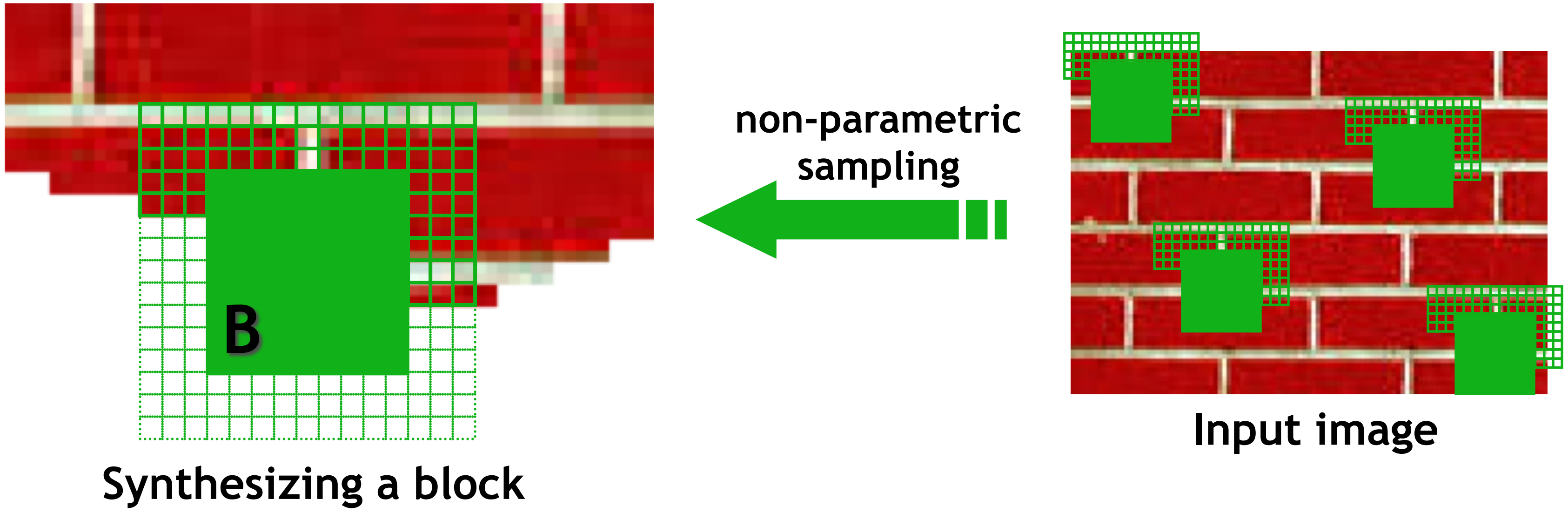
Concentric Layers

Gradient Sensitive

Summary

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

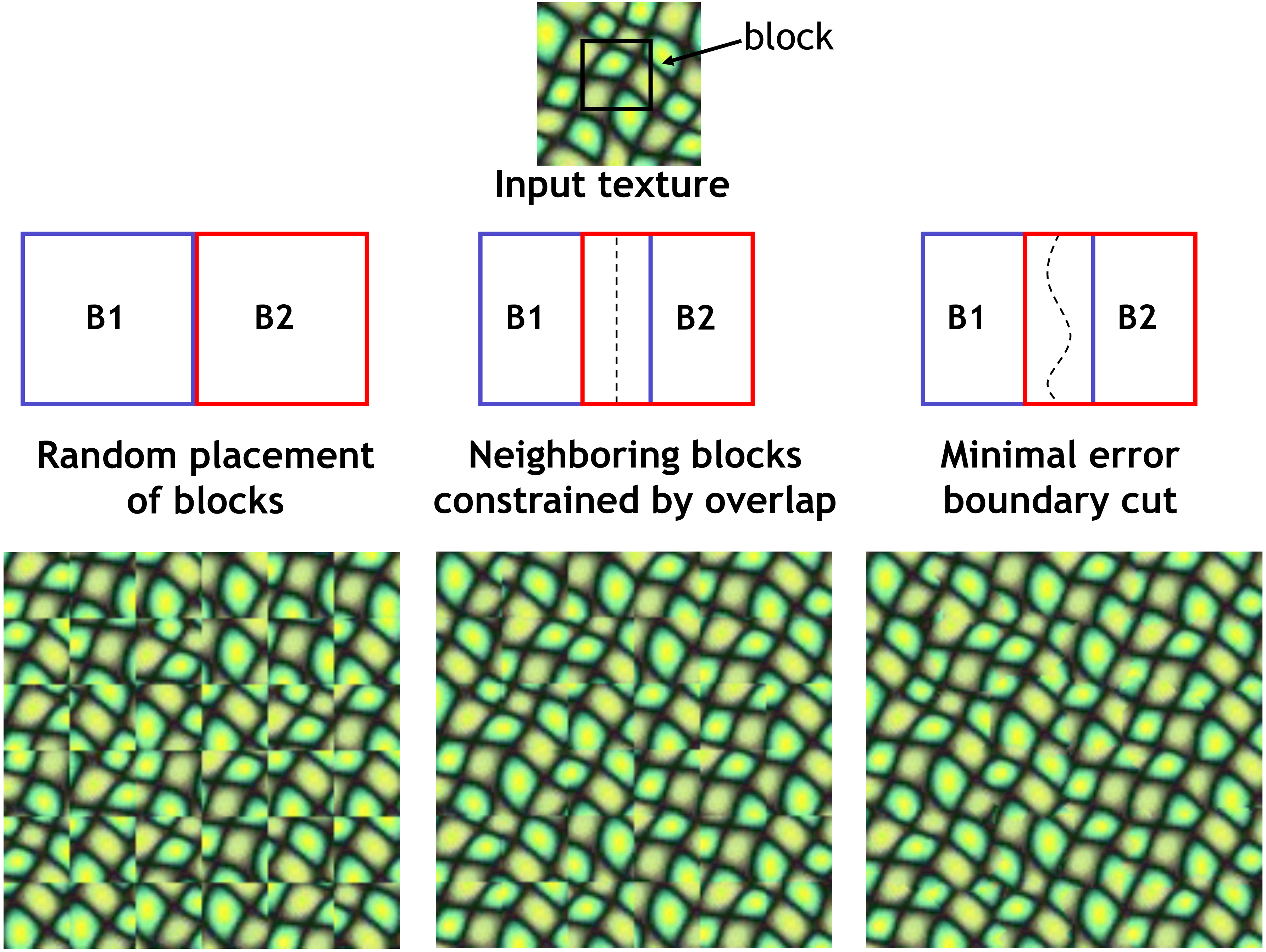
Image Quilting [Efros & Freeman 2001]



- <u>Observation:</u> neighbor pixels are highly correlated

<u>Idea:</u> unit of synthesis = block

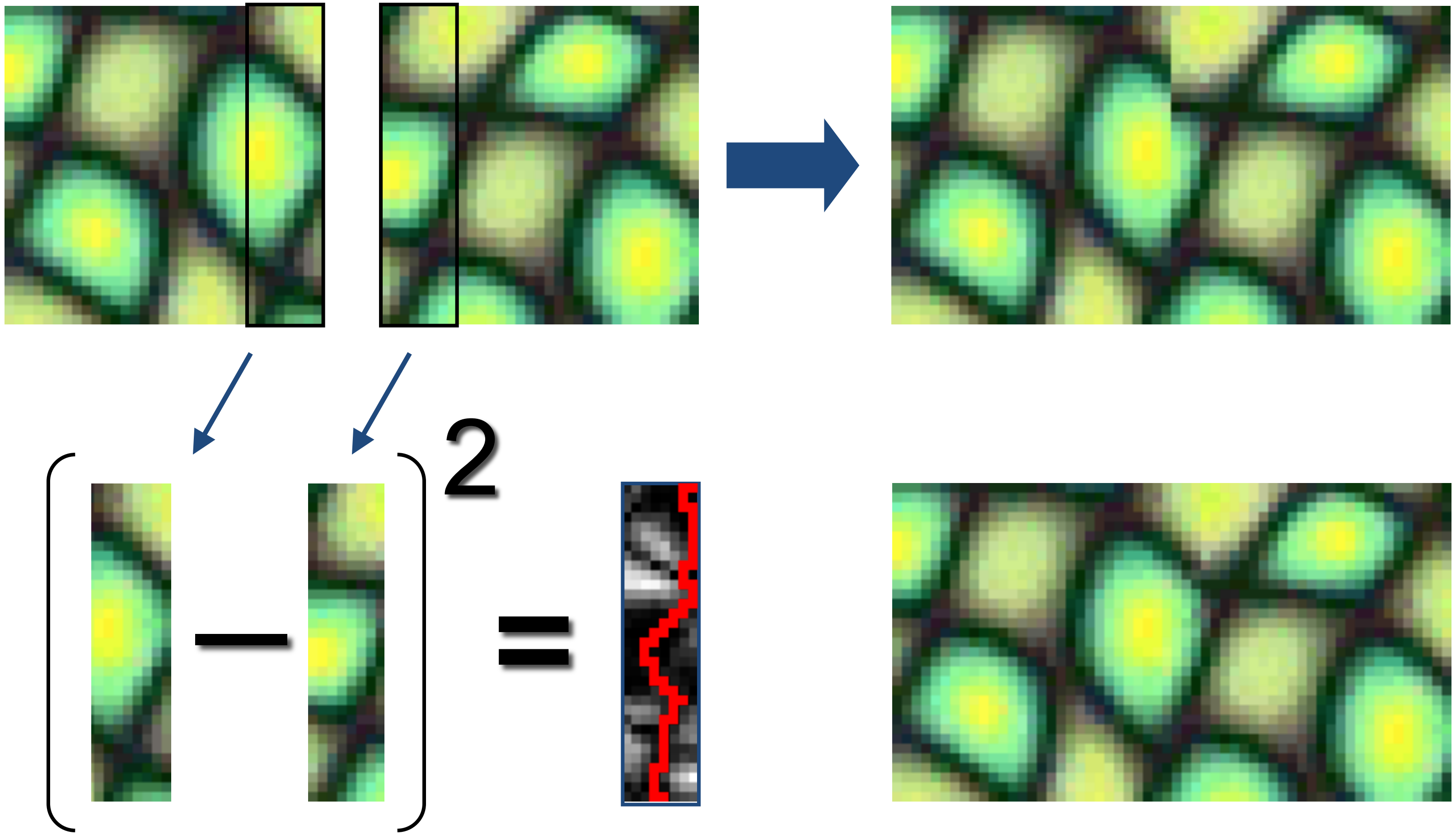
- Exactly the same but now we want P(**B**|N(**B**))
- Much faster: synthesize all pixels in a block at once


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

Minimal error boundary

overlapping blocks

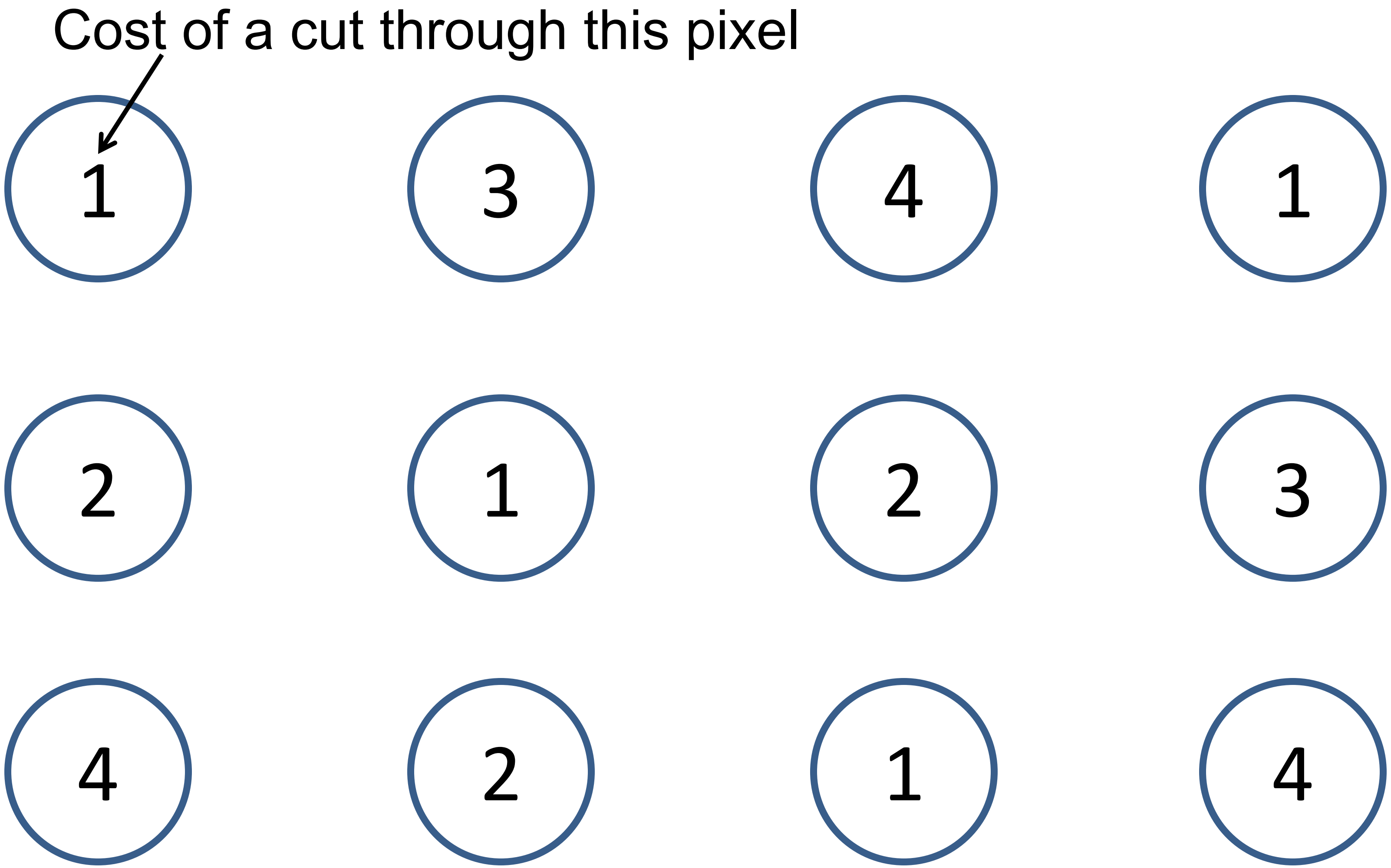
vertical boundary



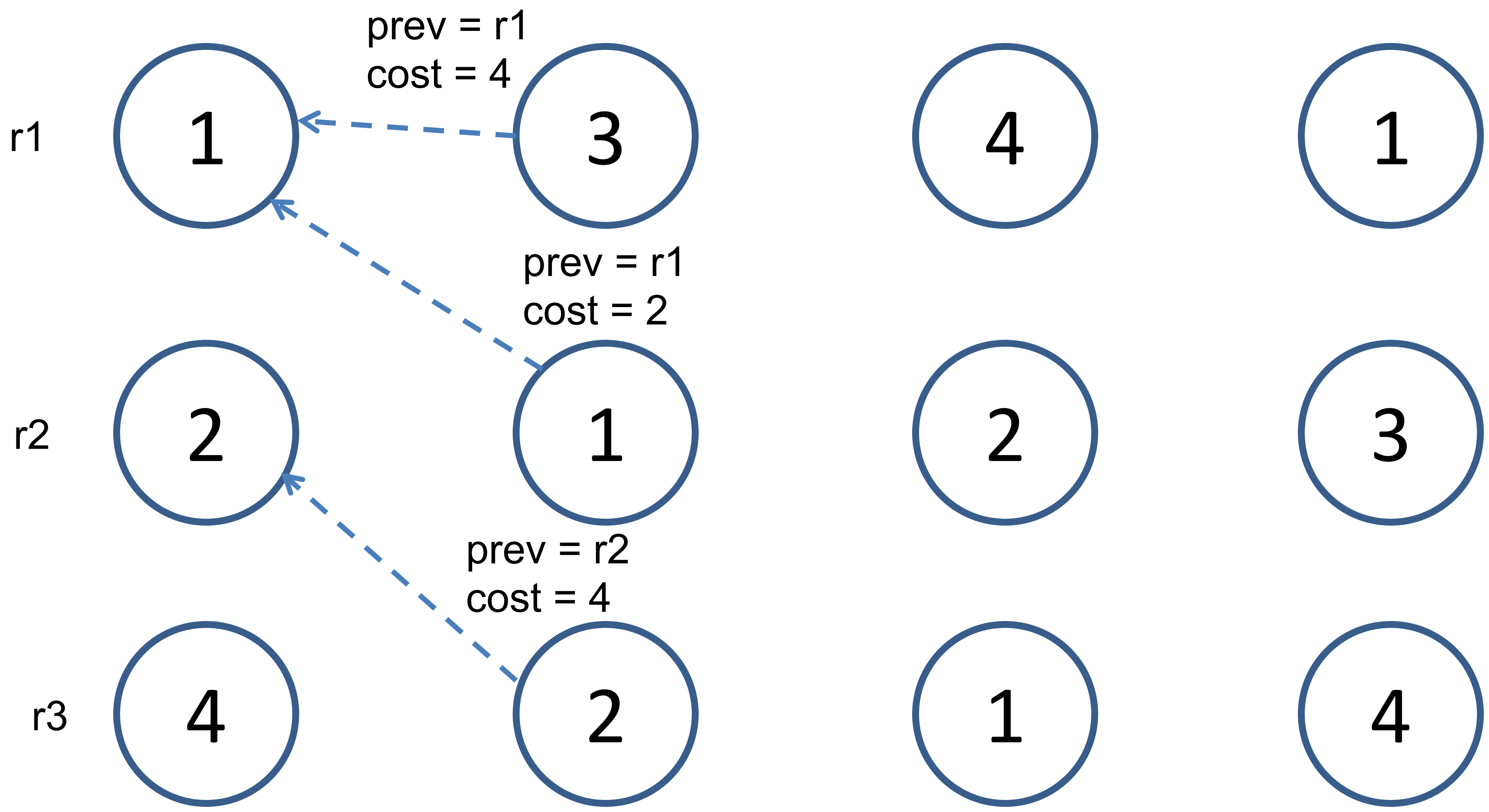
overlap error

min. error boundary

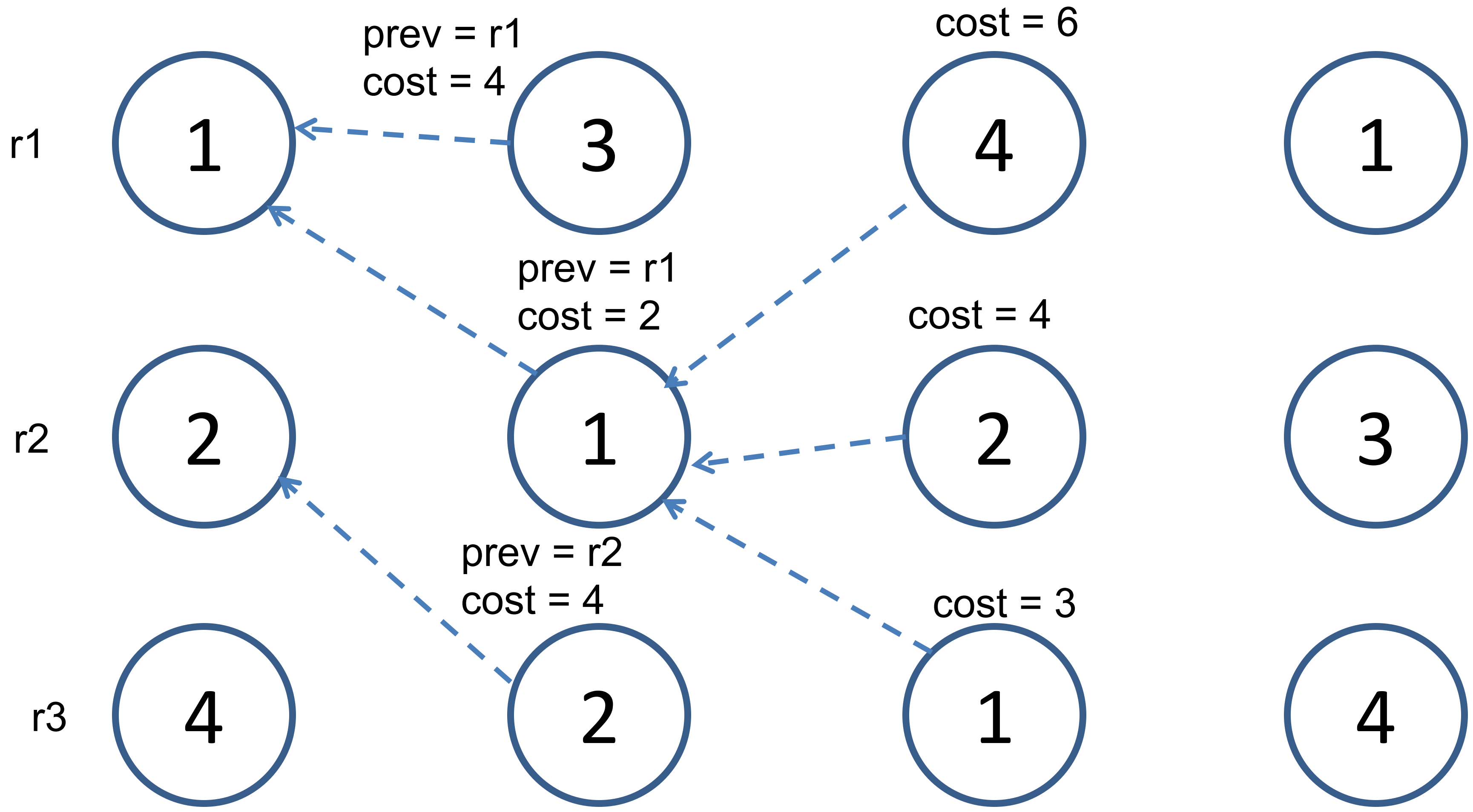
Solving for Minimum Cut Path



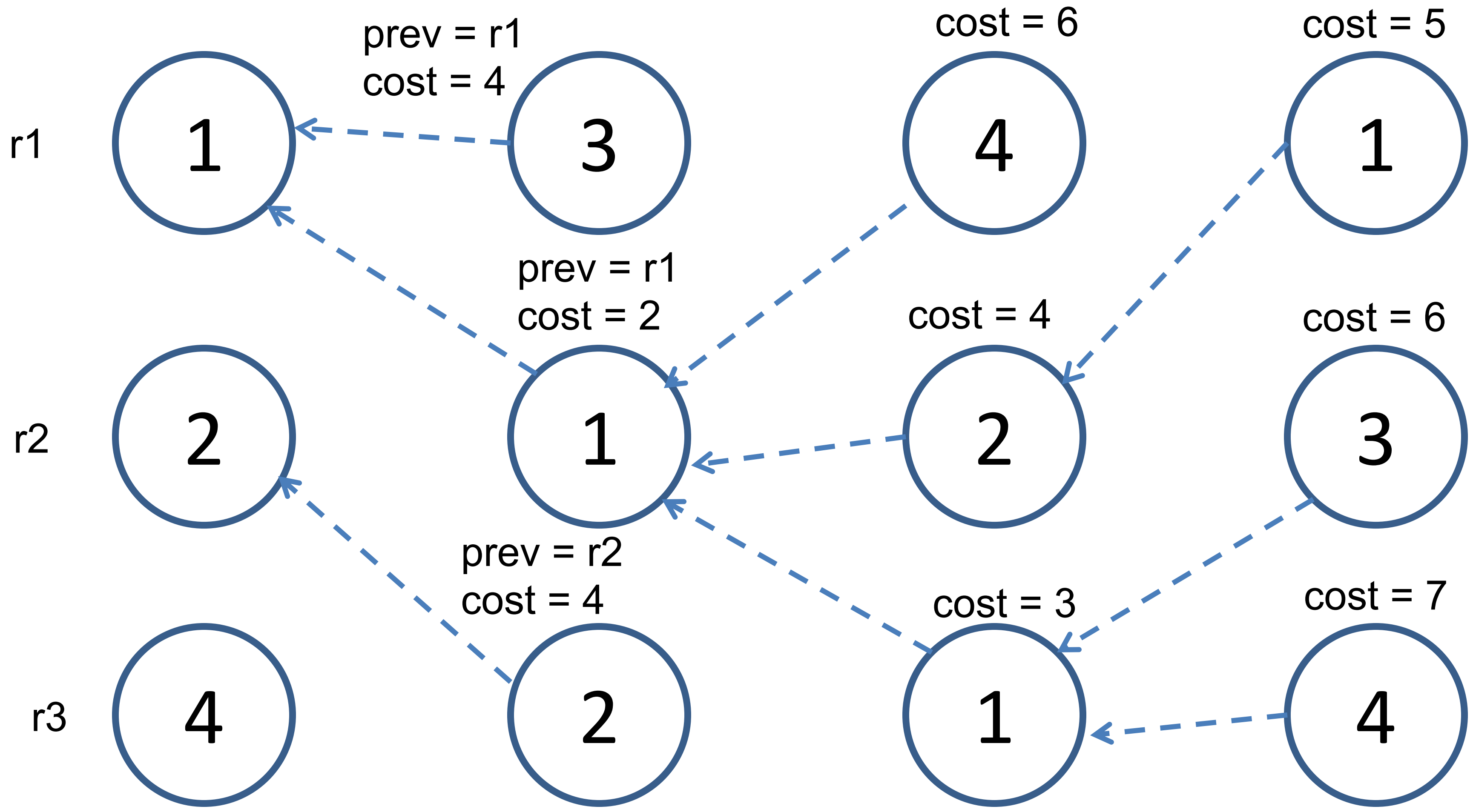
Solving for Minimum Cut Path



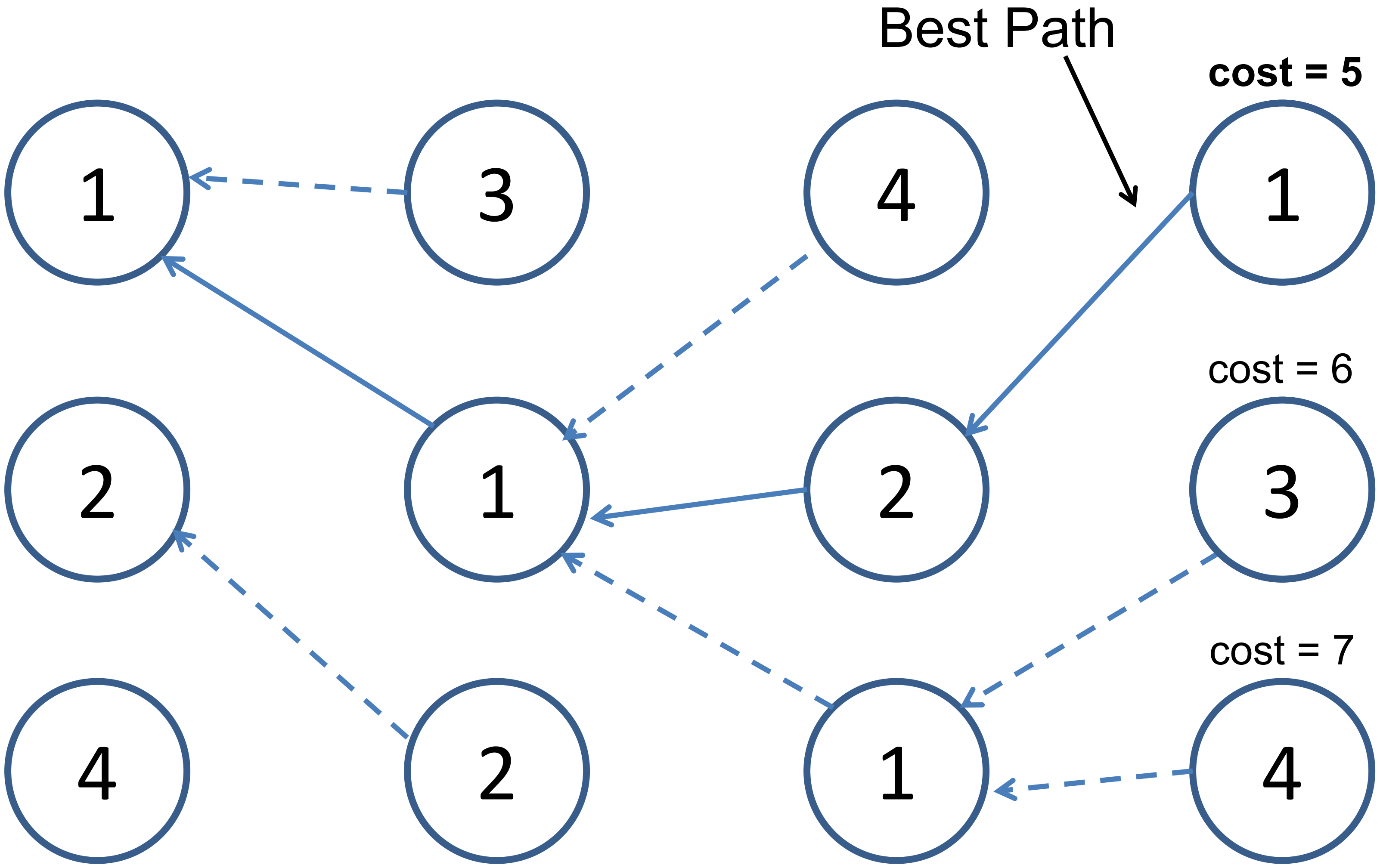
Solving for Minimum Cut Path



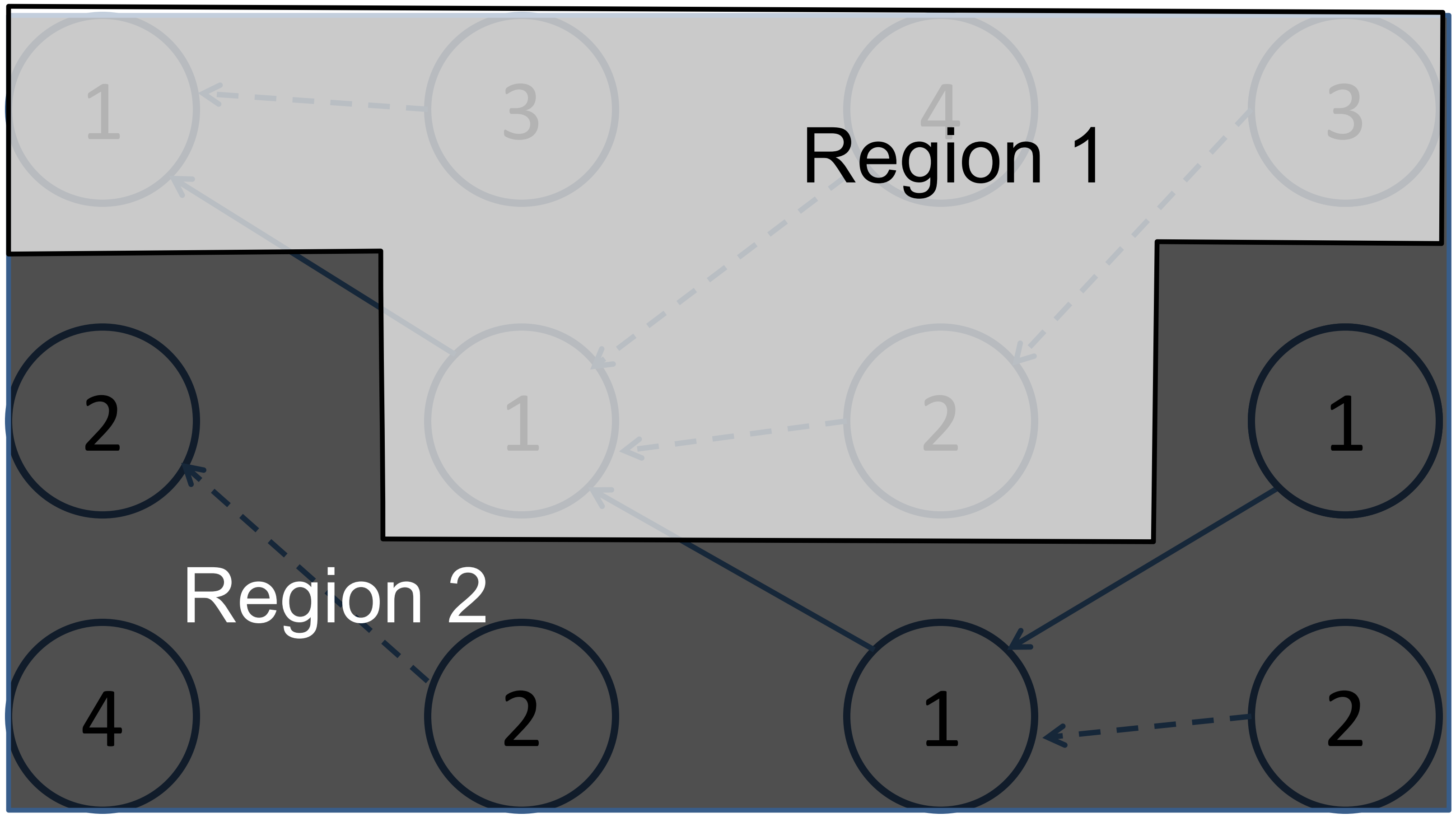
Solving for Minimum Cut Path



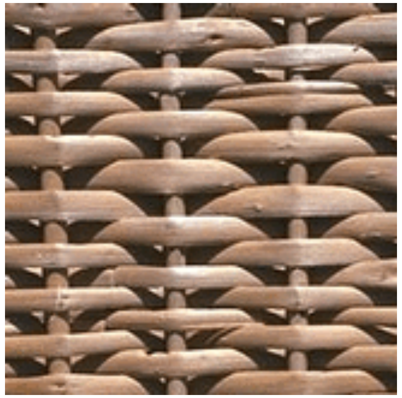
Solving for Minimum Cut Path

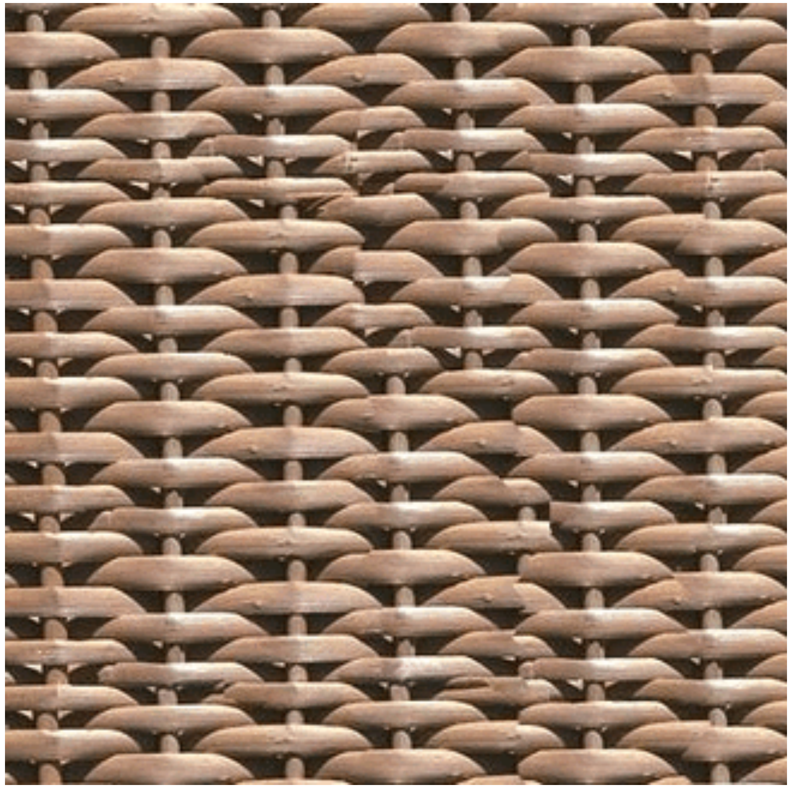


Solving for Minimum Cut Path

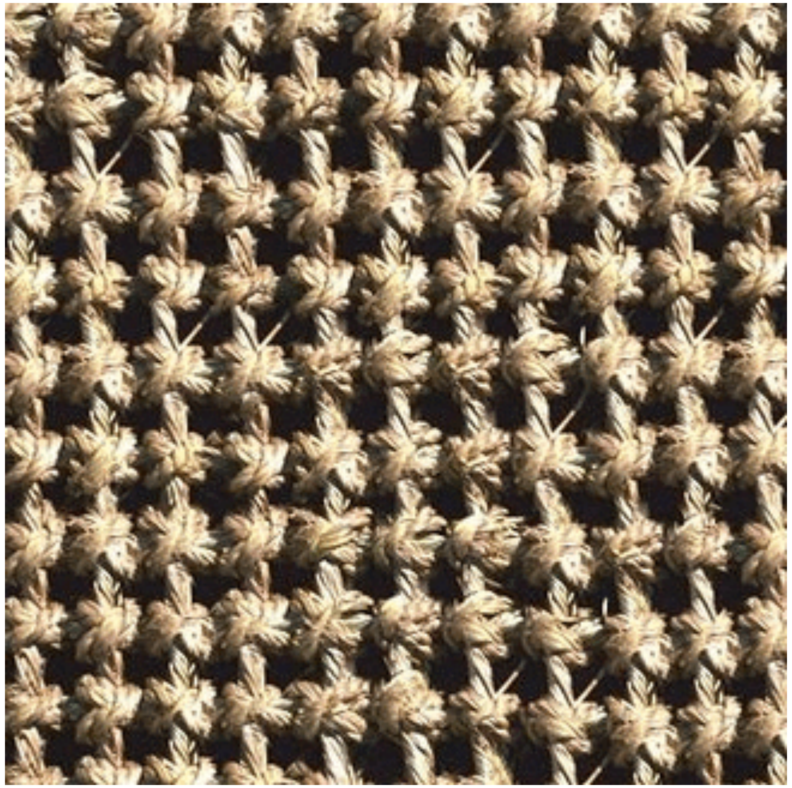


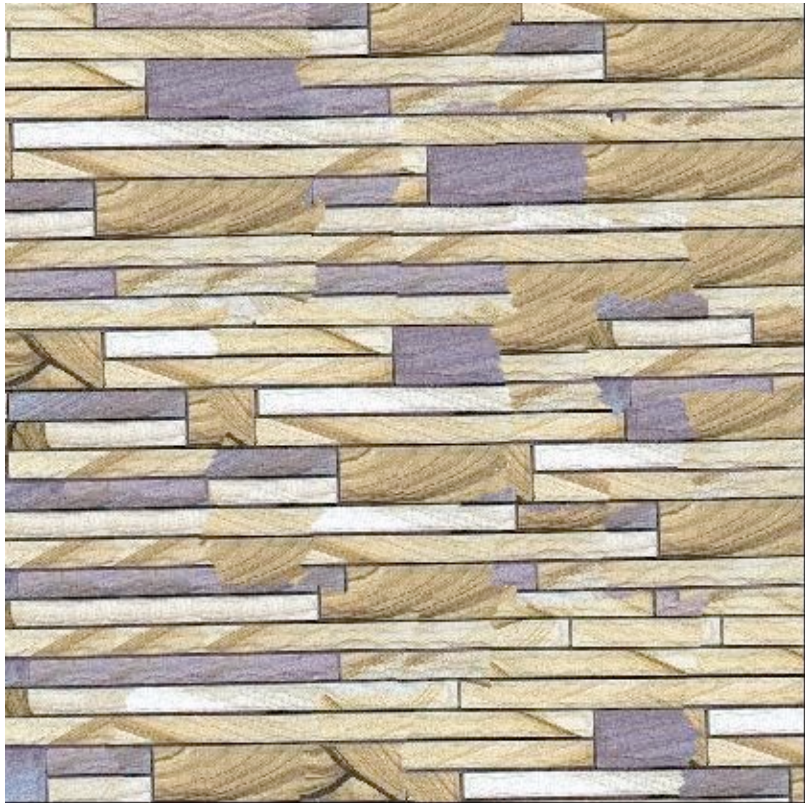
Mask Based on Best Path







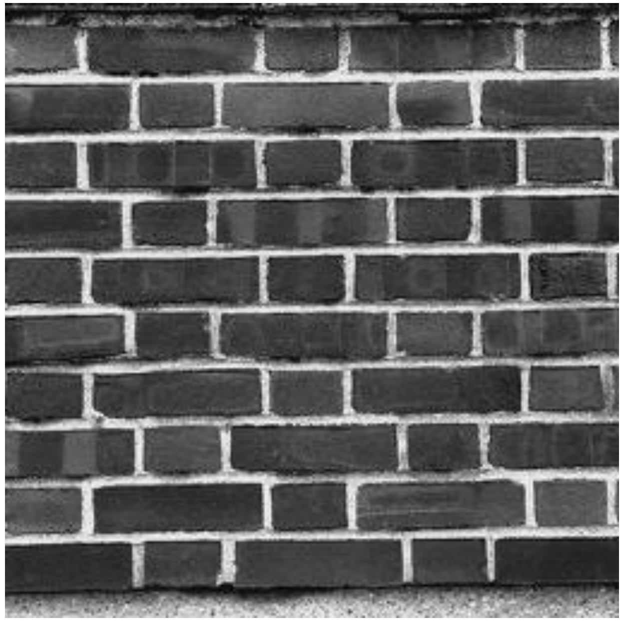




College Inn

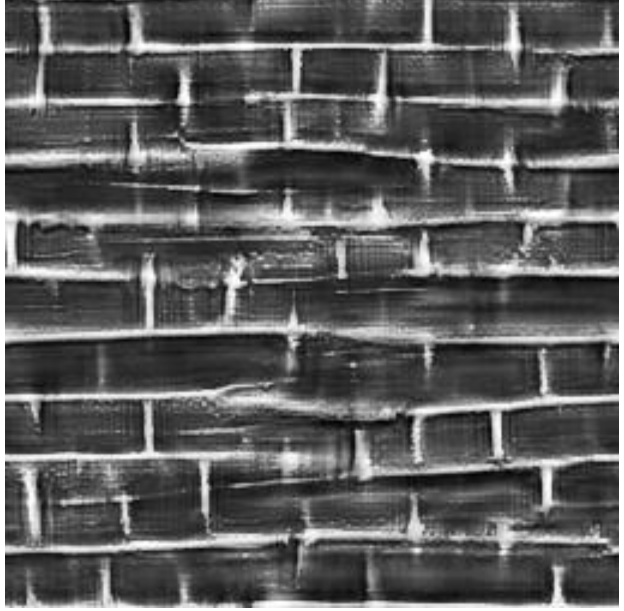

College Inn



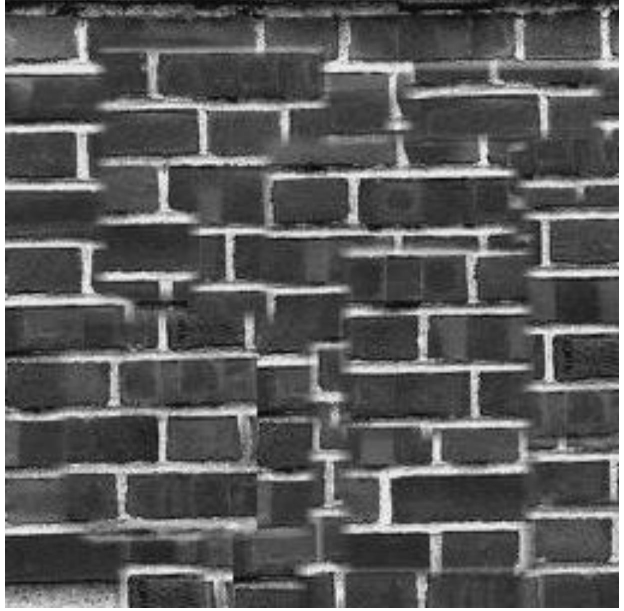




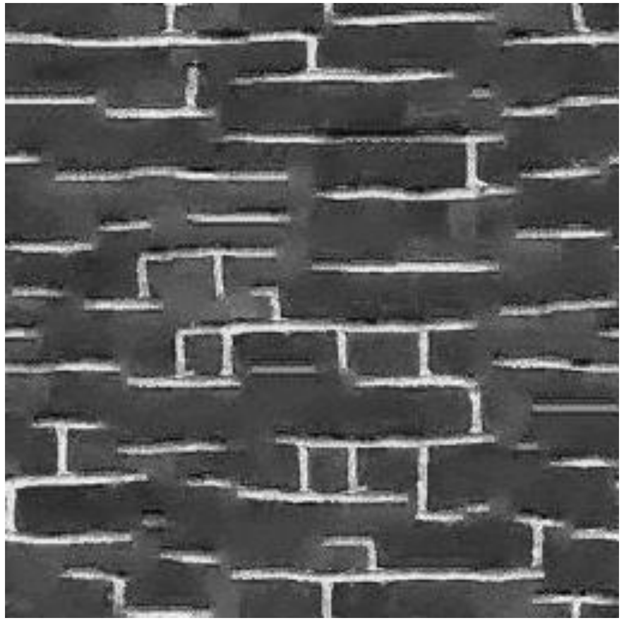
input image



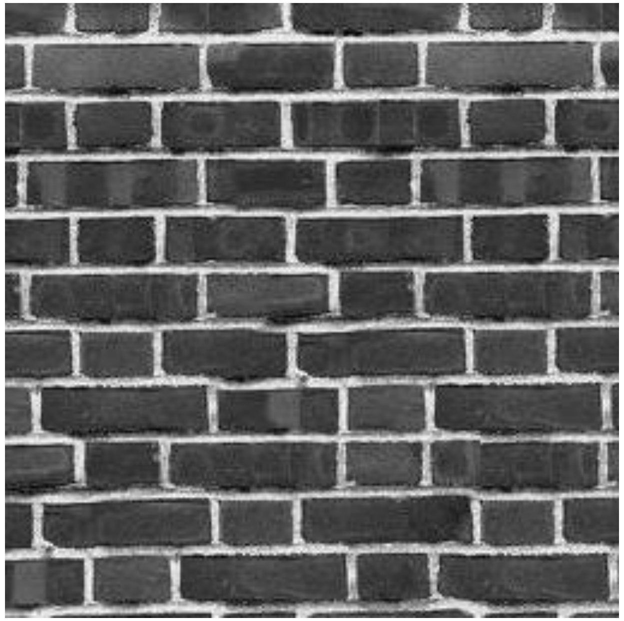
Portilla & Simoncelli



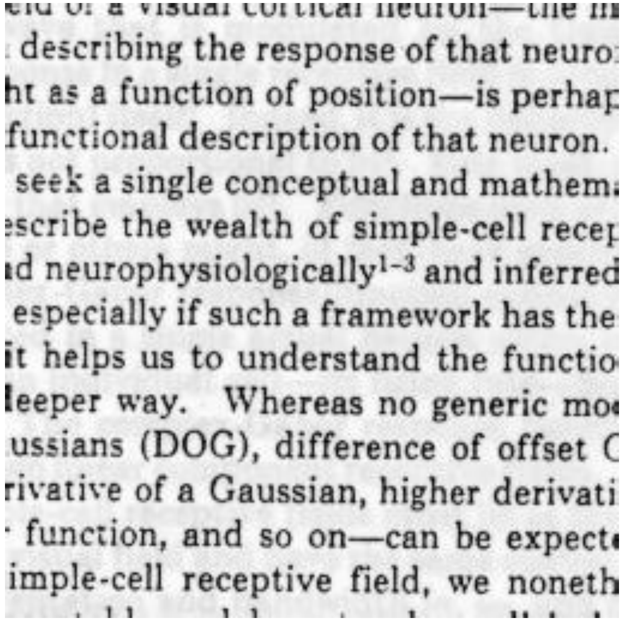
Xu, Guo & Shum



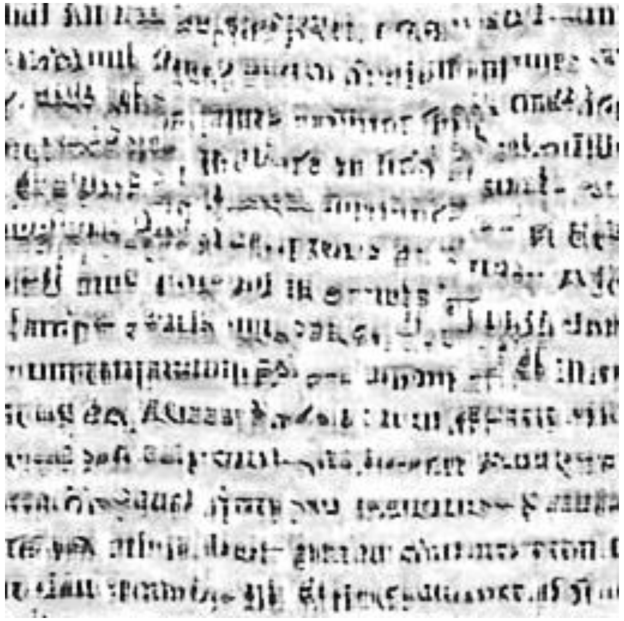
Wei & Levoy



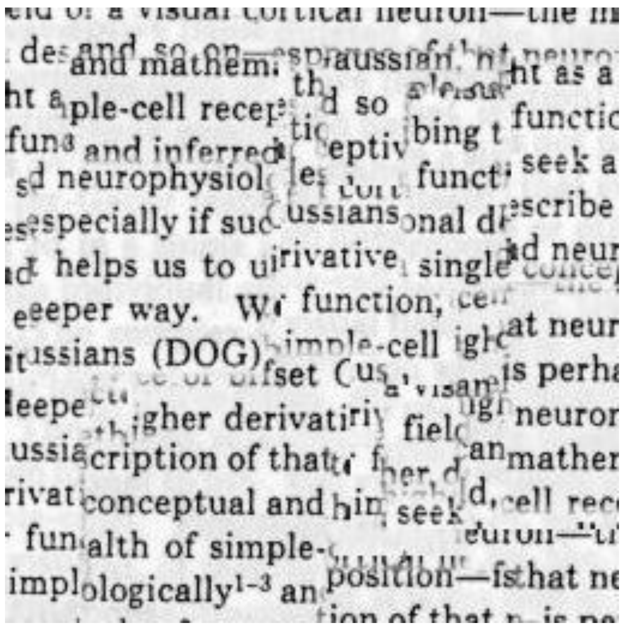
Quilting



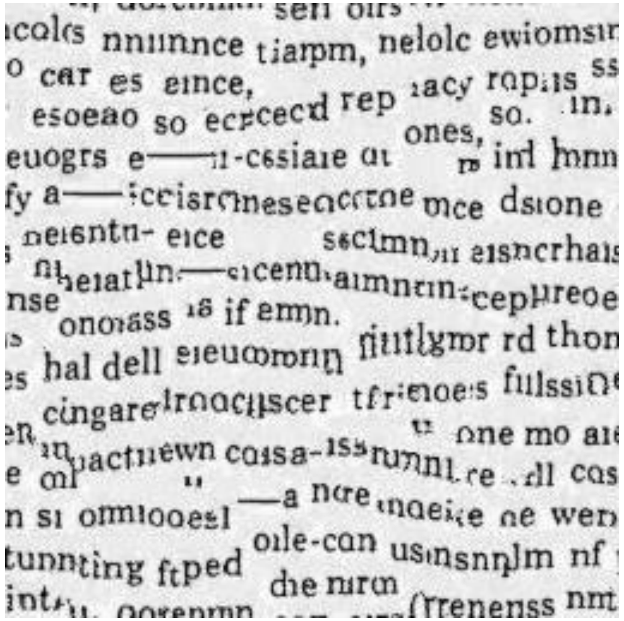
input image



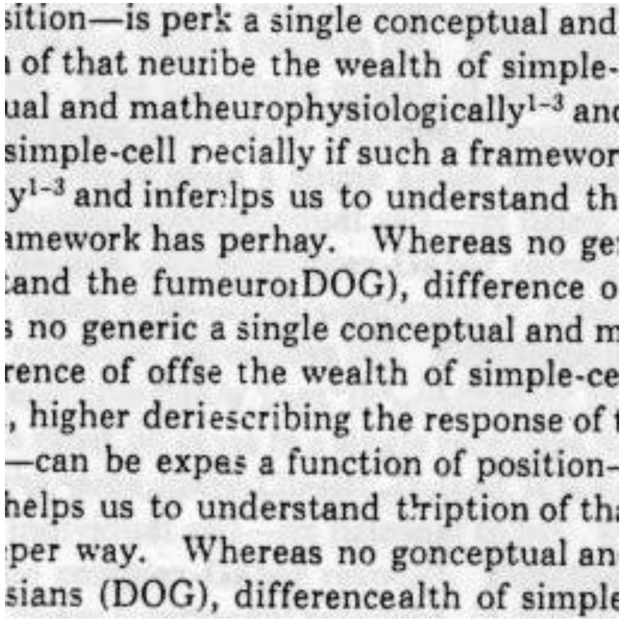
Portilla & Simoncelli



Xu, Guo & Shum

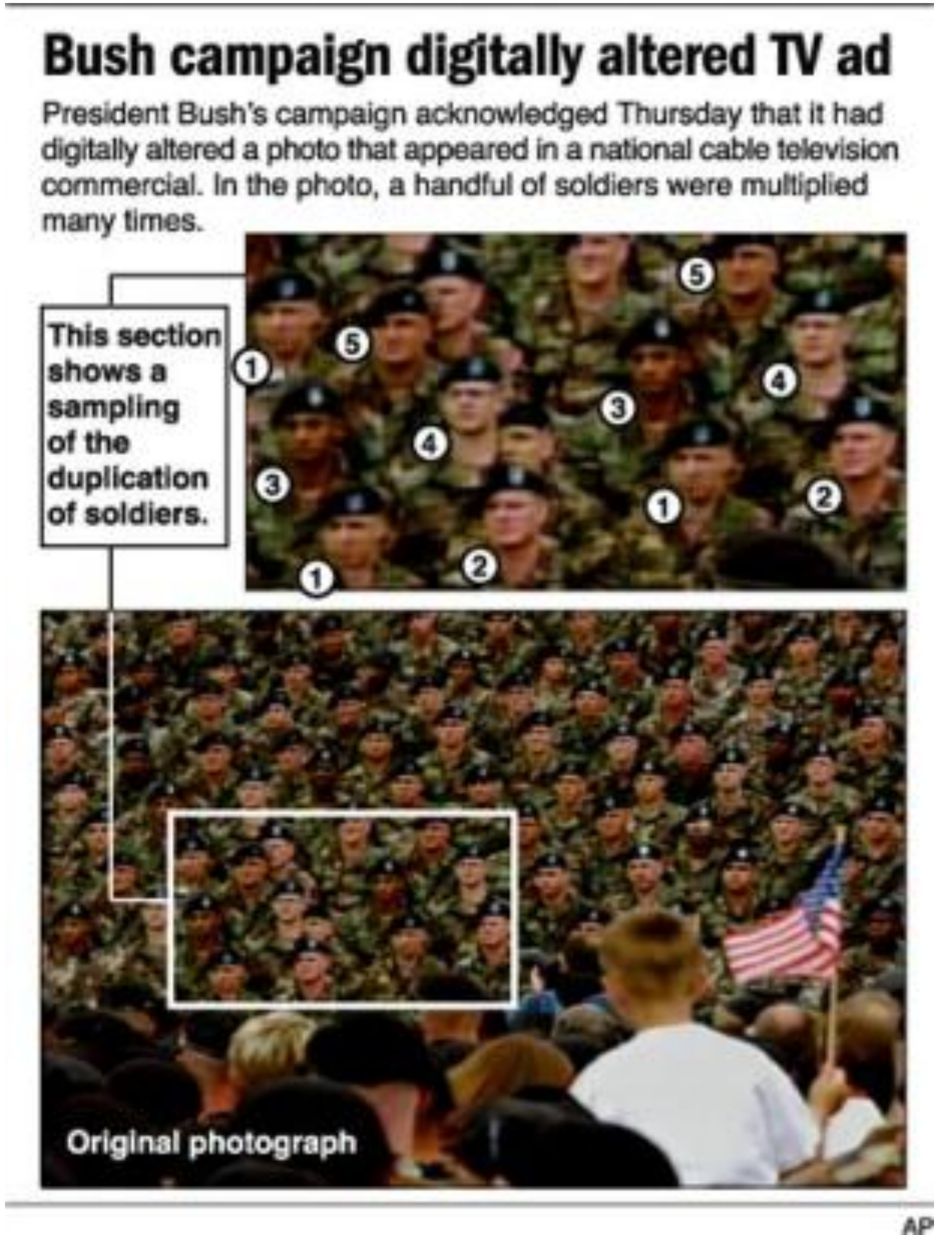


Wei & Levoy



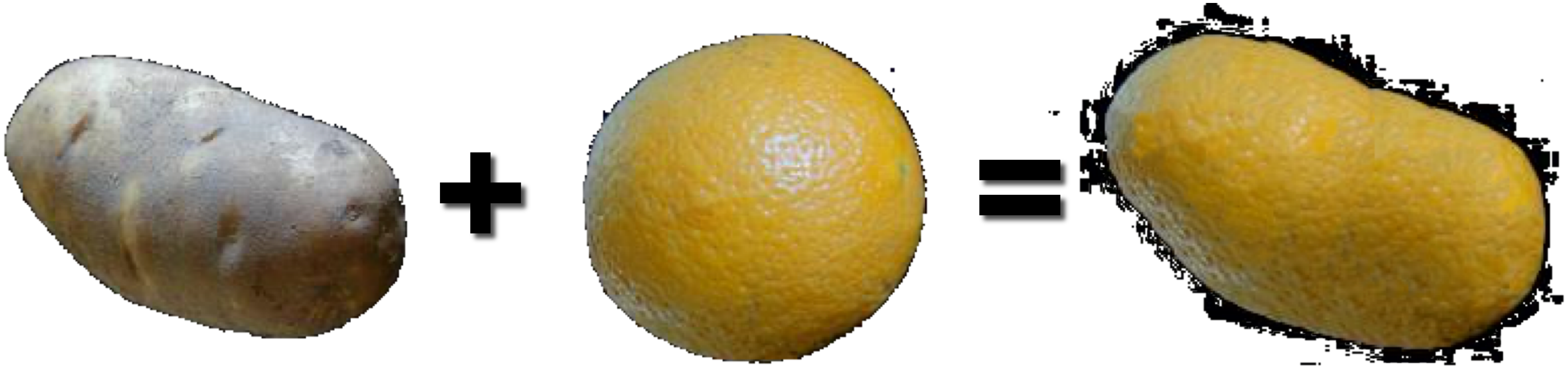
Quilting

Political Texture Synthesis

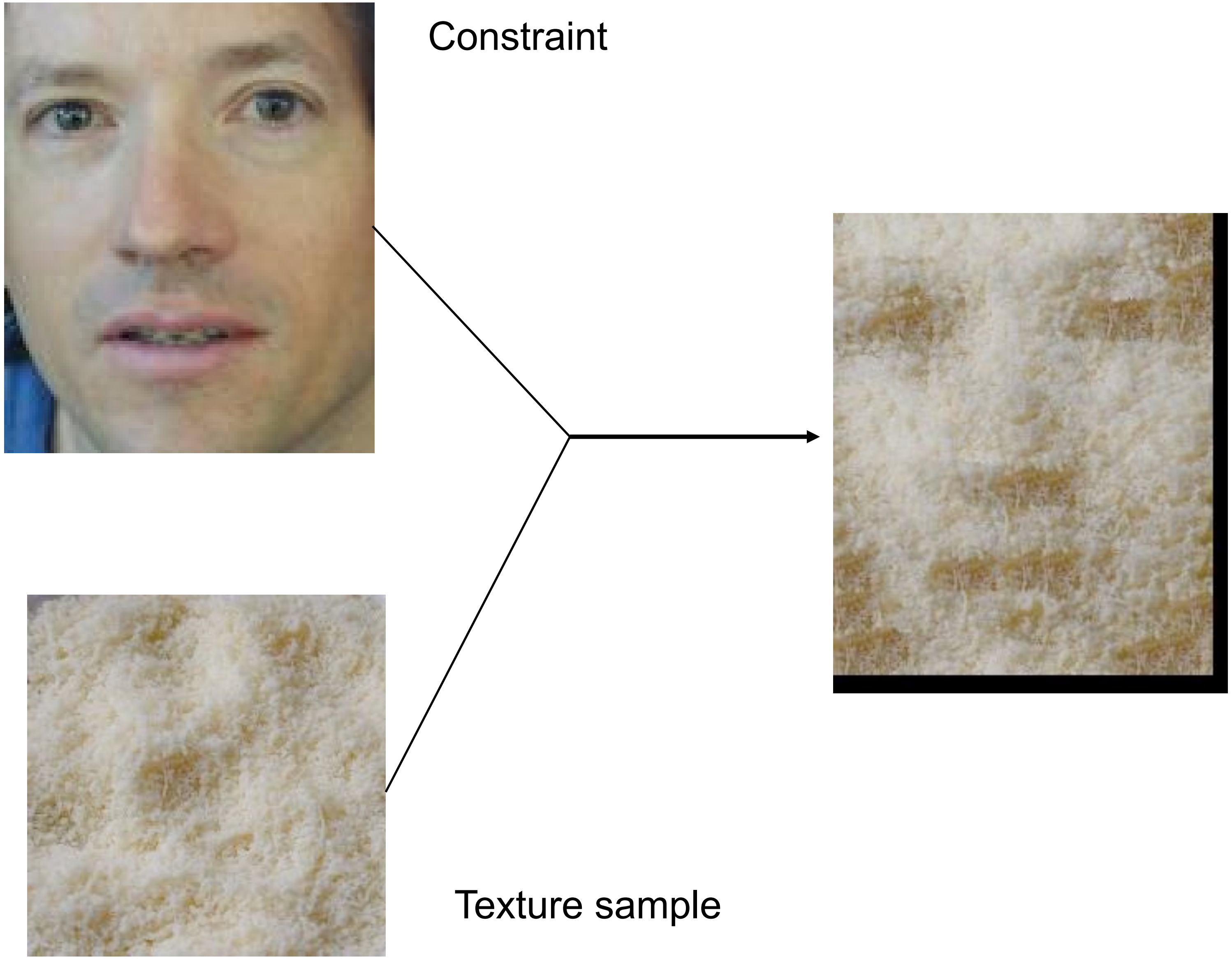


Texture Transfer

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

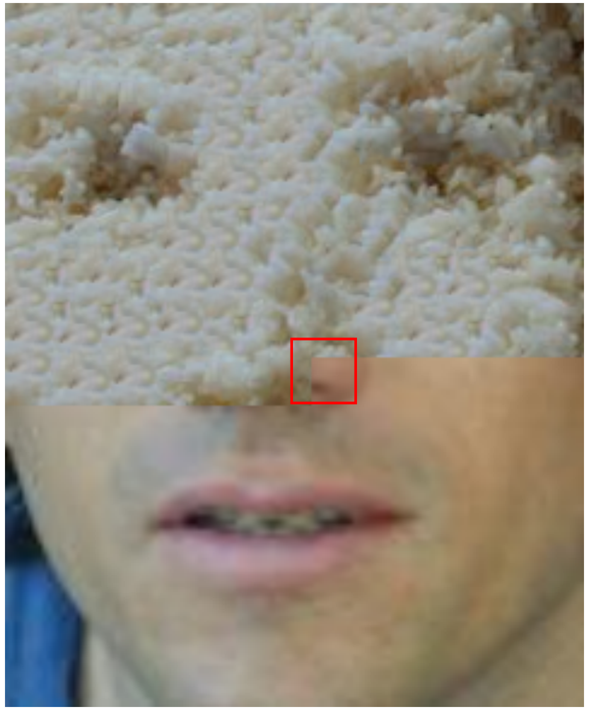


Texture Transfer



Texture Transfer

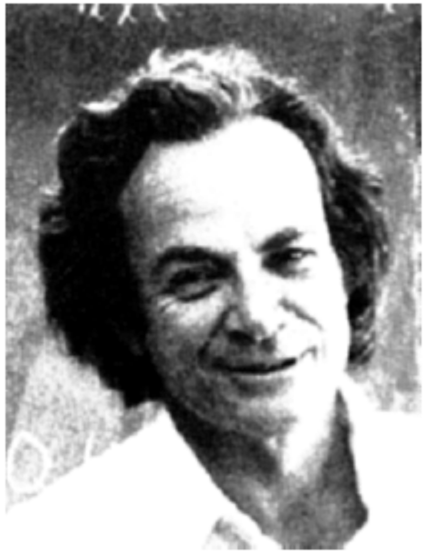
Take the texture from one image and "paint" it onto another object



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

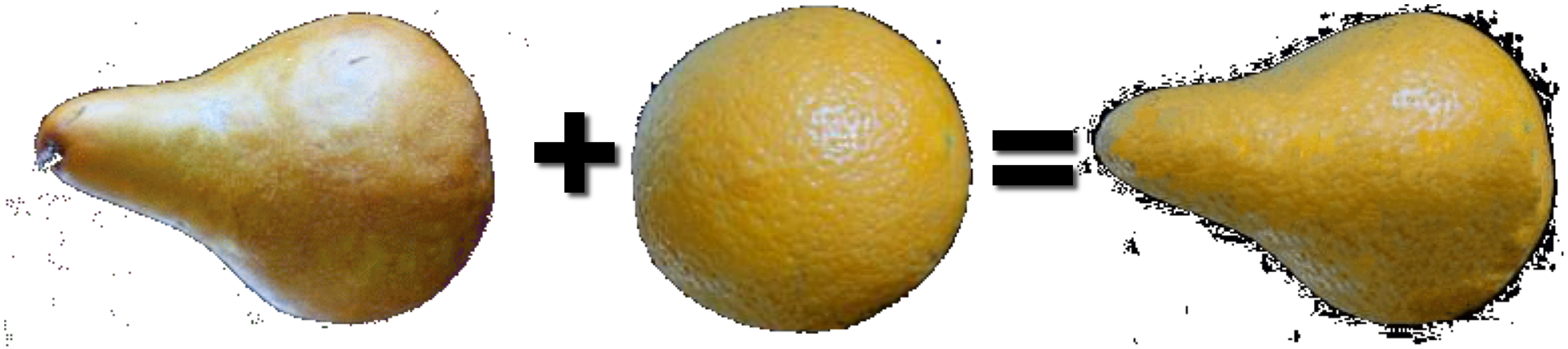
source texture



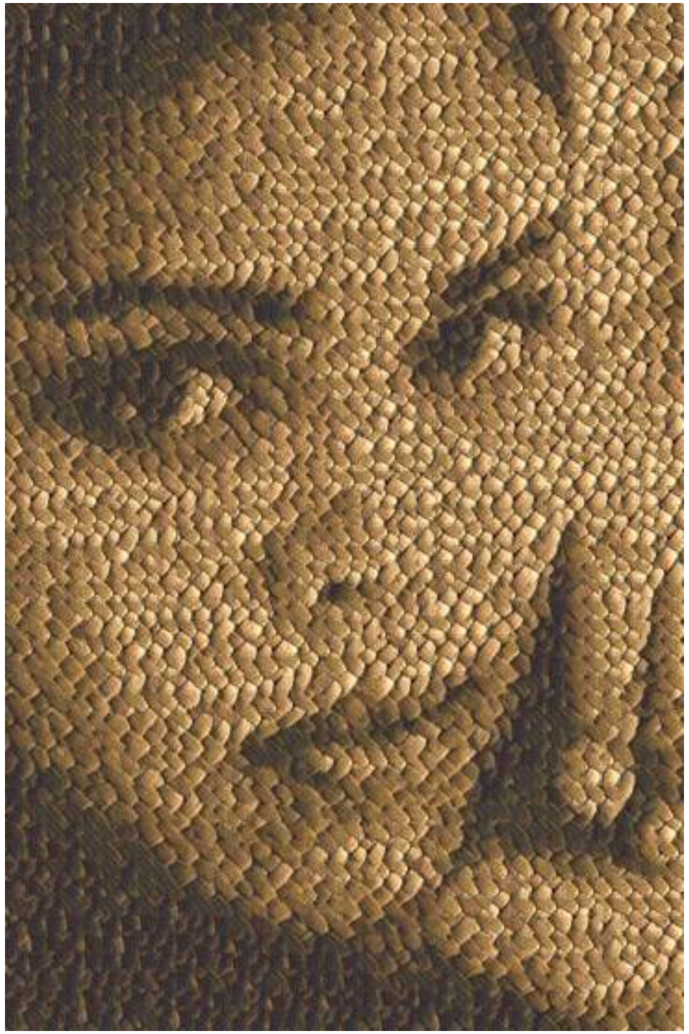
target image

correspondence maps

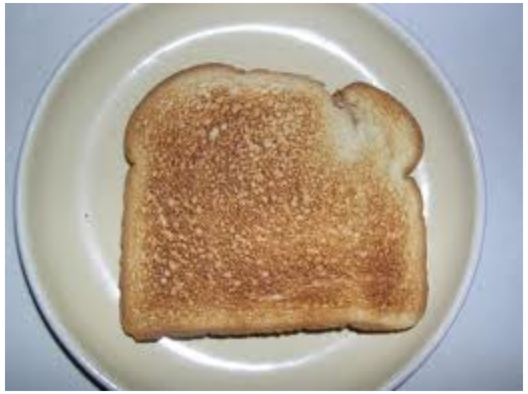
texture transfer result







Making sacred toast



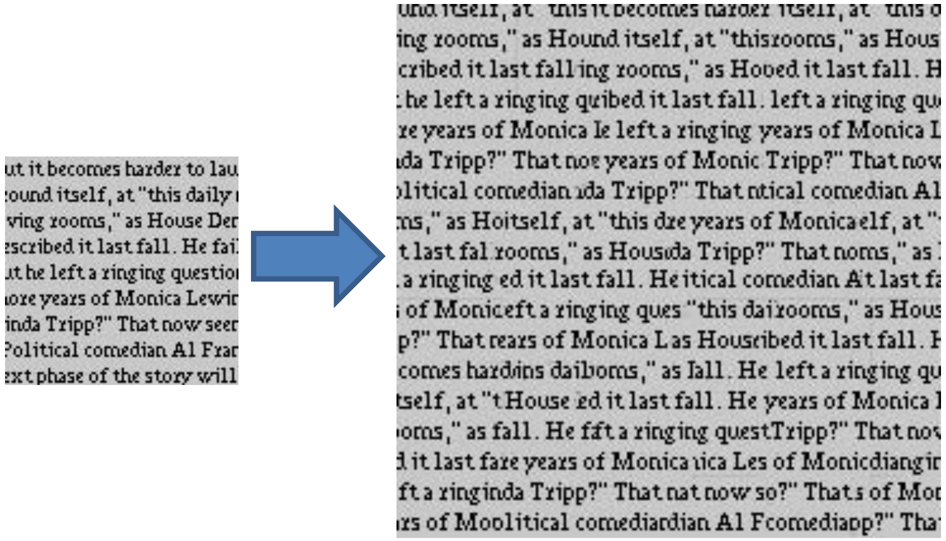
+

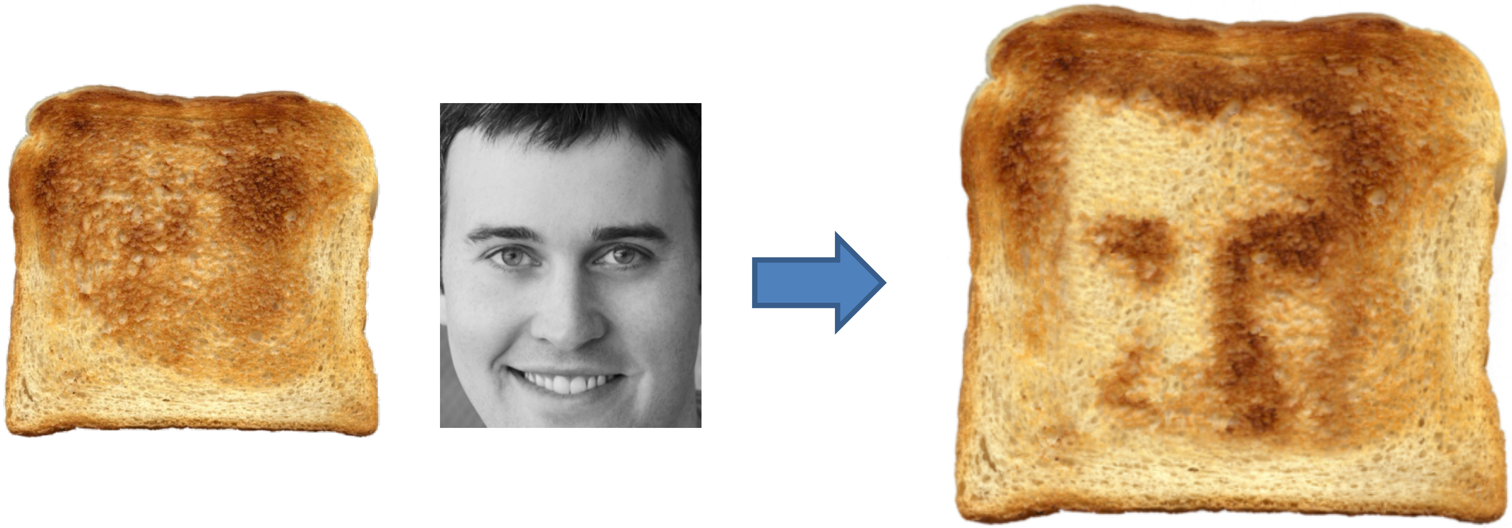
?

http://www.nbcnews.com/id/6511148/ns/us_news-weird_news/t/virgin-mary-grilled-cheese-sells/

Project 2: texture synthesis and transfer

- https://yxw.cs.illinois.edu/course/CS445/F21/projects/quilting/ComputationalPhotography_ProjectQuilting.html
- Note: this is significantly more challenging than the first project





Texture Synthesis and Transfer Recap

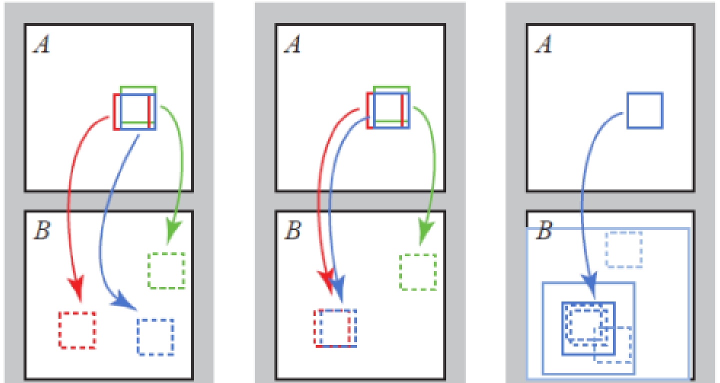
For each overlapping patch in the output image

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:

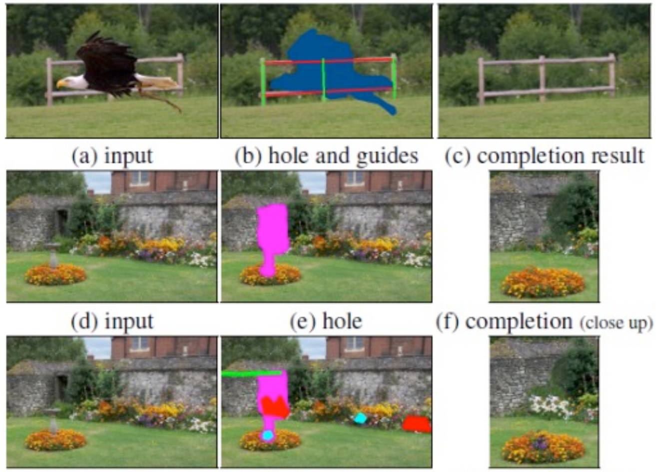
1. Randomly initialize matches
2. See if neighbor's offsets are better
3. Randomly search a local window for better matches
4. Repeat 3, 4 across image several times



(a) Initialization (b) Propagation (c) Search


(a) originals (b) random (c) $\frac{1}{4}$ iteration (d) $\frac{3}{4}$ iteration (e) 1 iteration (f) 2 iterations (g) 5 iterations

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


(a) input (b) hole and guides (c) completion result
(d) input (e) hole (f) completion (close up)
(g) same input (h) hole and guides (i) guided (close up)

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

Barnes et al. Siggraph 2009
http://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/index.php

Related idea: Image Analogies

A

::

A’

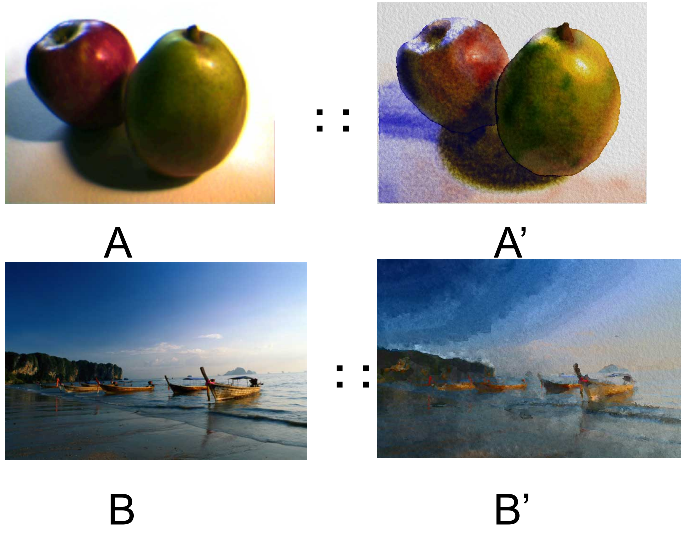
B

::

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/

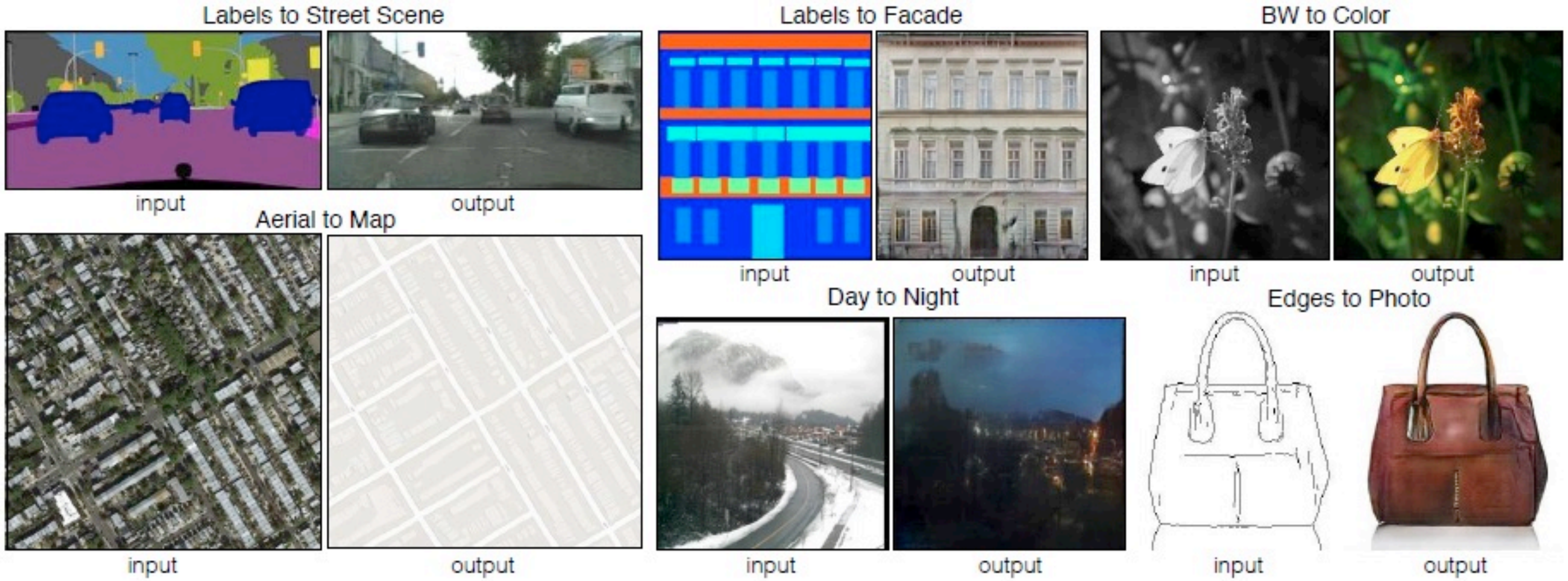
Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros

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

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

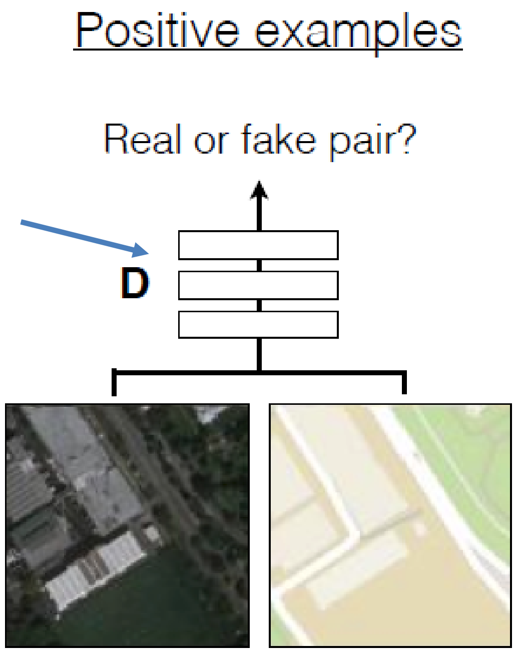
CVPR 2017

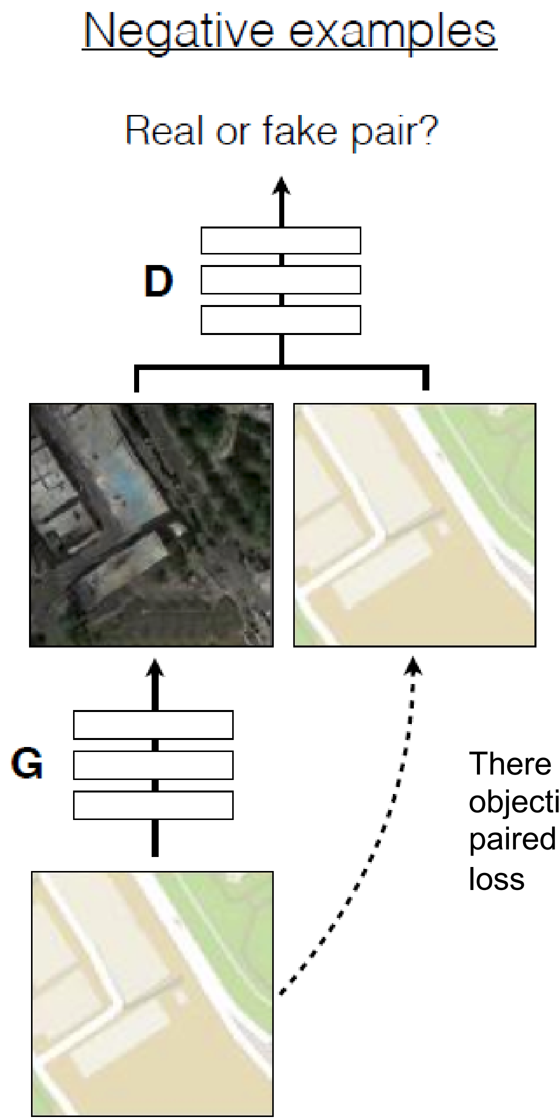
Learn to map from one image representation to another
- Trained from input/output pairs
- Patch memorization is implicit through learned representation



Learning to synthesize

Scores NxN patches for realism





There is also an objective to produce the paired image with an L1 loss

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

D tries to identify the fakes

Demos

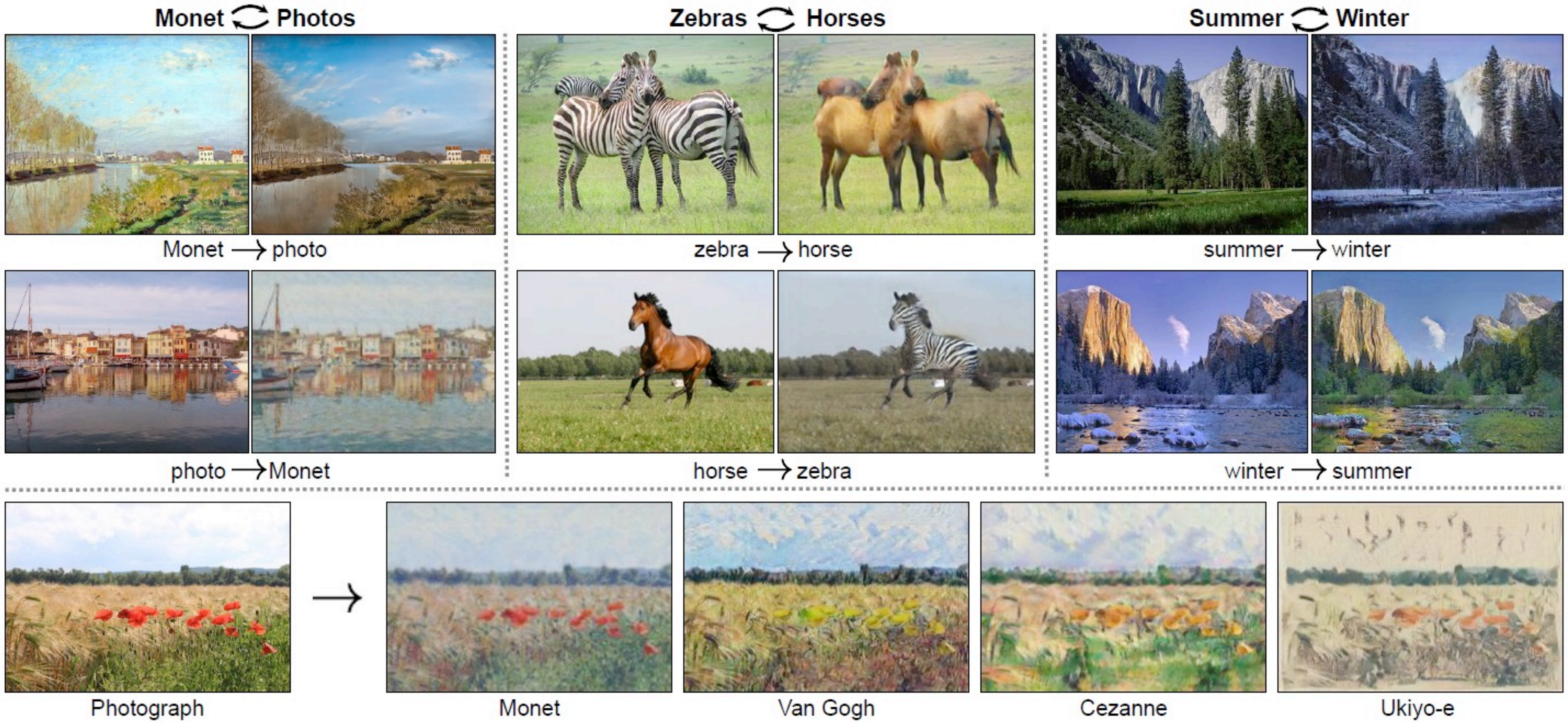
https://affinelayer.com/pixsrv/

Cycle GAN (ICCV 2017)

https://junyanz.github.io/CycleGAN/

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