

Data-driven methods: Video & Texture



© A.A. Efros

CS194: Intro to Computer Vision & Comp. Photography
Alexei Efros, UC Berkeley, Fall 2020

Michel Gondry train video

<http://www.youtube.com/watch?v=0S43lwBF0uM>

Class Choice award!



COMPSCI 194-26: Project 1

Images of the Russian Empire

Saurav Mittal | saurav@berkeley.edu

Weather Forecasting for Dummies™

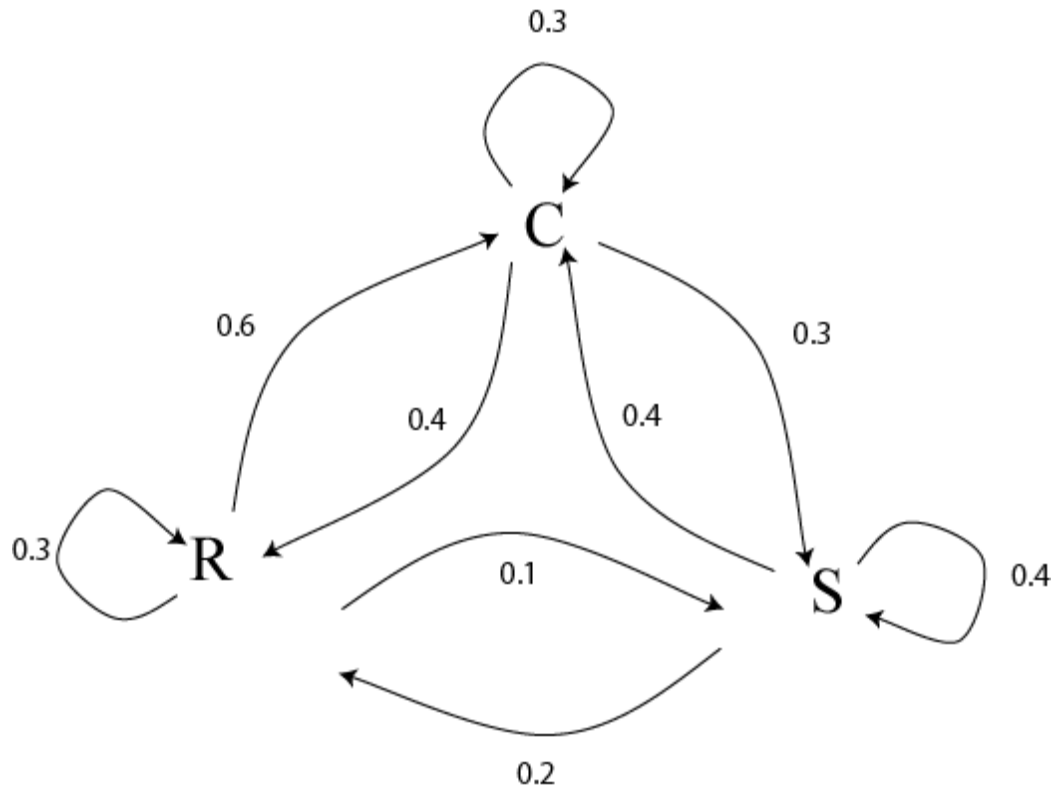
Let's predict weather:

- Given today's weather only, we want to know tomorrow's
- Suppose weather can only be {Sunny, Cloudy, Raining}

The “Weather Channel” algorithm:

- Over a long period of time, record:
 - How often S followed by R
 - How often S followed by S
 - Etc.
- Compute percentages for each state:
 - $P(R|S)$, $P(S|S)$, etc.
- Predict the state with highest probability!
- It's a Markov Chain

Markov Chain



$$\begin{pmatrix} 0.3 & 0.6 & 0.1 \\ 0.4 & 0.3 & 0.3 \\ 0.2 & 0.4 & 0.4 \end{pmatrix}$$

What if we know today and yestarday's weather?

Text Synthesis

[Shannon,'48] proposed a way to generate English-looking text using N-grams:

- Assume a generalized Markov model
- Use a large text to compute prob. distributions of each letter given N-1 previous letters
- Starting from a seed repeatedly sample this Markov chain to generate new letters
- Also works for whole words

WE NEED TO EAT CAKE

Mark V. Shaney (Bell Labs)

Results (using `alt.singles` corpus):

- *“As I've commented before, really relating to someone involves standing next to impossible.”*
- *“One morning I shot an elephant in my arms and kissed him.”*
- *“I spent an interesting evening recently with a grain of salt”*

Video Textures

Arno Schödl

Richard Szeliski

David Salesin

Irfan Essa

Microsoft Research, Georgia Tech

Still photos



Video clips



Video textures



Problem statement



video clip



video texture

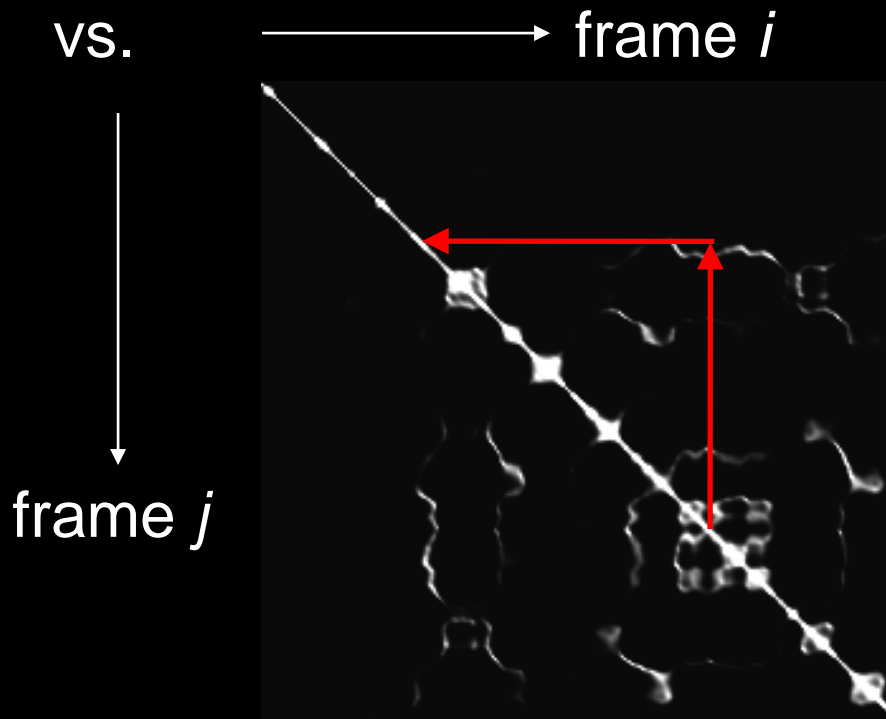
Our approach



- How do we find good transitions?

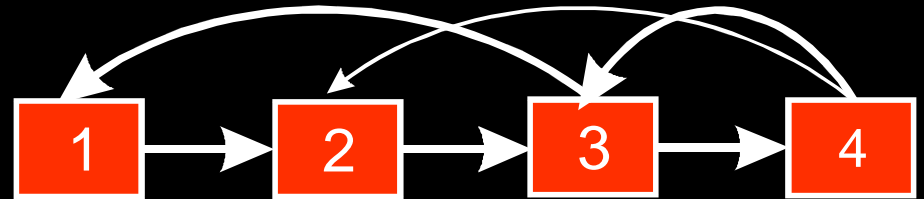
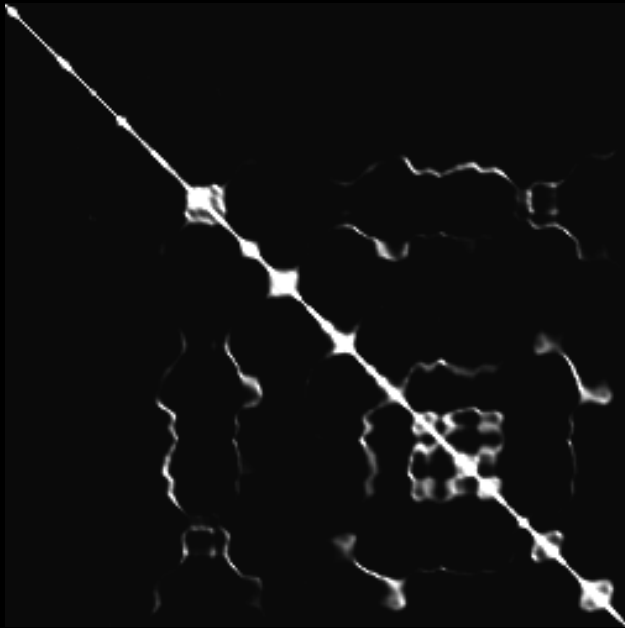
Finding good transitions

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



Similar frames make good transitions

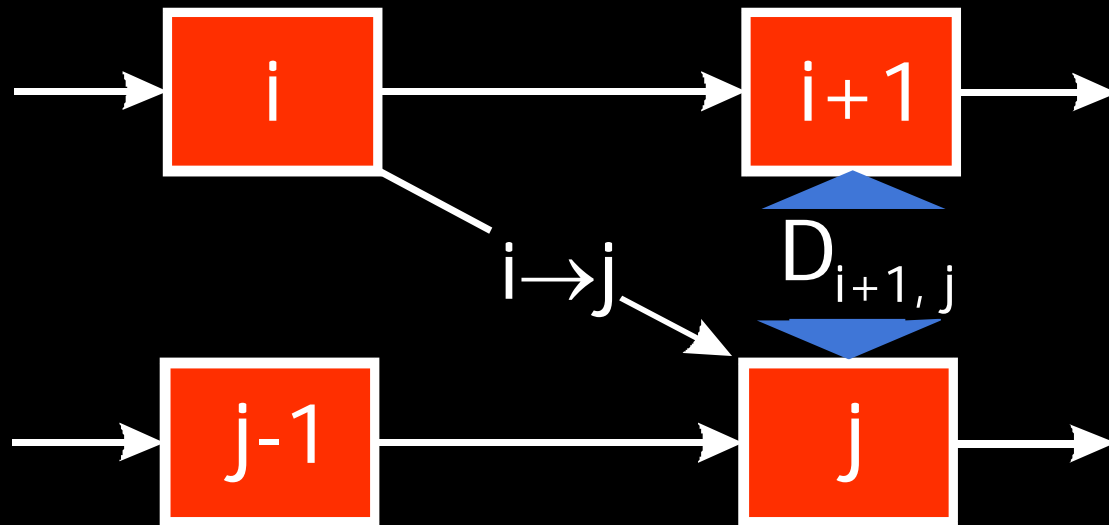
Markov chain representation



Similar frames make good transitions

Transition costs

- Transition from i to j if successor of i is similar to j
 - Cost function: $C_{i \rightarrow j} = D_{i+1, j}$



Transition probabilities

- Probability for transition $P_{i \rightarrow j}$ inversely related to cost:

- $P_{i \rightarrow j} \sim \exp (- C_{i \rightarrow j} / \sigma^2)$



high σ

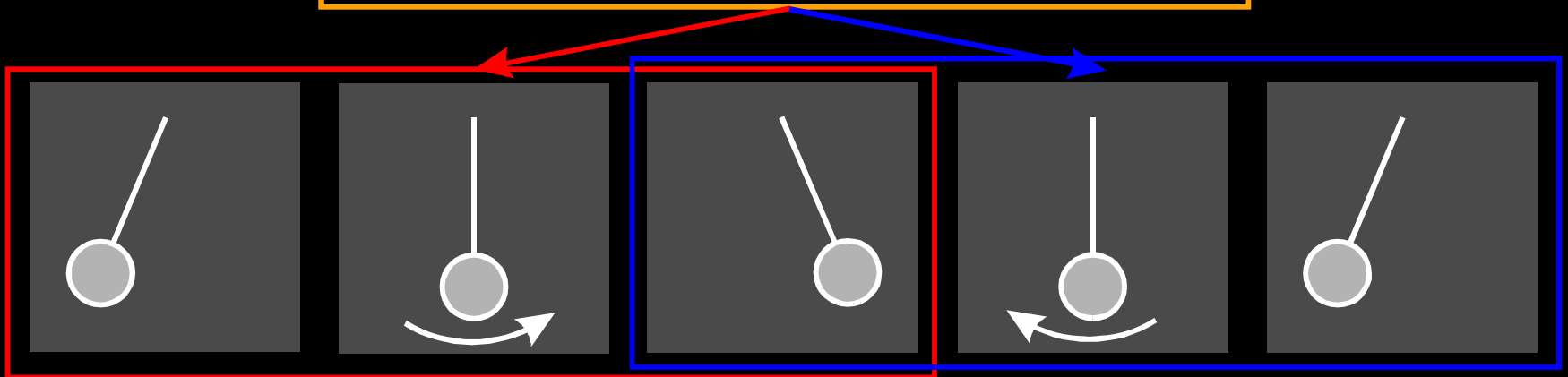
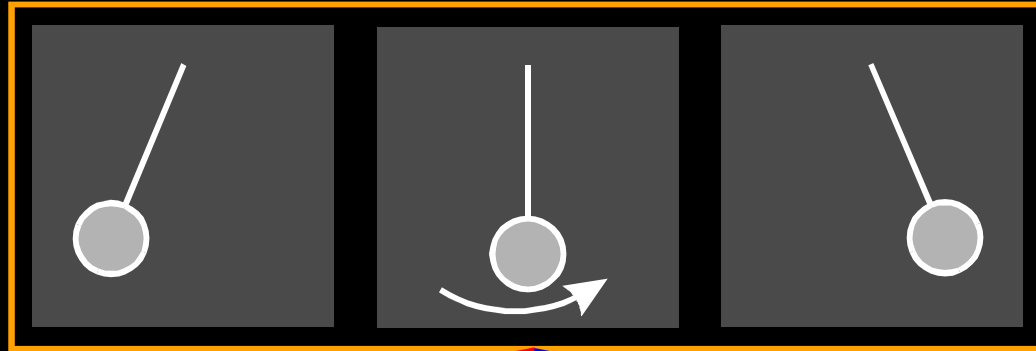


low σ

Preserving dynamics



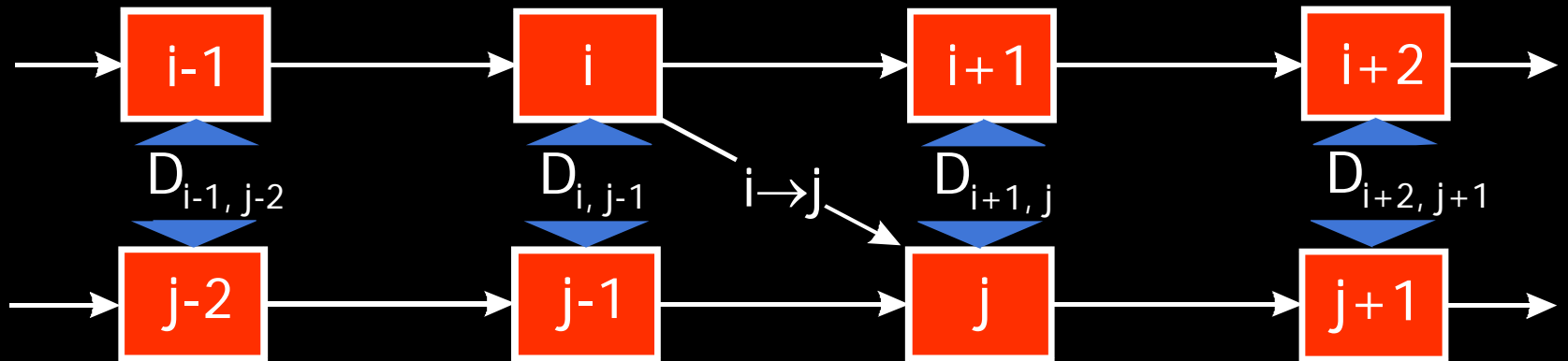
Preserving dynamics



Preserving dynamics

- Cost for transition $i \rightarrow j$

- $$C_{i \rightarrow j} = \sum_{k=-N}^{N-1} w_k D_{i+k+1, j+k}$$



Preserving dynamics – effect

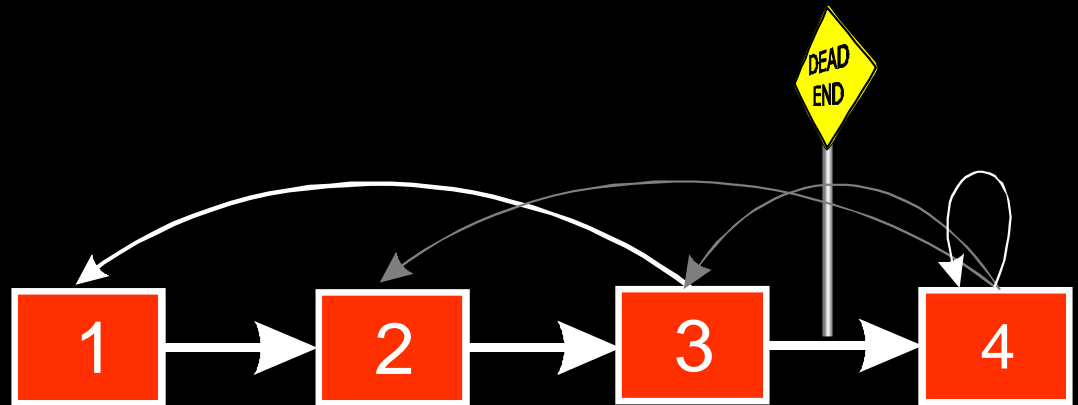
- Cost for transition $i \rightarrow j$

- $$C_{i \rightarrow j} = \sum_{k=-N}^{N-1} w_k D_{i+k+1, j+k}$$



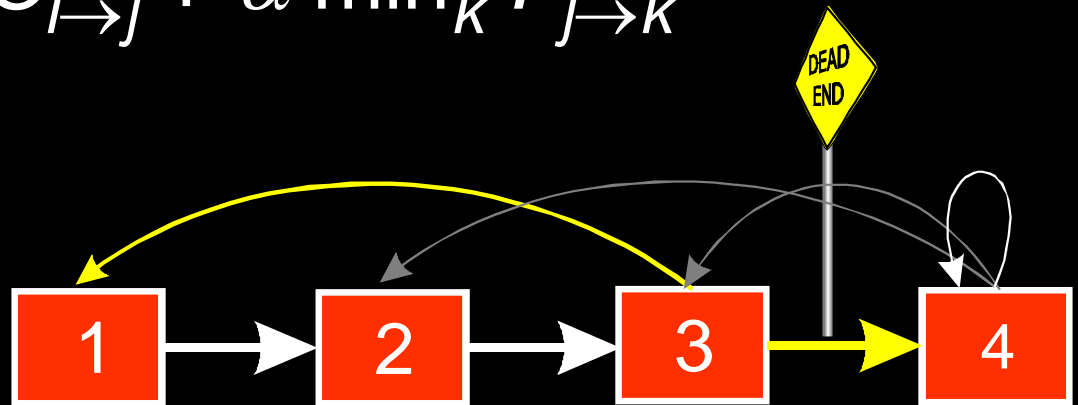
Dead ends

- No good transition at the end of sequence



Future cost

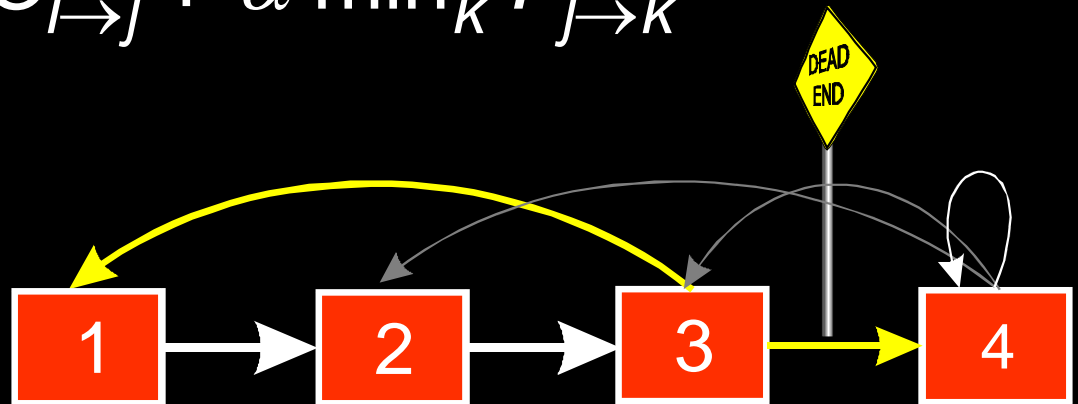
- Propagate future transition costs backward
- Iteratively compute new cost
 - $F_{i \rightarrow j} = C_{i \rightarrow j} + \alpha \min_k F_{j \rightarrow k}$



Future cost

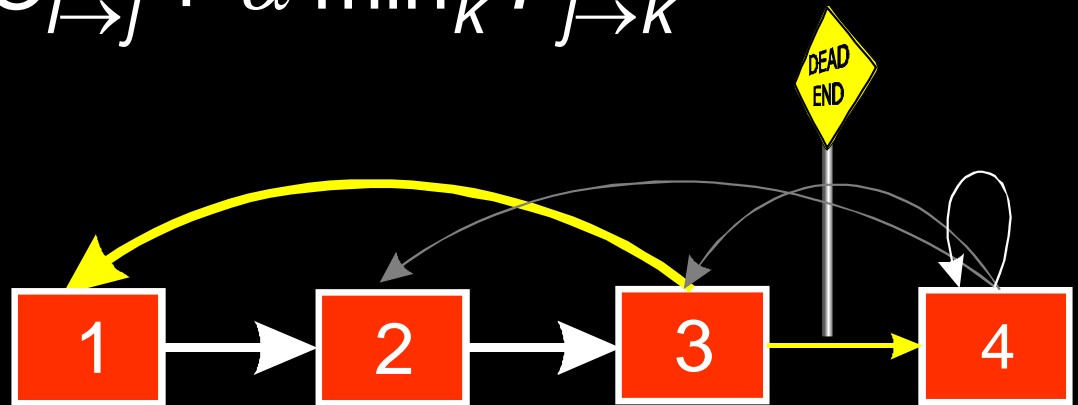
- Propagate future transition costs backward
- Iteratively compute new cost

- $$F_{i \rightarrow j} = C_{i \rightarrow j} + \alpha \min_k F_{j \rightarrow k}$$



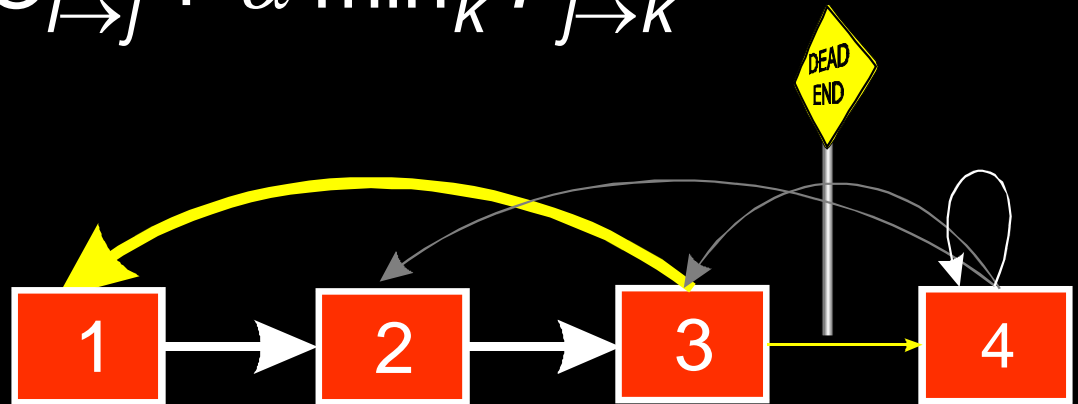
Future cost

- Propagate future transition costs backward
- Iteratively compute new cost
 - $F_{i \rightarrow j} = C_{i \rightarrow j} + \alpha \min_k F_{j \rightarrow k}$



Future cost

- Propagate future transition costs backward
- Iteratively compute new cost
 - $F_{i \rightarrow j} = C_{i \rightarrow j} + \alpha \min_k F_{j \rightarrow k}$

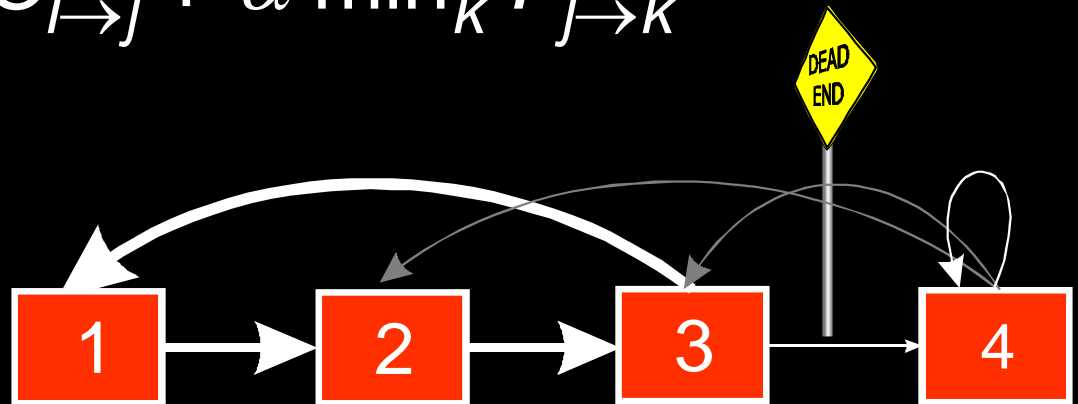


Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

- $$F_{i \rightarrow j} = C_{i \rightarrow j} + \alpha \min_k F_{j \rightarrow k}$$

- Q-learning



Final result



Finding good loops

- Alternative to random transitions
- Precompute set of loops up front



Video portrait



- c.f. Harry Potter

Region-based analysis

- Divide video up into regions



- Generate a video texture for each region

User-controlled video textures



slow



variable

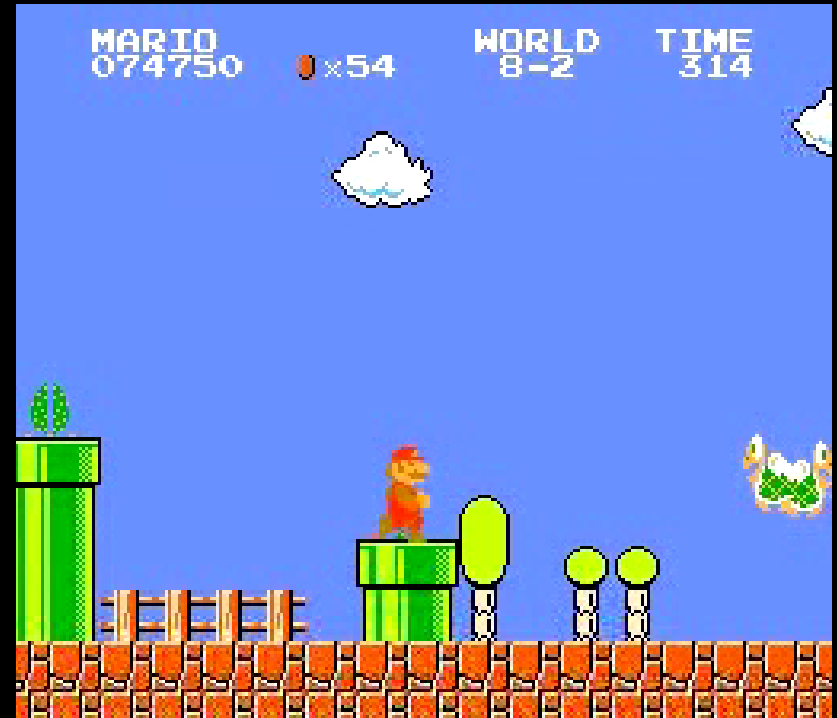


fast

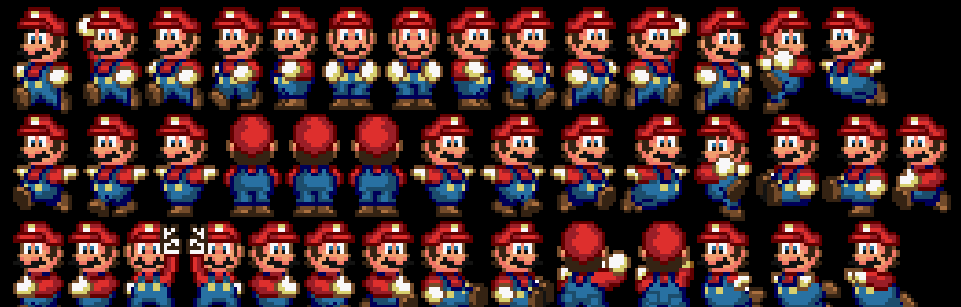
User selects target frame range

Video-based animation

- Like sprites in computer games
- Extract sprites from real video
- Interactively control desired motion



©1985 Nintendo of America Inc.



Video sprite extraction

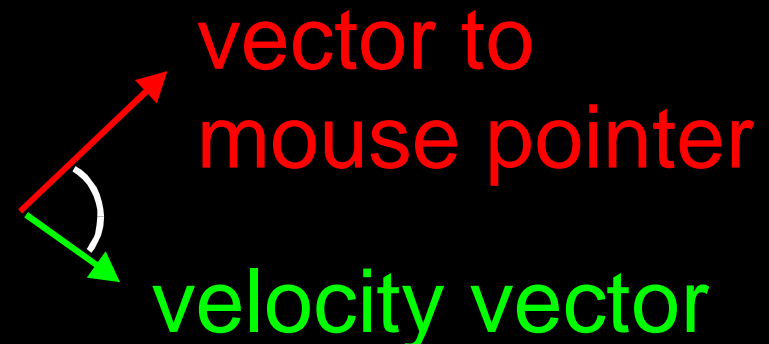


blue screen matting
and velocity estimation



Video sprite control

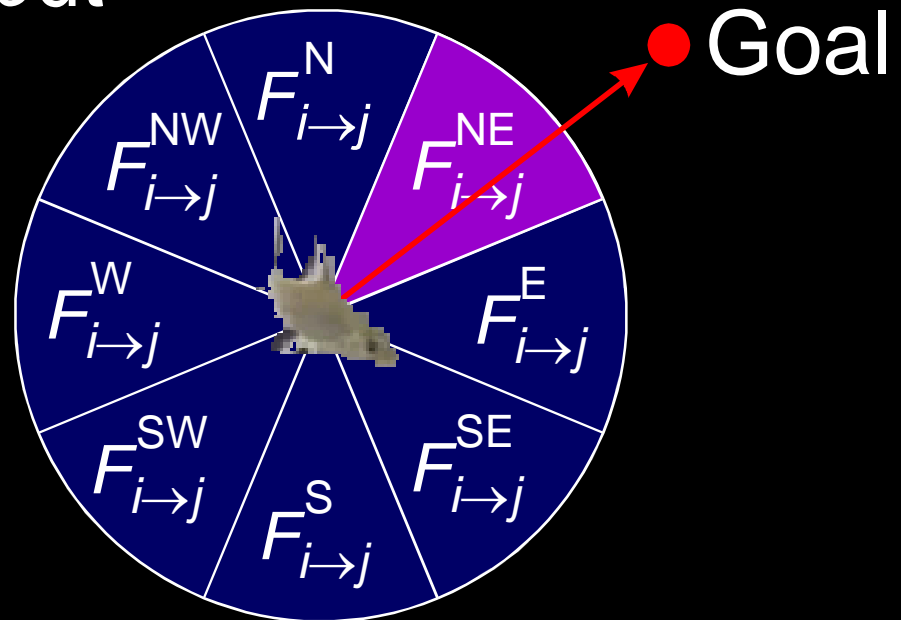
- Augmented transition cost:

$$C_{i \rightarrow j}^{\text{Animation}} = \alpha \underbrace{C_{i \rightarrow j}}_{\text{Similarity term}} + \beta \underbrace{\text{angle}}_{\text{Control term}}$$


The diagram illustrates the 'Control term' in the equation. It shows two vectors originating from a common point: a red vector pointing towards the upper right and a green vector pointing towards the lower right. An arc between the two vectors indicates the angle between them. The red vector is labeled 'vector to mouse pointer' in red text, and the green vector is labeled 'velocity vector' in green text.

Video sprite control

- Need future cost computation
- Precompute future costs for a few angles.
- Switch between precomputed angles according to user input
- [GIT-GVU-00-11]

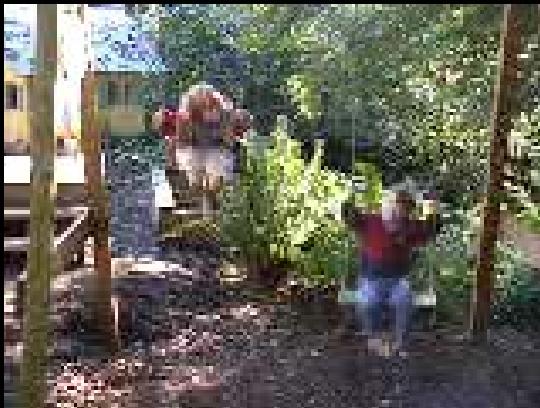


Interactive fish



Summary / Discussion

- Some things are relatively easy



Discussion

- Some are hard



“Amateur” by Lasse Gjertsen

<http://www.youtube.com/watch?v=JzqumbhfxRo>

similar idea:

<http://www.youtube.com/watch?v=MsBMG-p1HDM&feature=share&list=PLFFD733D0FF425290>

Hyperlapse Videos

https://www.youtube.com/watch?v=Wt_Y04xn84M

“Do As I Do” (ICCV 2003)

<https://youtu.be/UMJcpLIAwKg>

Texture

- Texture depicts spatially repeating patterns
- Many natural phenomena are textures



radishes



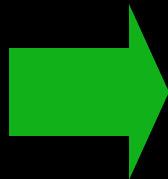
rocks



yogurt

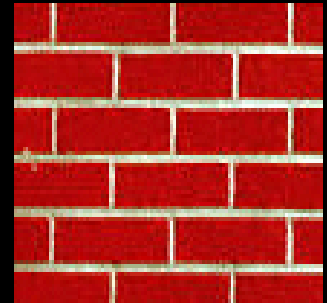
Texture Synthesis

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

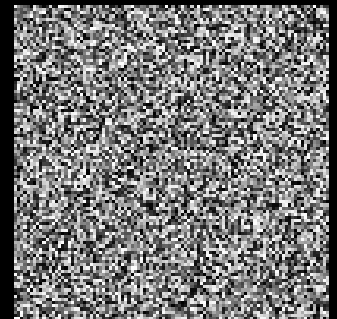


The Challenge

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



repeated

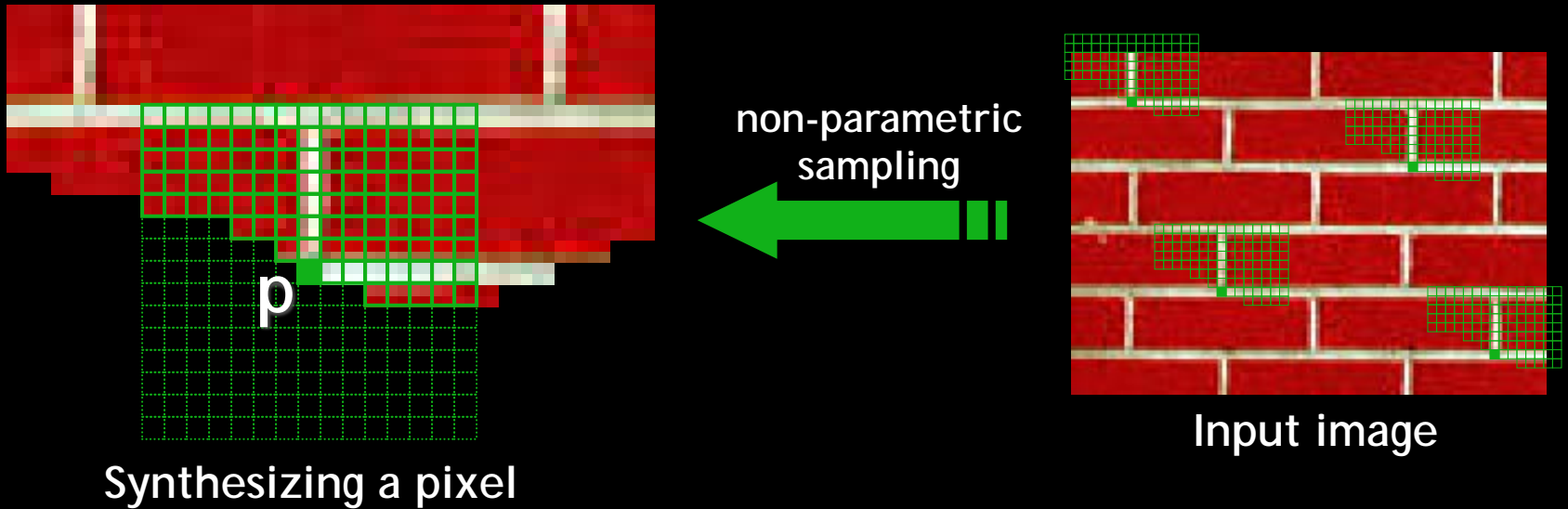


stochastic



Both?

Efros & Leung Algorithm

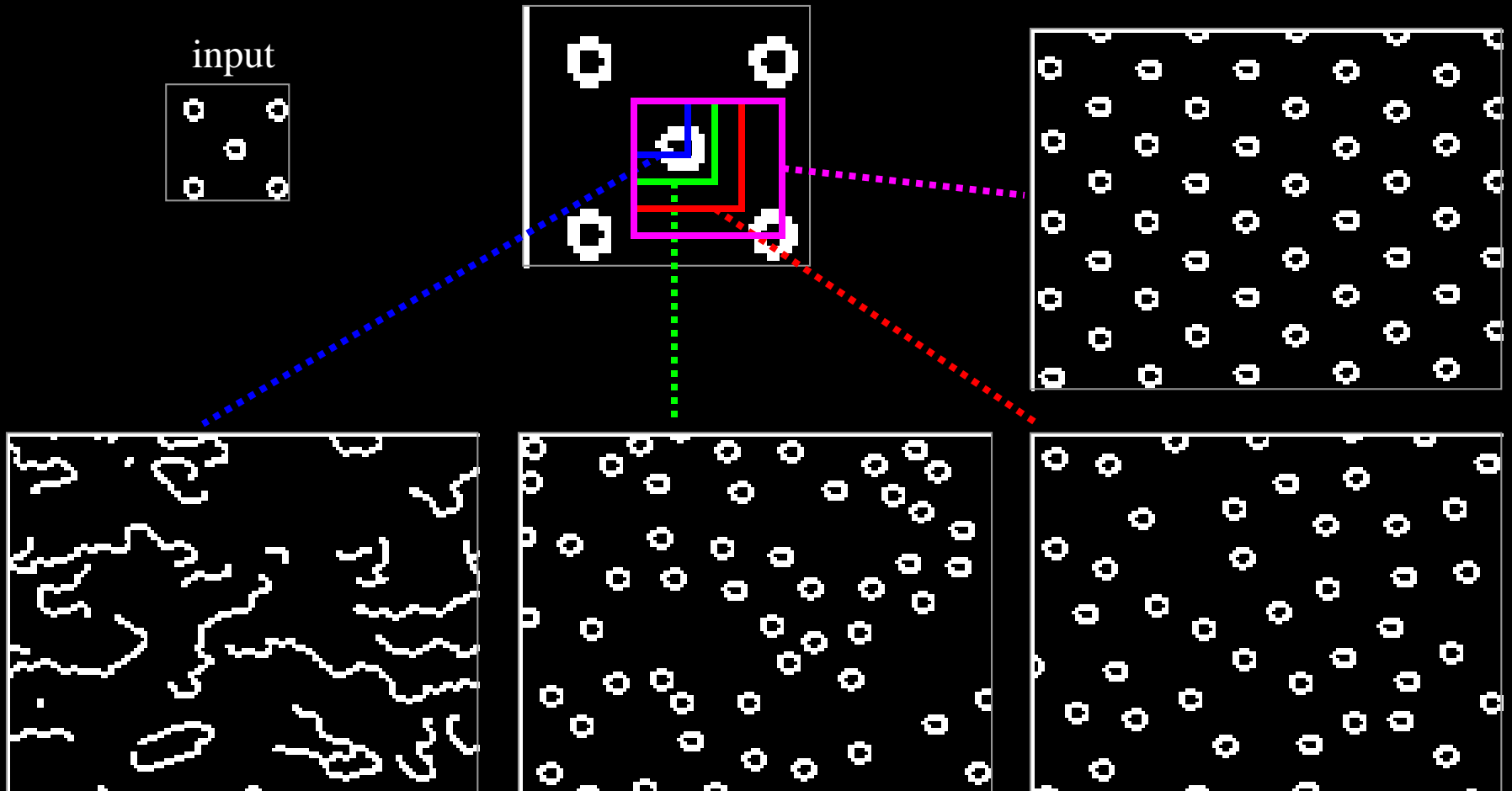


- Assuming Markov property, compute $P(\mathbf{p}|\mathbf{N}(\mathbf{p}))$
 - Building explicit probability tables infeasible
 - Instead, we *search the input image* for all similar neighborhoods — that's our pdf for \mathbf{p}
 - To sample from this pdf, just pick one match at random

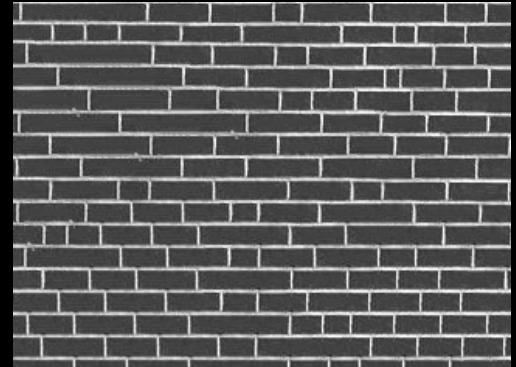
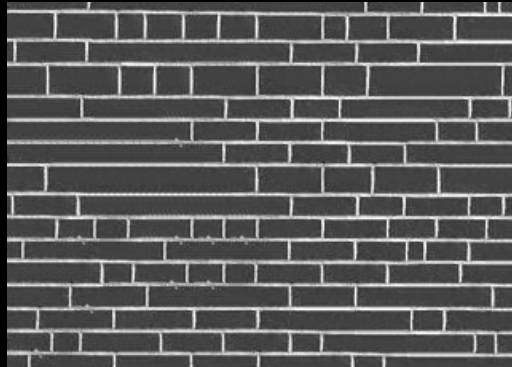
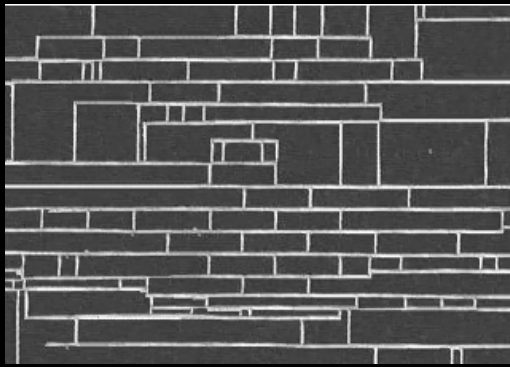
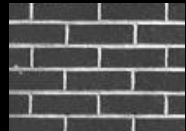
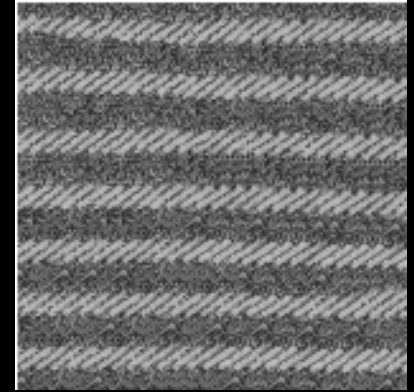
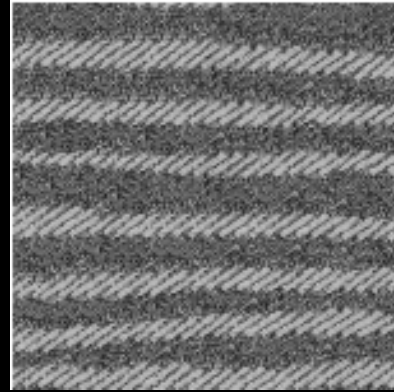
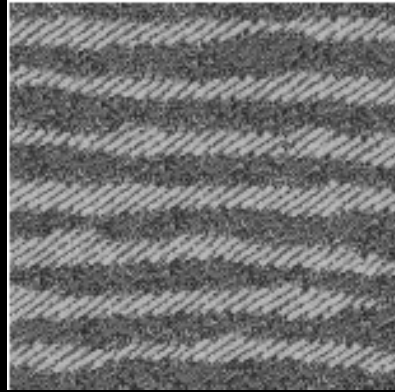
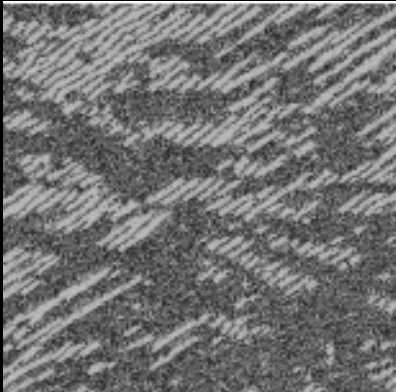
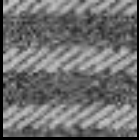
Some Details

- Growing is in “onion skin” order
 - Within each “layer”, pixels with most neighbors are synthesized first
 - If no close match can be found, the pixel is not synthesized until the end
- Using *Gaussian-weighted* SSD is very important
 - to make sure the new pixel agrees with its closest neighbors
 - Approximates reduction to a smaller neighborhood window if data is too sparse

Neighborhood Window



Varying Window Size

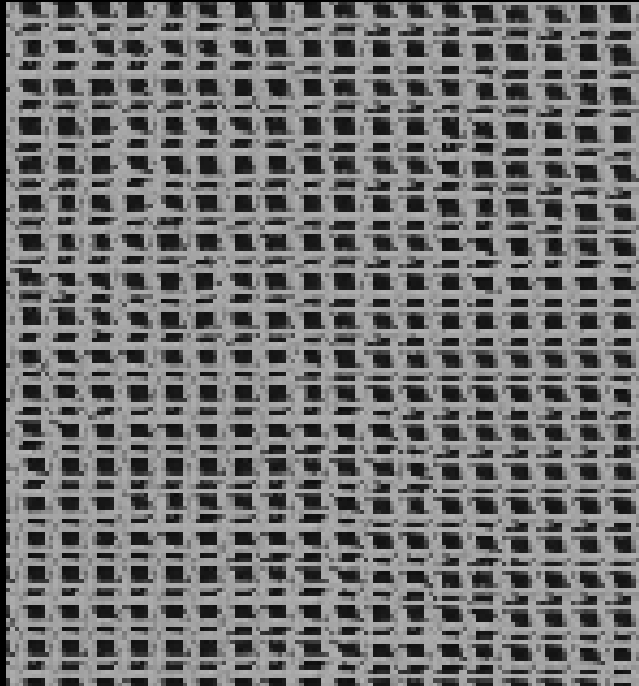
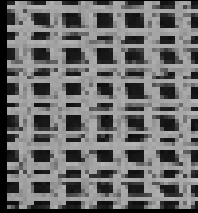


Increasing window size

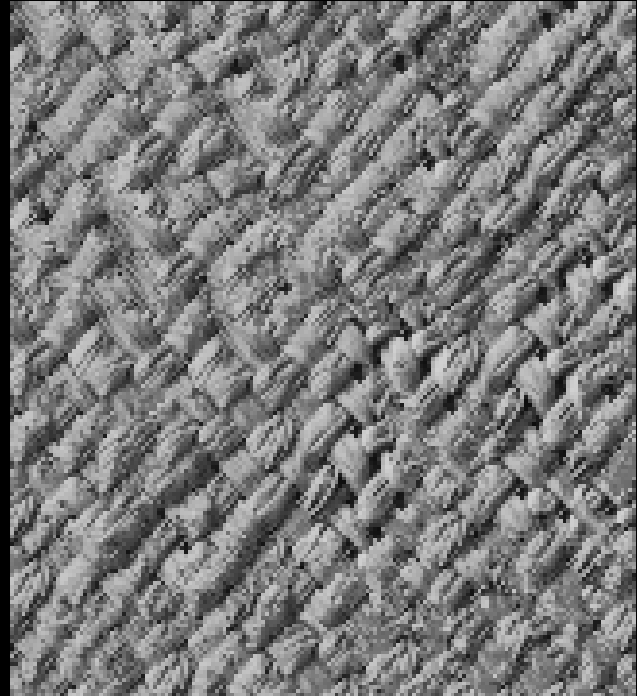


Synthesis Results

french canvas



rafia weave

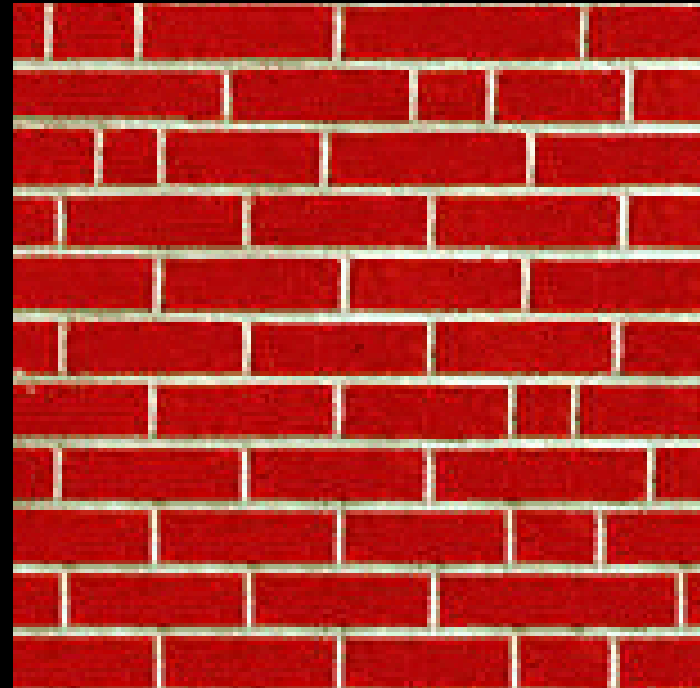
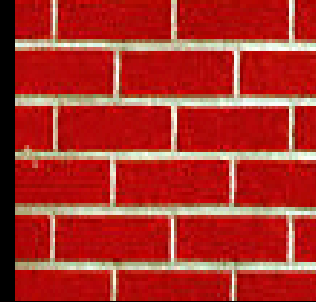


More Results

white bread



brick wall

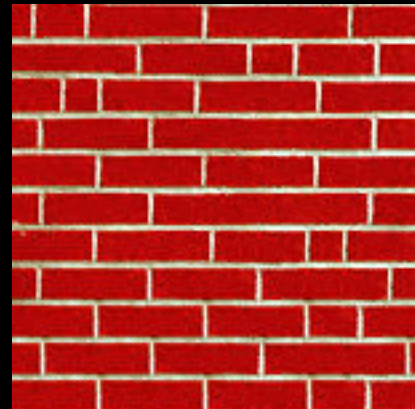
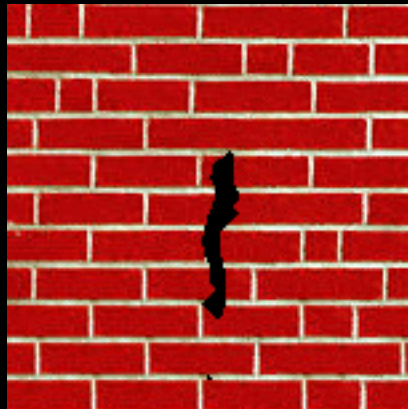


Homage to Shannon

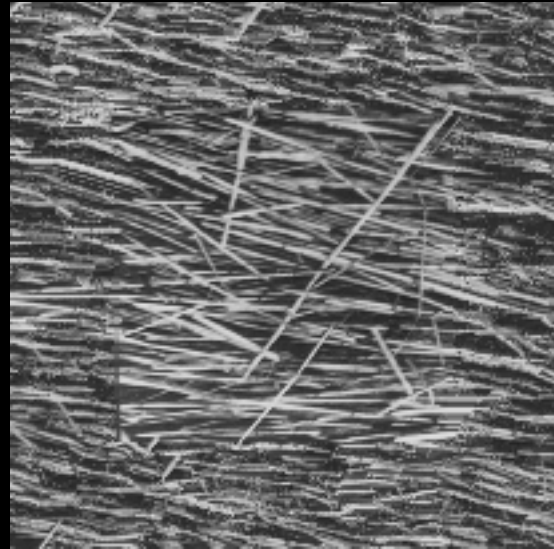
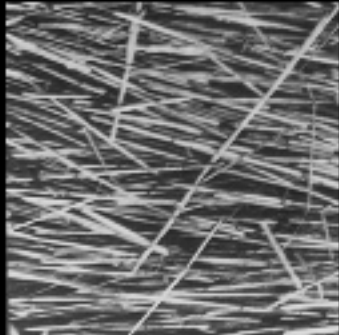
ing in the unsensational
r Dick Gephardt was fai
rful riff on the looming
ly asked, "What's your
tions?" A heartfelt sigh
story about the emergen
es against Clinton. "Boy
g people about continuin
hardt began, patiently obs
e, that the legal system h
g with this latest tanger

thaim, them. "Whnephartfe lartifelintomimer,
fel ck Clirtioout omaim thartfelins, f out s anento
the ry onst wartfe lck Gephtoomimeationl sigab
Cllooufti Clinut Cil riff on, hat's y'odn, parut tly
ons yontonsteht wasked, paim t sahe loo' riff on
nskoneploourtfeas leil A nst Clit, "Wleontongal s
k Clirtioouirtfepe.ong pme abegal fartfenstemem
tiensteneltorydt telemephminsverdt was agemer
ff ons artientont Cling peme as,rtfe atih,"Boui s
nal s fartfelt sig pedrtihdt ske abounutie aboutioo
tfaonewwas yow aboronthardt thatins fain, ped,
ains, them, pabout wasy arfuit couitly d, l n A h
ble emthrdngbooreme agas fa bontinsyst Clinut
ory about continst Clipeouinst 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.emungairt tfoun
The loocrystan loontieph, intly on, tieoplegatick C
aul fatiezonfly atie Dioniomi wal s f t begae ener
nthahgat's epenhbbas fan, "intchthory abons

Hole Filling



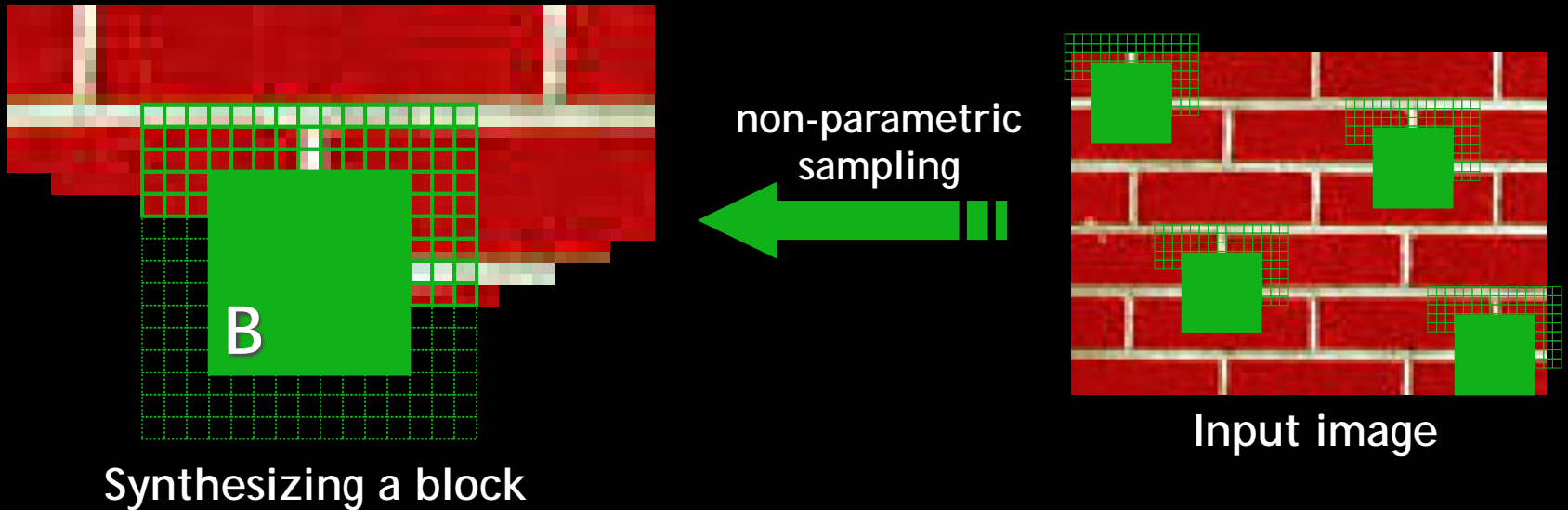
Extrapolation



Summary

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

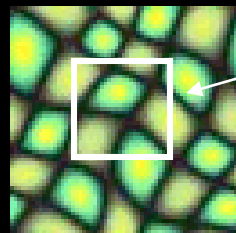
Image Quilting [Efros & Freeman]



- Observation: neighbor pixels are highly correlated

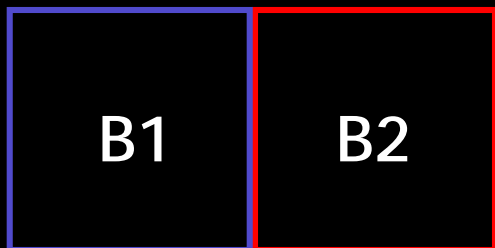
Idea: unit of synthesis = block

- Exactly the same but now we want $P(B|N(B))$
- Much faster: synthesize all pixels in a block at once
- Not the same as multi-scale!

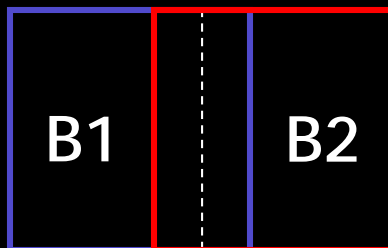


block

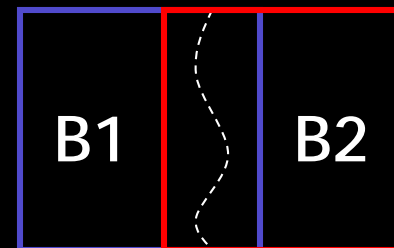
Input texture



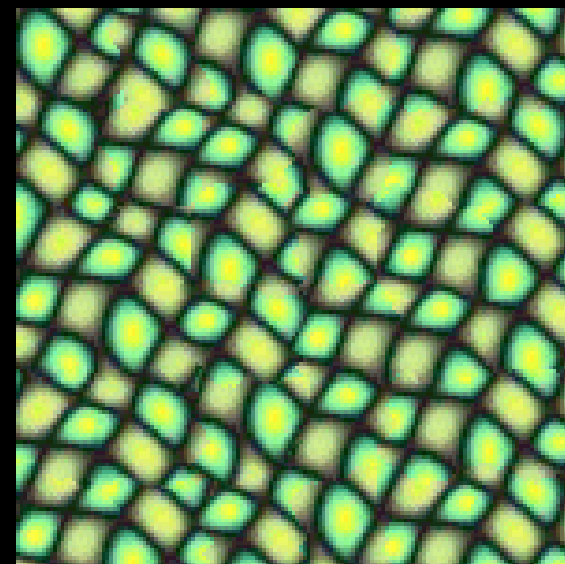
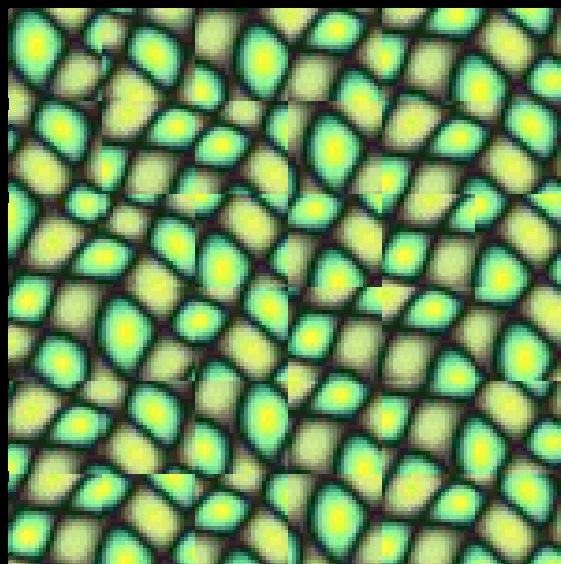
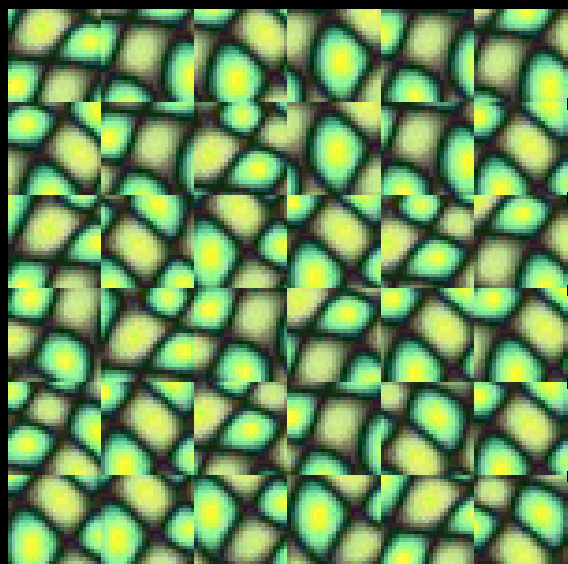
Random placement
of blocks



Neighboring blocks
constrained by overlap

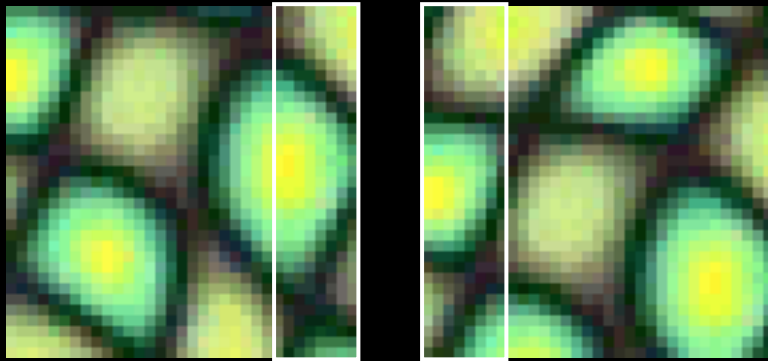


Minimal error
boundary cut

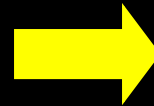
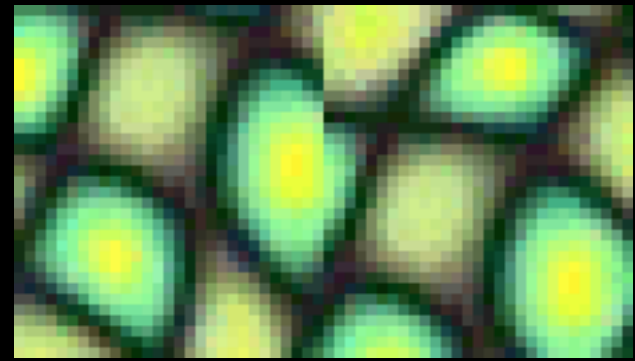


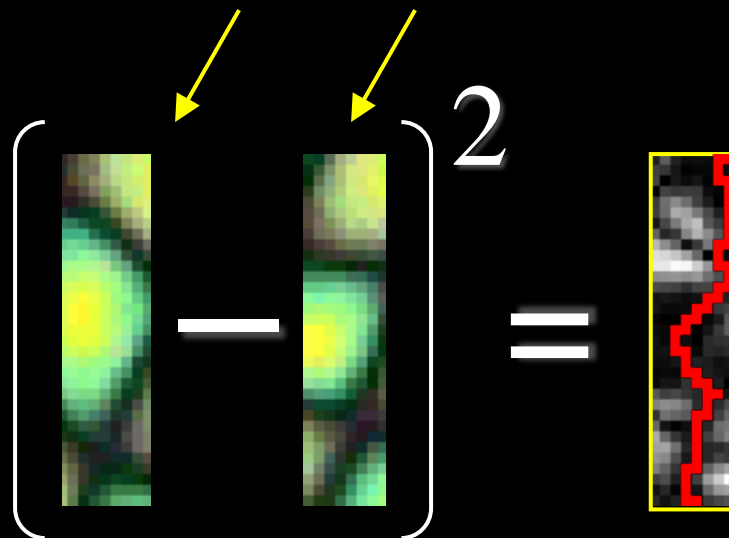
Minimal error boundary

overlapping blocks

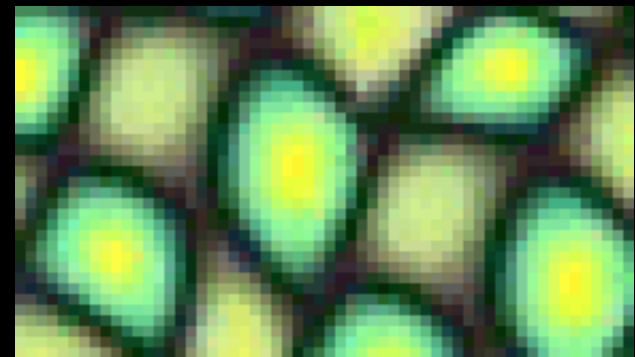


vertical boundary




$$\left[\text{block}_1 - \text{block}_2 \right]^2 = \text{error_strip}$$

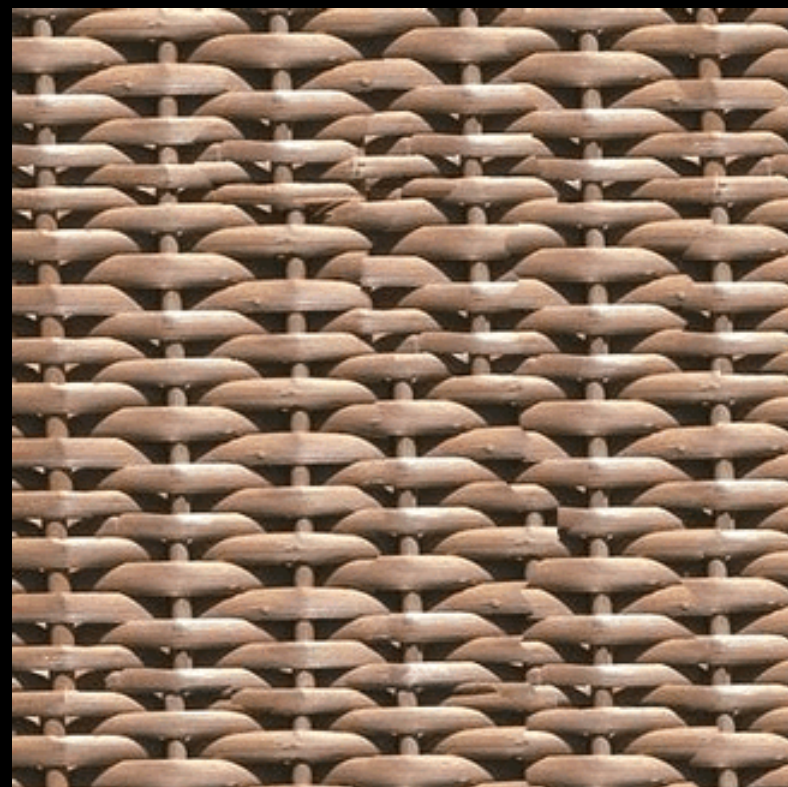
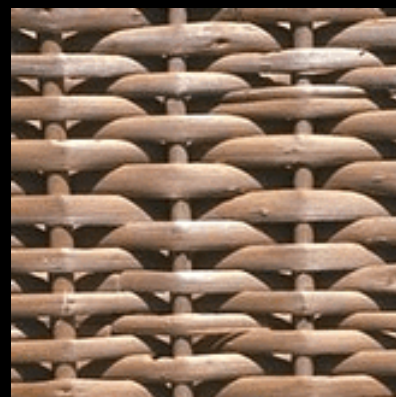
overlap error

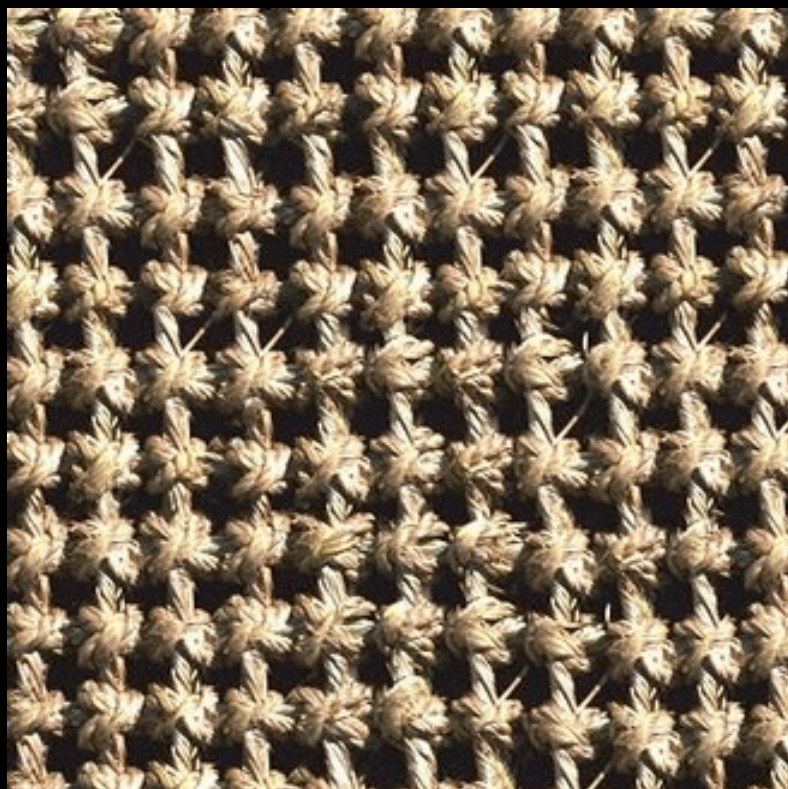


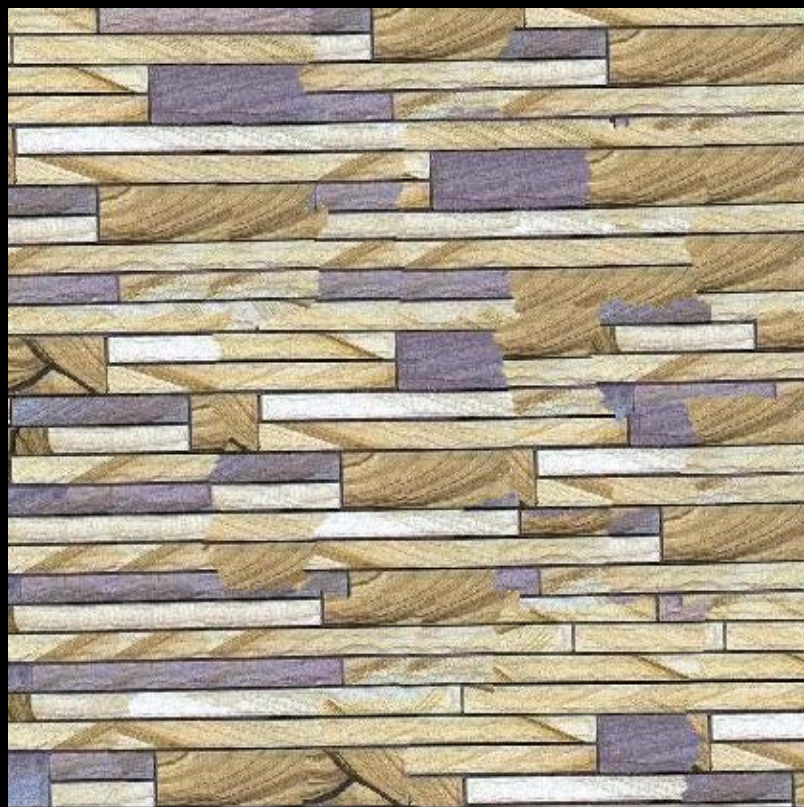
min. error boundary

Our Philosophy

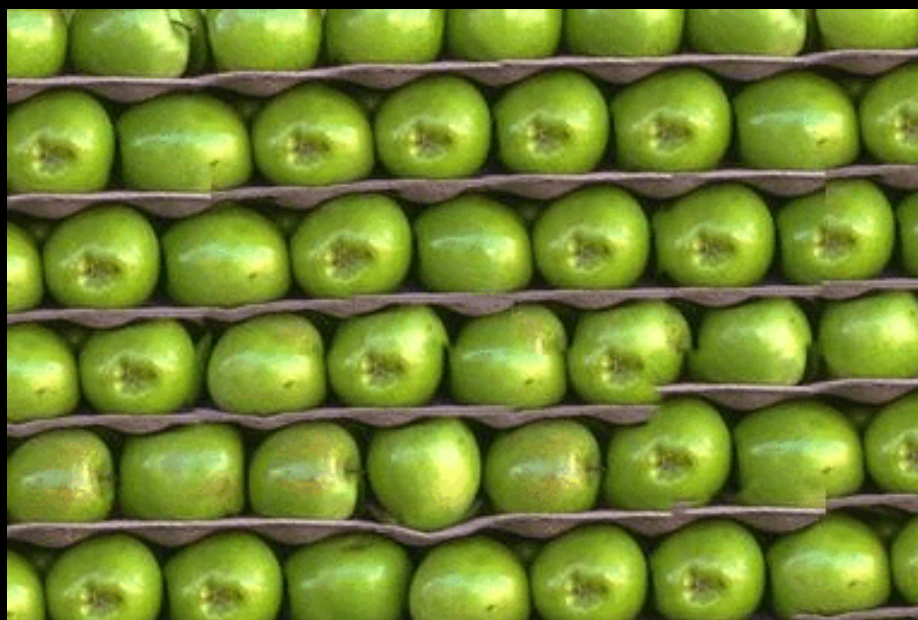
- The “Corrupt Professor’s Algorithm”:
 - Plagiarize as much of the source image as you can
 - Then try to cover up the evidence
- Rationale:
 - Texture blocks are by definition correct samples of texture so problem only connecting them together

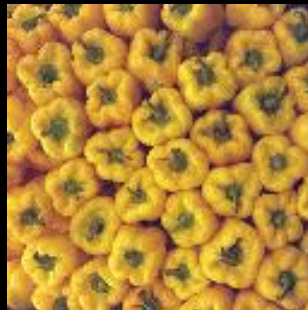
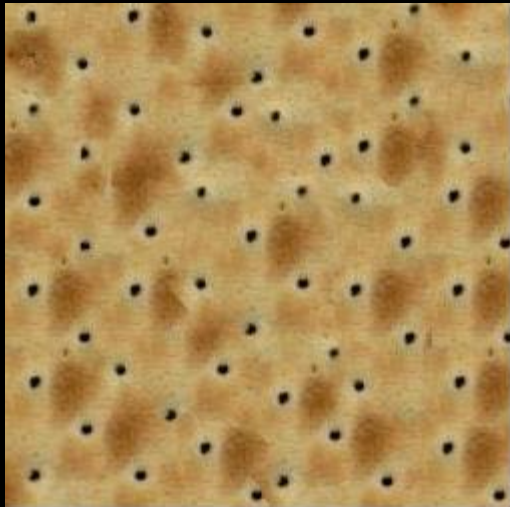






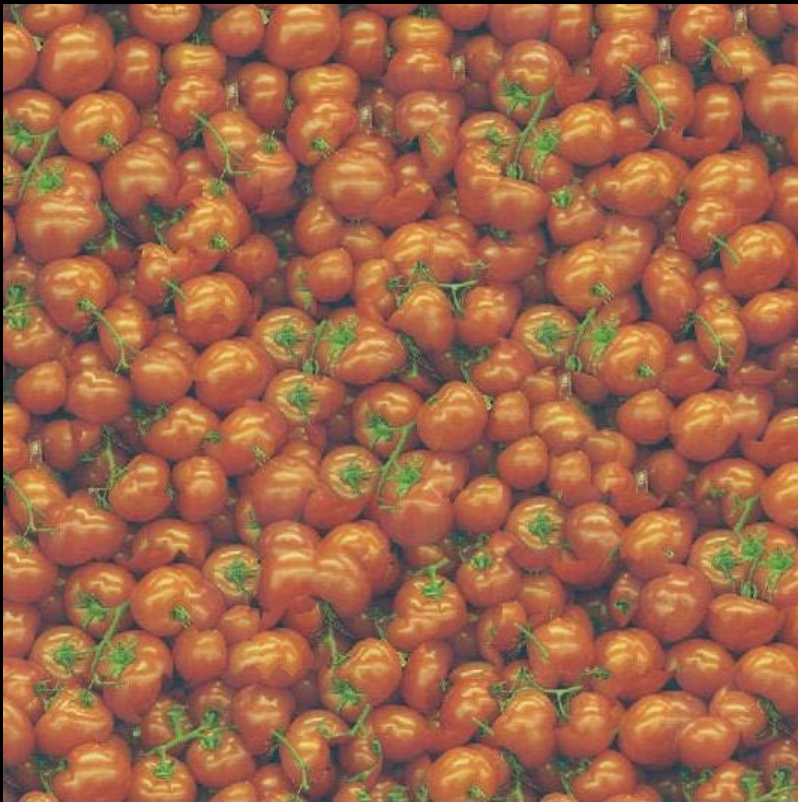


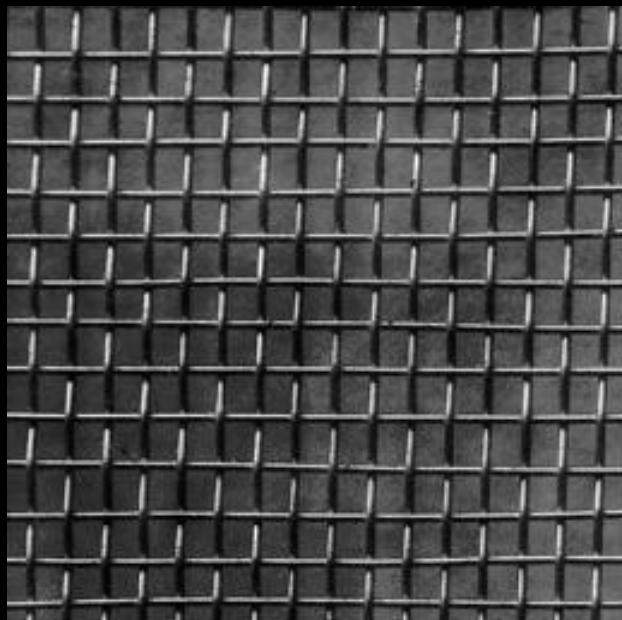




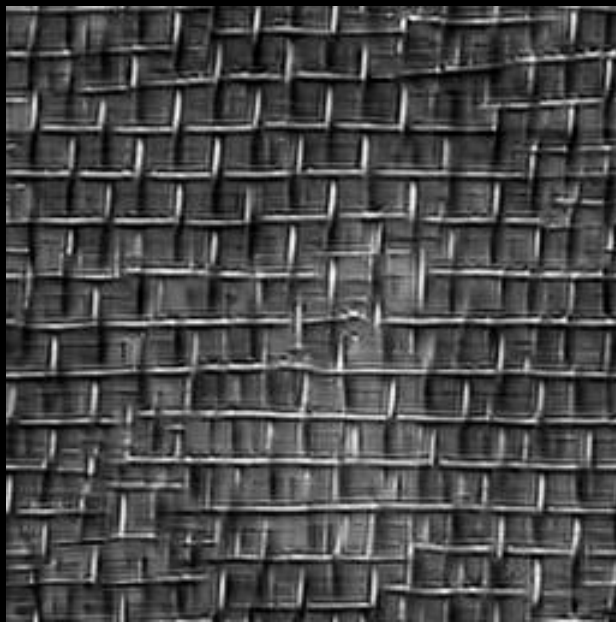


Failures (Chernobyl Harvest)

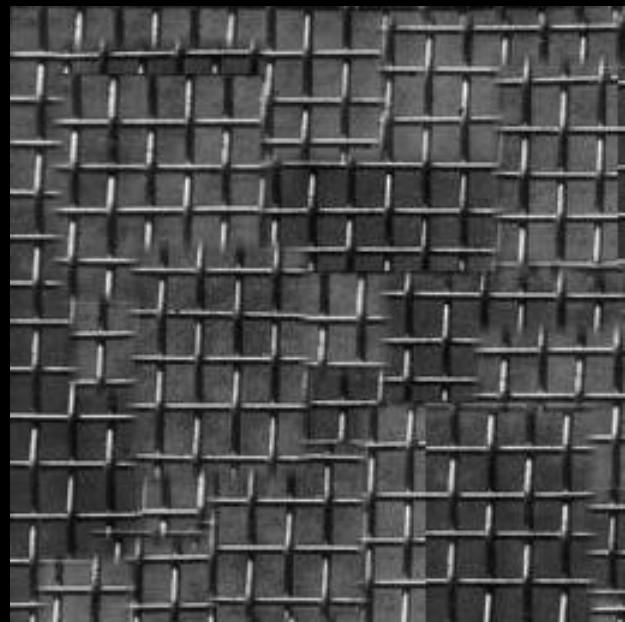




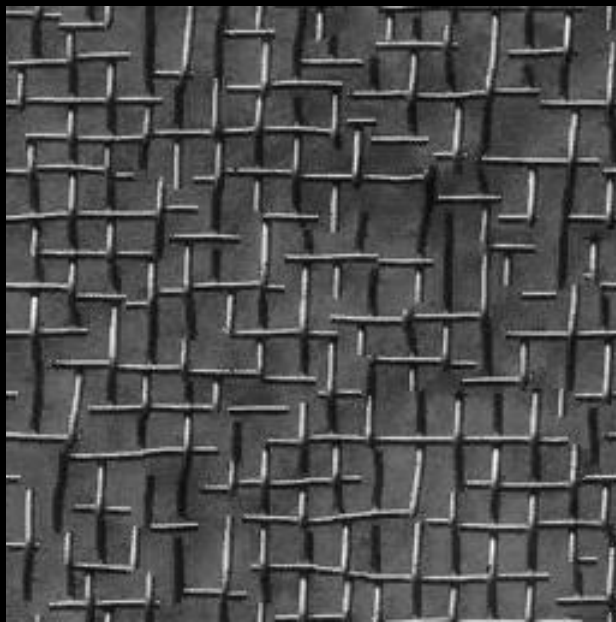
input image



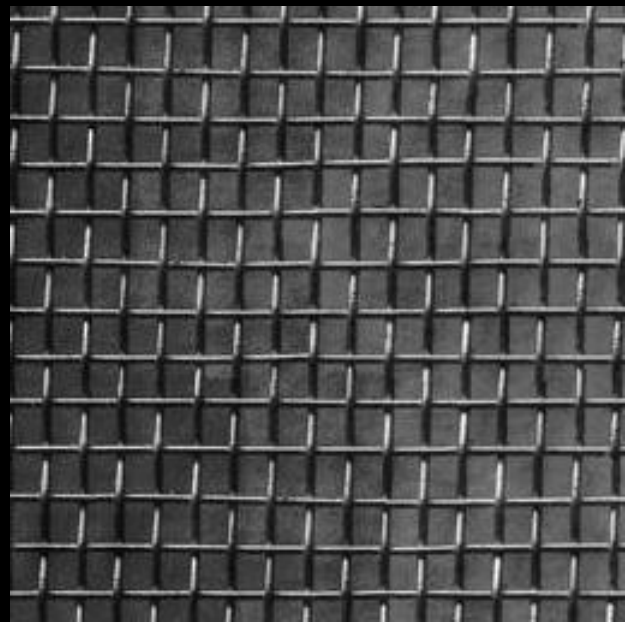
Portilla & Simoncelli



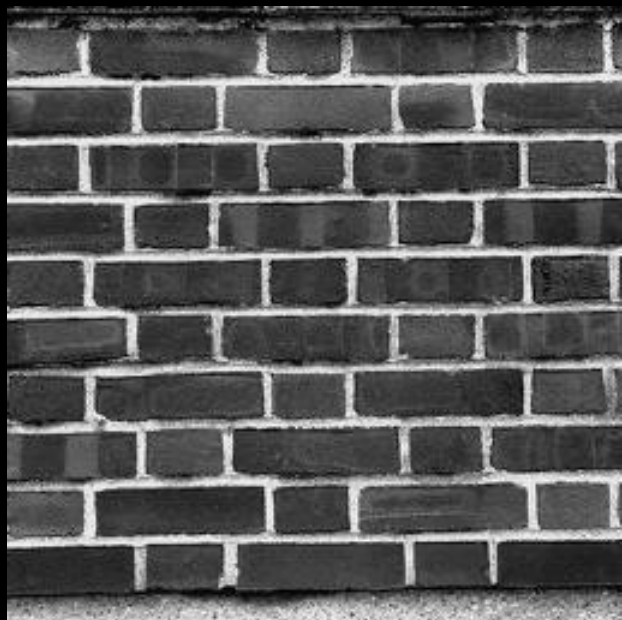
Xu, Guo & Shum



Wei & Levoy



Our algorithm



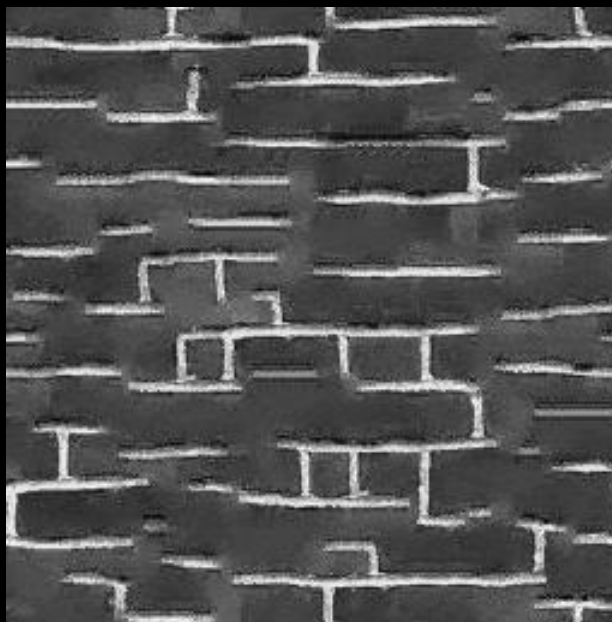
input image



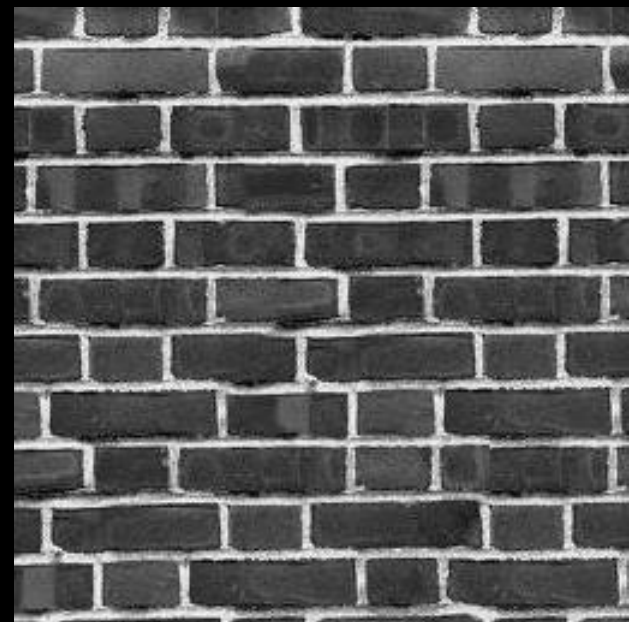
Portilla & Simoncelli



Xu, Guo & Shum



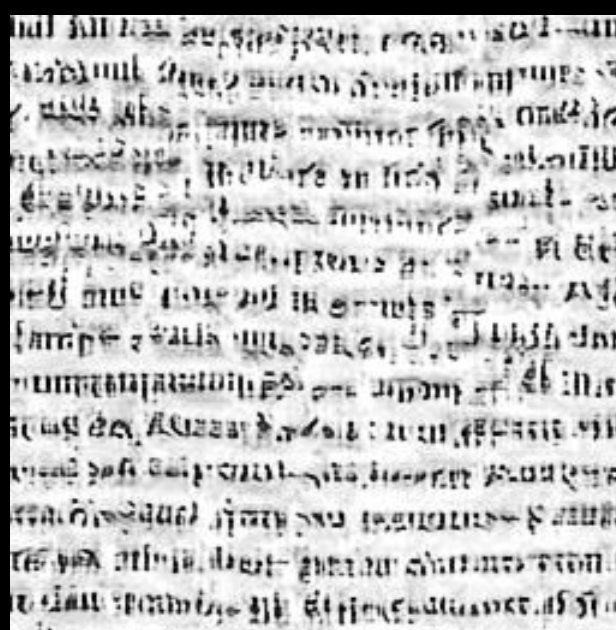
Wei & Levoy



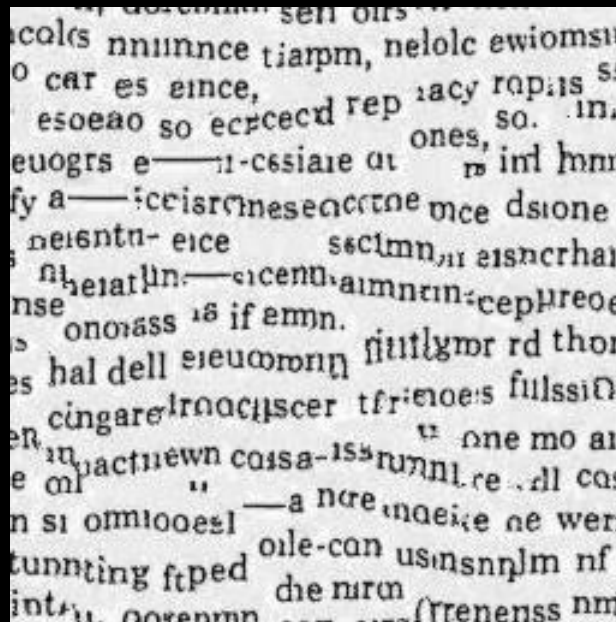
Our algorithm

...id 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
escribe the wealth of simple-cell recep
ad neurophysiologically¹⁻³ and inferred
especially if such a framework has the
it helps us to understand the functio
eeper way. Whereas no generic mo
ussians (DOG), difference of offset C
rivative of a Gaussian, higher derivati
function, and so on—can be expecte
imple-cell receptive field, we noneth

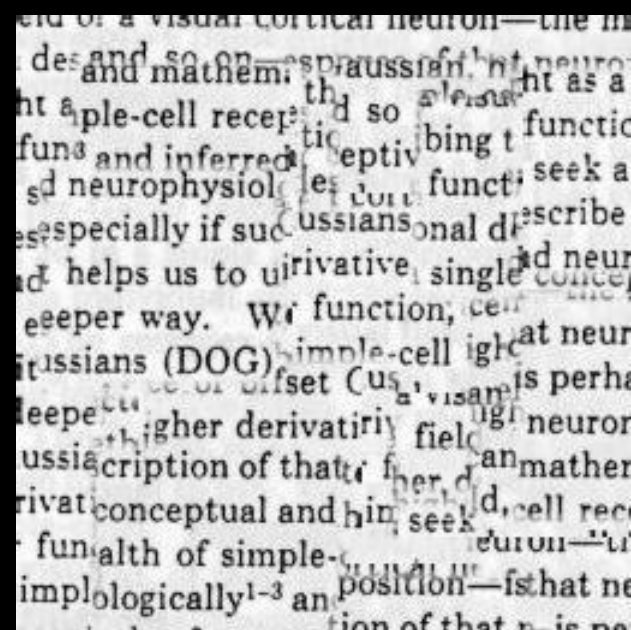
input image



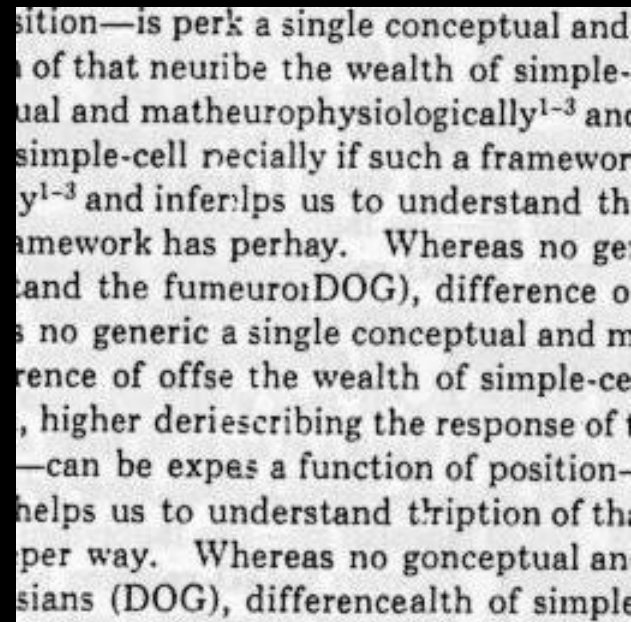
Portilla & Simoncelli



Wei & Levoy



Xu, Guo & Shum



Our algorithm

Application: Texture Transfer

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



Texture Transfer



Constraint

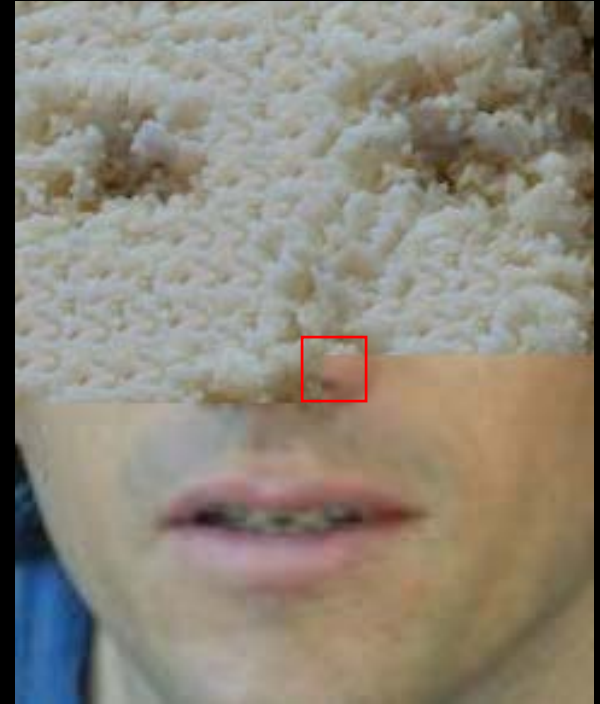


Texture sample



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. Similarity to the image being “explained”



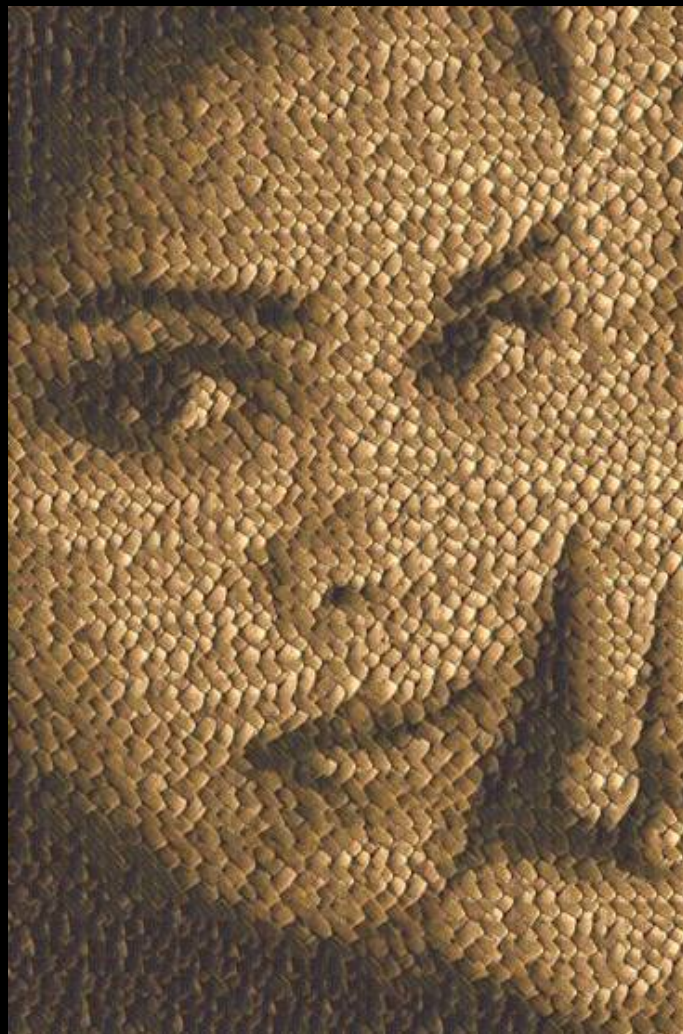


Image Analogies

Aaron Hertzmann^{1,2}

Chuck Jacobs²

Nuria Oliver²

Brian Curless³

David Salesin^{2,3}

¹**New York University**

²**Microsoft Research**

³**University of Washington**

Image Analogies



A



A'



B



B'



Image Analogies

Goal: Process an image by example



A

■
■



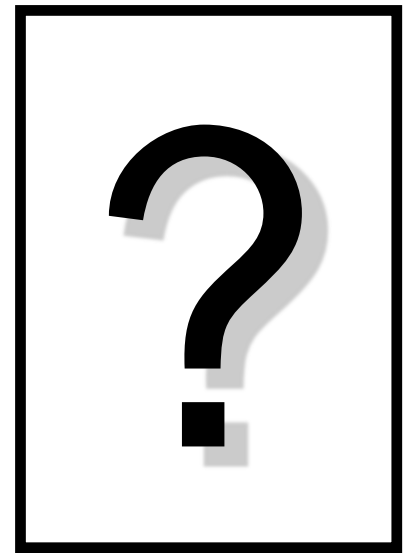
A'

■ ■
■ ■



B

■
■



B'

Non-parametric sampling



A

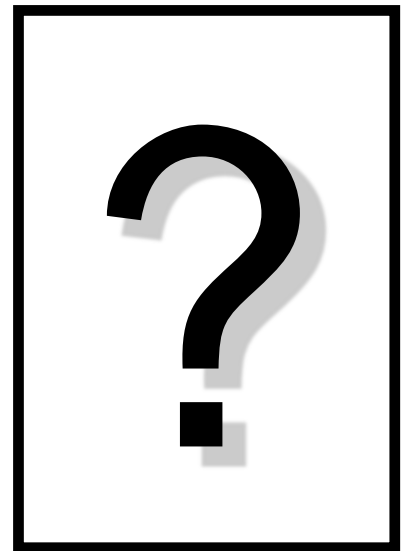


A'

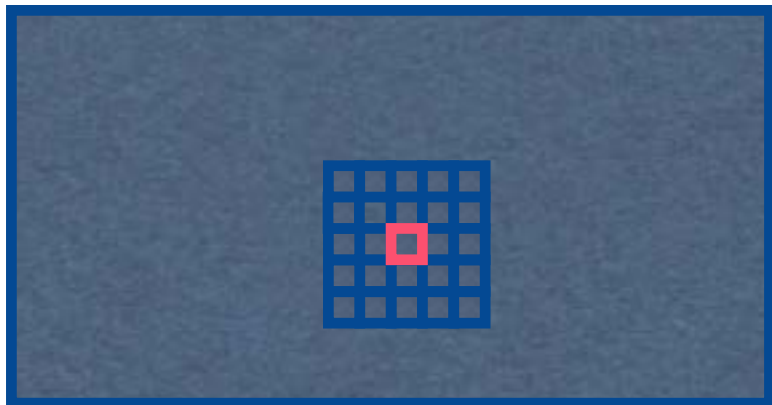
⋮



B

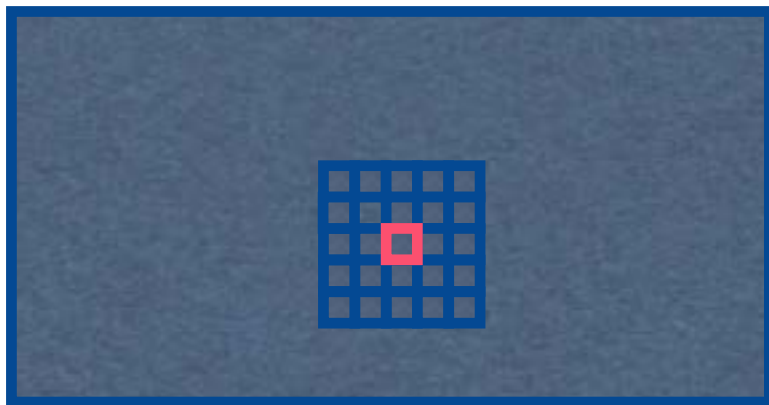


B'



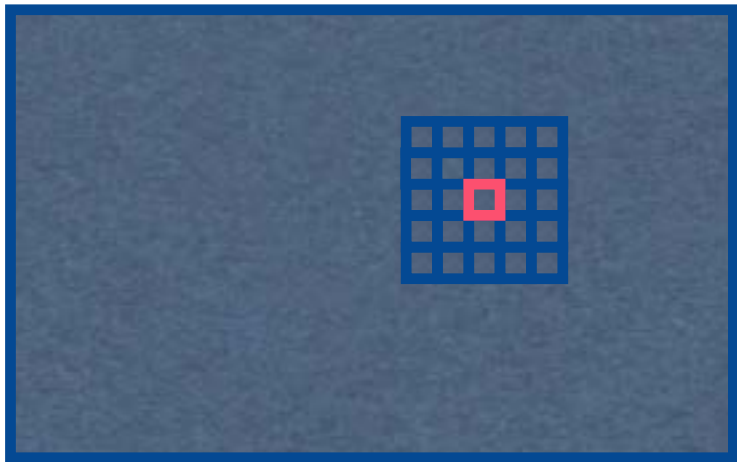
A

⋮



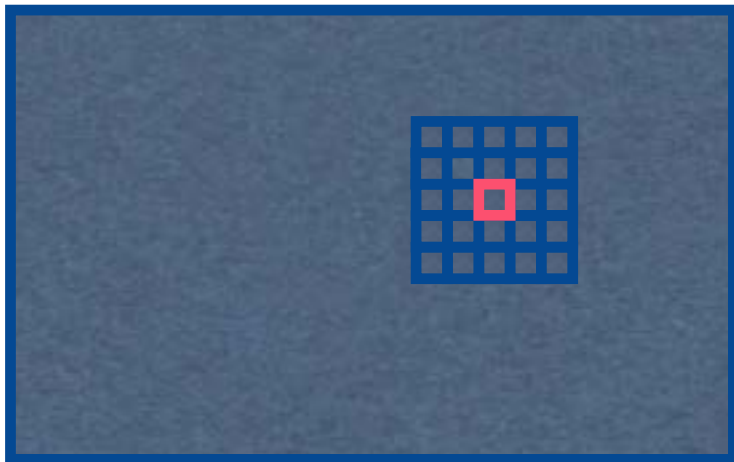
⋮

A'

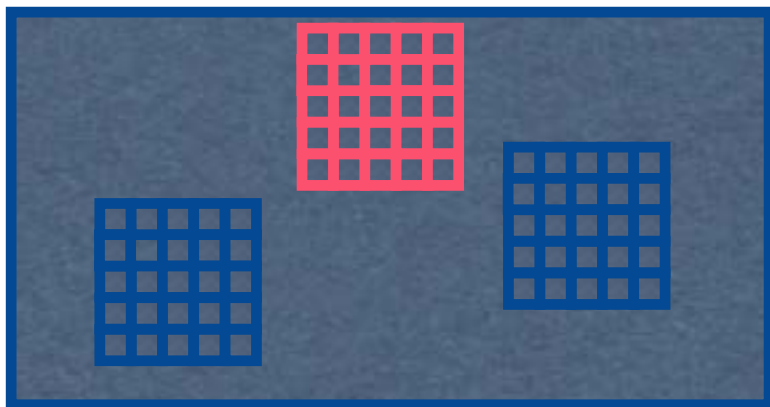


B

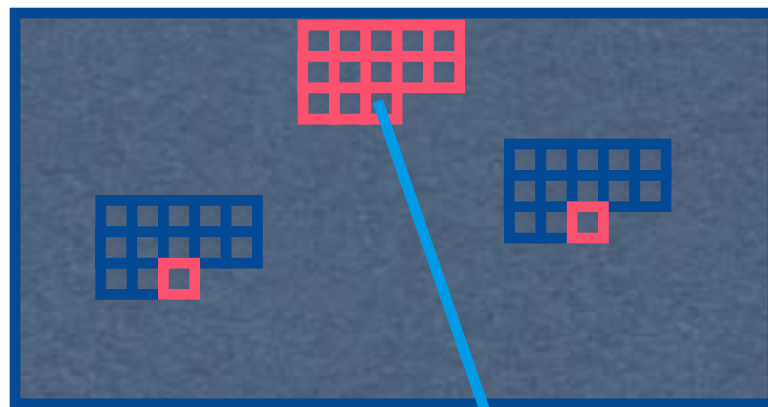
⋮



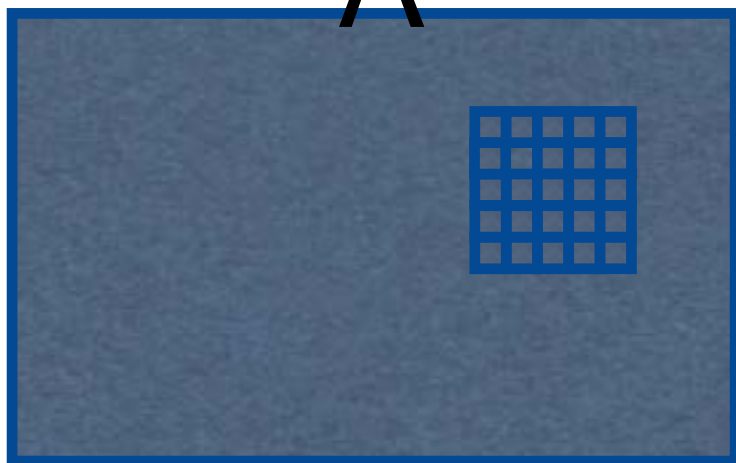
B'



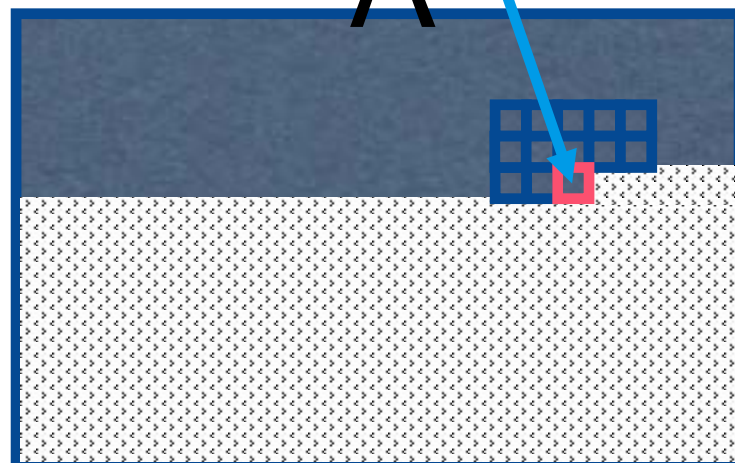
A



A'



B



B'

Blur Filter



Unfiltered source (A)



Filtered source (A')



Unfiltered target (B)

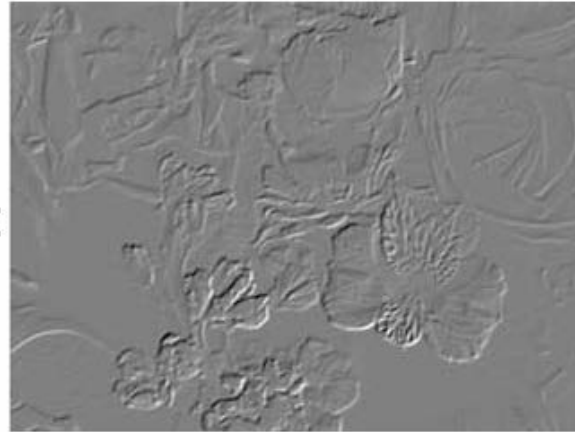


Filtered target (B')

Edge Filter



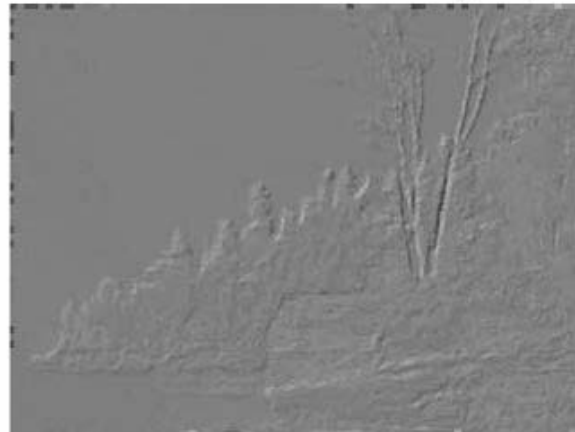
Unfiltered source (A)



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')

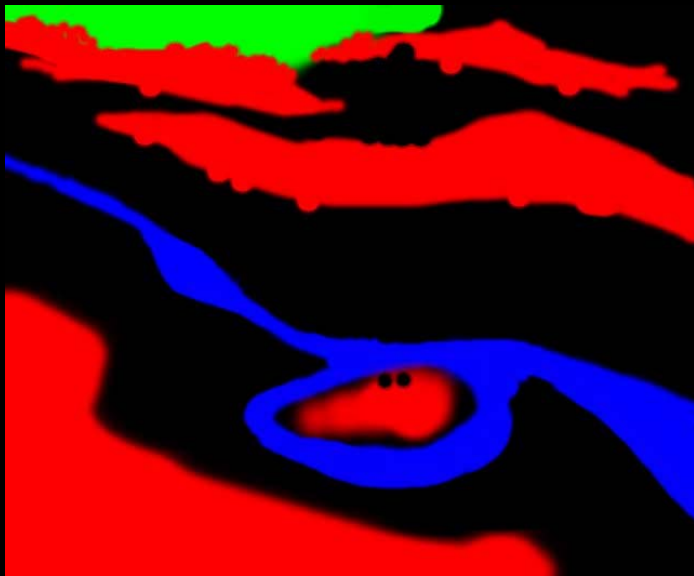
Texture-by-numbers



A



A'



B



B'

Super-resolution



A



A'

Super-resolution (result!)



B



B'