Texture Synthesis

Image Manipulation and Computational Photography
CS294-69 Fall 2011
Maneesh Agrawala

[Some slides from James Hays , Derek Hoiem, Alexei Efros and Fredo Durand]

Fourier Transform

To understand frequency ω let's reparametrize the signal by ω:

f(x) → Fourier Transform → ***F(ω)***

For every ω from 0 to inf, ***F(ω)*** holds the amplitude *A* and phase ϕ of the corresponding sine $A\sin(\omega x + \phi)$

How can *F* hold both? Complex number trick!

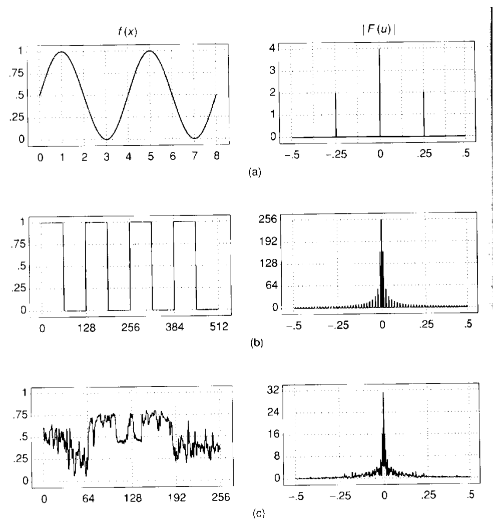
$$F(\omega) = R(\omega) + iI(\omega)$$

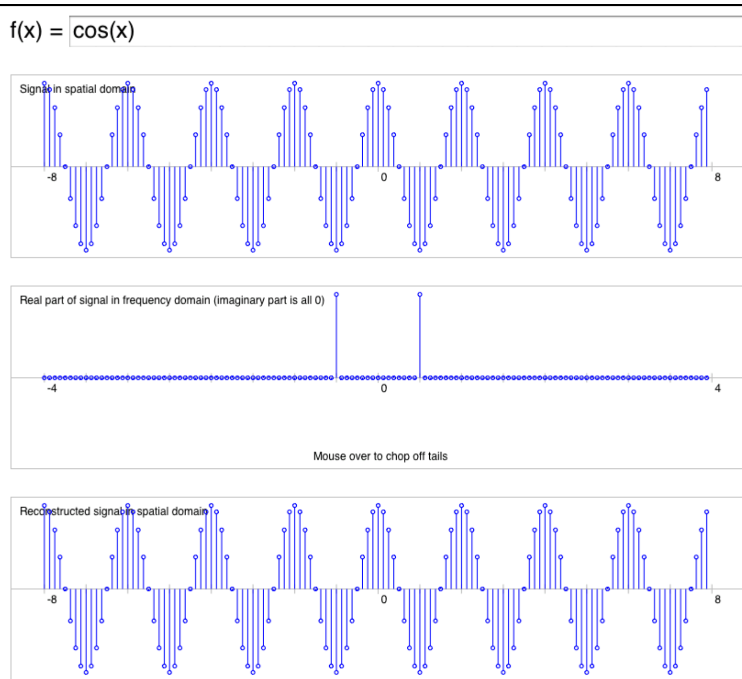
$$A = \pm\sqrt{R(\omega)^2 + I(\omega)^2} \qquad \phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$$

We can always go back:

F(ω) → Inverse Fourier Transform → ***f(x)***

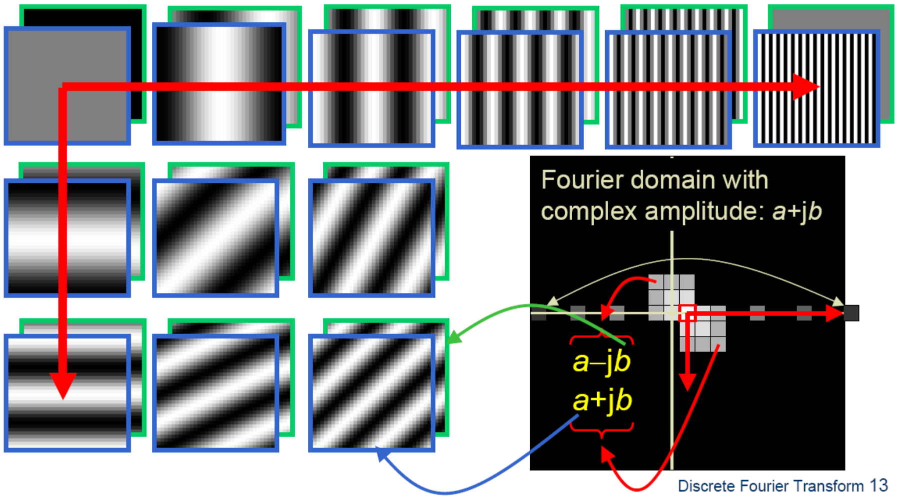
Frequency Spectra





http://madebyevan.com/dft/

Extension to 2D



in Matlab, check out: imagesc(log(abs(fftshift(fft2(im)))));

Due: A0 Hybrid Images

Robin Gaestel

Weather Forecasting for Dummies™

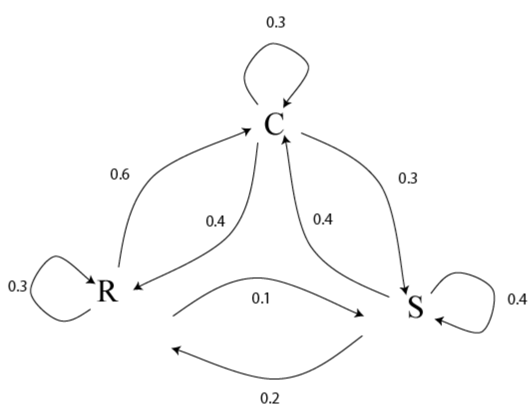
Let's predict weather:

- 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 yesterday'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"*

Topics

Video Texture

Synthesizing Image Textures

Video Textures

Arno Schödl
Richard Szeliski
David Salesin
Irfan Essa

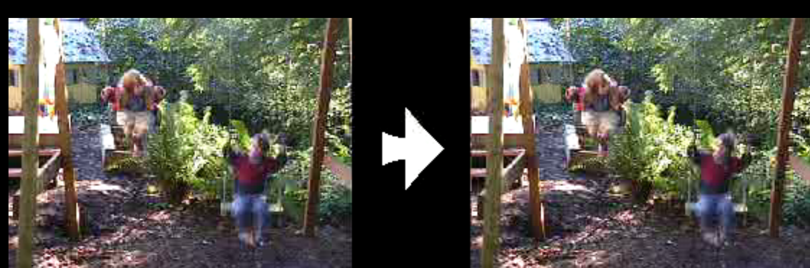
Microsoft Research, Georgia Tech

Still photos

Video clips

Video textures

Problem statement



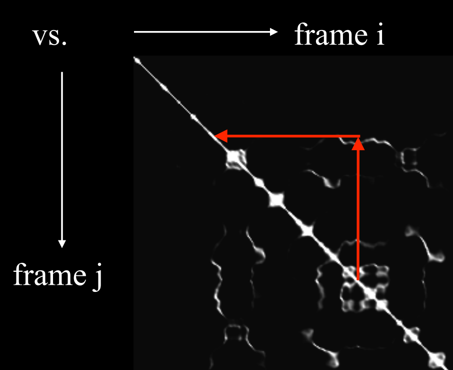
video clip video texture

Our approach

How do we find good transitions?

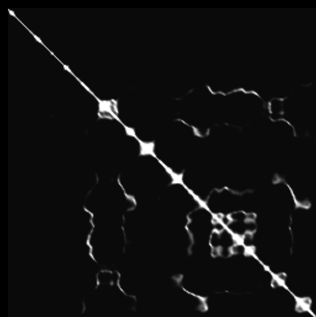
Finding good transitions

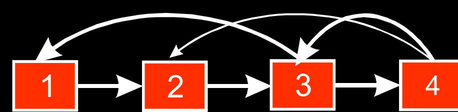
Compute L_2 distance $D_{i,j}$ between all frames



Similar frames make good transitions

Markov chain representation



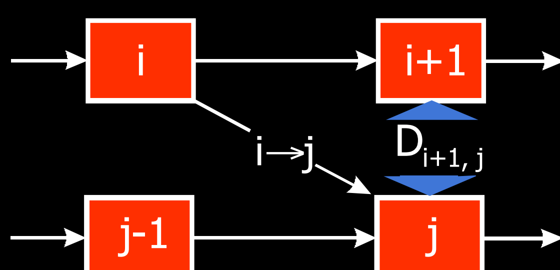


Similar frames make good transitions

Transition costs

Transition from i to j if successor of i is similar to j

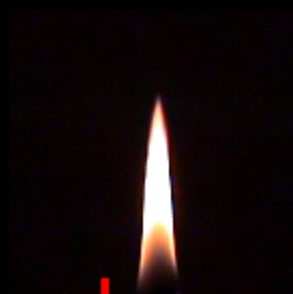
Cost function: $C_{i \to j} = D_{i+1,\, j}$



Transition probabilities

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

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

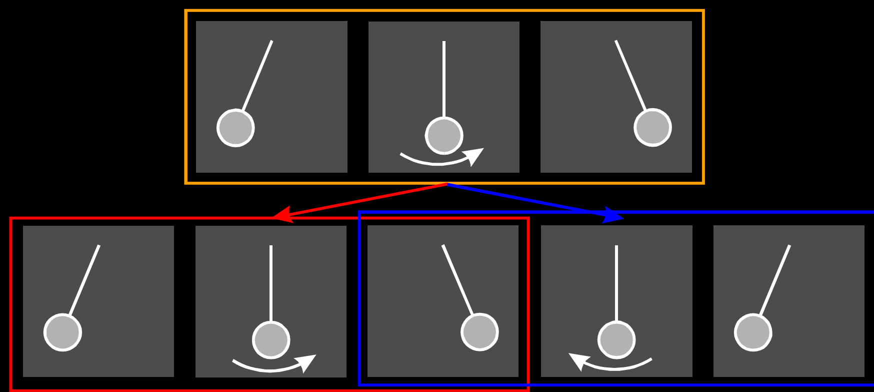


high σ

low σ

Preserving dynamics

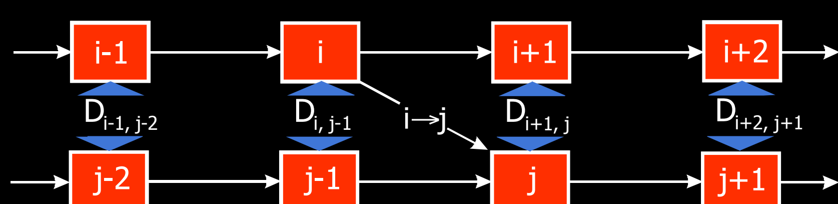
Preserving dynamics



Preserving dynamics

Cost for transition $i \to j$

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



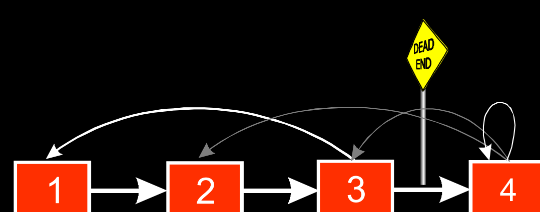
Preserving dynamics – effect

Cost for transition $i \to j$

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

Dead ends

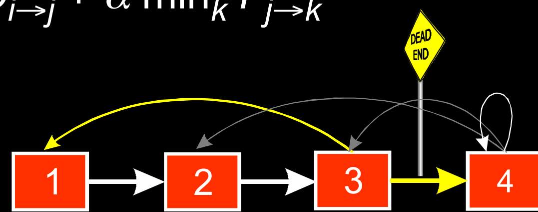
No good transition at the end of sequence



Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

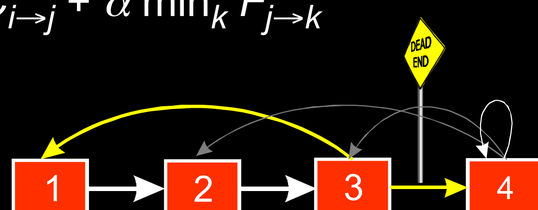
$$F_{i\to j} = C_{i\to j} + \alpha \min_k F_{j\to k}$$



Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

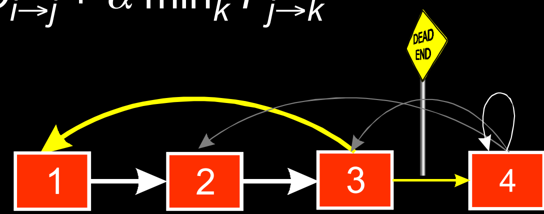
$$F_{i\to j} = C_{i\to j} + \alpha \min_k F_{j\to k}$$



Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

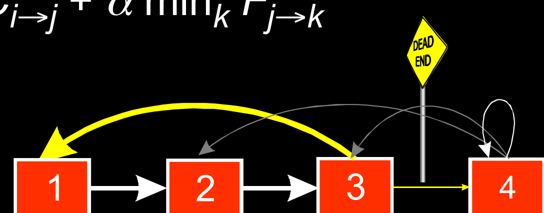
$$F_{i \to j} = C_{i \to j} + \alpha \min_k F_{j \to k}$$



Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

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

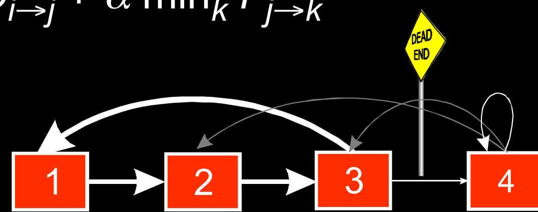


Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

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

- Q-learning



Future cost – effect

Video portrait

Useful for web pages

Region-based analysis

- Divide video up into regions

- Generate a video texture for each region

Automatic region analysis

User-controlled video textures

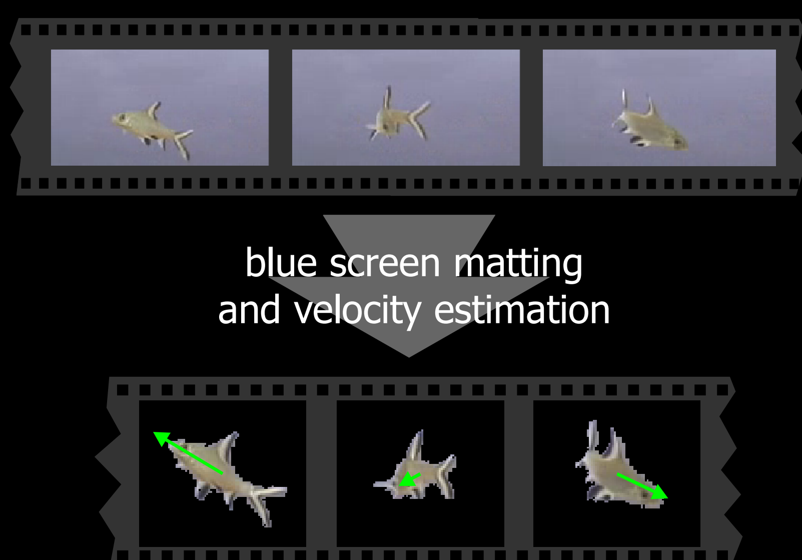
slow variable fast

User selects target frame range

Video-based animation

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

Video sprite extraction



Video sprite control

- Augmented transition cost:

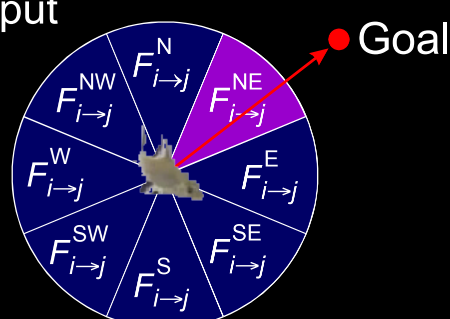
$$C^{\text{Animation}}_{i\to j} = \alpha \underbrace{C_{i\to j}}_{\text{Similarity term}} + \beta \underbrace{\text{angle}}_{\text{Control term}}$$

vector to mouse pointer

velocity vector

Video sprite control

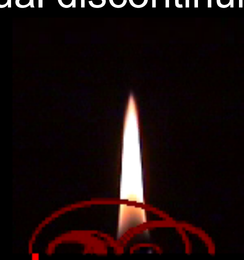
- Need future cost computation
- Precompute future costs for a few angles.
- Switch between precomputed angles according to user input
- [GIT-GVU-00-11]



Interactive fish

Summary

- Video clips → video textures
  - define Markov process
  - preserve dynamics
  - avoid dead-ends
  - disguise visual discontinuities



Discussion

- Some are hard

Panoramic Video Textures

Agarwala et al. SIGGRAPH 05

http://www.youtube.com/watch?v=vS6Dz-8_NjY

"Amateur" by Lasse Gjertsen

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

Michel Gondry train video

http://www.youtube.com/watch?v=ssJutXkpSIY

Image Texture

Texture depicts spatially repeating patterns

Many natural phenomena are textures

radishes

rocks

yogurt

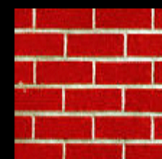
Texture Synthesis

Goal: create new samples of a given texture

Applications: virtual environments, hole-filling, texturing surfaces, …

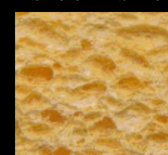
The Challenge

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



repeated

stochastic



Both?

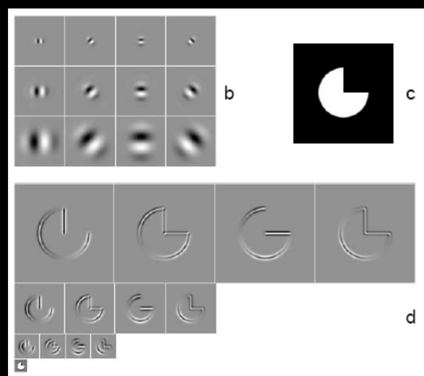
Heeger Bergen 1995

Seminal paper that introduced texture synthesis to the graphics community

Algorithm:

- Initialize *J* to noise
- Create multiresolution pyramids for *I* and *J*
- Match the histograms of *J*'s pyramid levels with *I*'s pyramid levels
- Loop until convergence
- Can be generalized to 3D

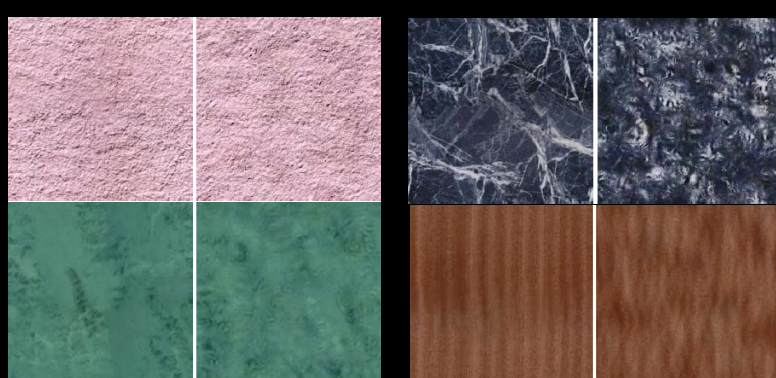
Heeger Bergen 1995 - Algorithm



- Image pyramids
  - Gaussian
  - Laplacian
- Steerable pyramids [SimoncelliFreeman95]
  - b): multiple scales of oriented filters
  - c): a sample image
  - d): results of filters in b) applied to c)

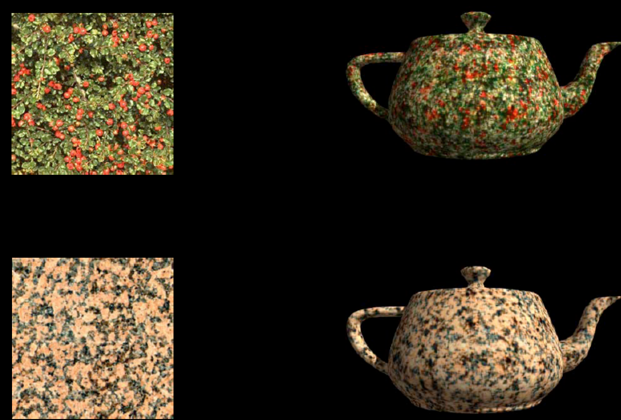
Heeger Bergen 1995 - Results

Successes Failures



I *J* *I* *J*

Heeger Bergen 1995 - Results



Heeger Bergen 1995 - Verdict

- Texture model:
  - Histograms of responses to various filters
- Avoiding copying:
  - Inherent in algorithm
- No user intervention required
- Captures stochastic textures well
- Does not capture structure
  - Lack of inter-scale constraints

De Bonet 1997

Propagate constraints downwards by matching statistics all the way up the pyramid

Feature vector: multiscale collection of filter responses for a given pixel

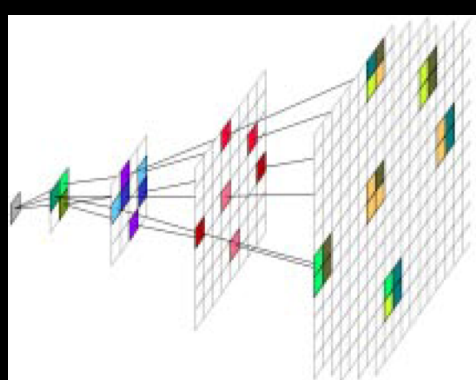
Algorithm:

– Initialize *J* to empty image

– Create multiresolution pyramids for *I* and *J*

– For each pixel in level of *J*, randomly choose pixel from corresponding level of *I* that has similar feature vector

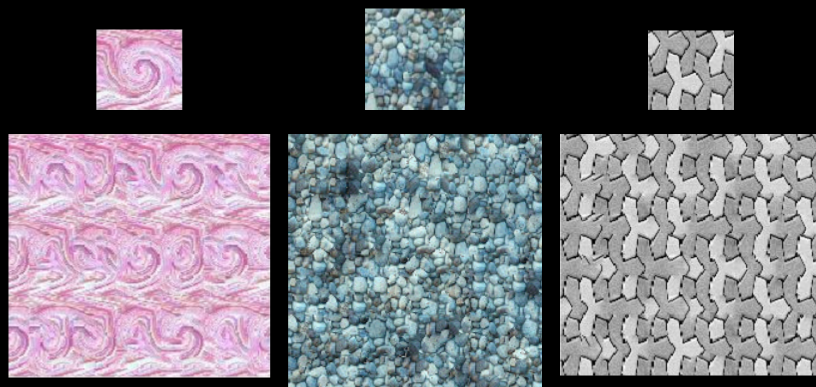
De Bonet 1997 - Algorithm



6 feature vectors shown

Notice how they share parent information

De Bonet 1997 - Results



De Bonet 1997 - Verdict

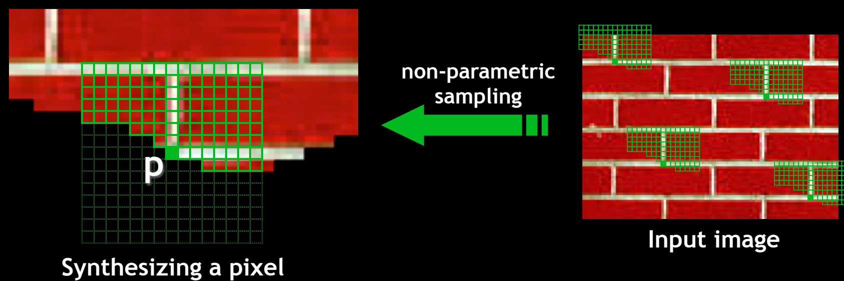
Texture model:
- Feature vector containing multiscale responses to various filters

Avoiding copying:
- Random choice of pixels with ‘close’ feature vectors, but copying still frequent on small scale

Individual per-filter thresholds cumbersome

Feature vectors used in later synthesis work

Efros & Leung 1999 - Algorithm



Assuming Markov property, compute P(**p**|N(**p**))

Building explicit probability tables infeasible

Instead, *search the input image* for all similar neighborhoods — that's the pdf for **p**

To sample from this pdf, just pick one match at random

Some Details

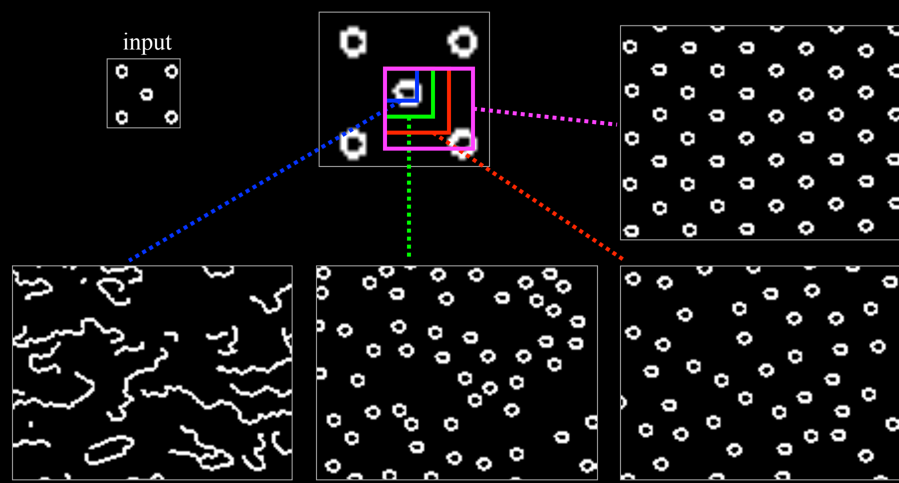
Growing is in "onion skin" order

- Pixels with most neighbors synthesized first
- If no close match found, the pixel is not synthesized until later

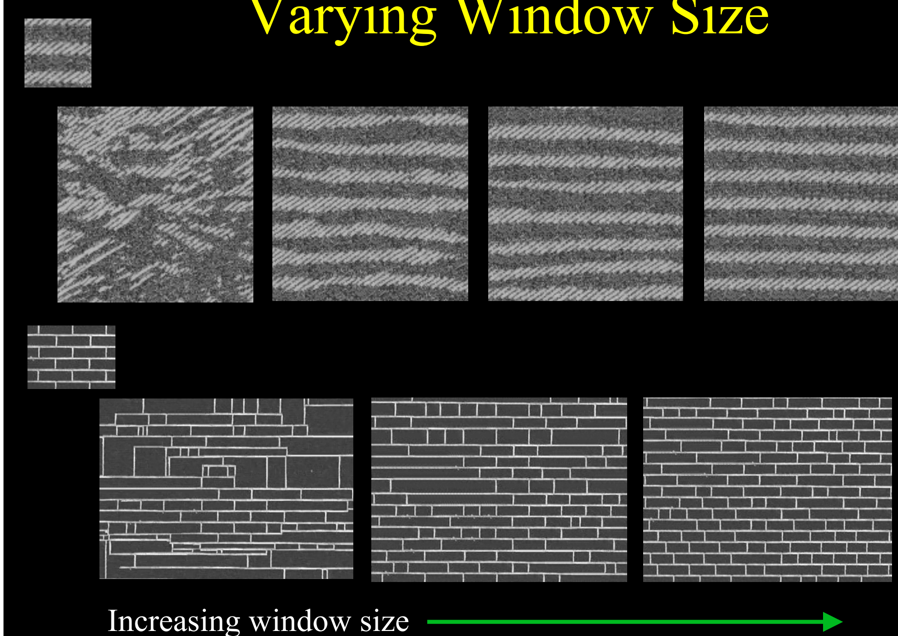
Using *Gaussian-weighted* SSD is very important

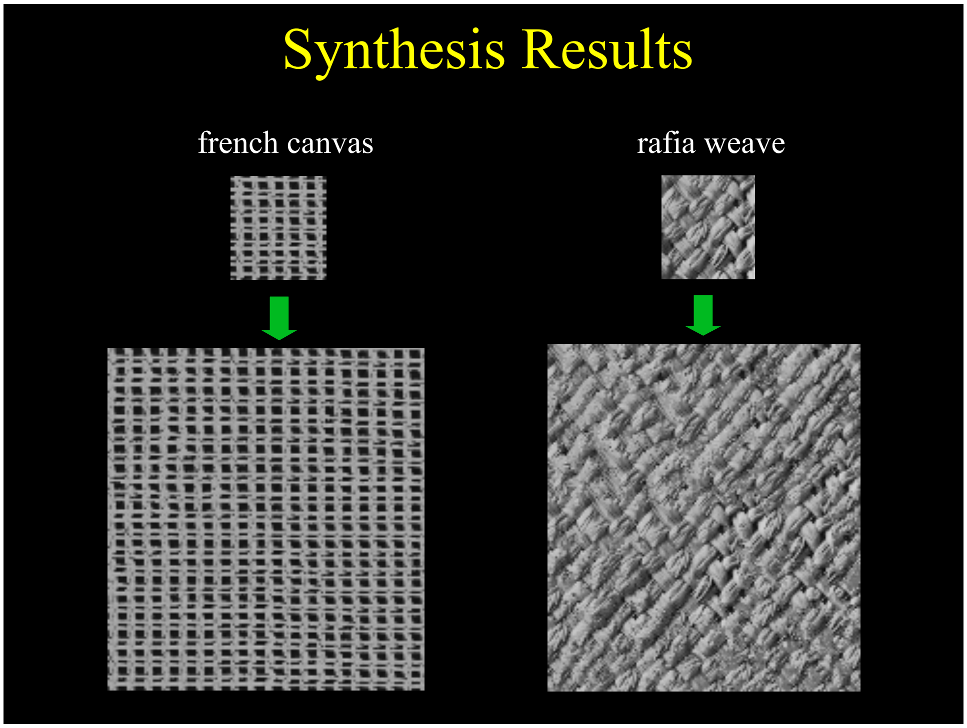
- to make sure the new pixel agrees with its closest neighbors
- Approximates reduction to a smaller neighborhood window if data is too sparse

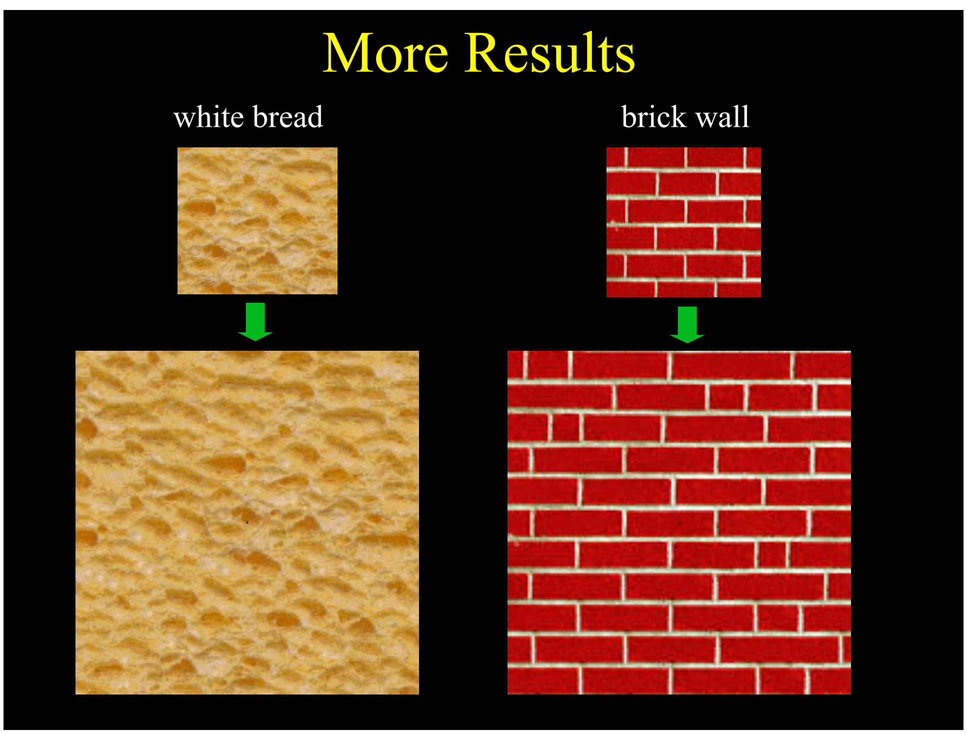
Neighborhood Window



Varying Window Size

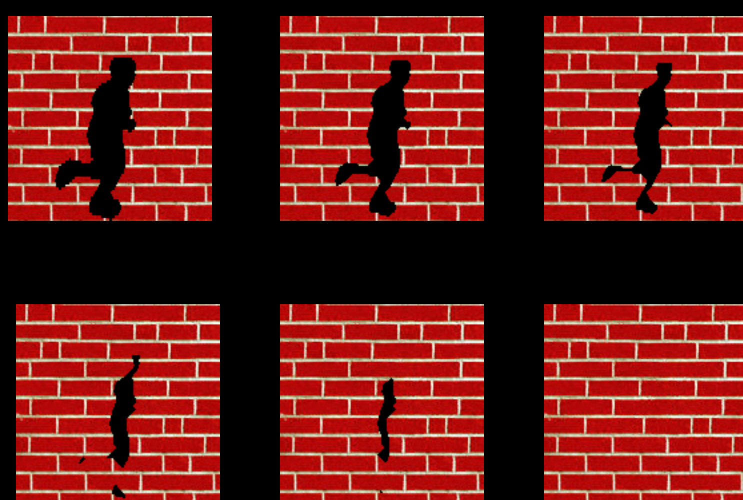



Synthesis Results
french canvas
rafia weave

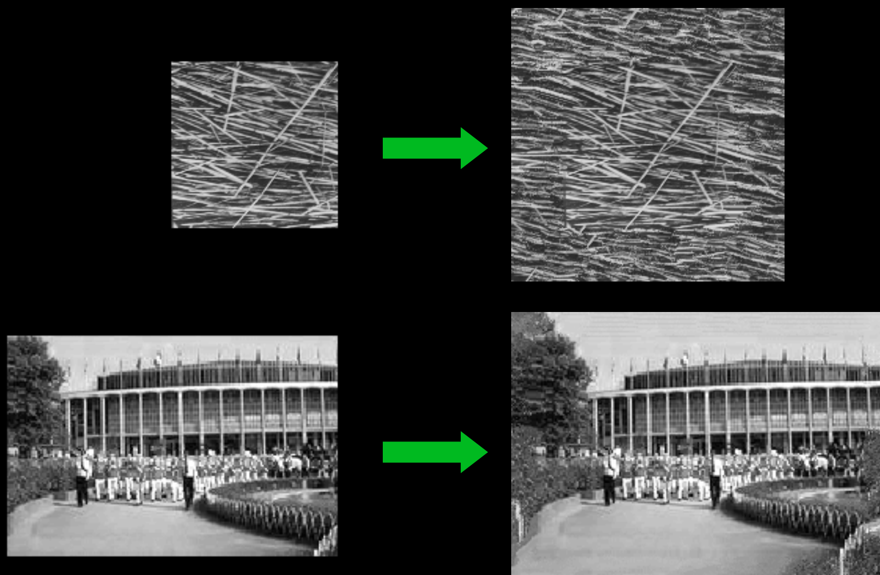

More Results
white bread
brick wall

Homage to Shannon

Hole Filling



Extrapolation



Efros Leung 1999 – Verdict

Texture model:
- MRF

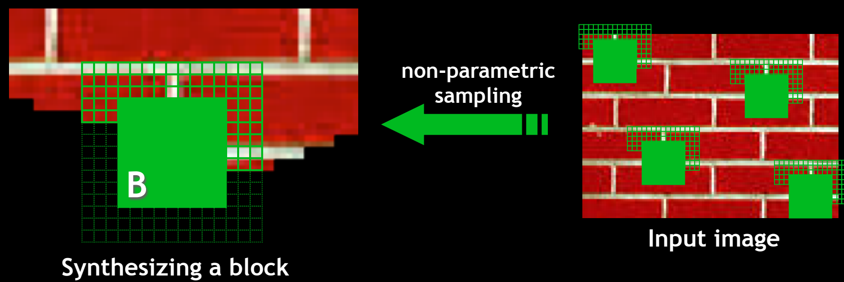
Avoiding copying:
- MRF

Neighborhood size = largest feature size

Markov model is surprisingly good
- "I spent an interesting evening recently with a grain of salt."

Search is very slow with large neighborhoods

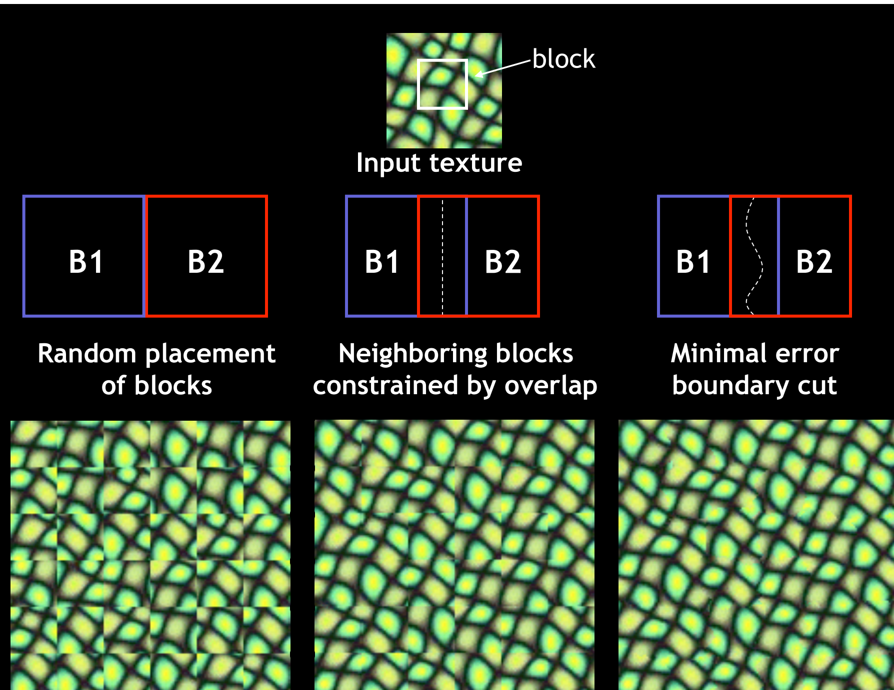
Image Quilting [Efros & Freeman]



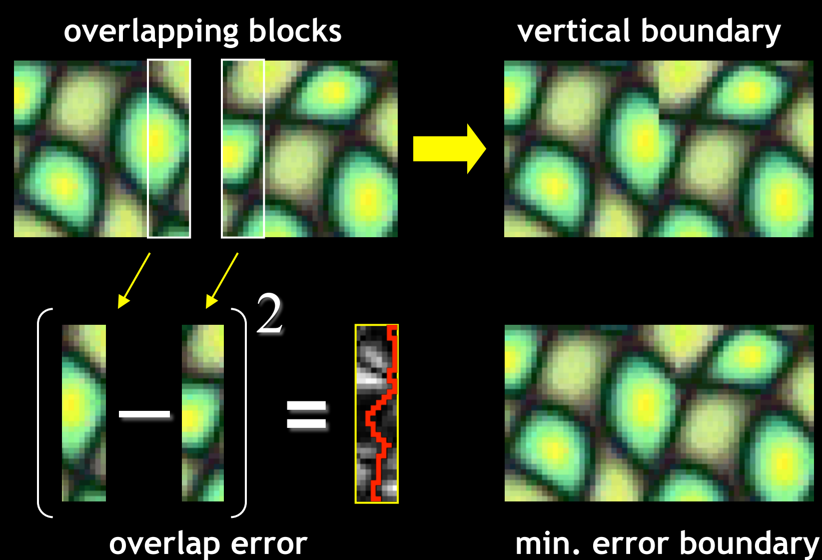
Observation: neighbor pixels are highly correlated

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!



Minimal error boundary



Philosophy

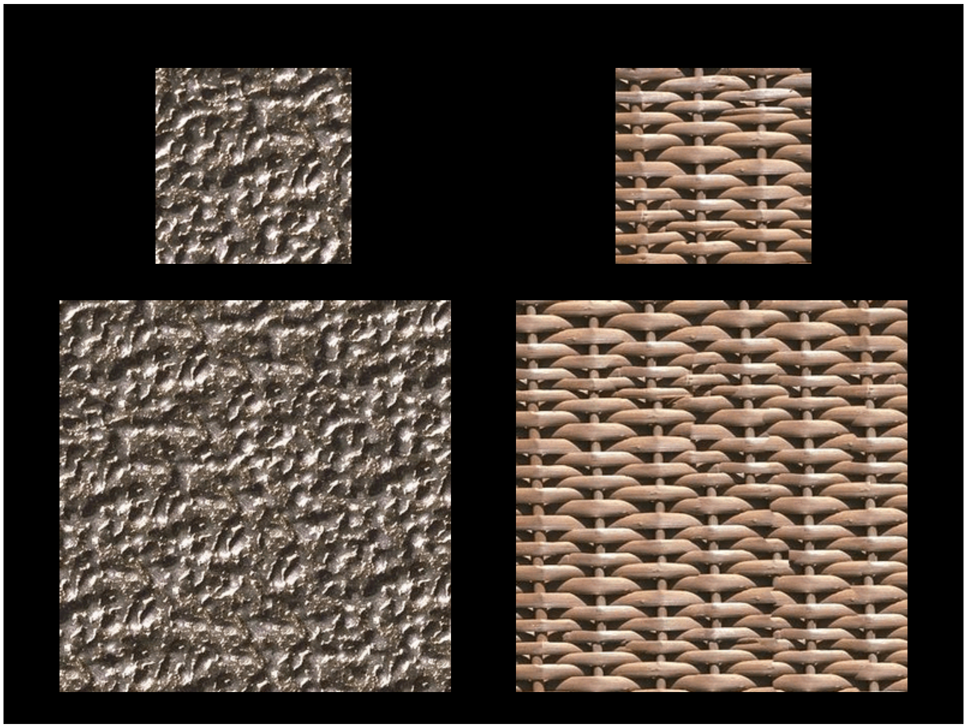
The "Corrupt Professor's Algorithm":

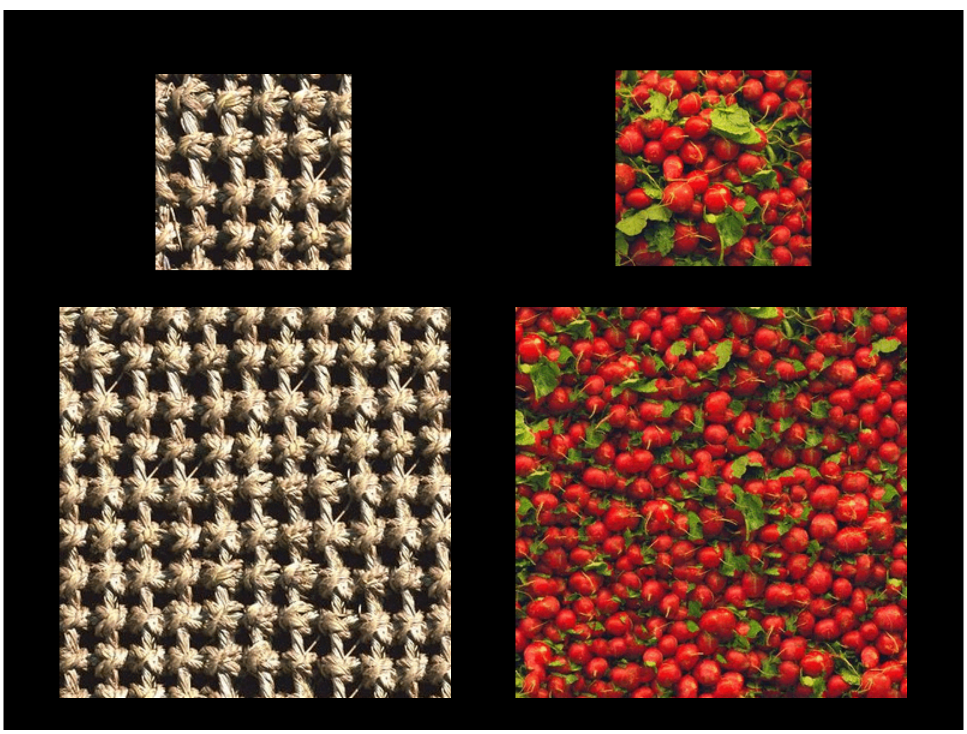
Plagiarize as much of the source image as you can

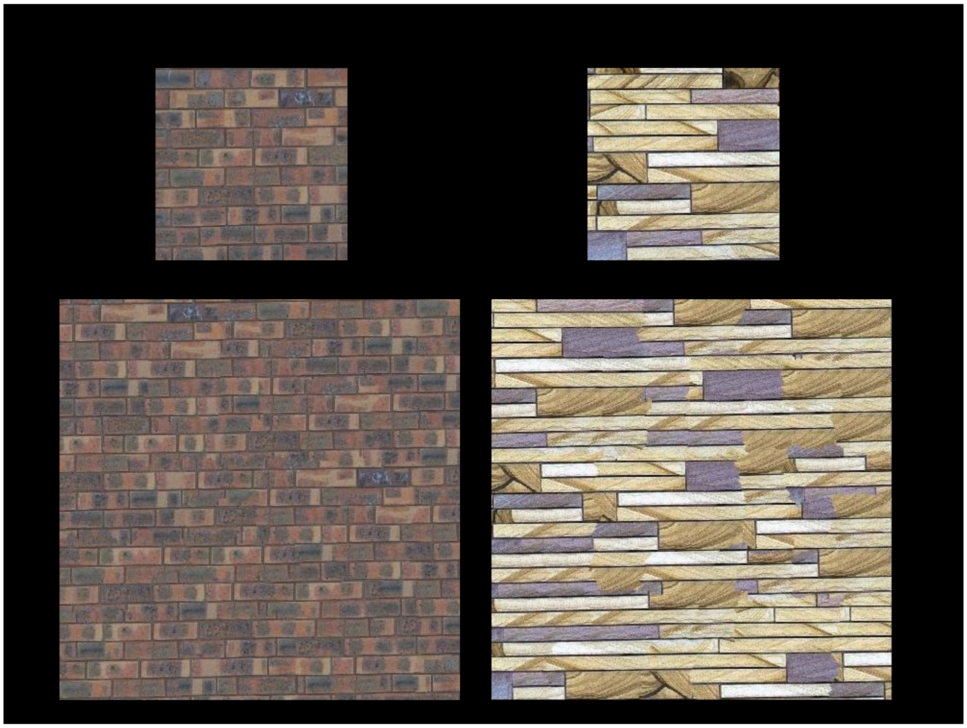
Then try to cover up the evidence

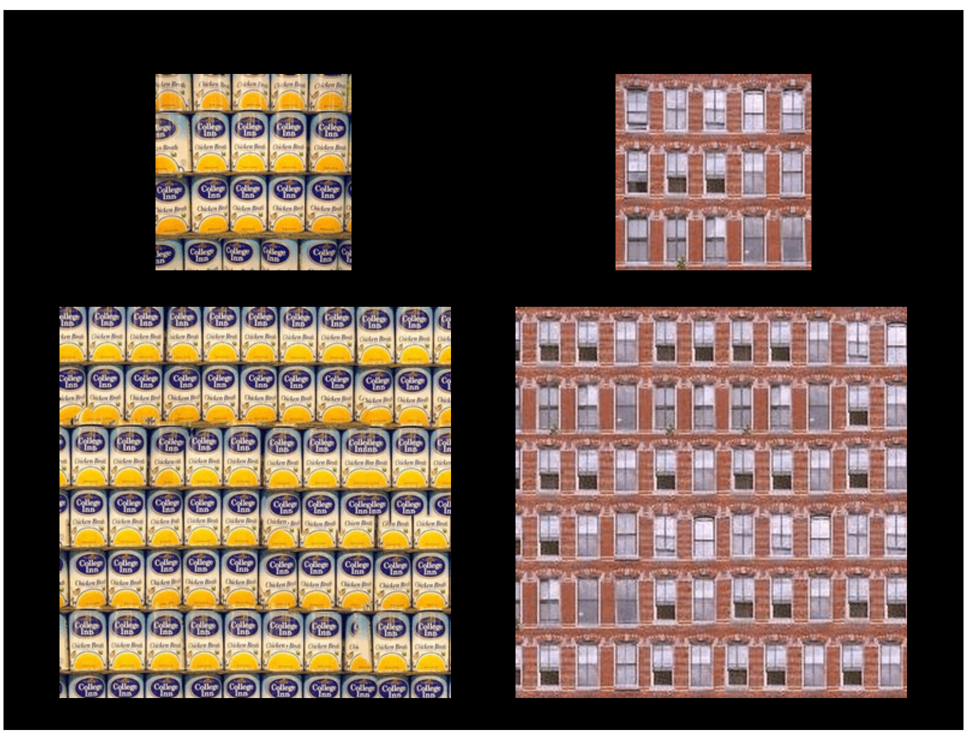
Rationale:

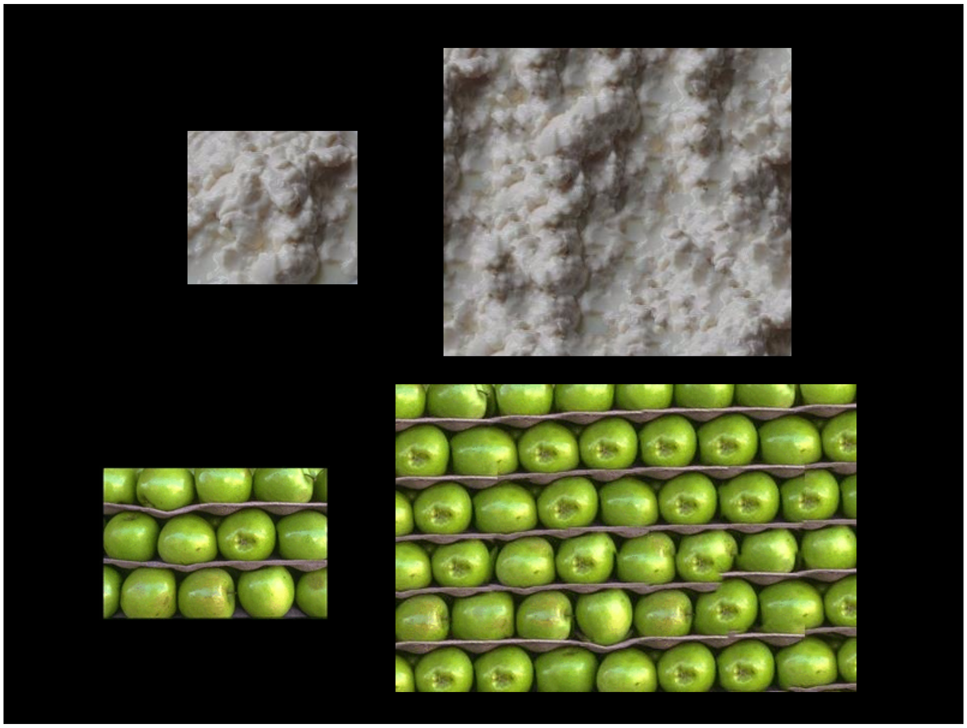
Texture blocks are by definition correct samples of texture so problem only connecting them together

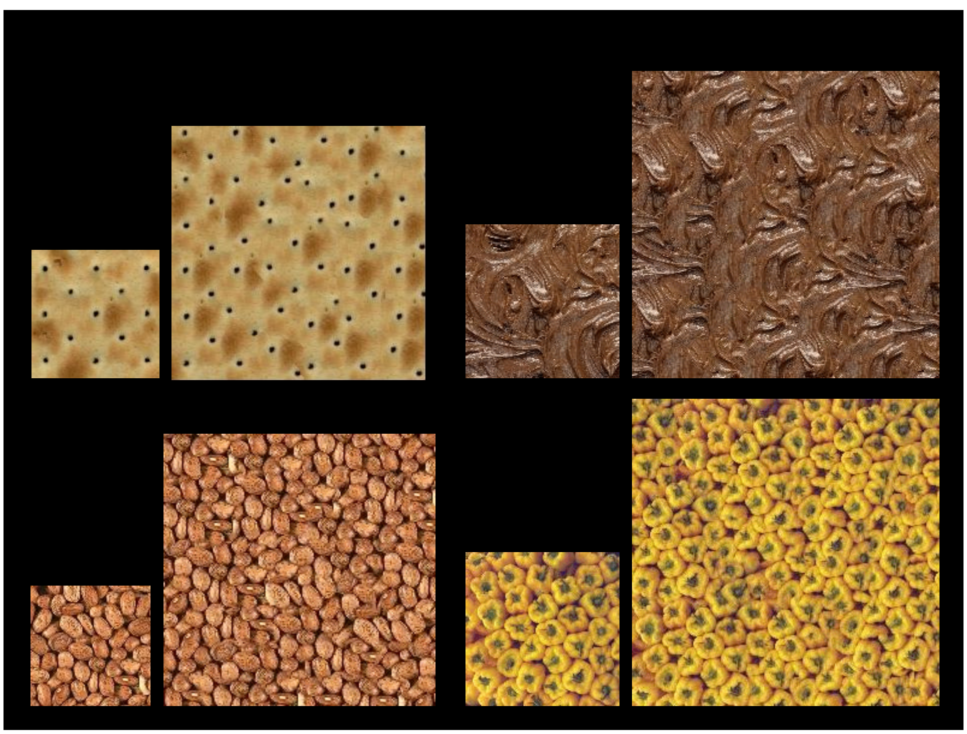












Efros Freeman 2001 - Verdict

Texture model:
- MRF

Avoiding copying:
- Randomized patch selection, but still noticeable

Patch size is a hard parameter to understand

Results are surprisingly good given algorithm

Multiscale goes on a brief hiatus

A1 Texture Synthesis

Due Mon Sep 26

Implement application of texture synthesis

Image analogies, Hole-filling, Patchmatch Structured hybrids ….

Adequate to implement, best solutions go beyond:

Every technique has some limitations (well written papers usually describe some of them). Develop techniques to address one or more limitations?

Sometimes different papers present different techniques for addressing the same problem Implement competing techniques and compare their strengths and weaknesses.

It may be possible to combine ideas from multiple papers to produce a new hybrid technique that addresses a new problem. Develop a new way to combine the texture synthesis techniques your have read about to solve a new problem.

1 person = 1 paper,

2 people = 1 paper + issue from list above or 2 papers,

3 people = 2 papers + issue from list above