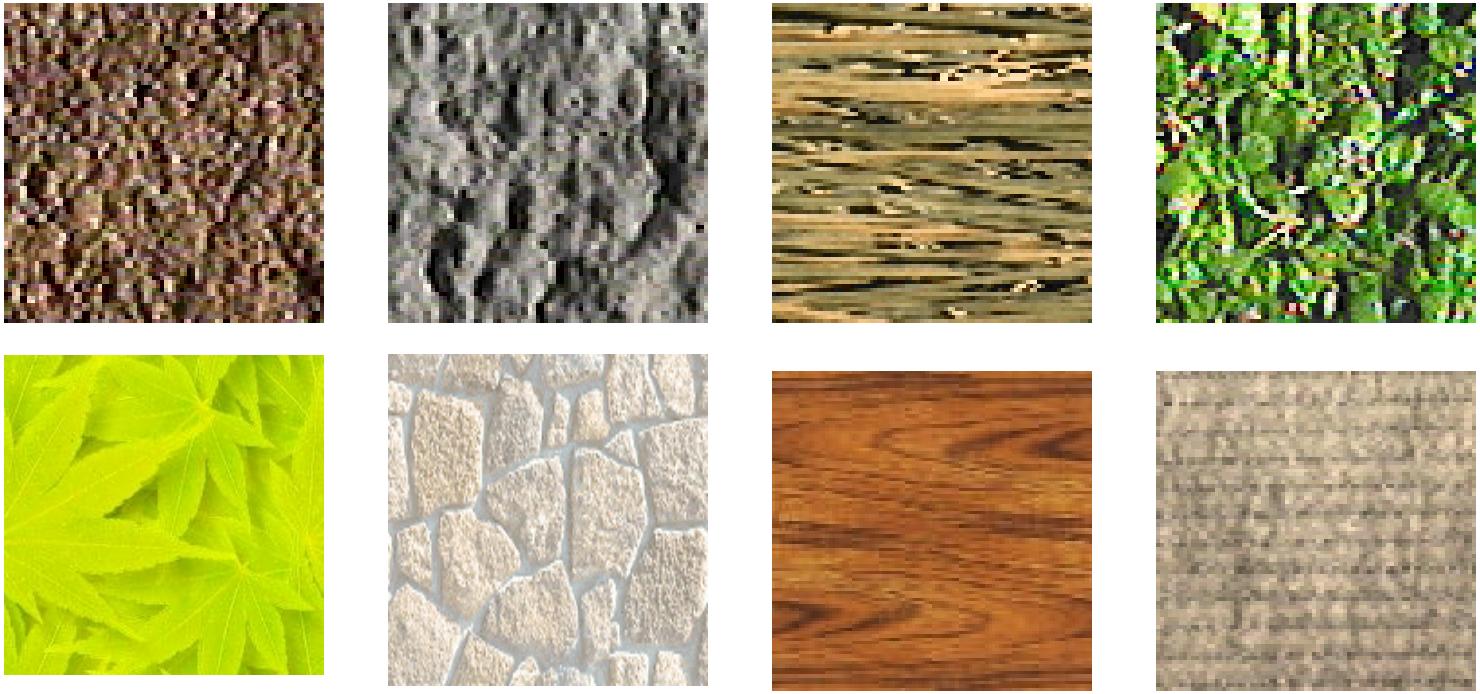
Texture

CS 554 – Computer Vision

Pinar Duygulu

Bilkent University

Texture



Easy to recognize, hard to define

Texture

Whether an effect is a texture or not depends on the scale at which it is viewed

A leaf that occupies most of the image – object

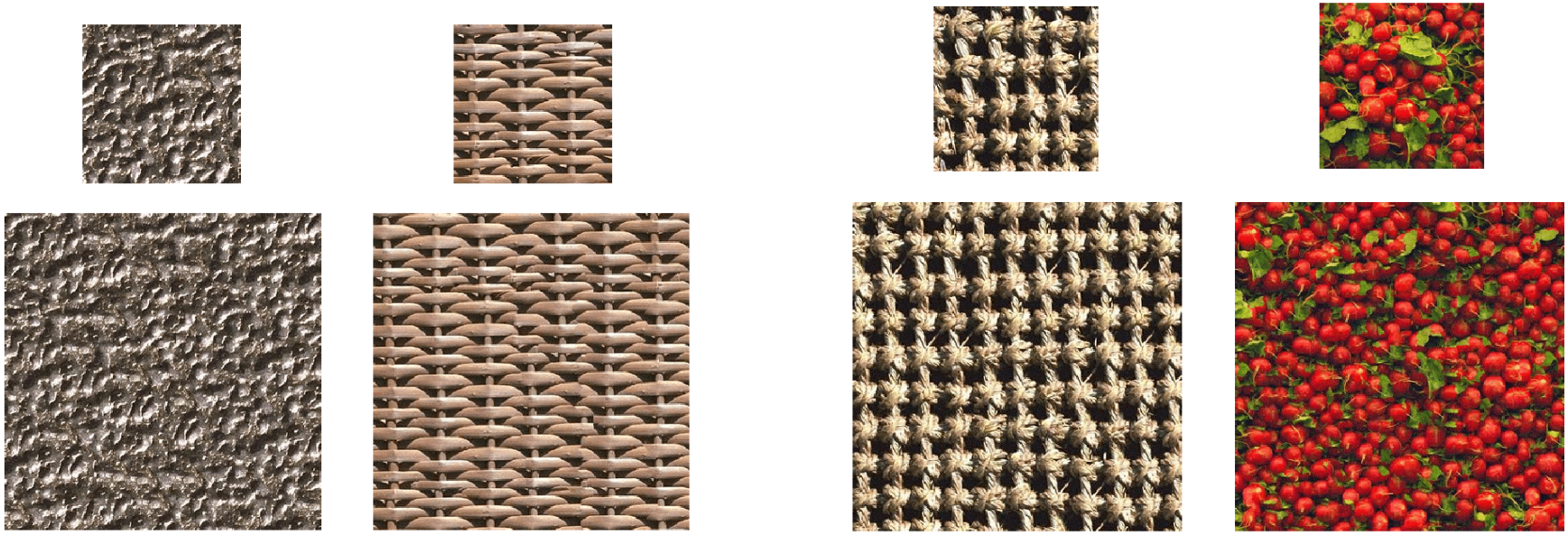
Foliage of a tree - texture

Texture

- Texture segmentation : segmenting an image into regions of constant texture
- Texture synthesis : construct large regions of texture from small example images
- Shape from texture : recovering surface orientation or shape from image texture
- Key issue: representing texture

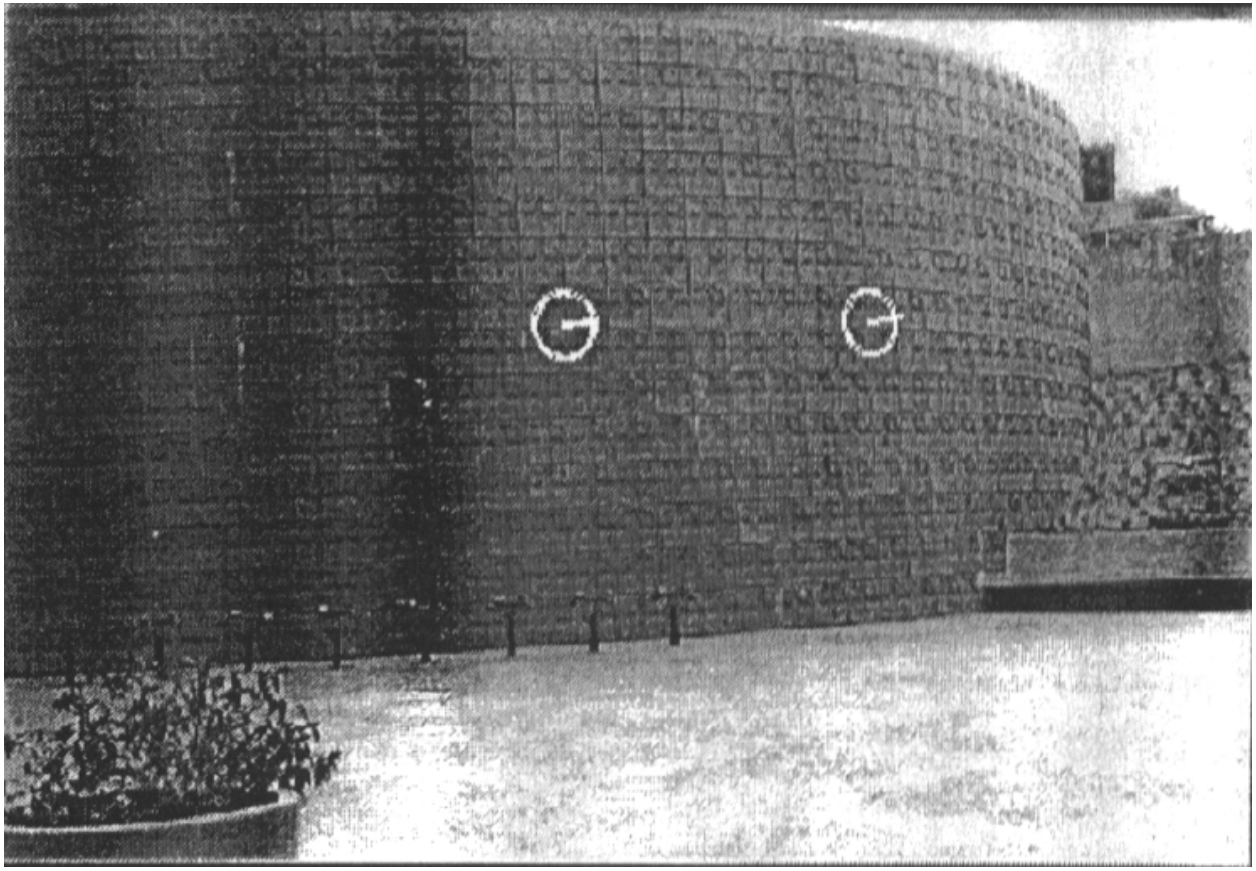
Texture Segmentation

Texture Synthesis



Efros & Freeman

Shape from Texture

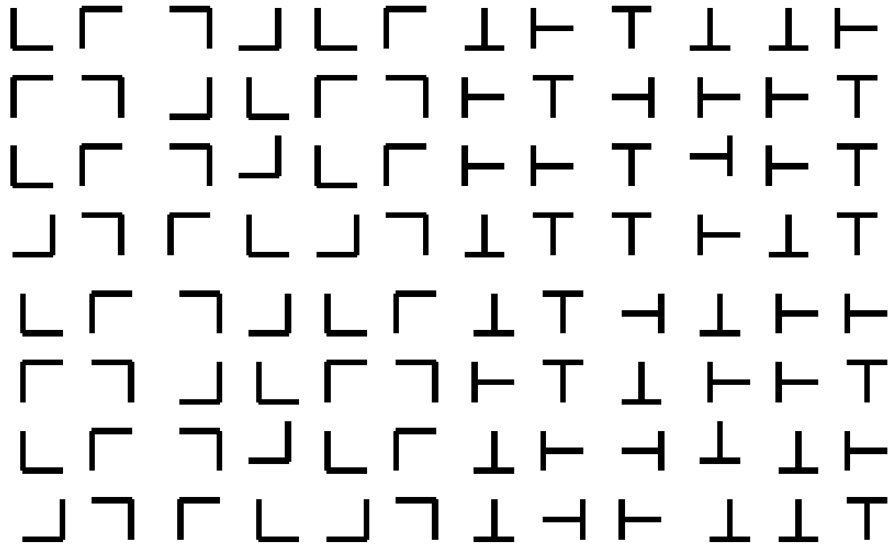


Texture looks same at different points on a surface

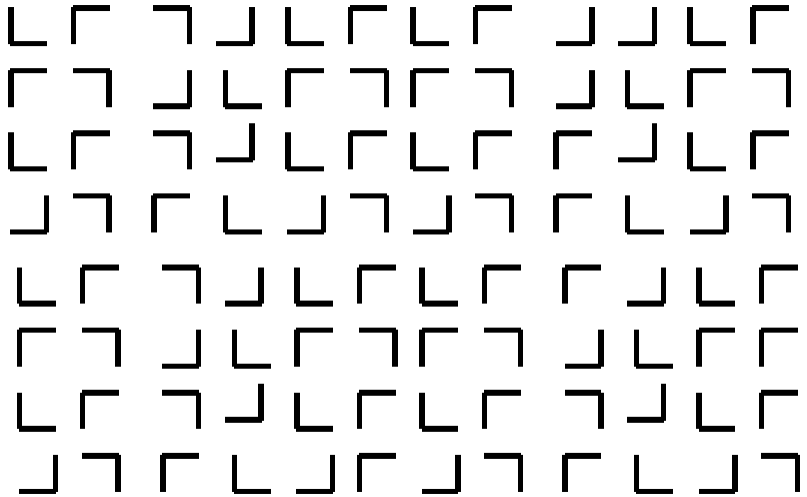
deformation is a cue to the shape of the surface

Adapted from David Forsyth, UC Berkeley

Pre-attentive Texture Discrimination



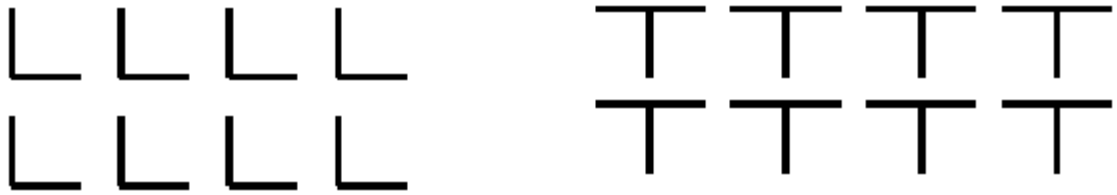
Same or different textures?



Adapted from Bill Freeman, MIT

Textons

Image textures generally consists of organized patterns of regular sub-elements

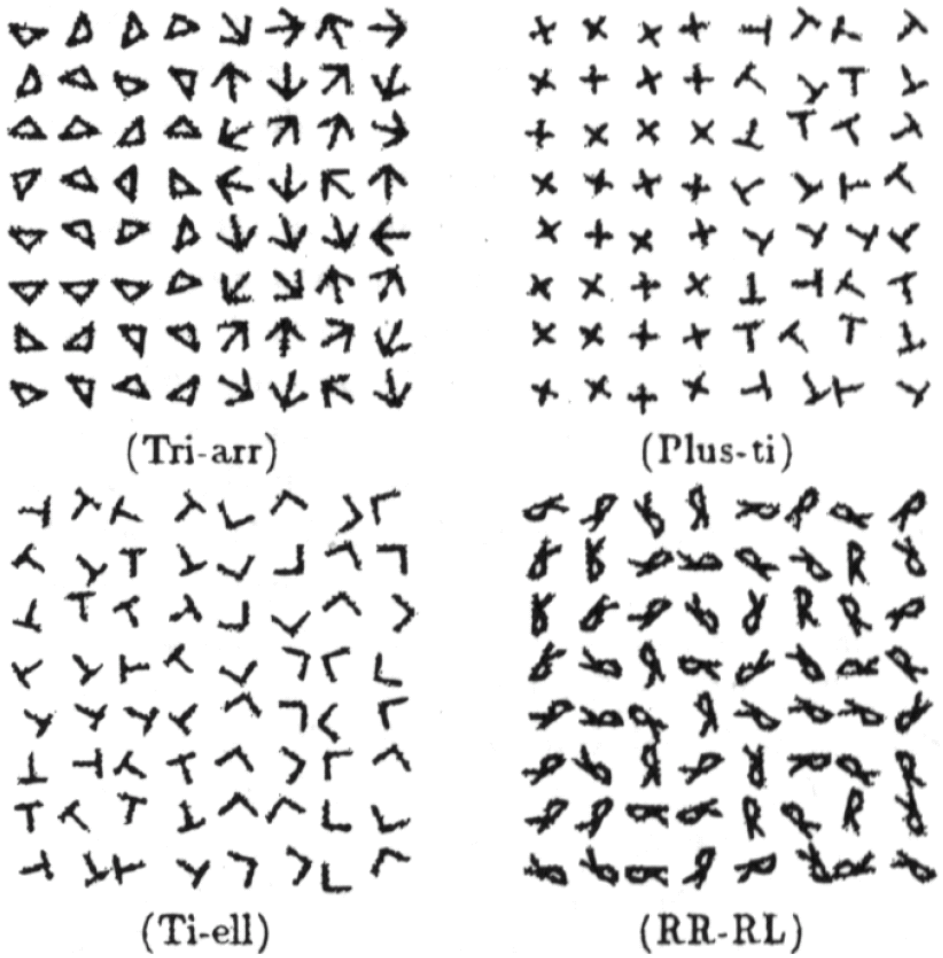


One natural way to represent texture is to find the textons and then describe the way in which they are laid out

It generally required a human to look at the texture in order to decide what those fundamental units were...

Adapted from Bill Freeman, MIT

Malik & Perona, CVPR 1989



A set of texture examples used in experiments with human subjects to tell how easily various types of textures can be discriminated

Bergen and Adelson, Nature 1988

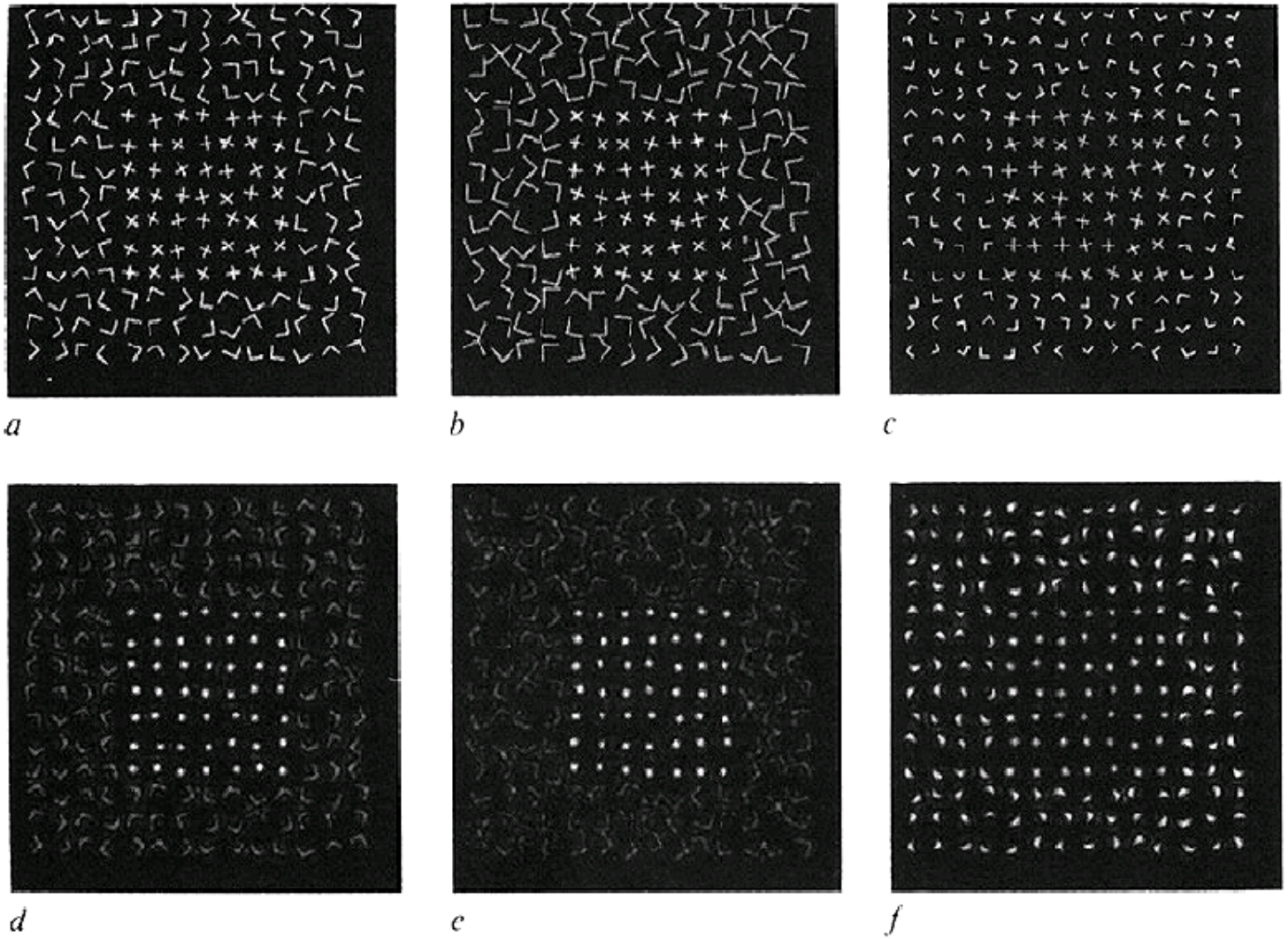


Fig. 1 *Top row*, Textures consisting of Xs within a texture composed of Ls. The micropatterns are placed at random orientations on a randomly perturbed lattice. *a*, The bars of the Xs have the same length as the bars of the Ls. *b*, The bars of the Ls have been lengthened by 25%, and the intensity adjusted for the same mean luminance. Discriminability is enhanced. *c*, The bars of the Ls have been shortened by 25%, and the intensity adjusted for the same mean luminance. Discriminability is impaired. *Bottom row*: the responses of a size-tuned mechanism *d*, response to image *a*; *e*, response to image *b*; *f*, response to image *c*.

Adapted from Bill Freeman, MIT

Representing Texture

- Instead of looking for patterns at the level of arrow-heads or triangles, look for simpler pattern elements (e.g. dots and bars)

- Easy : Find patterns by filtering the image

- Remember : There is a strong response to the filter when the image pattern in a neighborhood looks similar to the filter kernel

- Represent a texture in terms of the response of a collection of filters in different scales and orientations

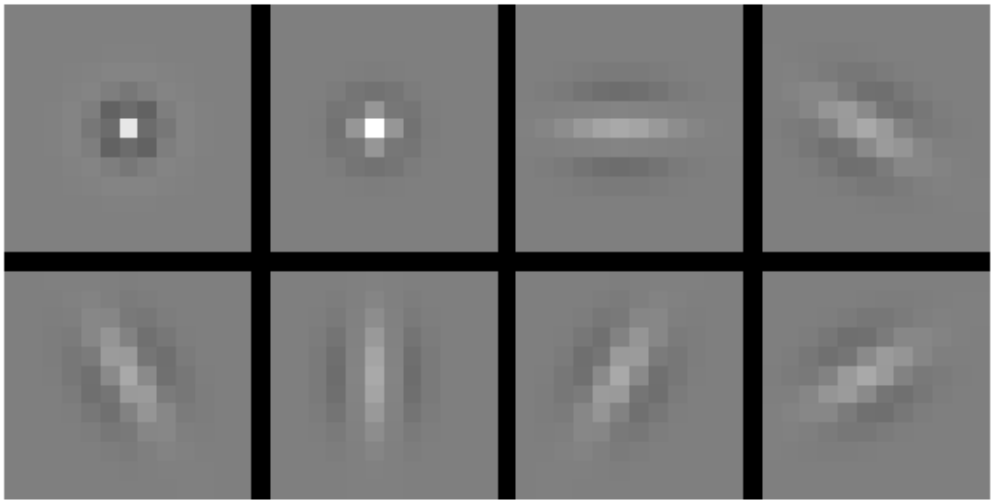
Representing Texture

- Spot filters – they respond strongly to small regions that differ from their neighbors

- Bar filters – respond to oriented structure

- To Obtain these filters -- form a weighted difference of Gaussian filters at different scales

Spots and Bars by weighted sums of Gaussians



A set of eight filters used for expanding images into a series of responses (zero represented by a mid-grey level, lighter values being positive and darker values being negative)

Spot : weighted sum of three concentric, symmetric Gaussians

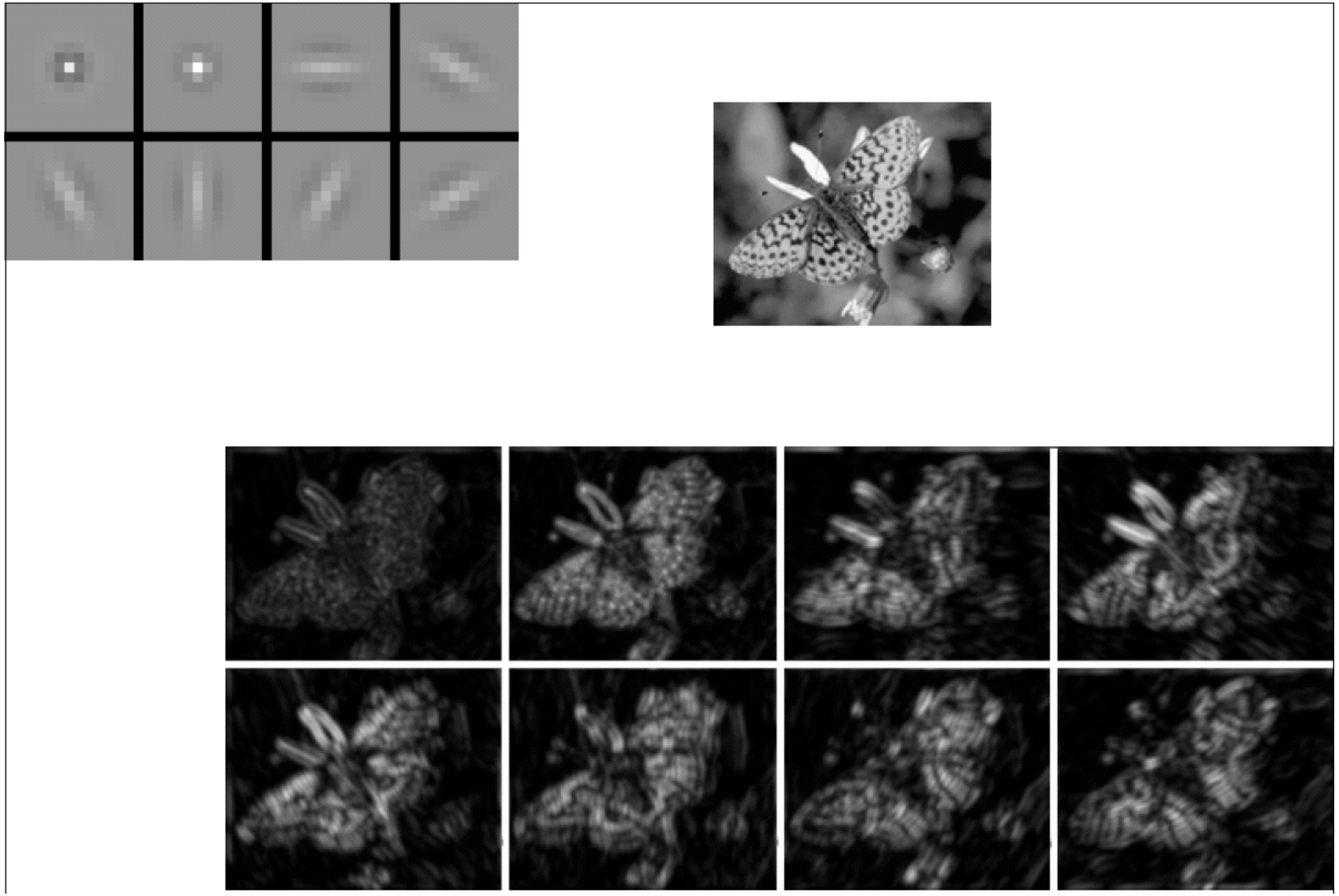
e.g : with weights 1, -2, and 1 and corresponding sigmas 0.62, 1 and 1.6

Oriented bars : weighted sum of three oriented Gaussians which are offset with respect to each other.

e.g. horizontal - with Gaussian weights –1, 2 and –1, where sigma in x direction is 2 and sigma in y direction is 1. The centers are offset along the y axis lying at (0,1) (0,0) and (0,-1)

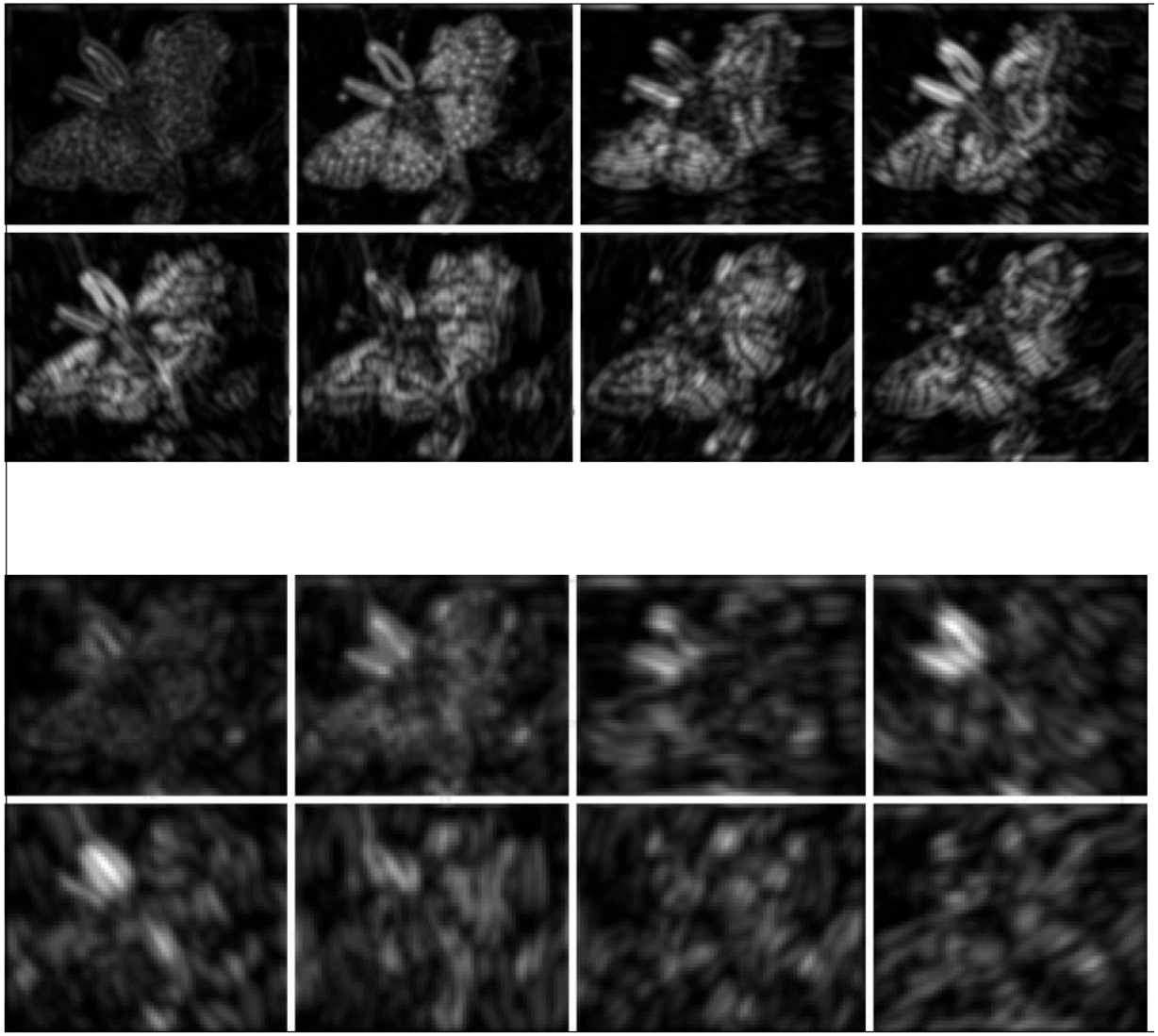
Adapted from David Forsyth, UC Berkeley

Spots and Bars by weighted sums of Gaussians



Adapted from David Forsyth, UC Berkeley

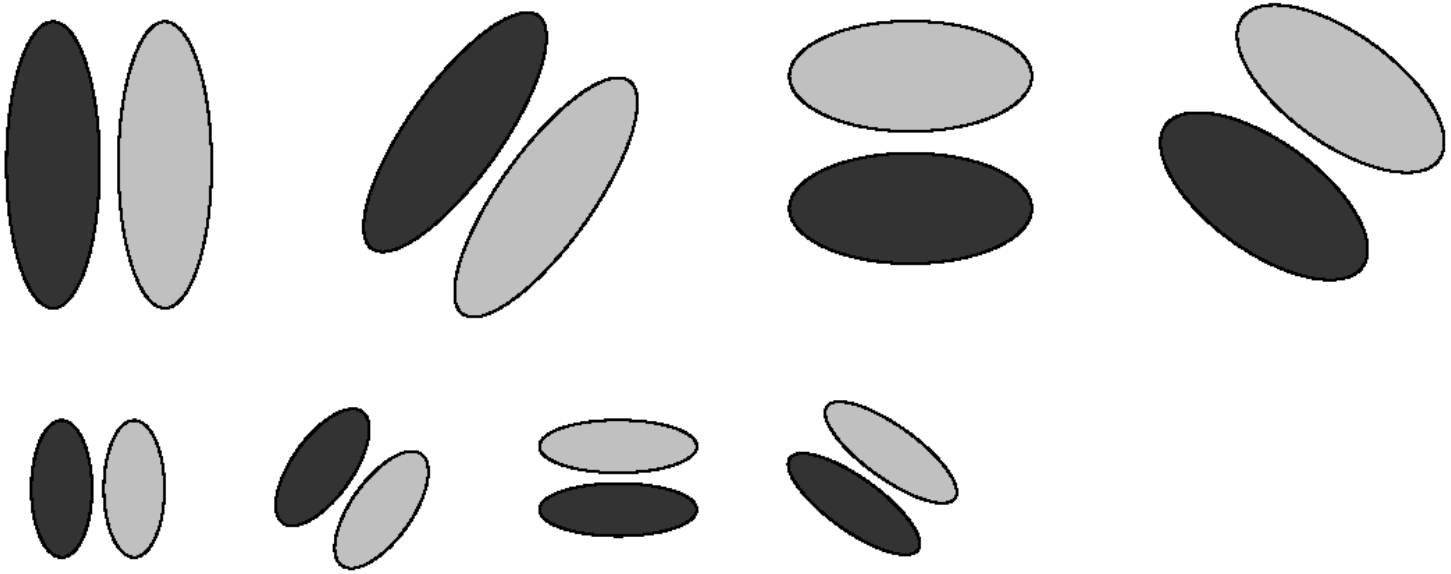
Spots and Bars by weighted sums of Gaussians



Coarser Scale

Adapted from David Forsyth, UC Berkeley

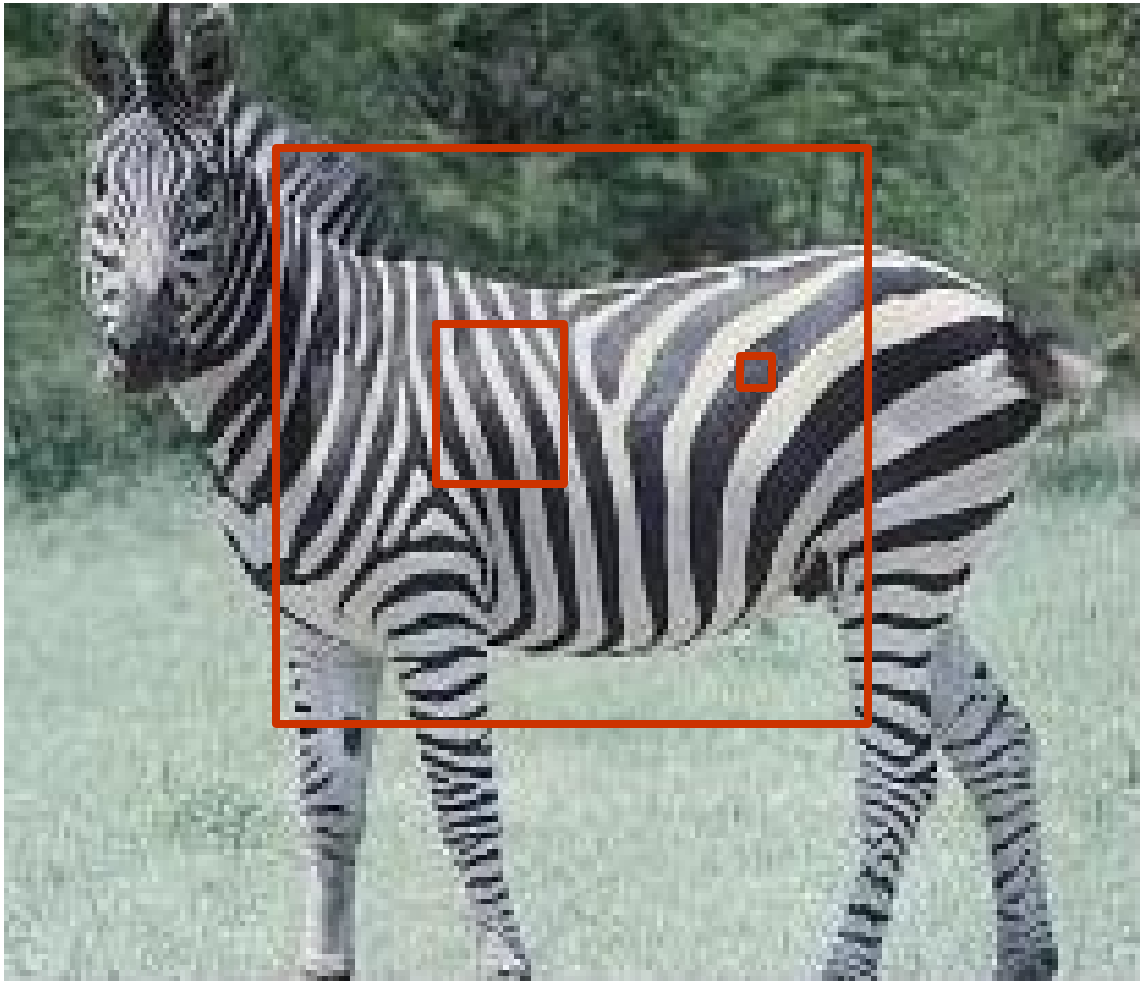
How many filters and what orientation



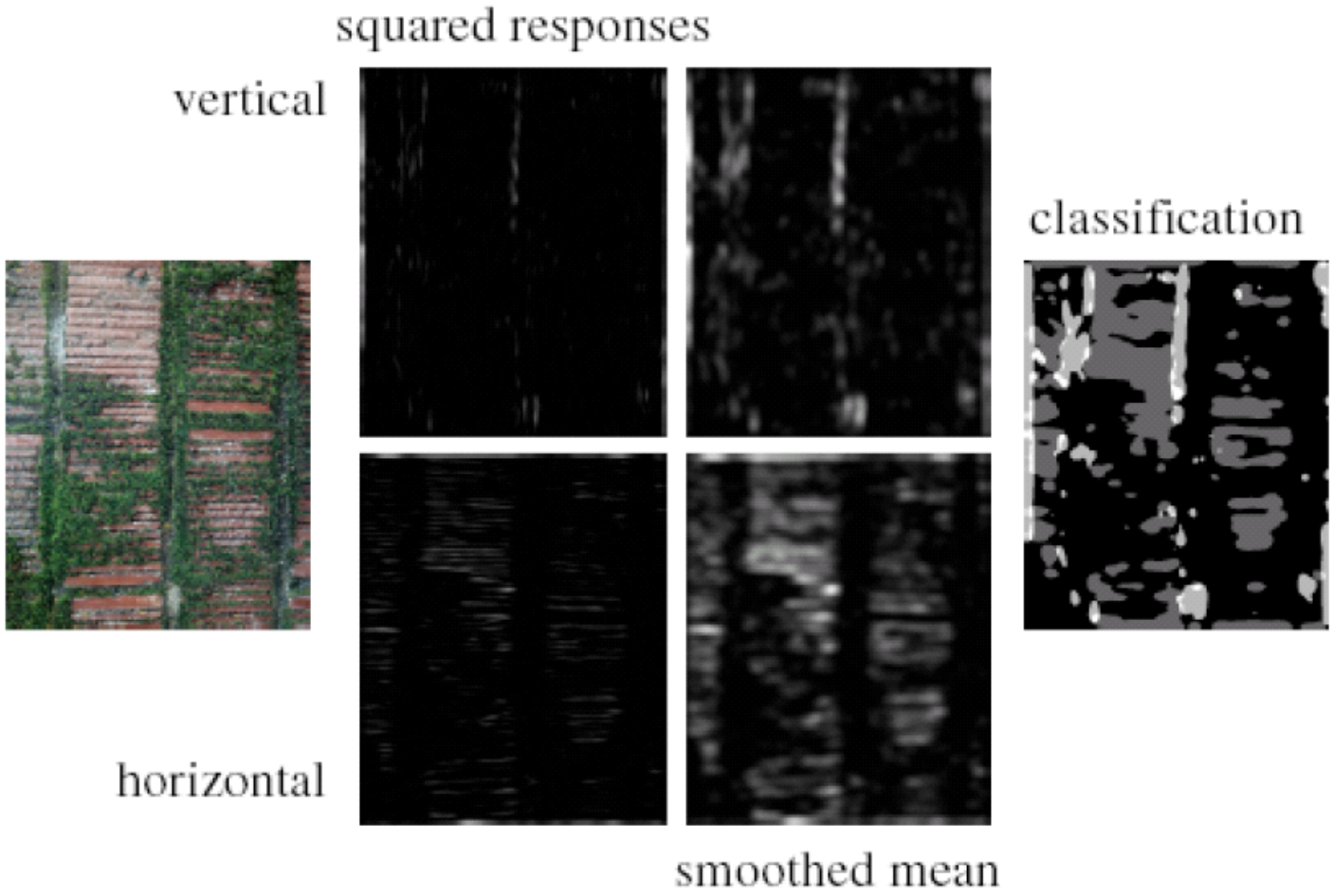
use lots of filters, multi-orientation & scale
6 orientation is a good number
Basic spot and bar combination is similar to complicated filters

Malik & Perona, 1995

Window size – Scale to describe the texture



Texture Segmentation



Squared – to count black-to-white pixels same as white-to-black pixels
Smooth – equivalent to estimating the mean of the squared filter outputs in some window
Pass to classifier – to describe the texture

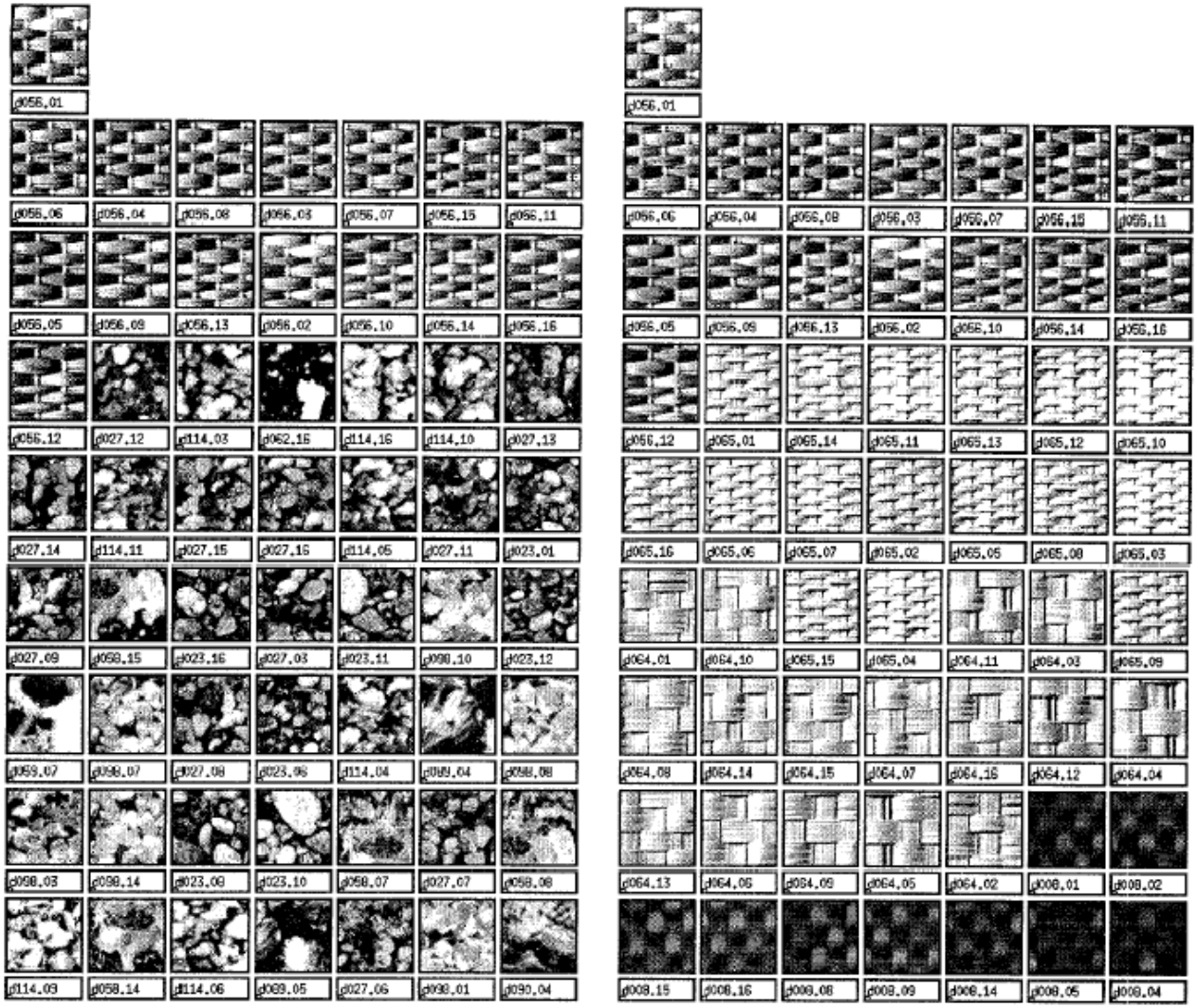
Choice of statistics

- What statistics should be collected

- mean of the squared filter output
  - Differentiates spoty windows from stripey windows
  - (Malik and Perona)
- Use mean and standard deviation
  - Texture matching
  - (Ma and Manjunath)

Adapted from David Forsyth, UC Berkeley

Texture Similarity

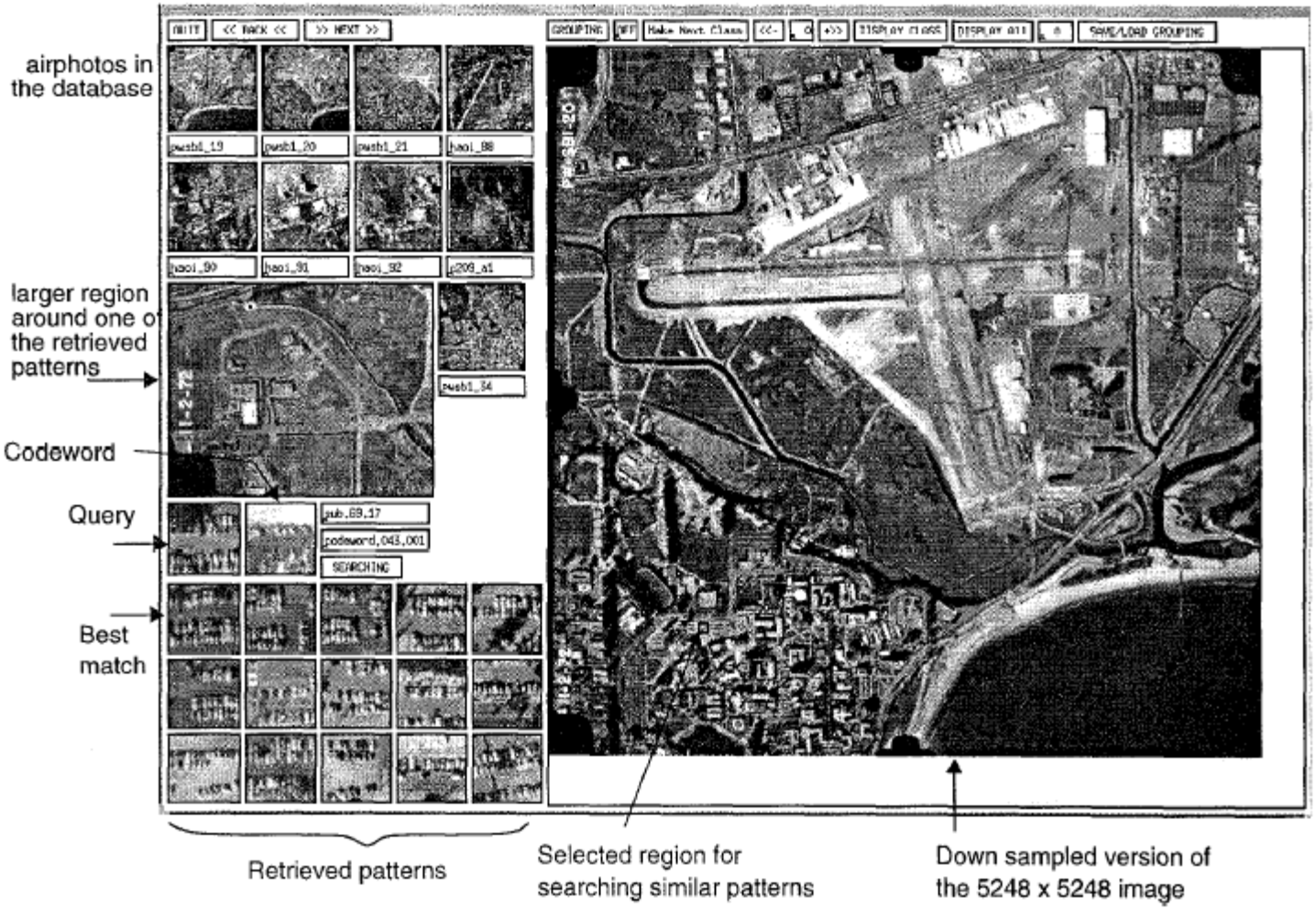


(a) before learning similarity measures

(b) after learning similarity measures

Ma & Manjunath, CVPR 1996

Application – Satellite Images



Ma & Manjunath, CVPR 1996

Representing textures - summary

Textures are made up of quite stylised subelements, repeated in meaningful ways

Representation:

- find the subelements, and represent their statistics

But what are the subelements, and how do we find them?

- recall normalized correlation
- find subelements by applying filters, looking at the magnitude of the response

What filters?

- experience suggests spots and oriented bars at a variety of different scales
- details probably don't matter

What statistics?

- within reason, the more the merrier.
- At least, mean and standard deviation
- better, various conditional histograms.

Adapted from Bill Freeman, MIT

Analysis

- Representing texture using the statistics of filter outputs requires convolving the image with many filters at many scales – can be done systematically
- Analysis : process of convolving an image with a range of filters
- Gaussian pyramid – an example of image analysis by a bank of filters (smoothing filters)
- Handles scale systematically by subsampling the image once it has been smoothed – generating next coarsest scale is easier, because we don't process all the information

The Gaussian Pyramid

Adapted from David Forsyth, UC Berkeley

Down-Sampling

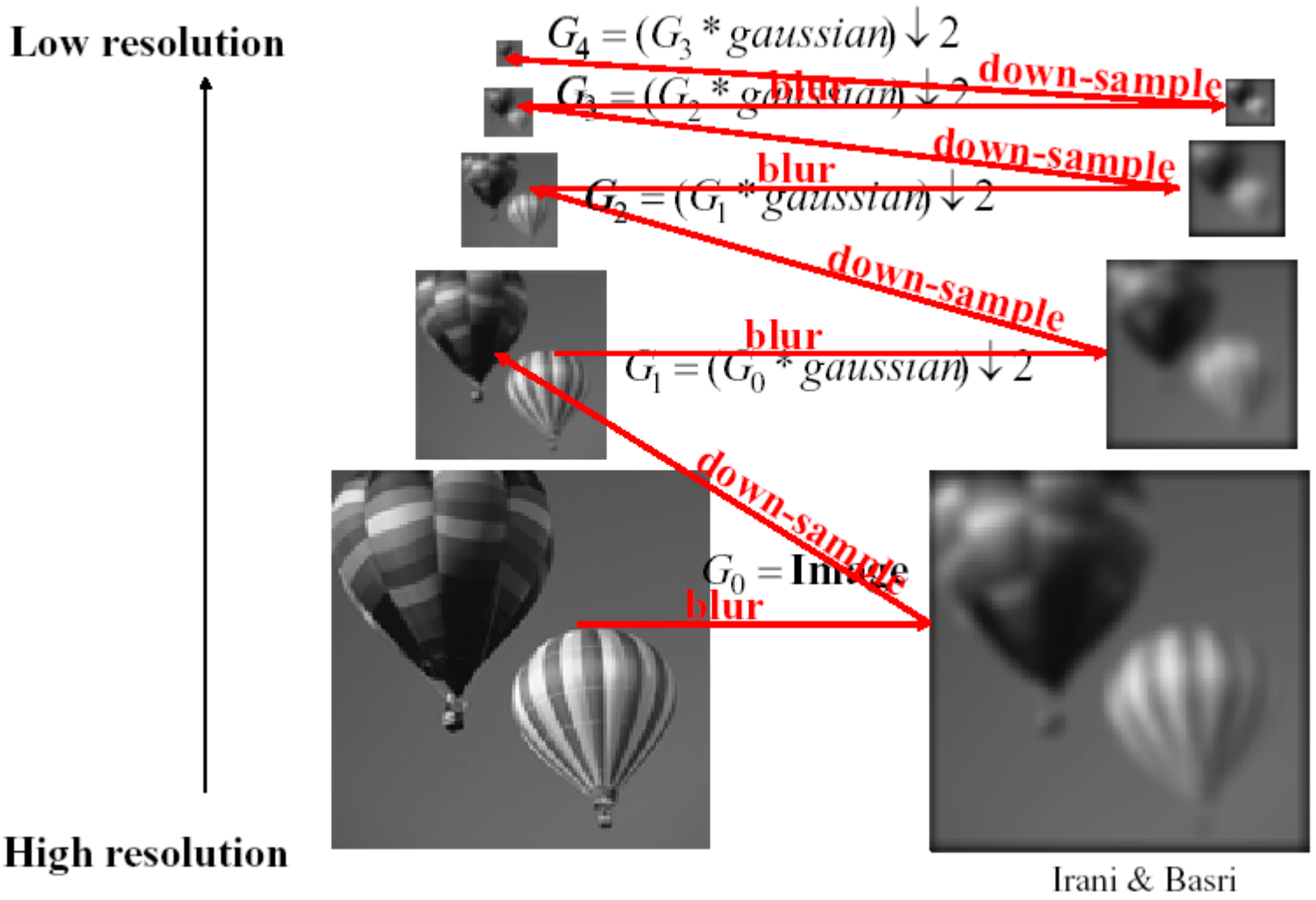
Sub-sampling an Image with a factor of two

Apply a low-pass filter to the original image
(a Gaussian with a σ of between one and two pixels is usually an acceptable choice)

Create a new image whose dimensions on edge are half those of the old image

Set the value of the i,j th pixel of the new image to the value of the 2i, 2j th pixel of the filtered image

Gaussian Pyramid



adapted from Michael Black, Brown University

Laplacian pyramid

- In fact – Gaussian is highly redundant
  - Because each layer is a low-pass filtered version of the previous layer – we are representing the lowest spatial frequencies many times
- Laplacian pyramid – low redundancy
  - Makes use of the fact that a coarse layer of the Gaussian pyramid predicts the appearance of the next finer layer.
  - With an upsampling operator we can produce a version of coarser layer of the same size as the next finer layer
  - We only need to store the difference between this prediction and the next layer itself.
- However it does not deal with orientation

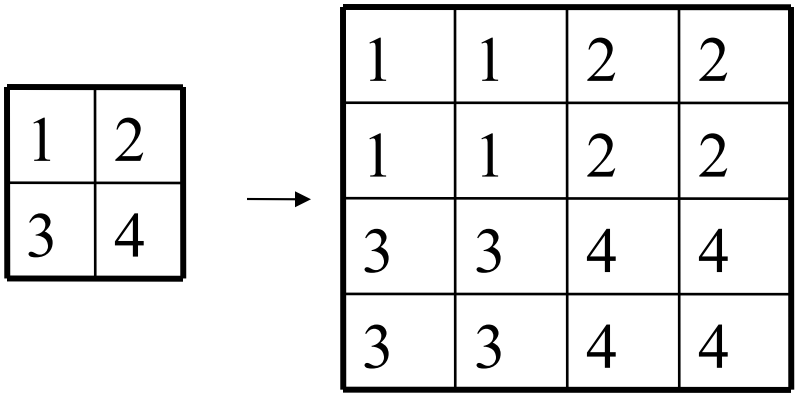
Upsampling

$S^{\uparrow}(I)$

upsampling operator

It takes an image at level n+1 to an image at level n.

In particular, it takes an image, and produces an image twice the size in each dimension. The four elements of the output image at (2j - 1, 2k - 1); (2j, 2k -1); (2j - 1, 2k); and (2j, 2k) all have the same value as the j, k'th element of I.



Analysis - Building a Laplacian Pyramid from an image

coarsest scale of the Laplacian pyramid is same as the coarsest scale of the Gaussian Pyramid.

Each of the finer layers are the difference between a layer of Gaussian pyramid and a prediction obtained by upsampling the next coarsest layer of the Gaussian pyramid

$$P_{\text{Laplacian}}(\mathcal{I})_m = P_{\text{Gaussian}}(\mathcal{I})_m$$

(where m is the coarsest level) and

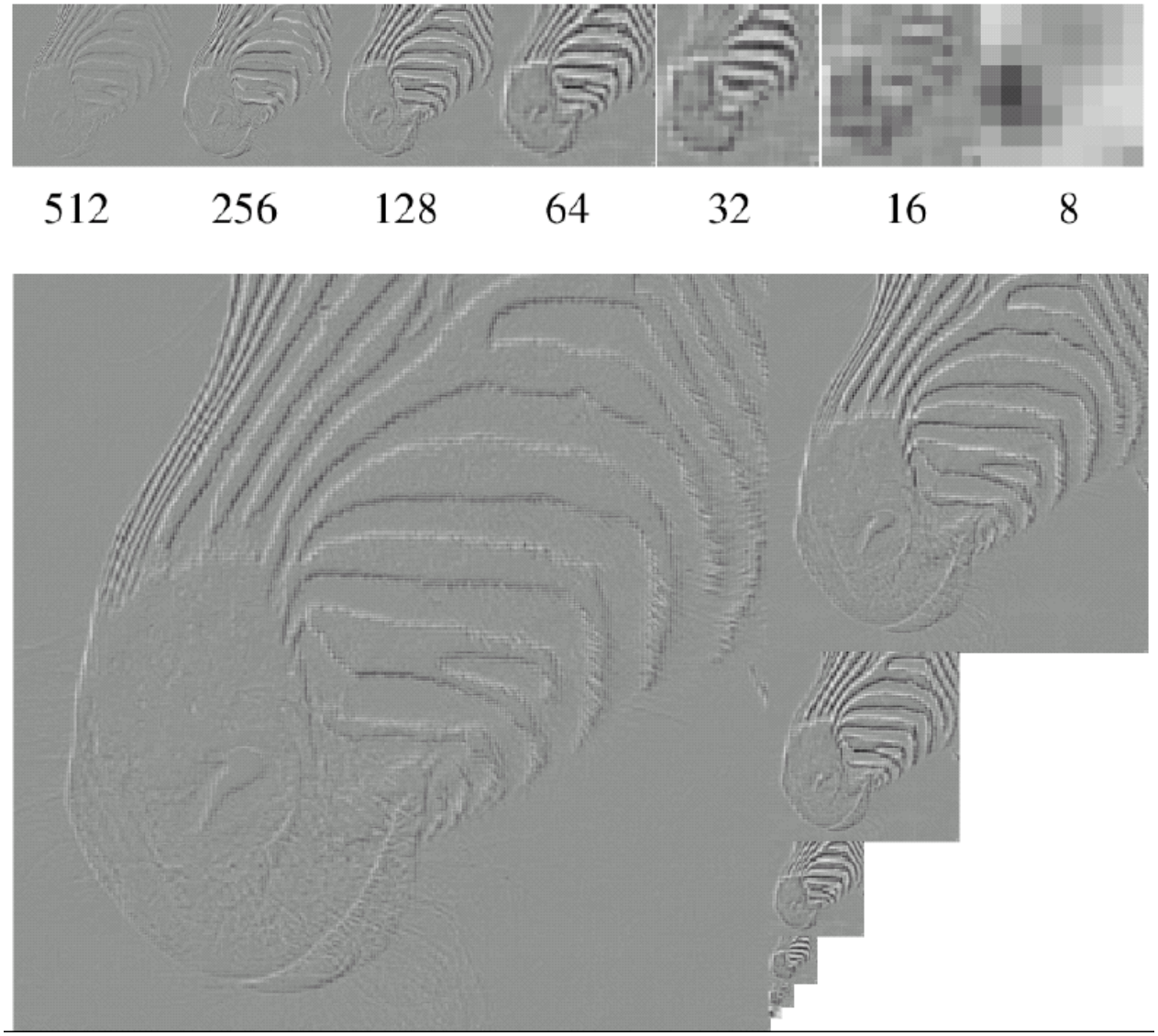
$$\begin{aligned} P_{\text{Laplacian}}(\mathcal{I})_k &= P_{\text{Gaussian}}(\mathcal{I})_k - S^{\uparrow}(P_{\text{Gaussian}}(\mathcal{I})_{k+1}) \\ &= (Id - S^{\uparrow}S^{\downarrow}G_{\sigma})P_{\text{Gaussian}}(\mathcal{I})_k \end{aligned}$$

Building a Laplacian Pyramid from an image

- Algorithm

```
Form a Gaussian pyramid

Set the coarsest layer of the Laplacian pyramid to be
the coarsest layer of the Gaussian pyramid

For each layer, going from next to coarsest to finest

  Obtain this layer of the Laplacian pyramid by
  upsampling the next coarser layer, and subtracting
  it from this layer of the Gaussian pyramid

end
```

The Laplacian Pyramid



Adapted from David Forsyth, UC Berkeley

Synthesis

- Recover an image from from its Laplacian Pyramid
  - Recover the Gaussian pyramid from the Laplacian pyramid
  - Take the finest scale of the Gaussian pyramid
  - Coarsest scale of Gaussian = Coarsest scale of Laplacian
  - Upsample it
  - Add the next coarsest scale of Laplacian pyramid

Synthesis

- Algorithm

```
Set the working image to be the coarsest layer

For each layer, going from next to coarsest to finest

  upsample the working image and add the current layer
  to the result

  set the working image to be the result of this operation

end
The working image now contains the original image
```

Smoothing and Low-pass filters

- Convolving an image with a Gaussian with standard deviation σ is same as multiplying the Fourier transform of the image with a Gaussian of standard deviation of 1/σ

- If we smooth with a very small standard deviation, all but the highest spatial frequencies will be preserved

- If we smooth with a very large sigma then the result will be pretty much the average value of the image

- Gaussian pyramid – a set of low-pass filtered versions of the image

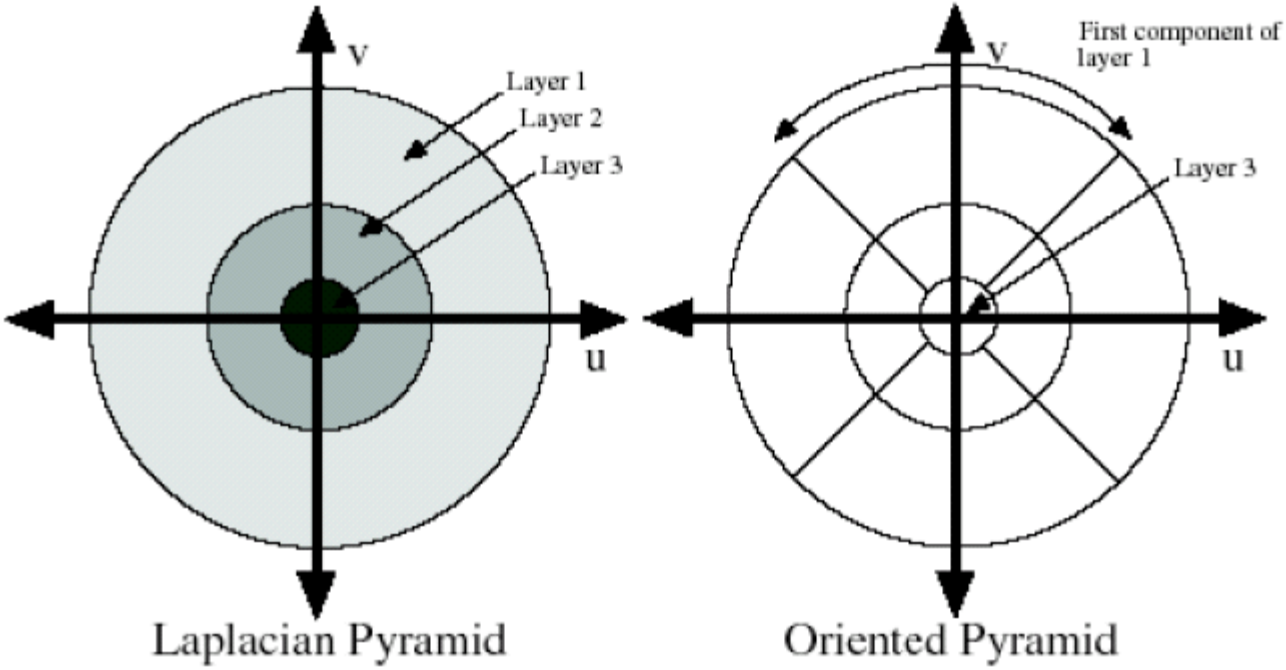
Adapted from David Forsyth, UC Berkeley

Band-pass filters and Orientation Selective Operators

- Band-pass filter :
  - a filter which has high gain for some range of spatial frequencies and low gain for higher and for lower spatial frequencies
    - An example :Smooth an image with the difference of two Gaussians, one with small sigma and one with large sigma
      - Insensitive to orientation
- Alternative :
  - orientation selective
  - Responds most strongly to signals that have a particular range of spatial frequencies and orientations

Adapted from David Forsyth, UC Berkeley

Laplacian pyramid vs. Oriented pyramids



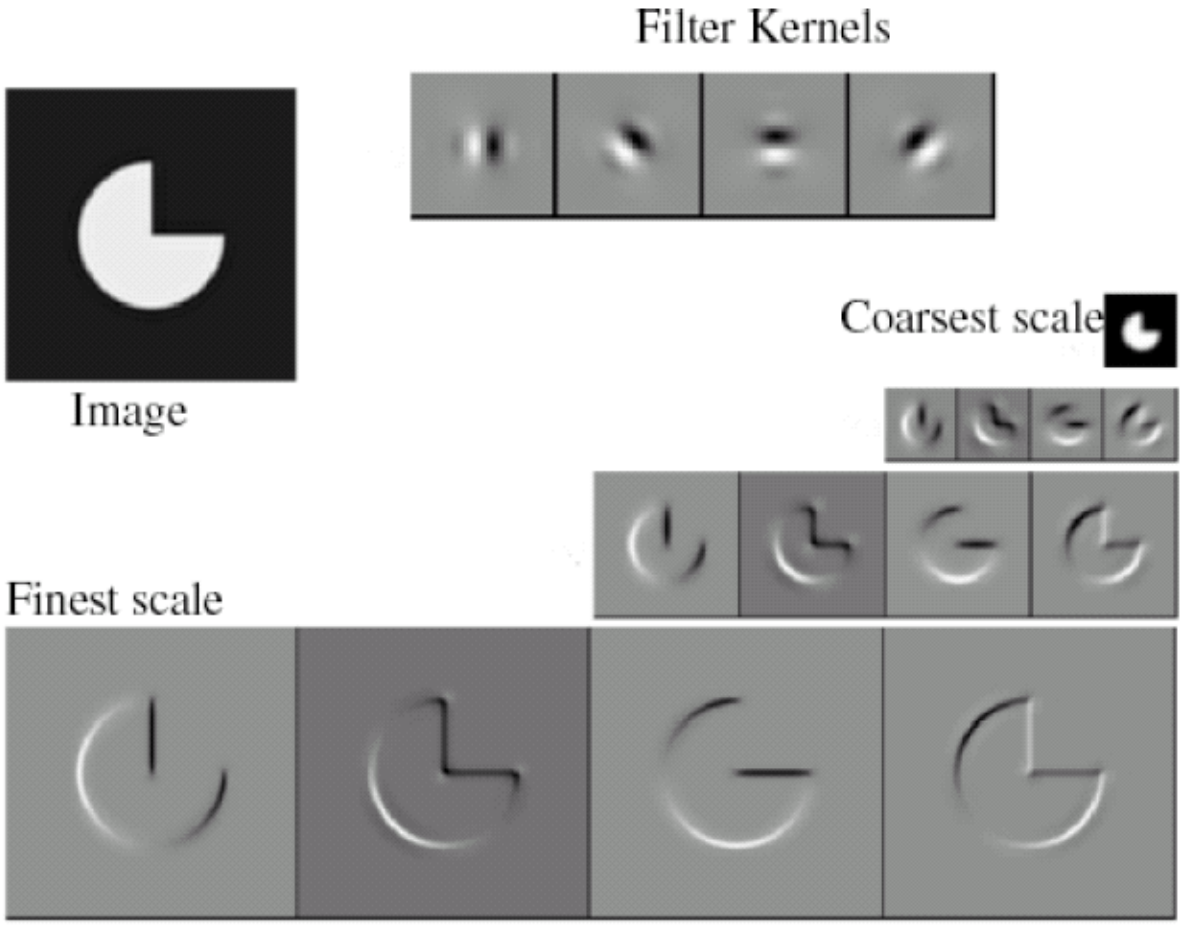
Adapted from David Forsyth, UC Berkeley

Oriented Pyramids

- Laplacian pyramid is orientation independent
- Apply an oriented filter to determine orientations at each layer
  - by clever filter design, we can simplify synthesis
  - this represents image information at a particular scale and orientation

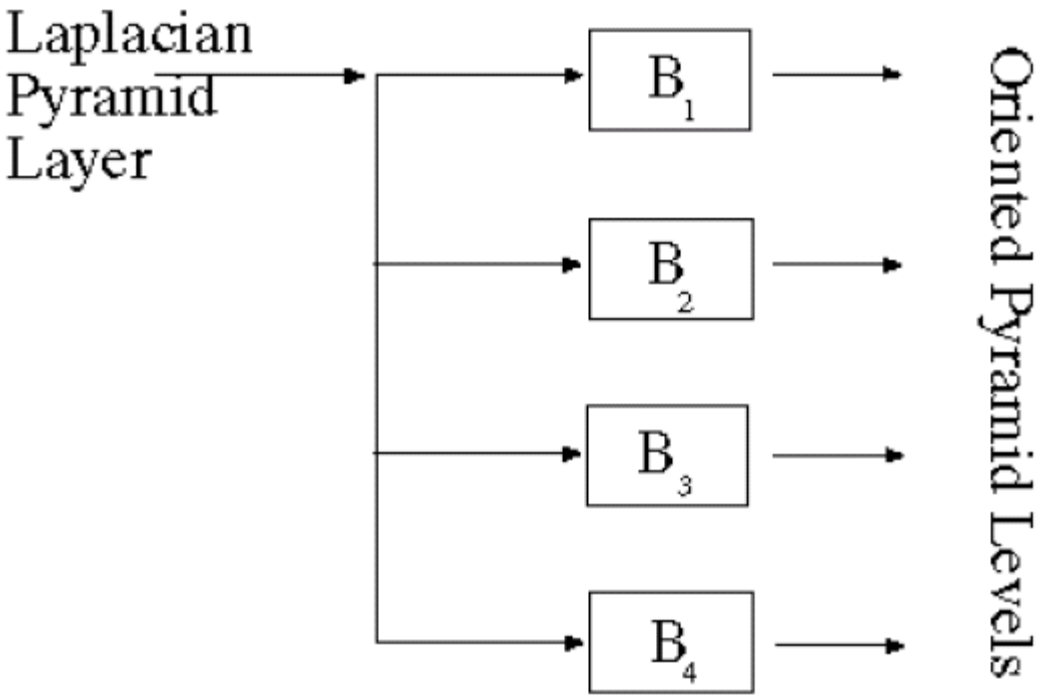
Adapted from David Forsyth, UC Berkeley

Oriented Pyramids



Adapted from David Forsyth, UC Berkeley

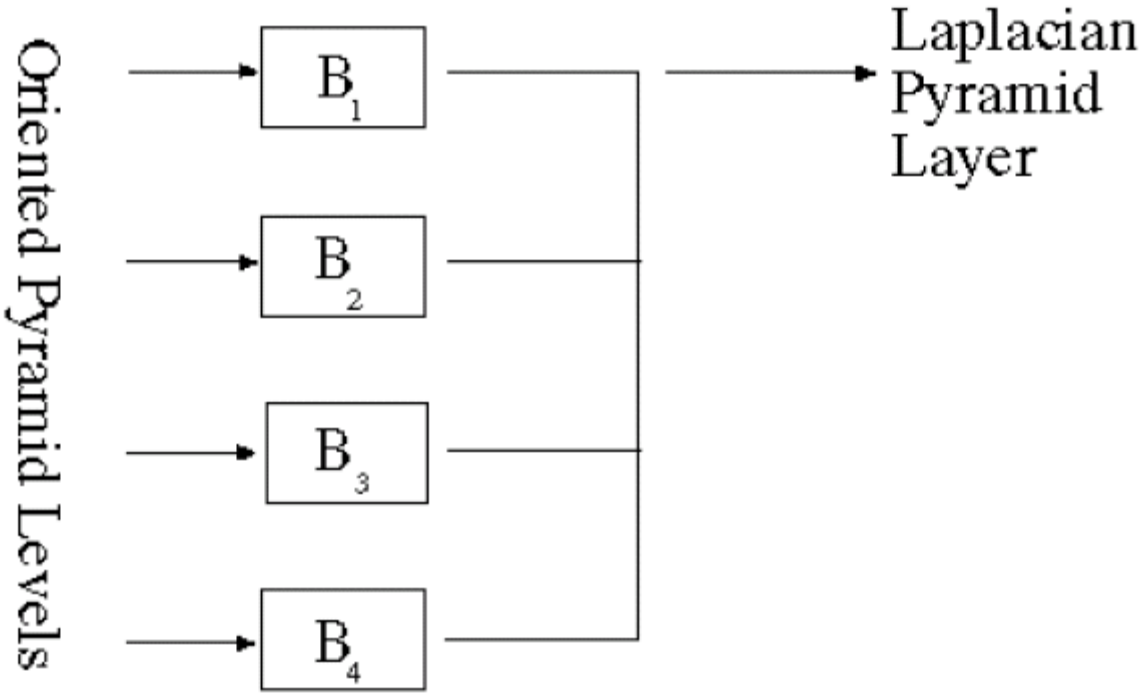
Analysis



Take the layers of Laplacian Pyramid and apply oriented filters

Adapted from David Forsyth, UC Berkeley

Synthesis



Re-filter the layers and add them

Adapted from David Forsyth, UC Berkeley

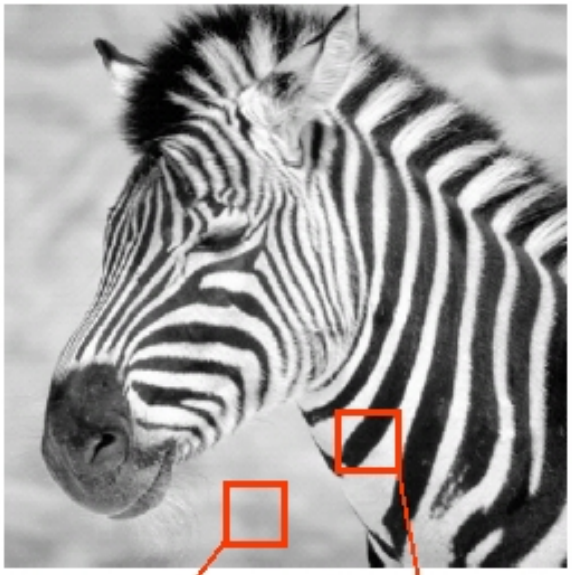
Texture

Local image characteristics based on:

- Scale
- Frequency
- Direction

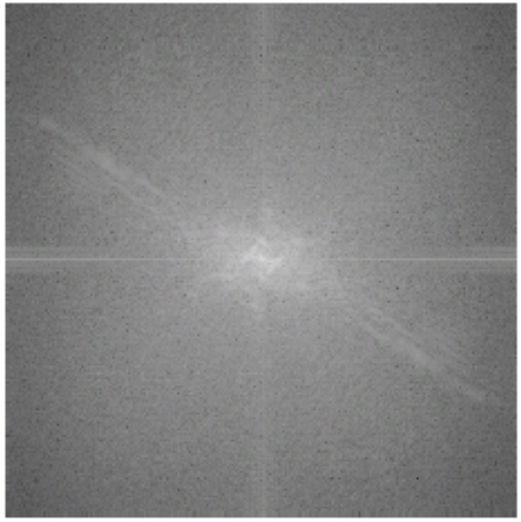
Adapted from Martial Hebert, CMU

Fourier Transform



Low frequency content
in all directions

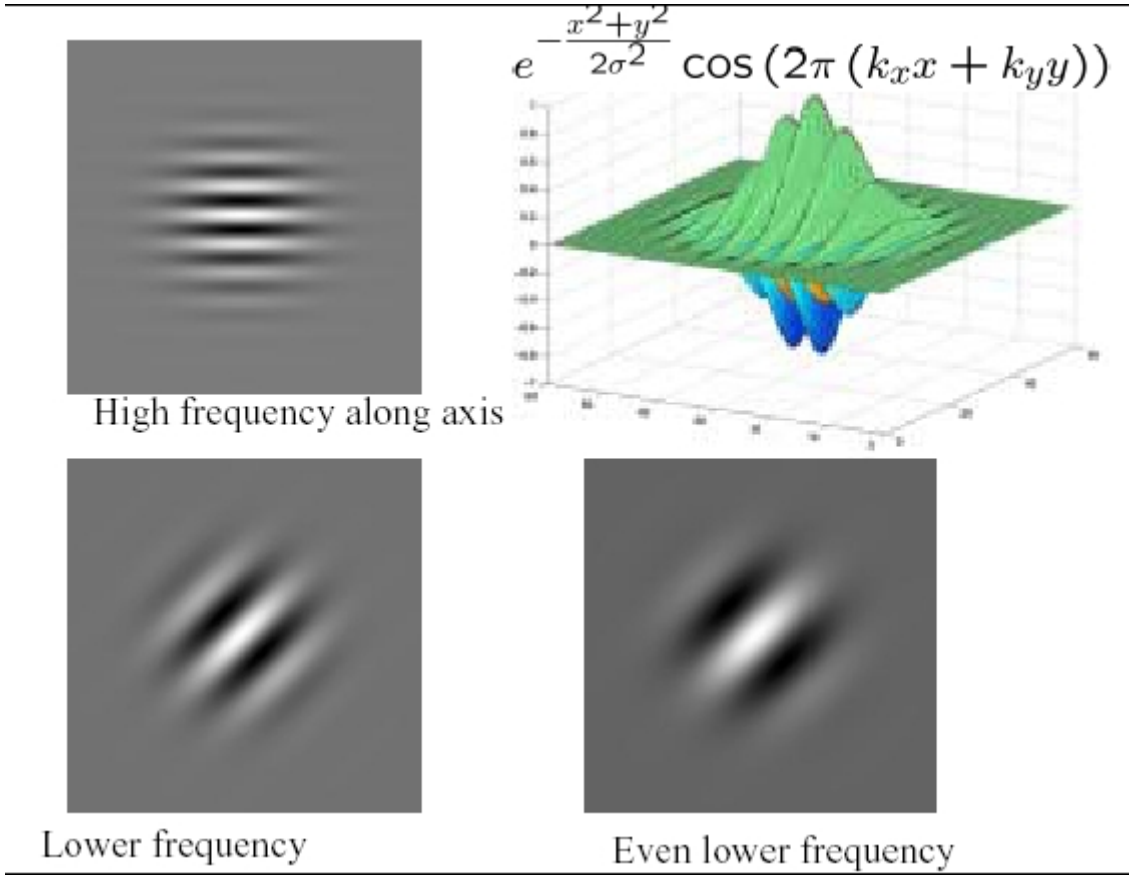
High frequency at
45° orientation



Fourier Transform = content
of entire image at all frequencies
and orientations

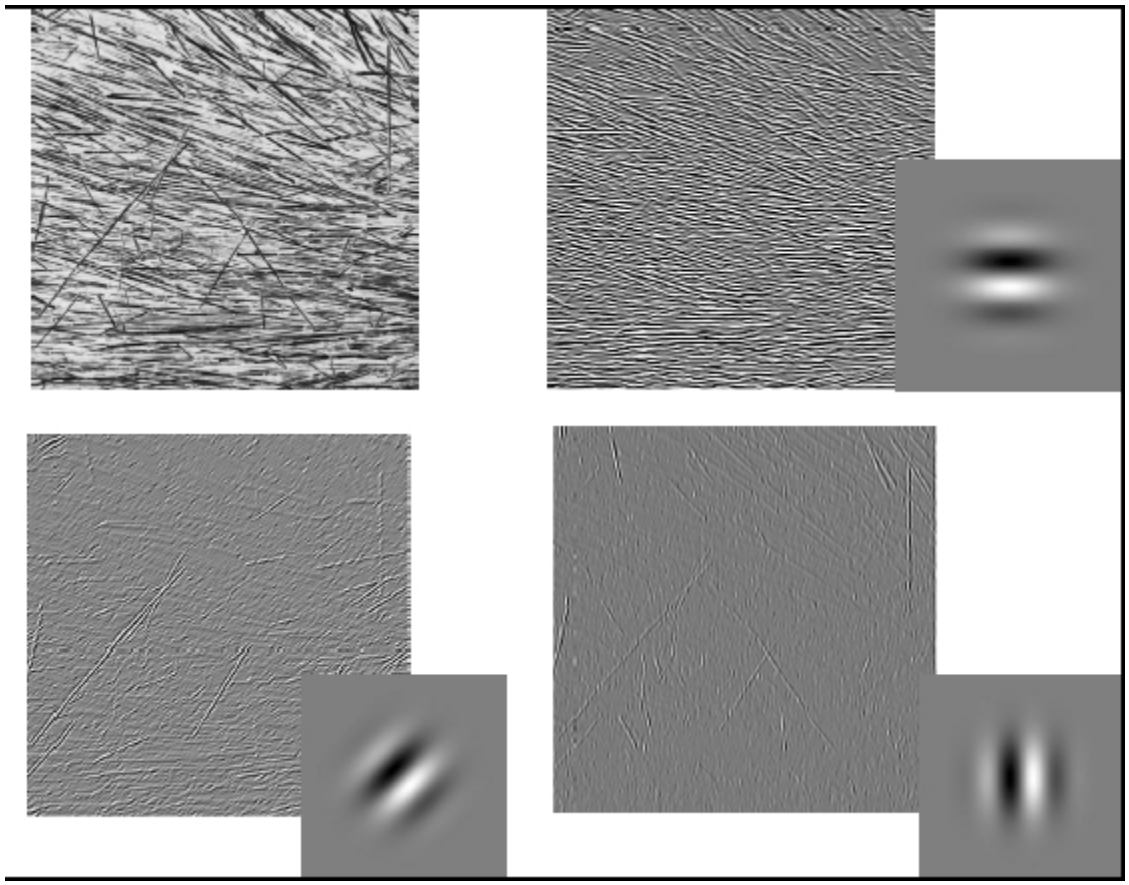
Adapted from Martial Hebert, CMU

Gabor Filter



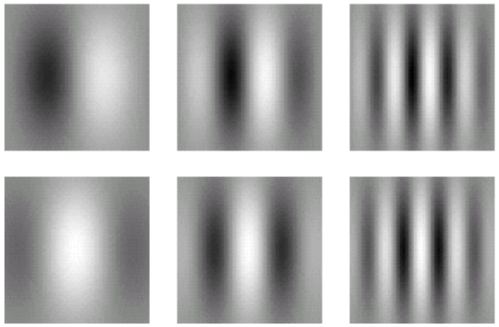
Adapted from Martial Hebert, CMU

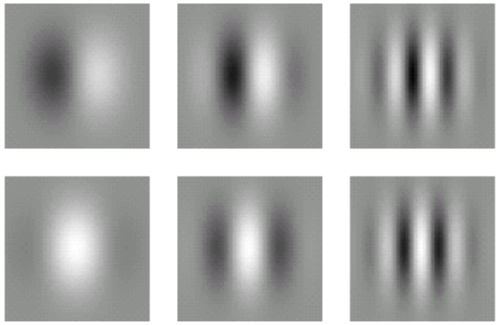
Gabor Filter



Adapted from Martial Hebert, CMU

Gabor filters





Gabor filters at different scales and spatial frequencies

top row shows anti-symmetric (or odd) filters, bottom row the symmetric (or even) filters.

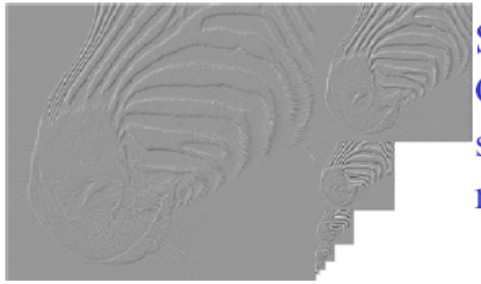
Adapted from David Forsyth, UC Berkeley

Image Pyramids

- Gaussian

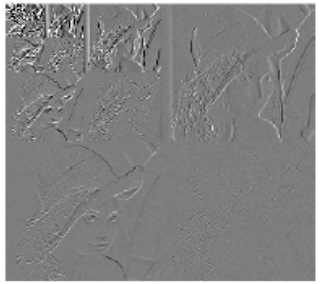
Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- Laplacian



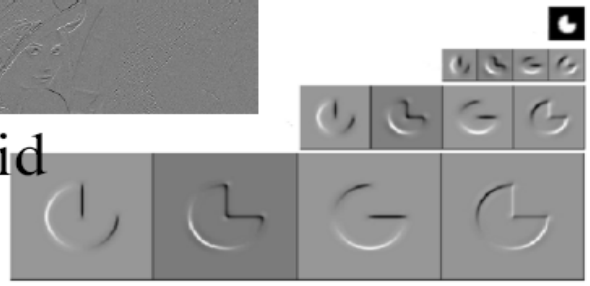
Shows the information added in Gaussian pyramid at each spatial scale. Useful for noise reduction & coding.

- Wavelet/QMF



Bandpassed representation, complete, but with aliasing and some non-oriented subbands.

- Steerable pyramid

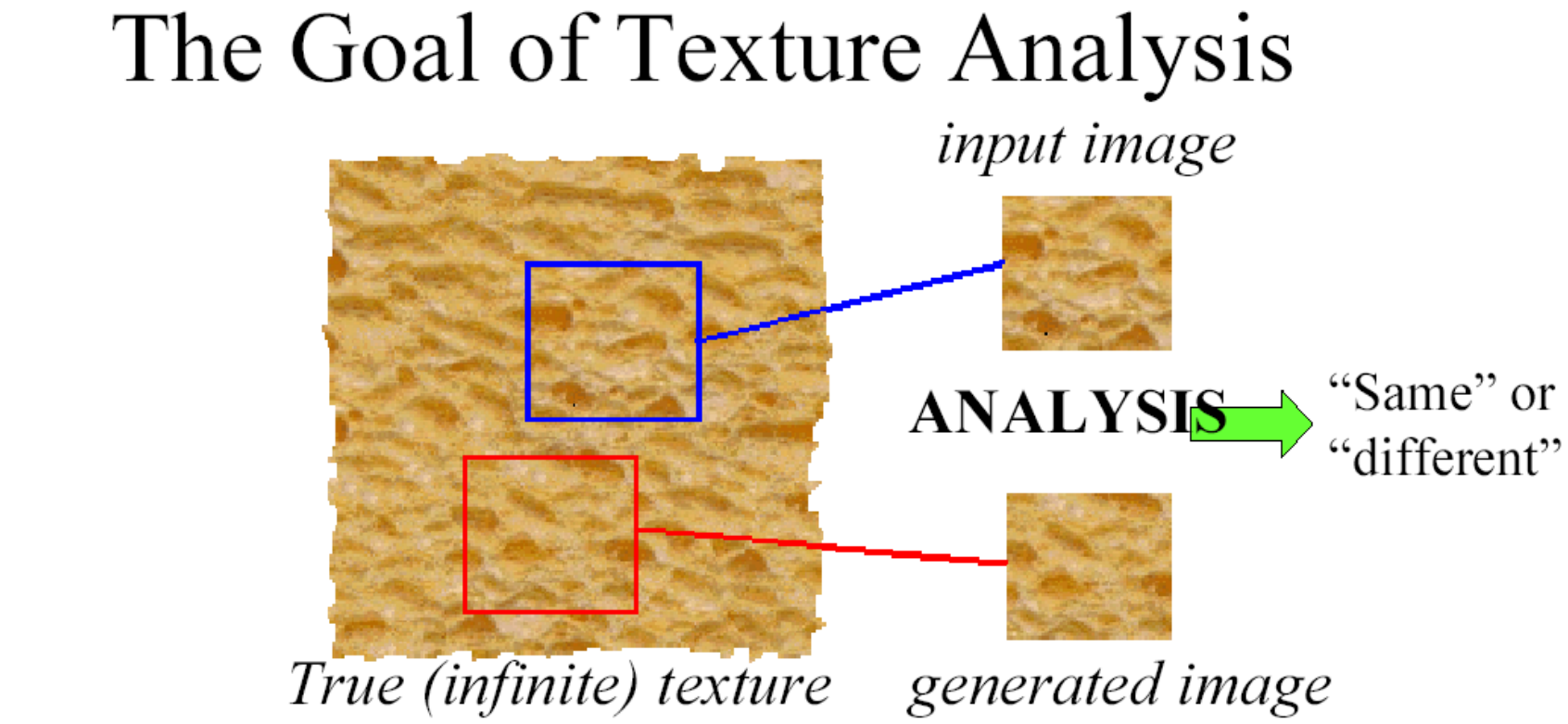


Shows components at each scale and orientation separately. Non-aliased subbands. Good for texture and feature analysis.

6

Adapted from Bill Freeman, MIT

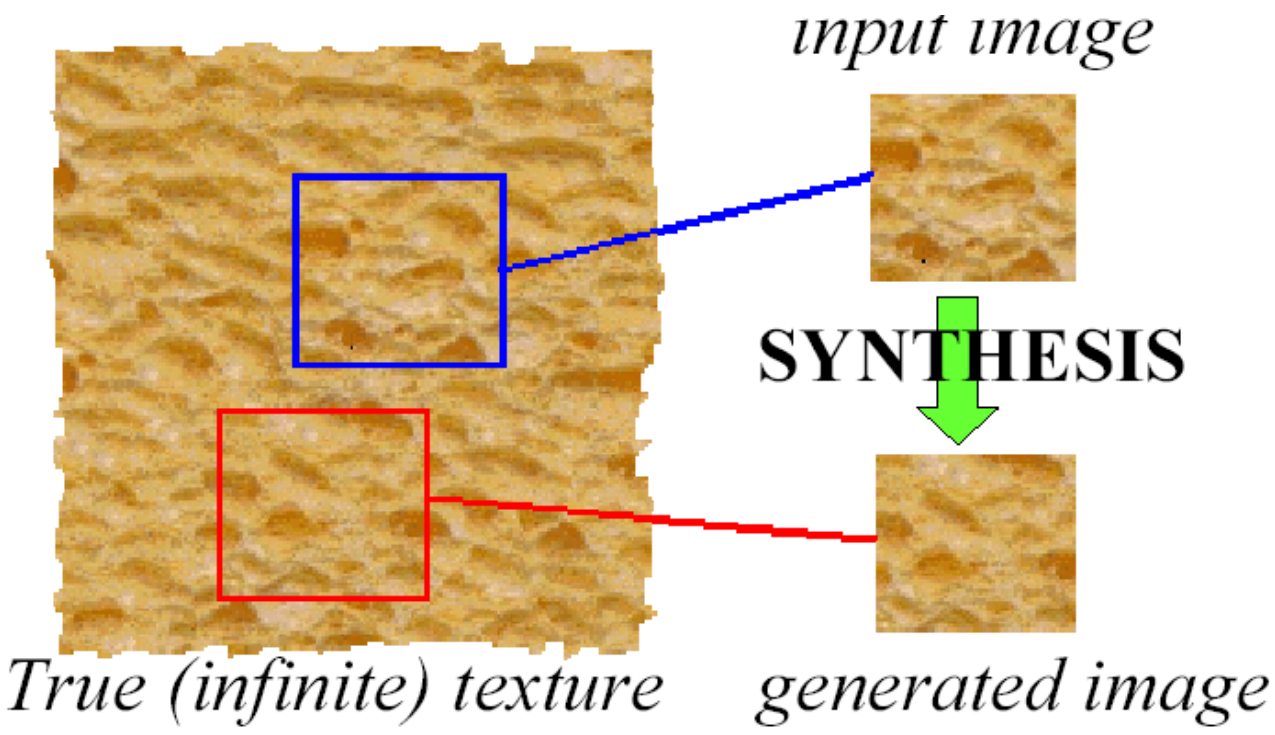
The Goal of Texture Analysis



Compare textures and decide if they're made of the same "stuff".

Adapted from Bill Freeman, MIT

The Goal of Texture Synthesis



Given a finite sample of some texture, the goal is to synthesize other samples from that same texture

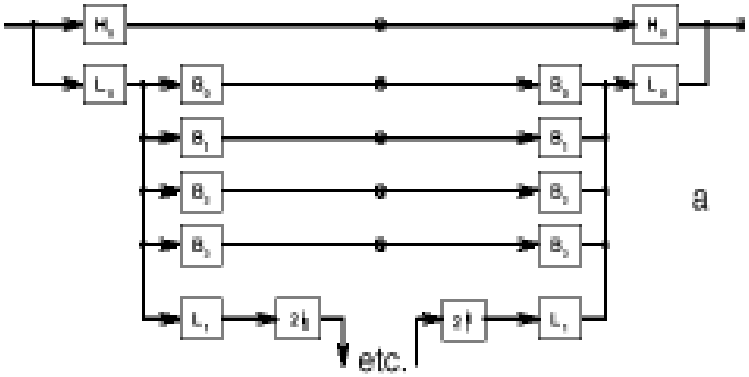
- The sample needs to be "large enough"

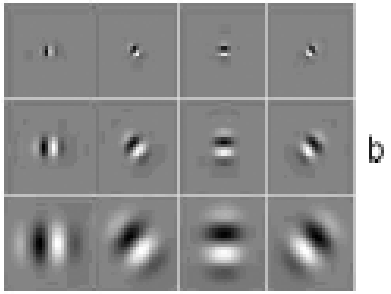
Adapted from Bill Freeman, MIT

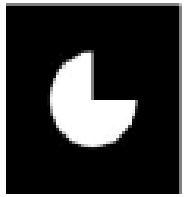
Texture Synthesis

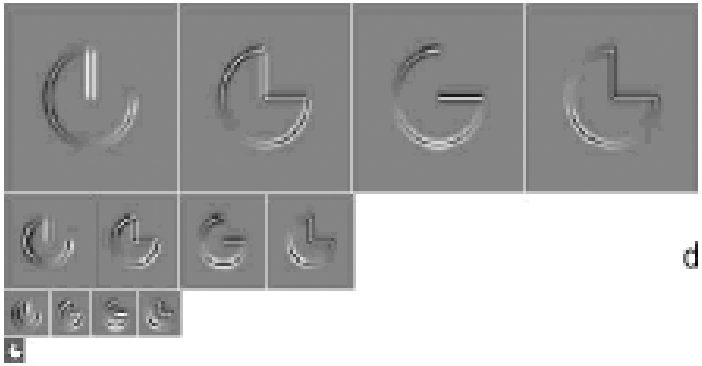
- Rendering is more realistic if the objects are textured
- Texture mapping for large objects requires a substantial amount of texture
- Tiling texture images will work poorly- the borders
- Strategy: to think of a texture as a sample from some probability distribution and then to try and obtain other samples from that same distribution.

Texture Synthesis – I Heeger and Bergen









If two homogenous texture samples are drawn from the same probability model, then (if the samples are big enough) histograms of the outputs of various applied to the samples will be the same.

take a noise image and adjust it until the histogram of responses of various filters on that noise image looks like the histogram of responses of these filters on the texture sample.

Texture Synthesis – I

```
make a working image from noise
match the working image histogram to the example image histogram
make a pyramid pe from the example image

until convergence
  Make a pyramid pw from the working image
  For each layer in the two pyramids
    match the histogram of pw's layer to that of pe's layer
  end
  synthesize the working image from the pyramid pw
end
```

Adapted from David Forsyth, UC Berkeley

Texture Synthesis – I

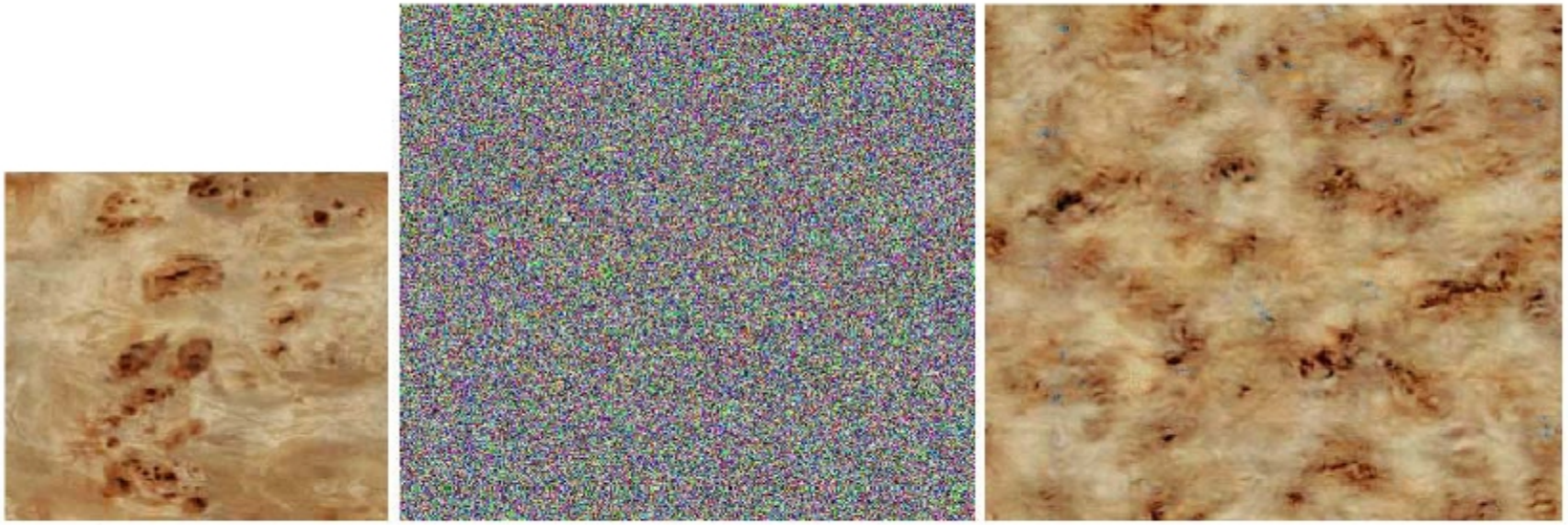


Figure 2: (Left) Input digitized sample texture: burled mappa wood. (Middle) Input noise. (Right) Output synthetic texture that matches the appearance of the digitized sample. Note that the synthesized texture is larger than the digitized sample; our approach allows generation of as much texture as desired. In addition, the synthetic textures tile seamlessly.

yields quite good results on a substantial variety of textures,

fail when there are conditional relations in the texture that are important for most natural textures, the histogram of filter responses at different scales and orientations is not independent.

Texture Synthesis – I

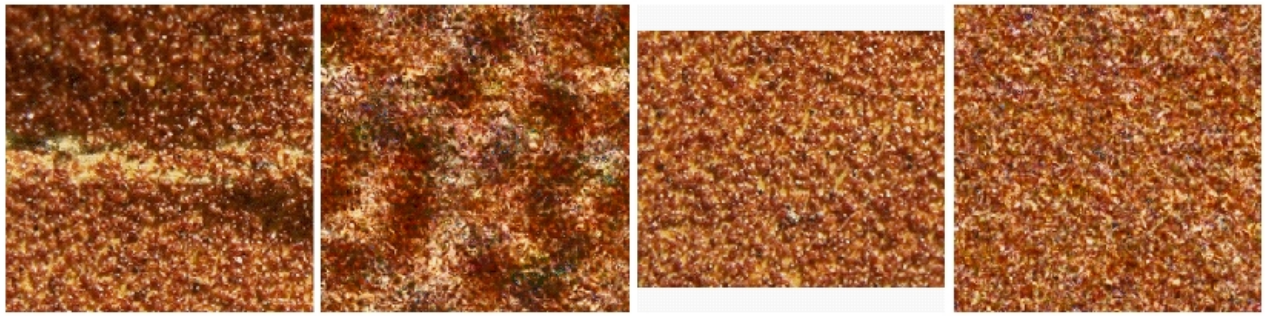
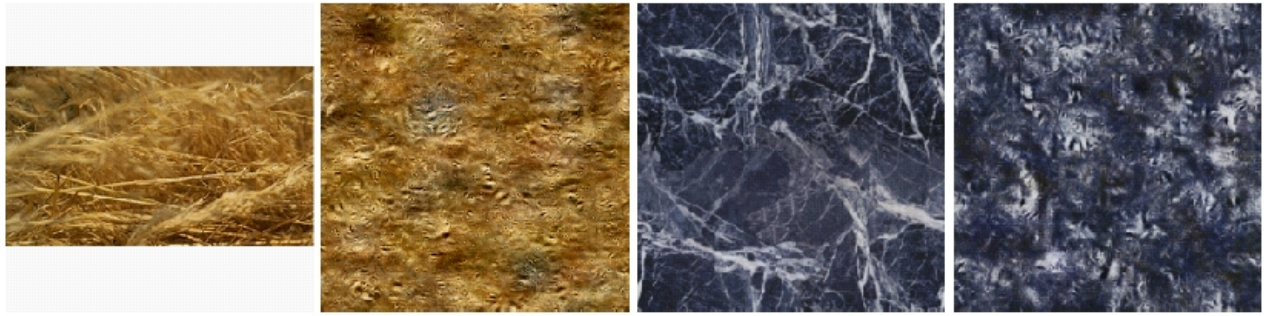


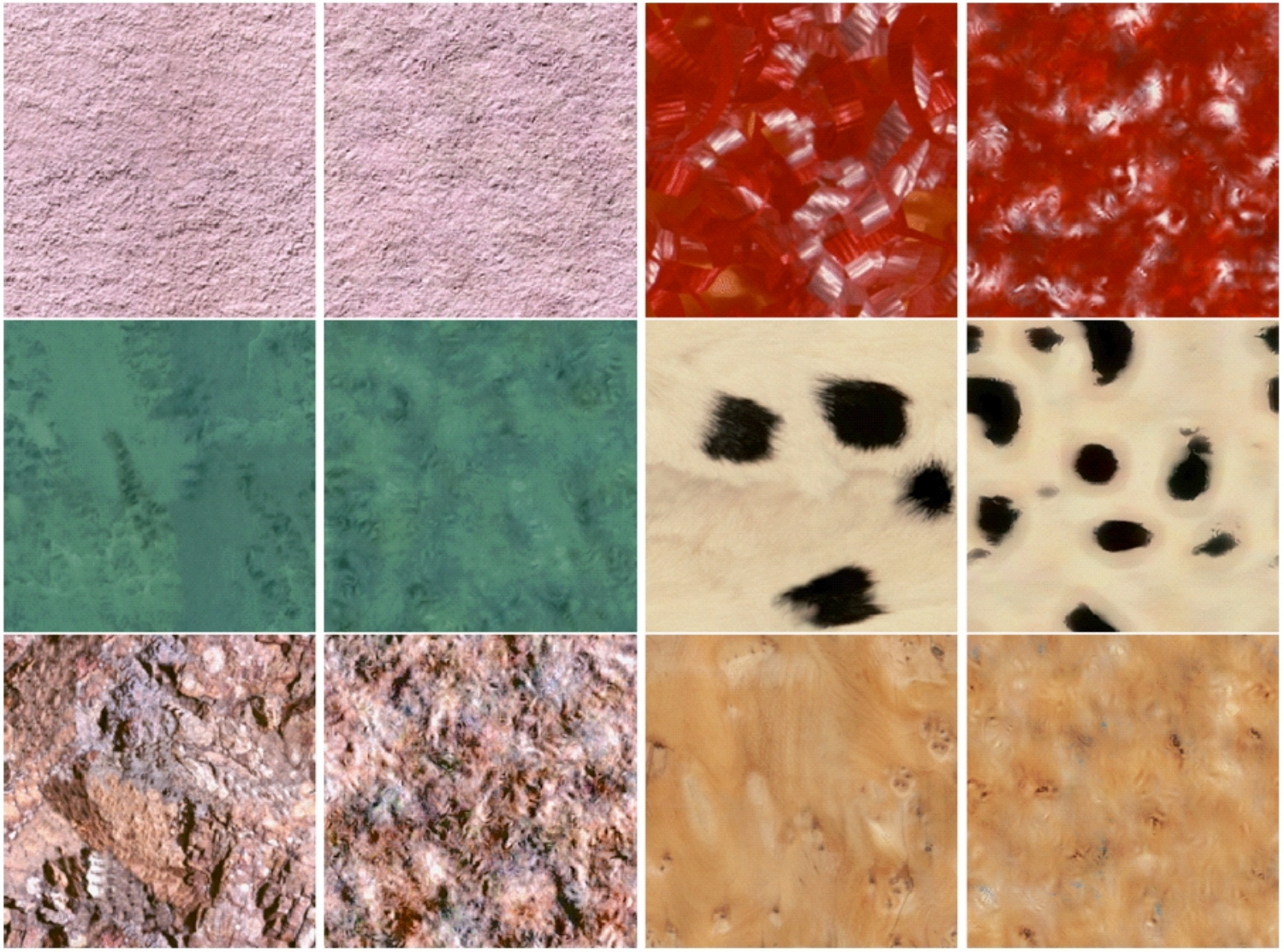
Figure 7: (Left pair) Inhomogeneous input texture produces blotchy synthetic texture. (Right pair) Homogenous input

Figure 8: Examples of failures: wood grain and red coral.



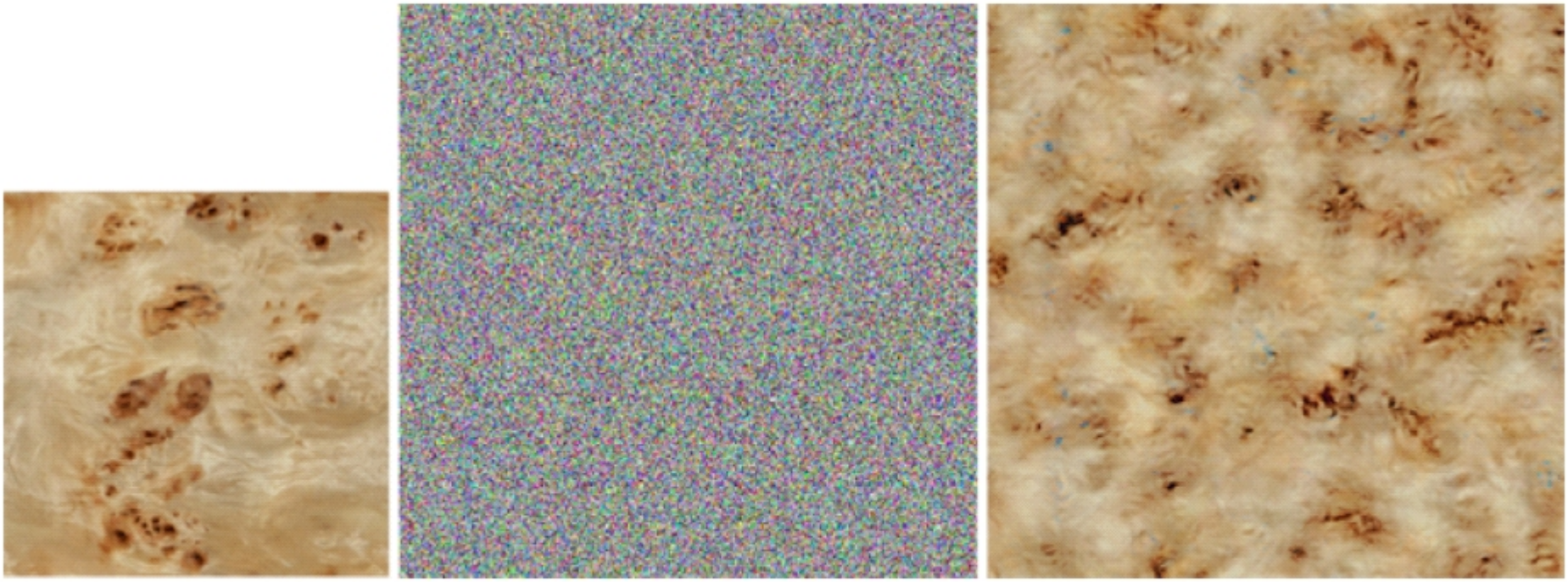
Adapted from David Forsyth, UC Berkeley

Texture Synthesis – I



Adapted from David Forsyth, UC Berkeley

Texture Synthesis – I



Adapted from David Forsyth, UC Berkeley

Texture Synthesis – II (de Bonet)

- (de Bonet)
- Take conditional structure into account

• Synthesize pyramid

– coarsest scale is the coarsest scale of the image

– elements at a finer scale are obtained by matching parent structures, choosing uniformly

• Now synthesize image from pyramid

Adapted from David Forsyth, UC Berkeley

Texture Synthesis – II

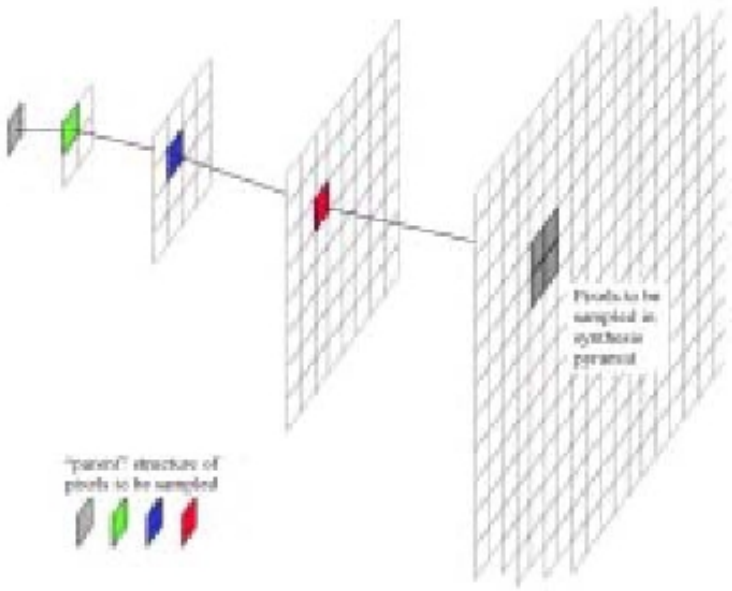


Figure 8: The distribution from which pixels in the synthesis pyramid are sampled is conditioned on the "parent" structure of those pixels. Each element of the parent structure contains a vector of the feature measurements at that location and scale.

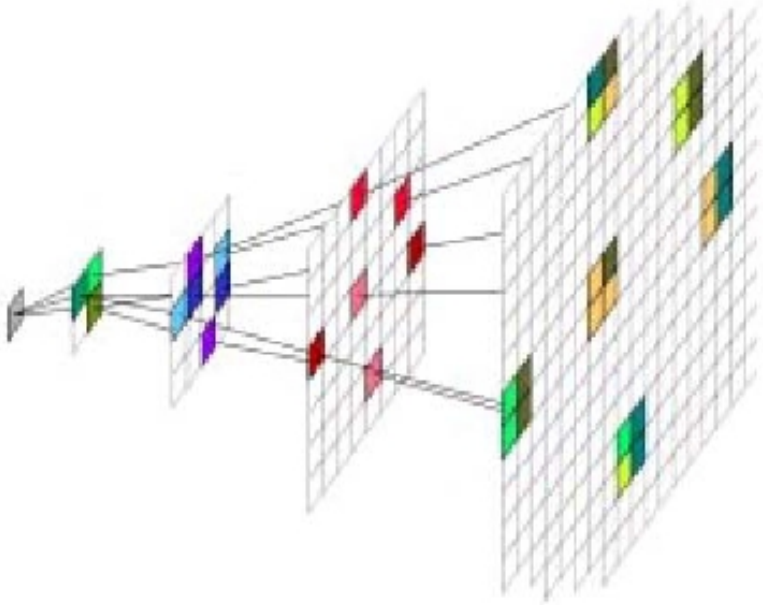
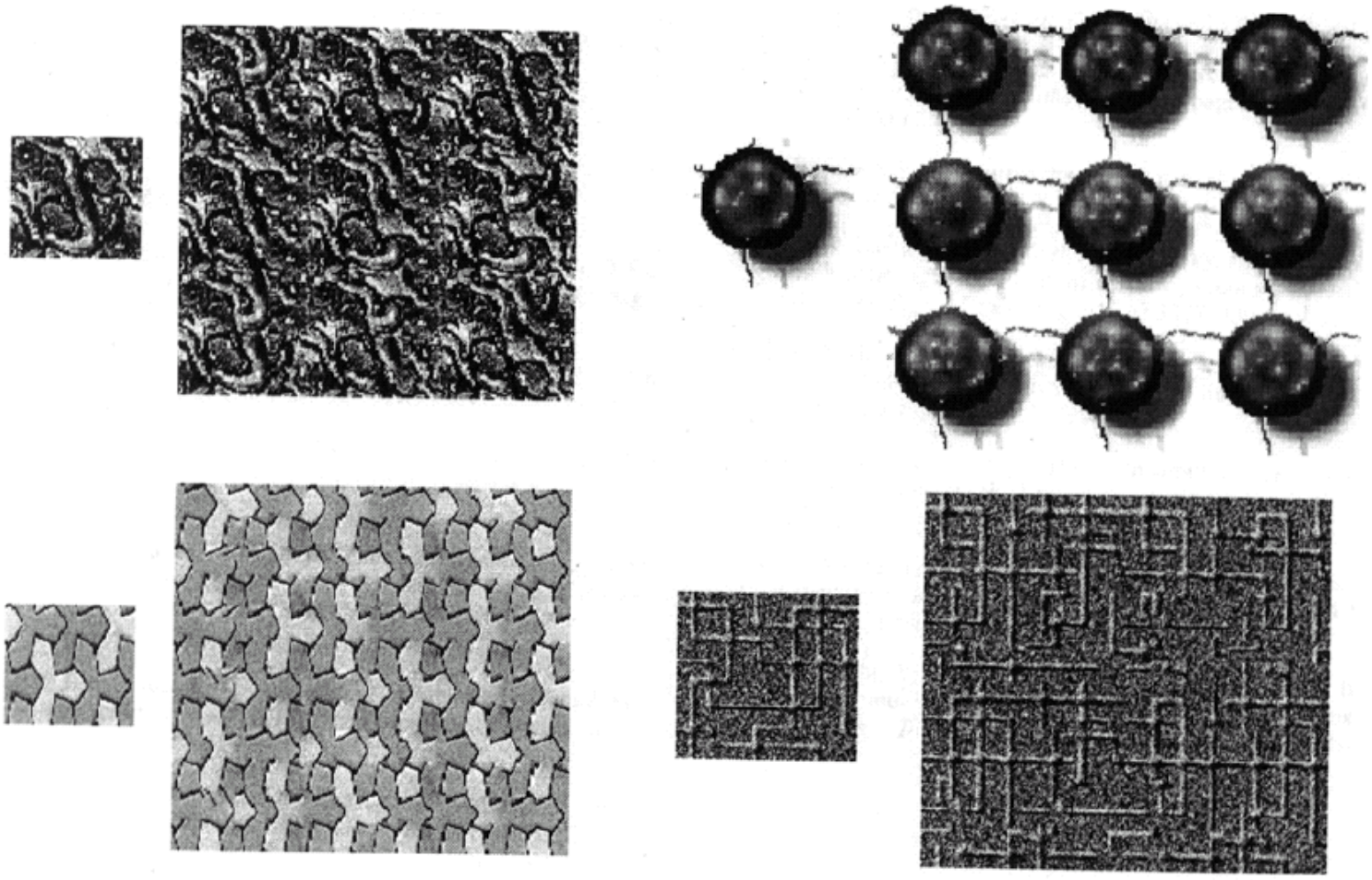


Figure 9: An input texture is decomposed to form an analysis pyramid, from which a new synthesis pyramid is sampled, conditioned on local features within the pyramids. A filter bank of local texture measures, based on psychophysical models, are used as features.

Texture Synthesis – II

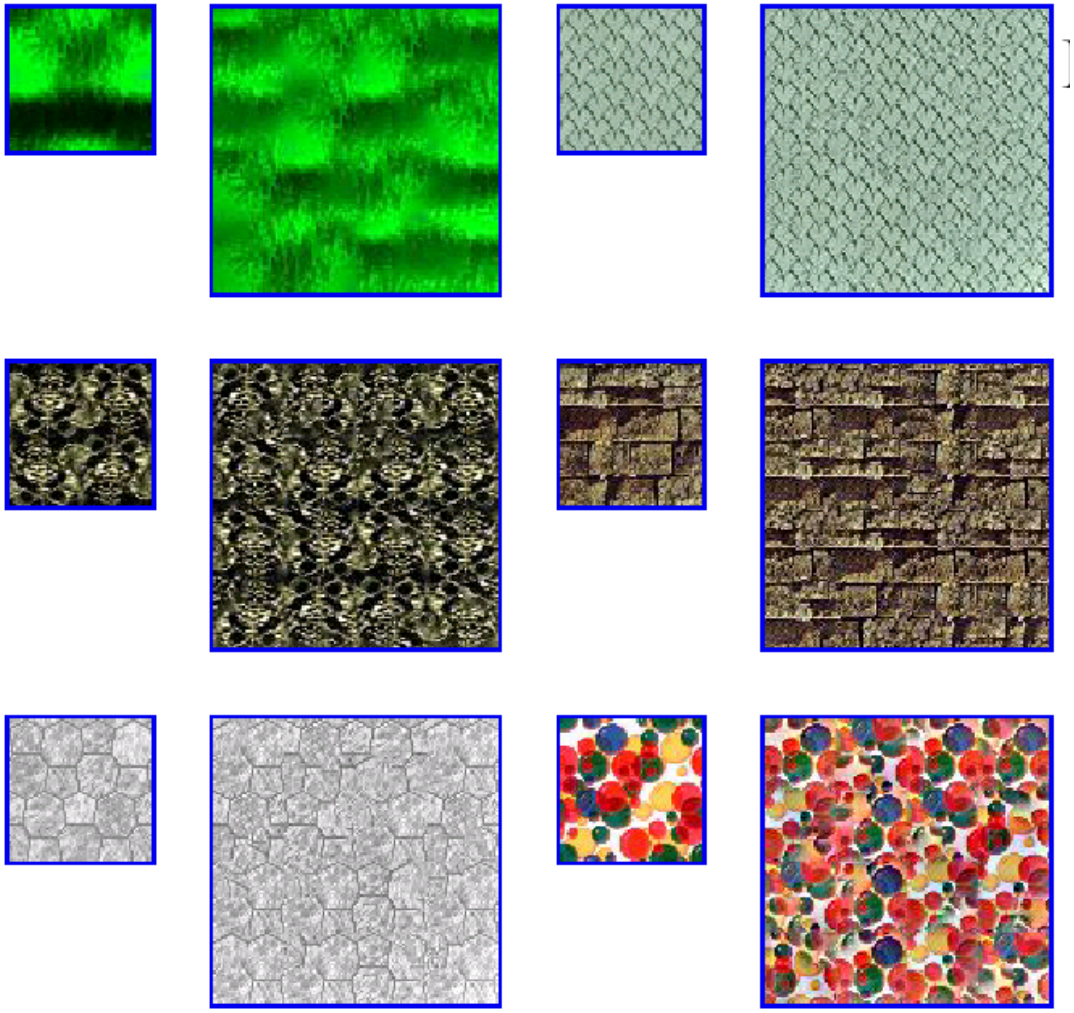
```
make a pyramid pe from the example image
make an empty pyramid pw, corresponding to the image to
be synthesized

set the coarsest scale layer of pw to be the same as the
coarsest scale level of pe; if pw is bigger than pe, then
replicate copies of pe to fill it

for each other layer l of pe, going from coarsest to finest
  for each element e of the layer

    obtain all elements with
    the same parent structure

    choose one of this collection uniformly at random

    insert the value of this element into e

  end
end

synthesize the texture image from the pyramid pw
```

Texture Synthesis – II

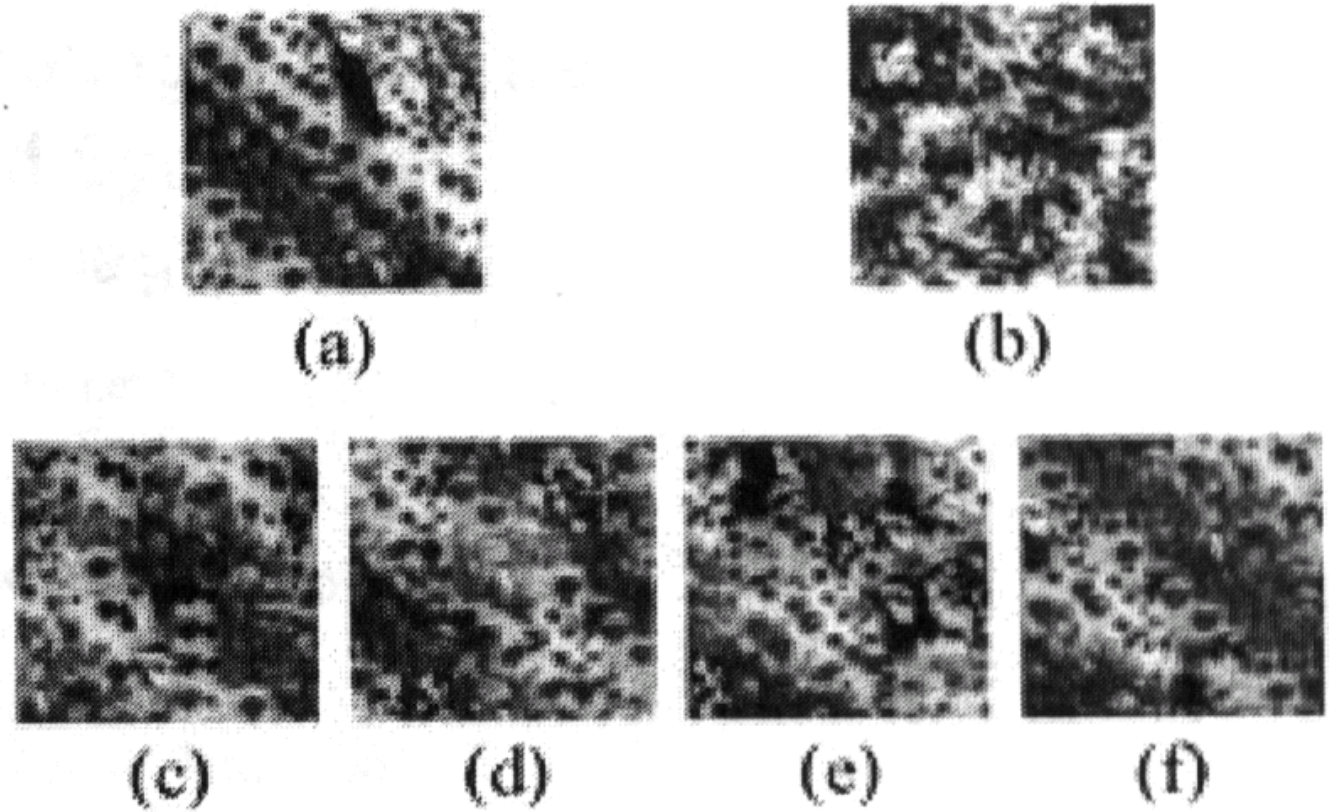


Adapted from David Forsyth, UC Berkeley

Texture Synthesis – II



Texture Synthesis – II



The texture synthesized by histogram equalization is shown on the top right. The bottom row shows textures synthesized using DeBonet's algorithm , which doesn't require an independence assumption.

Adapted from David Forsyth, UC Berkeley

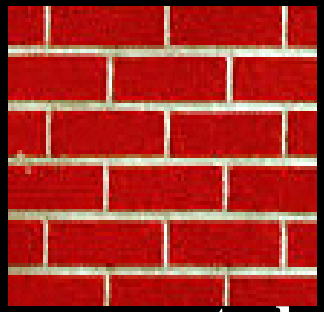
Texture Synthesis–III (Efros&Leung,Efros&Freeman)

- Use image as a source of probability model
- Choose pixel values by matching neighbourhood, then filling in
- Matching process
  - look at pixel differences
  - count only synthesized pixels

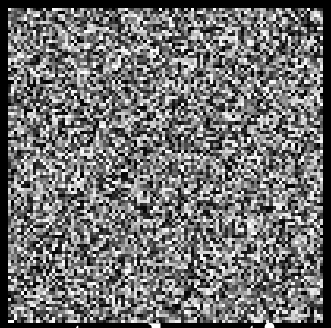
Following slides are from Alyosha Efros, CMU

The Challenge

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



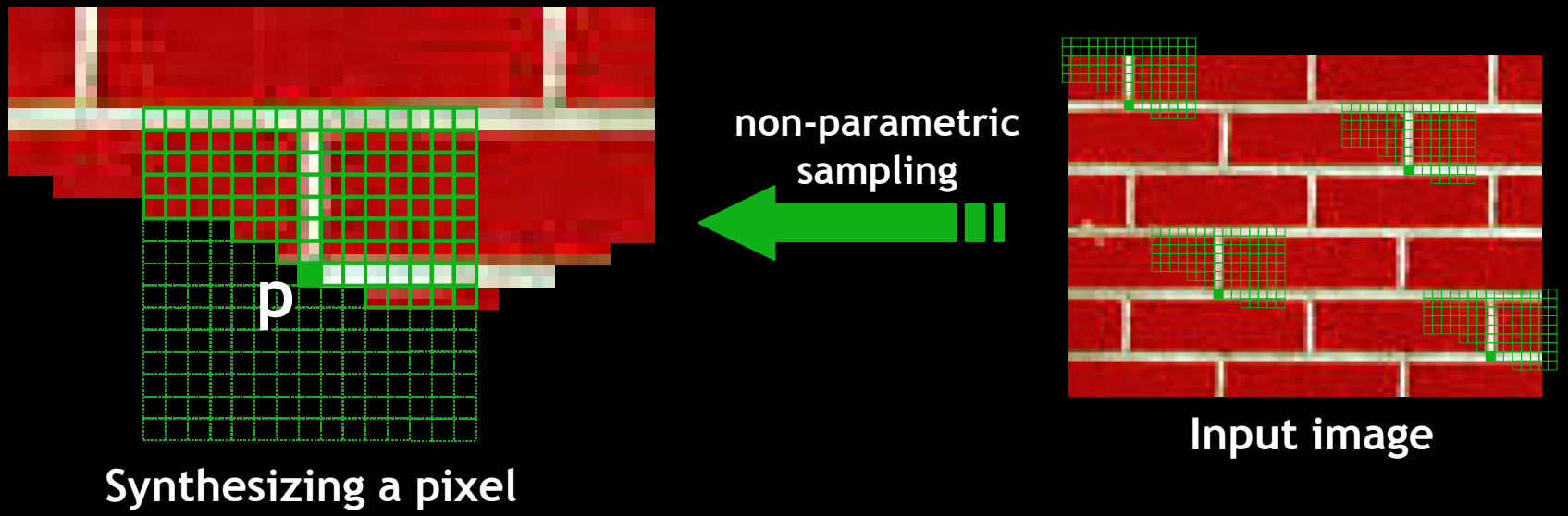
repeated



stochastic

Both?

Efros & Leung Algorithm

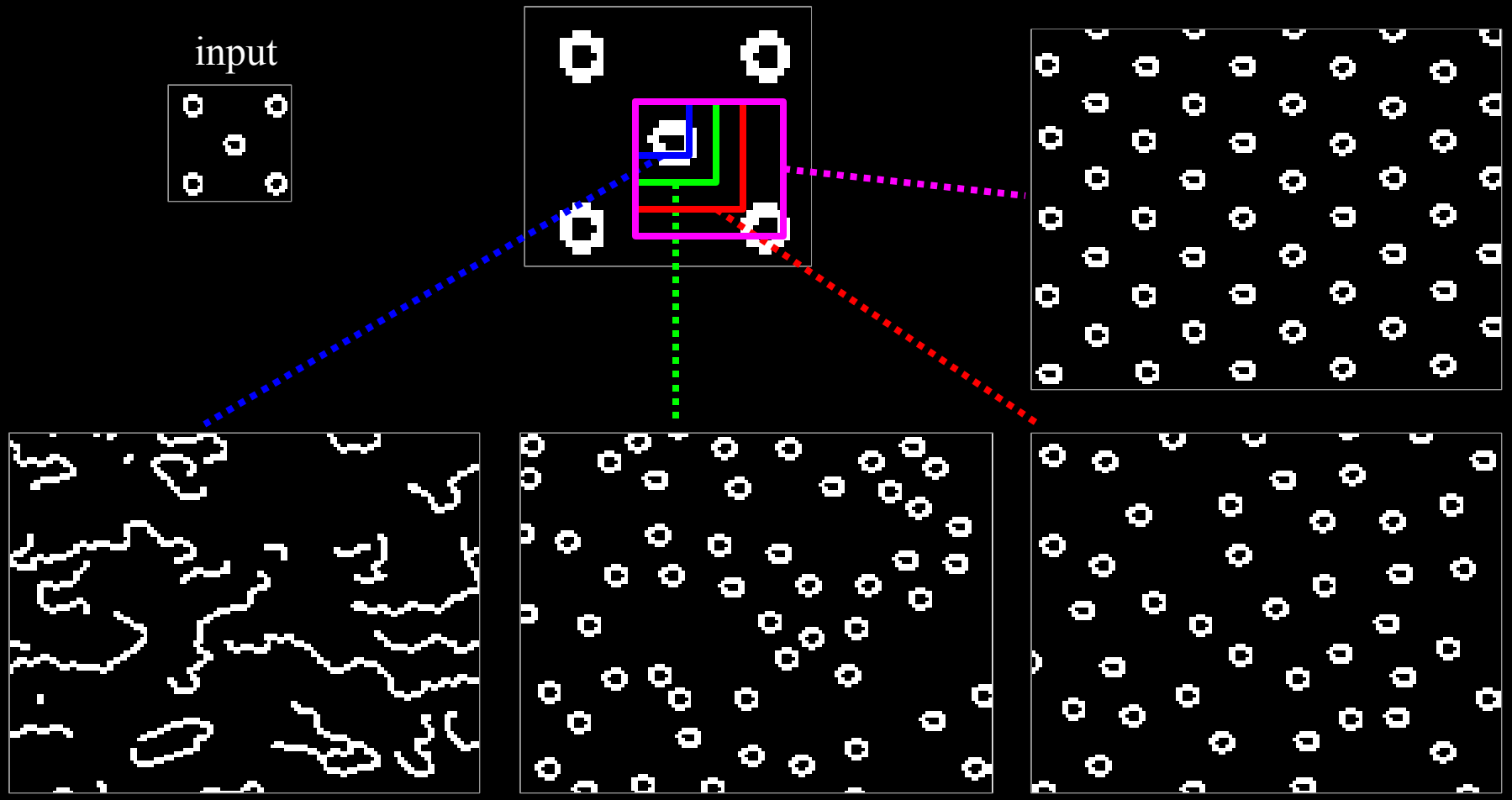


- Assuming Markov property, compute P(**p**|N(**p**))
  - Building explicit probability tables infeasible
  - Instead, we *search the input image* for all similar neighborhoods — that's our pdf for **p**
  - To sample from this pdf, just pick one match at random

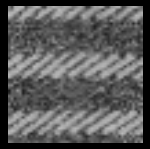
Some Details

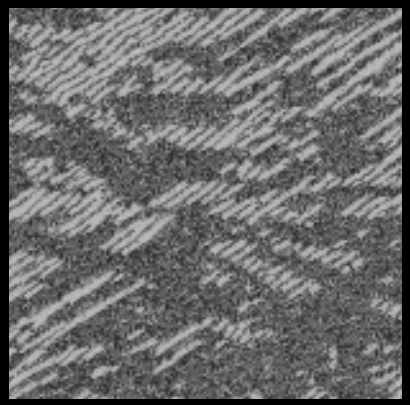
- Growing is in "onion skin" order
  - Within each "layer", pixels with most neighbors are synthesized first
  - If no close match can be found, the pixel is not synthesized until the end
- Using *Gaussian-weighted* SSD is very important
  - to make sure the new pixel agrees with its closest neighbors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

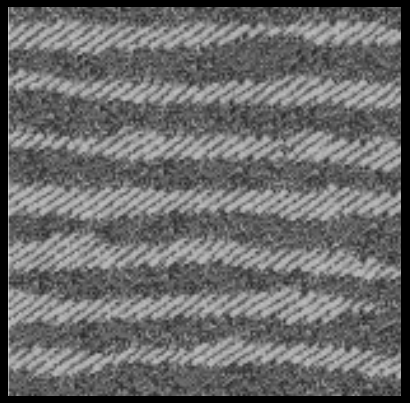
Neighborhood Window

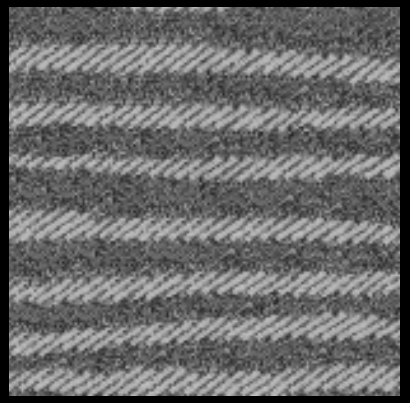


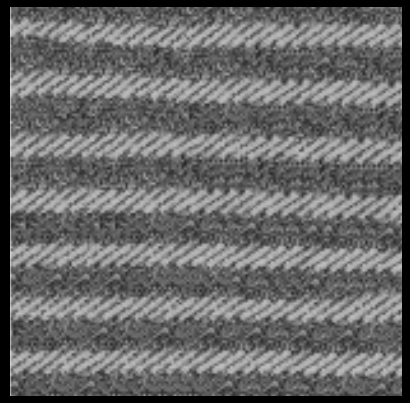
Varying Window Size

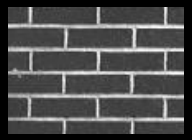


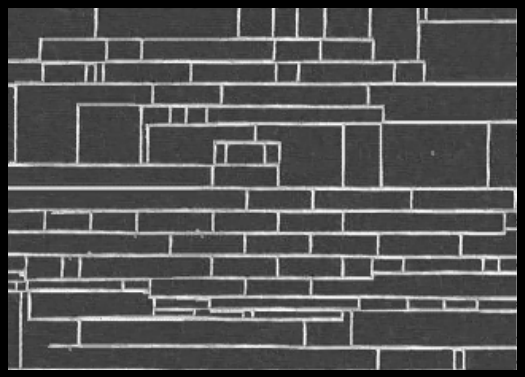


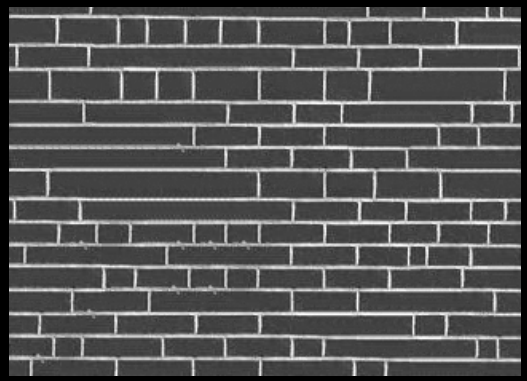


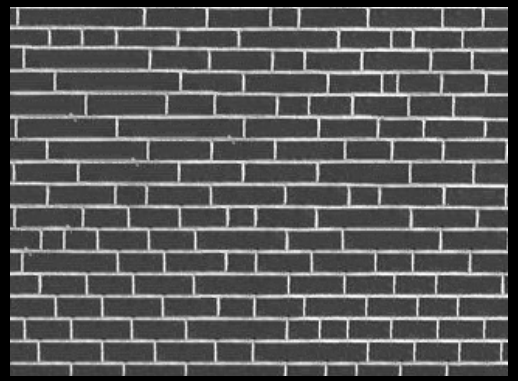








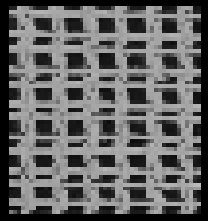


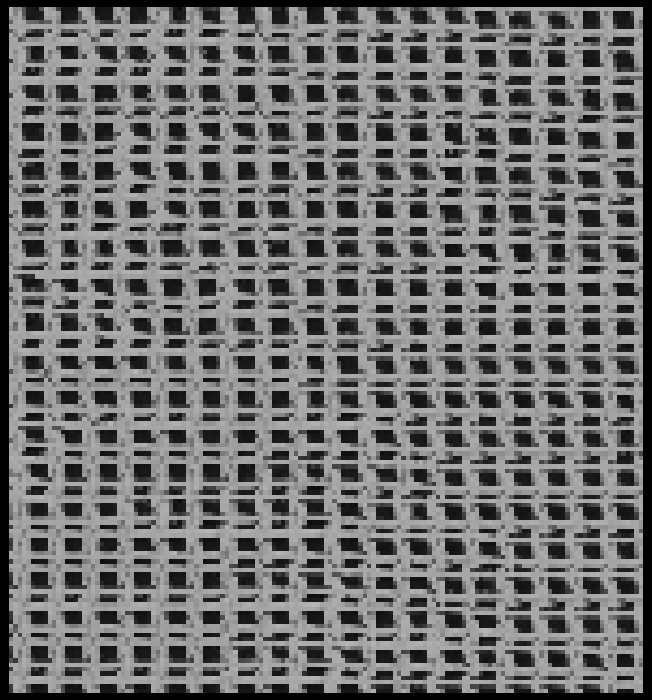


Increasing window size

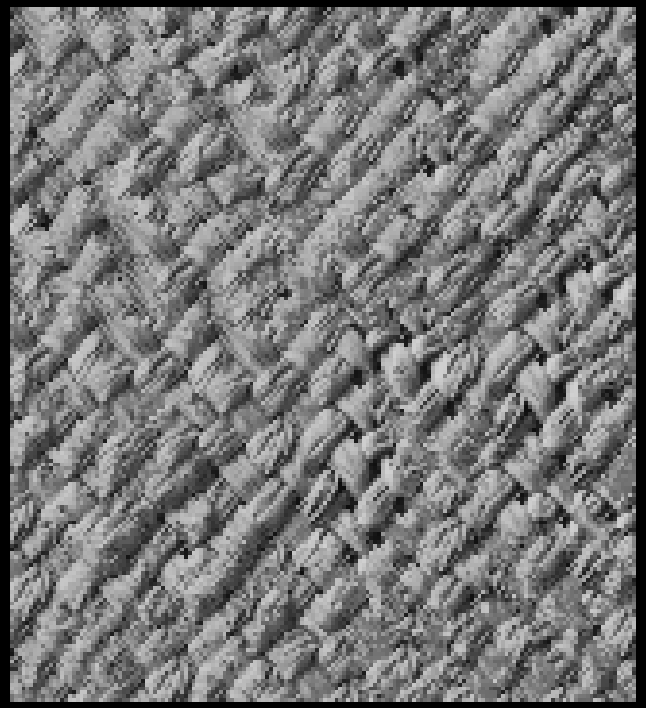
Synthesis Results

french canvas



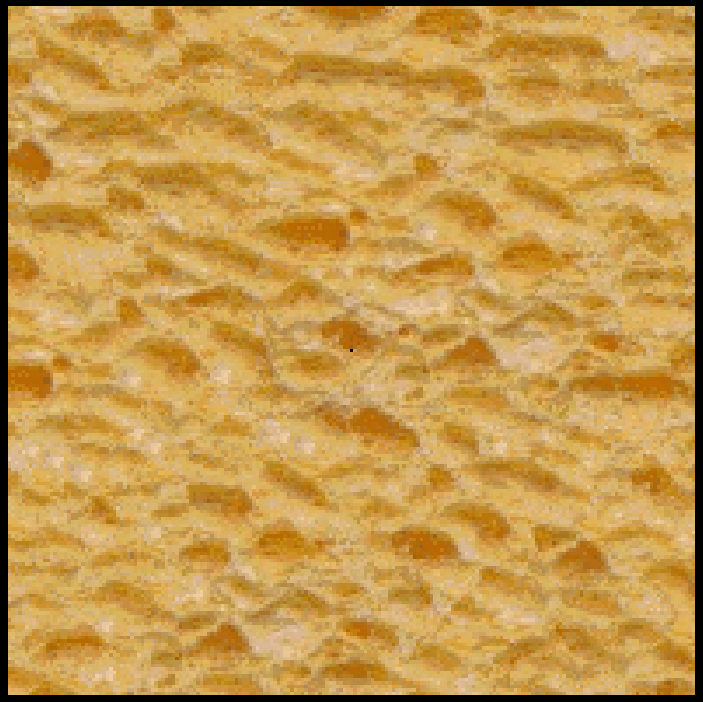


rafia weave

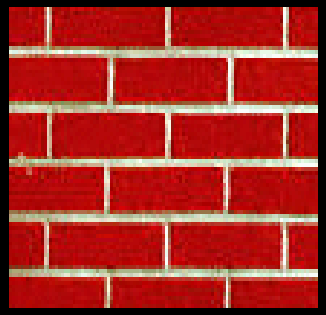


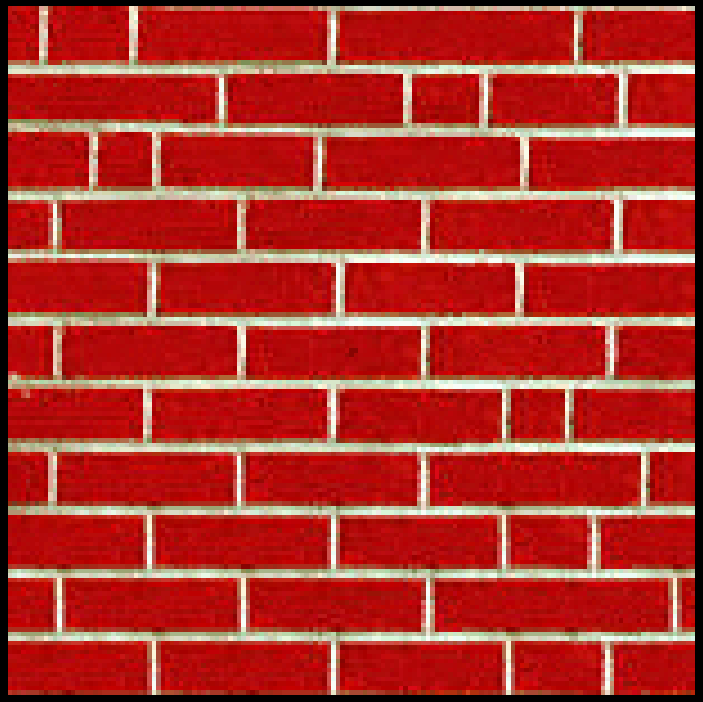
More Results

white bread

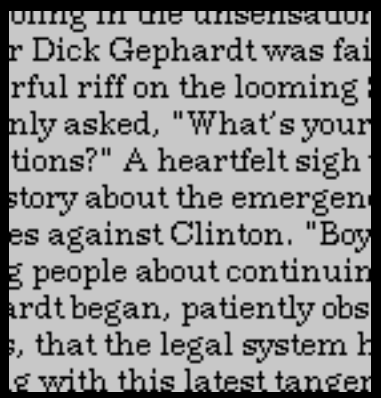


brick wall

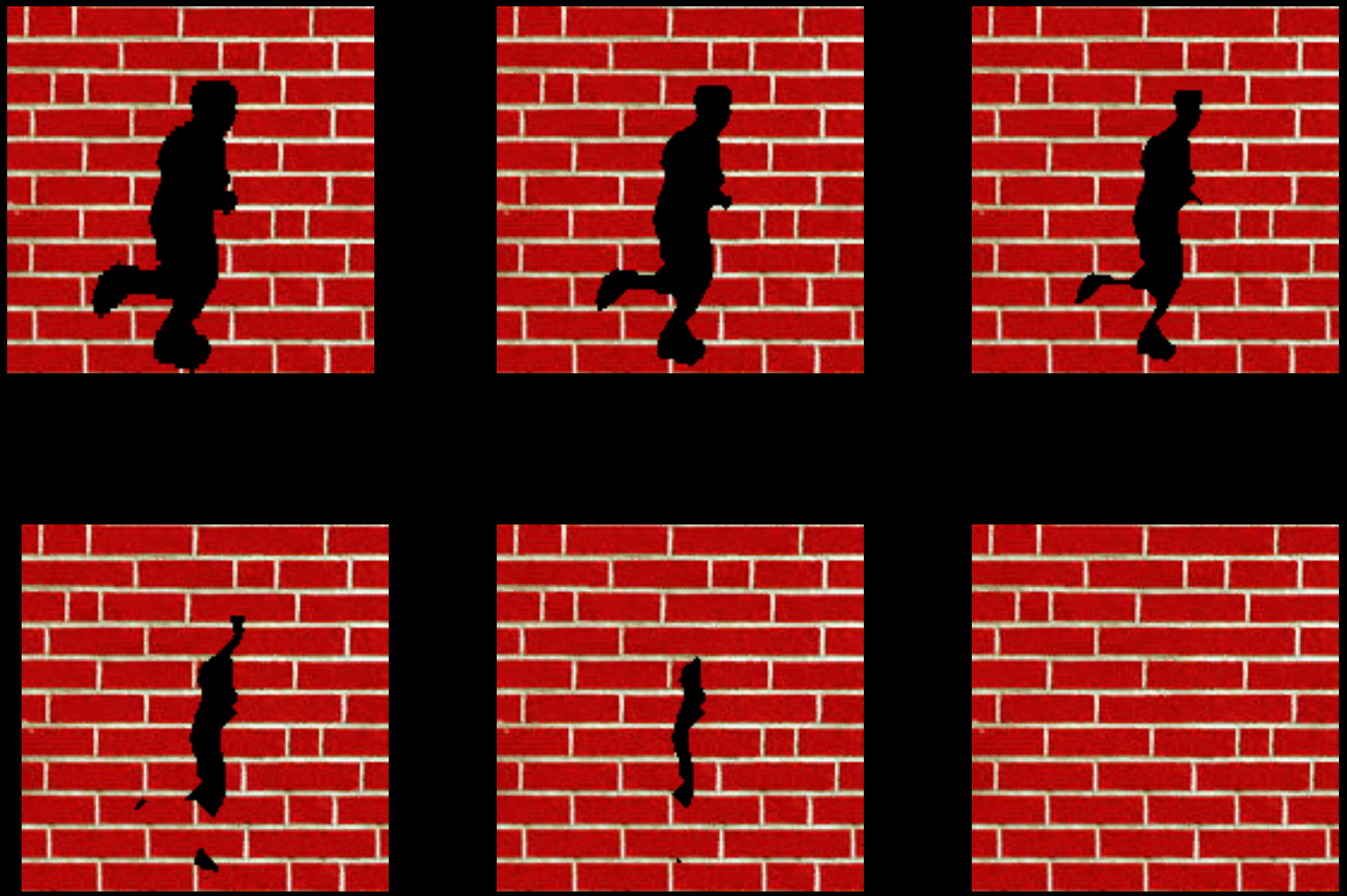




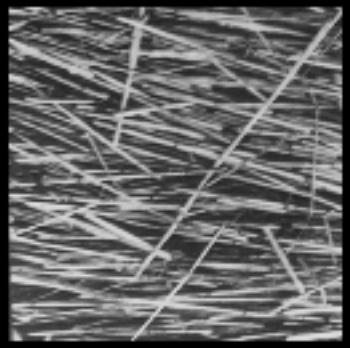
Homage to Shannon

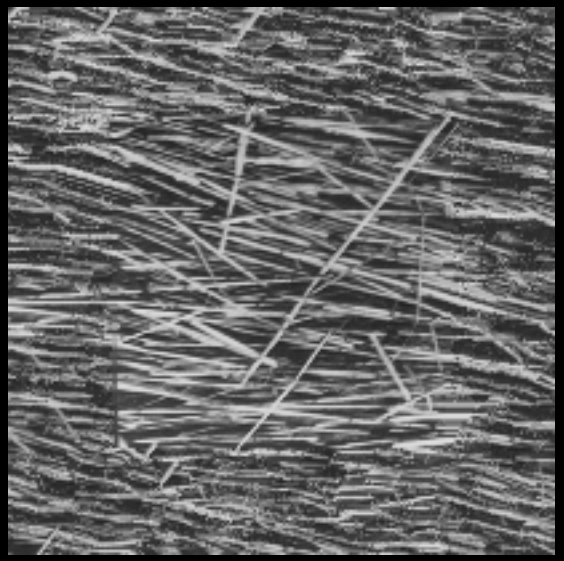


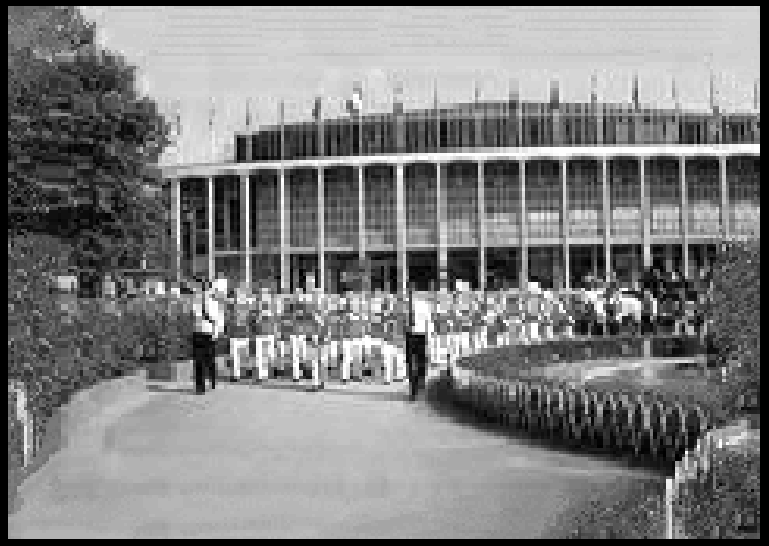
Hole Filling



Extrapolation



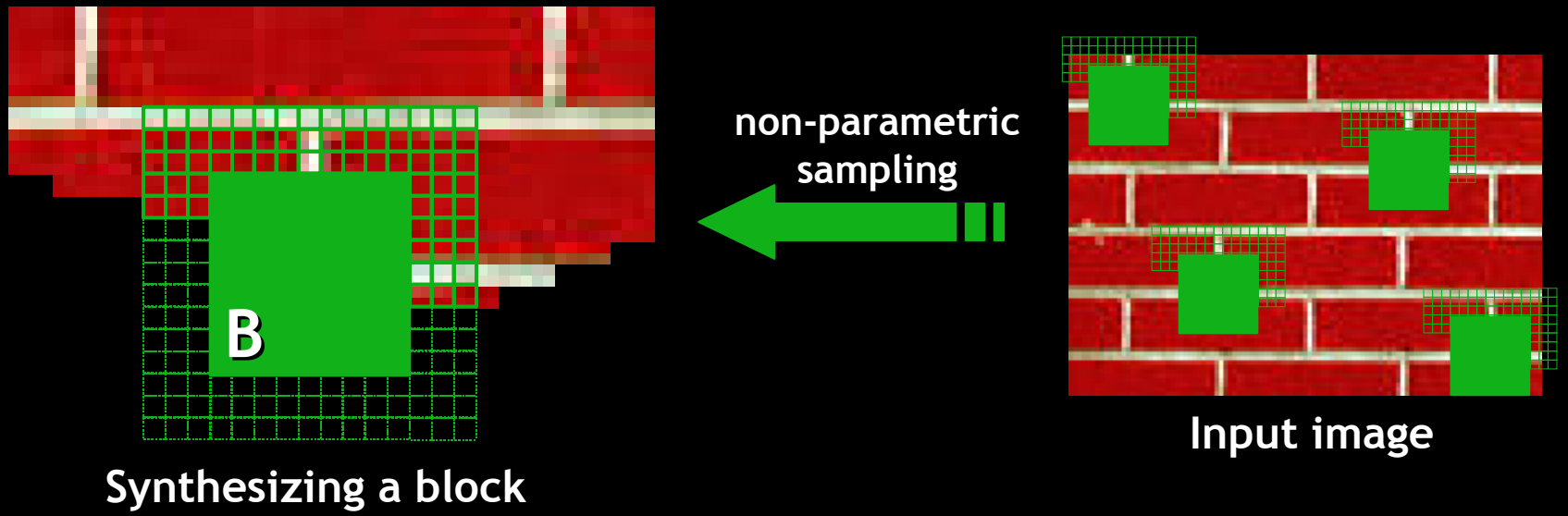




Summary

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

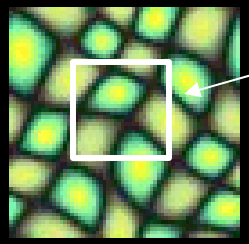
Image Quilting [Efros & Freeman]



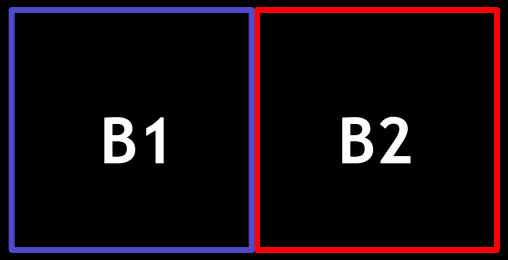
- Observation: neighbor pixels are highly correlated

Idea: unit of synthesis = block

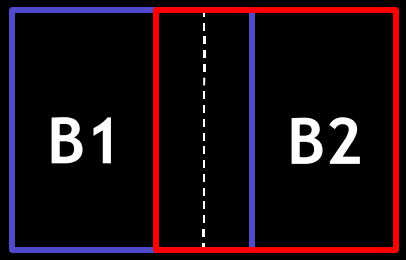
- Exactly the same but now we want P(**B**|N(**B**))
- Much faster: synthesize all pixels in a block at once
- Not the same as multi-scale!



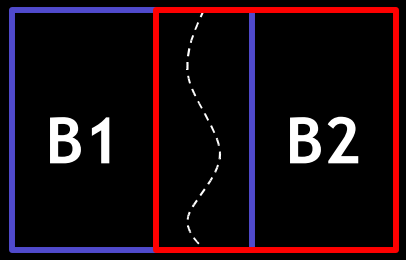
Input texture



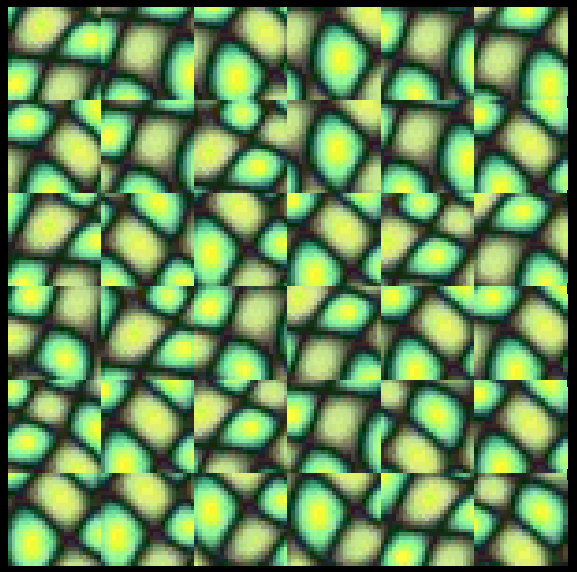
Random placement of blocks

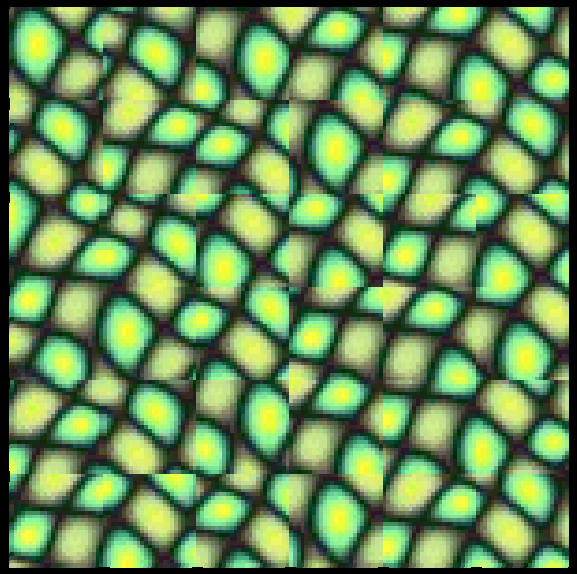


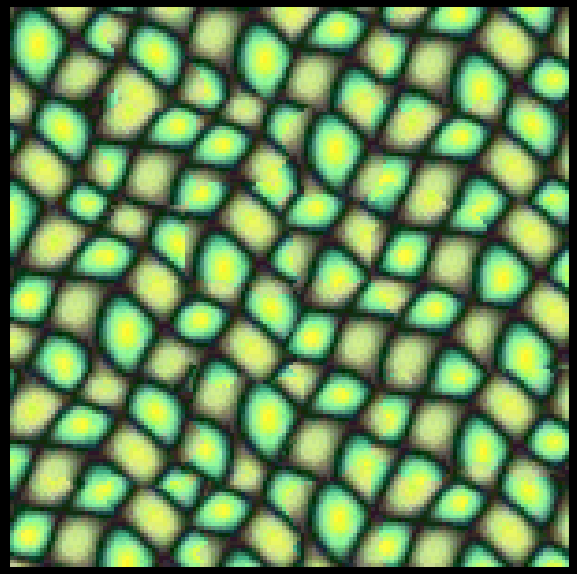
Neighboring blocks constrained by overlap



Minimal error boundary cut

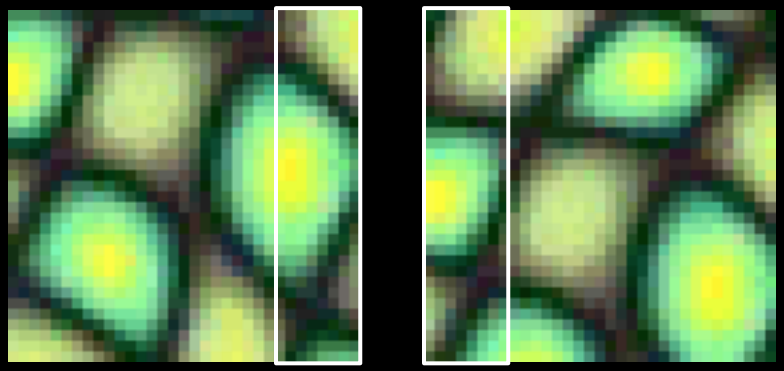




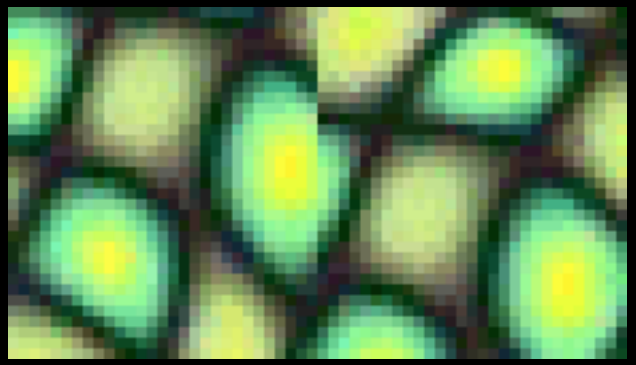


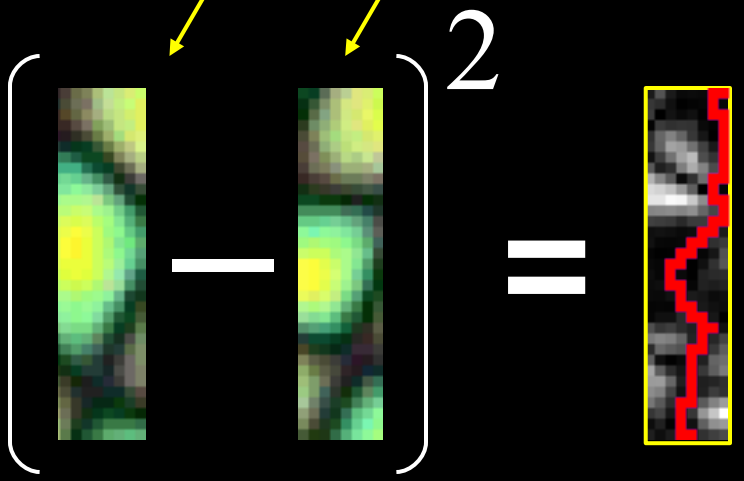
Minimal error boundary

overlapping blocks

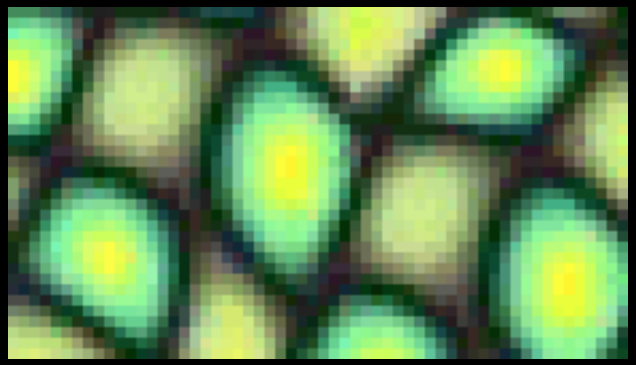


vertical boundary





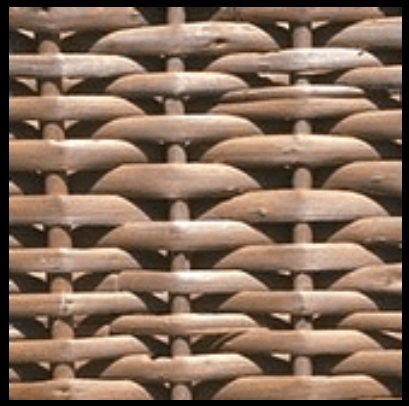
overlap error

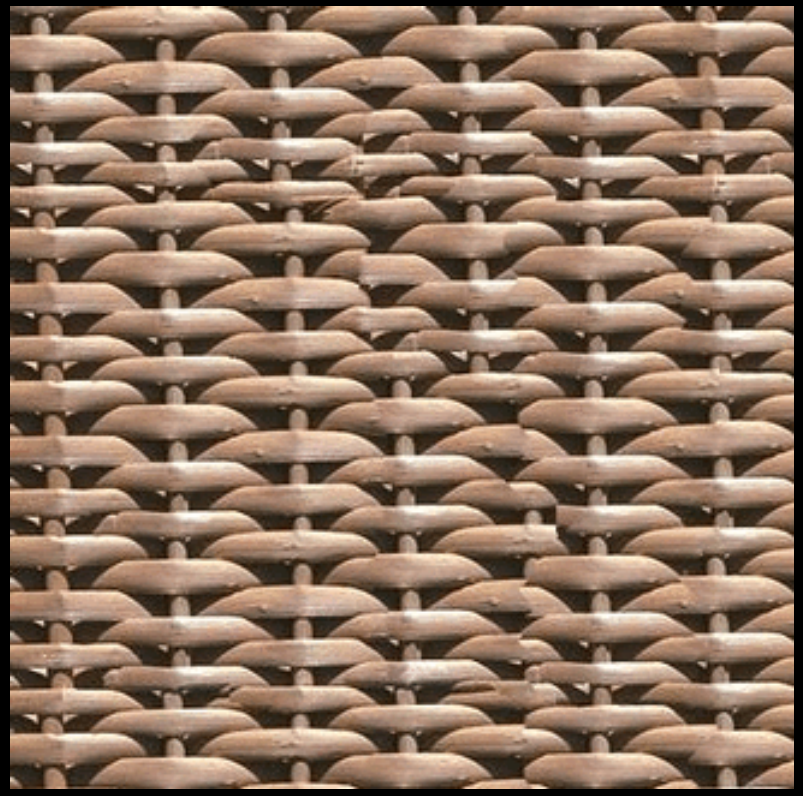


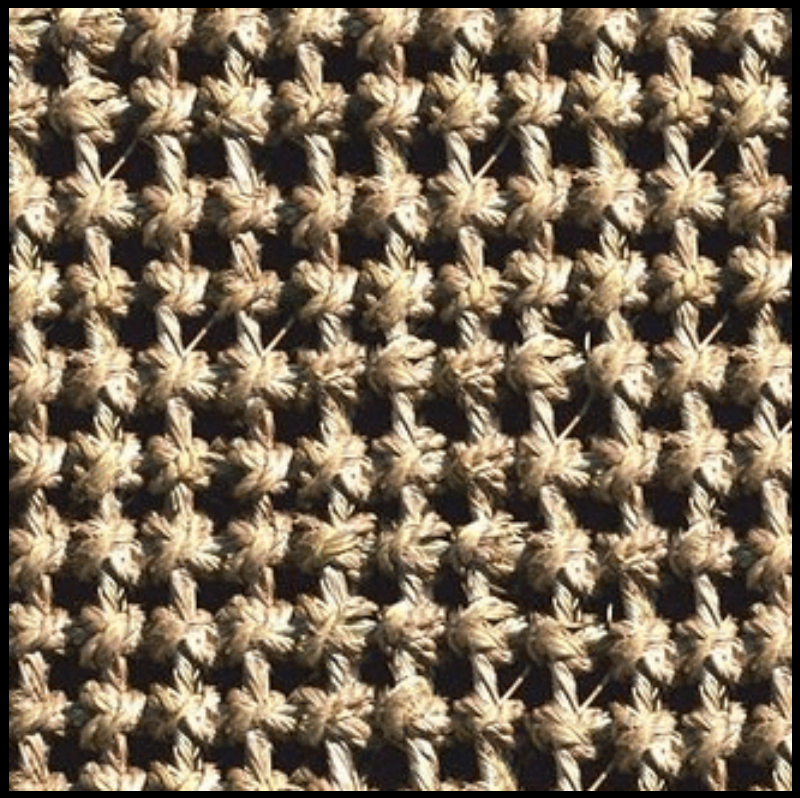
min. error boundary

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

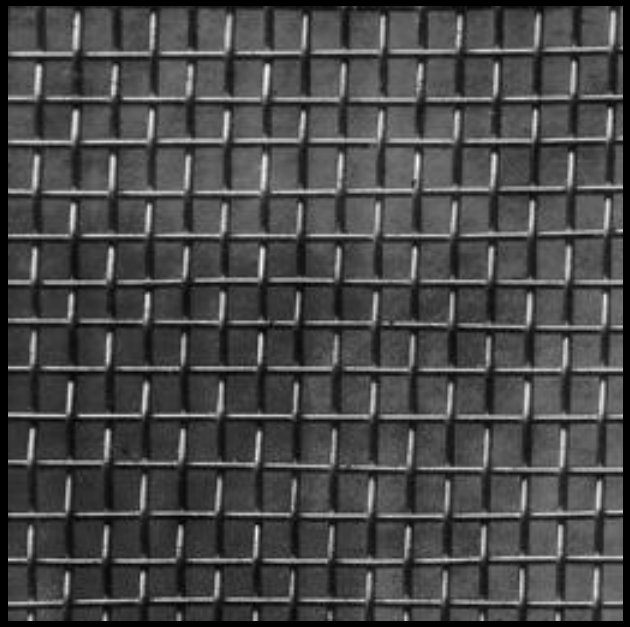




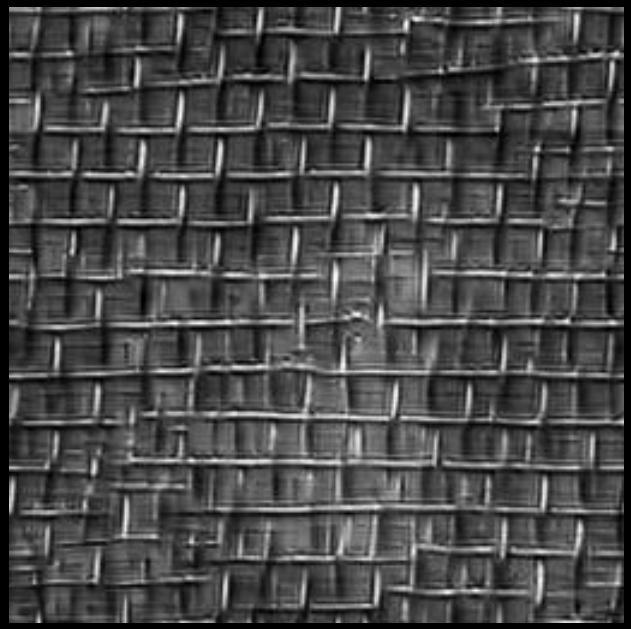



College Inn
Chicken Broth

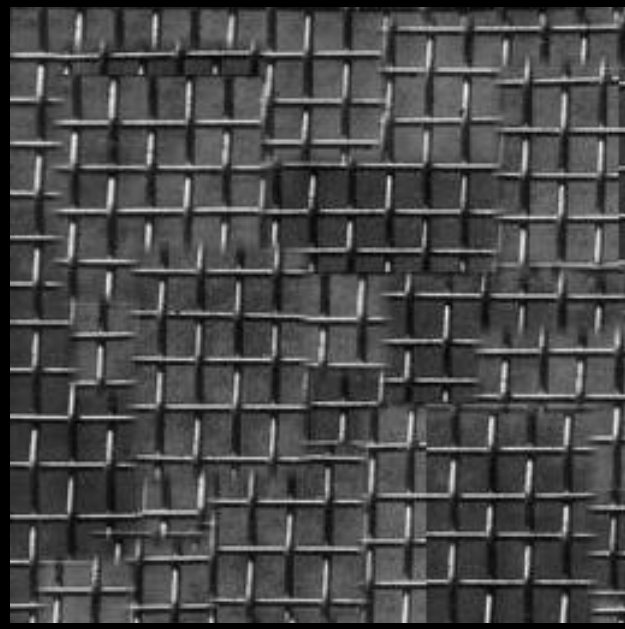

College Inn
Chicken Broth



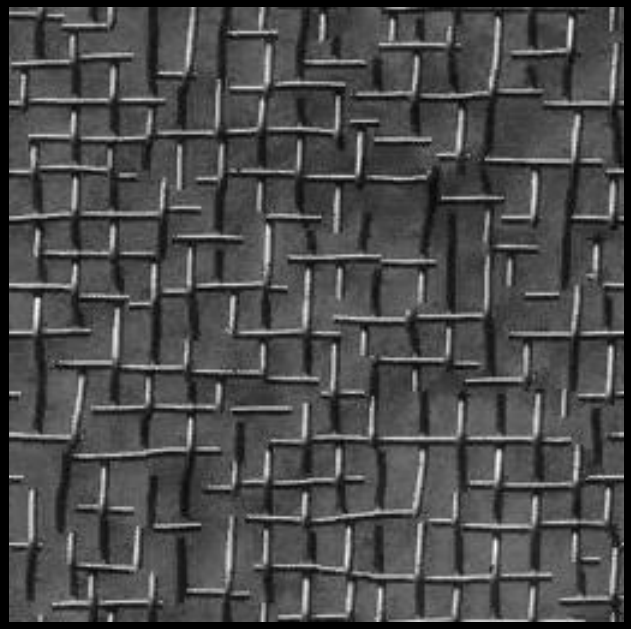
input image



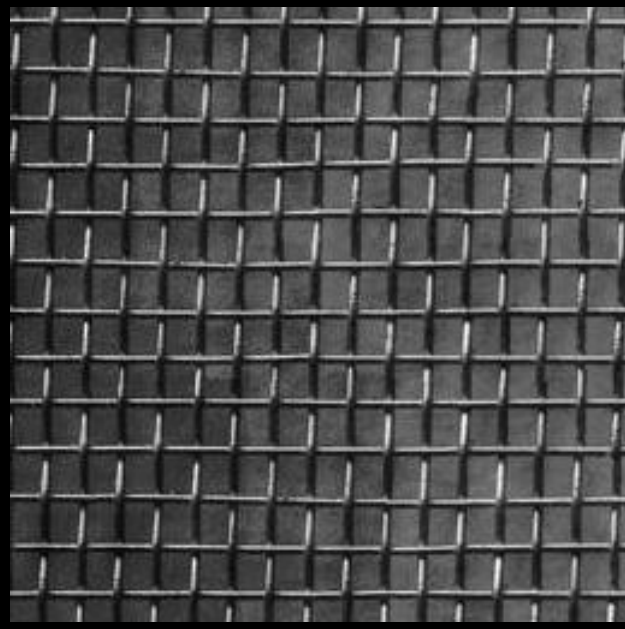
Portilla & Simoncelli



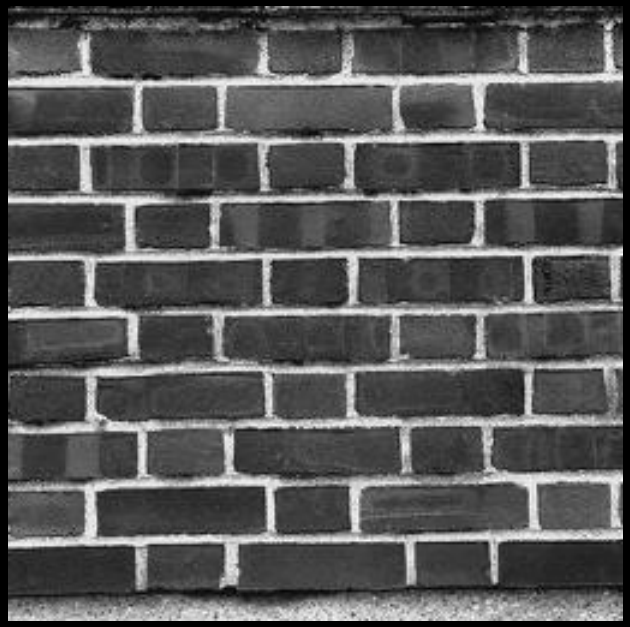
Xu, Guo & Shum



Wei & Levoy

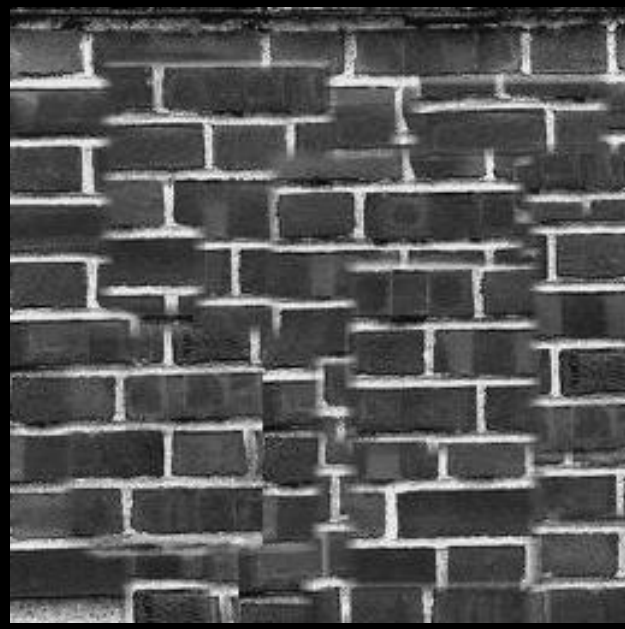


Our algorithm

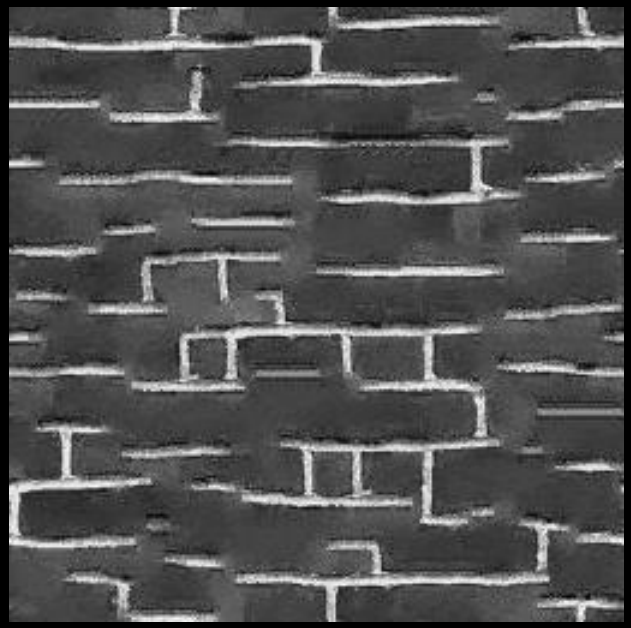


input image

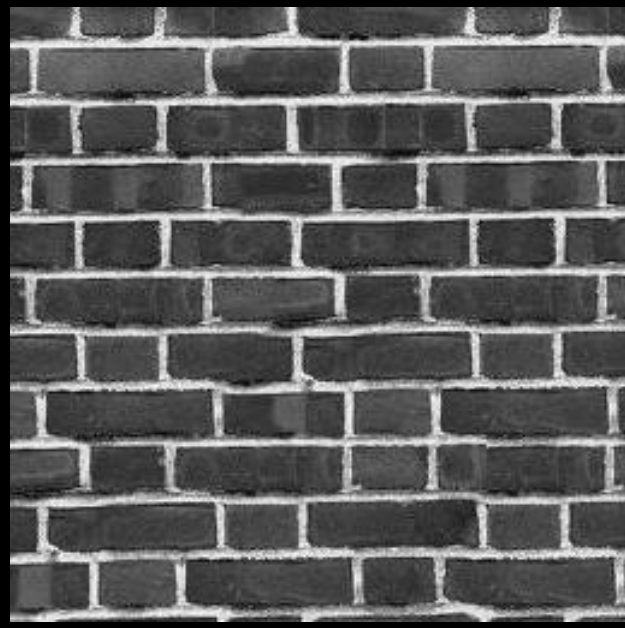
Portilla & Simoncelli



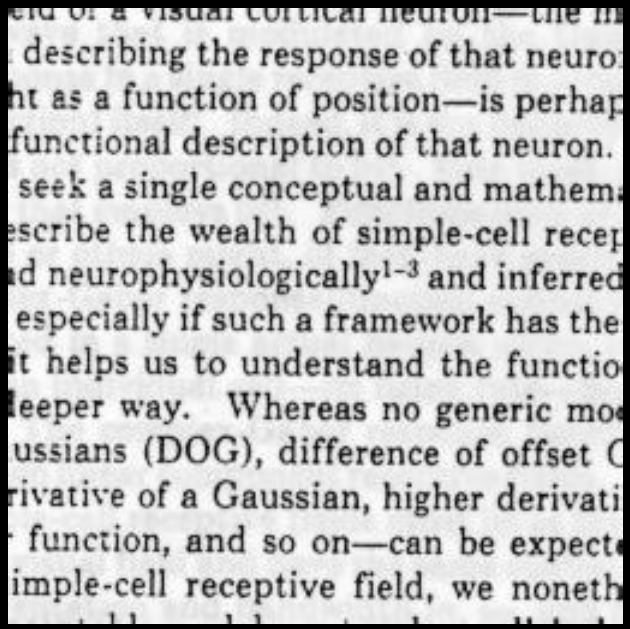
Xu, Guo & Shum



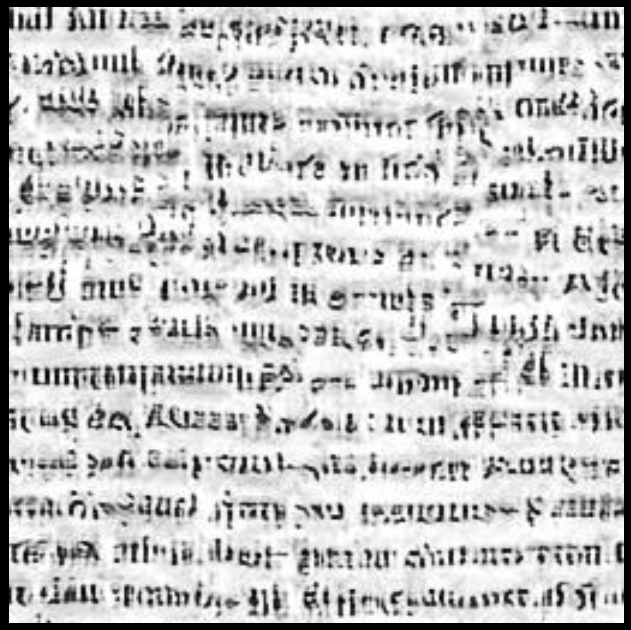
Wei & Levoy



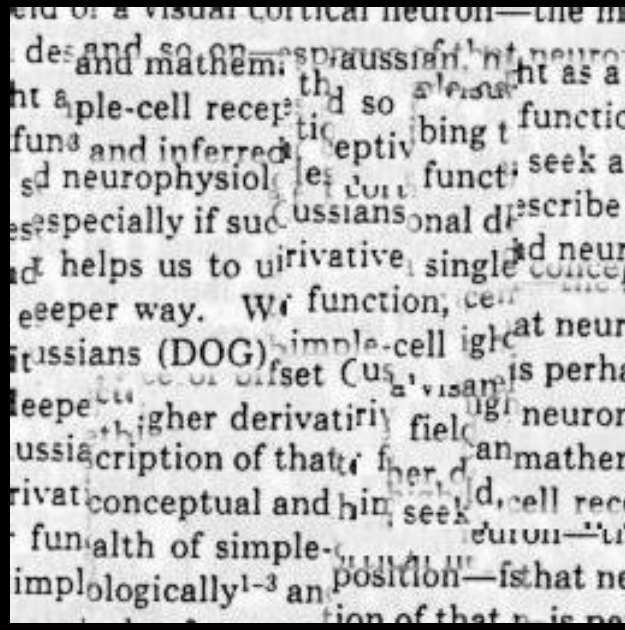
Our algorithm



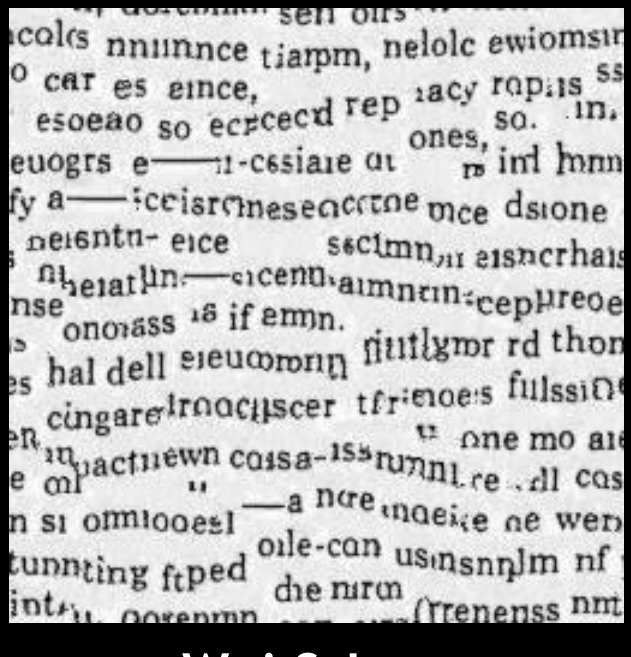
input image



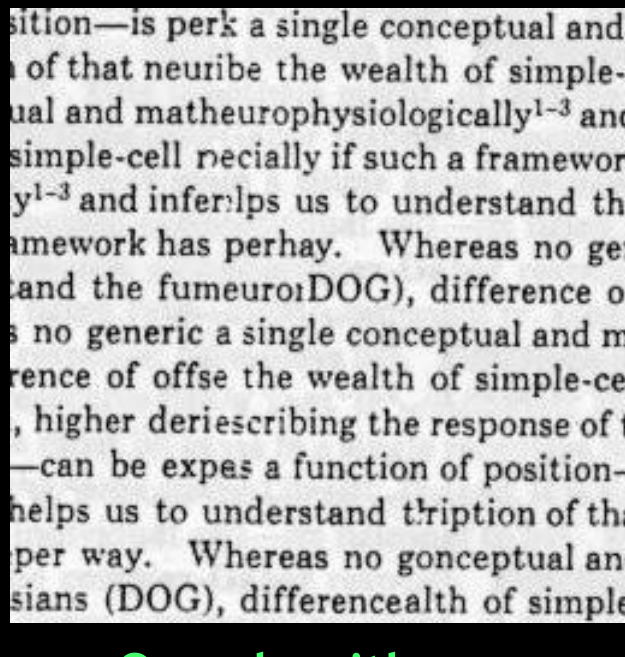
Portilla & Simoncelli



Xu, Guo & Shum

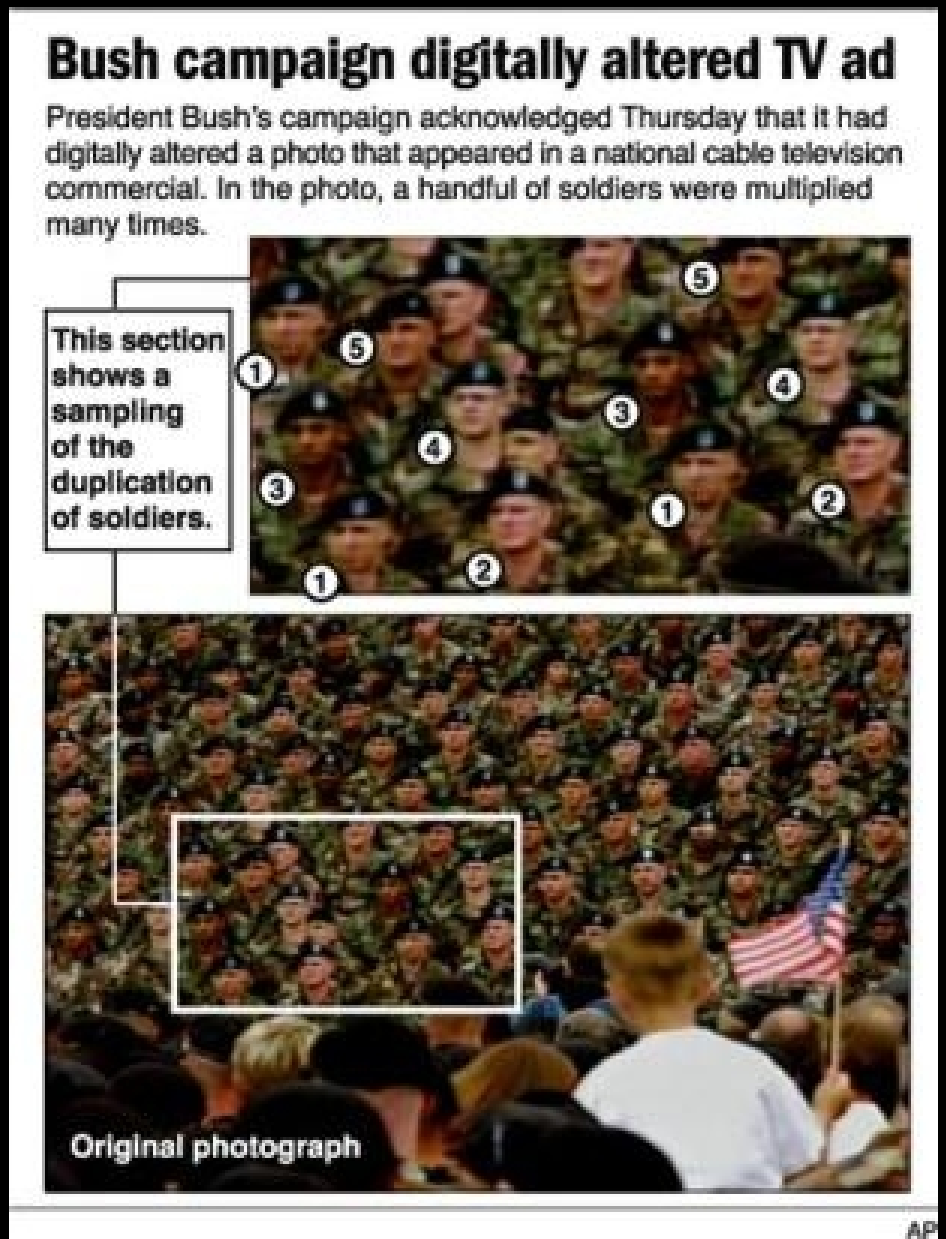


Wei & Levoy



Our algorithm

Political Texture Synthesis!

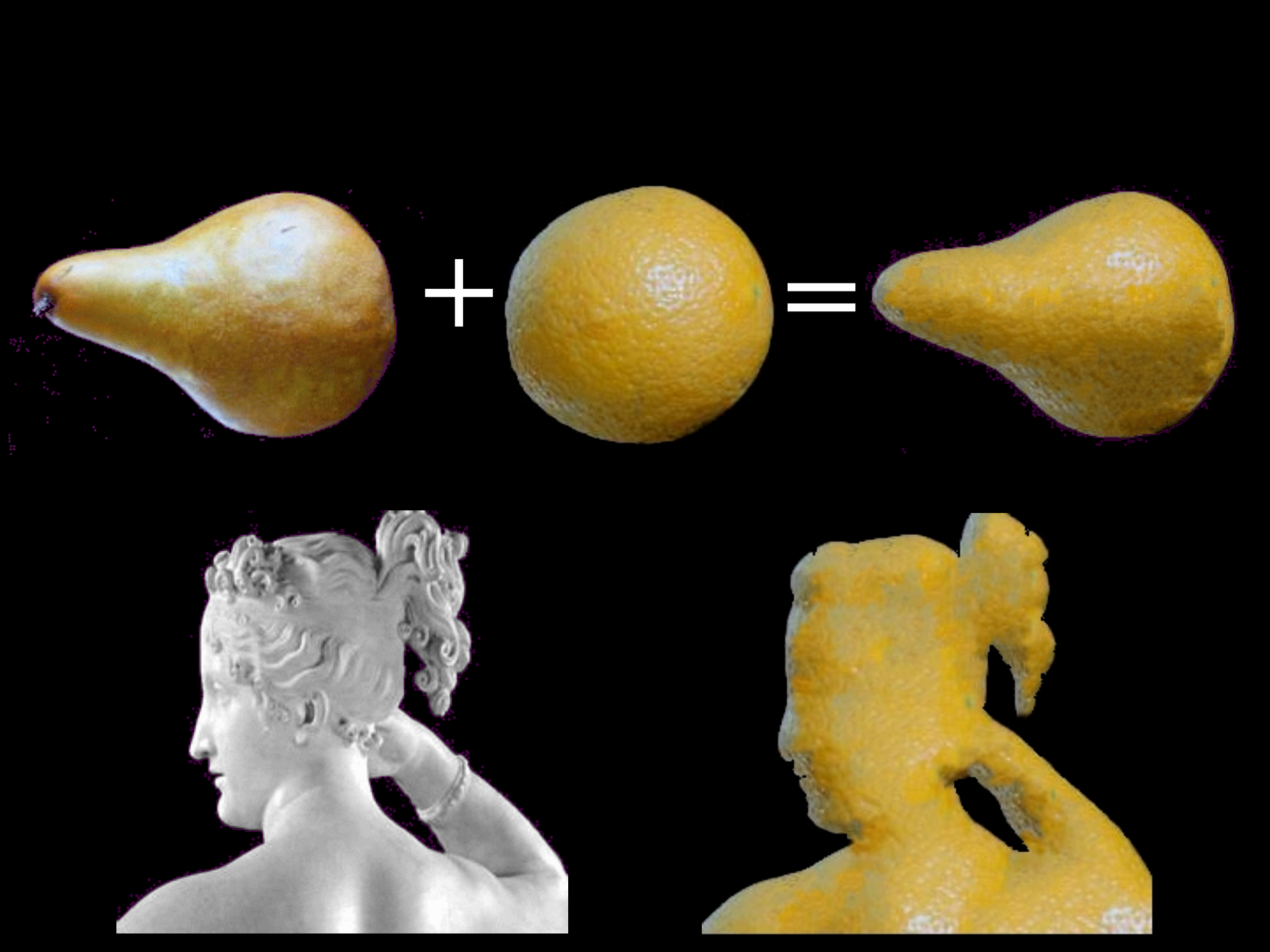


Application: Texture Transfer

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

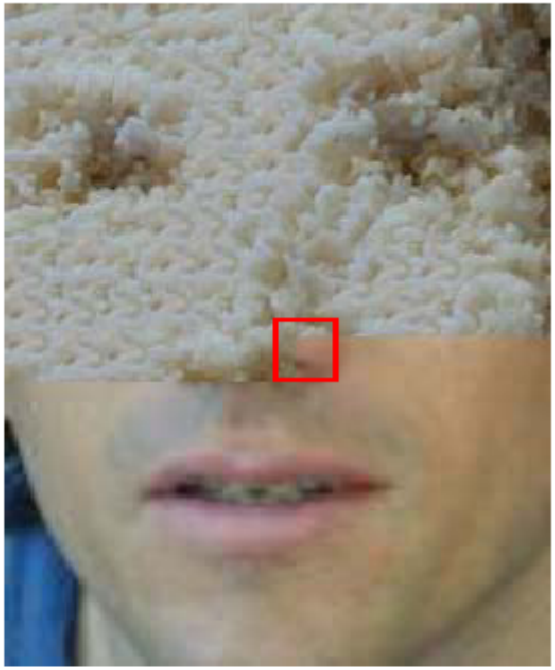
Same as texture synthesis, except an additional constraint:

1. Consistency of texture
2. Similarity to the image being "explained"



Texture Transfer

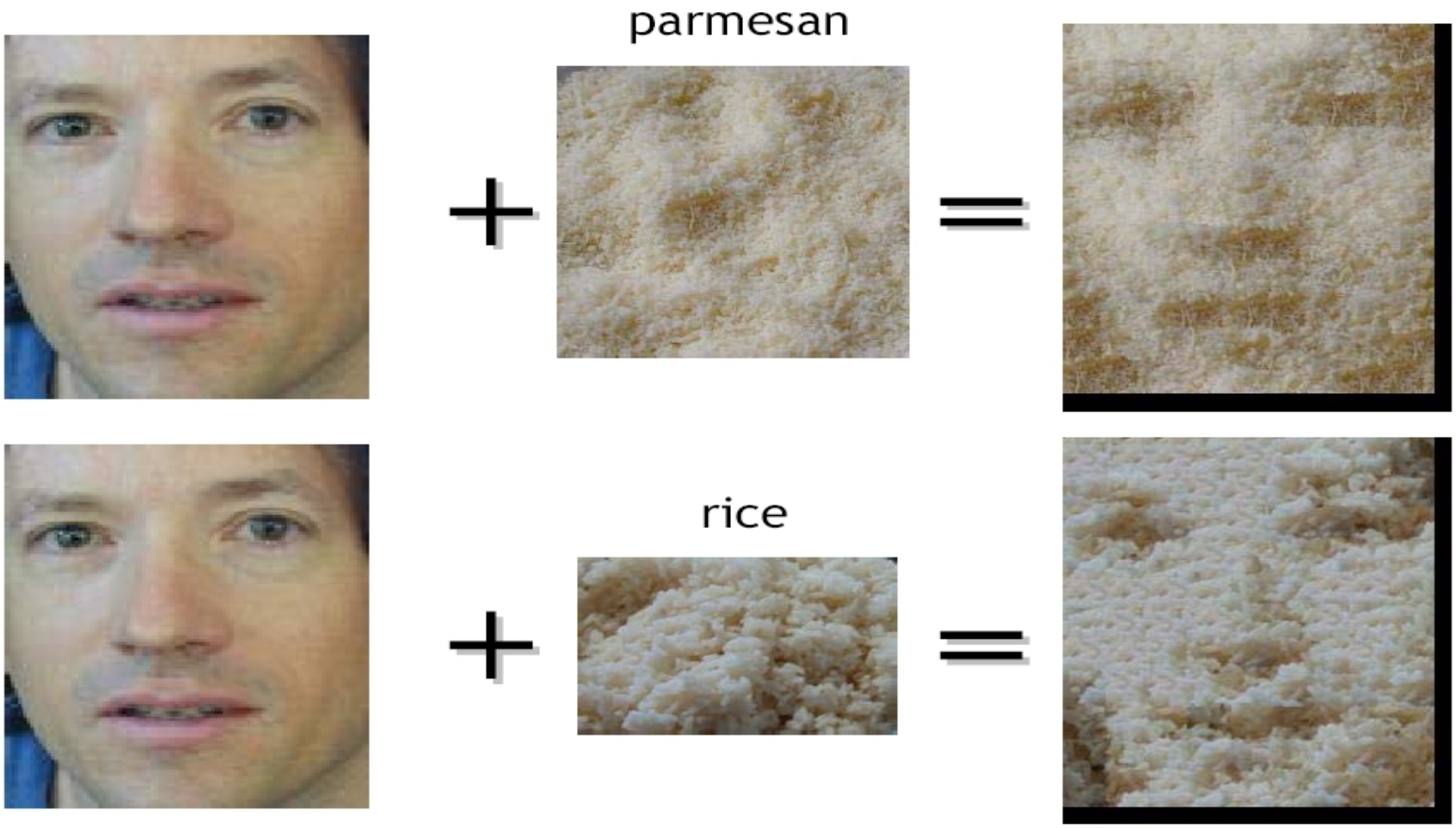
- Take the texture from one object and "paint" it onto another object
  - This requires separating texture and shape
  - That's HARD, but we can cheat
  - Assume we can capture shape by boundary and rough shading
-



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

Adapted from Bill Freeman, MIT

Texture Transfer



Adapted from Bill Freeman, MIT

Texture Transfer

Adapted from Bill Freeman, MIT

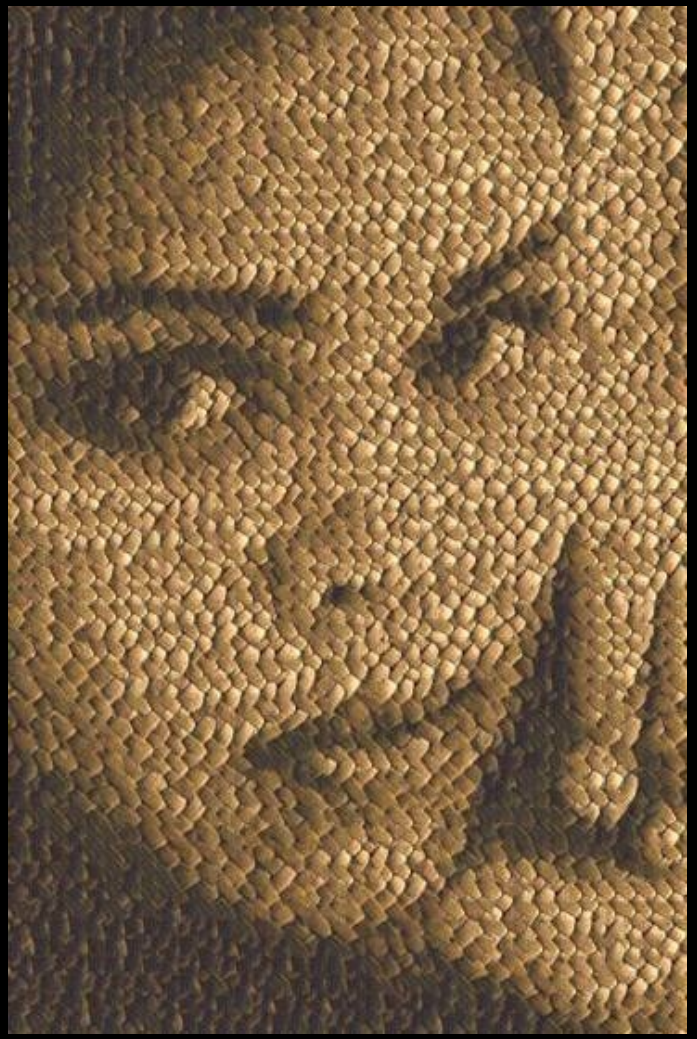
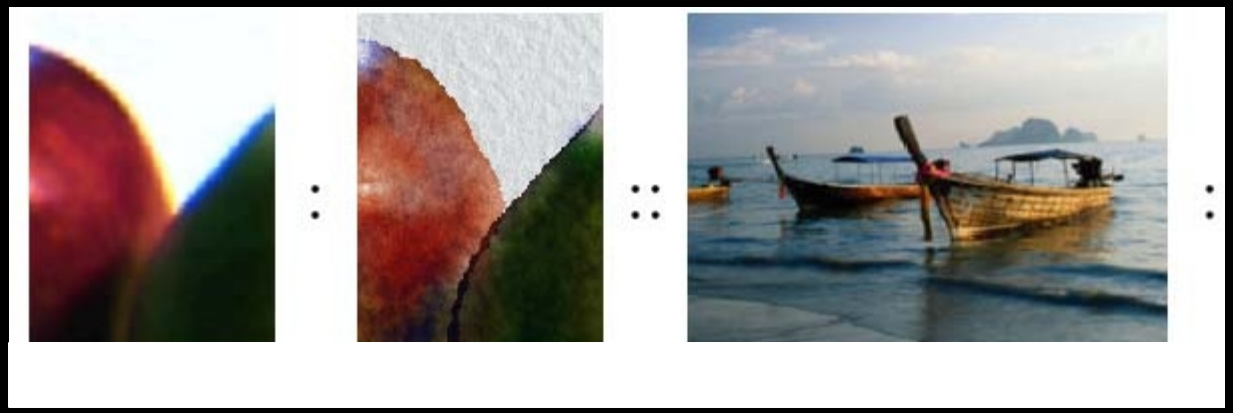


Image Analogies

- Solution:



::

:

B

B’

::

:

B

B’

Learn to Blur

Unfiltered source (A)

Filtered source (A')

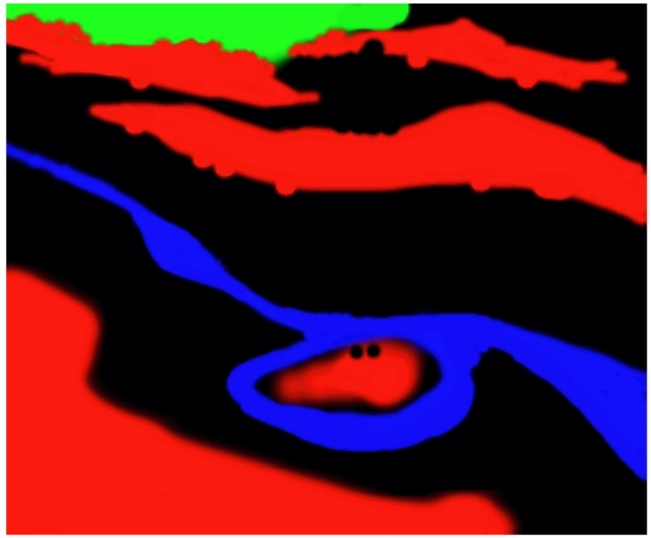
Unfiltered target (B)

Filtered target (B')

Texture by Numbers

Unfiltered source (A)

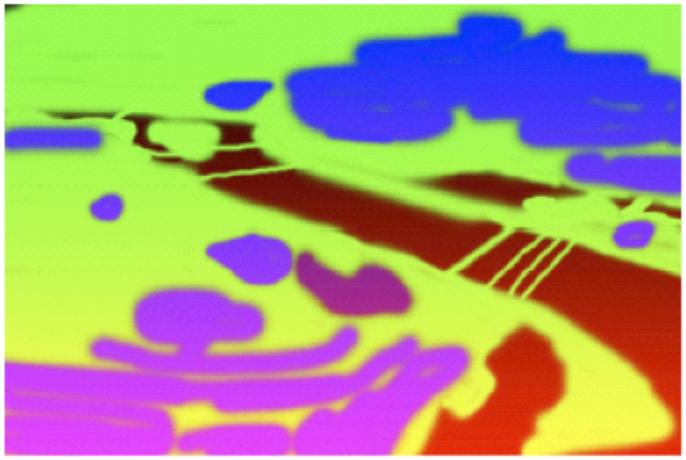
Filtered source (A')



Unfiltered (B)

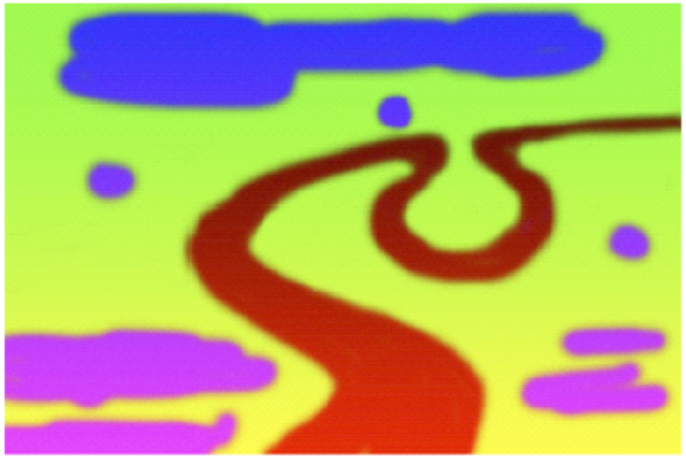
Filtered (B')

Texture Synthesis



Unfiltered source (A)

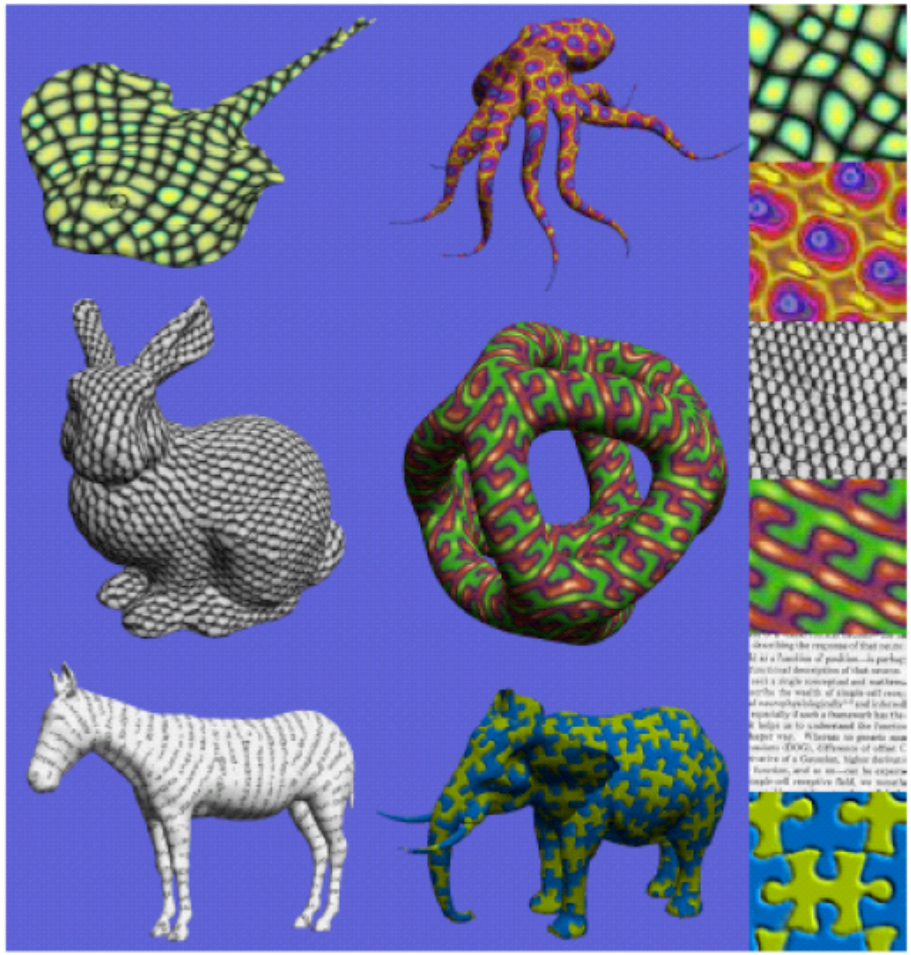
Filtered source (A')



From "Image analogies", Herzmann et al, SIGGRAPH 2001

Adapted from David Forsyth, UC Berkeley

Texture Synthesis



From “Texture synthesis on surfaces”, Turk, SIGGRAPH 2001

Adapted from David Forsyth, UC Berkeley