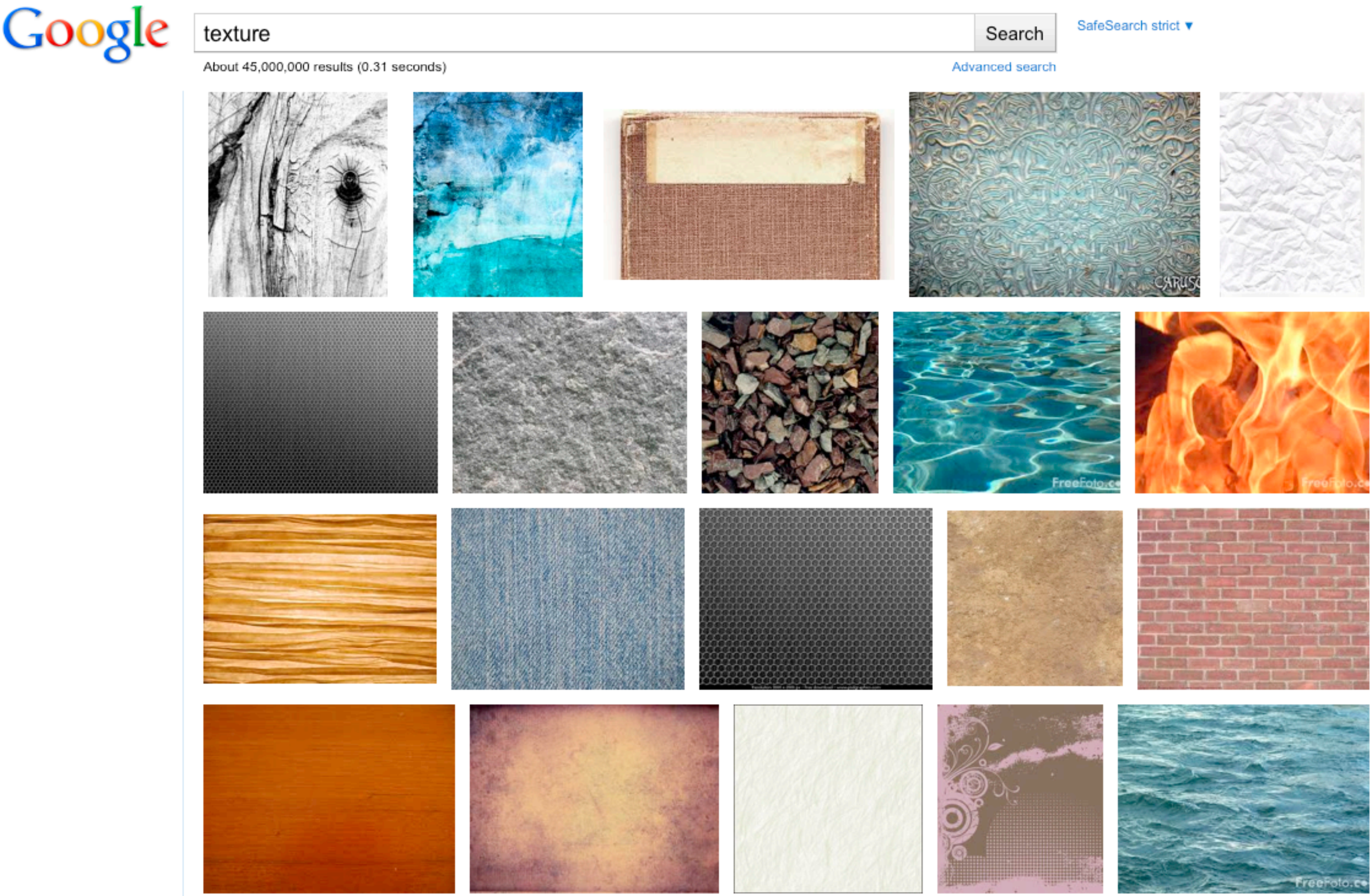
Lecture 16

Textures

What is a texture?


Google
texture
Search
SafeSearch strict ▼
About 45,000,000 results (0.31 seconds)
Advanced search

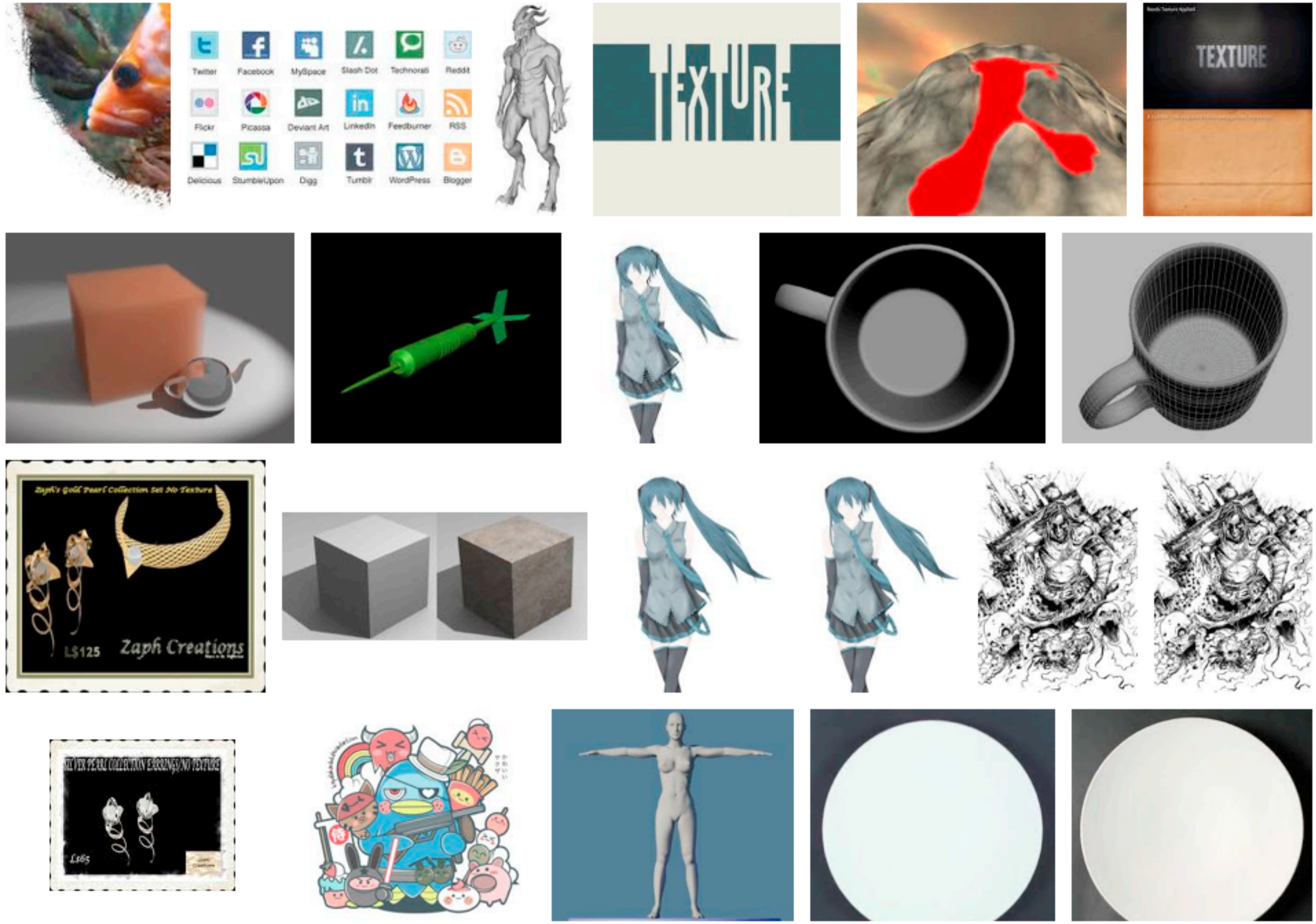
no texture

Search

SafeSearch strict ▼

About 30,800,000 results (0.81 seconds)

Advanced search




GRANITE BAY FARMS
Organic Red Raspberry Vinaigrette
GRANITE BAY FARMS
Organic Spicy Mango Salad Dressing


La Bernardine
La Bernardine
La Bernardine


Bandana's
SPICY
BAR-B-Q SAUCE
Bandana's
SWEET & SMOKY
BAR-B-Q SAUCE
Bandana's
HOT
BAR-B-Q SAUCE
Bandana's
Original
BAR-B-Q SAUCE


TASTY WINES
CRISPY
TASTY WINES
TOASTY
TASTY WINES
VELVETY
TASTY WINES
SPICY
TASTY WINES
BODY

Which textures are we going to talk about in this lecture?

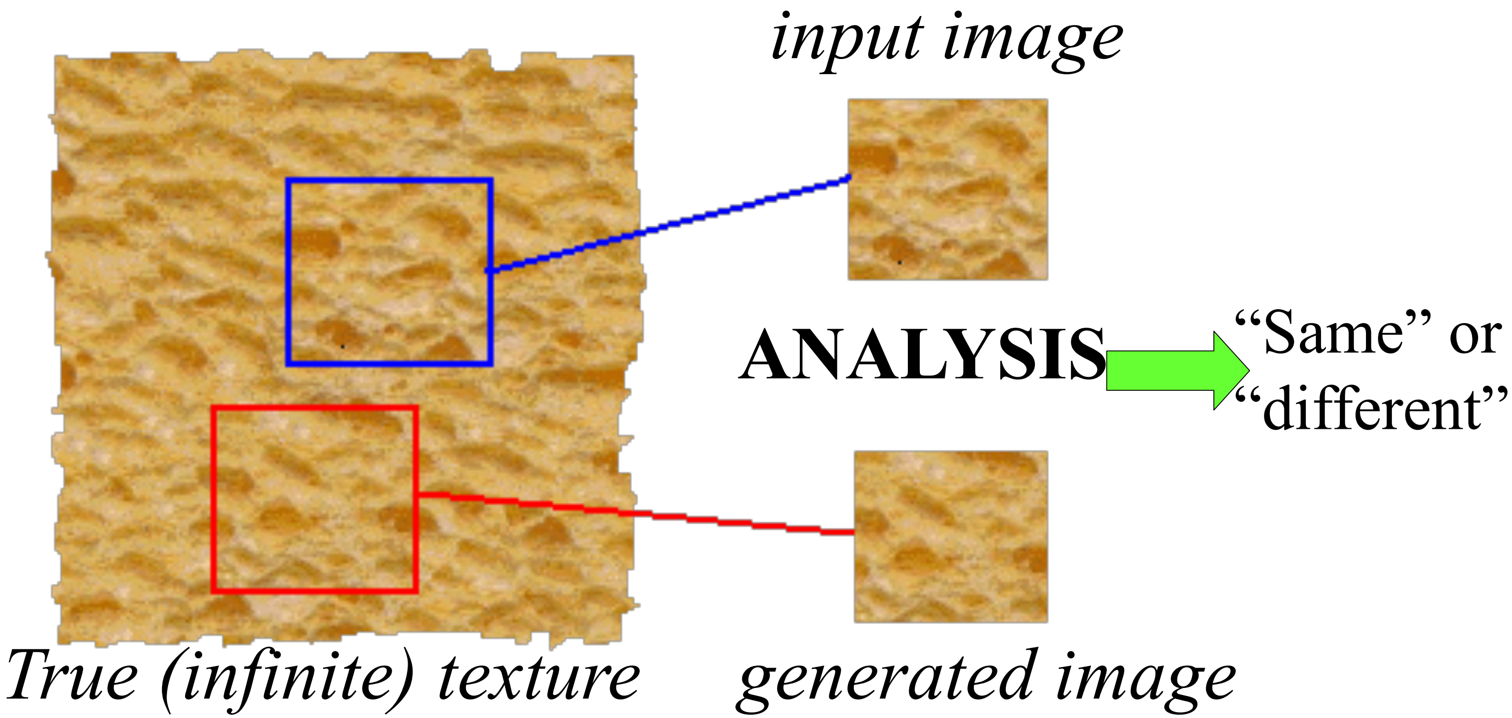
Stationary

Stochastic

When are two textures similar?

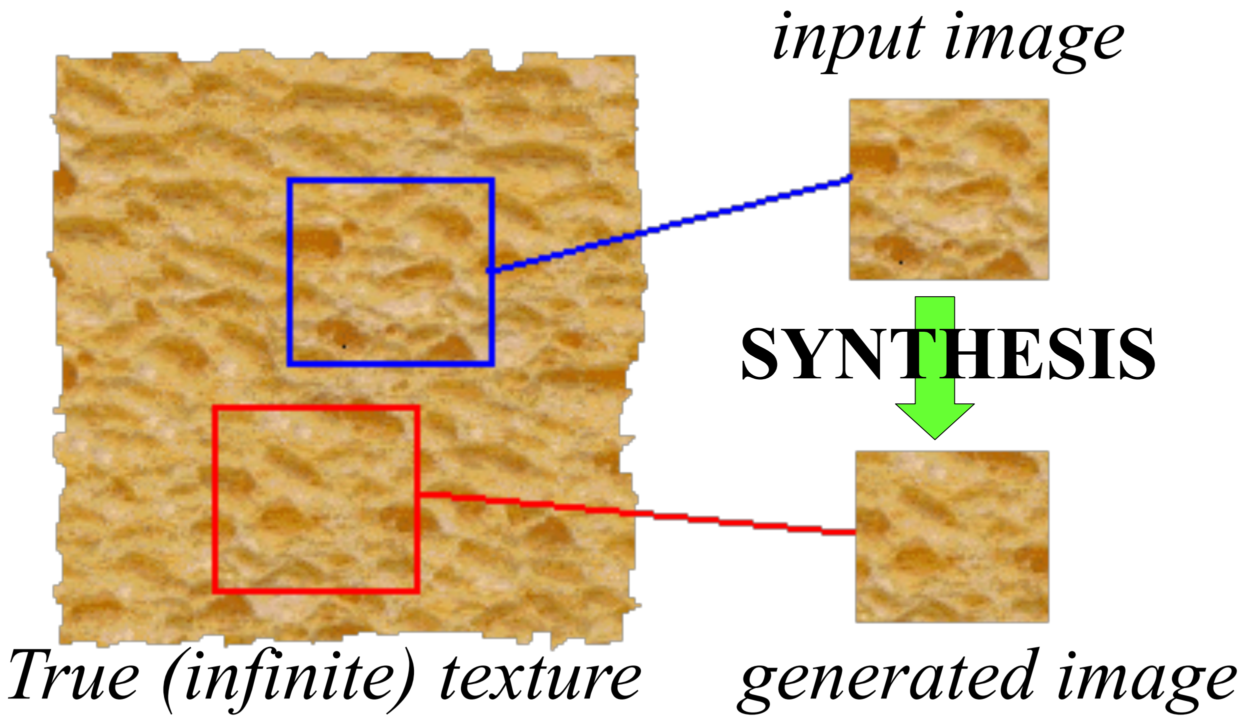
All these images are different instances of the same texture
We can differentiate between them, but they seem generated by the same process

Texture Analysis



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

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“

Let’s get a feeling of the mechanisms for texture perception

What is special about texture perception?

- Pre-attentive texture discrimination
- Perception of sets and summary statistics
- Crowding

REVIEW ARTICLES

Textons, the elements of texture perception, and their interactions

Bela Julesz
Bell Laboratories, Murray Hill, New Jersey 07974, USA

Research with texture pairs having identical second-order statistics has revealed that the pre-attentive texture discrimination system cannot globally process third- and higher-order statistics, and that discrimination is the result of a few local conspicuous features, called textons. It seems that only the first-order statistics of these textons have perceptual significance, and the relative phase between textons cannot be perceived without detailed scrutiny by focal attention.

Bela Julesz, "Textons, the Elements of Texture Perception, and their Interactions". Nature 290: 91-97. March, 1981.

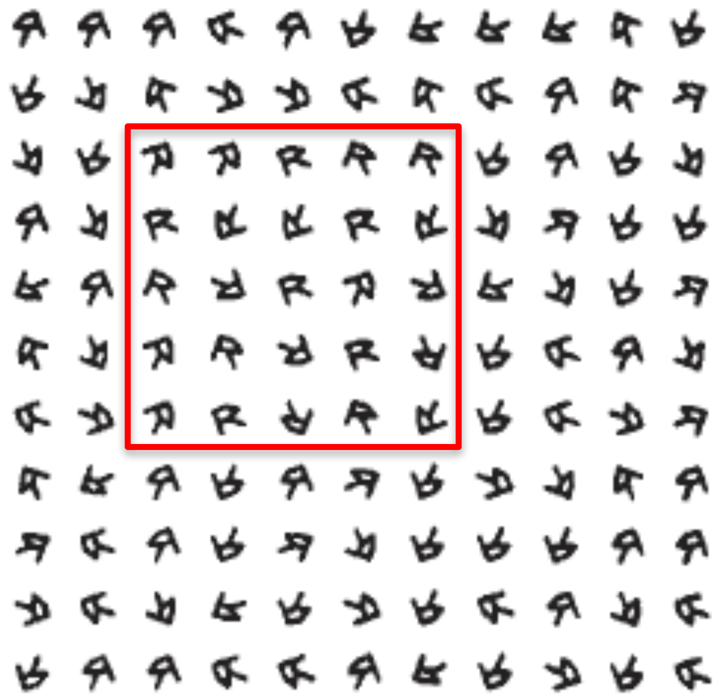
Pre-attentive texture discrimination

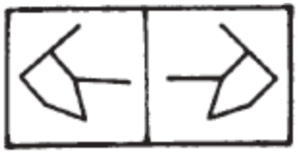
Bela Julesz, "Textons, the Elements of Texture Perception, and their Interactions". Nature 290: 91-97. March, 1981.

Pre-attentive texture discrimination

Bela Julesz, "Textons, the Elements of Texture Perception, and their Interactions". Nature 290: 91-97. March, 1981.

Pre-attentive texture discrimination





This texture pair is pre-attentively indistinguishable. Why?

Bela Julesz, "Textons, the Elements of Texture Perception, and their Interactions". Nature 290: 91-97. March, 1981.

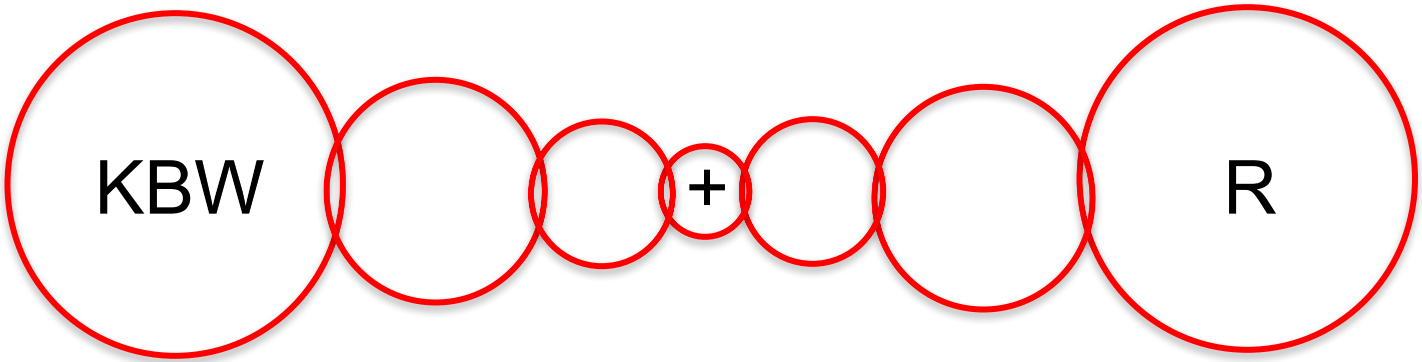
PERSPECTIVE

nature neuroscience

The uncrowded window of object recognition

Denis G Pelli & Katharine A Tillman

Crowding

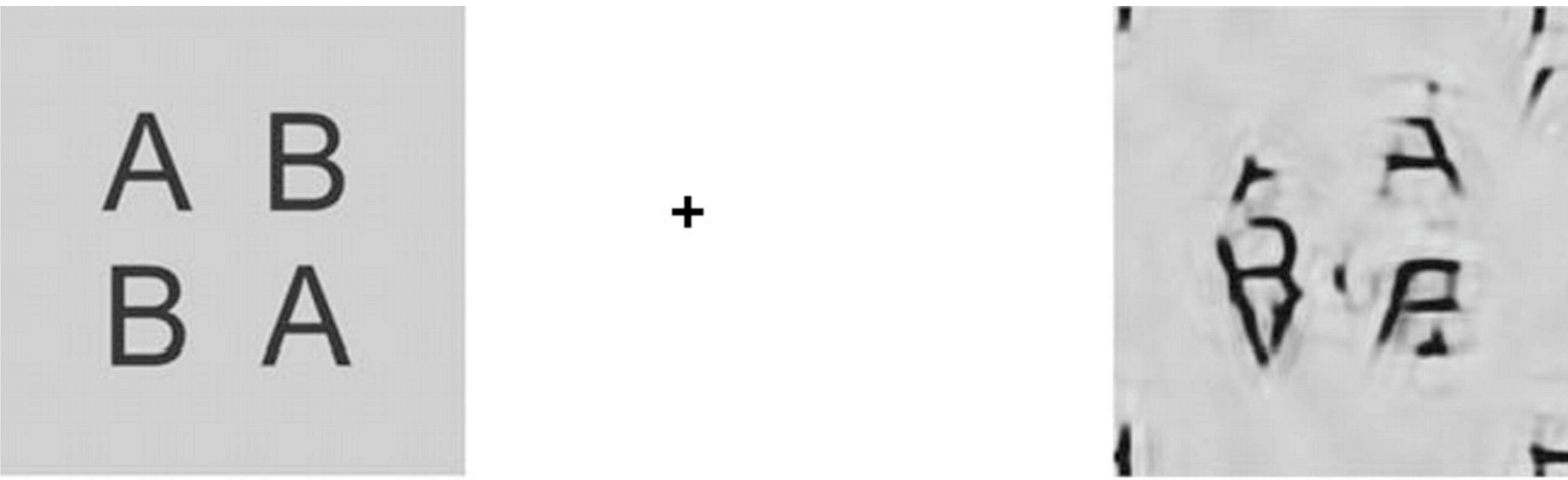


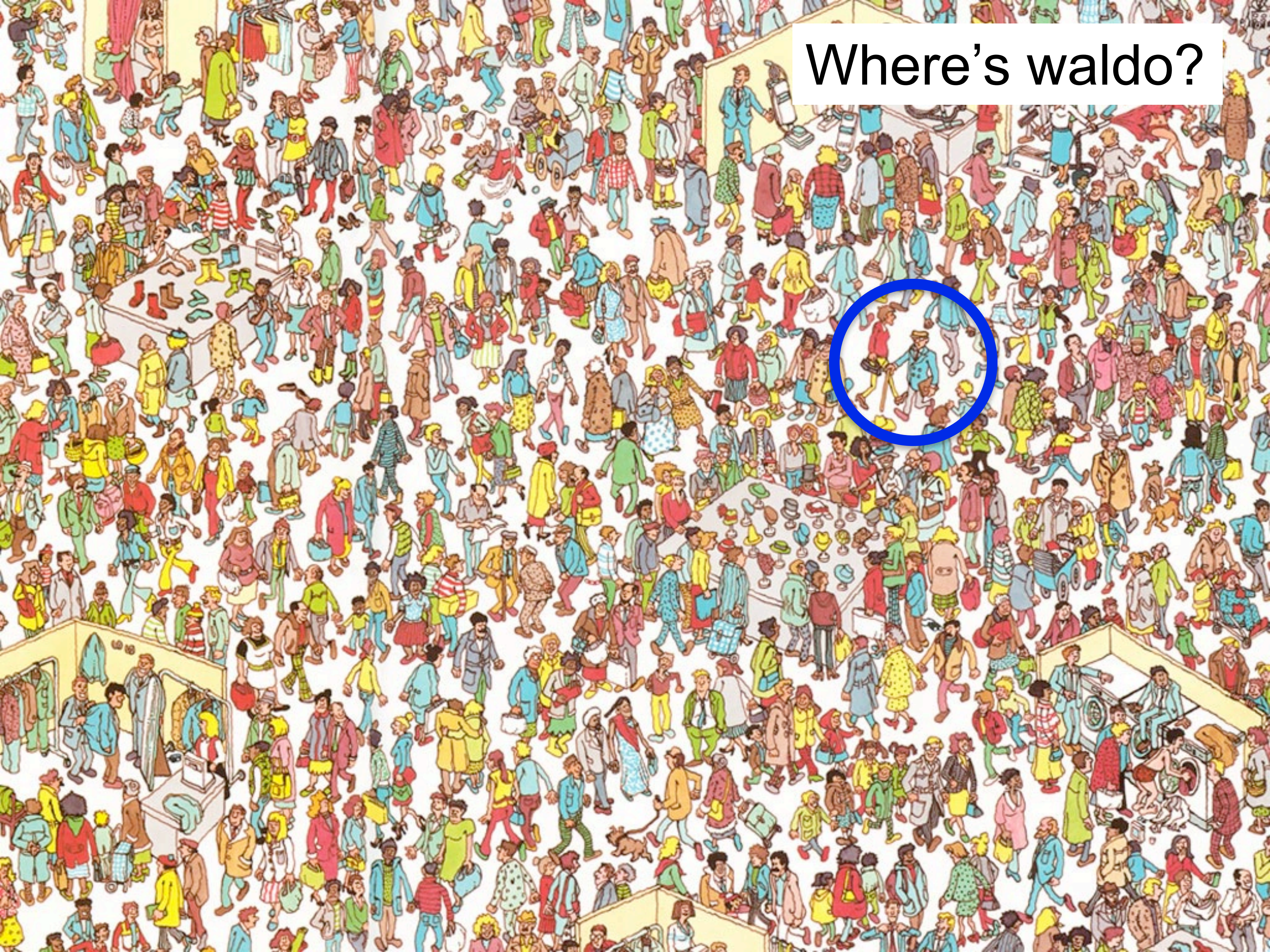
Pelli, D. G., Cavanagh, P., Desimone, R., Tjan, B., & Treisman, A. (2007). Crowding: Including illusory conjunctions, surround suppression, and attention. Journal of Vision, 7(2):i, 1, http://journalofvision.org/7/2/i/

A summary-statistic representation in peripheral vision explains visual crowding

Benjamin Balas [1],
Lisa Nakano [2] and
Ruth Rosenholtz [3]

Journal of Vision
November 19, 2009 vol. 9 no. 12



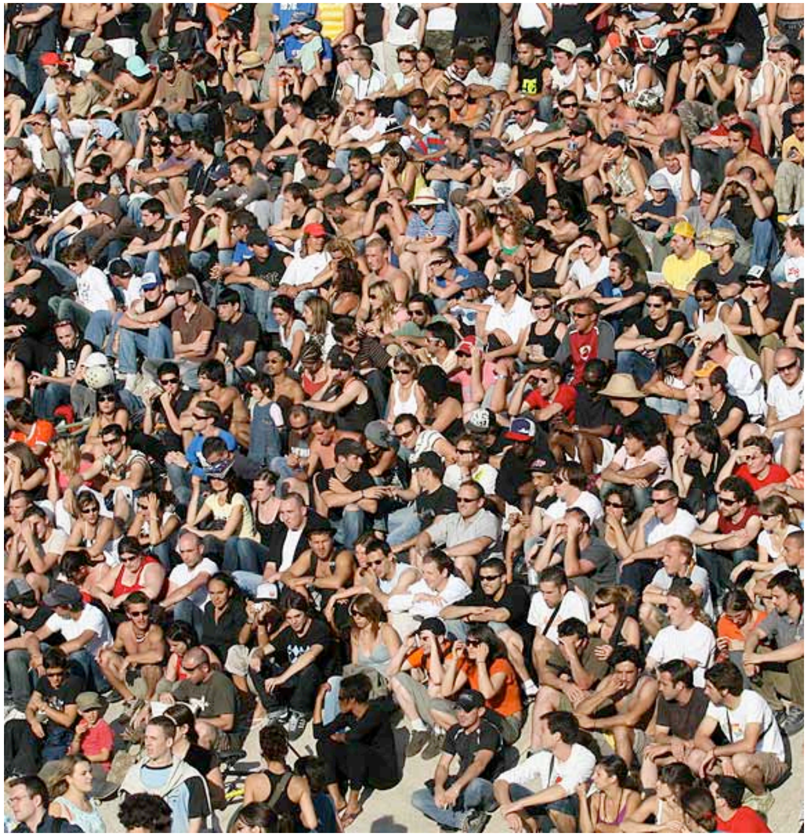

Where’s waldo?

Research Article

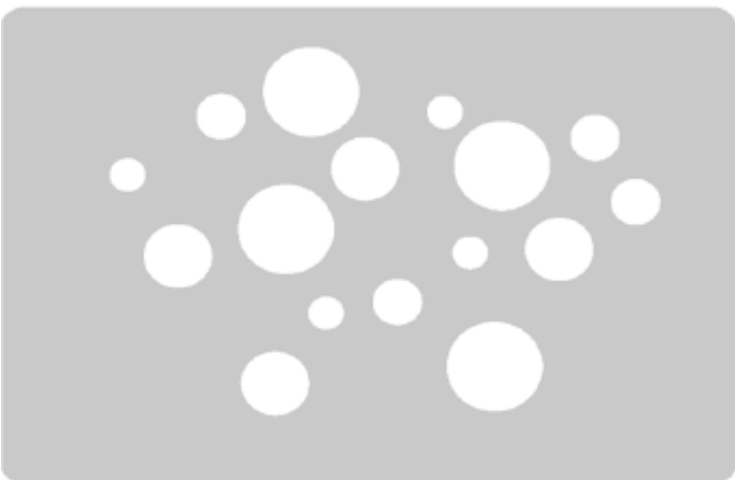
SEEING SETS: Representation by Statistical Properties

Dan Ariely
Massachusetts Institute of Technology

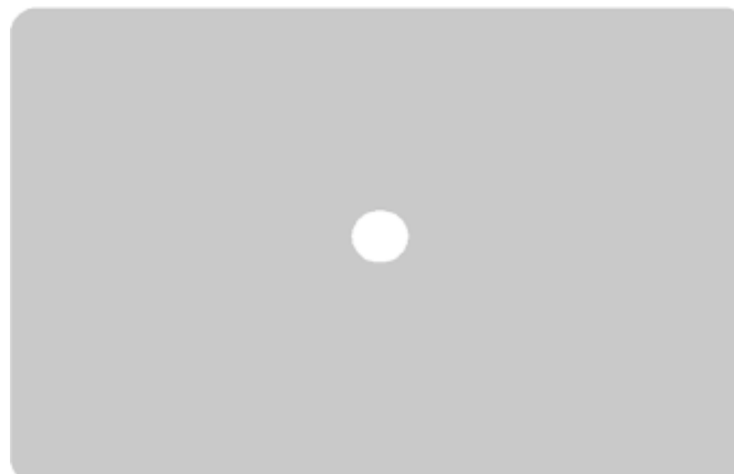
Representation of sets



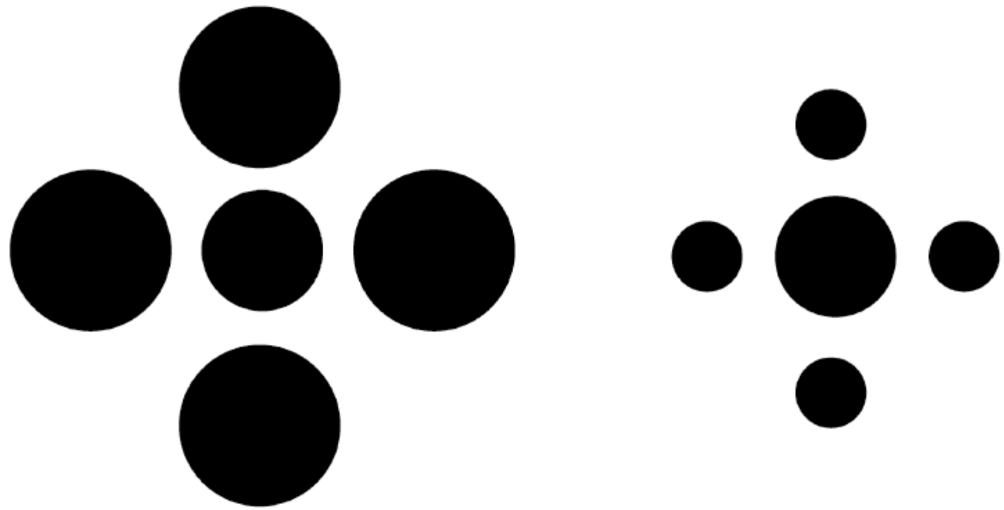
Set



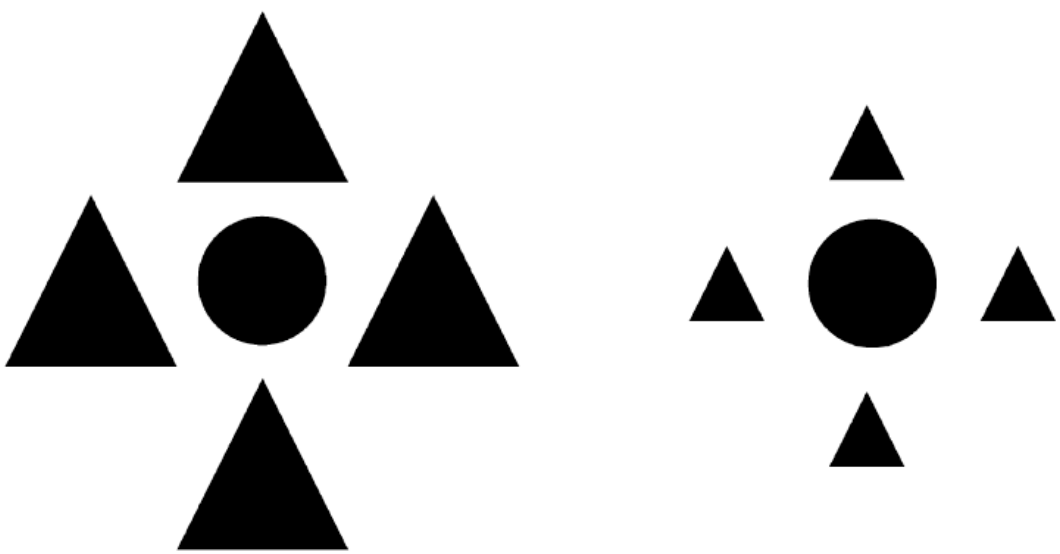
Is this element a member of the set?



Ebbinghaus illusion



The central circle is judged relative to the set properties of the circles surrounding it



Attenuated by reducing the set grouping.

Representation

What a model should account for:

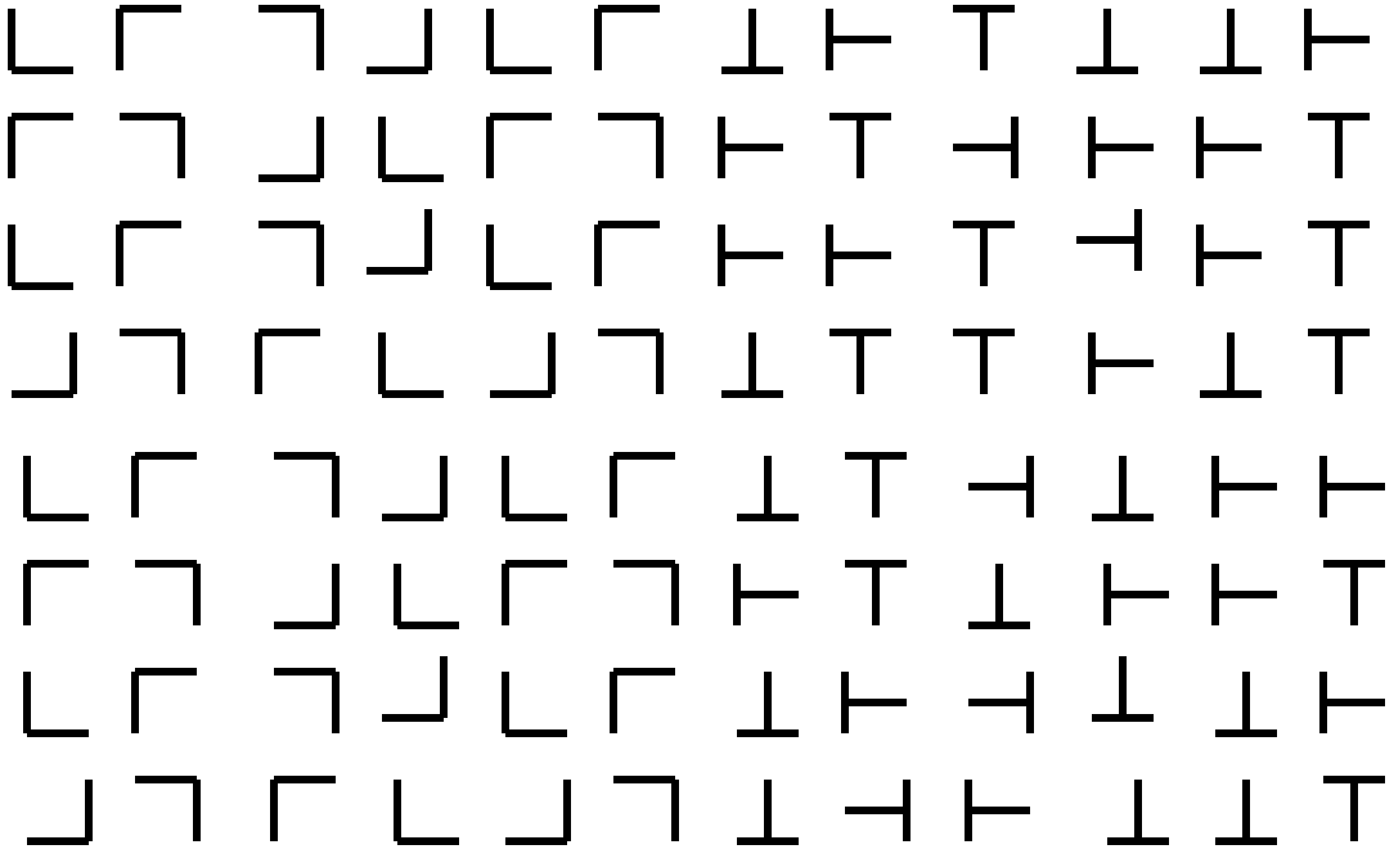
1. **Biological plausibility**: The stages of the model should be motivated by, and be consistent with, known physiological mechanisms of early vision.

2. **Generality**: The model should be general enough that it can be tested on any arbitrary gray-scale image.

3. **Quantitative match with psychophysical data**: The model should make a quantitative **prediction about the salience of the boundary** between any two textured regions. Rank ordering of the discriminability of different texture pairs should agree with that measured psychophysically.

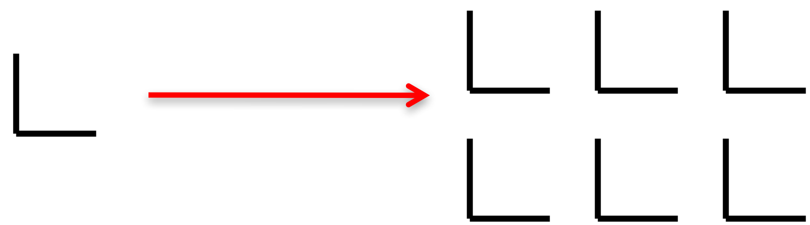
From Malik & Perona, 1990

Julesz - Textons

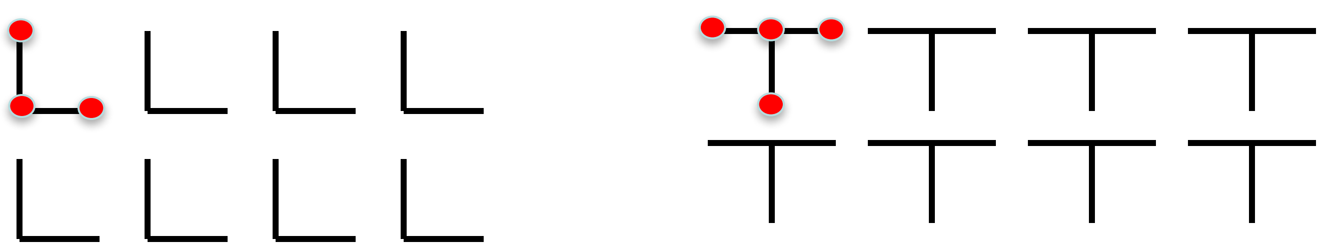


Julesz - Textons

Textons: fundamental texture elements.



Textons might be represented by features such as terminators, corners, and intersections within the patterns…

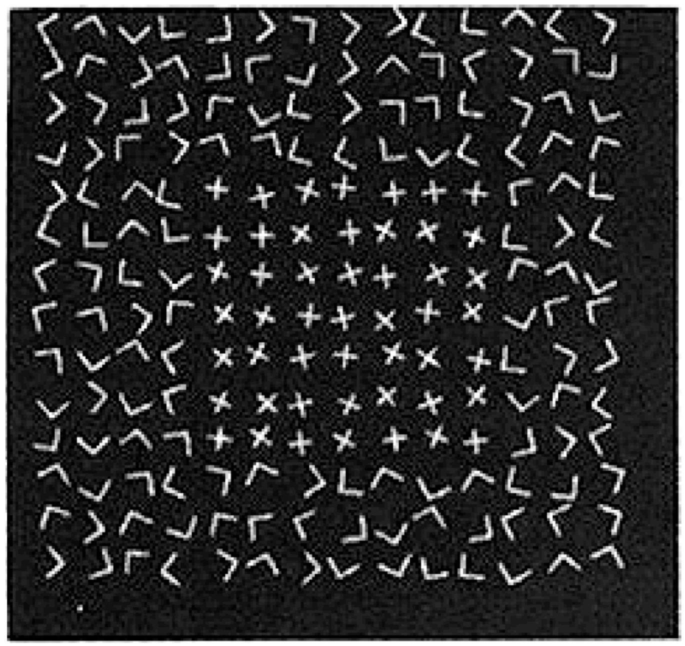


Nature, Vol. 333. No. 6171. pp. 363-364, 26 May 1988

Early vision and texture perception

James R. Bergen* & Edward H. Adelson**

* SRI David Sarnoff Research Center, Princeton,
New Jersey 08540, USA
** Media Lab and Department of Brain and Cognitive Science,



Observation: the Xs look smaller than the Ls.

"We note here that simpler, lower-level mechanisms tuned for size may be sufficient to explain this discrimination."

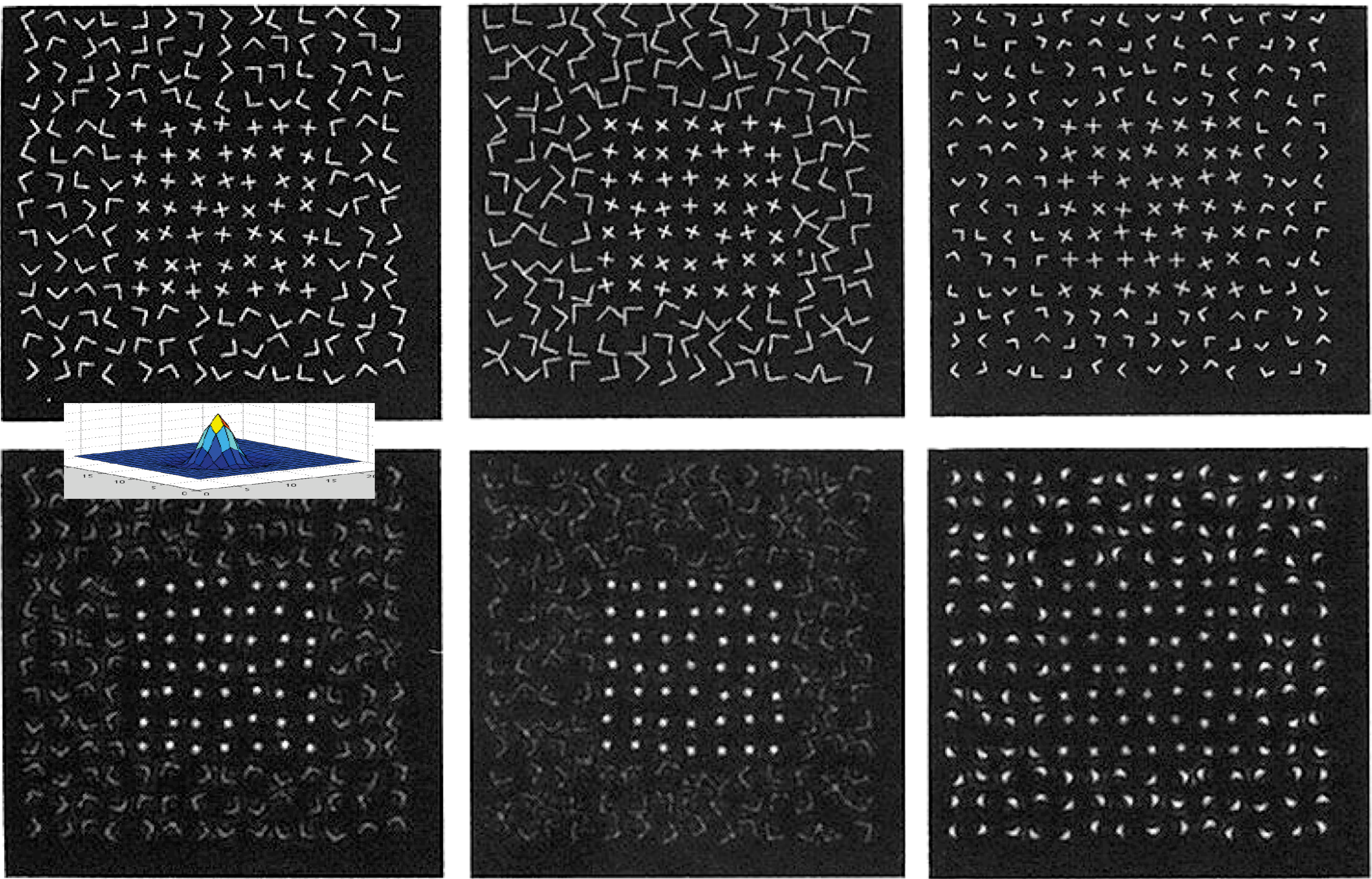
Early vision and texture perception

James R. Bergen* & Edward H. Adelson**

Ls 25% larger

contrast adjusted to keep mean constant

Ls 25% shorter



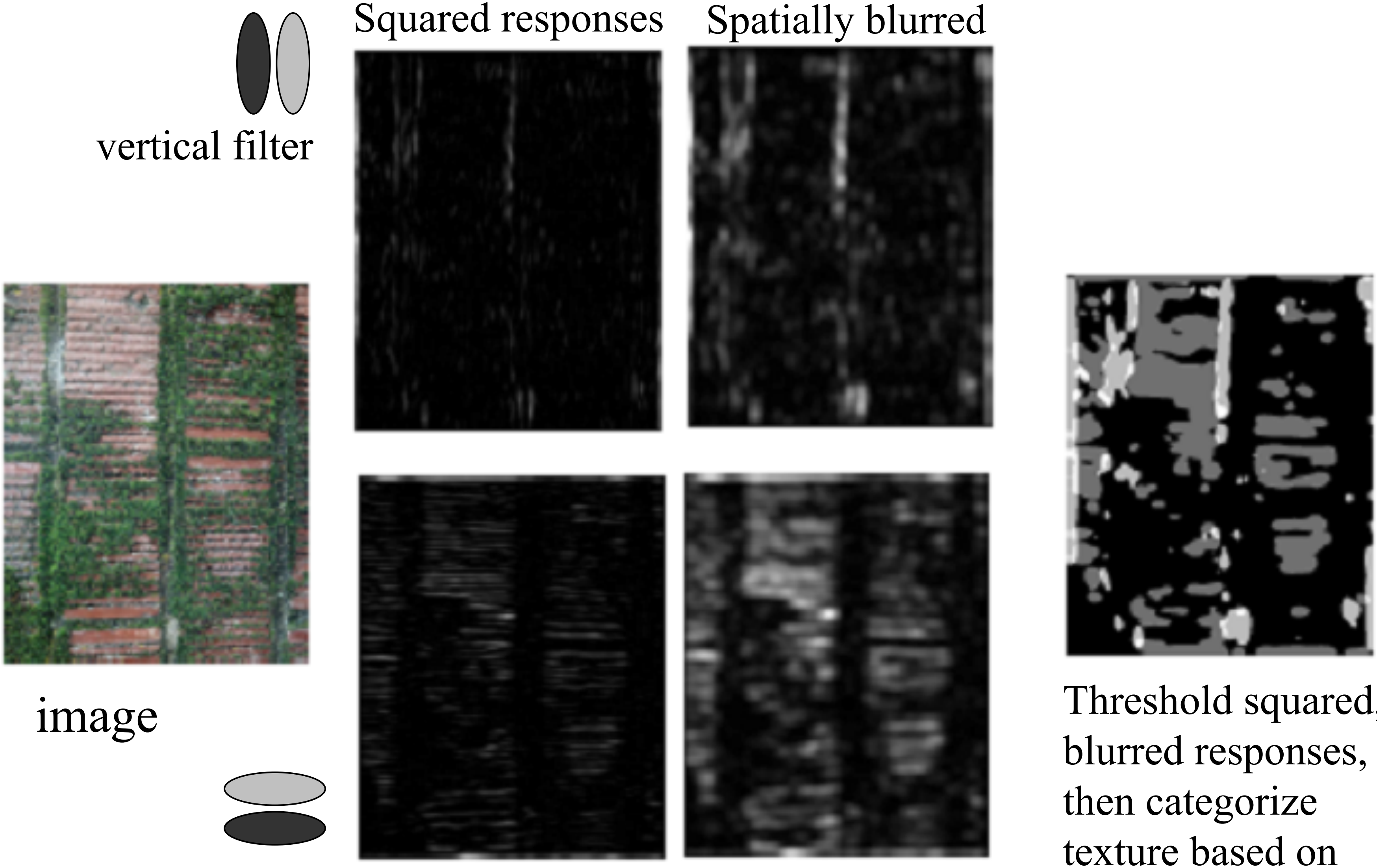
Preattentive texture discrimination with early vision mechanisms

Jitendra Malik and Pietro Perona

Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, California 94720

Received July 7, 1989; accepted December 28, 1989

We present a model of human preattentive texture perception. This model consists of three stages: (1) convolution of the image with a bank of even-symmetric linear filters followed by half-wave rectification to give a set of responses modeling outputs of V1 simple cells, (2) inhibition, localized in space, within and among the neural-response profiles that results in the suppression of weak responses when there are strong responses at the same or nearby locations, and (3) texture-boundary detection by using wide odd-symmetric mechanisms. Our model can predict the salience of texture boundaries in any arbitrary gray-scale image. A computer implementation of this model has been tested on many of the classic stimuli from psychophysical literature. Quantitative predictions of the degree of discriminability of different texture pairs match well with experimental measurements of discriminability in human observers.



vertical filter
Squared responses
Spatially blurred
image
horizontal filter
Threshold squared, blurred responses, then categorize texture based on those two bits

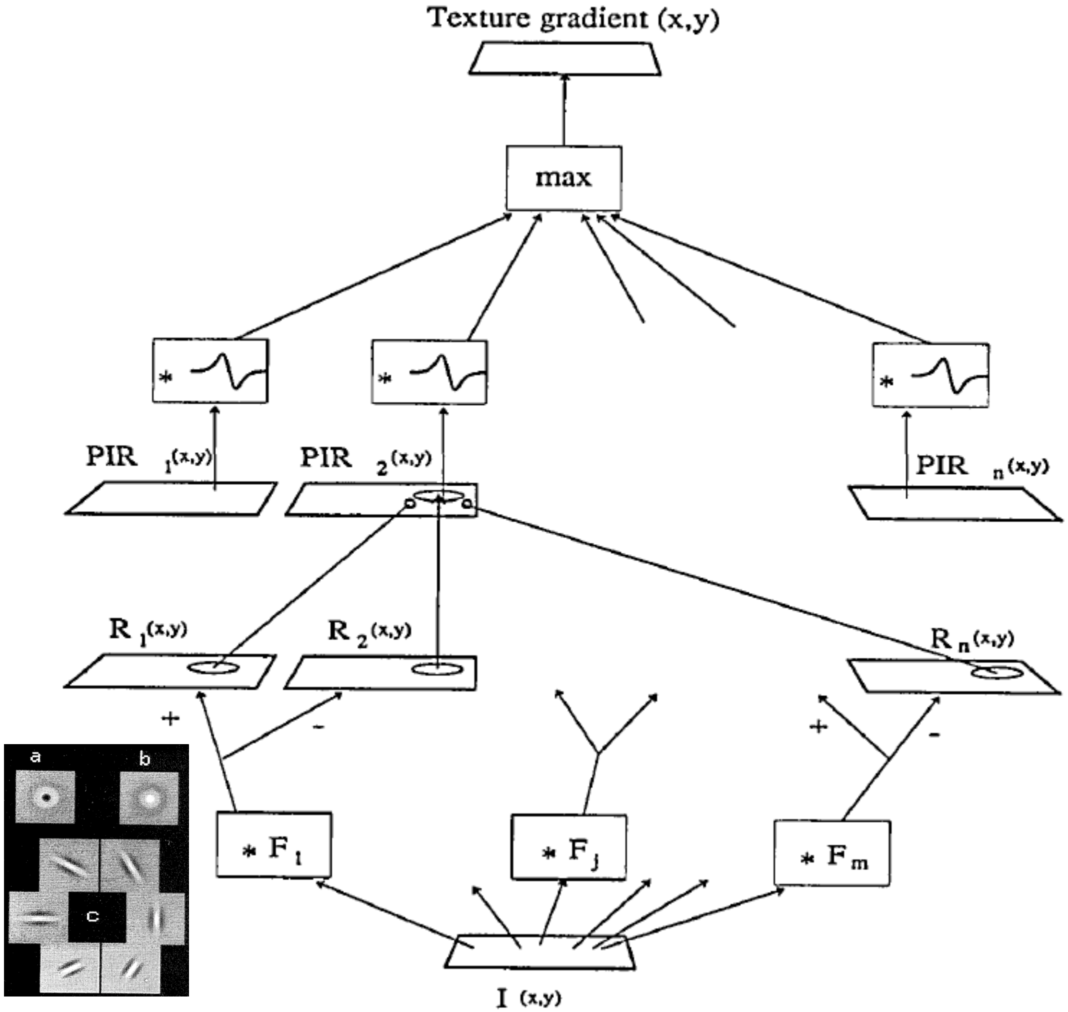
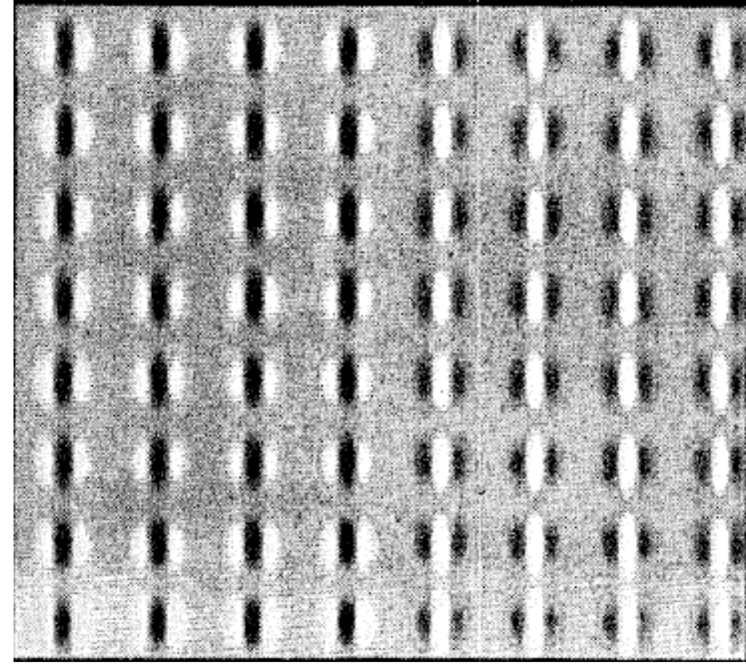
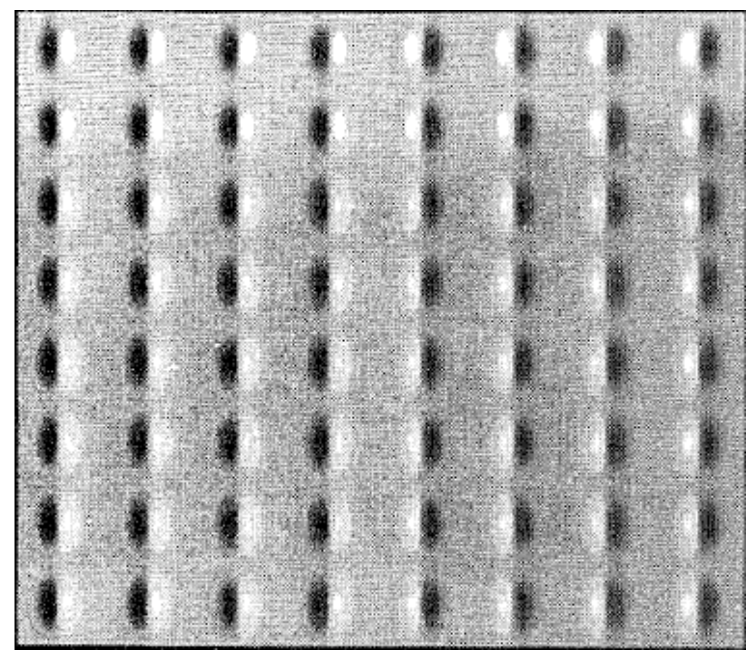


Fig. 1. Simplified schematics of our model for texture perception. The image (bottom) is filtered using the kernels $F_1 \ldots F_m$ and is half-wave rectified to give the set of simple-cell responses $R_1 \ldots R_n$. The postinhibition responses $\mathrm{PIR}_1 \ldots \mathrm{PIR}_n$ are computed by thresholding the R_i and taking the maximum of the result over small neighborhoods. The thresholds depend on the activity of all channels. The texture gradient is computed by taking the maximum of the responses of wide odd-symmetric filters acting on the postinhibition responses PIR_i.



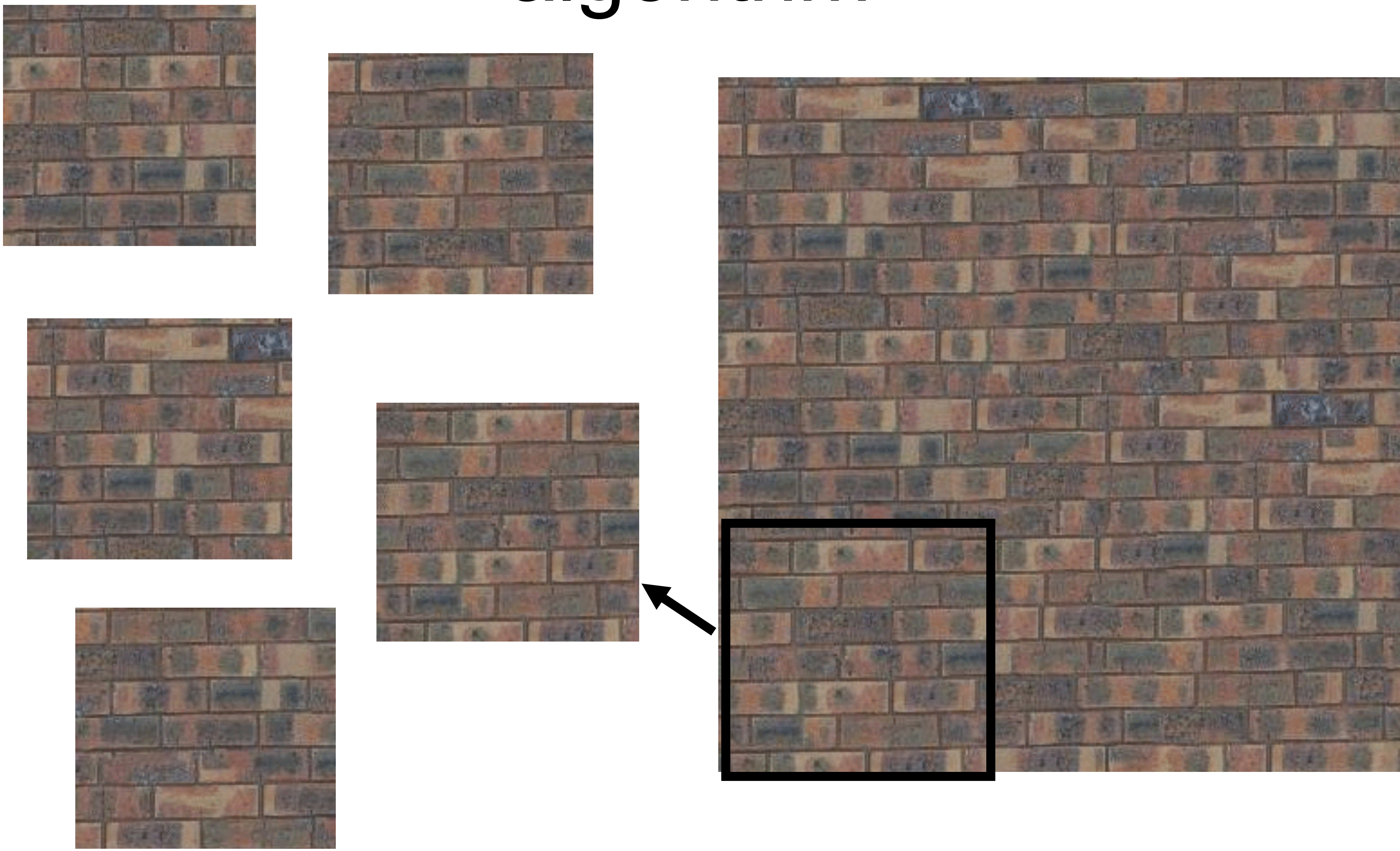


Two big families of models

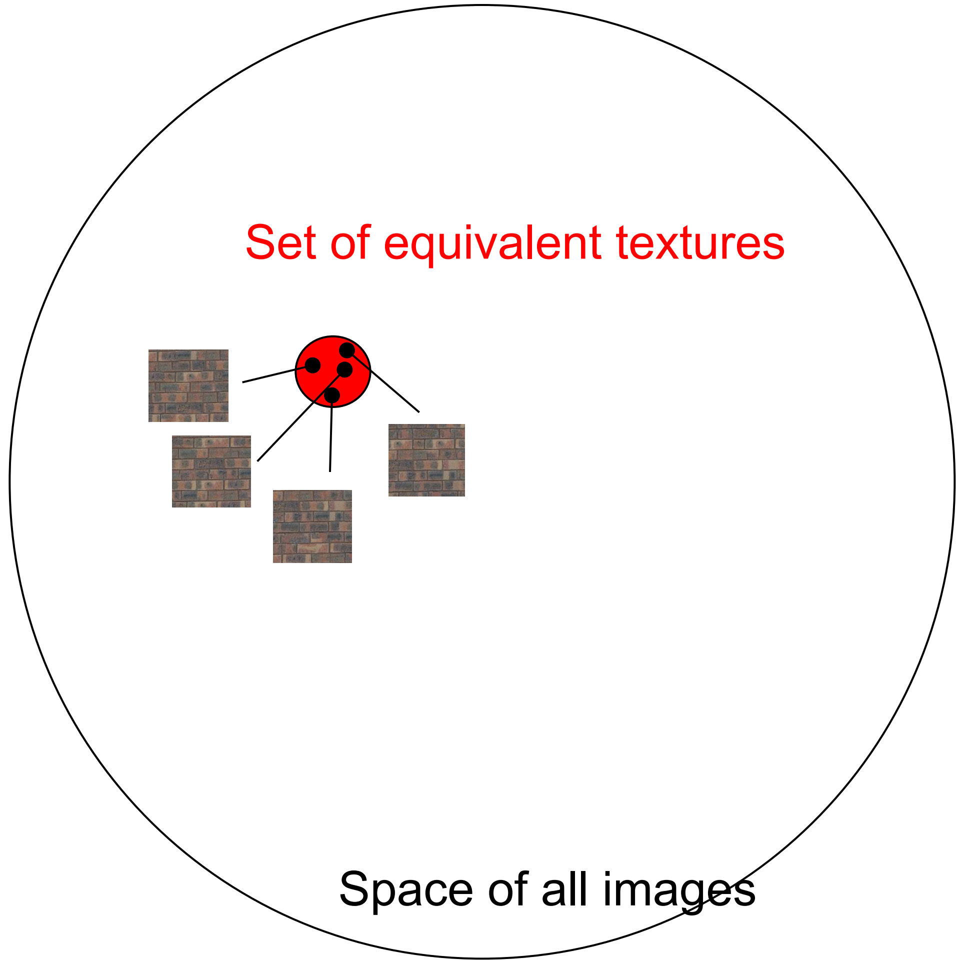
1- Parametric models of filter outputs

2- Example-based non-parametric models

The trivial texture synthesis algorithm

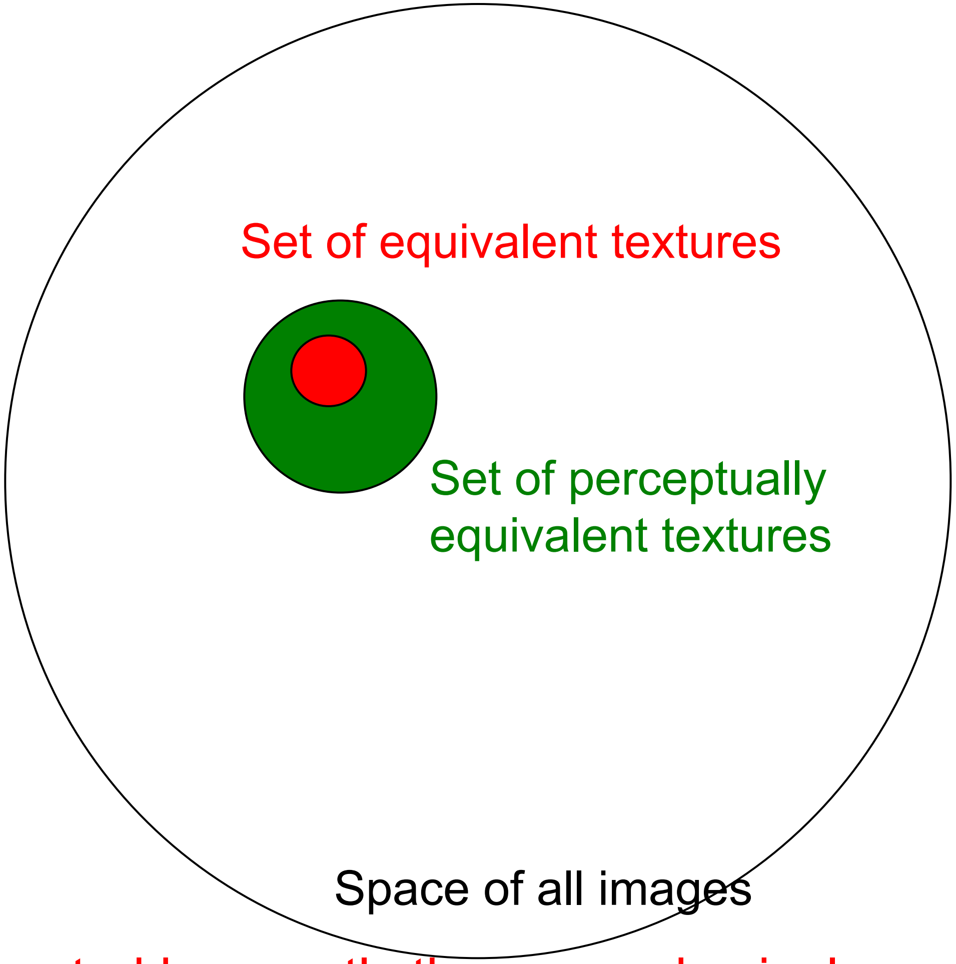


Texture synthesis and representation



Set of equivalent textures: generated by exactly the same physical process

Texture synthesis and representation



Set of equivalent textures: generated by exactly the same physical process

Set of perceptually equivalent textures: “well, they just look the same to me”

If matching the averaged squared filter values is a good way to match a given texture, then maybe matching the entire marginal distribution (eg, the histogram) of a filter's response would be even better.

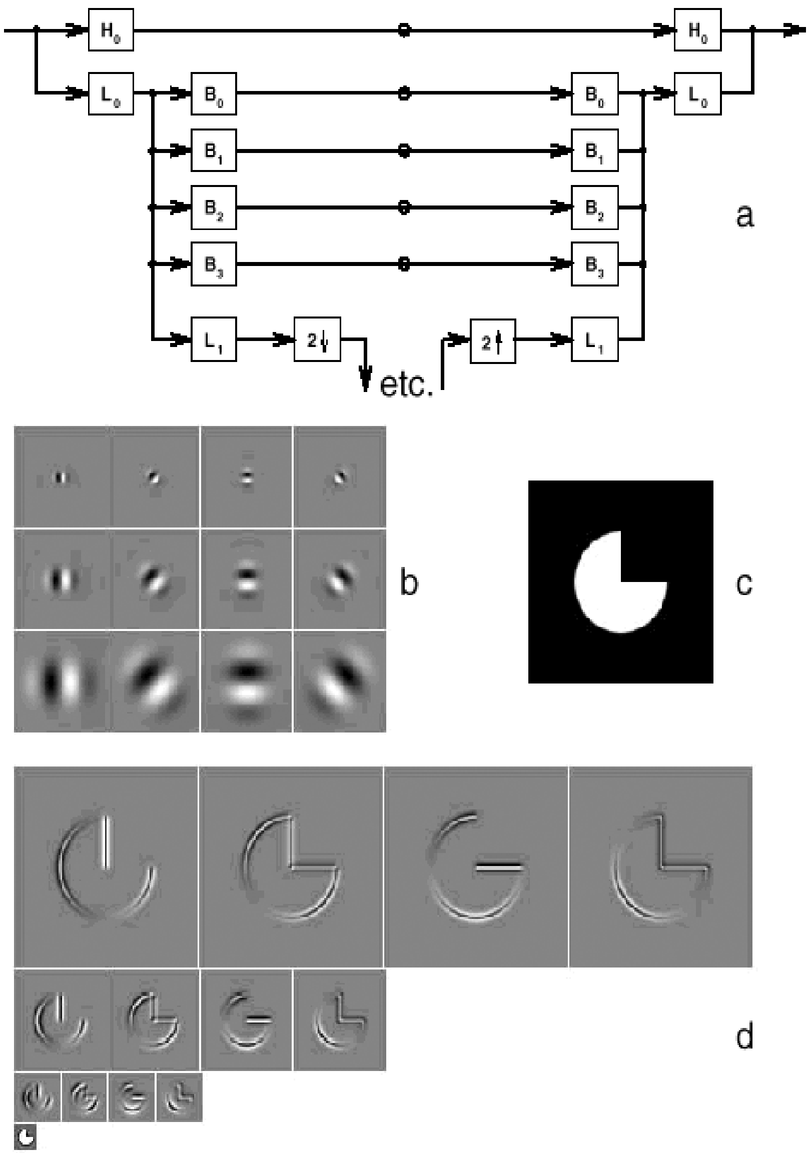
Jim Bergen proposed this…

Pyramid-Based Texture Analysis/Synthesis

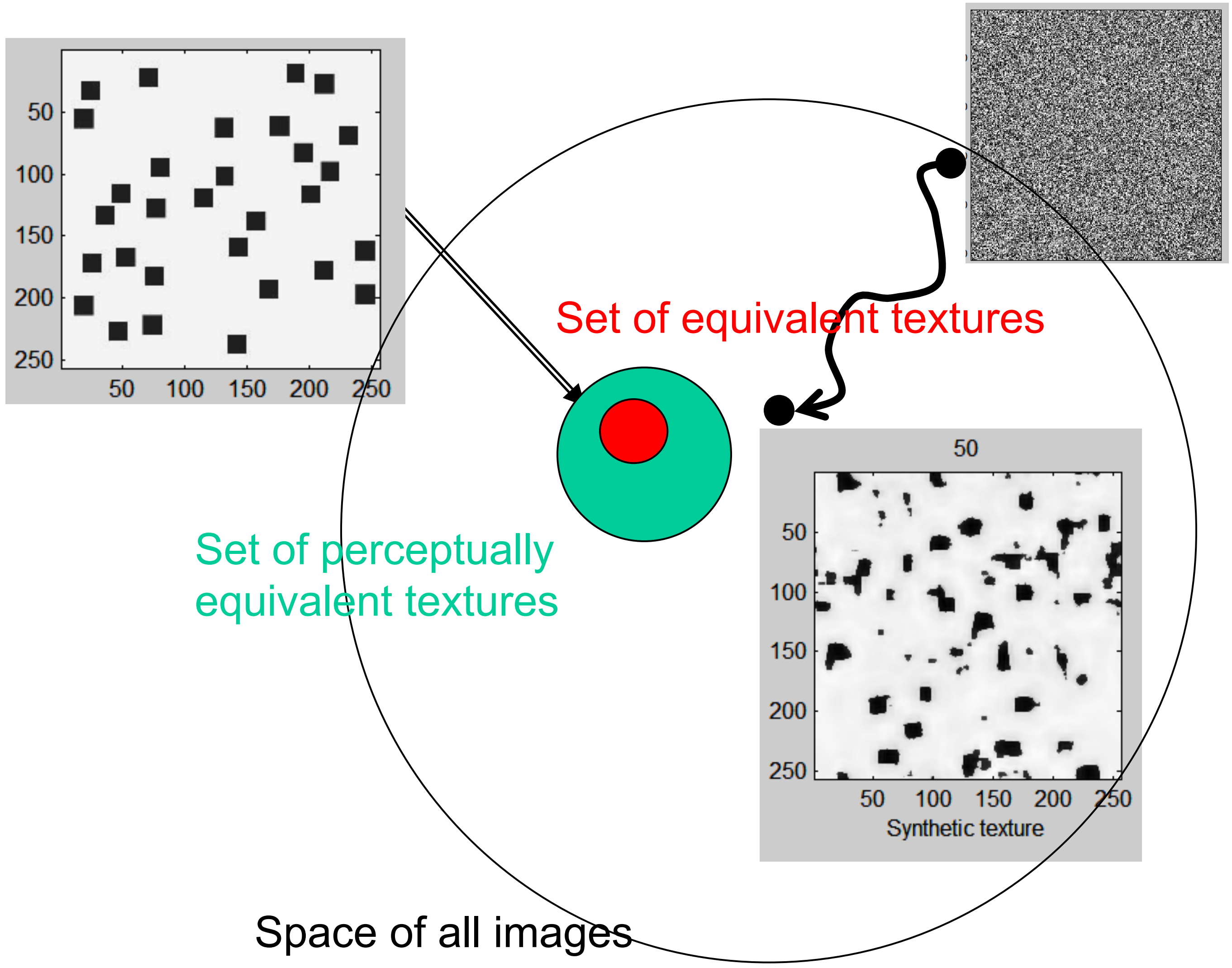
David J. Heeger[‡]
Stanford University

James R. Bergen[†]
SRI David Sarnoff Research Center

SIGGRAPH 1994



The main idea: it works by ‘kind of’ projecting a random image into the set of equivalent textures



Overview of the algorithm

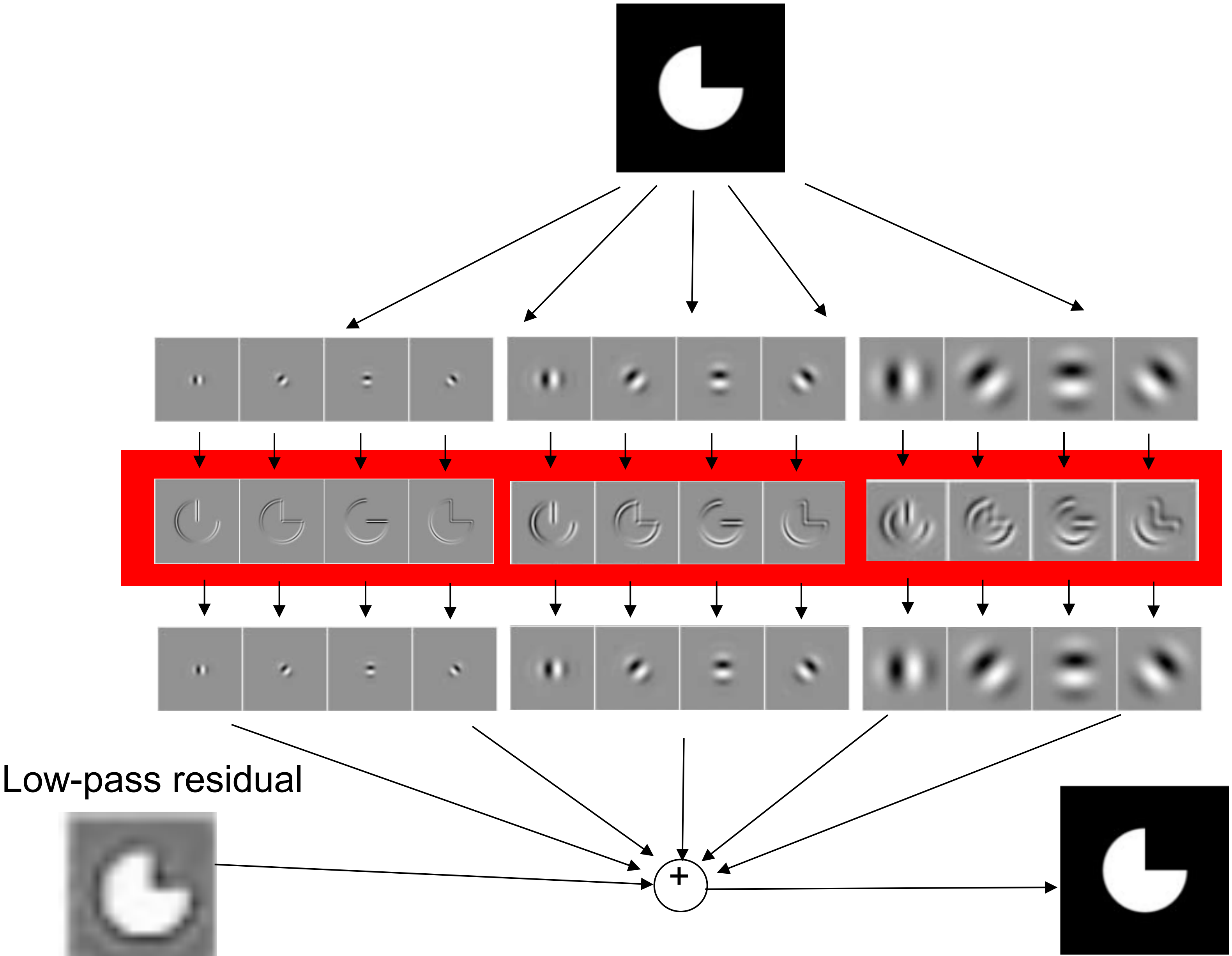
```
Match-texture(noise,texture)
 Match-Histogram (noise,texture)
 analysis-pyr = Make-Pyramid (texture)
 Loop for several iterations do
    synthesis-pyr = Make-Pyramid (noise)
    Loop for a-band in subbands of analysis-pyr
         for s-band in subbands of synthesis-pyr
         do
         Match-Histogram (s-band,a-band)
    noise = Collapse-Pyramid (synthesis-pyr)
    Match-Histogram (noise,texture)
```

Two main tools:

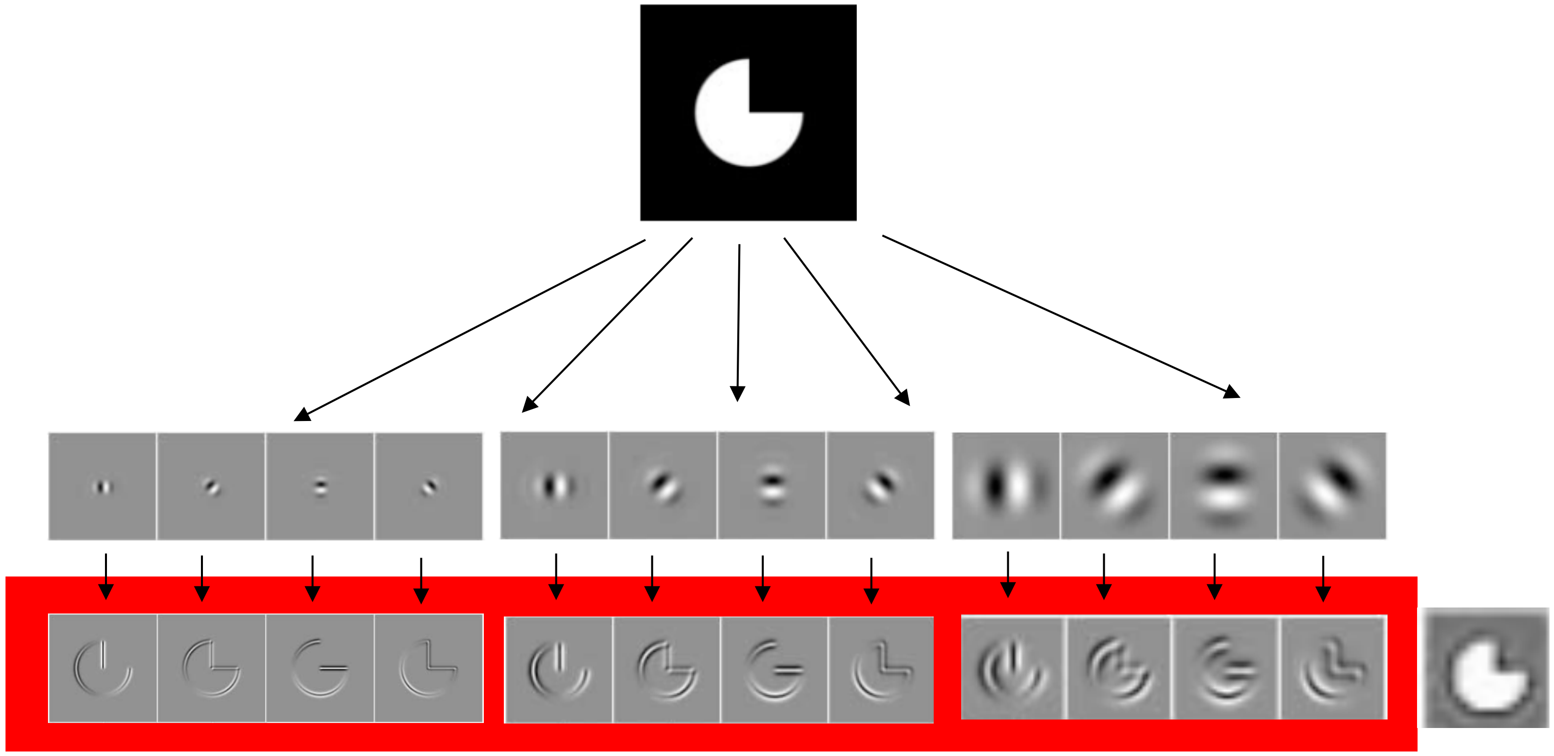
1- steerable pyramid

2- matching histograms

1-The steerable pyramid

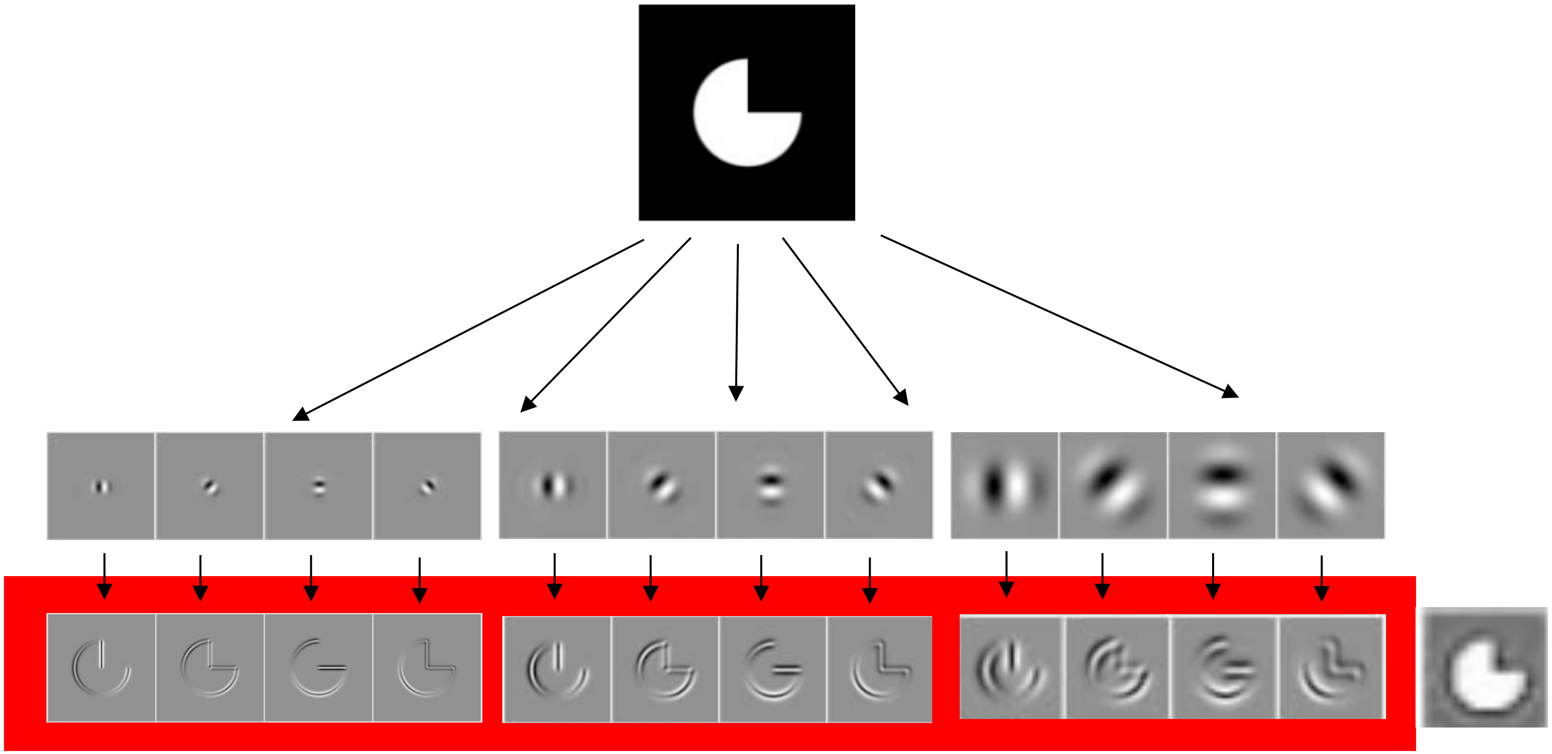


1-The steerable pyramid



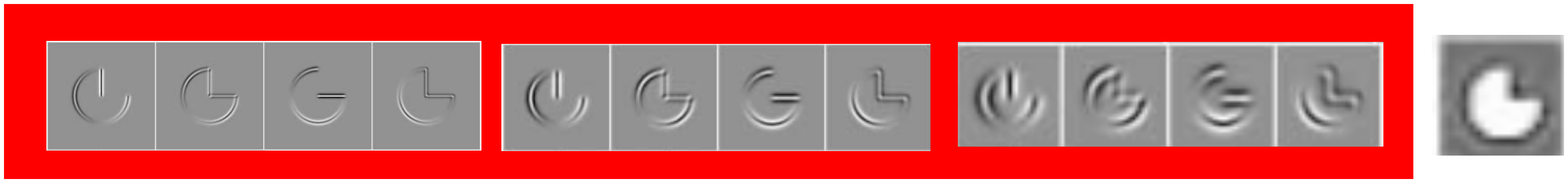
But why do I want to represent images like this?

1-The steerable pyramid



Argument used by H & B: Statistical measures in the subband representation seem to provide a “distance” between textures that correlates with human perception better than pixel-based representations.

1-The steerable pyramid



In general seems a good idea to have a representation that:

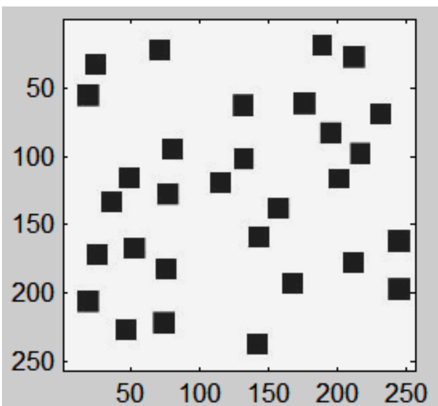
-Preserves all image information (we can go back to the image)

-Provides more independent channels of information than pixel values (we can mess with each band independently)

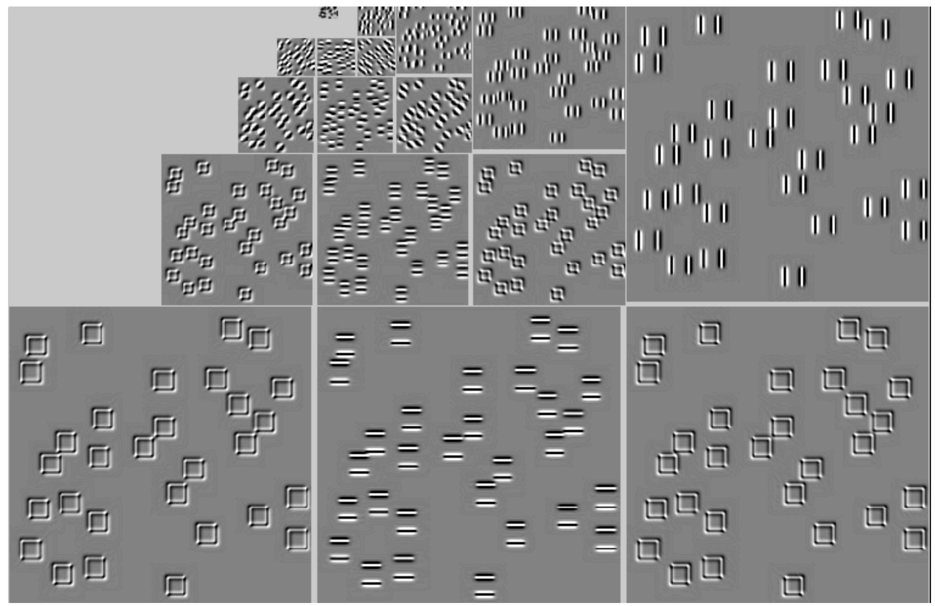
But all this is just indirectly related to the texture synthesis task. But let assume is good enough…

1-The steerable pyramid

Input texture



Steerable pyr



Overview of the algorithm

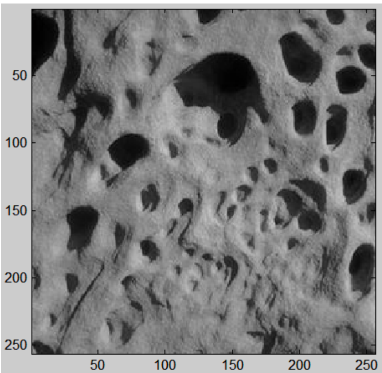
```
Match-texture(noise,texture)
  Match-Histogram (noise,texture)
  analysis-pyr = Make-Pyramid (texture)
  Loop for several iterations do
     synthesis-pyr = Make-Pyramid (noise)
     Loop for a-band in subbands of analysis-pyr
          for s-band in subbands of synthesis-pyr
          do
          Match-Histogram (s-band,a-band)
     noise = Collapse-Pyramid (synthesis-pyr)
     Match-Histogram (noise,texture)
```

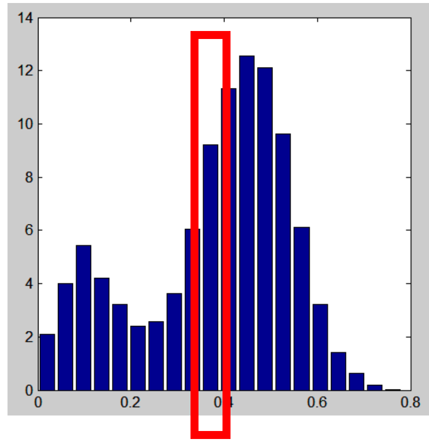
Two main tools:

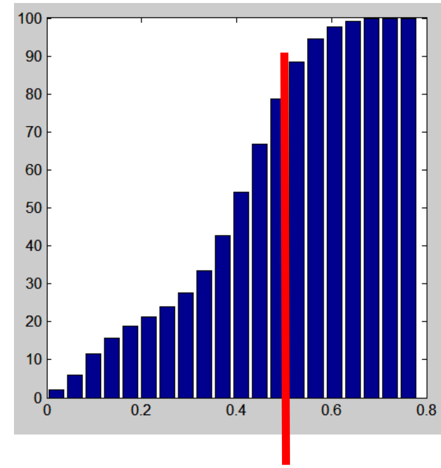
1- steerable pyramid

2- matching histograms

2-Matching histograms

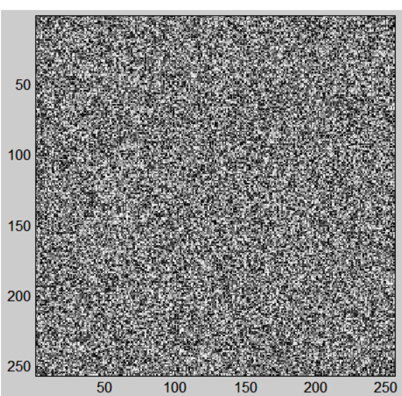


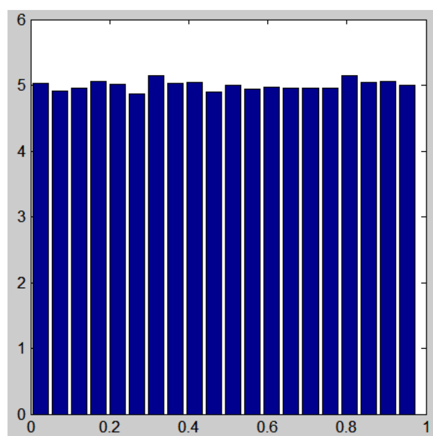


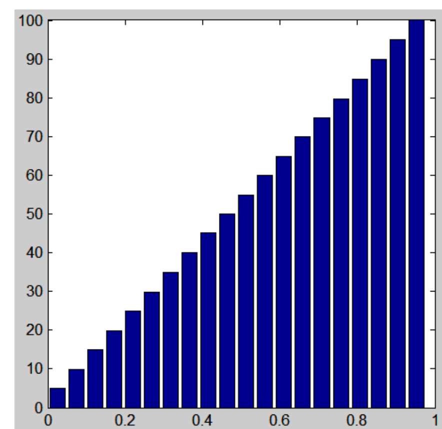


9% of pixels have an intensity value within the range[0.37, 0.41]

75% of pixels have an intensity val smaller than 0.5



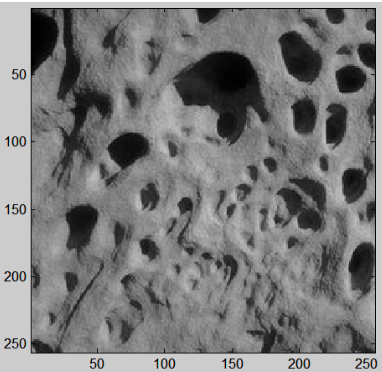


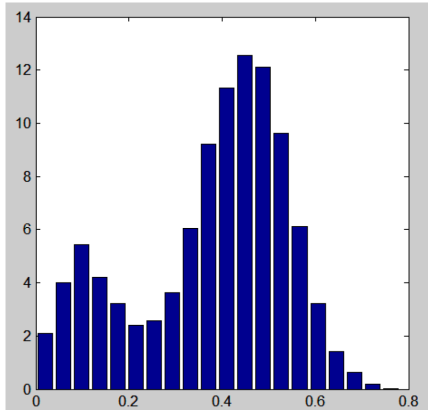


5% of pixels have an intensity value within the range[0.37, 0.41]

2-Matching histograms

Z(x,y)

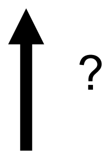




We look for a transformation of the image Y

Y’ = f (Y)

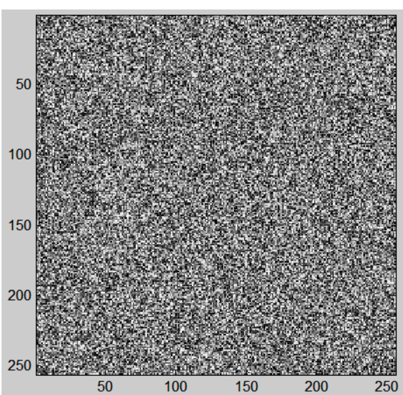
Such that
Hist(Y) = Hist(f(Z))

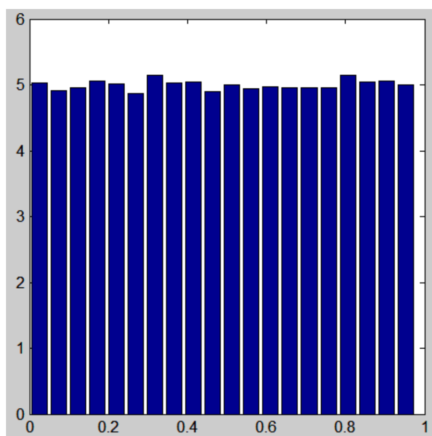


Problem: there are infinitely many functions that can do this transformation.

A natural choice is to use *f* being:
- pointwise non linearity
- stationary
- monotonic (most of the time invertible)

Y(x,y)

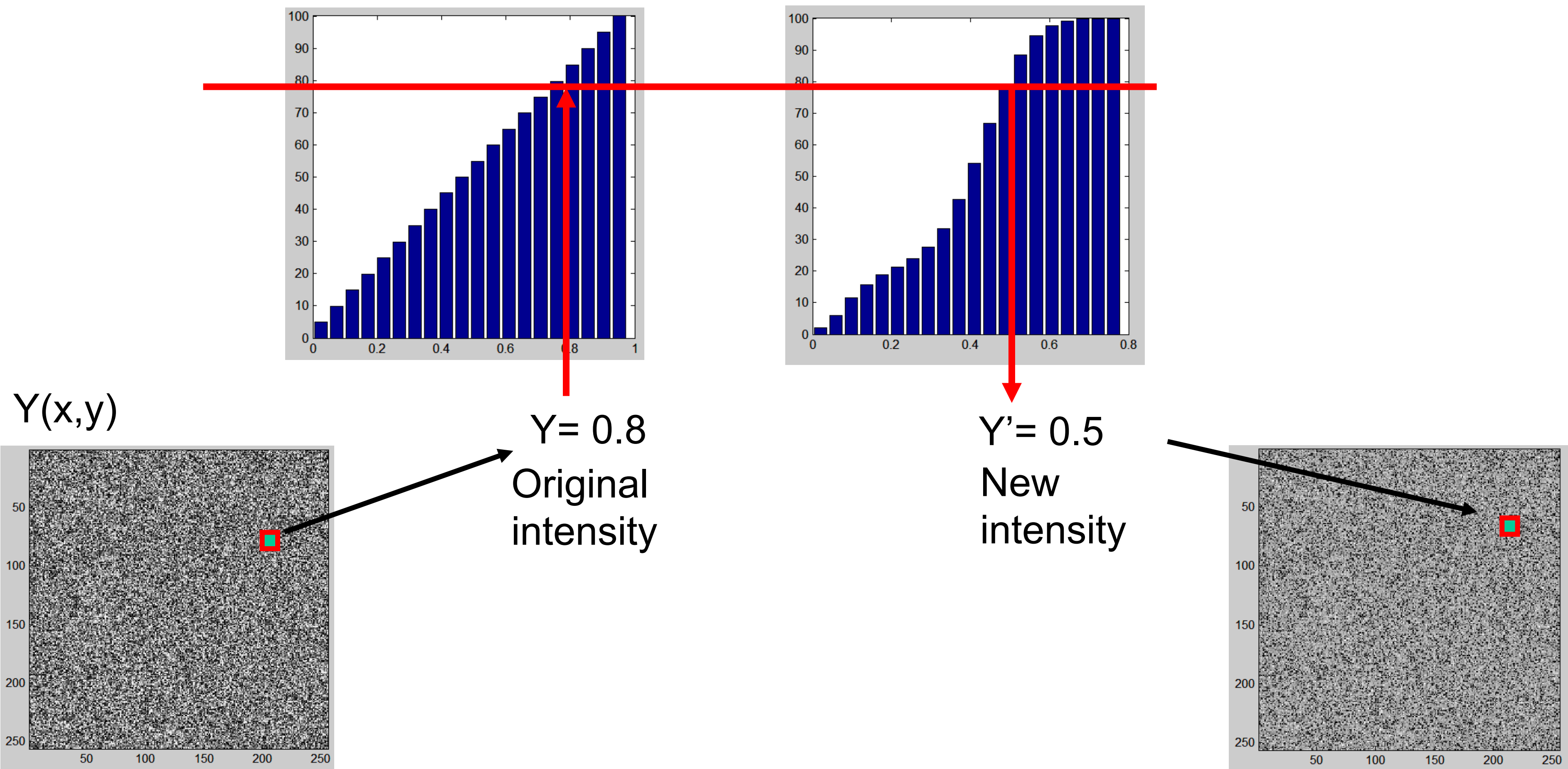




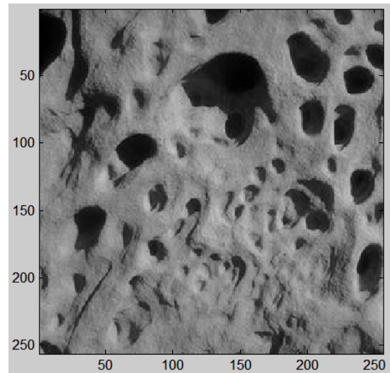
2-Matching histograms

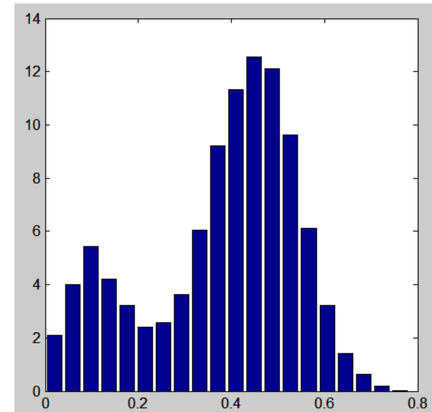
The function f is just a look up table: it says, change all the pixels of value Y into a value f(Y).

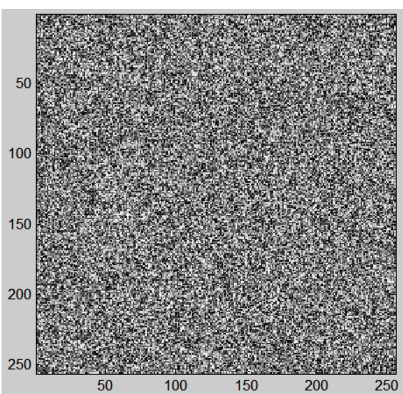
$$Y' = f\,(Y)$$



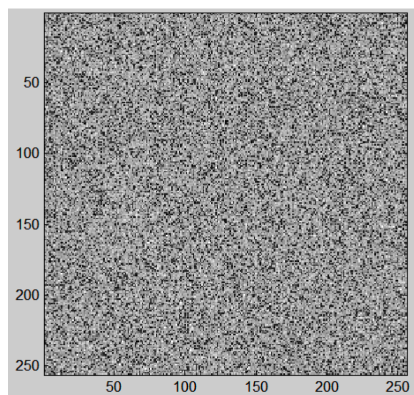
2-Matching histograms

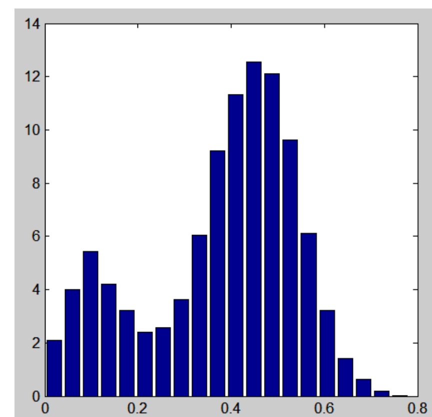




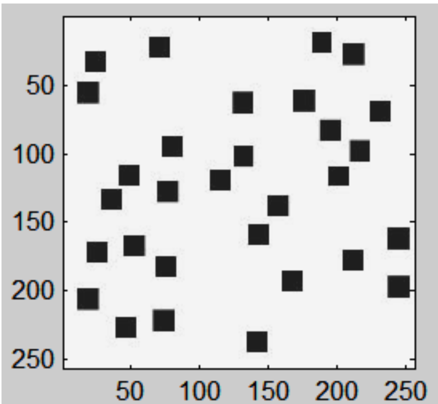


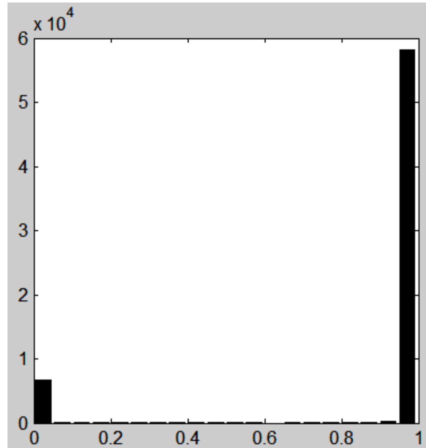
Y’ = f (Y)



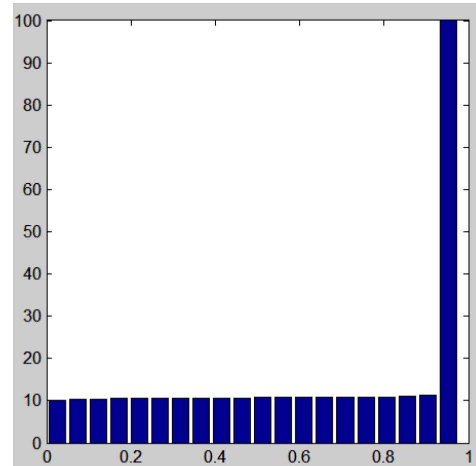


Another example: Matching histograms



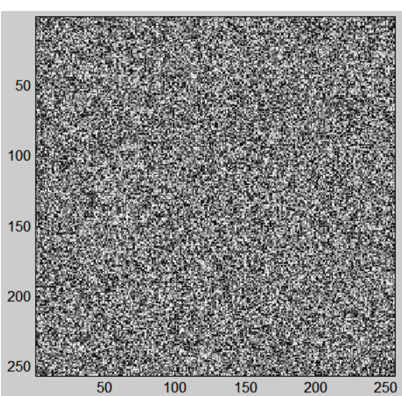


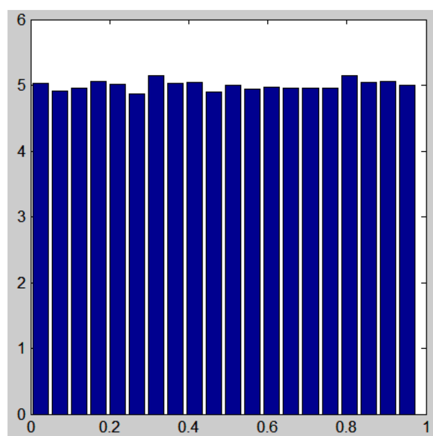
?

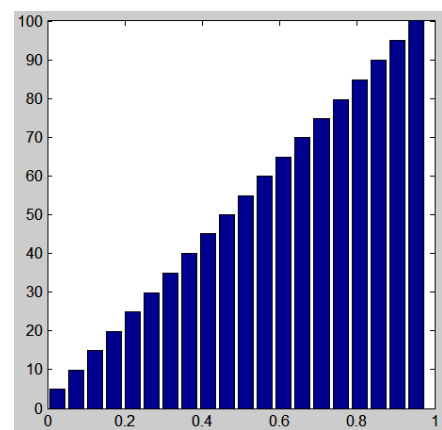


10% of pixels are black
and 90% are white

?



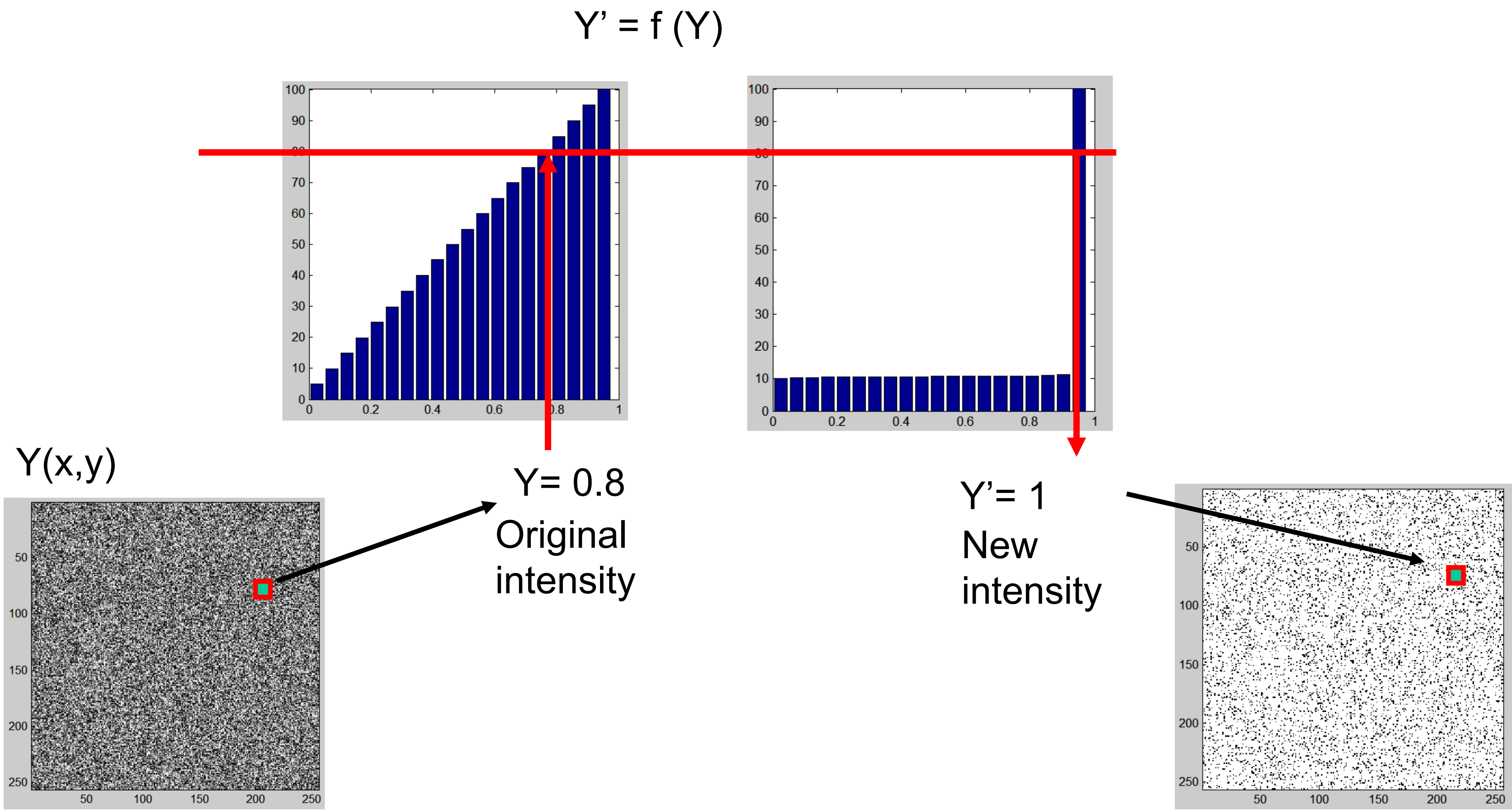




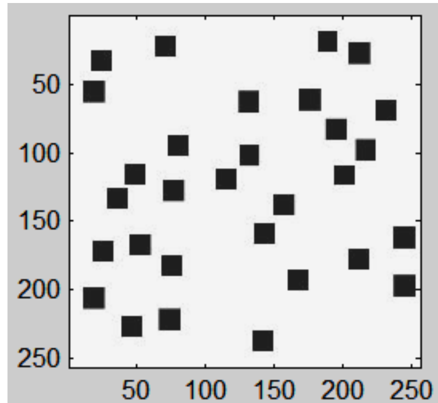
5% of pixels have an intensity value
within the range[0.37, 0.41]

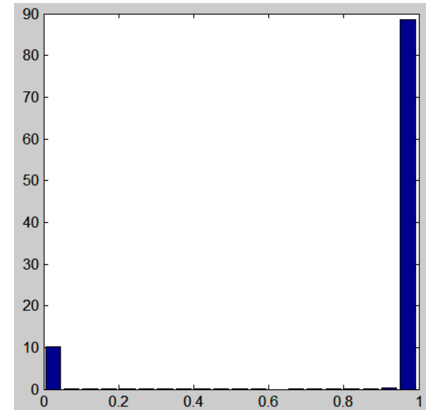
Another example: Matching histograms

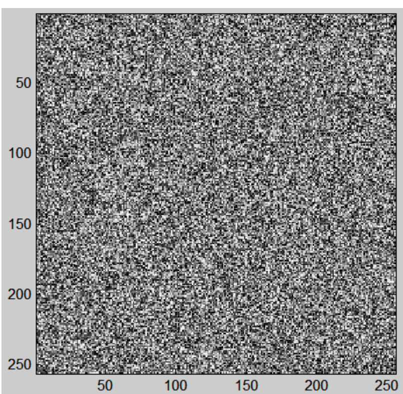
The function f is just a look up table: it says, change all the pixels of value Y into a value f(Y).



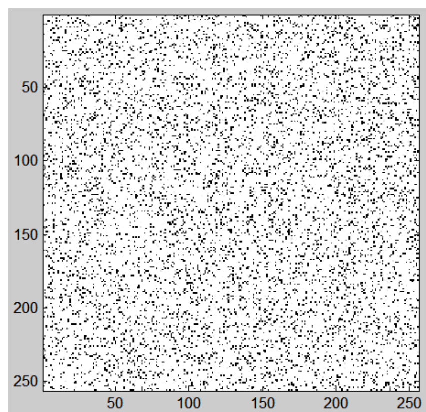
Another example: Matching histograms

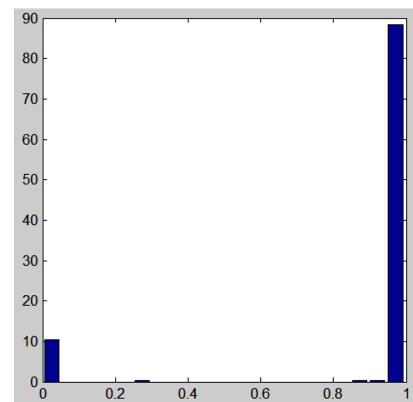






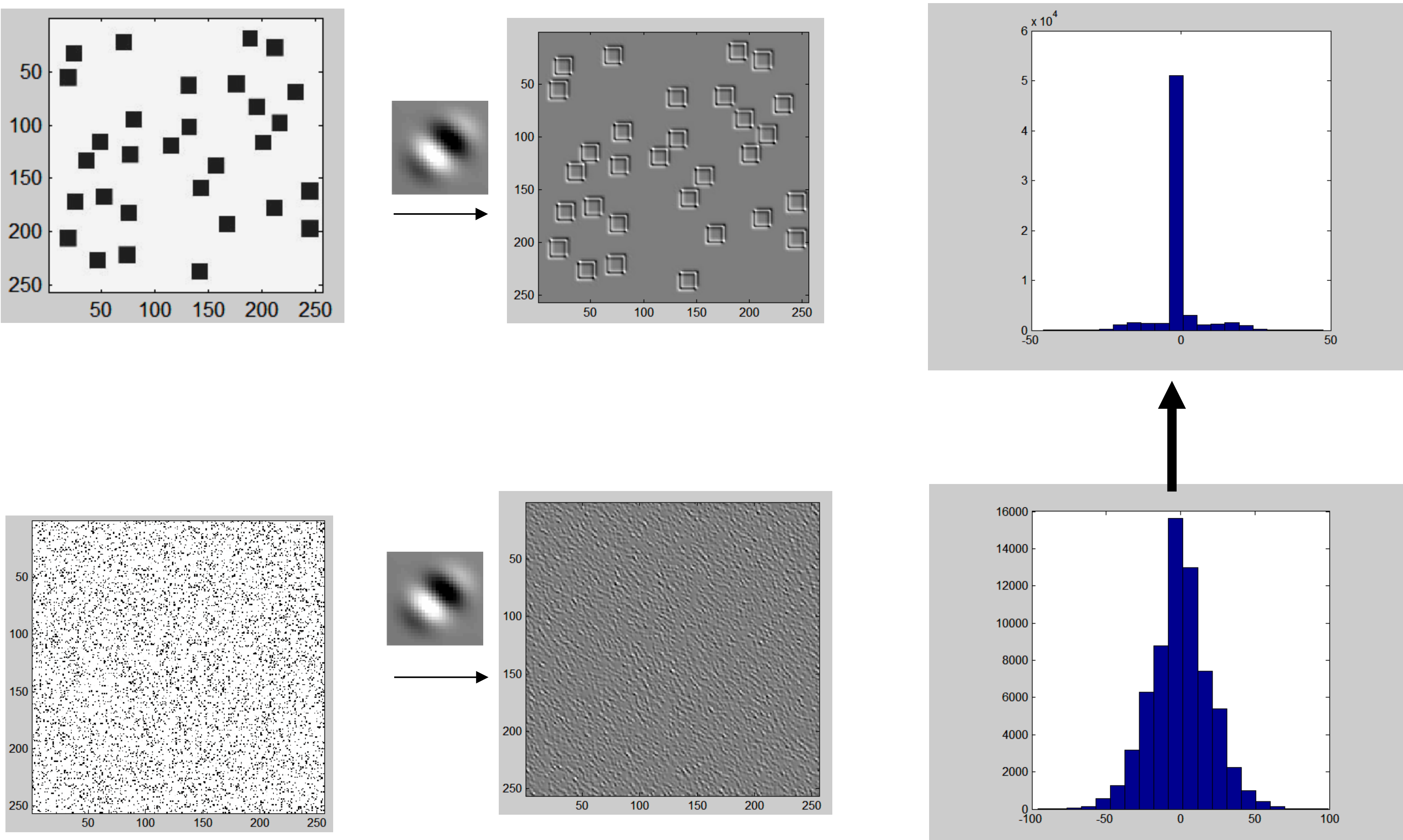
Y’ = f (Y)



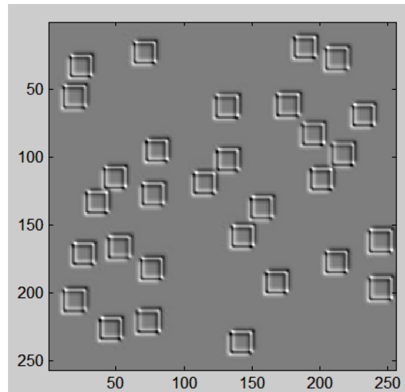


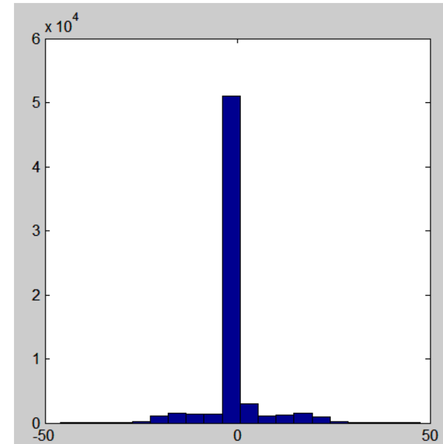
In this example, f is a step function.

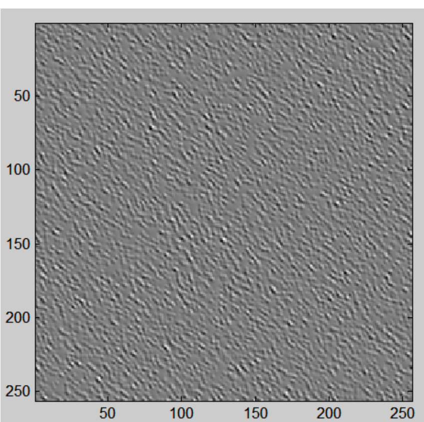
Matching histograms of a subband



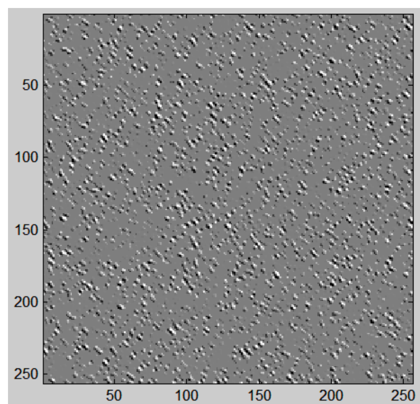
Matching histograms of a subband

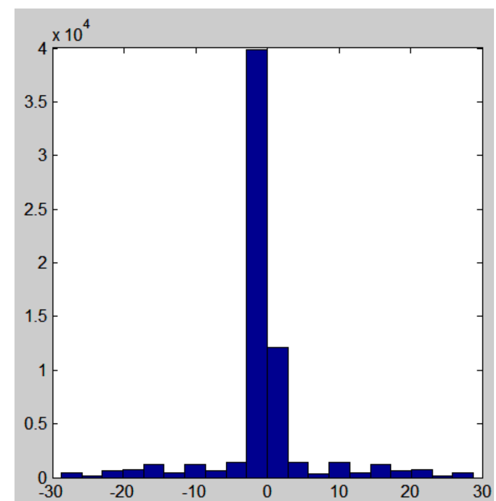






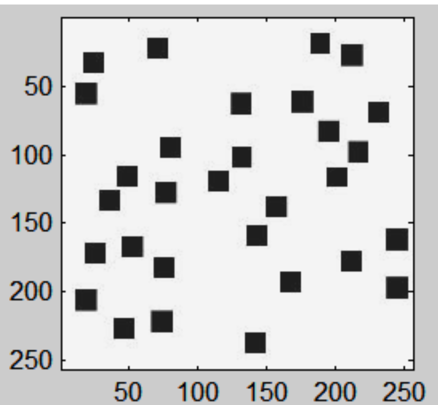
Y’ = f (Y)



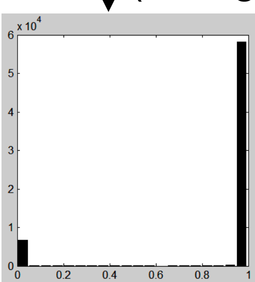


Texture analysis

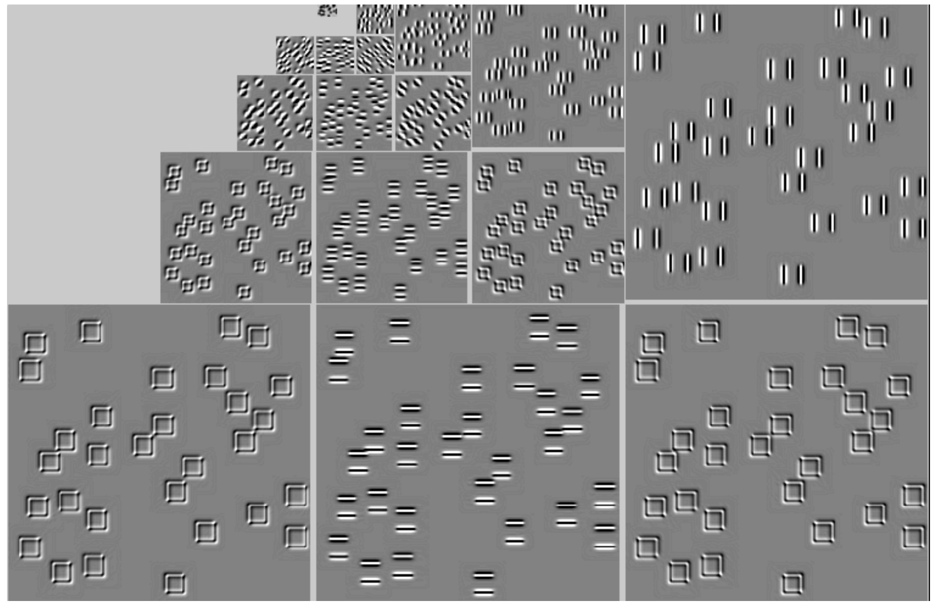
Input texture



(histogram)

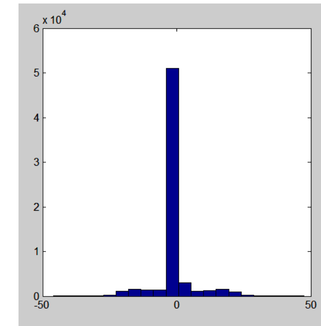


Wavelet decomposition (steerable pyr)



(Steerable pyr; Freeman & Adelson, 91)

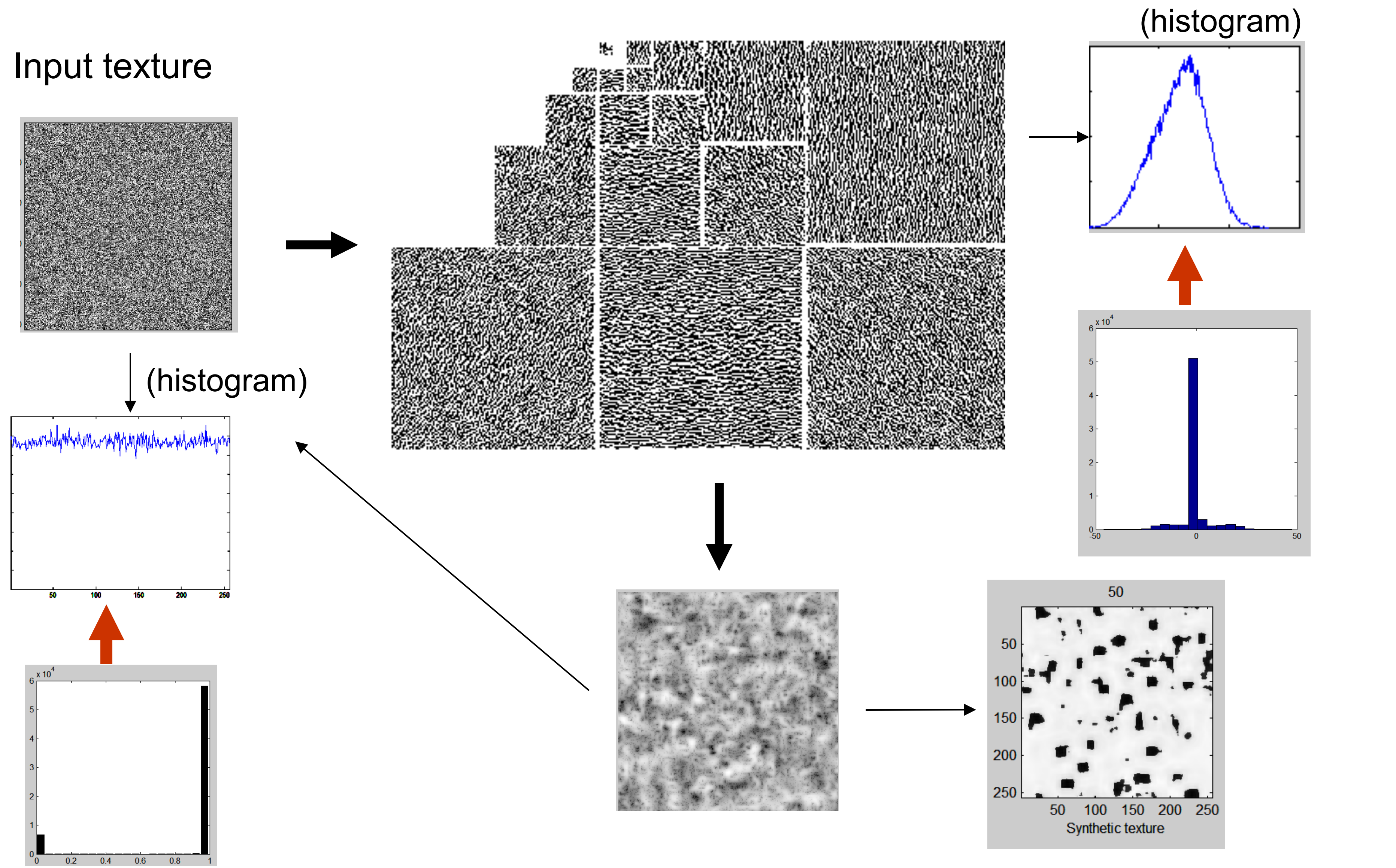
(histogram)



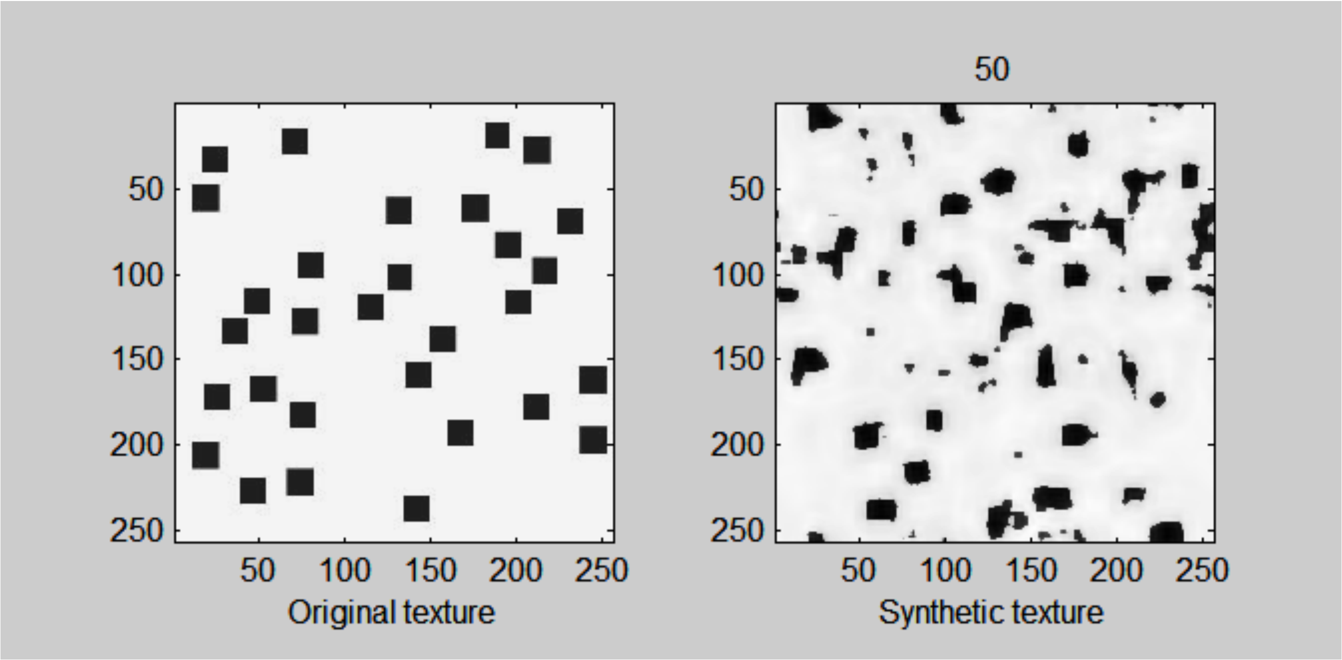
The texture is represented as a collection of marginal histograms.

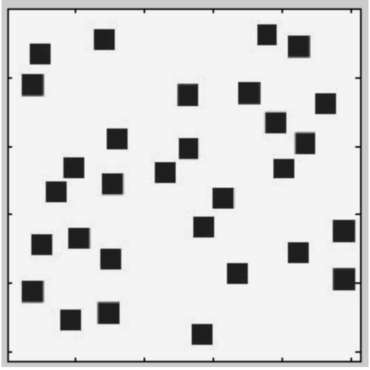
Texture synthesis

Heeger and Bergen, 1995



Why does it work? (sort of)



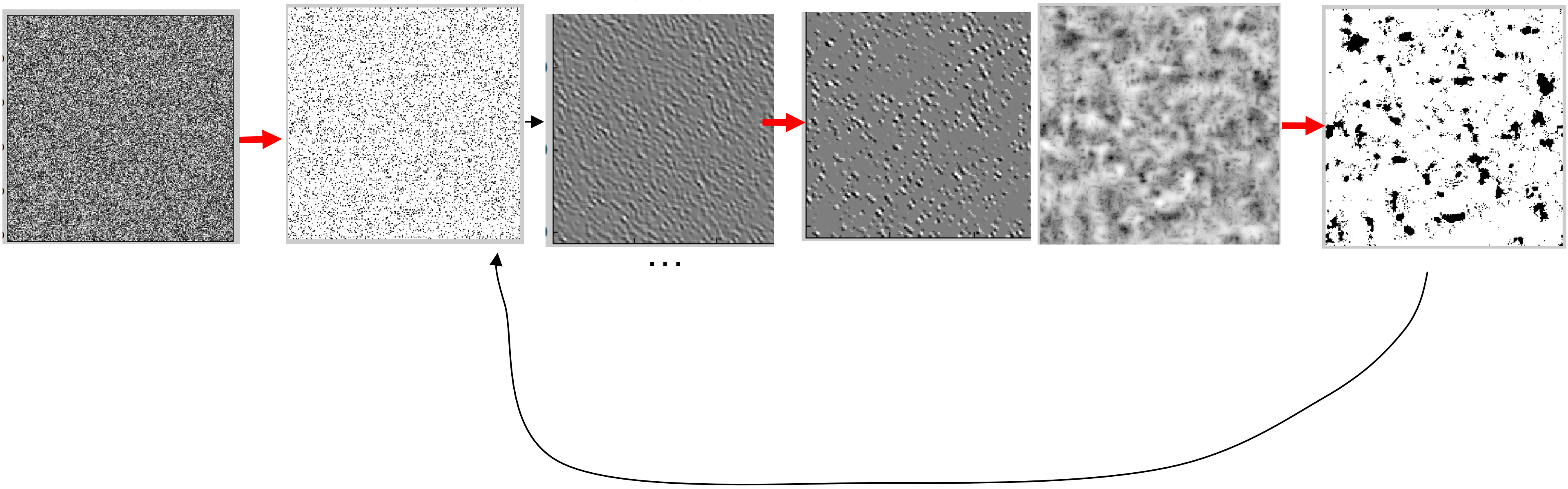


Why does it work? (sort of)

The black and white blocks appear by thresholding (f) a blobby image

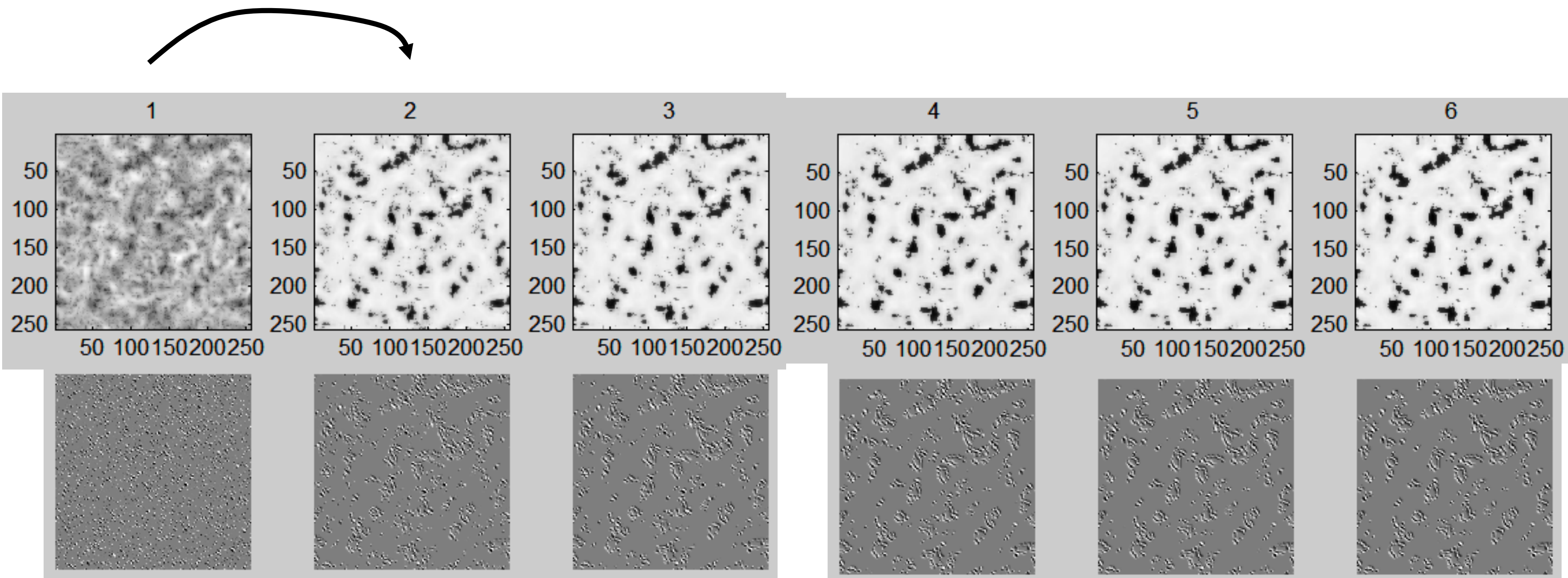
Iteration 0

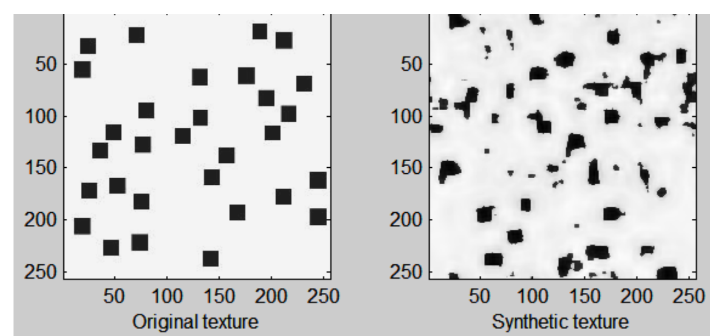
Filter bank



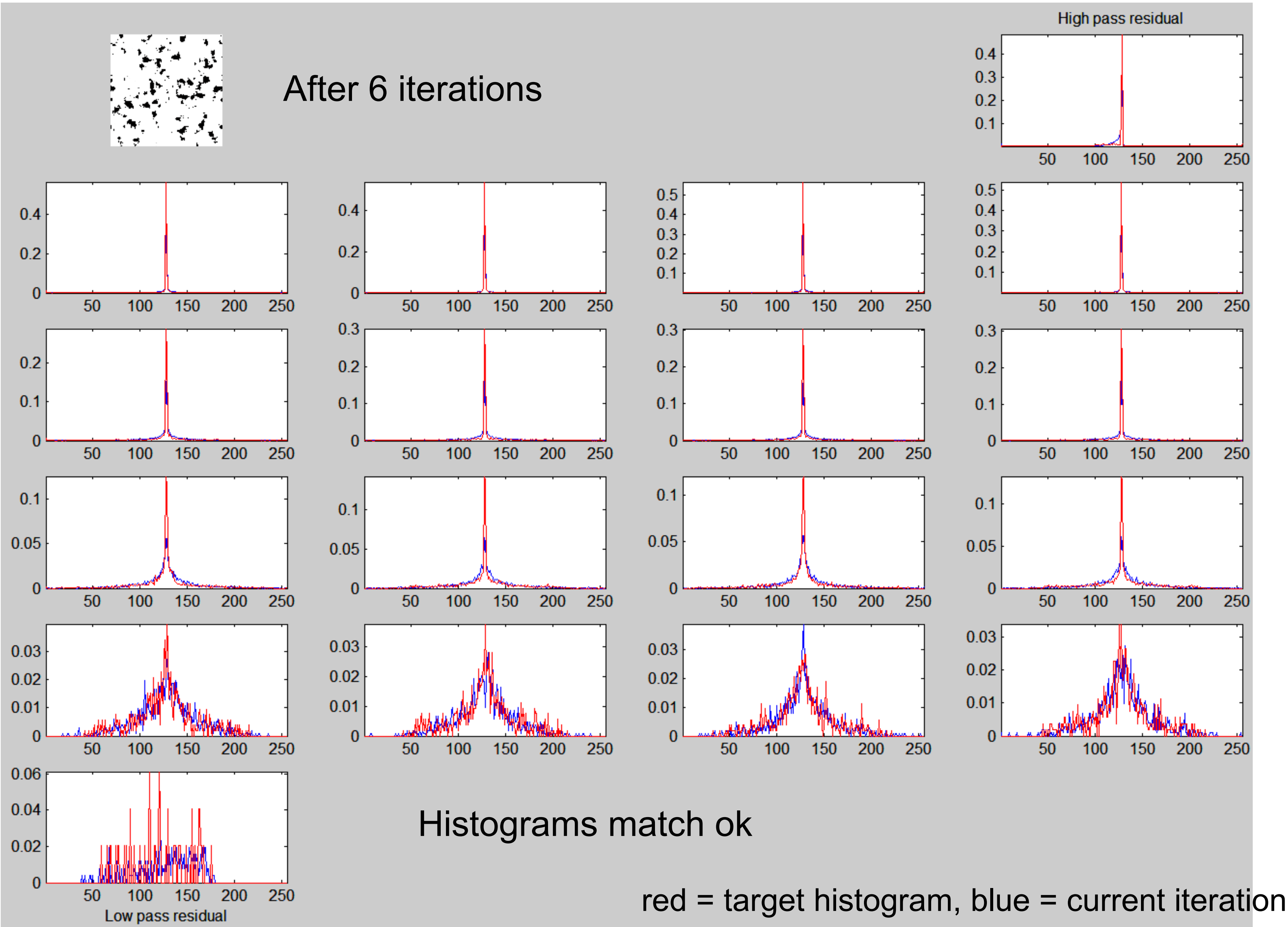
Why does it work? (sort of)

The black and white blocks appear by thresholding (f) a blobby image

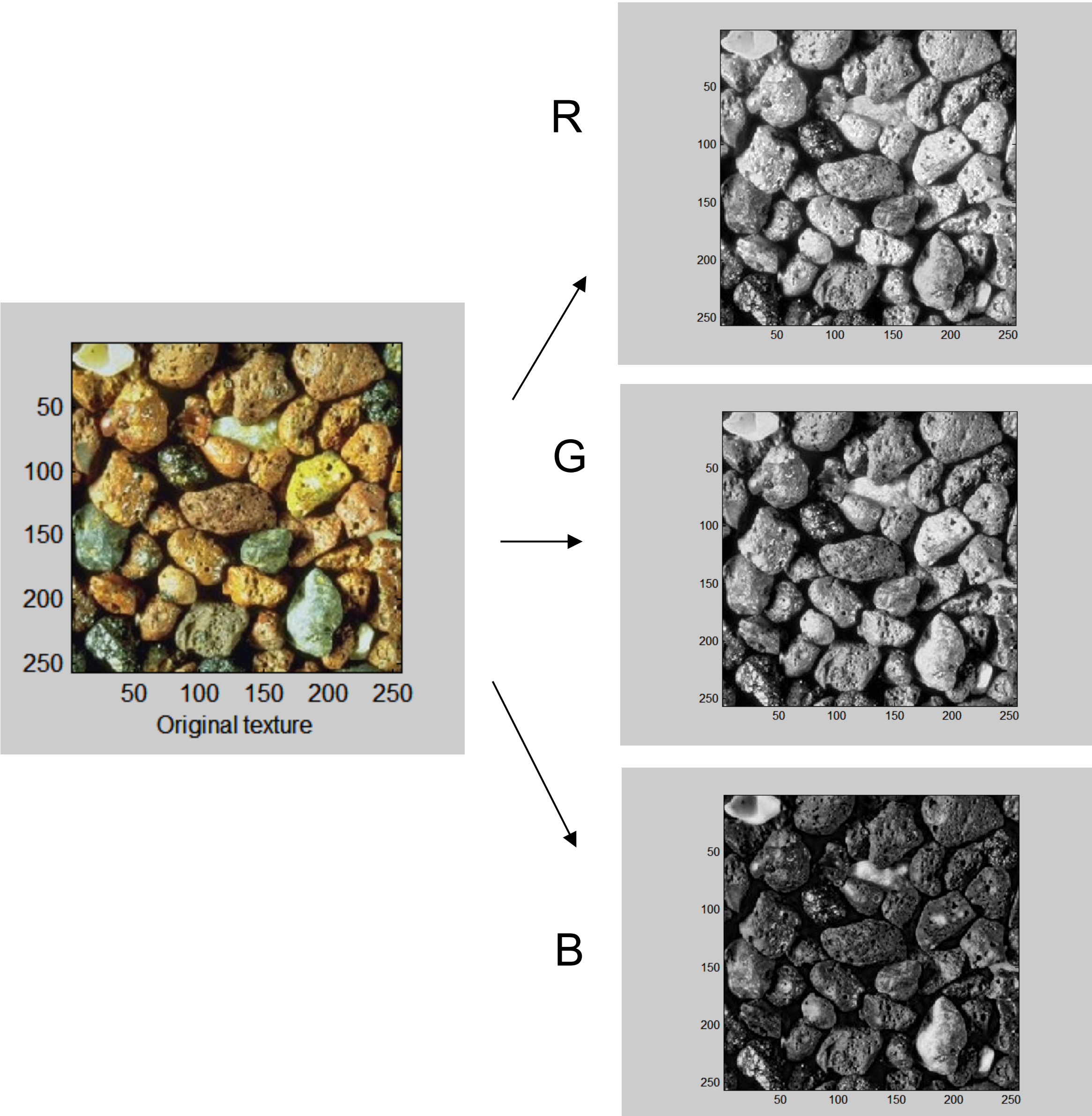




Why does it work? (sort of)

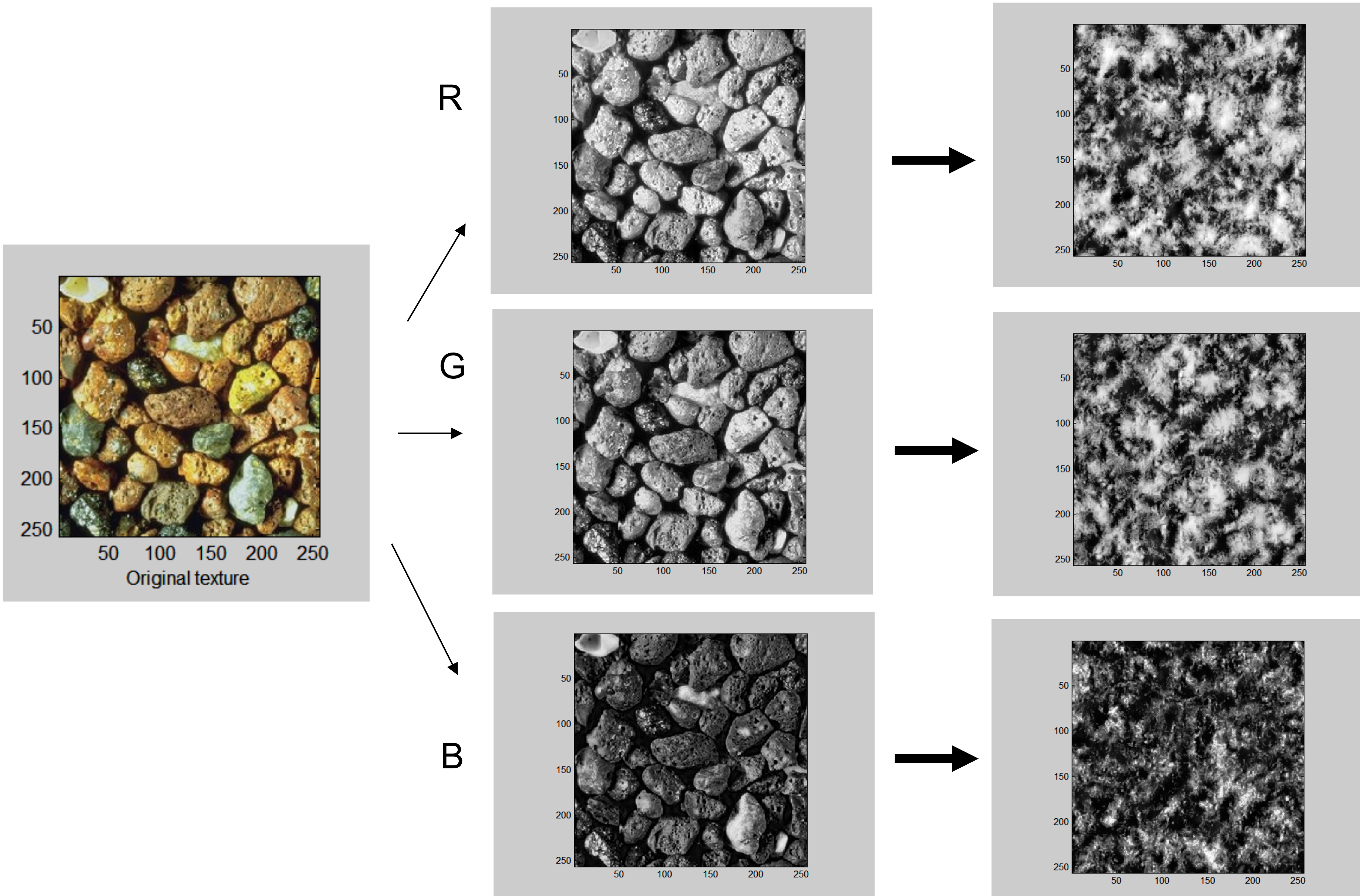


Color textures



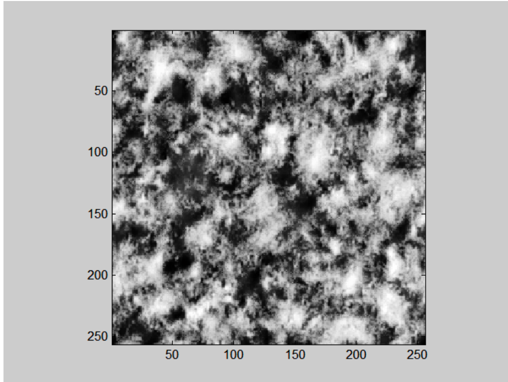
Three textures

Color textures

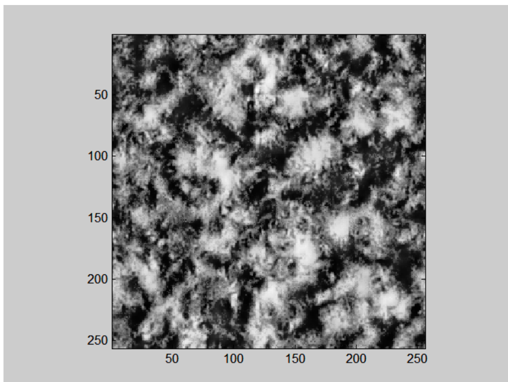


Color textures

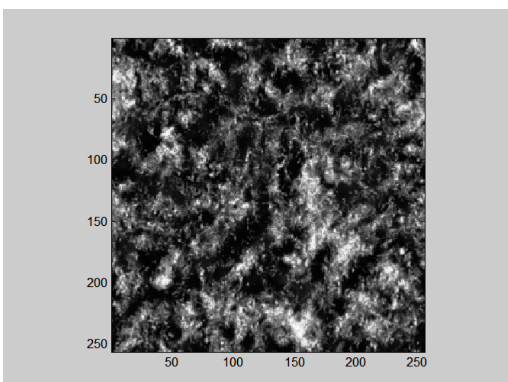
R



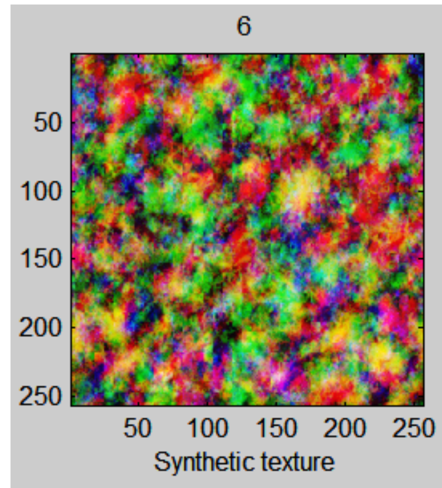
G

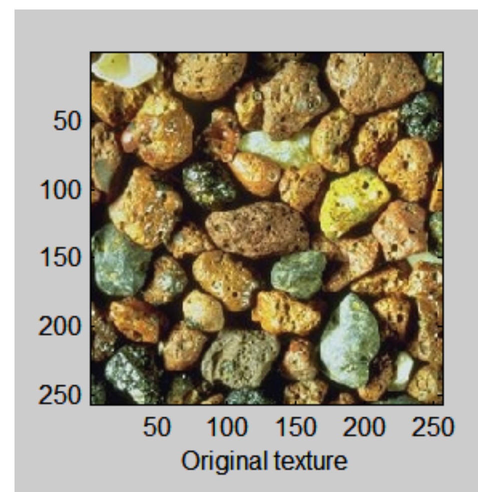


B



This does not work

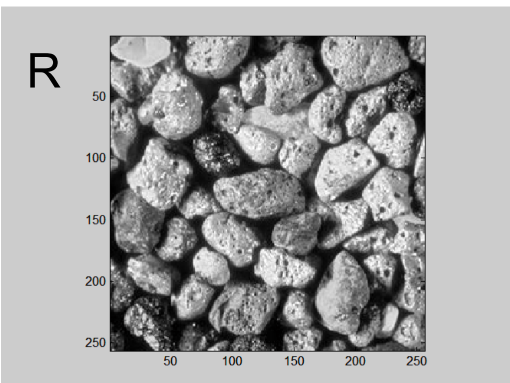


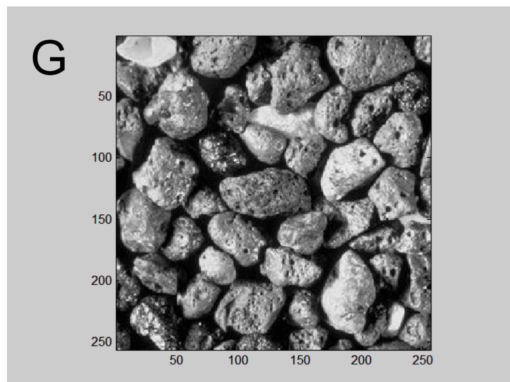


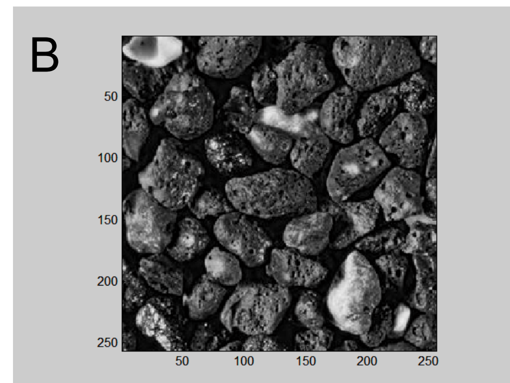
Color textures

Problem: we create new colors not present in the original image.

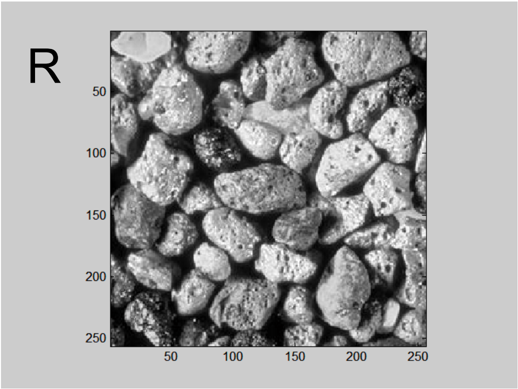
Why? Color channels are not independent.

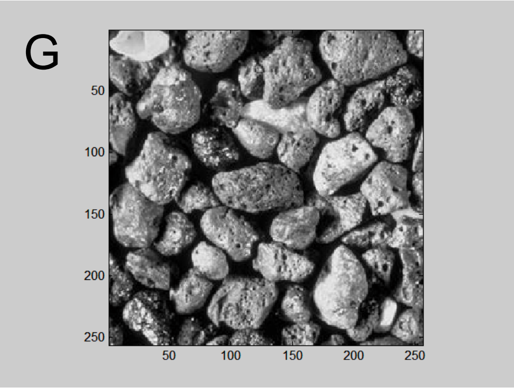


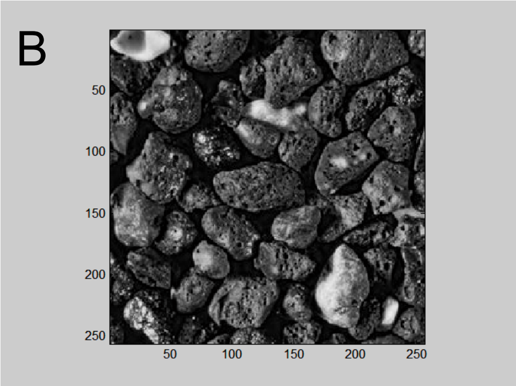




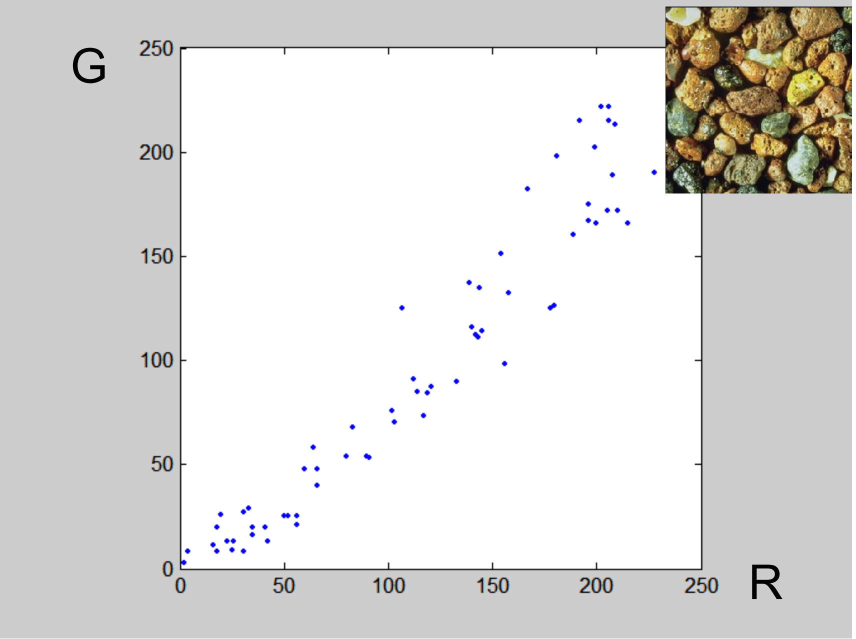
PCA and decorrelation

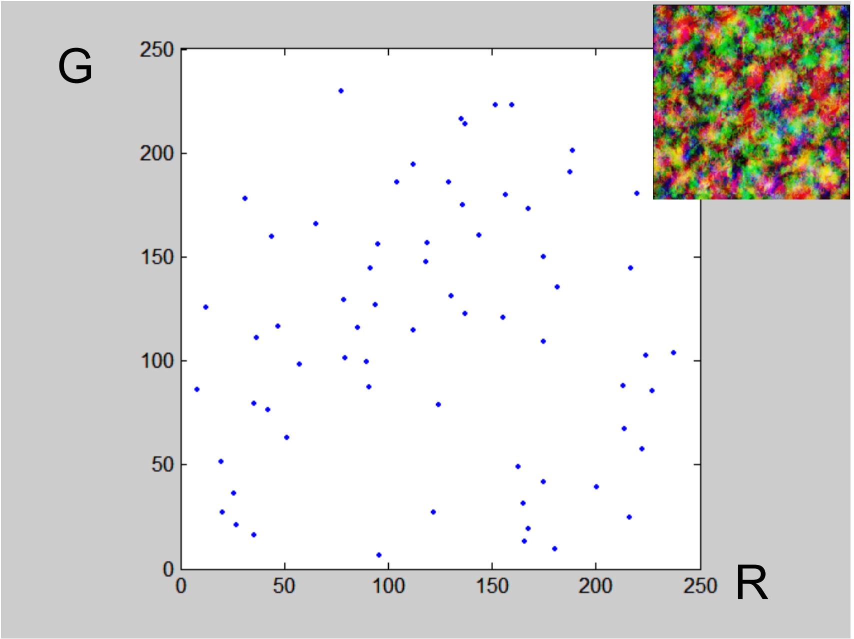






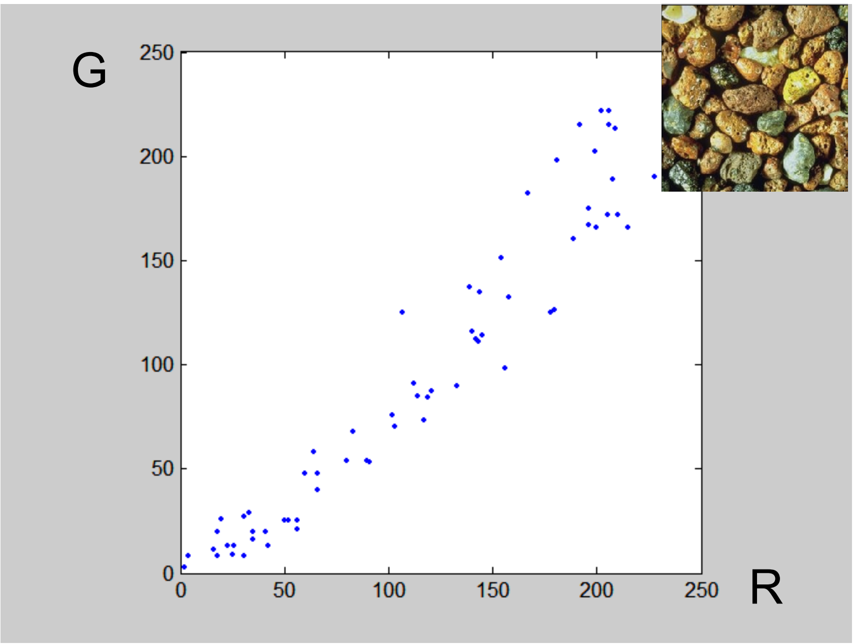
In the original image, R and G are correlated, but, after synthesis,…



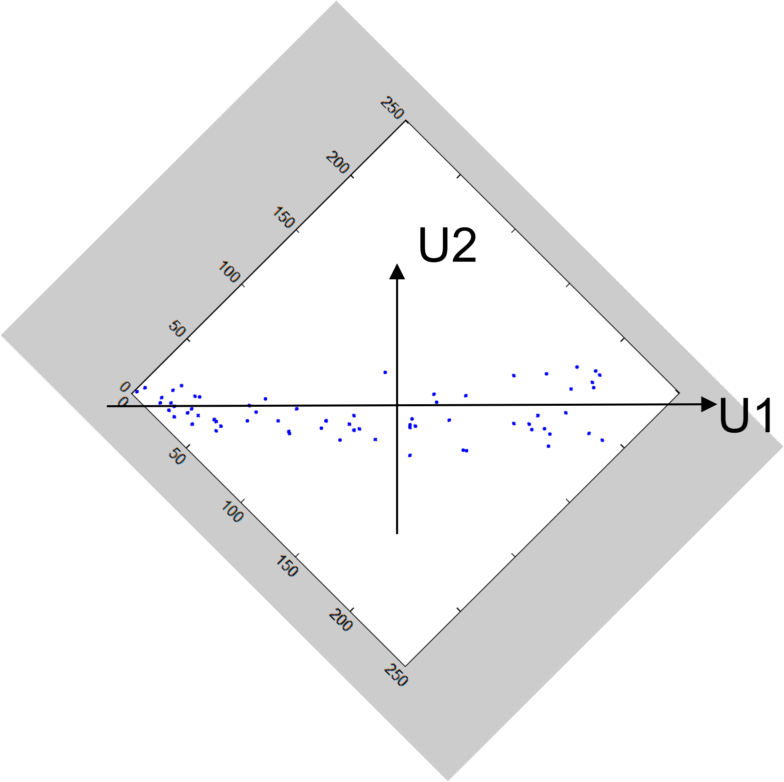


PCA and decorrelation

The texture synthesis algorithm assumes that the channels are independent.
What we want to do is some rotation

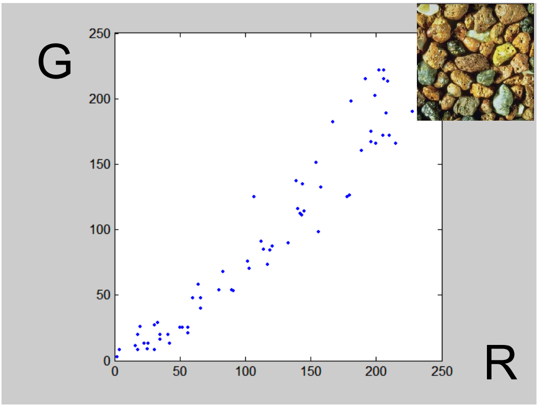


Rotation



See that in this rotated space, if I specify one coordinate the other remains unconstrained.

PCA and decorrelation



correlation(R,G)

$$C = \begin{matrix} 1.0000 & 0.9303 & 0.6034 \\ 0.9303 & 0.9438 & 0.6620 \\ 0.6034 & 0.6620 & 0.5569 \end{matrix}$$

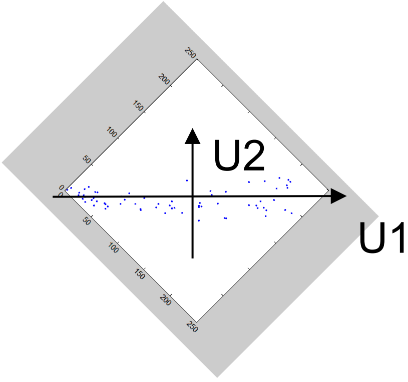
PCA finds the principal directions of variation of the data.
It gives a decomposition of the covariance matrix as:

$$C = D\,D'$$

$$D = \begin{matrix} 0.6347 & 0.6072 & 0.4779 \\ 0.6306 & -0.0496 & -0.7745 \\ 0.4466 & -0.7930 & 0.4144 \end{matrix}$$

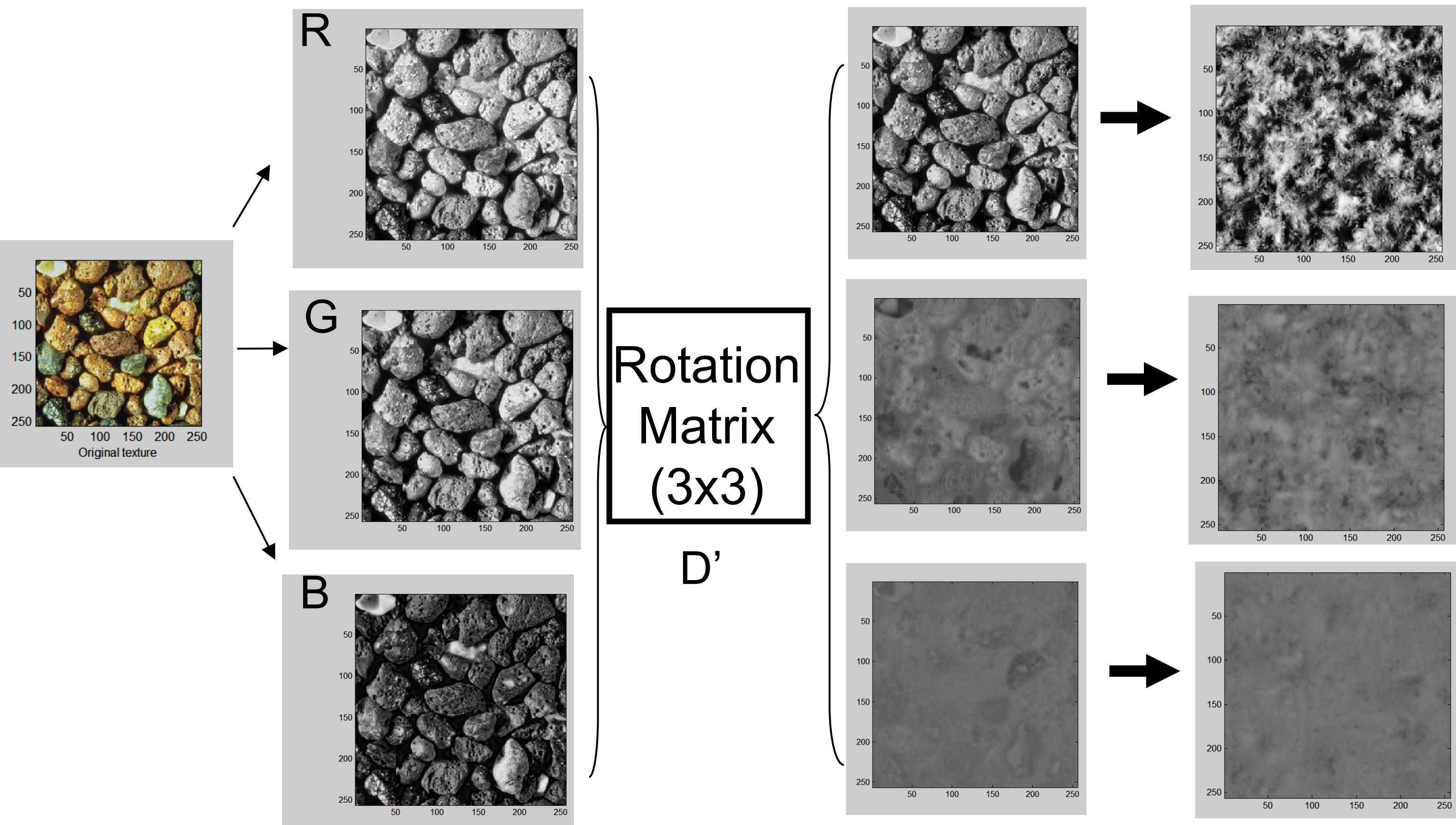
By transforming the original data (RGB) using D we get:

$$\underbrace{\boxed{\begin{matrix} U1 \\ U2 \\ U3 \end{matrix}}}_{3 \times \text{Npixels}} = \underbrace{\boxed{D'}}_{3 \times 3} \underbrace{\boxed{\begin{matrix} R \\ G \\ B \end{matrix}}}_{3 \times \text{Npixels}}$$



The new components (U1,U2,U3) are decorrelated.

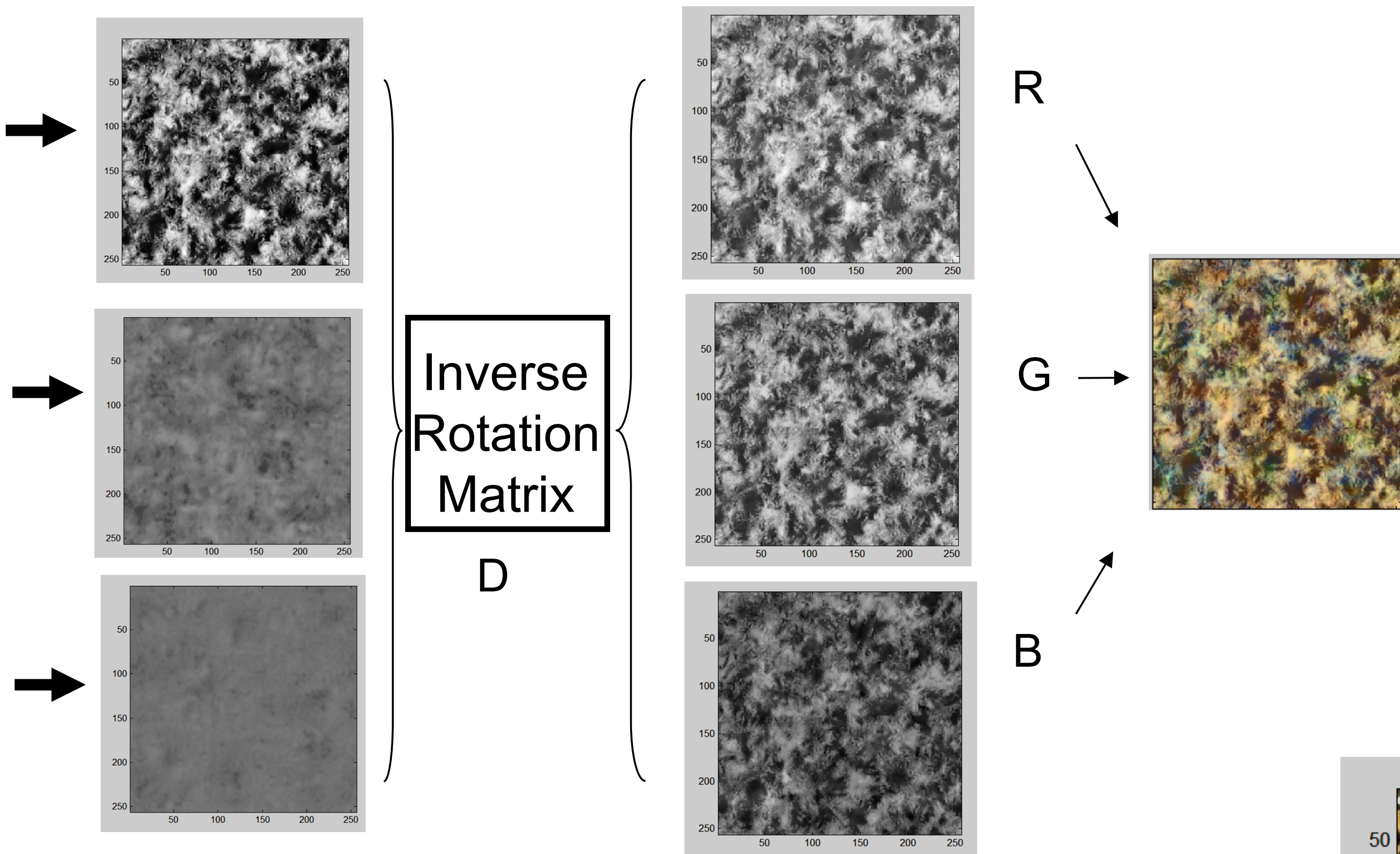
Color textures

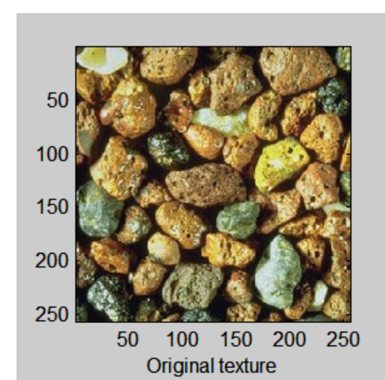


These three textures
look similar
(high dependency)

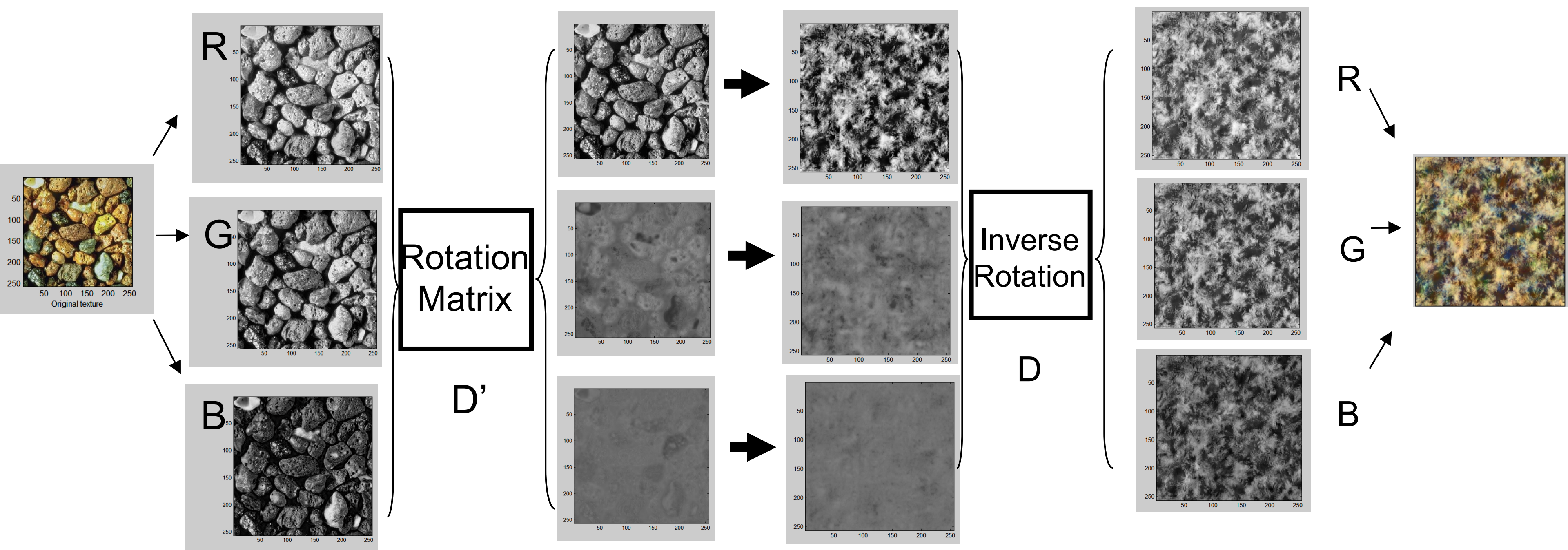
These three textures
Look less similar
(lower dependency)

Color textures





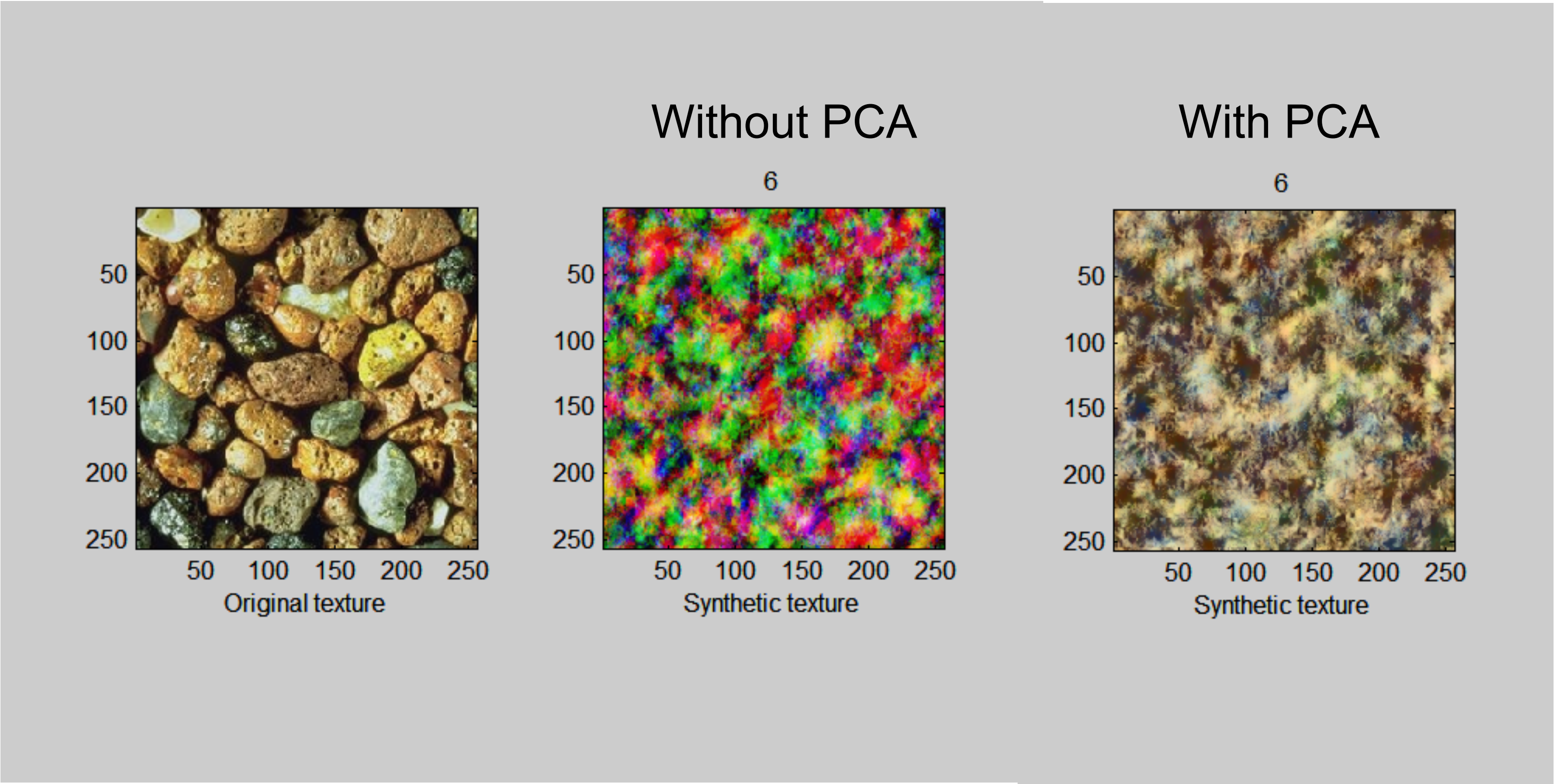
Color textures



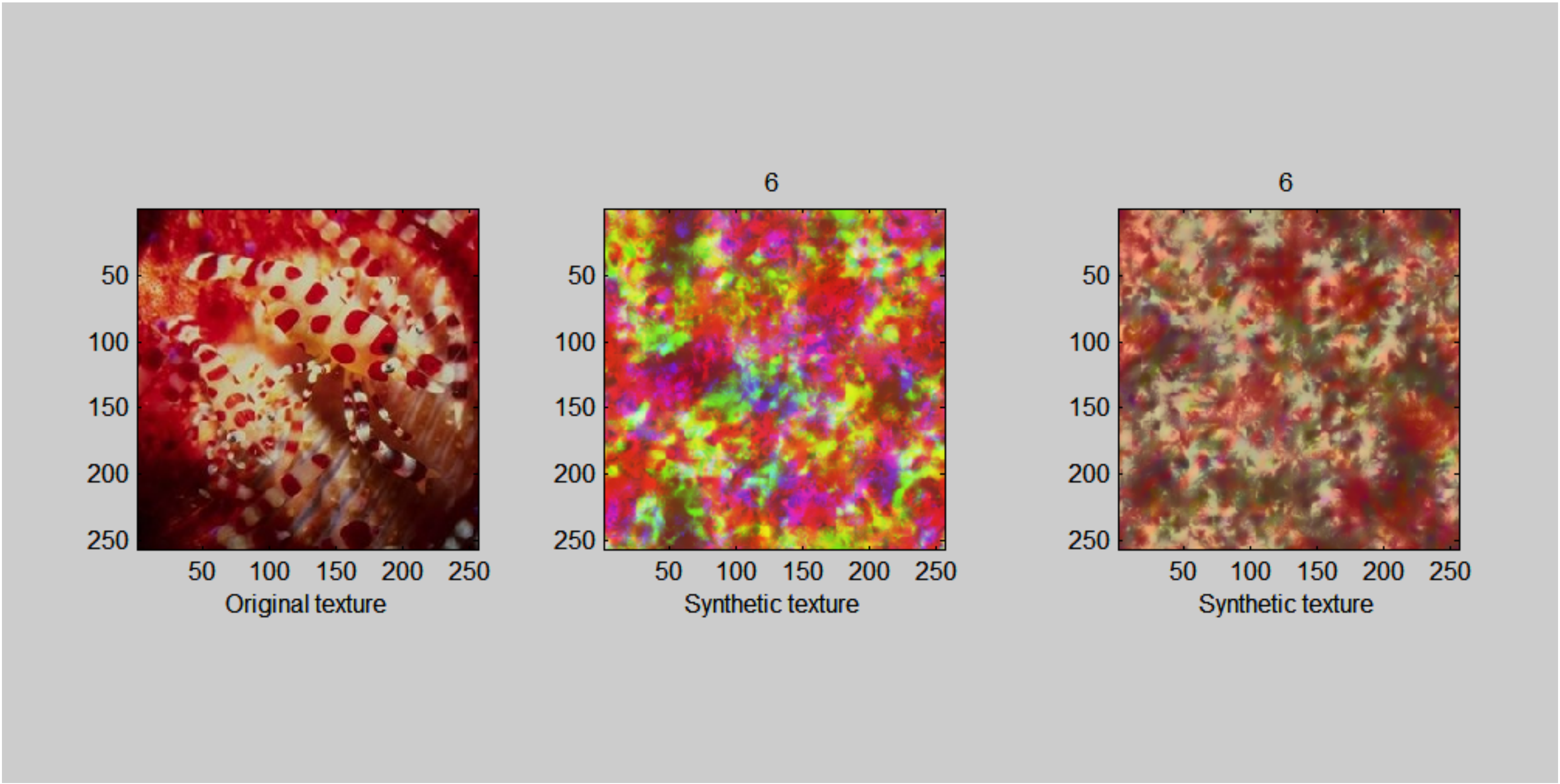
These three textures
look similar
(high dependency)

These three textures
Look less similar
(lower dependency)

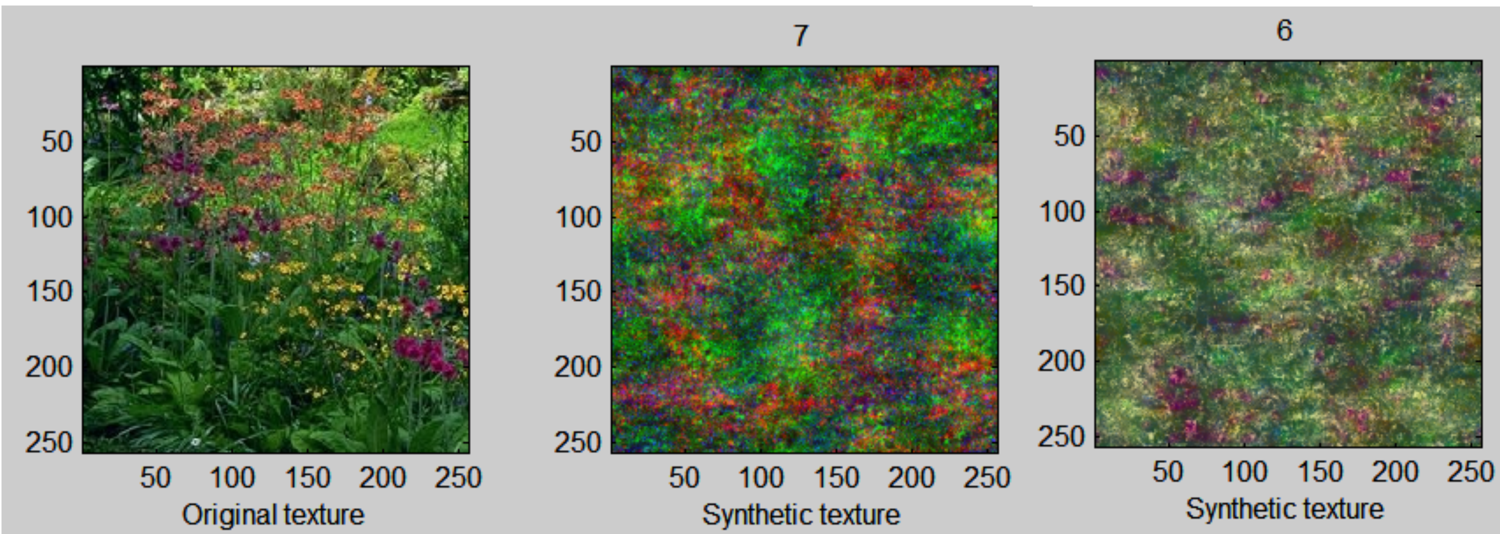
Color channels



Color channels



Color channels



Examples from the paper

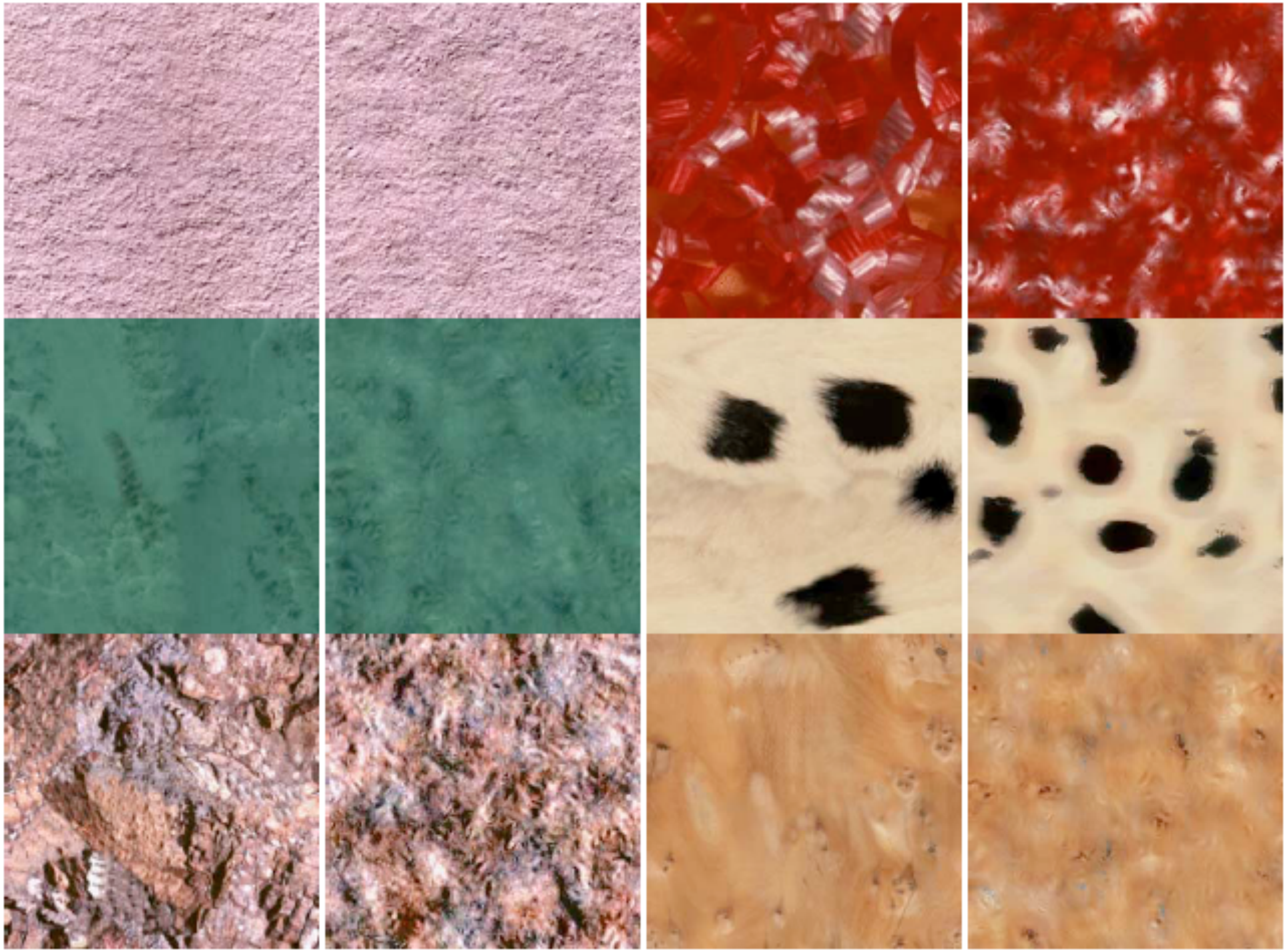


Figure 3: In each pair left image is original and right image is synthetic: stucco, iridescent ribbon, green marble, panda fur, slag stone, figured yew wood.

Heeger and Bergen, 1995

Examples from the paper

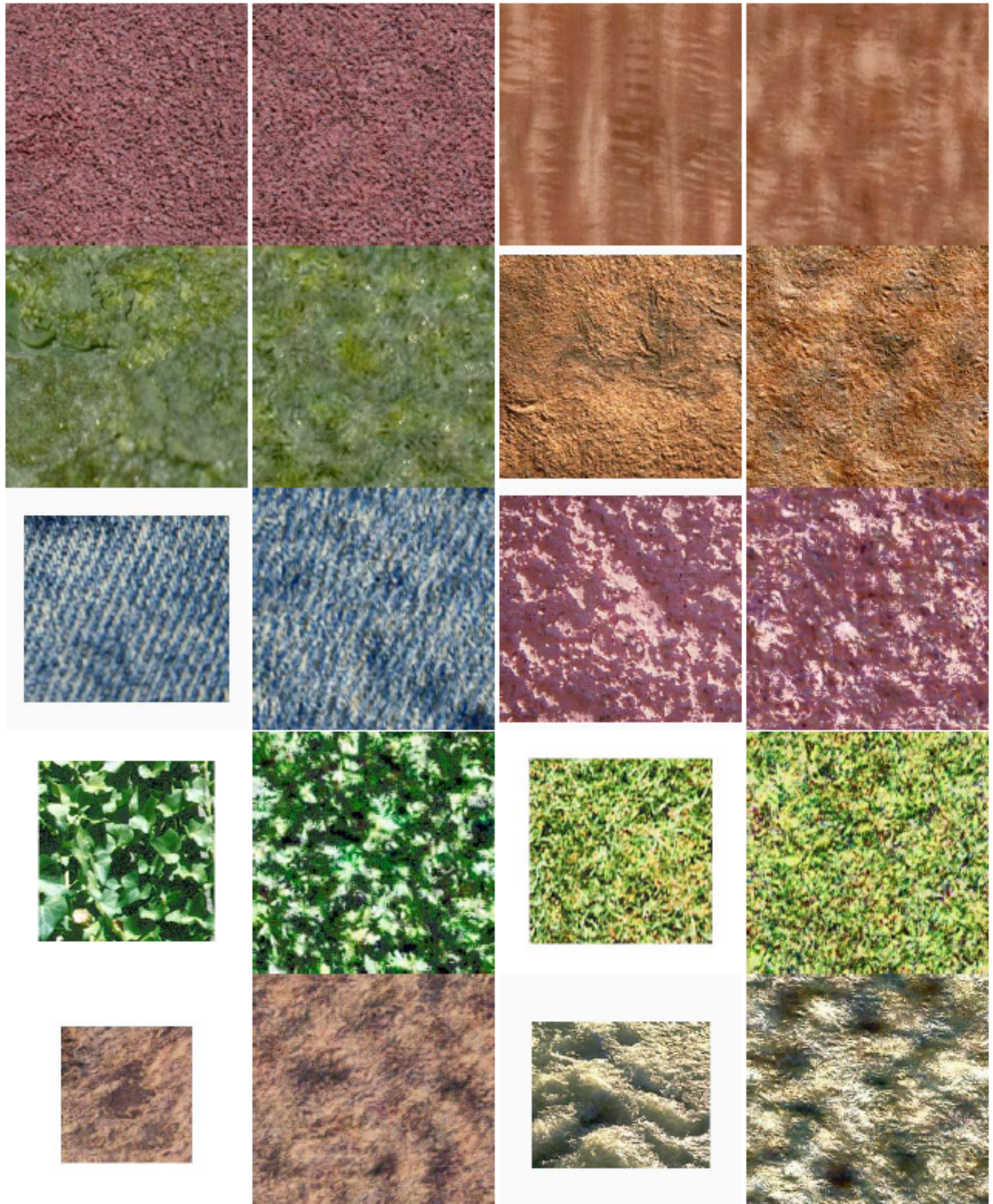
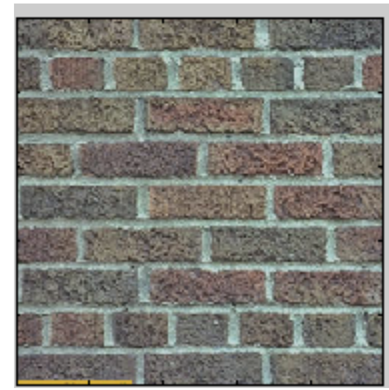


Figure 4: In each pair left image is original and right image is synthetic: red gravel, figured sepele wood, brocolli, bark paper, denim, pink wall, ivy, grass, sand, surf.

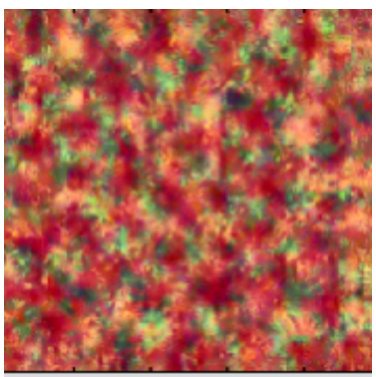
Examples not from the paper

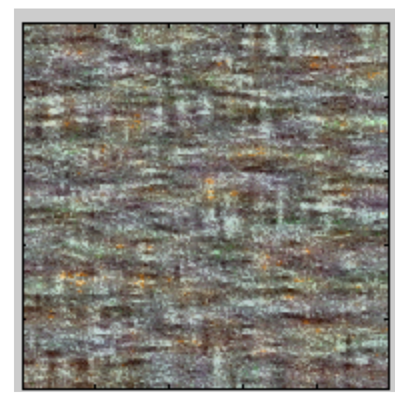
Input texture





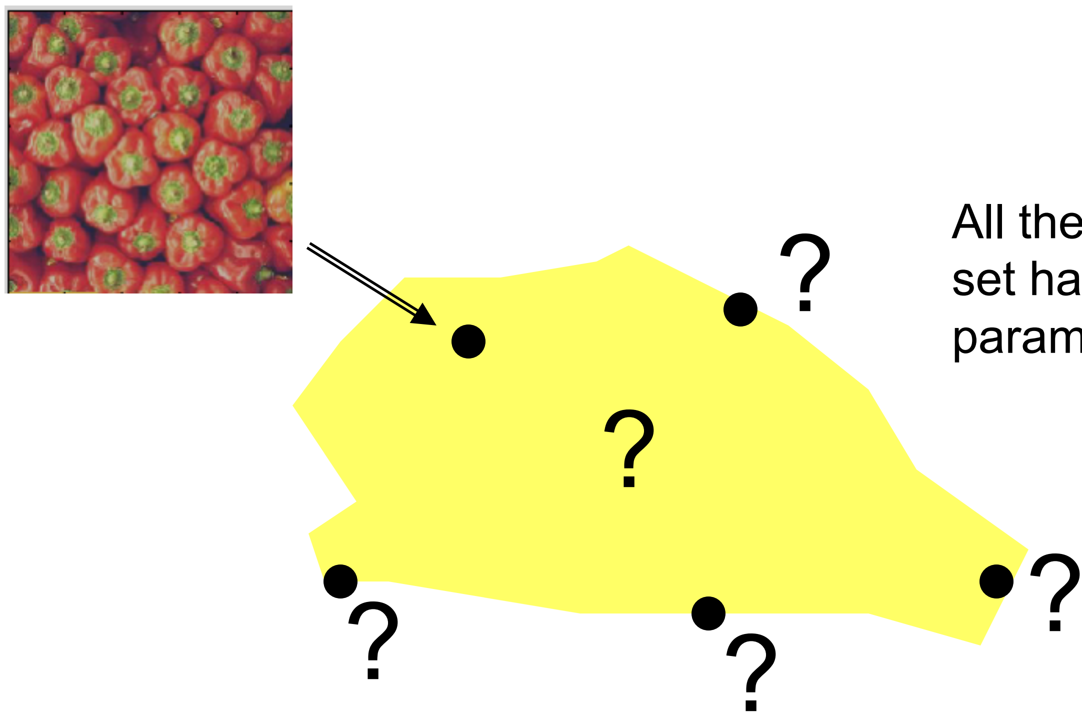
Synthetic texture





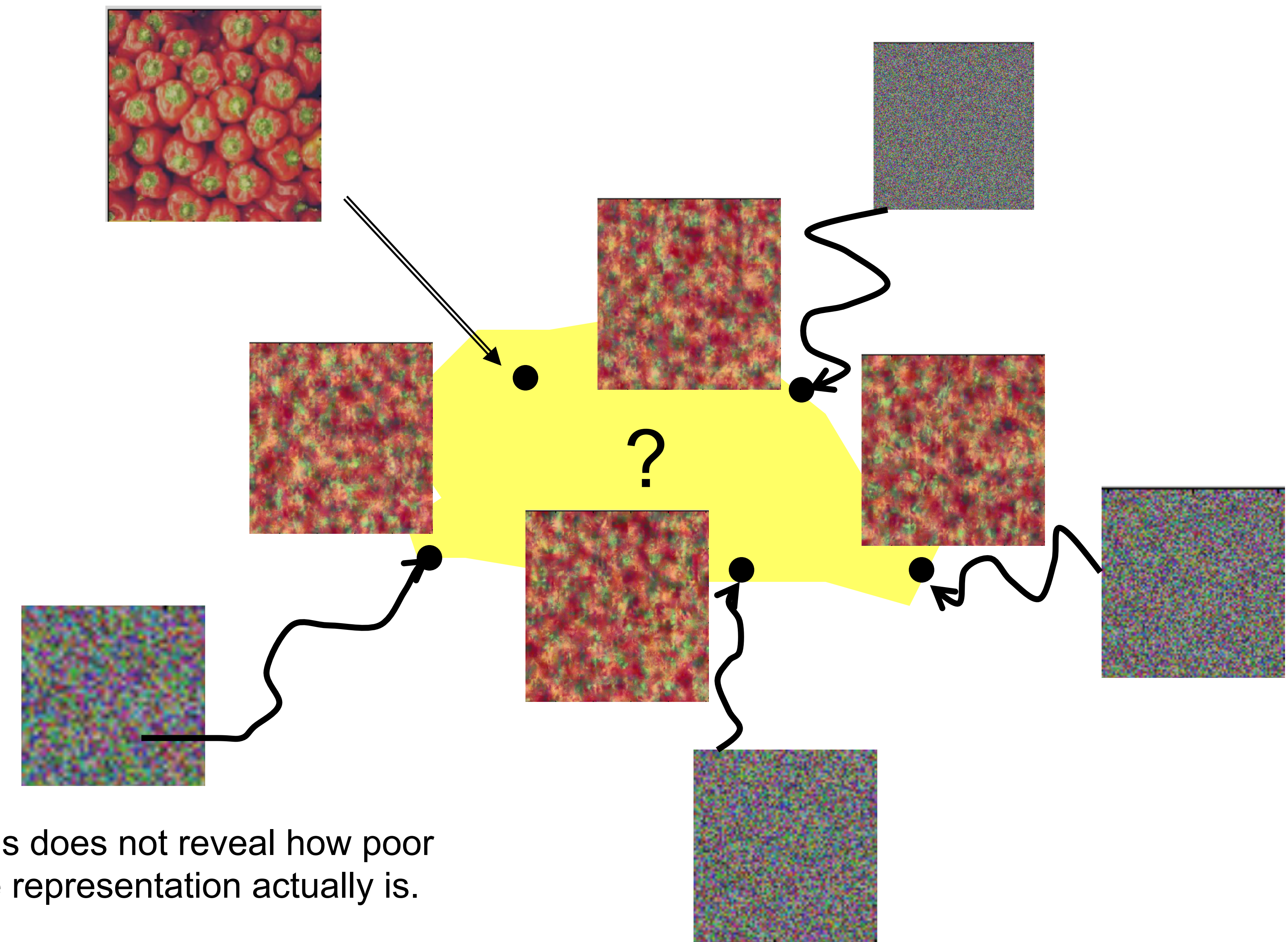
But, does it really work even when it seems to work?

But, does it really work???
How to measure how well the representation constraints the set of equivalent textures?



All the textures in this set have the same parameters.

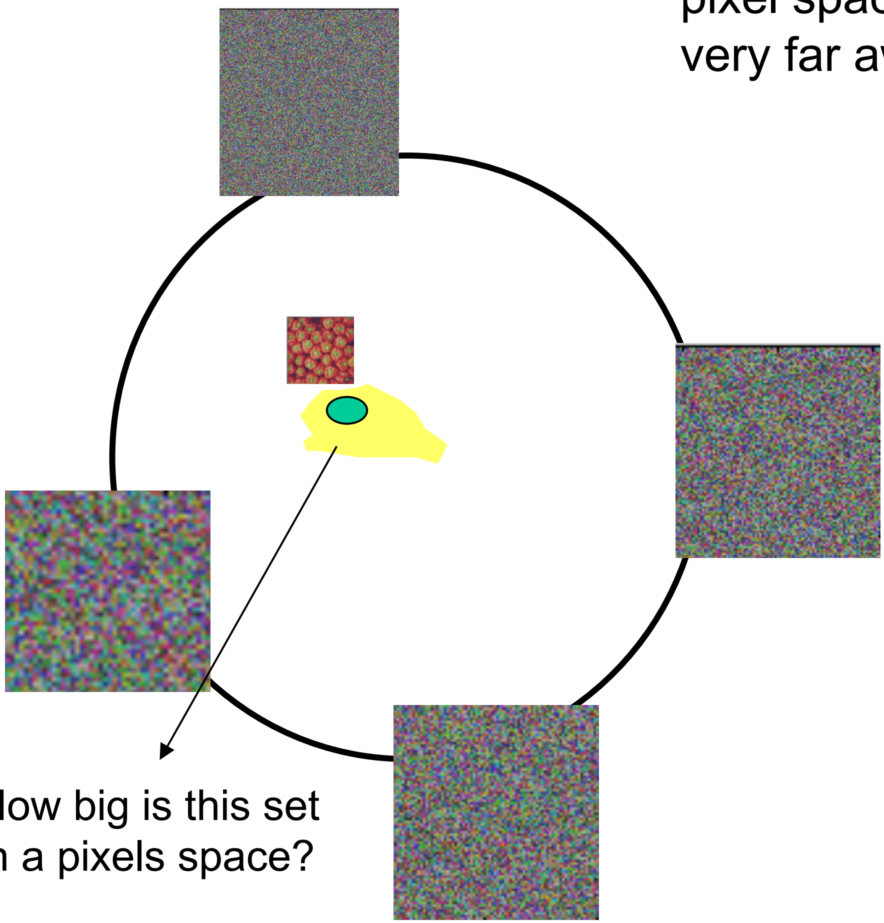
How to identify the set of equivalent textures?



This does not reveal how poor
the representation actually is.

We need a space that is more perceptual

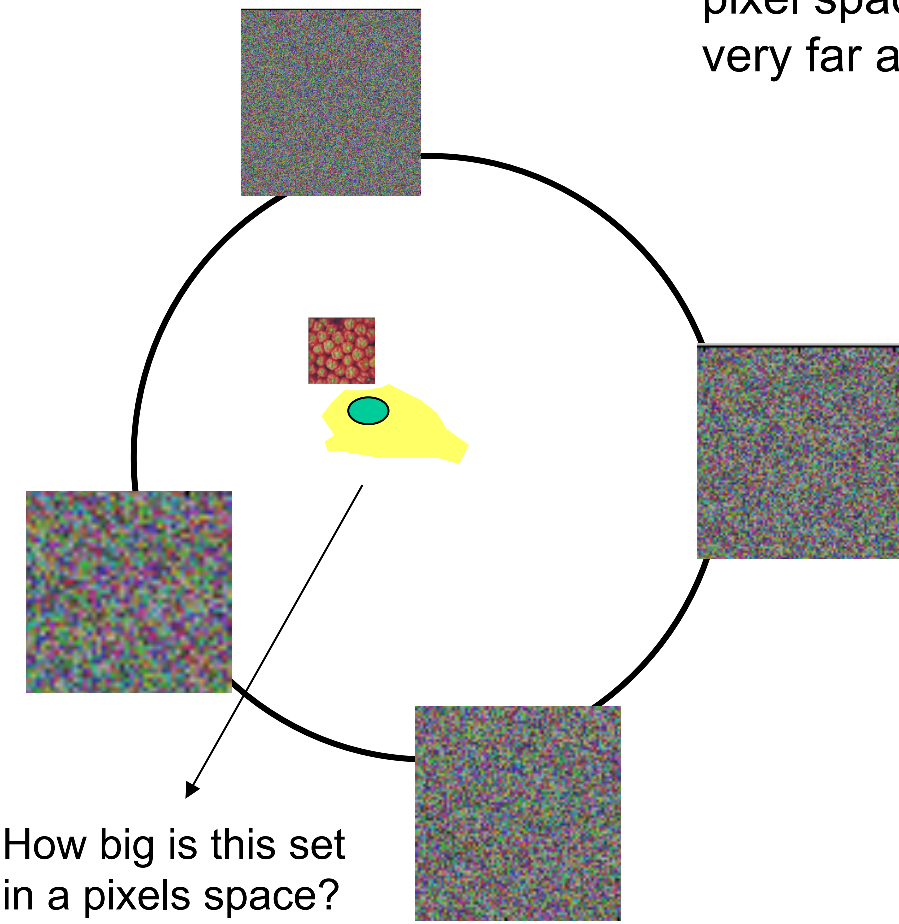
In a perceptual space all these noise images are very close. But in pixel space, they are very far away.

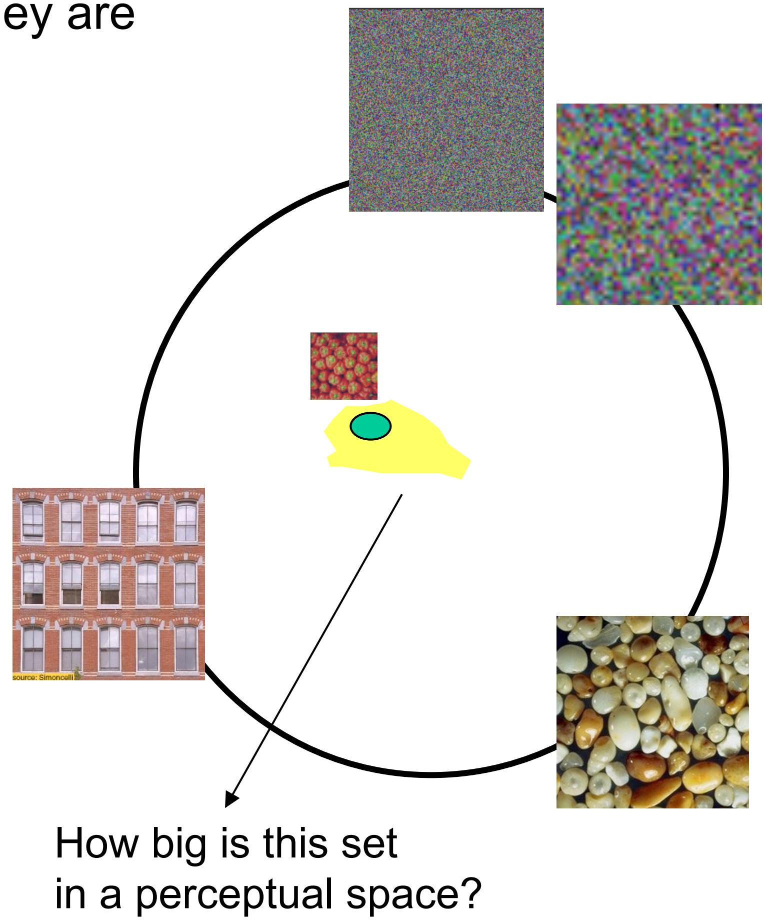


How big is this set in a pixels space?

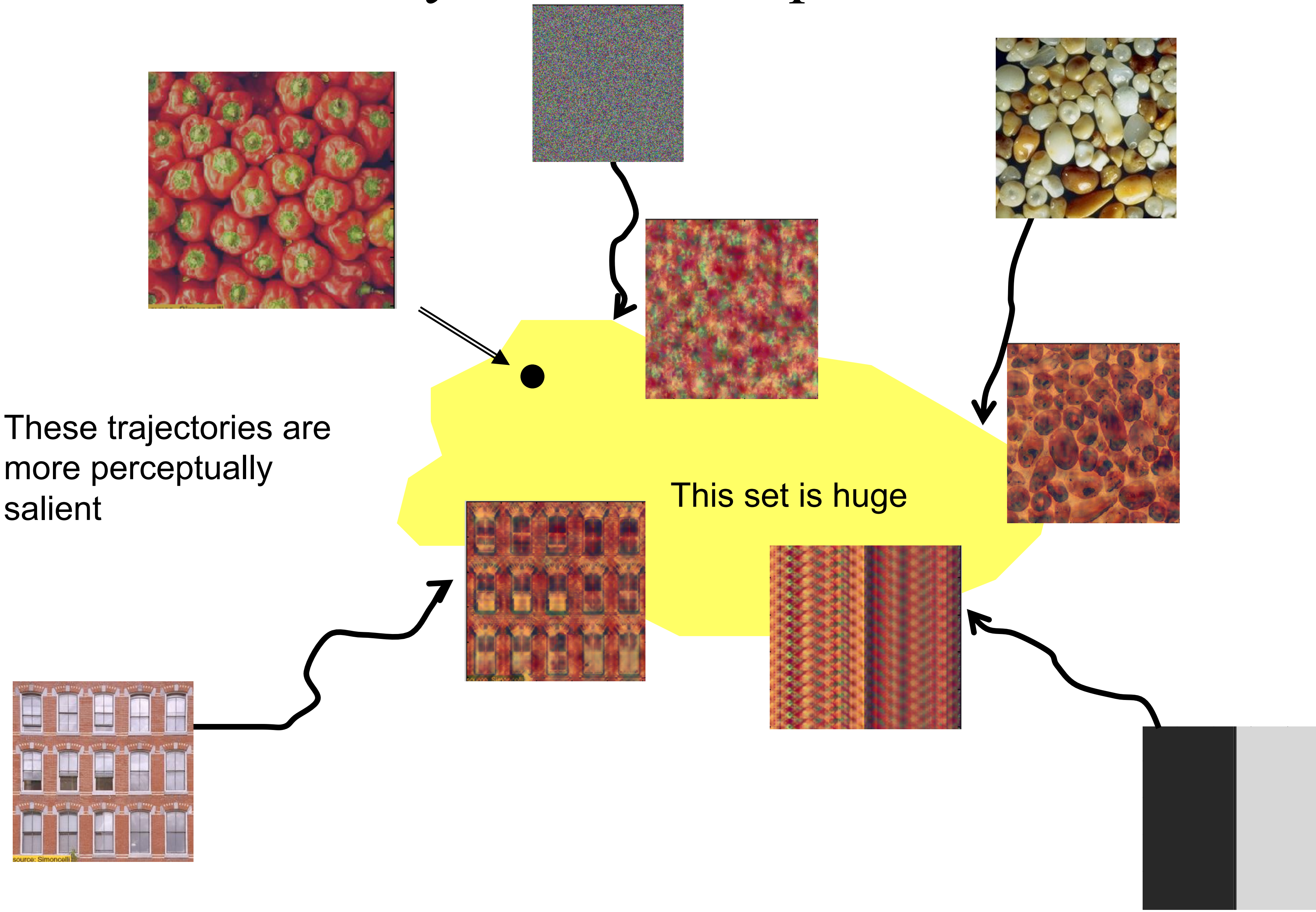
We need a space that is more perceptual

In a perceptual space all these noise images are very close. But in pixel space, they are very far away.

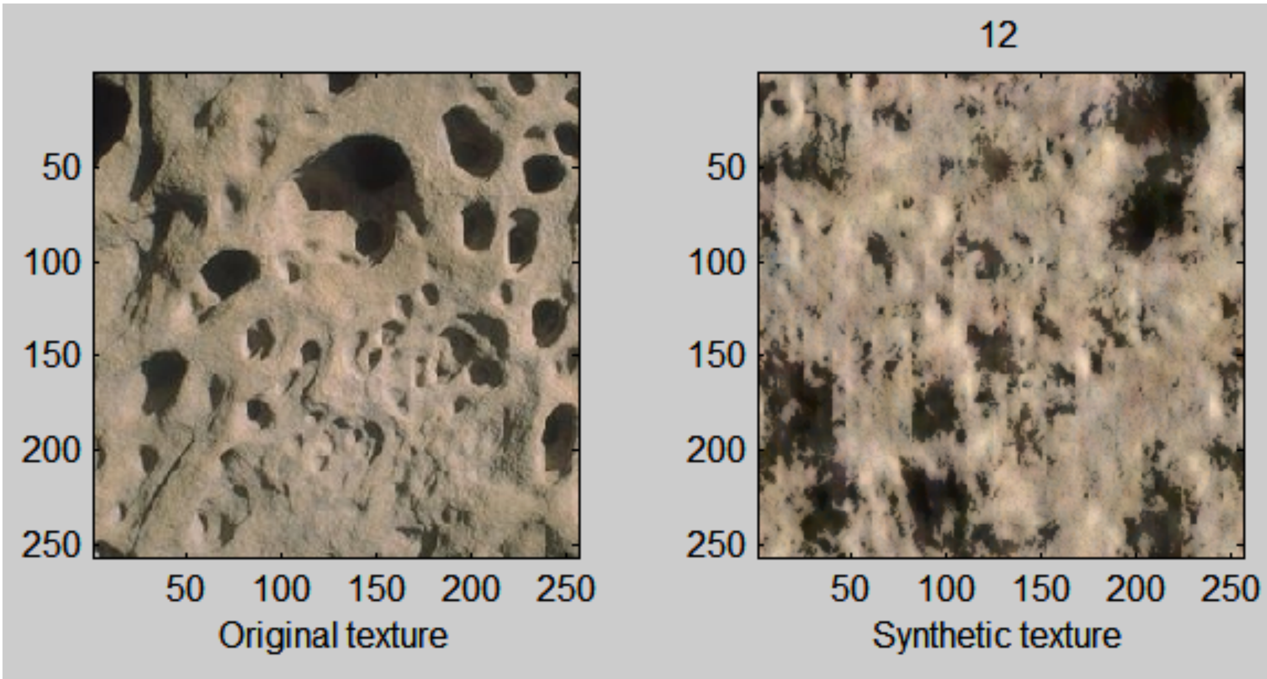




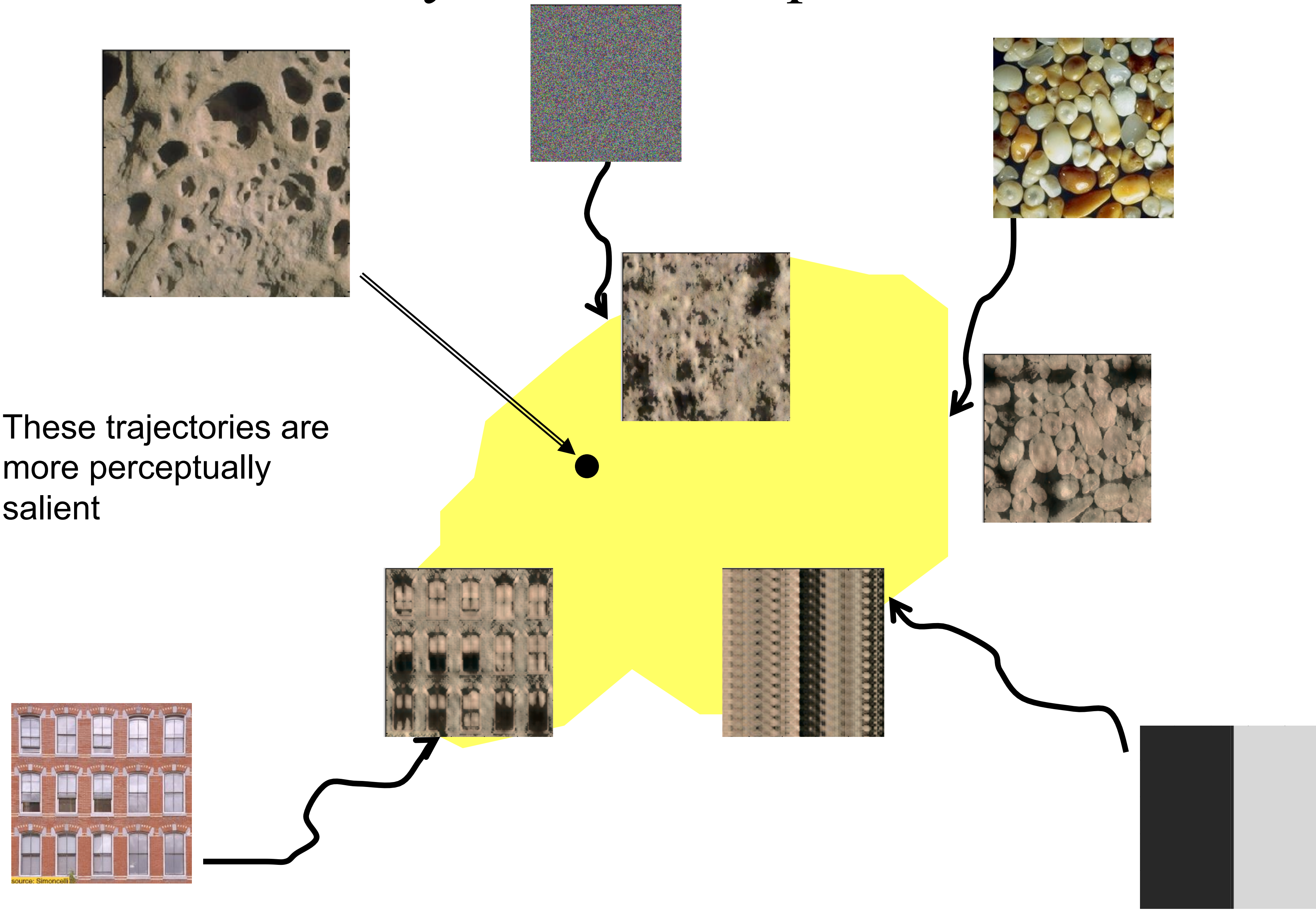
How to identify the set of equivalent textures?



How to identify the set of equivalent textures?



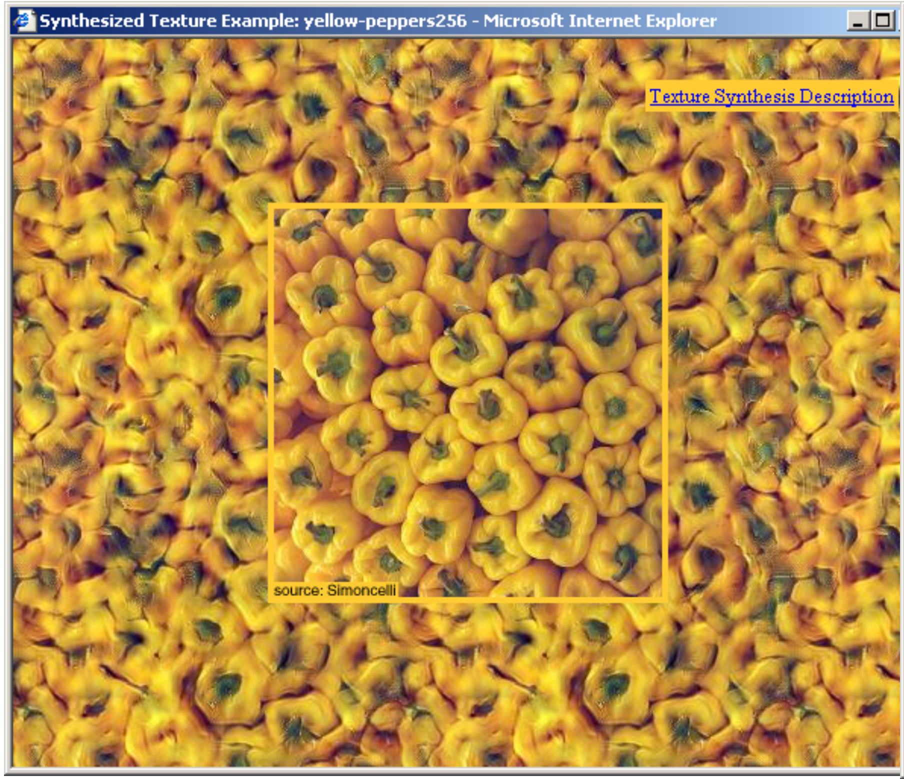
How to identify the set of equivalent textures?

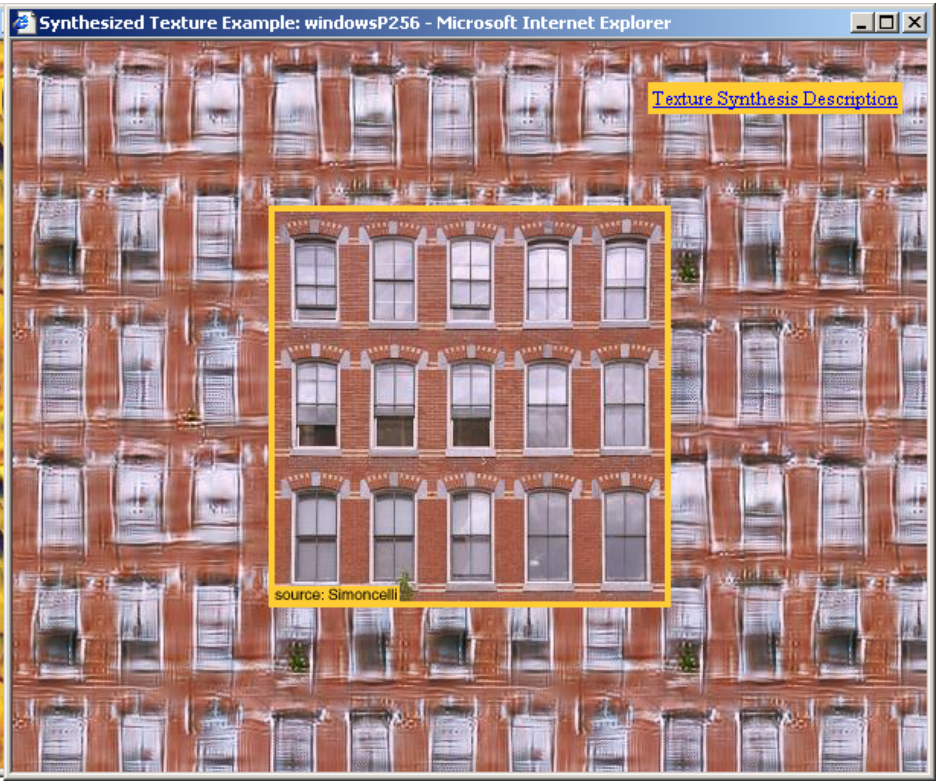


Portilla and Simoncelli

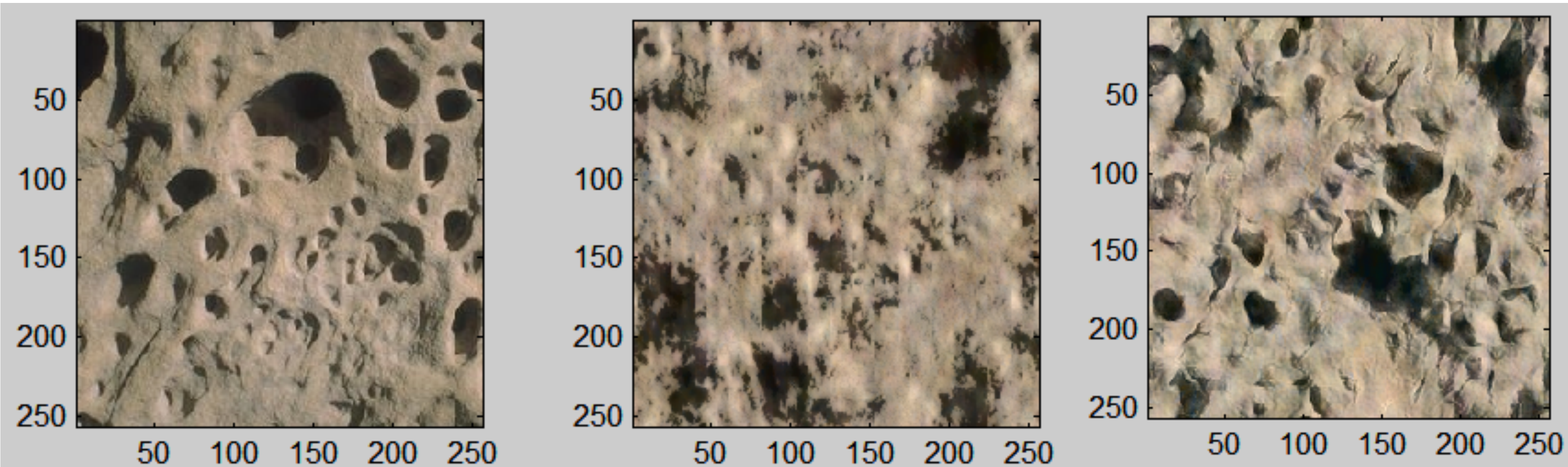
- Parametric representation, based on Gaussian scale mixture prior model for images.
- About 1000 numbers to describe a texture.
- Ok results; maybe as good as DeBonet.

Portilla and Simoncelli





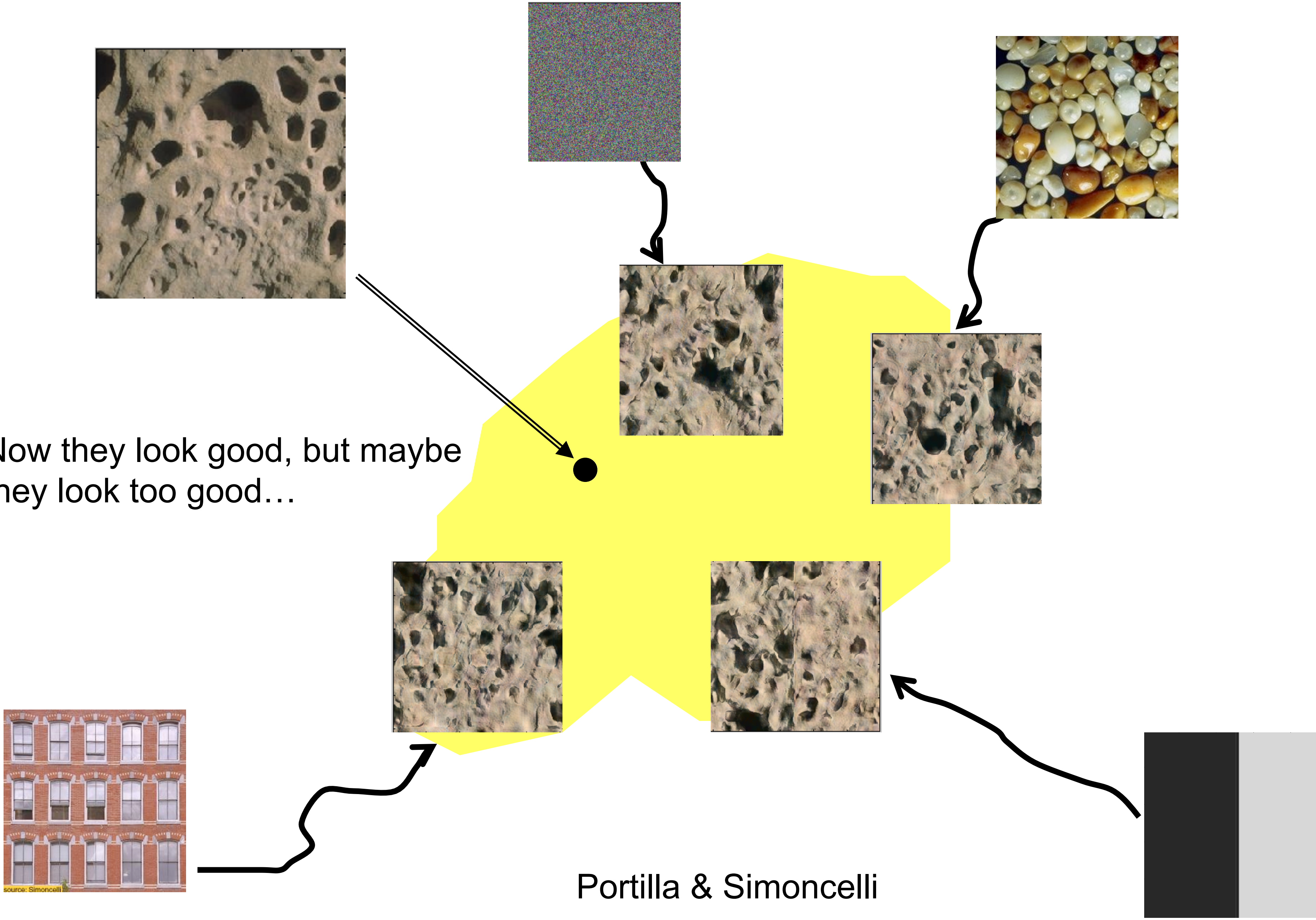
Portilla & Simoncelli



Heeger & Bergen

Portilla & Simoncelli

How to identify the set of equivalent textures?



Now they look good, but maybe they look too good…

Portilla & Simoncelli

Journal of Vision
November 19, 2009 vol. 9 no. 12

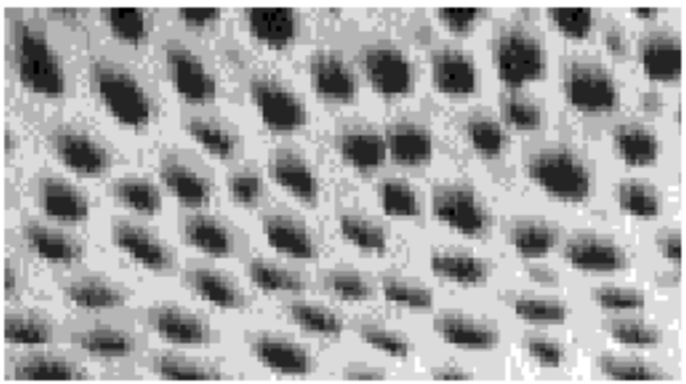
A summary-statistic representation in peripheral vision explains visual crowding

Benjamin Balas [1],
Lisa Nakano [2] and
Ruth Rosenholtz [3]

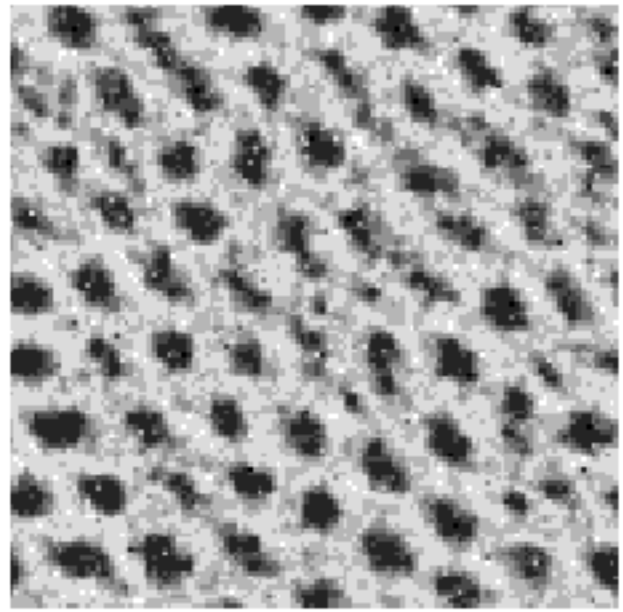
Zhu, Wu, & Mumford, 1998

- Principled approach. Based on an assumption of heavy-tailed distributions for an over-complete set of filters.
- Synthesis quality not great, but ok.

Zhu, Wu, & Mumford



a



b

- Cheetah Synthetic

De Bonet (and Viola)

SIGGRAPH 1997

Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images

Jeremy S. De Bonet –
Learning & Vision Group
Artificial Intelligence Laboratory
Massachusetts Institute of Technology

EMAIL: jsd@ai.mit.edu
HOMEPAGE: http://www.ai.mit.edu/_jsd

DeBonet

Learn: use filter conditional statistics across scale.

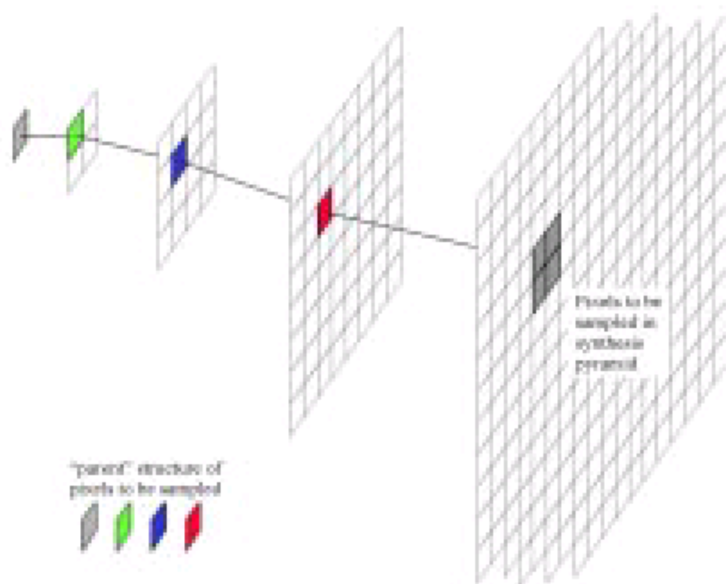


Figure 8: The distribution from which pixels in the synthesis pyramid are sampled is conditioned on the "parent" structure of those pixels. Each element of the parent structure contains a vector of the feature measurements at that location and scale.

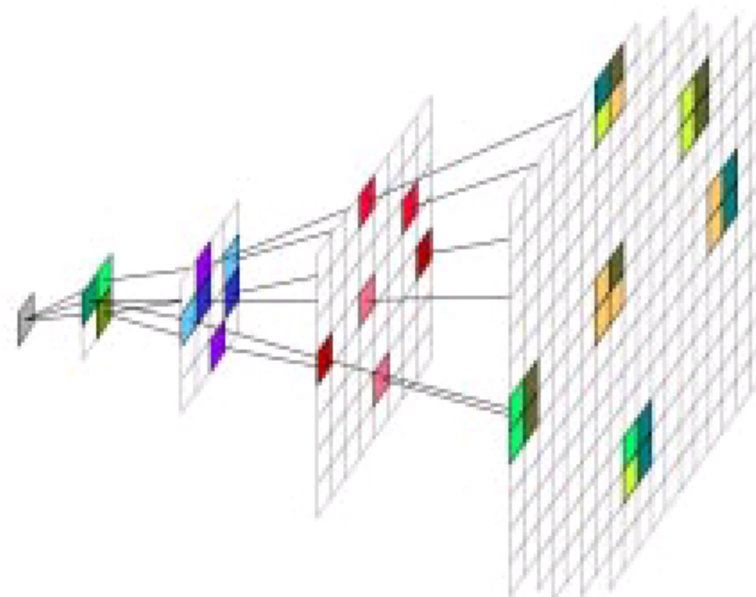
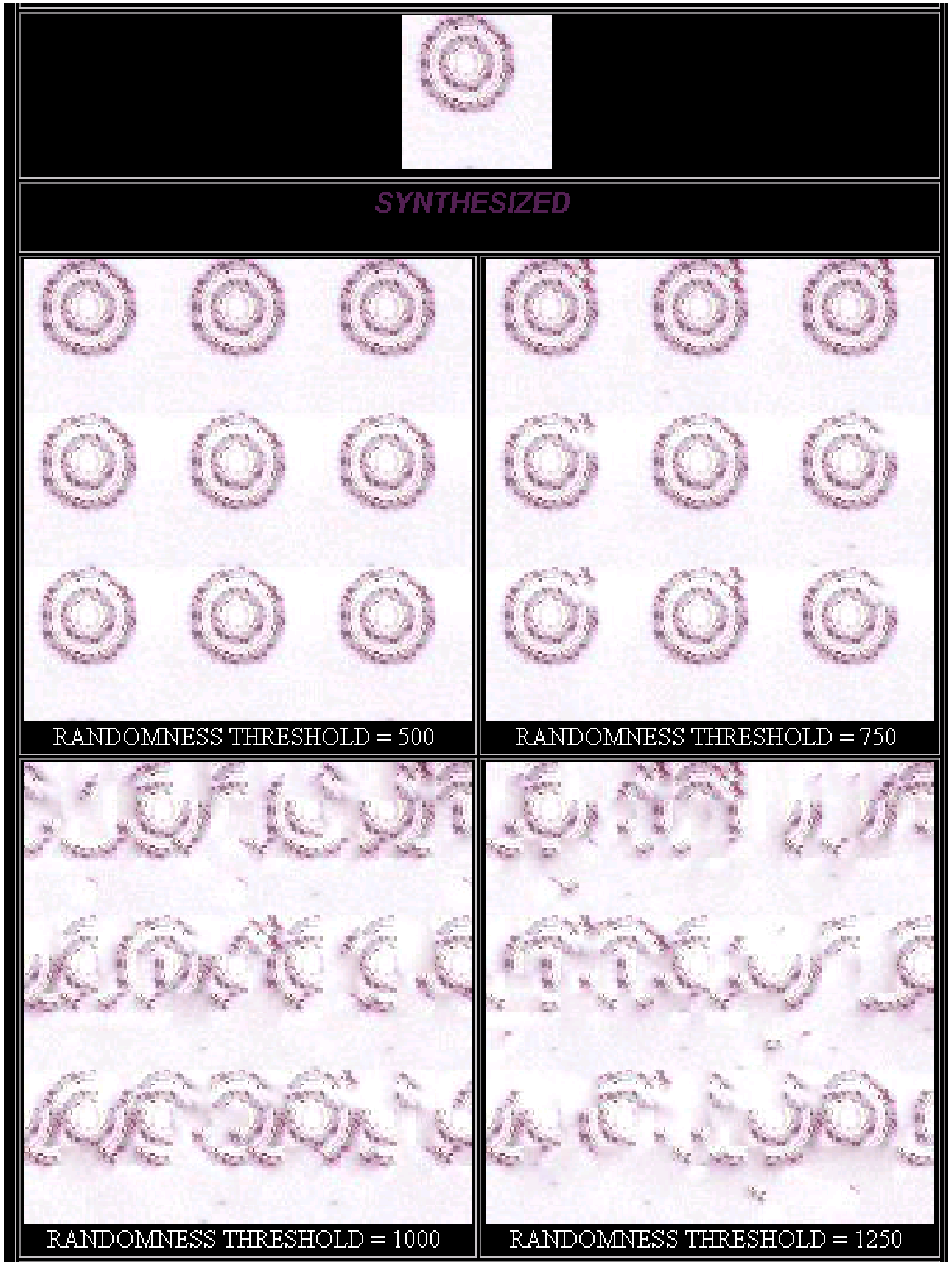
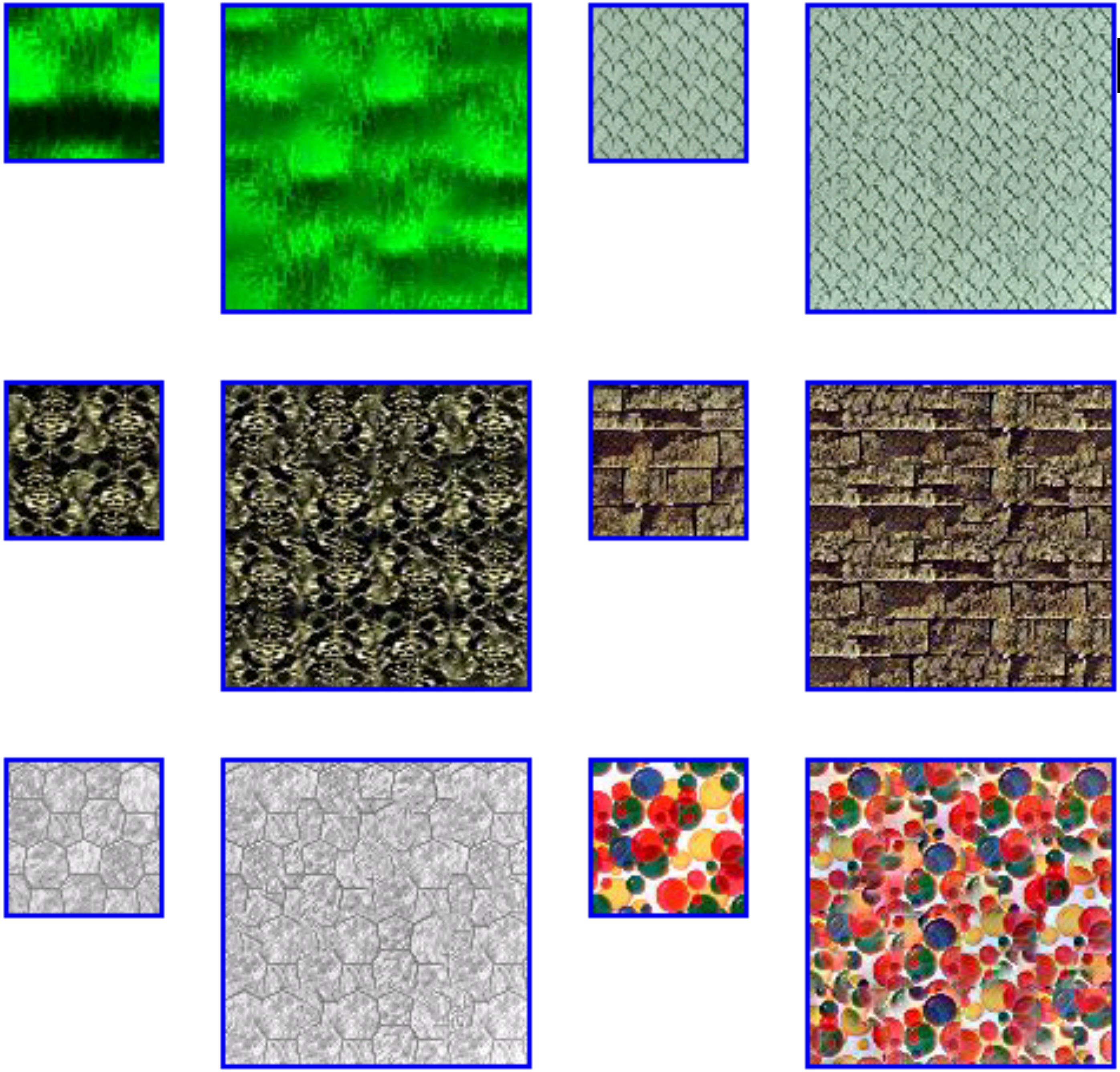


Figure 9: An input texture is decomposed to form an analysis pyramid, from which a new synthesis pyramid is sampled, conditioned on local features within the pyramids. A filter bank of local texture measures, based on psychophysical models, are used as features.

DeBonet


SYNTHESIZED
RANDOMNESS THRESHOLD = 500
RANDOMNESS THRESHOLD = 750
RANDOMNESS THRESHOLD = 1000
RANDOMNESS THRESHOLD = 1250

DeBonet



Two big families of models

1- Parametric models of filter outputs

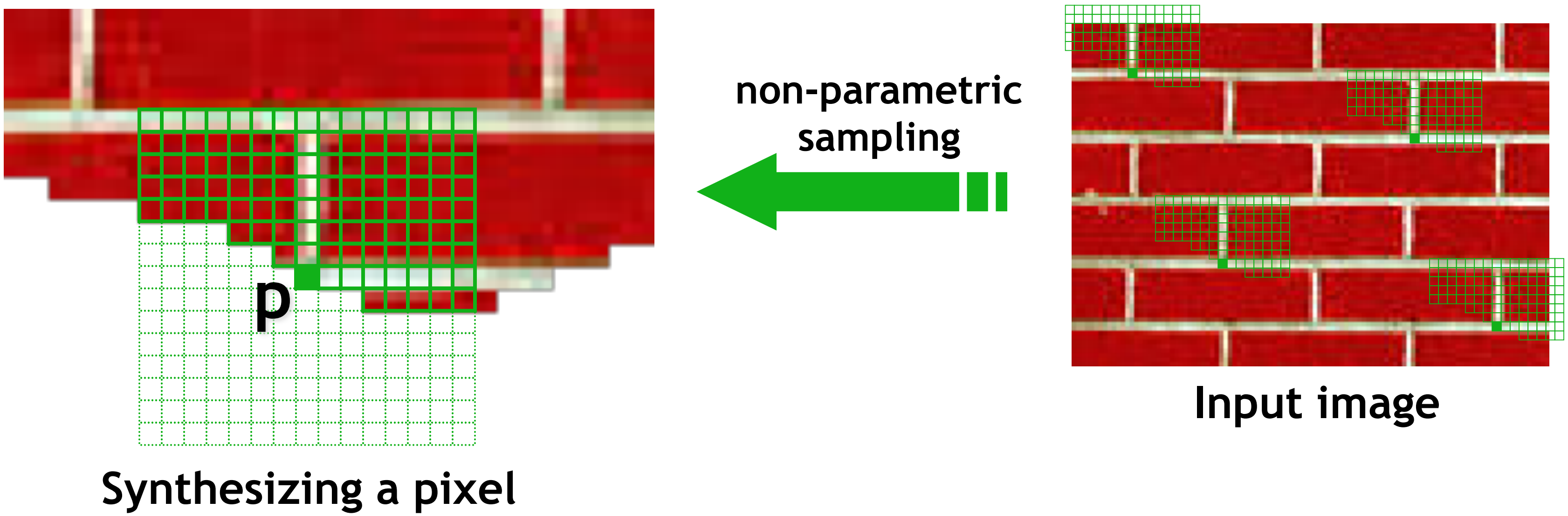
2- Example-based non-parametric models

IEEE International Conference on Computer Vision, Corfu, Greece, September 1999

Texture Synthesis by Non-parametric Sampling

Alexei A. Efros and Thomas K. Leung
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776, U.S.A.
{efros,leungt}@cs.berkeley.edu

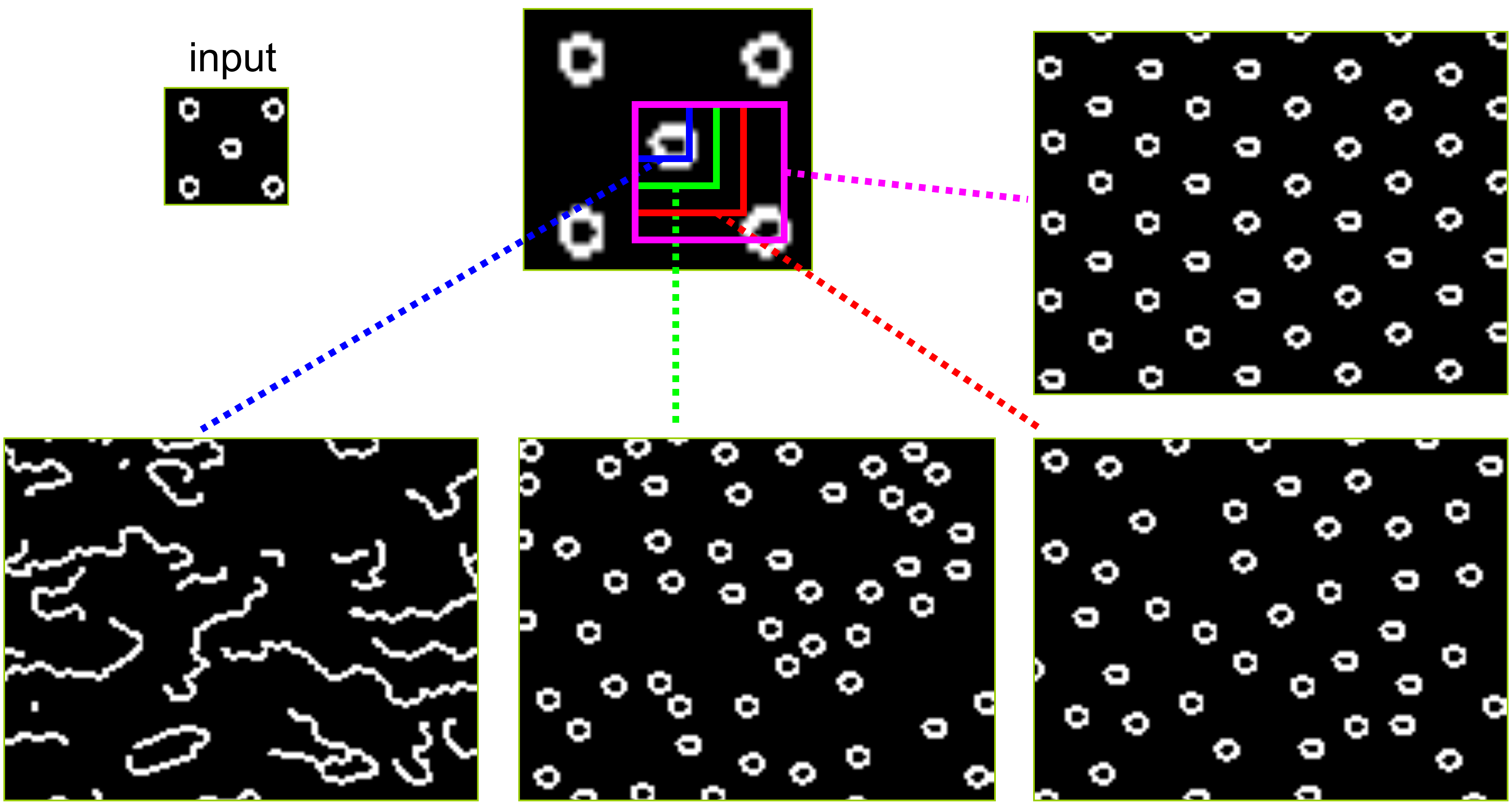
Efros & Leung Algorithm



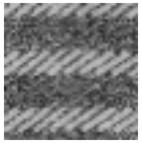
Assuming Markov property, compute P(**p**|N(**p**))

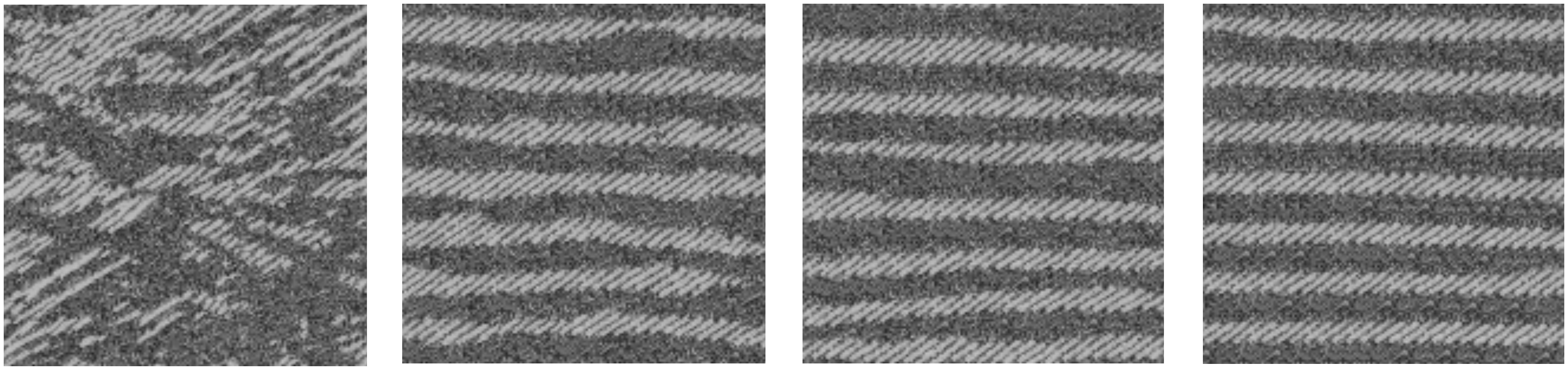
- Building explicit probability tables infeasible
- Instead, we *search the input image* for all similar neighborhoods — that's our pdf for **p**
- To sample from this pdf, just pick one match at random

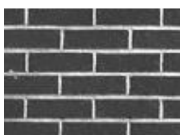
Neighborhood Window

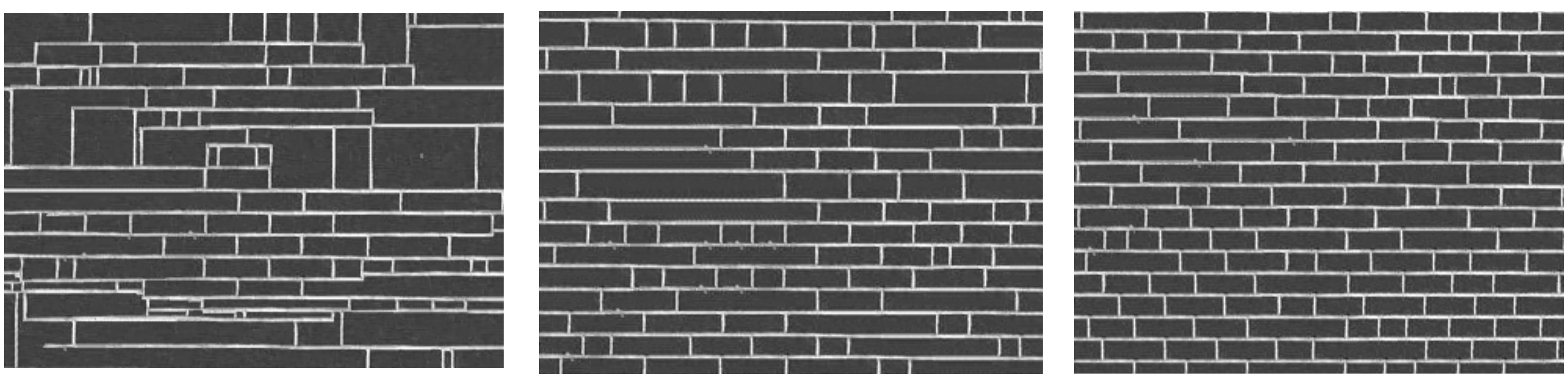


Varying Window Size





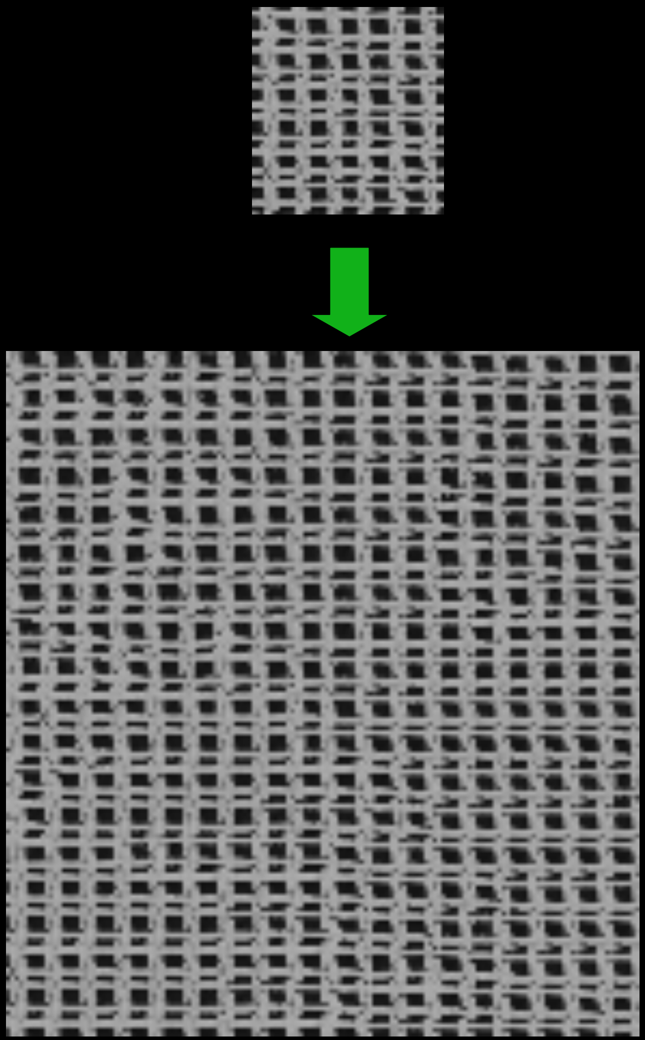




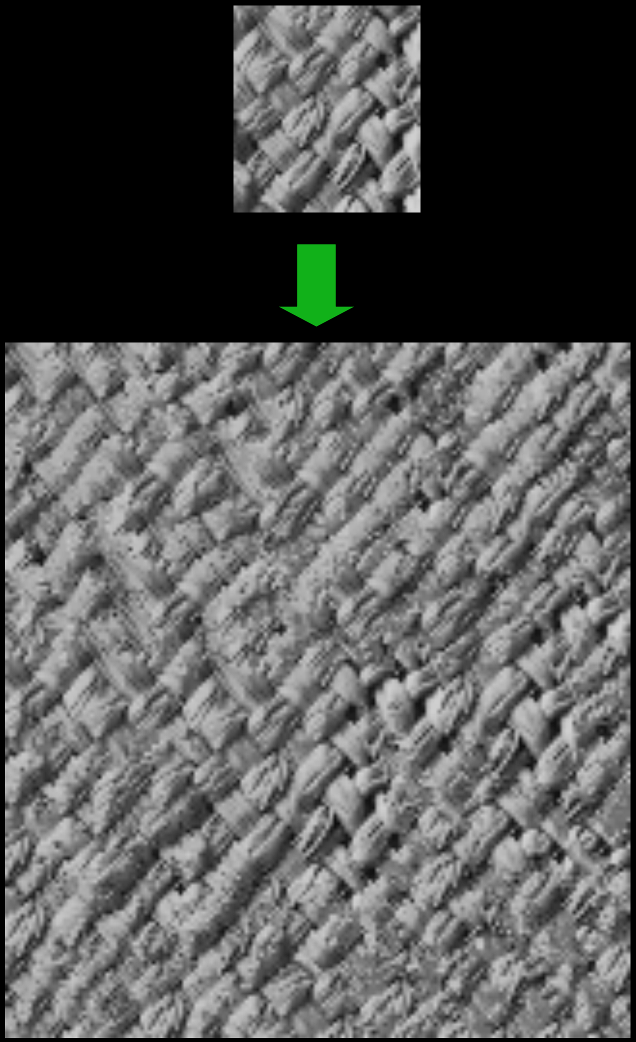
Increasing window size

Synthesis Results

french canvas

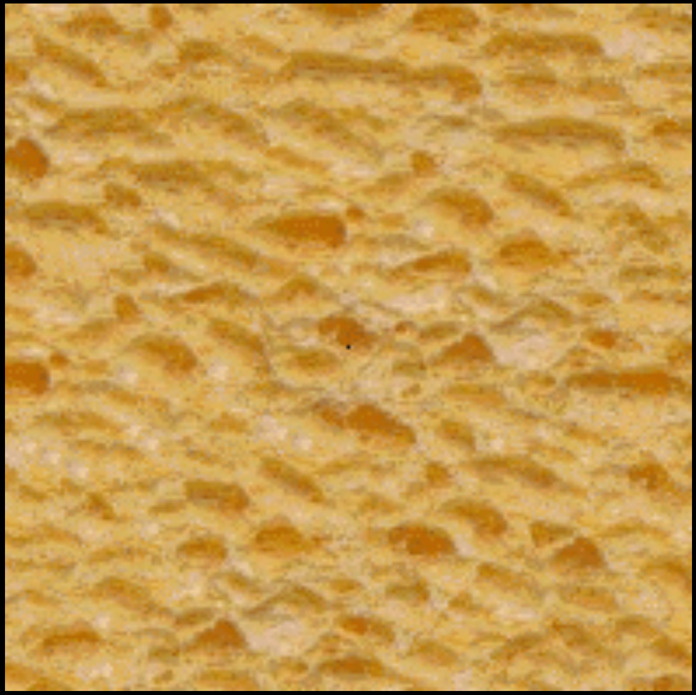


rafia weave

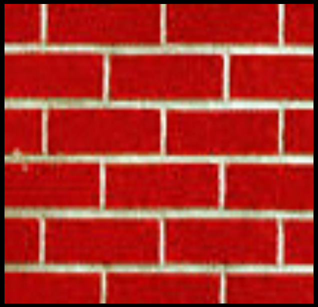


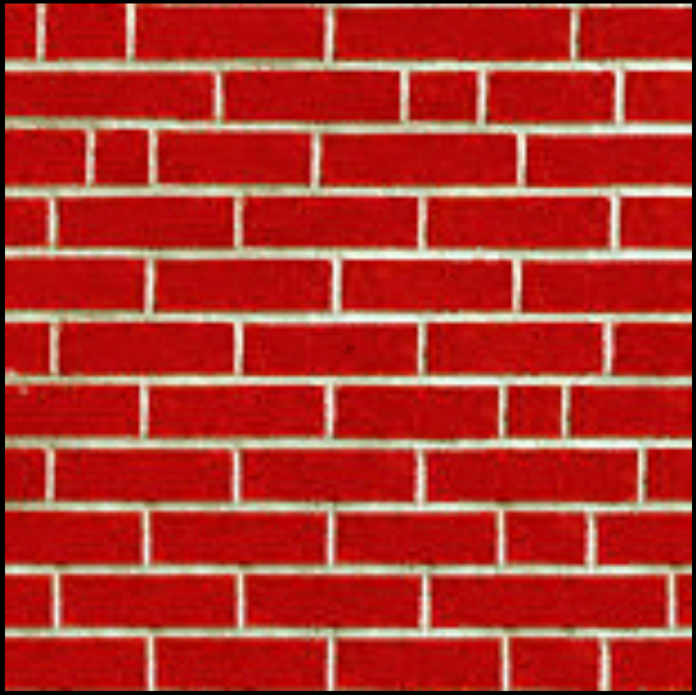
More Results

white bread

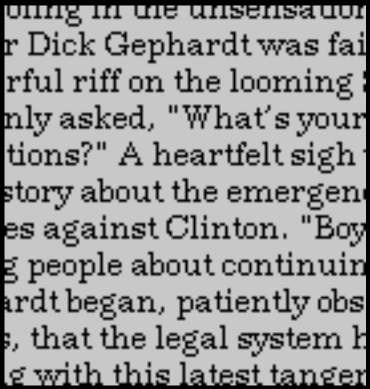


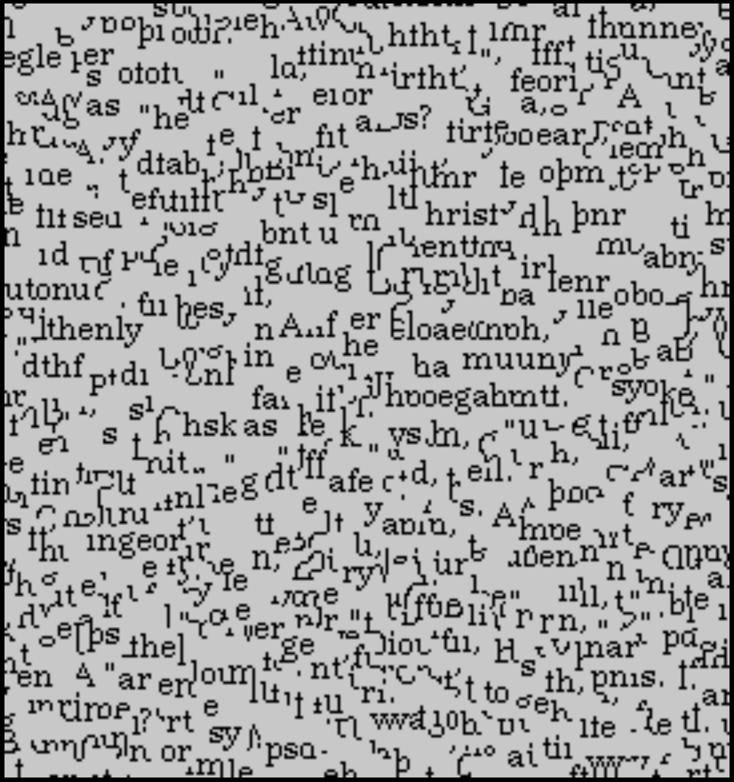
brick wall

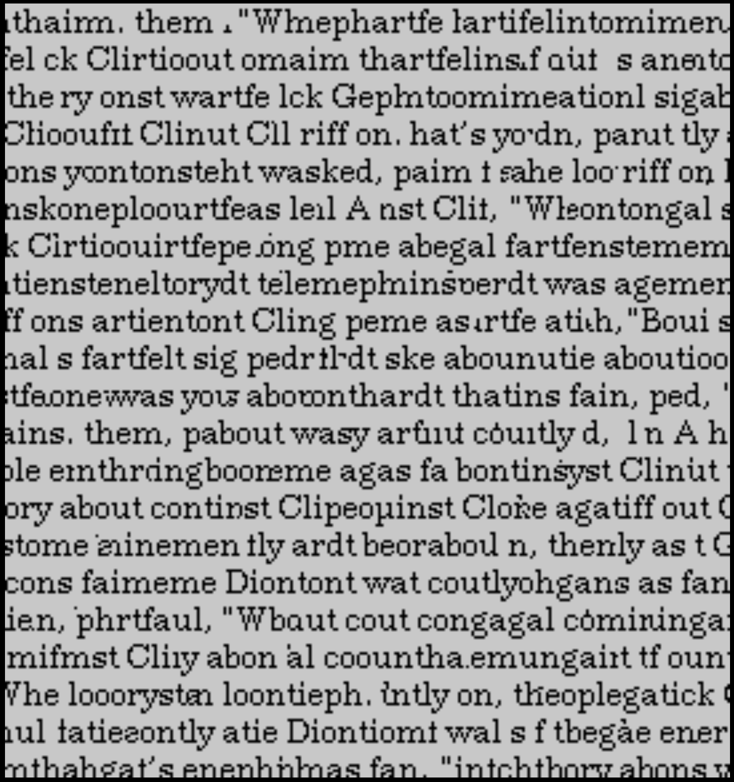




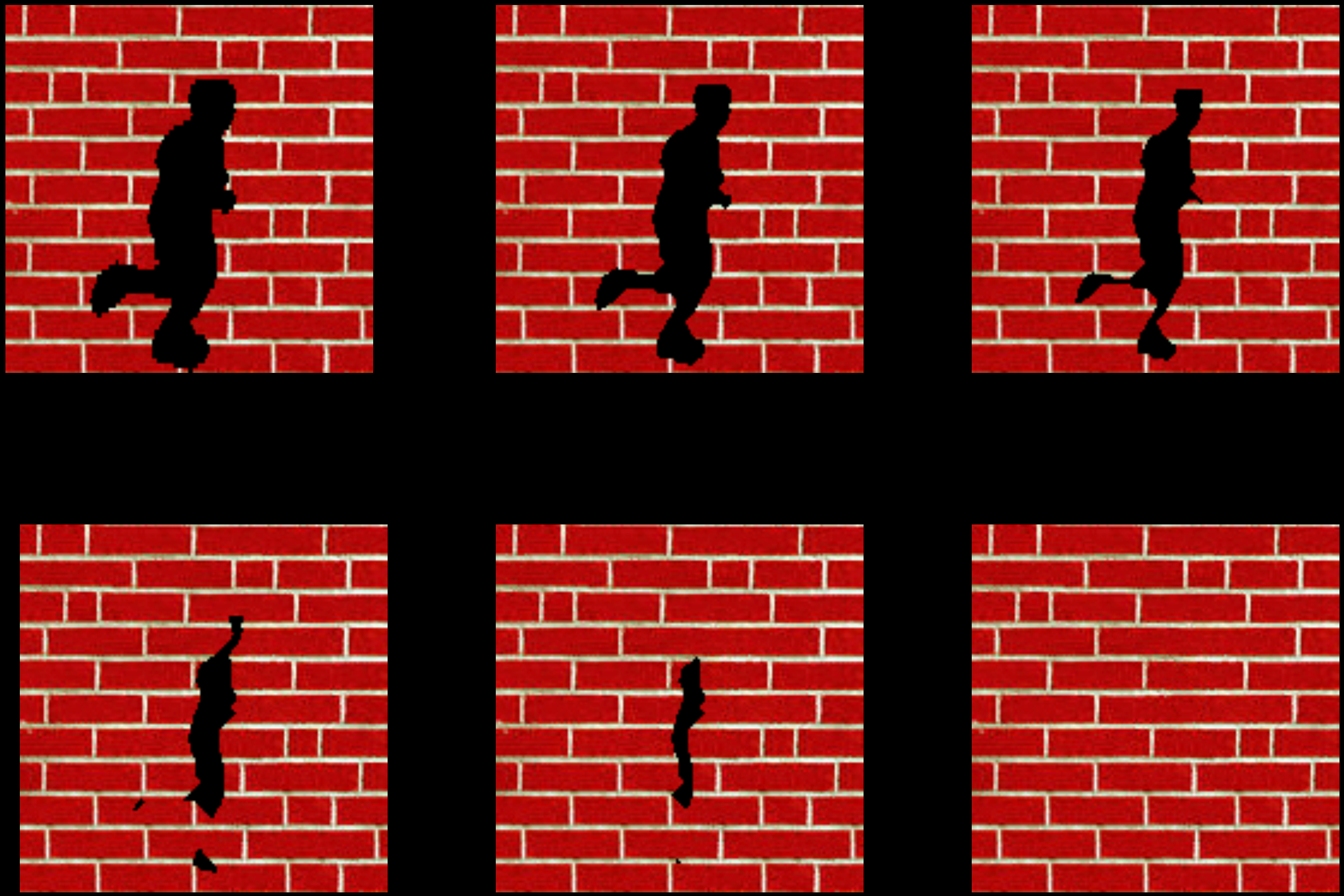
Homage to Shannon



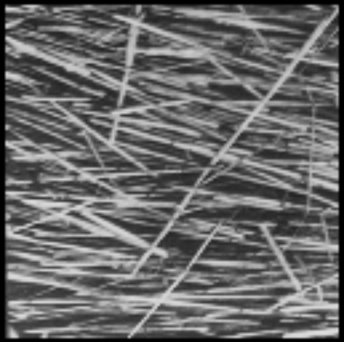




Hole Filling



Extrapolation



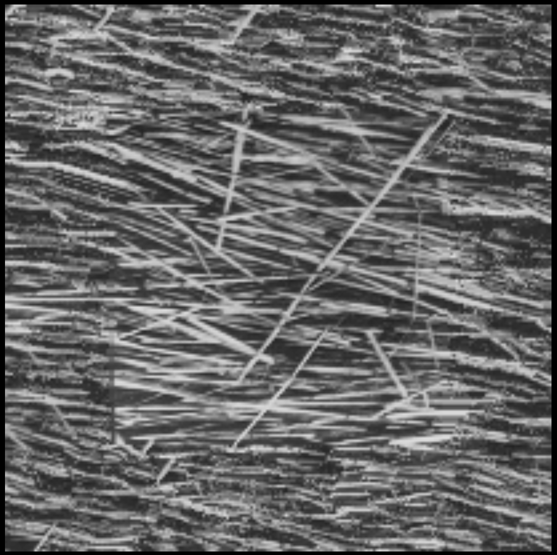
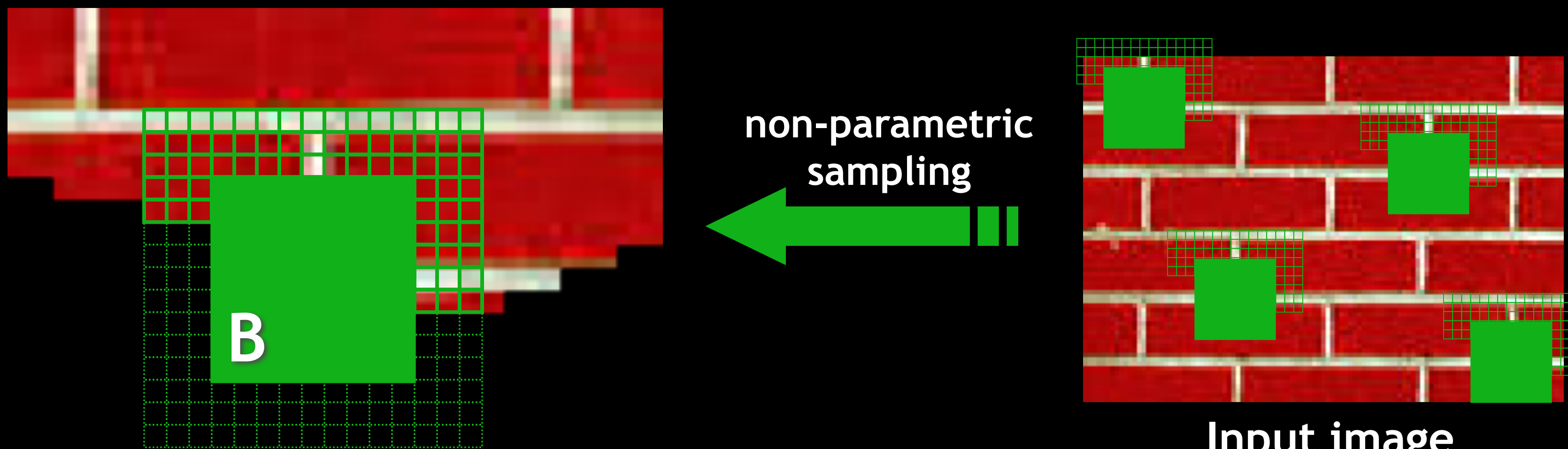


Image Quilting [Efros & Freeman]

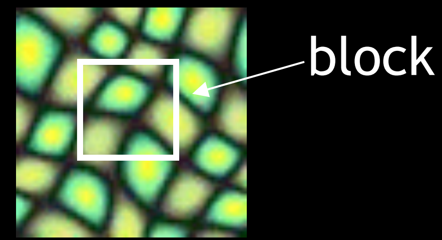


Synthesizing a block

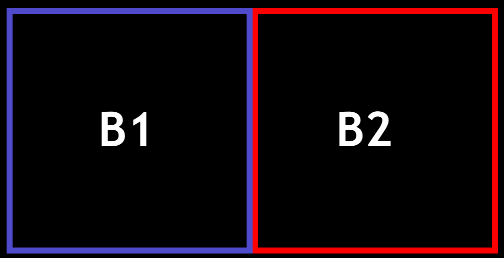
- Observation: neighbor pixels are highly correlated

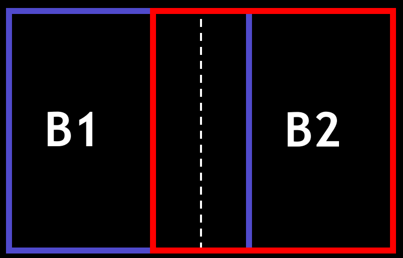
Idea: unit of synthesis = block

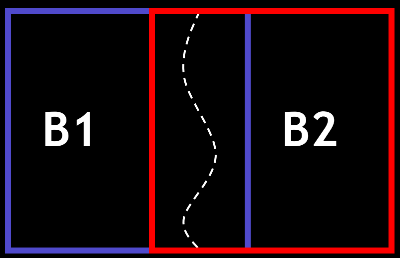
- Exactly the same but now we want P(**B**|N(**B**))
- Much faster: synthesize all pixels in a block at once
- Not the same as multi-scale!



Input texture



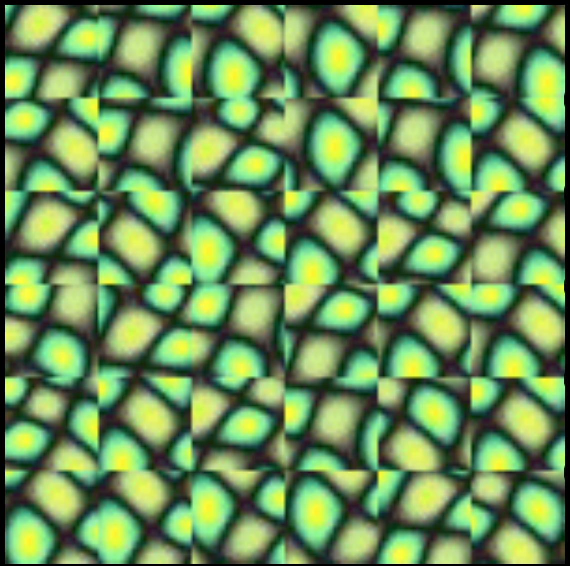


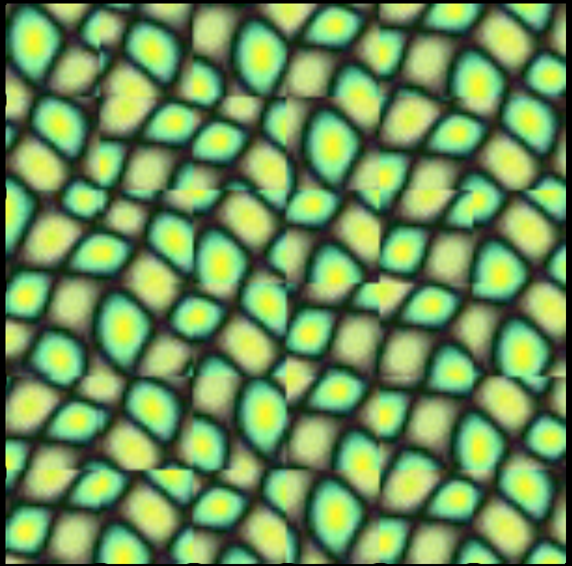


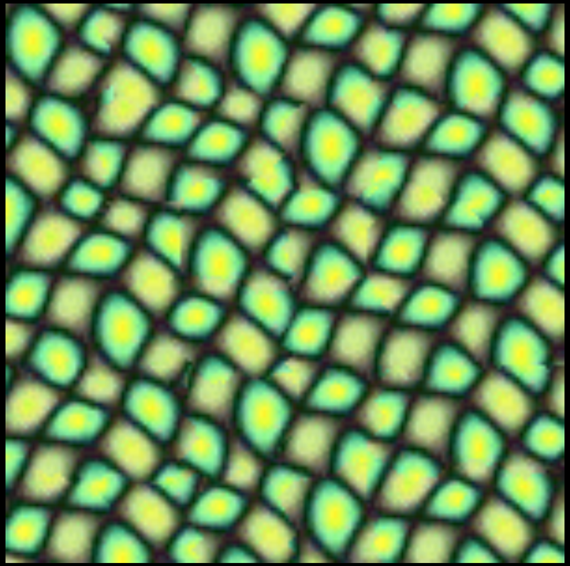
Random placement of blocks

Neighboring blocks constrained by overlap

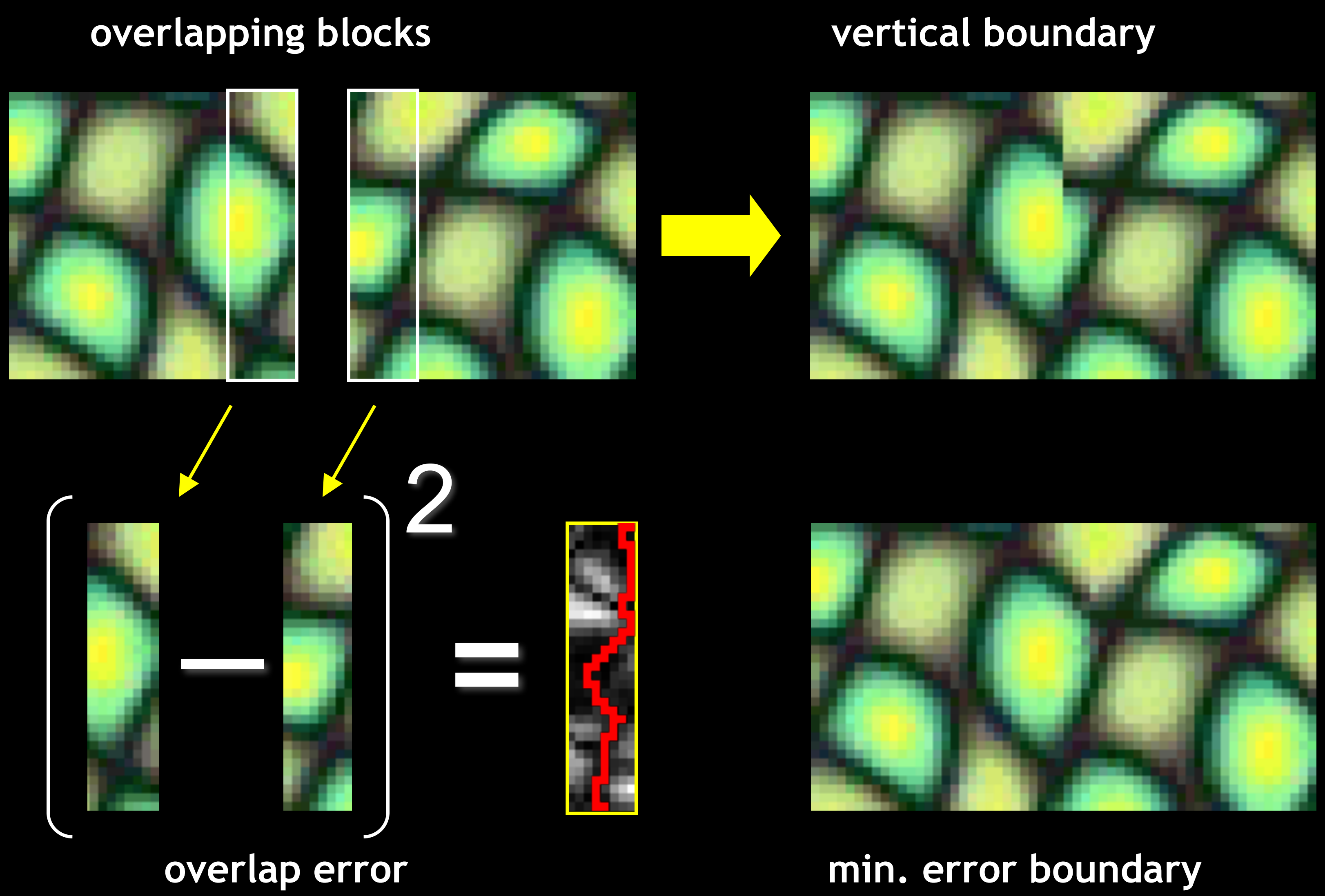
Minimal error boundary cut





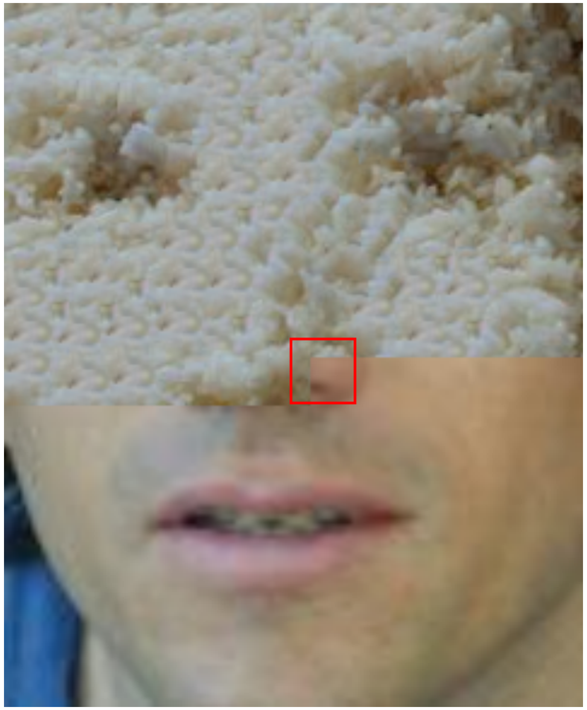


Minimal error boundary



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**

\+

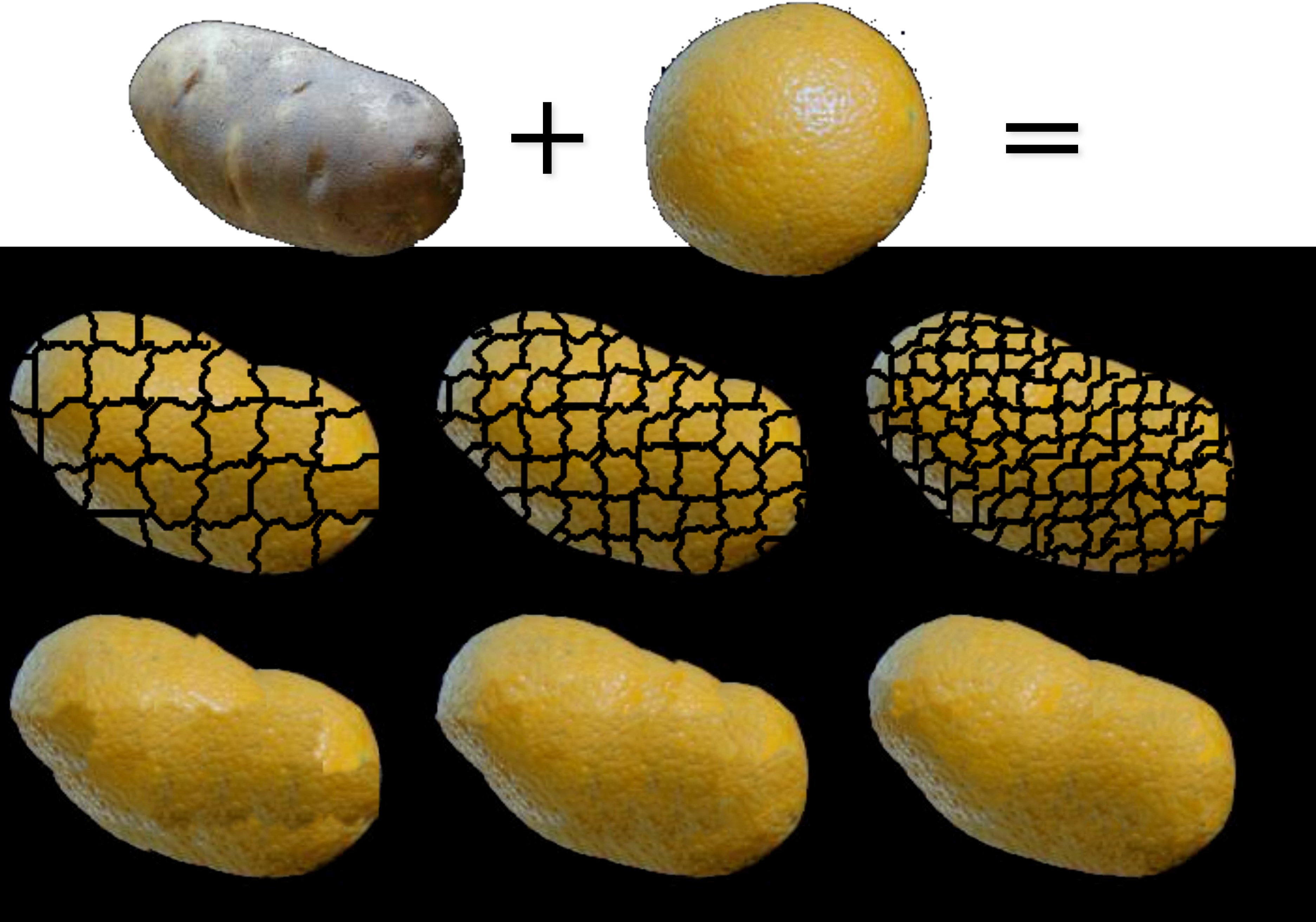
parmesan

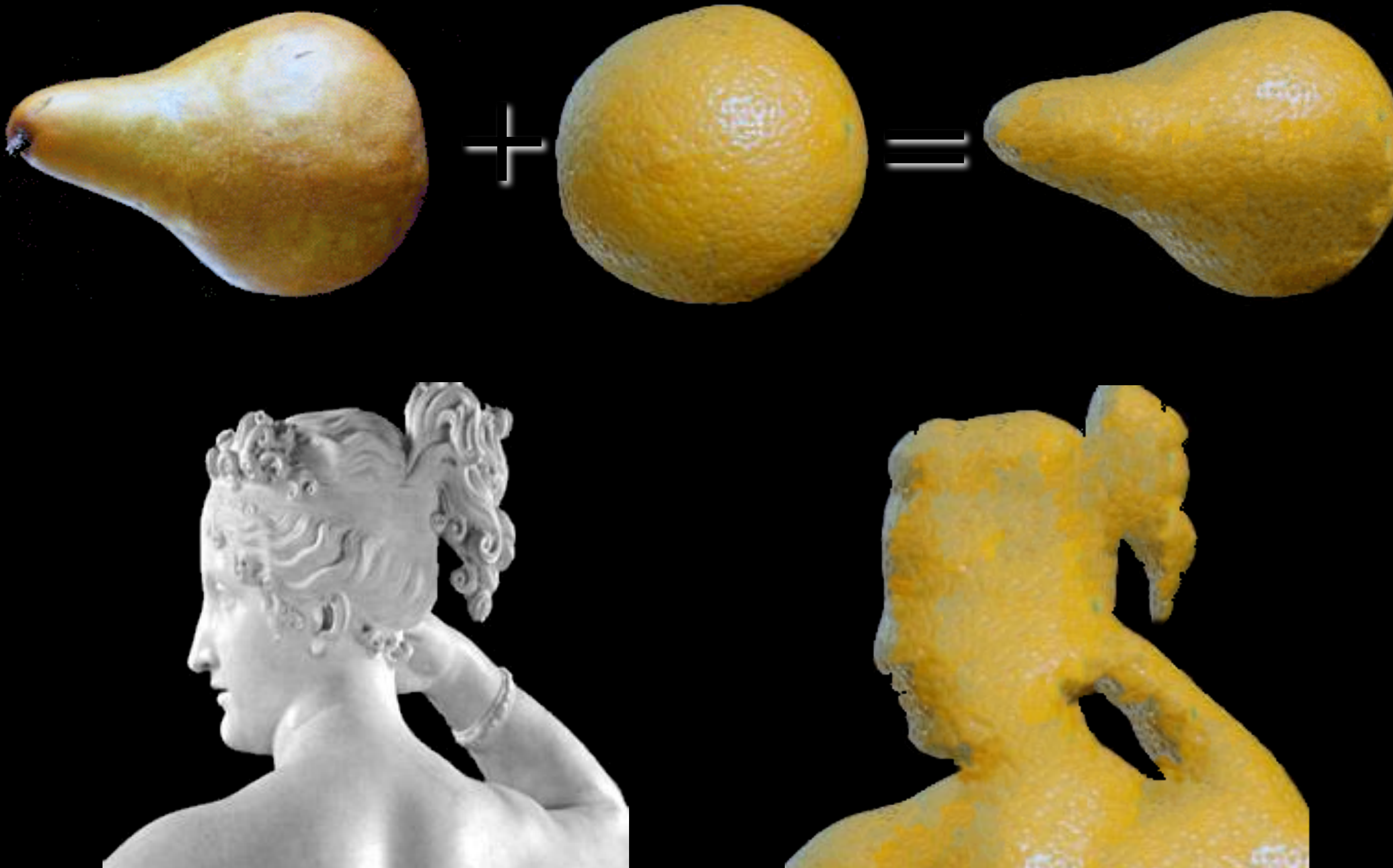
=

\+

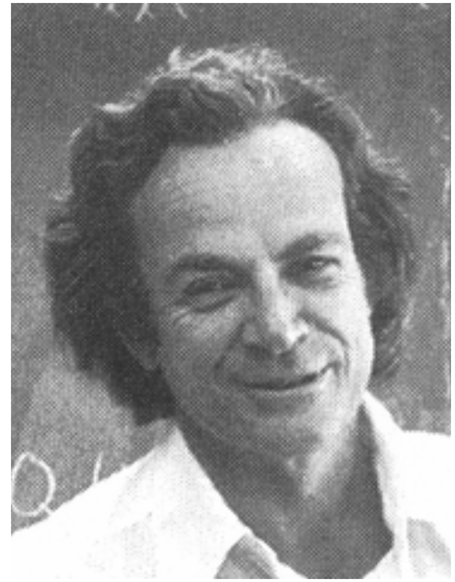
rice

=





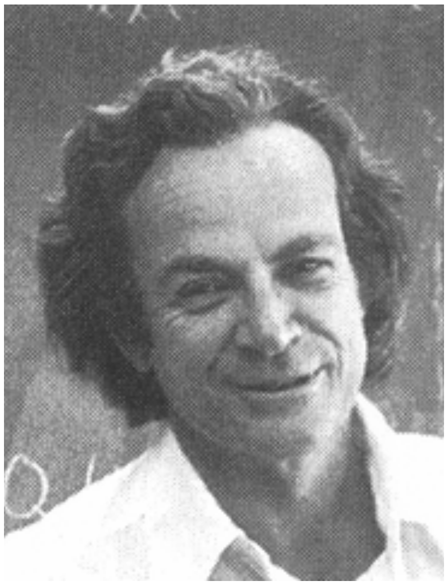
Source
texture



Target
image

Source
correspondence
image

Target
correspondence
image



+

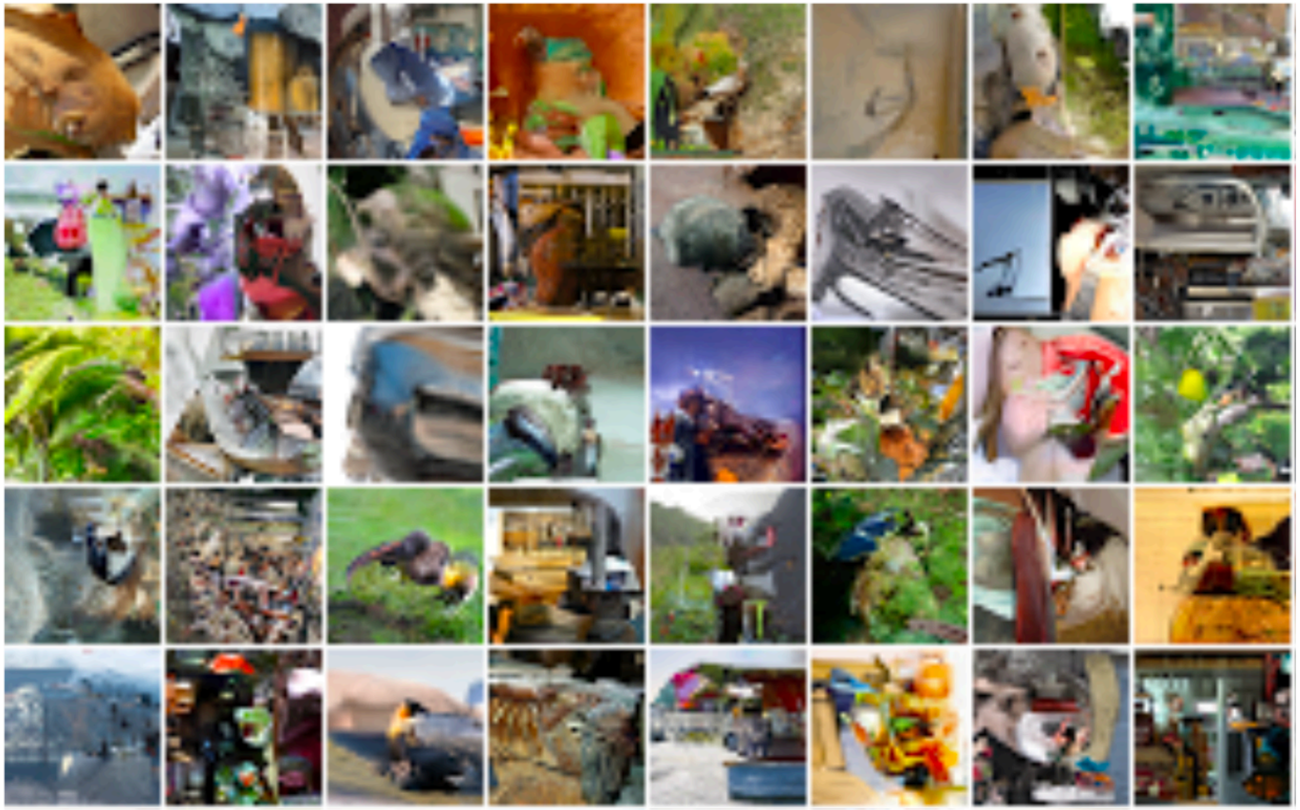
=

Pixel Recurrent Neural Networks

https://arxiv.org/pdf/1601.06759.pdf

Aäron van den Oord AVDNOORD@GOOGLE.COM
Nal Kalchbrenner NALK@GOOGLE.COM
Koray Kavukcuoglu KORAYK@GOOGLE.COM

Google DeepMind



samples from a model trained on ImageNet 32x32 (right) images.

Conditional Image Generation with PixelCNN Decoders

Aäron van den Oord
Google DeepMind
avdnoord@google.com

Nal Kalchbrenner
Google DeepMind
nalk@google.com

Oriol Vinyals
Google DeepMind
vinyals@google.com

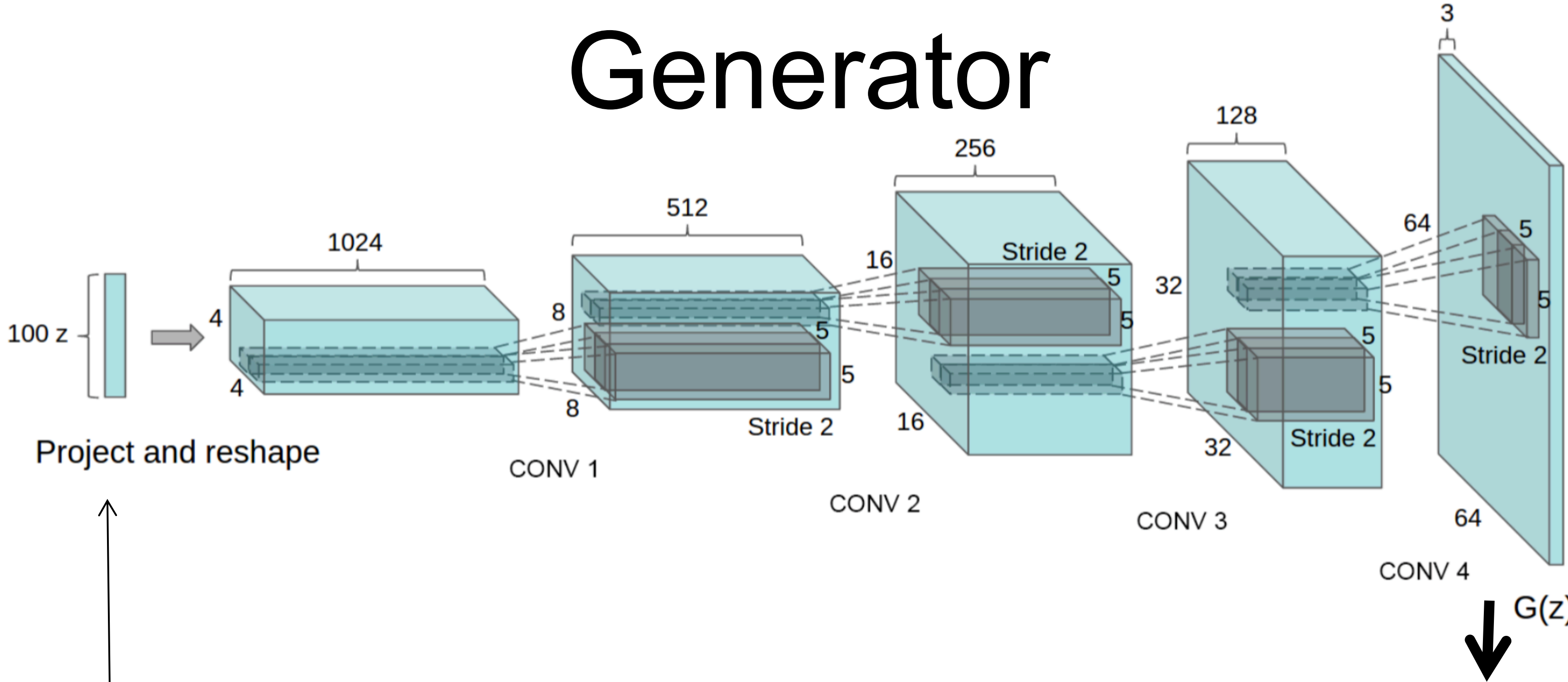
Lasse Espeholt
Google DeepMind
espeholt@google.com

Alex Graves
Google DeepMind
gravesa@google.com

Koray Kavukcuoglu
Google DeepMind
korayk@google.com

Figure 4: **Left**: source image. **Right**: new portraits generated from high-level latent representation.

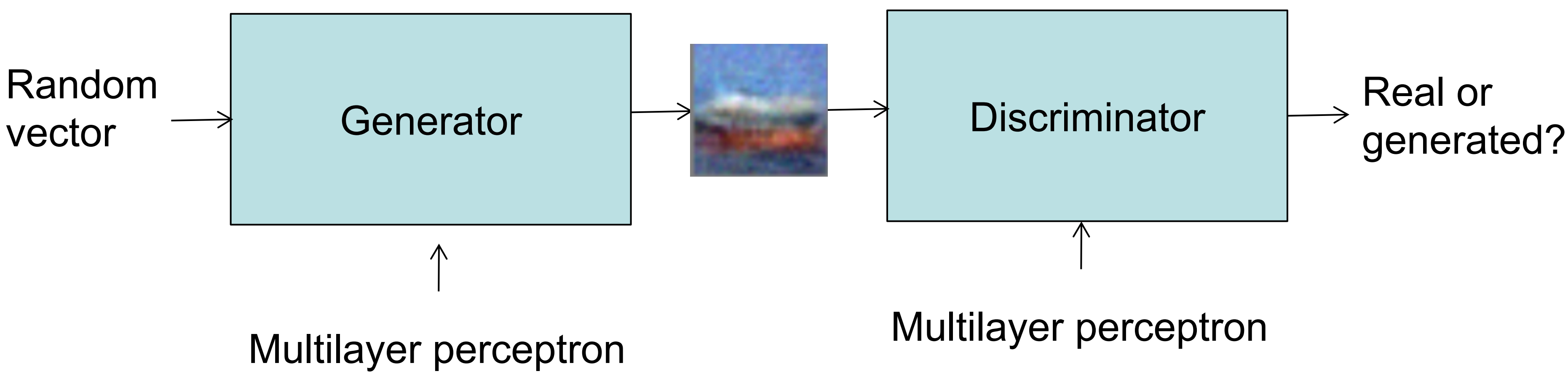
Generator



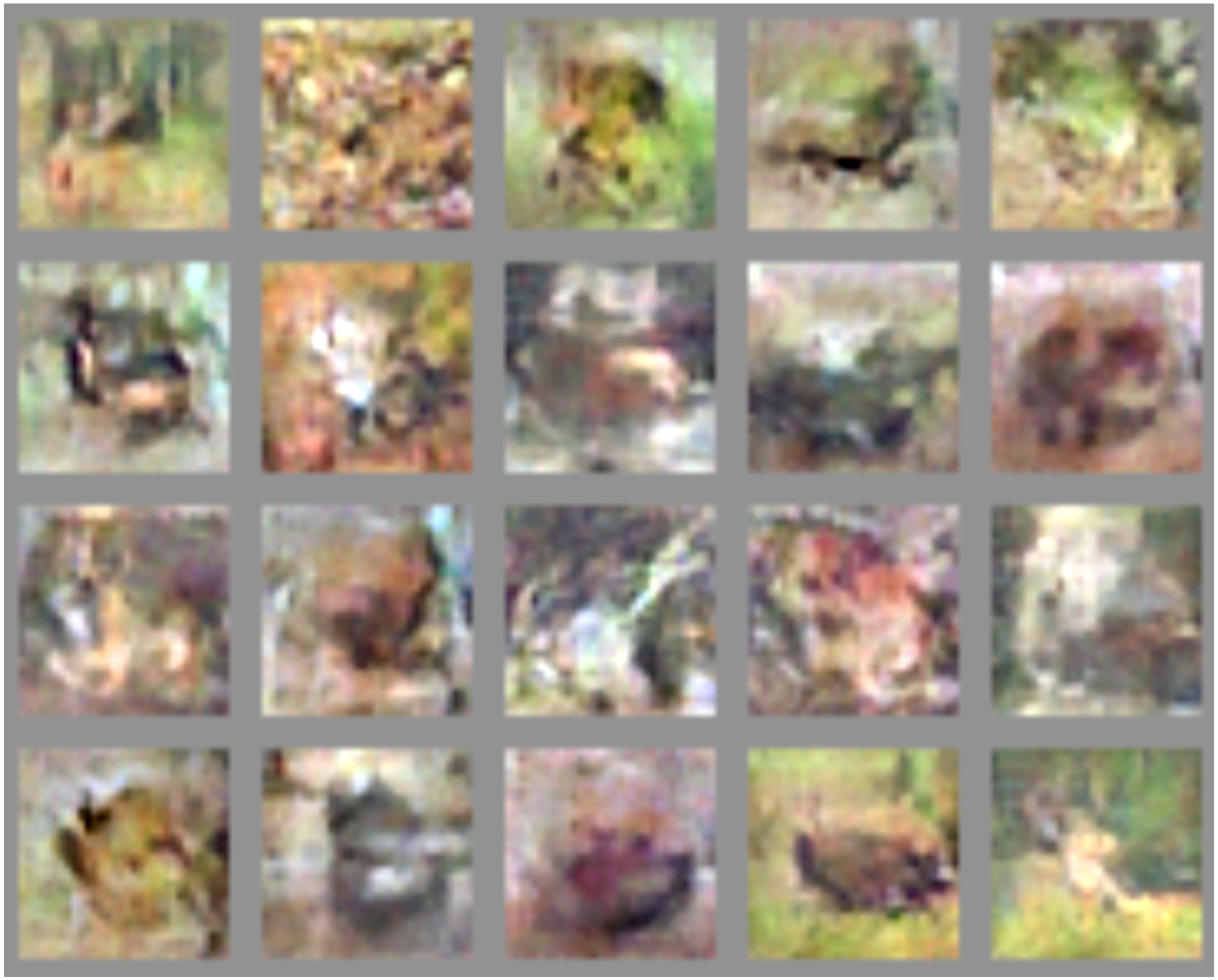
Random uniform vector (100 numbers)

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie,[*] Mehdi Mirza, Bing Xu, David Warde-Farley,
Sherjil Ozair,[†] Aaron Courville, Yoshua Bengio[‡]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7



Generated images



Trained with CIFAR-10

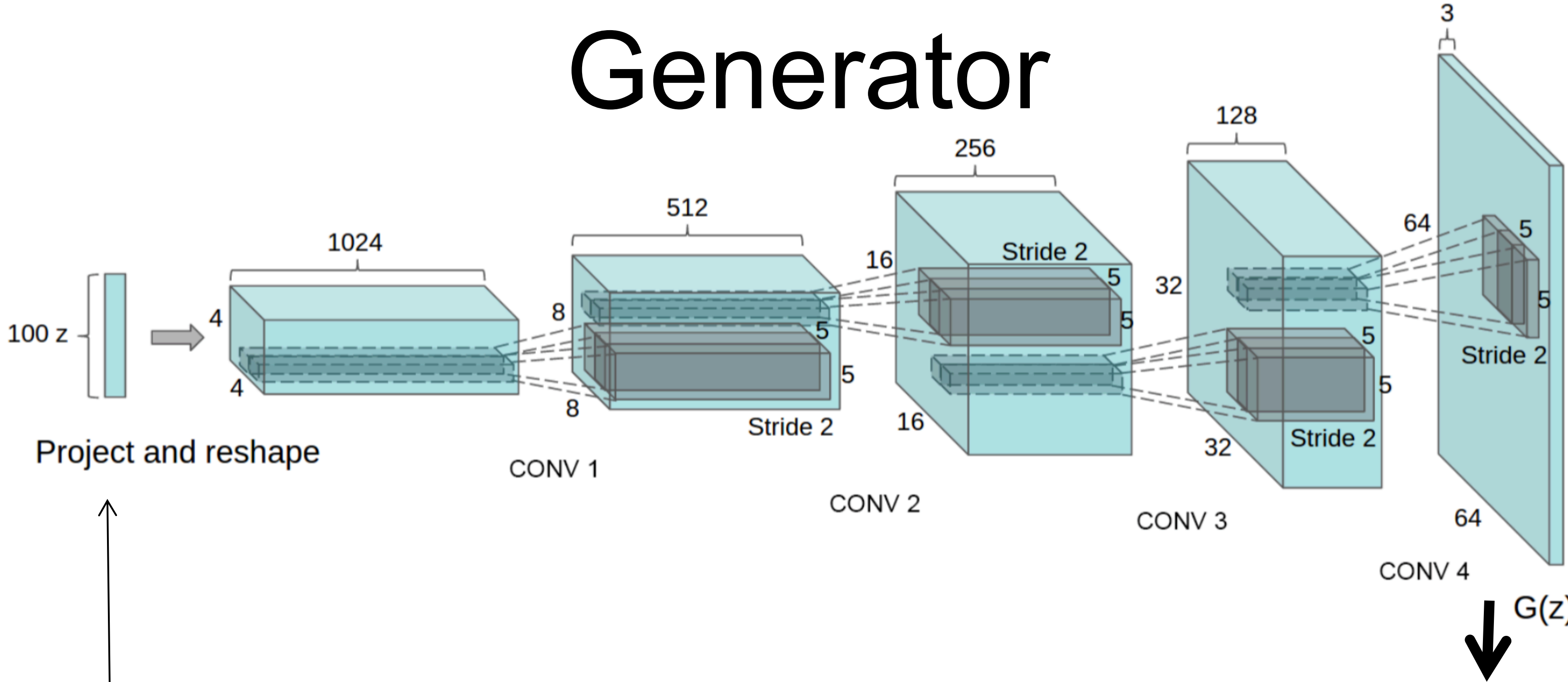
Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

Alec Radford & Luke Metz
indico Research
Boston, MA
{alec,luke}@indico.io

Soumith Chintala
Facebook AI Research
New York, NY
soumith@fb.com

Introduced a form of ConvNet more stable under adversarial training than previous attempts.

Generator



Random uniform
vector (100 numbers)

Synthesizing the preferred inputs for neurons in neural networks via deep generator networks

Anh Nguyen
anguyen8@uwyo.edu

Alexey Dosovitskiy
dosovits@cs.uni-freiburg.de

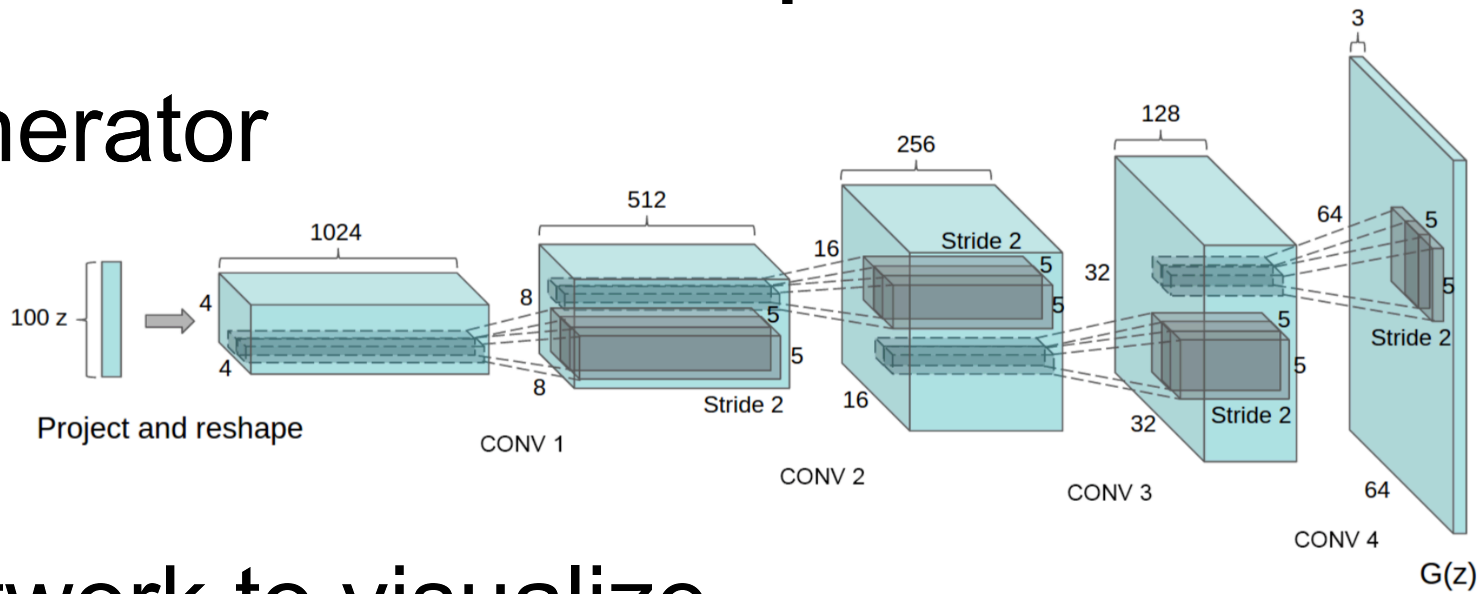
Jason Yosinski
jason@geometricintelligence.com

Thomas Brox
brox@cs.uni-freiburg.de

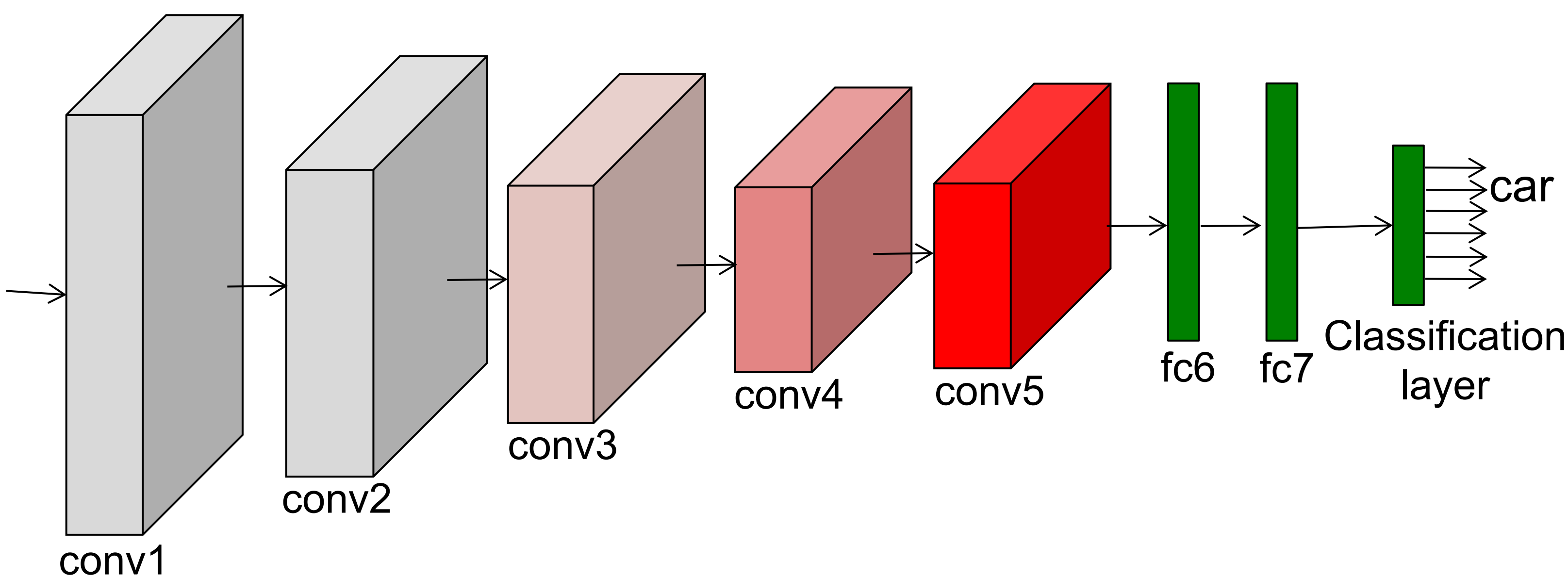
Jeff Clune
jeffclune@uwyo.edu

Two components

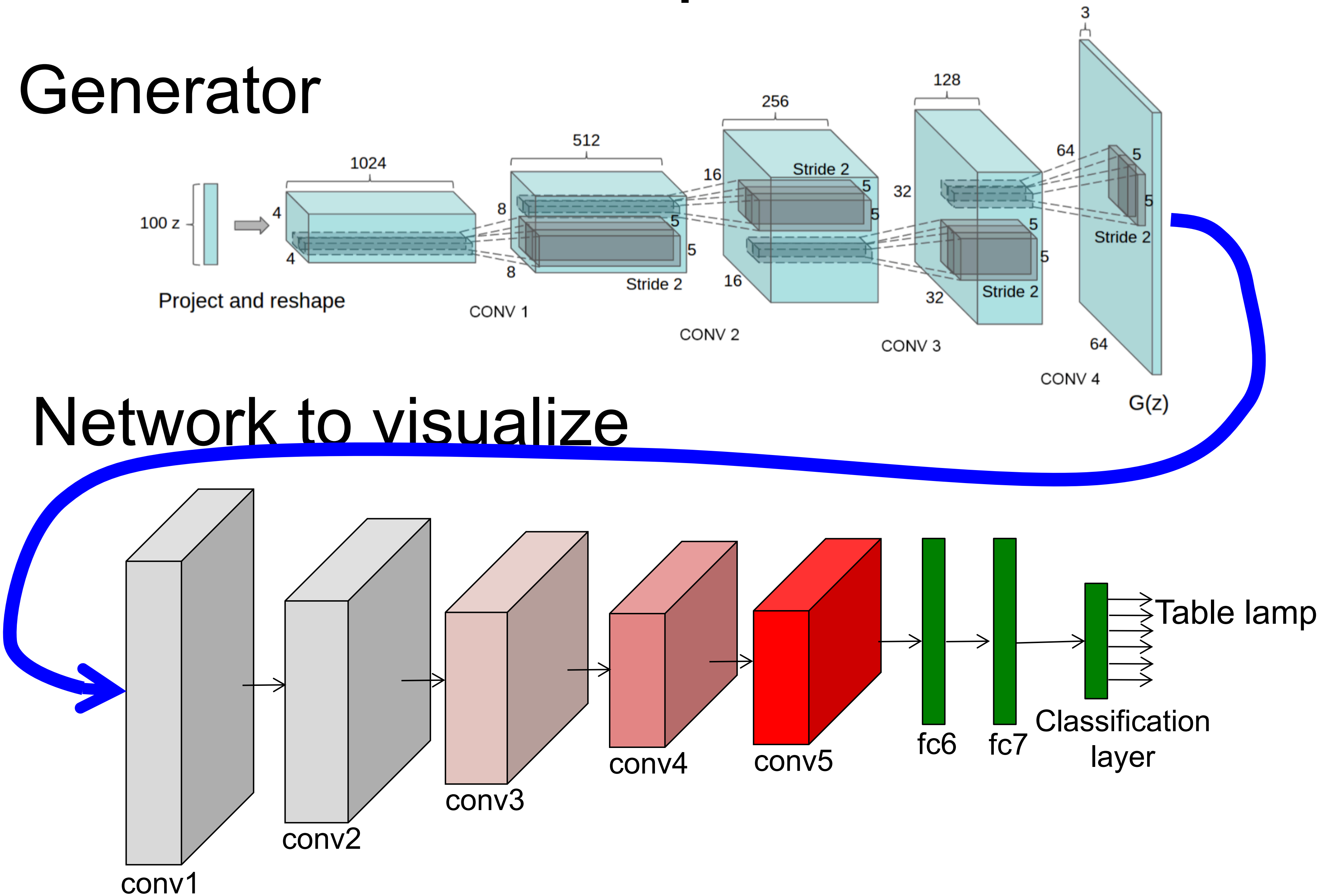
Generator



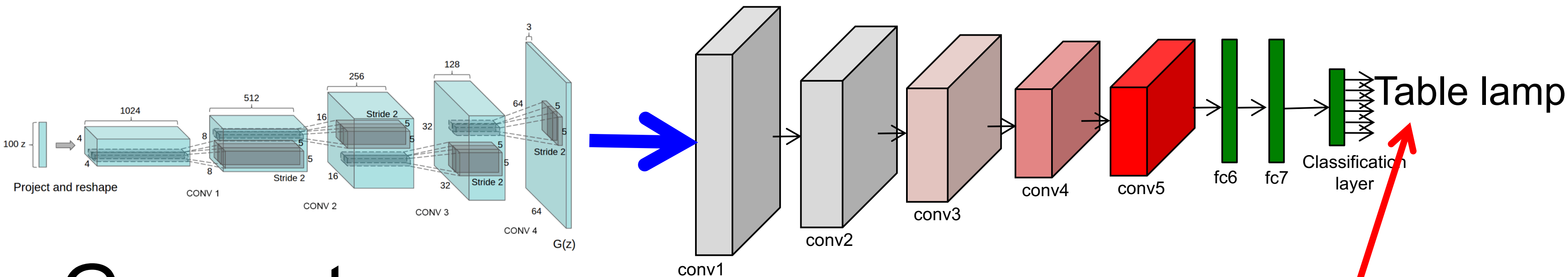
Network to visualize



Two components



Two components



Generator

Unit to visualize

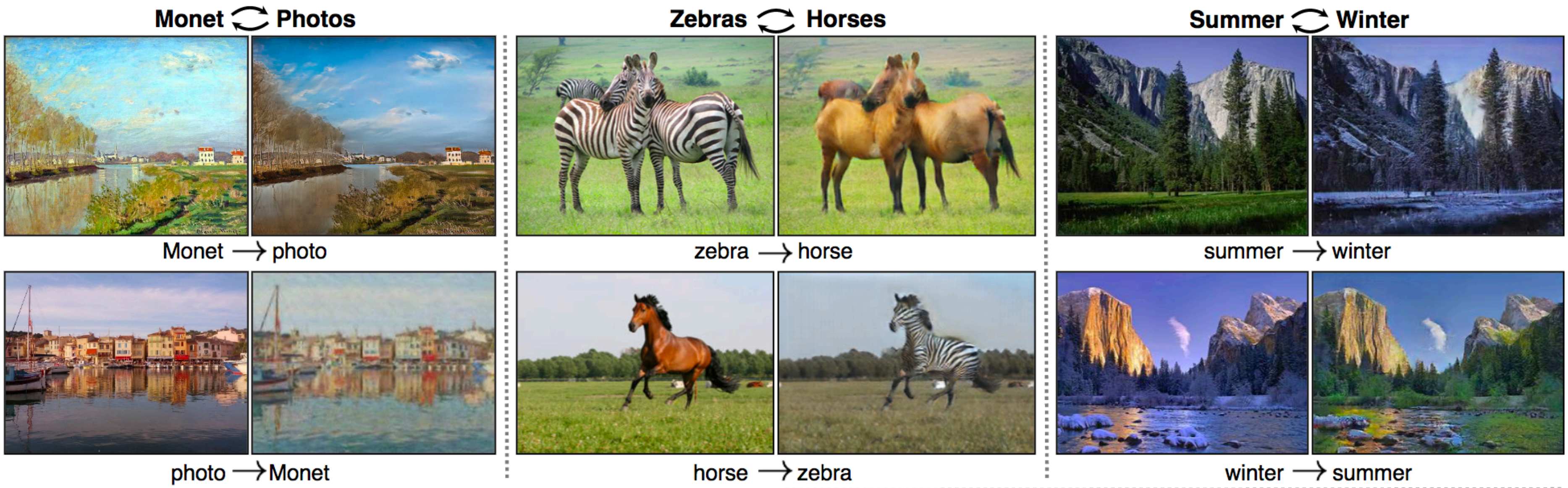
Synthesizing Images Preferred by CNN

ImageNet-Alexnet-final units (class units)

Nguyen A, Dosovitskiy A, Yosinski J, Brox T, Clune J. (2016). "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks.". arXiv:1605.09304.

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

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



https://arxiv.org/pdf/1703.10593.pdf

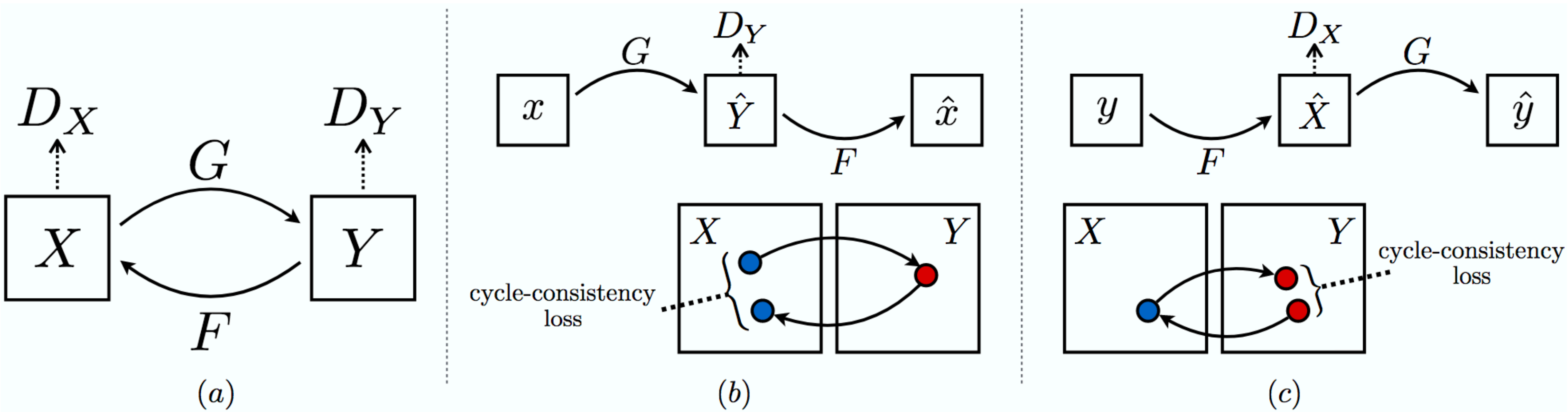


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

https://arxiv.org/pdf/1703.10593.pdf