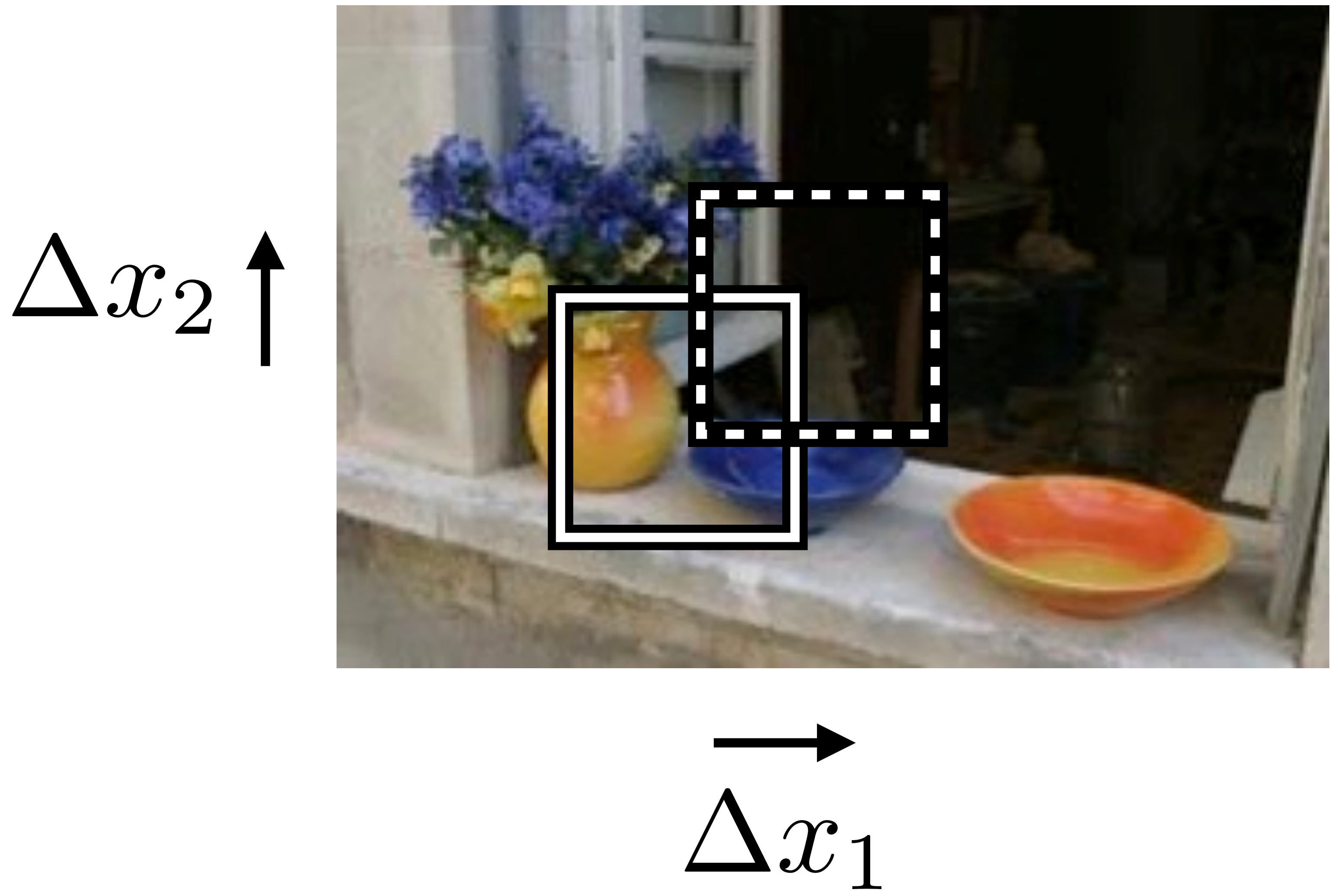
Harris Corners



$$\text{SSD} = \sum_{\mathcal{R}} |I(\mathbf{x}) - I(\mathbf{x} + \Delta\mathbf{x})|^2$$

$$= \Delta\mathbf{x}^T \mathbf{H} \Delta\mathbf{x}$$

$$\mathbf{H} = \sum_{\mathcal{R}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

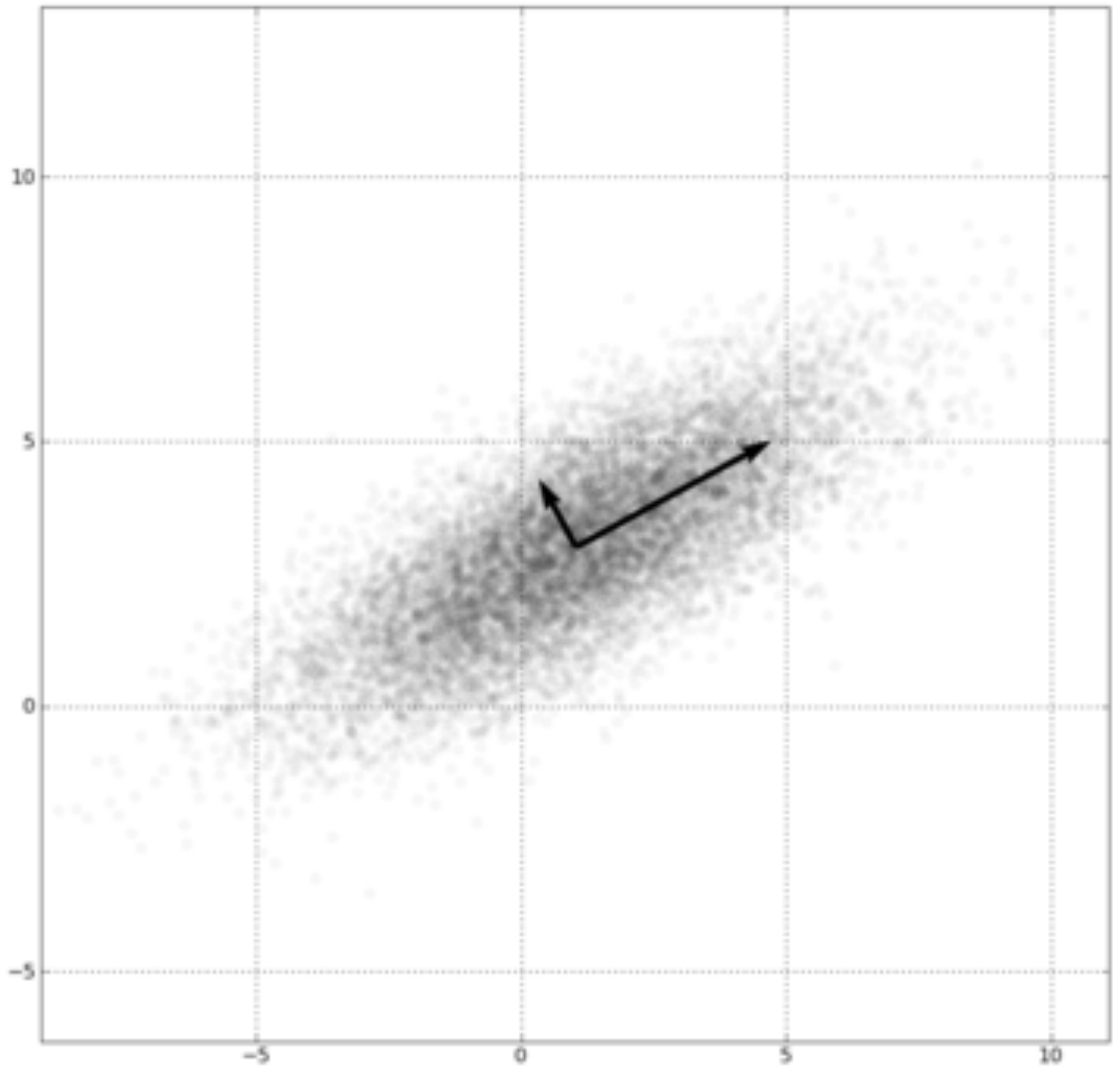
SSD function must be large **for all shifts** $\Delta\mathbf{x}$ for a corner / 2D structure

This implies that **both eigenvalues of** $\mathbf{H}$ must be **large**

Note that $\mathbf{H}$ is a **2x2 matrix**

Recap: Computing **Eigenvalues** and **Eigenvectors**

10.2



https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors

Harris Corner Detection

1. Compute image gradients ~~over small region~~

2. Compute the covariance matrix

3. Compute eigenvectors and eigenvalues

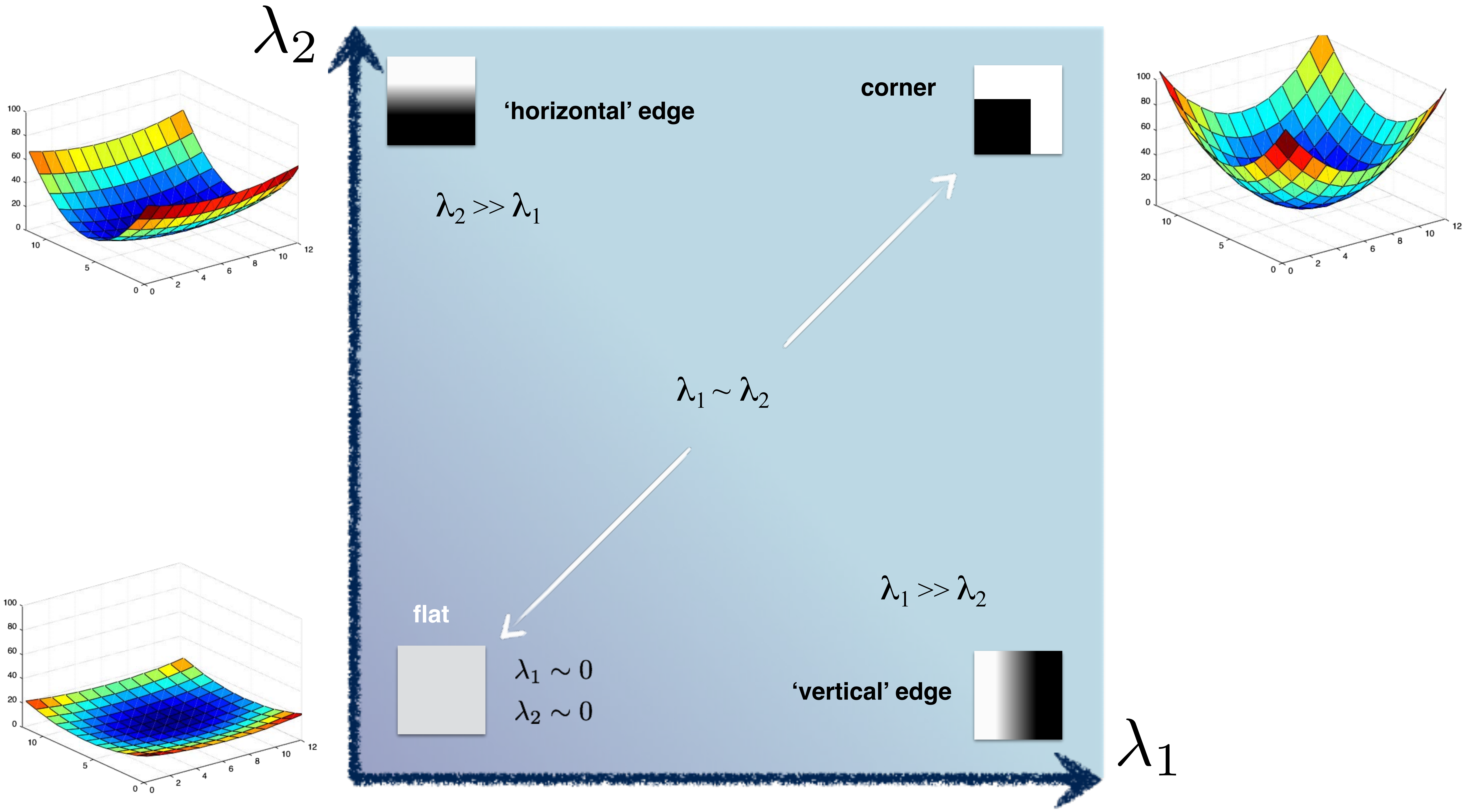
4. Use threshold on eigenvalues to detect corners

$$I_x = \frac{\partial I}{\partial x} \qquad I_y = \frac{\partial I}{\partial y}$$

$$\begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix}$$

Slide Adopted: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Threshold on Eigenvalues to **Detect Corners**

(a function of)

Harris & Stephens (1988)

$$\det(C) - \kappa \operatorname{trace}^2(C)$$

Kanade & Tomasi (1994)

$$\min(\lambda_1, \lambda_2)$$

Nobel (1998)

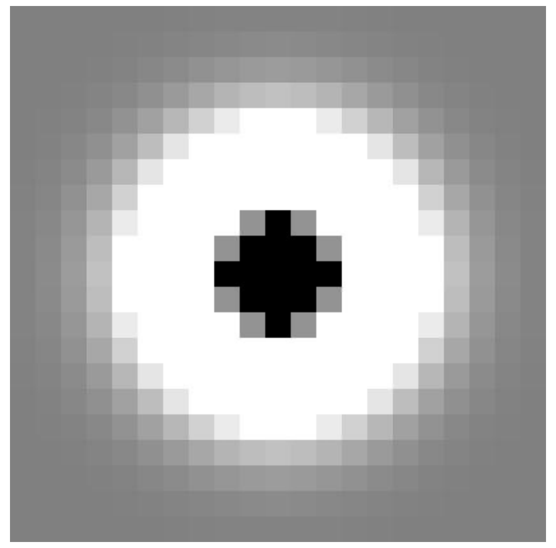
$$\frac{\det(C)}{\operatorname{trace}(C) + \epsilon}$$

10.3

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

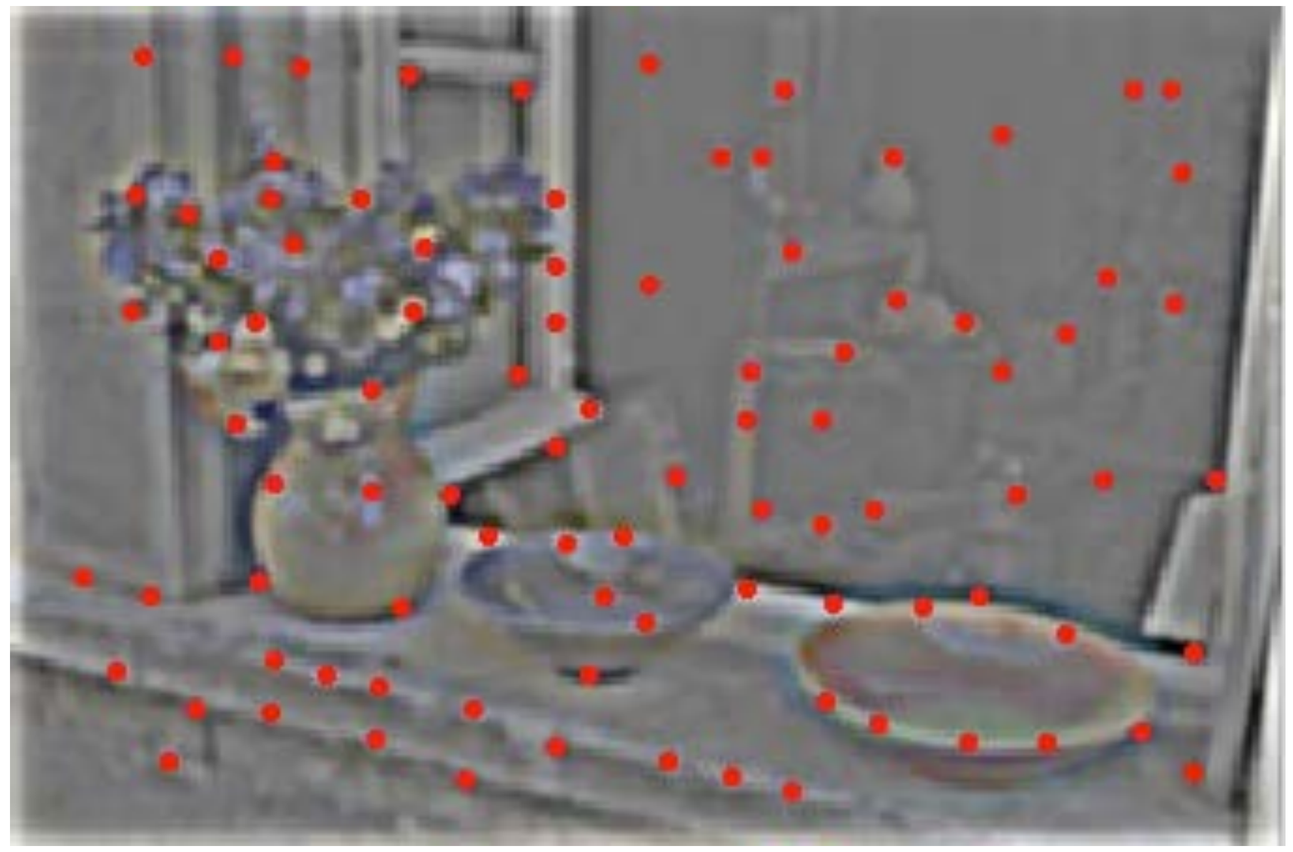
Difference of Gaussian

DoG = centre-surround filter

 *  = 

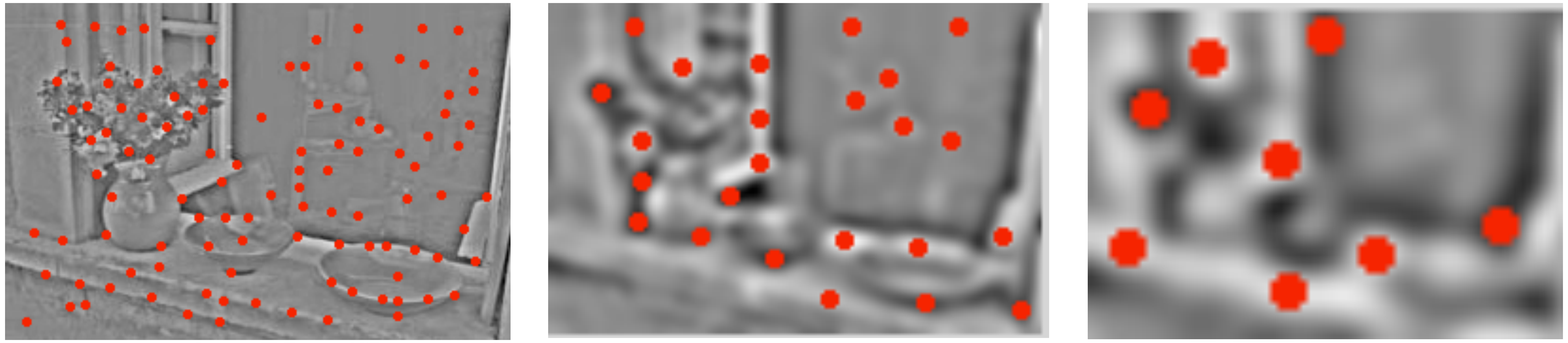
- Find local-maxima of the centre surround response

Non-maximal suppression:
These points are maxima in
a 10 pixel radius →



Difference of Gaussian

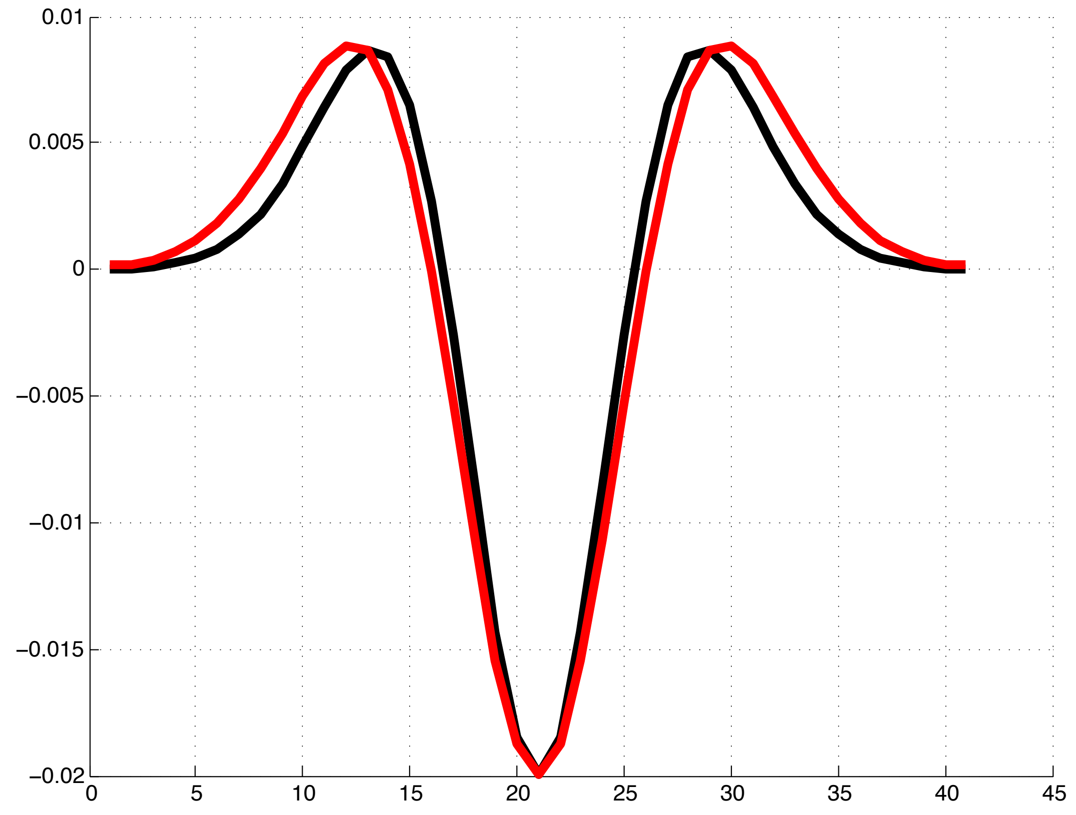
DoG detects blobs at scale that depends on the Gaussian standard deviation(s)



Note: DOG ≈ Laplacian of Gaussian

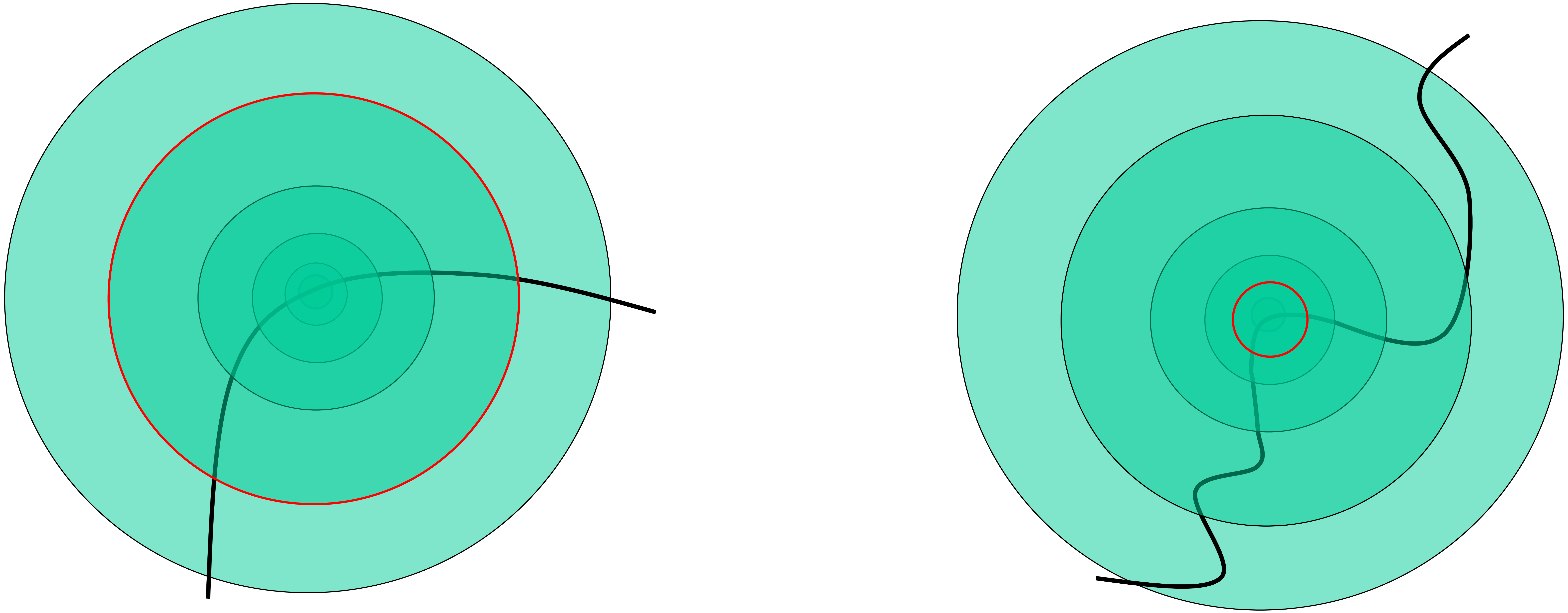
$$\text{red} = [1\ \ -2\ 1] * g(x;\ 5.0)$$

$$\text{black} = g(x;\ 5.0) - g(x;\ 4.0)$$



Scale Invariant Interest Point Detection

Find local maxima in both **position** and **scale**

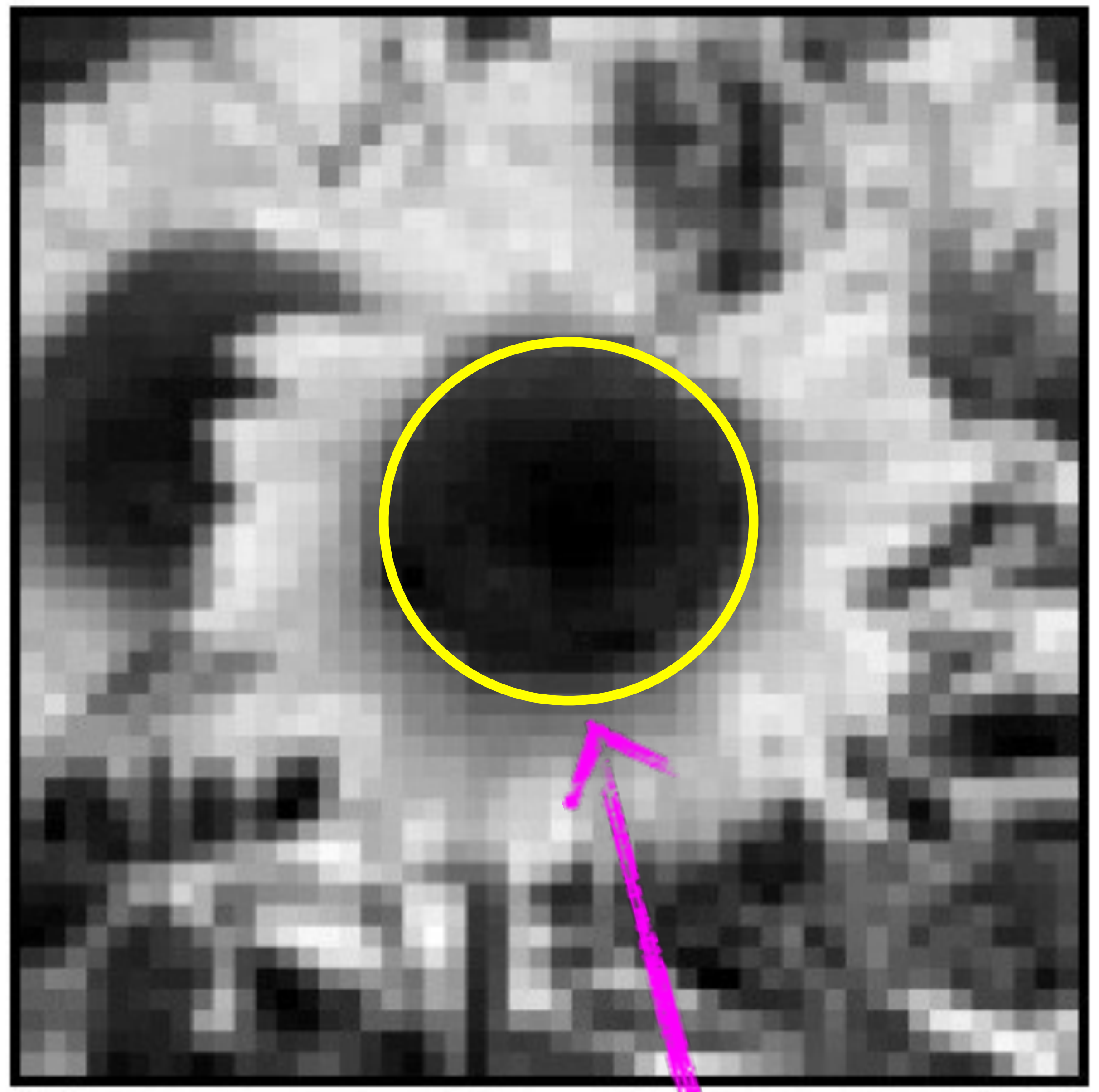


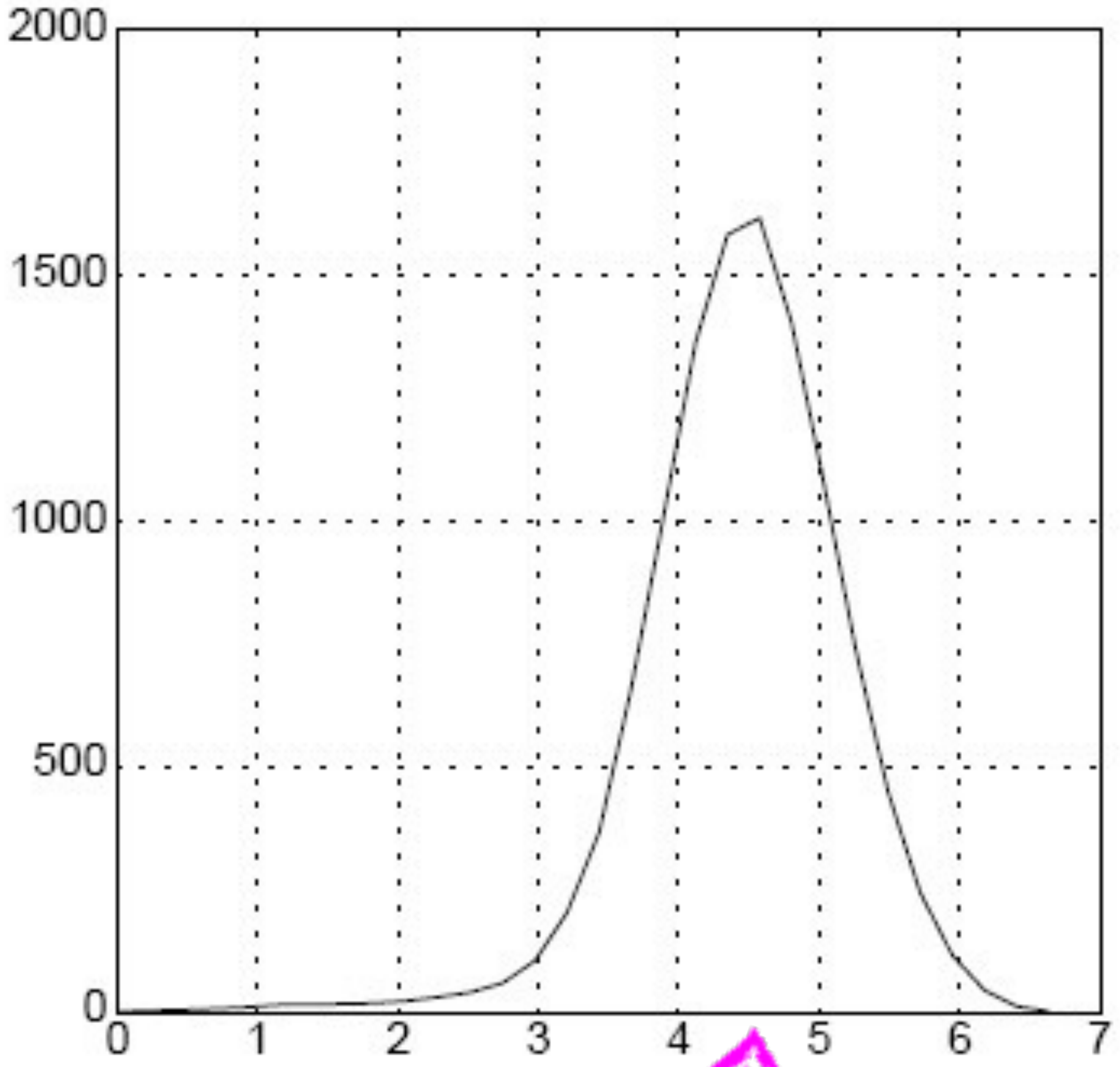
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Characteristic Scale

characteristic scale - the scale that produces peak filter response



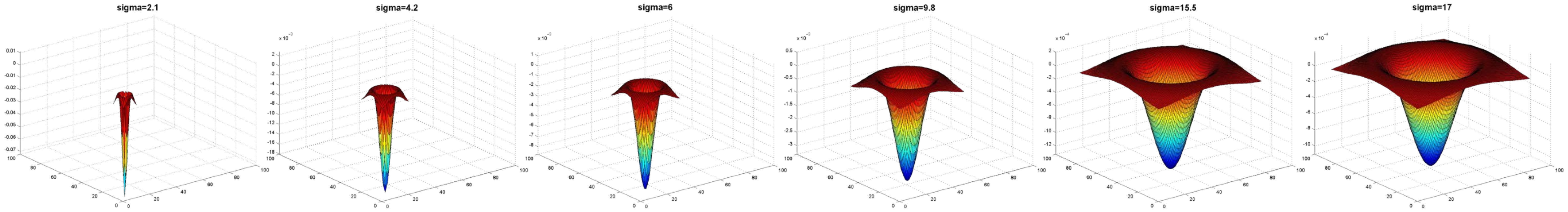


characteristic scale

we need to search over characteristic scales

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Applying **Laplacian** Filter at Different **Scales**

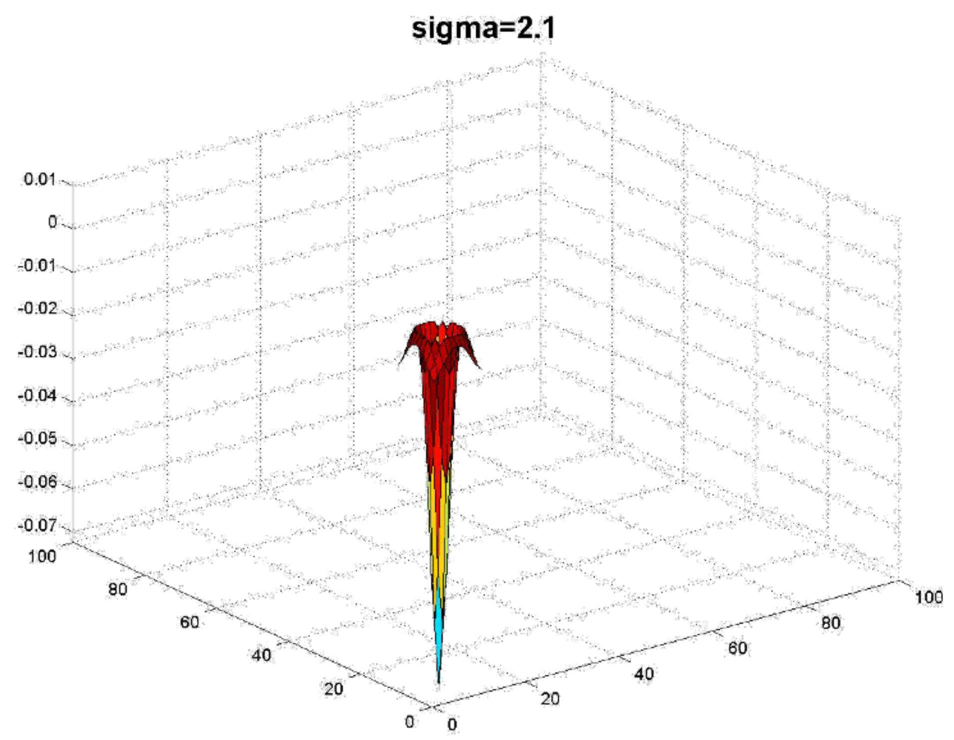


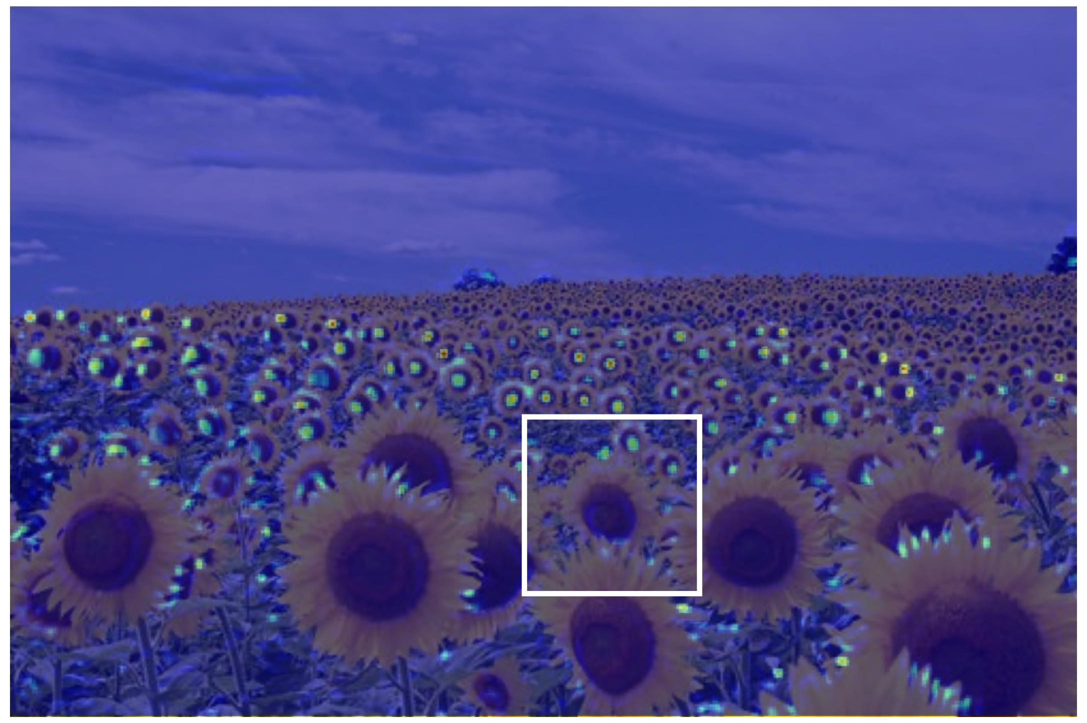
Full size

3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

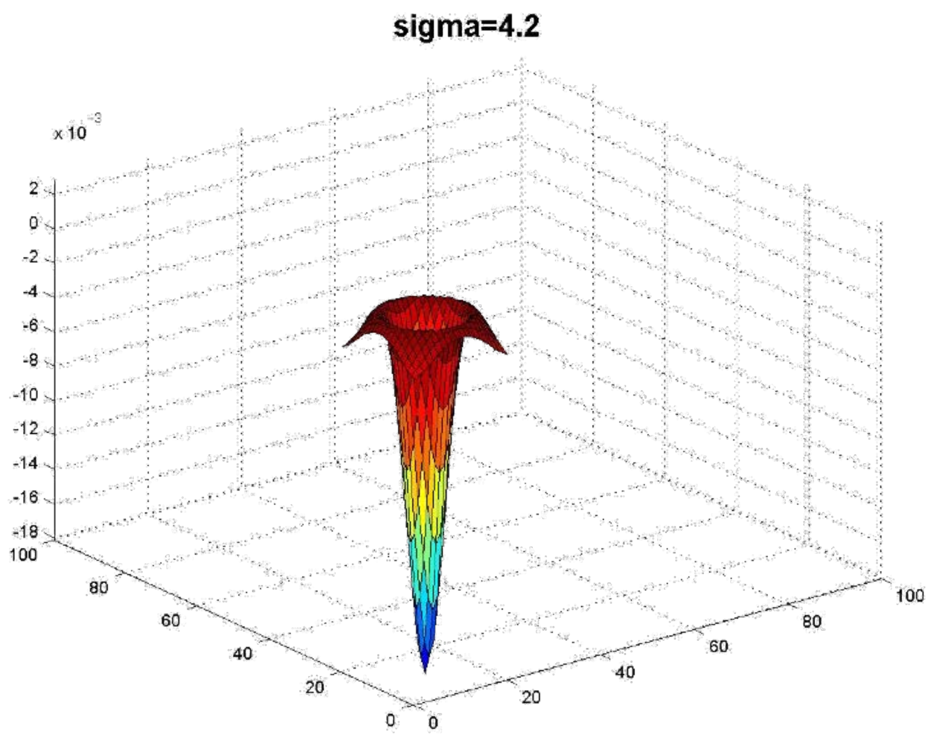
Applying **Laplacian** Filter at Different **Scales**

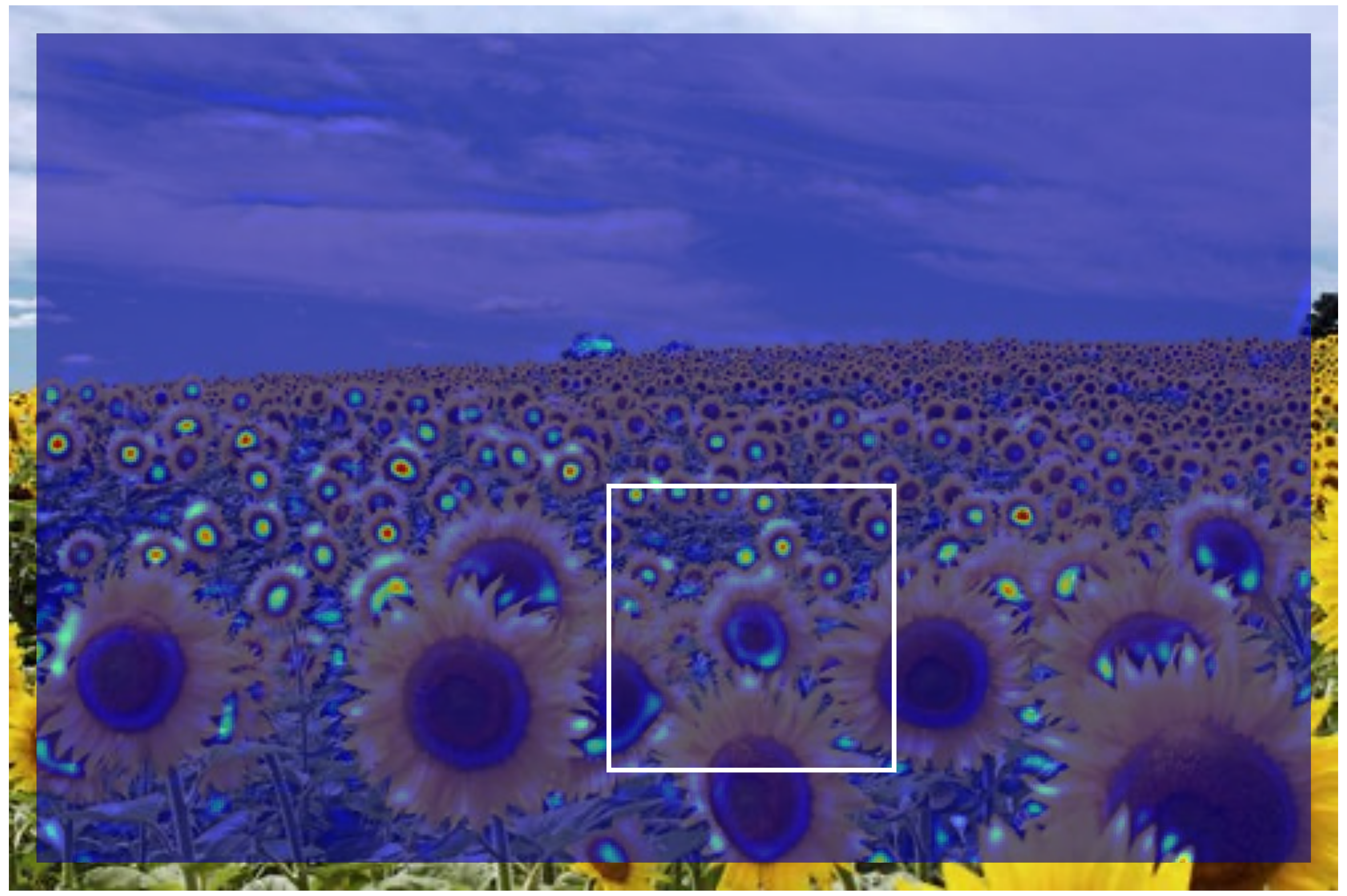


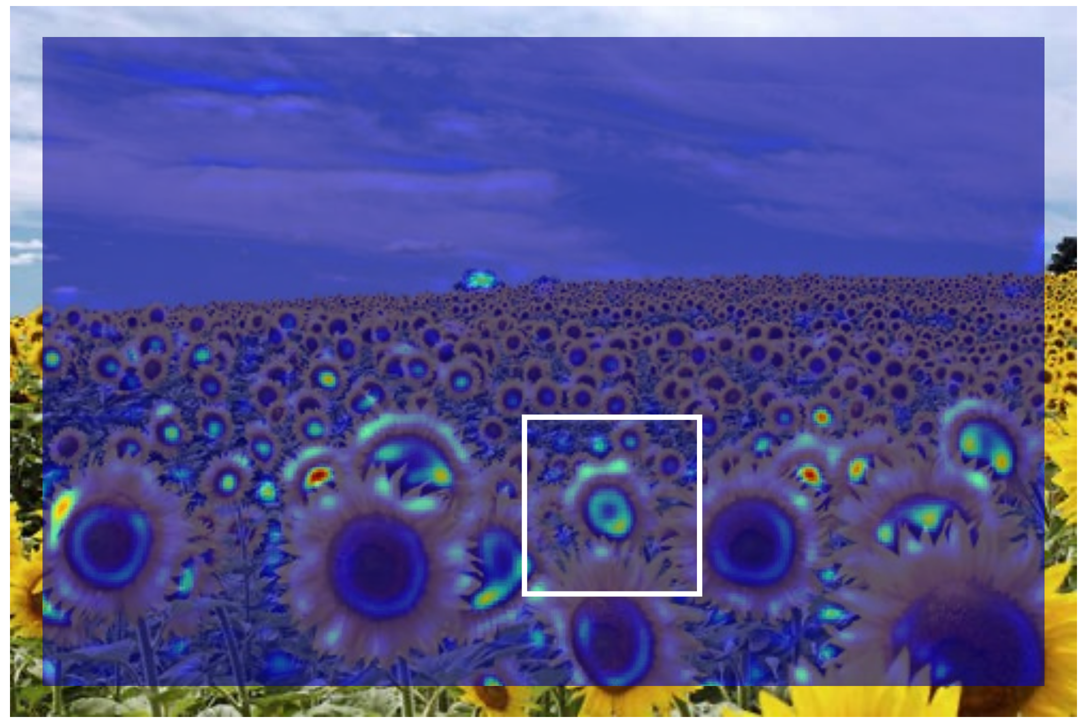


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Applying **Laplacian** Filter at Different **Scales**

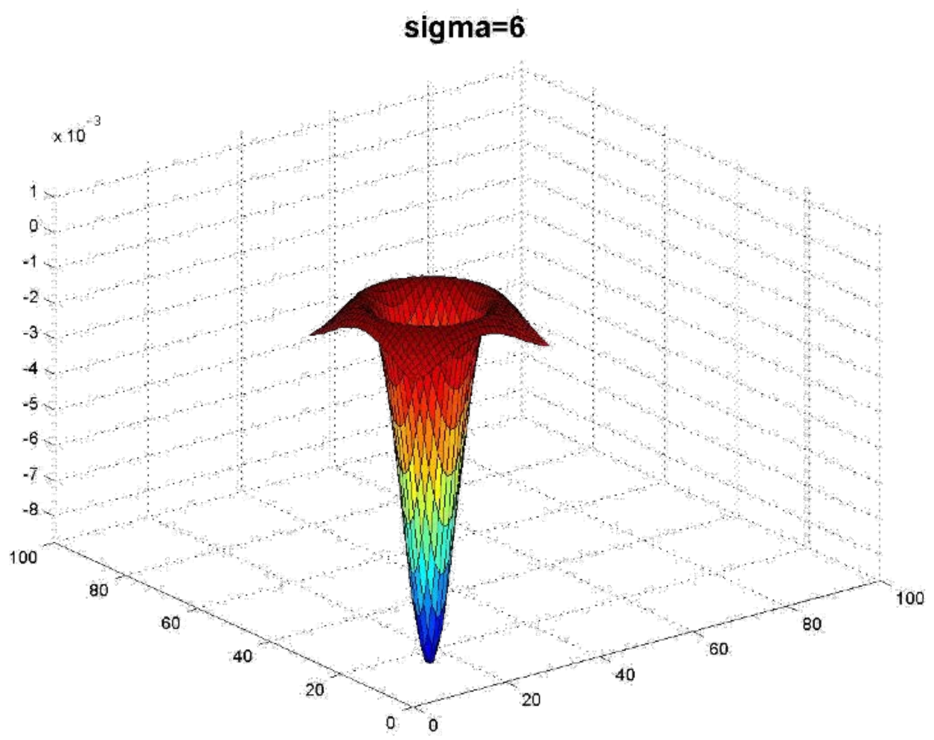


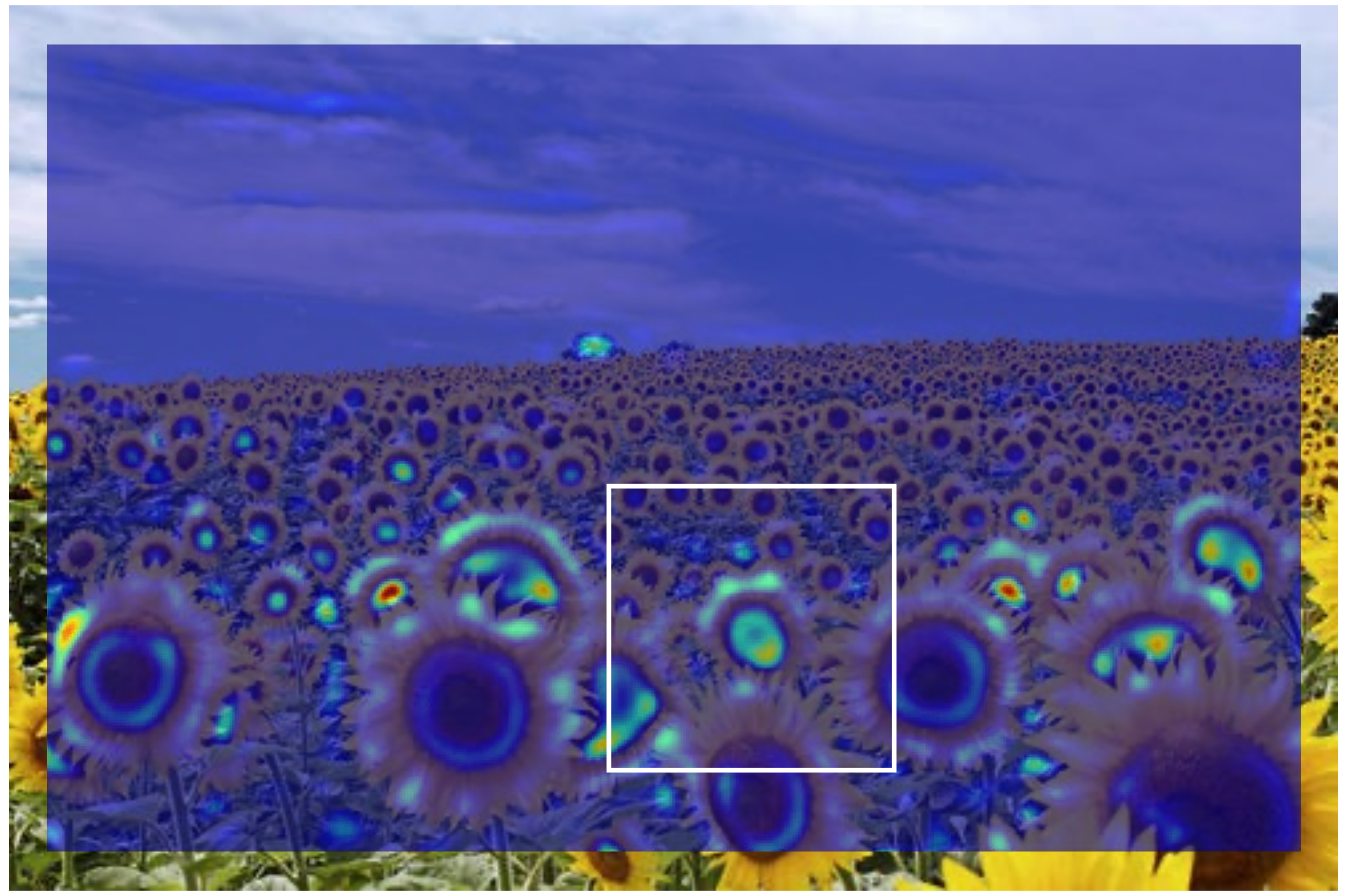




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

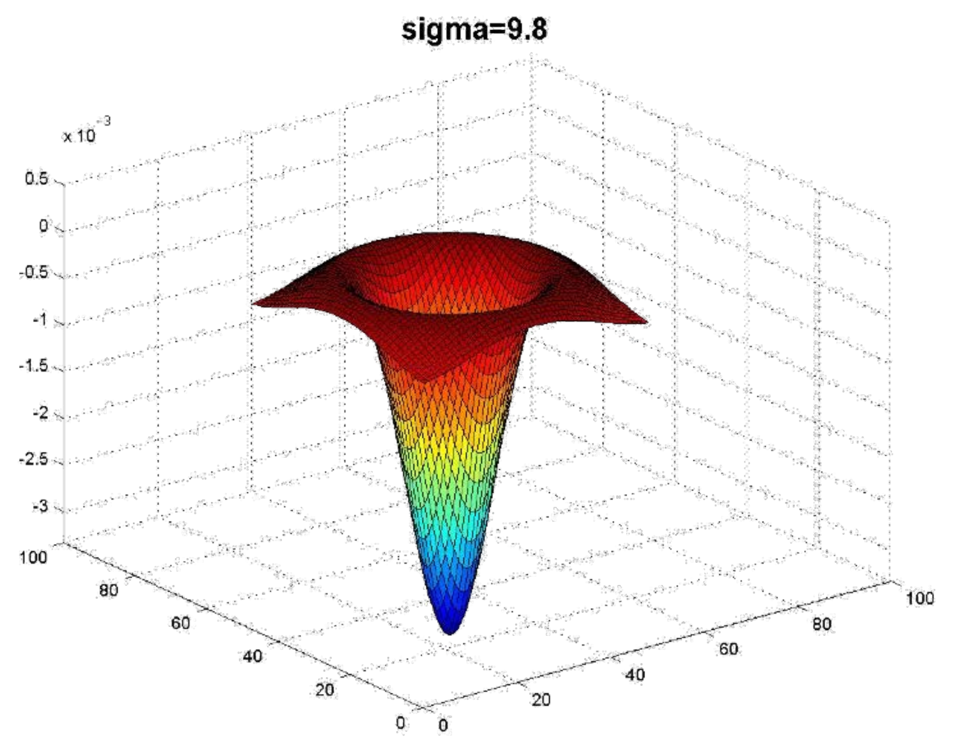
Applying **Laplacian** Filter at Different **Scales**

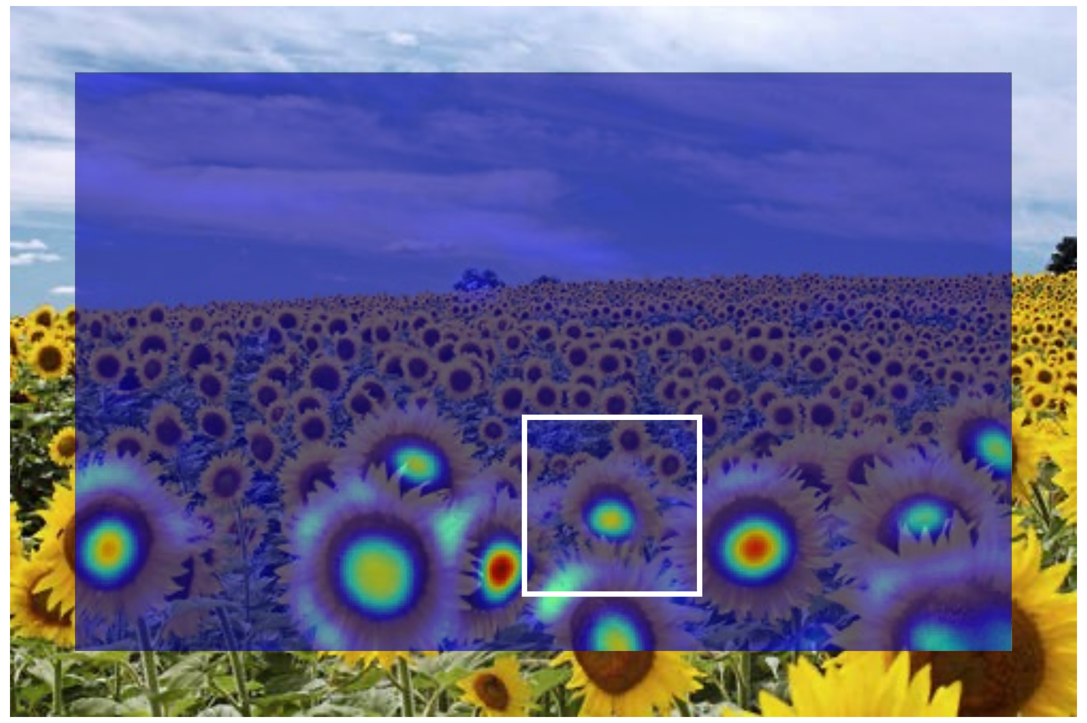




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

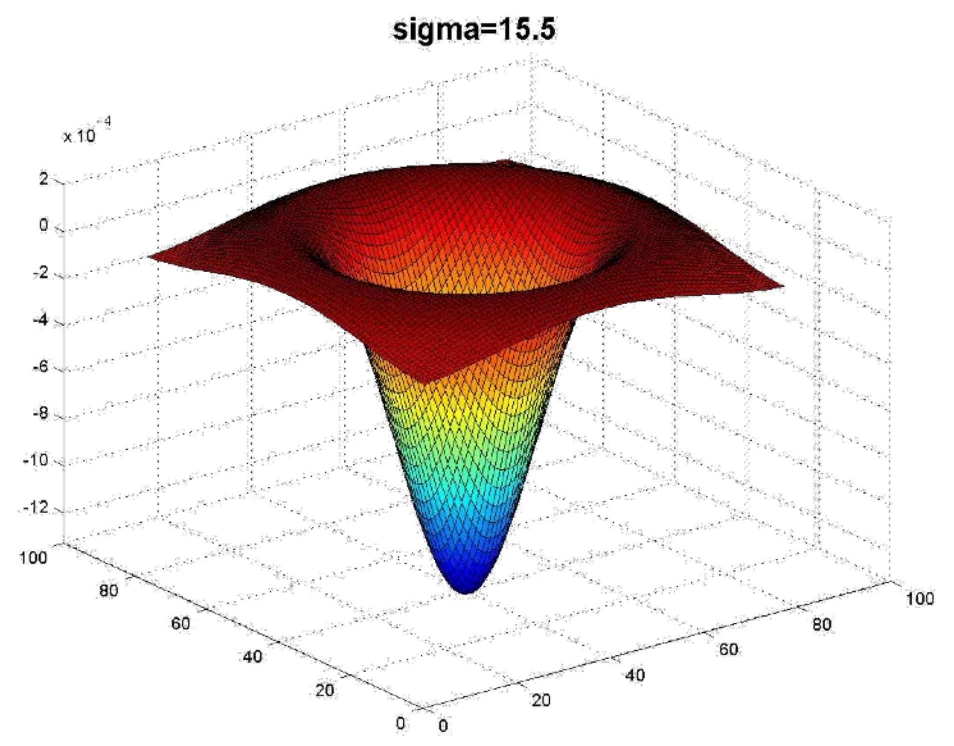
Applying **Laplacian** Filter at Different **Scales**

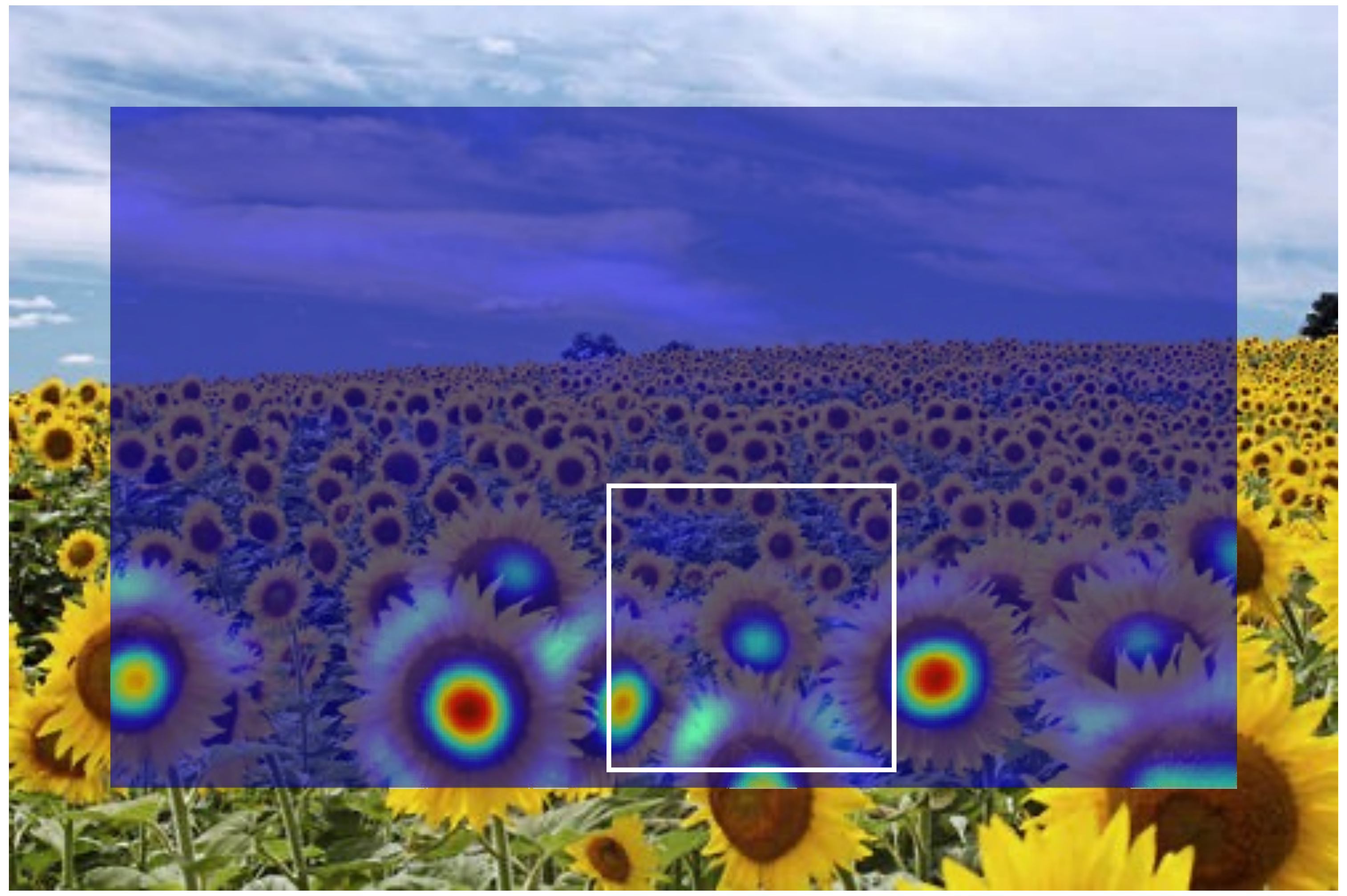


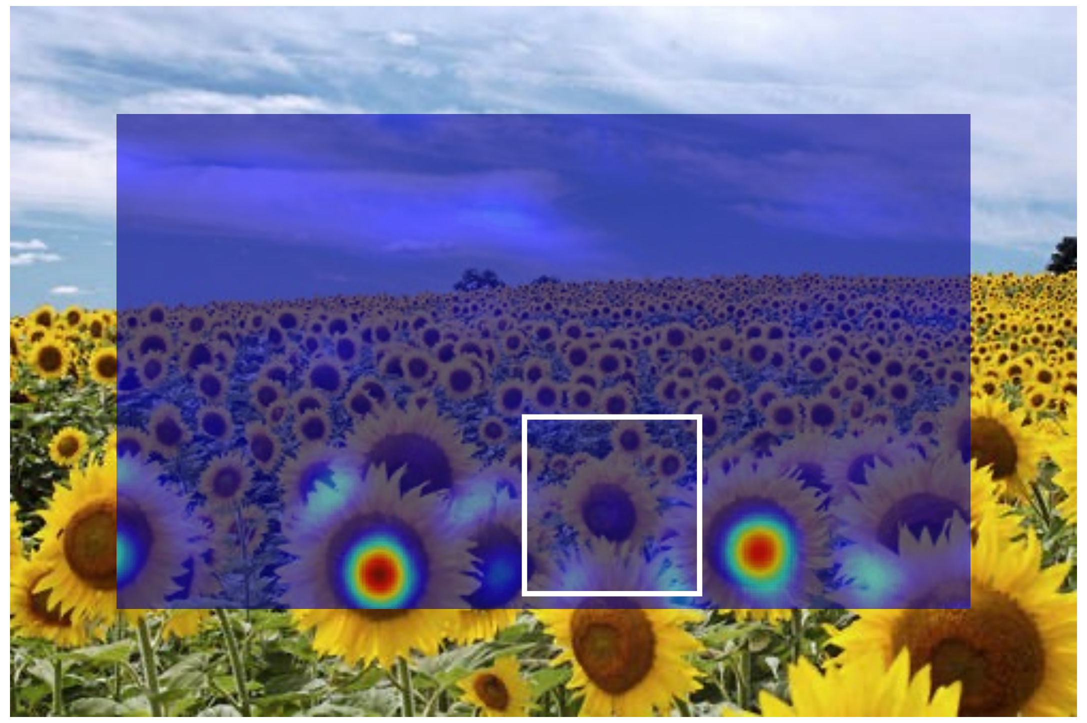


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Applying **Laplacian** Filter at Different **Scales**

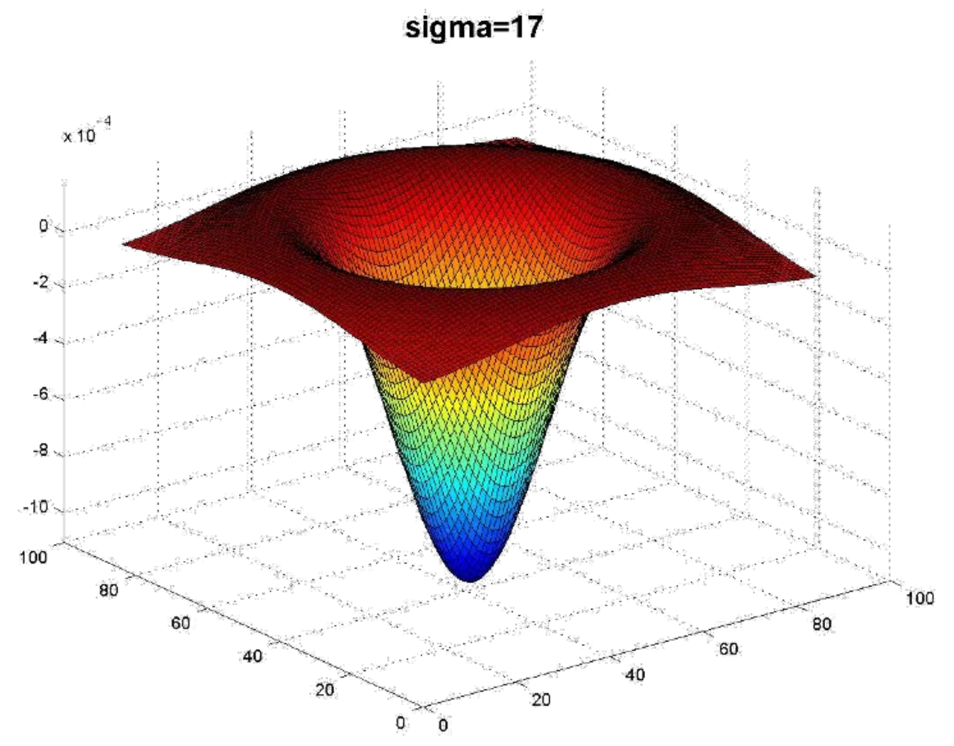


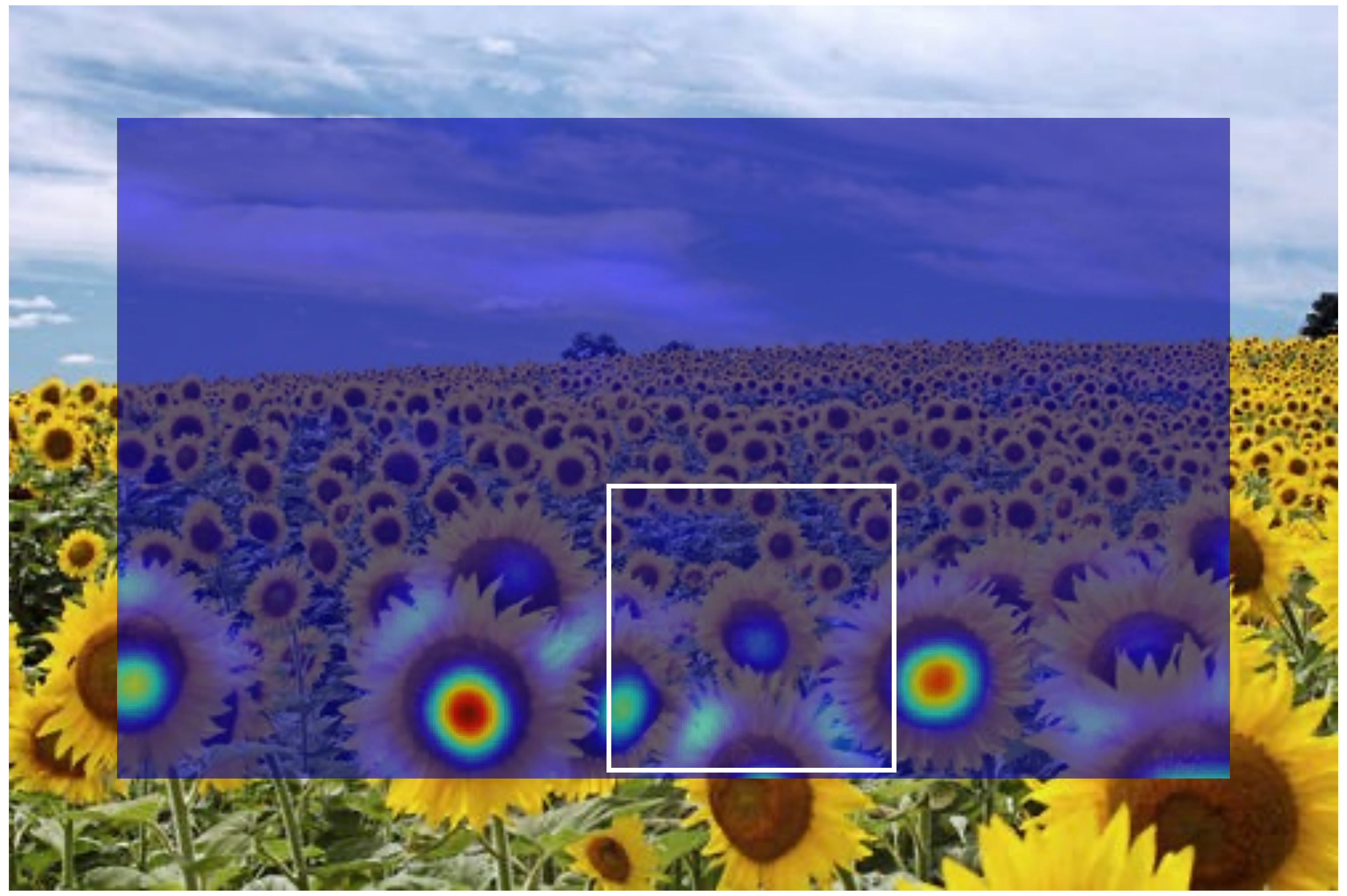


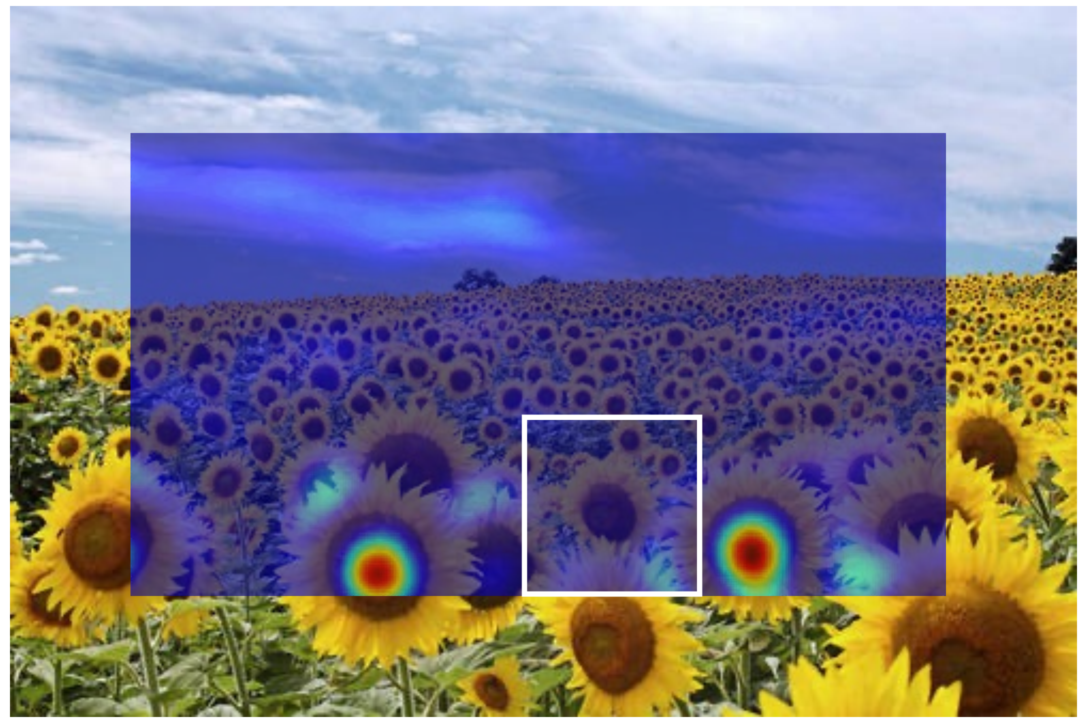


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Applying **Laplacian** Filter at Different **Scales**

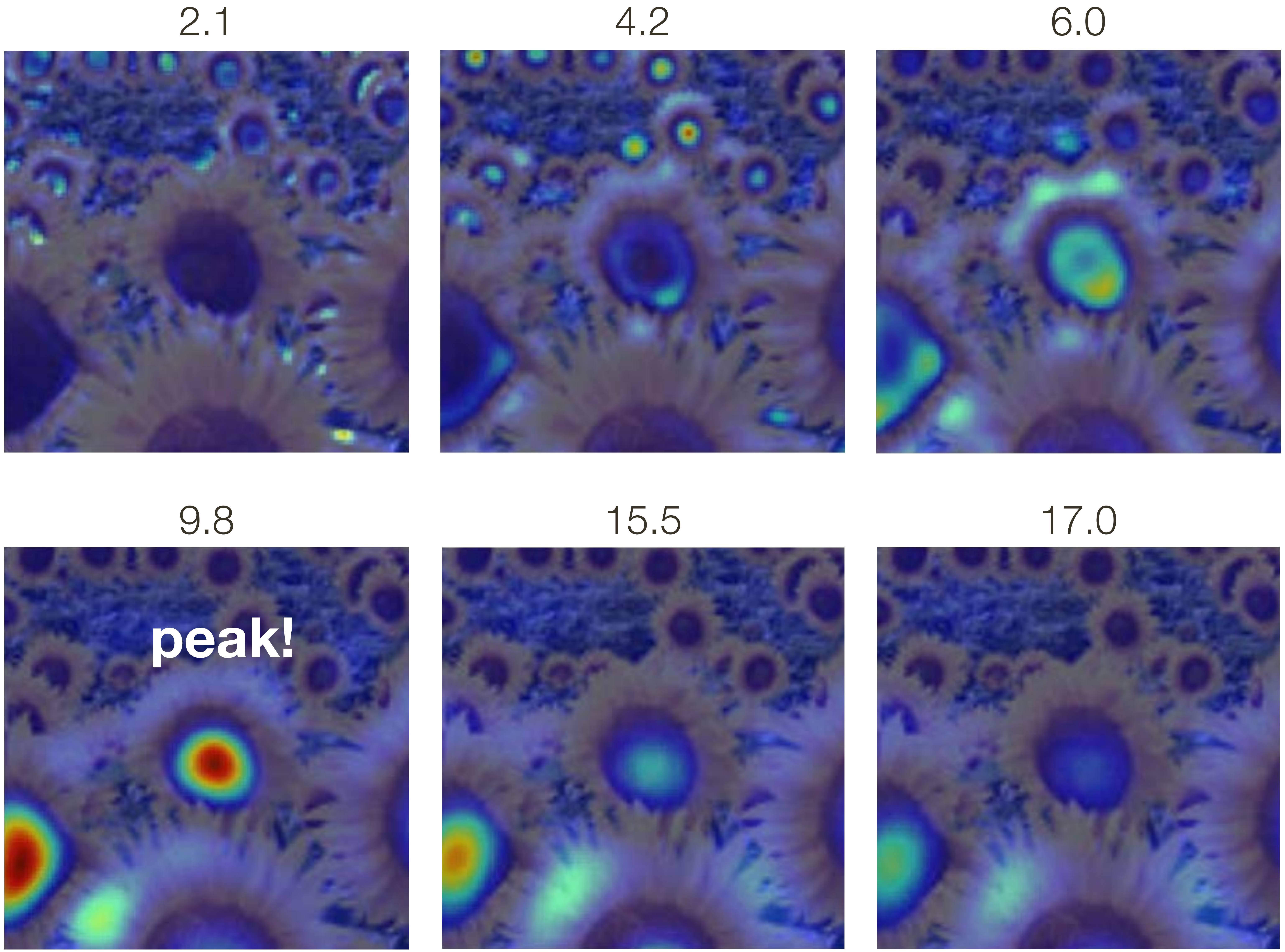






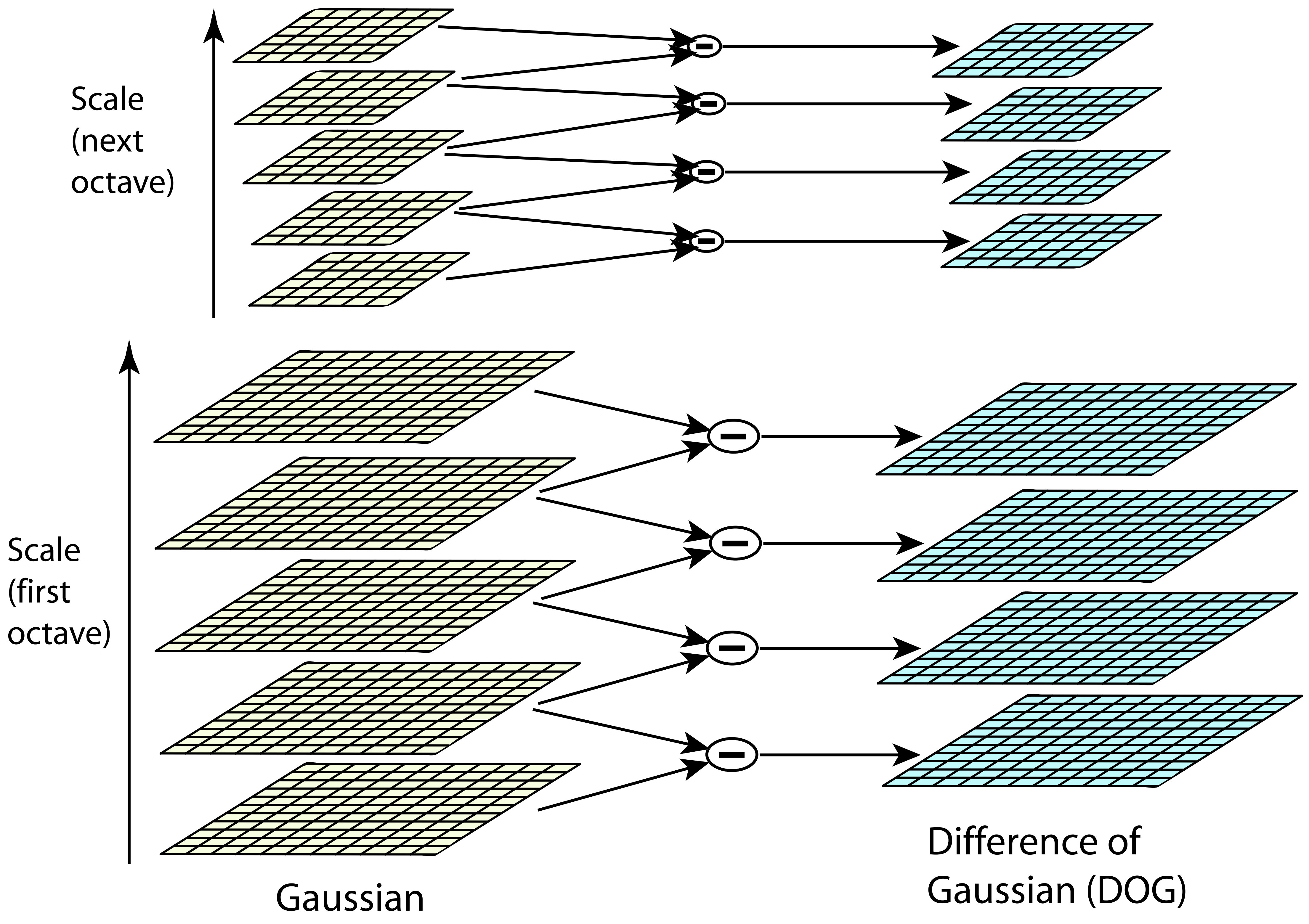
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

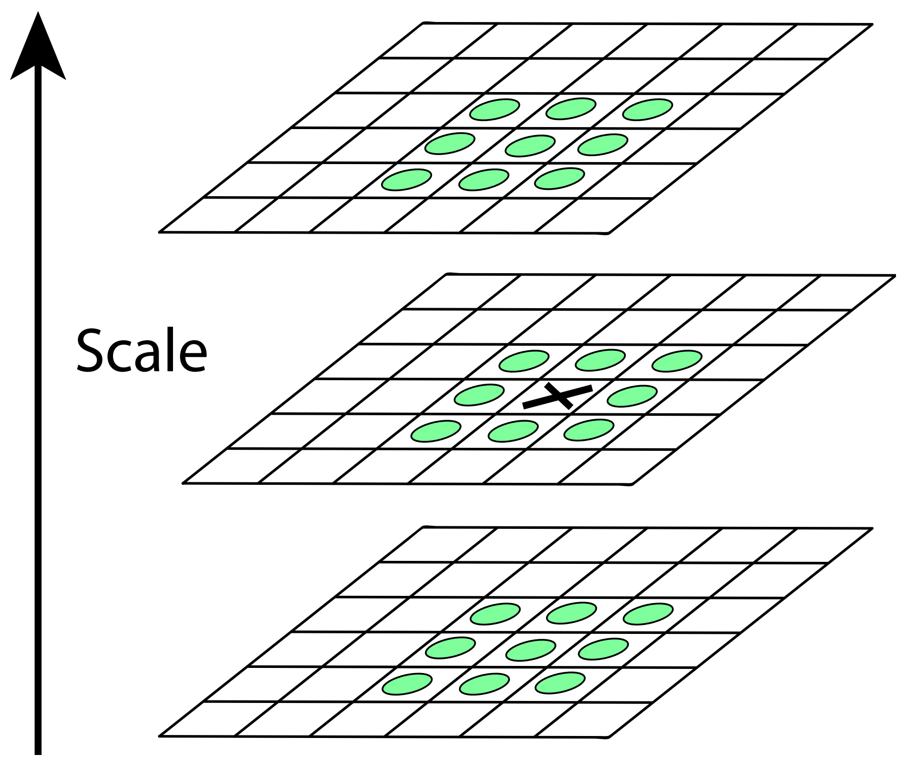
Applying **Laplacian** Filter at Different **Scales**



Scale Selection

A DOG (Laplacian) Pyramid is formed with multiple scales per ocatve

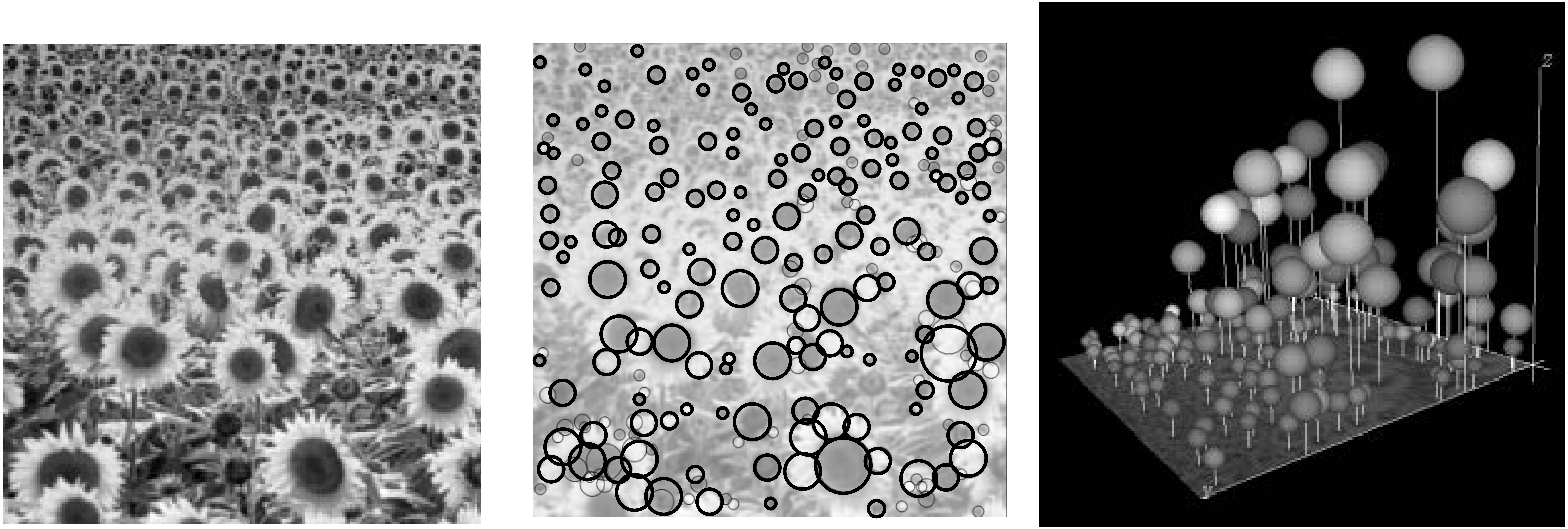




Detections are local maxima in a 3x3x3 scale-space window

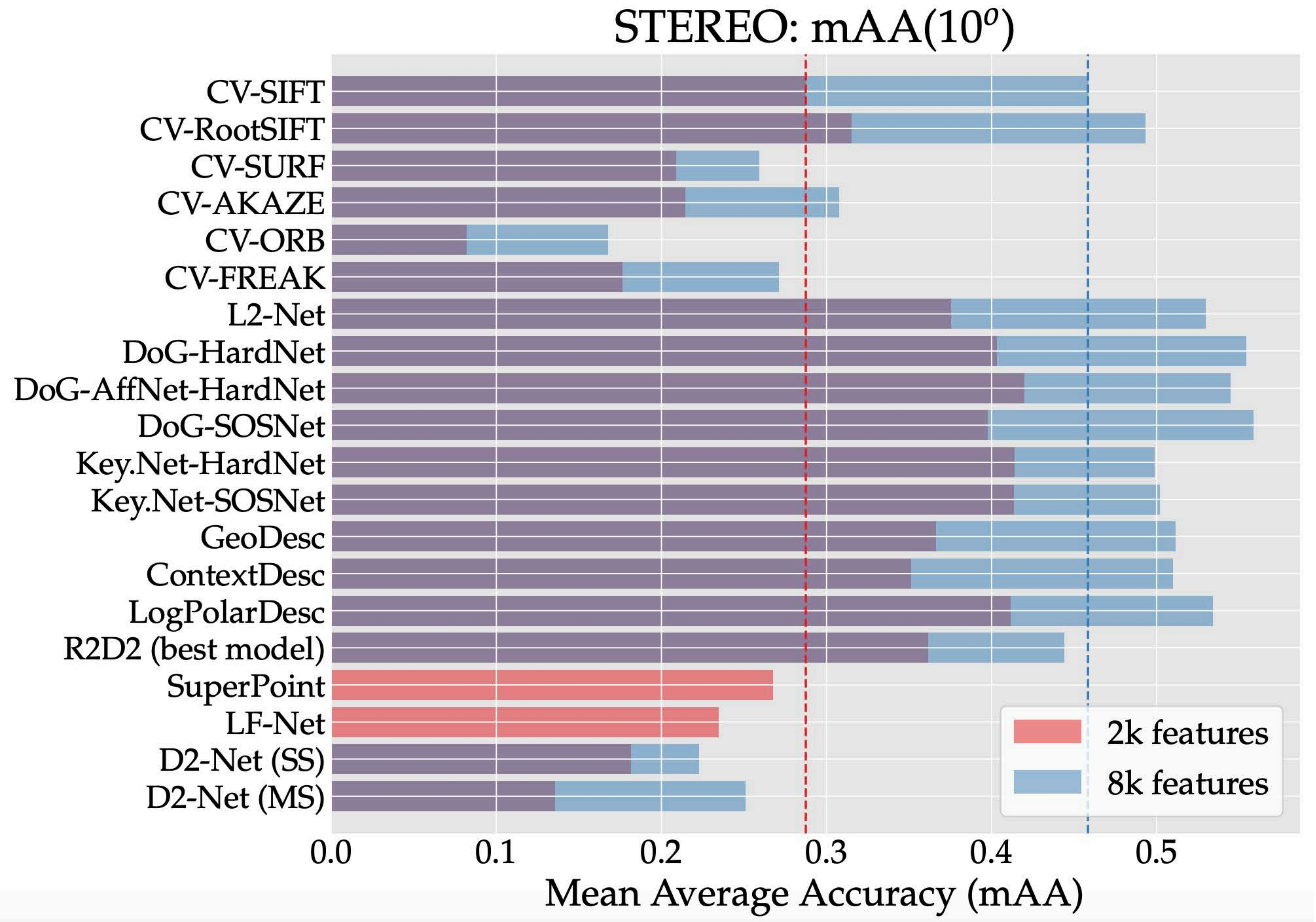
Scale Selection

Maximising the DOG function in scale as well as space performs scale selection



[T. Lindeberg]

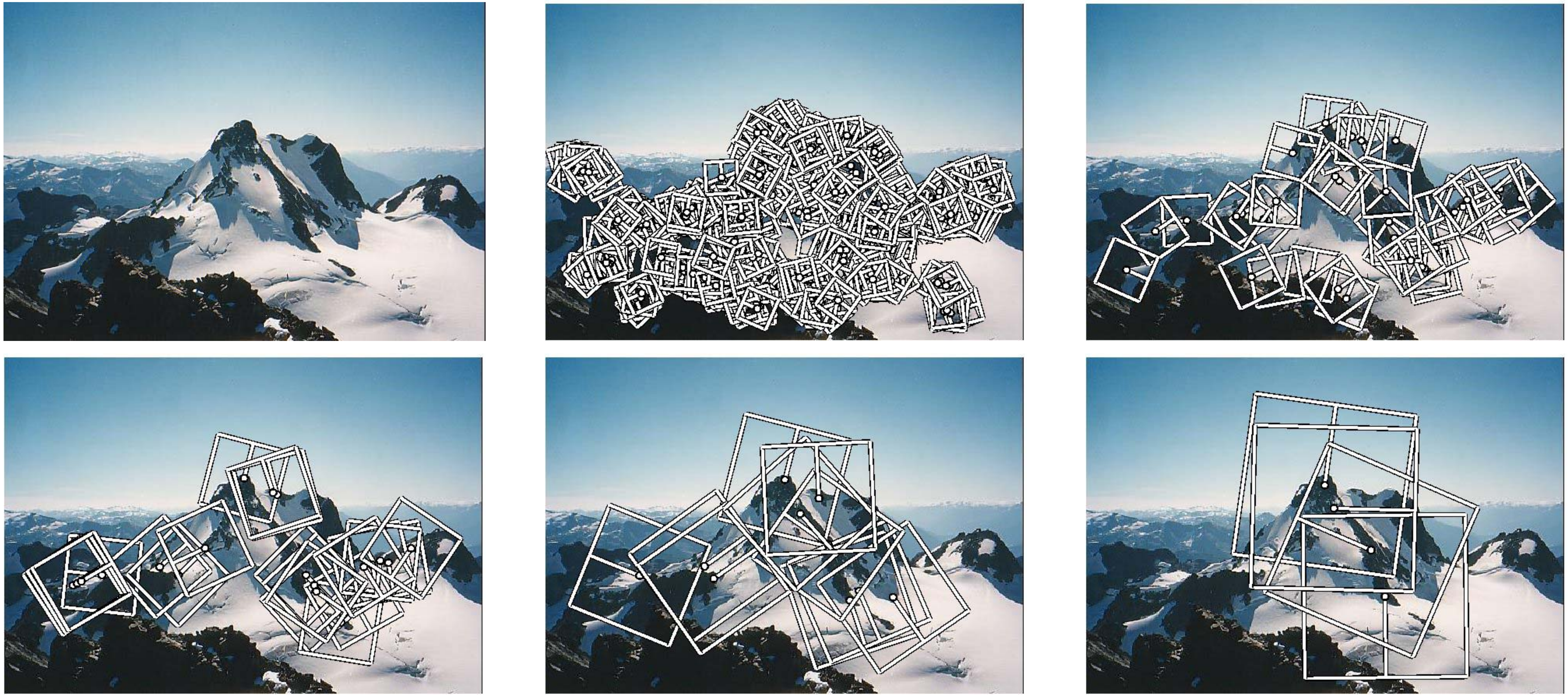
Difference of Gaussian blobs in 2020



Multi-Scale Harris Corners

```
For each level of the Gaussian pyramid

    compute Harris feature response

For each level of the Gaussian pyramid

    if local maximum and cross-scale

    save scale and location of feature
```
(x, y, s)

Multi-Scale Harris Corners



Summary

Edges are useful image features for many applications, but suffer from the aperture problem

Canny Edge detector combines edge filtering with linking and hysteresis steps

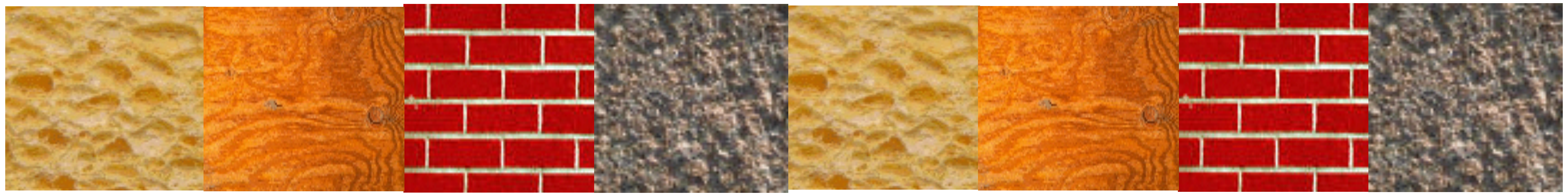
Corners / Interest Points have 2D structure and are useful for correspondence

Harris corners are minima of a local SSD function

DoG maxima can be reliably located in scale-space and are useful as interest points

THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 11: Texture

(unless otherwise stated slides are taken or adopted from **Bob Woodham, Jim Little** and **Fred Tung**)

Menu for Today

Topics:

— Texture **Analysis, Synthesis**

— Filter Banks, Data-driven Methods

Readings:

— **Today's** Lecture: Forsyth & Ponce (2nd ed.) 3.1-3.3

Reminders:

— **Midterm** is right after reading break! **February 24th 12:30 pm**

— **Quiz 3:** Wednesday (Feb 12th)

— **Assignment 2:** due **Feb 13th**

Learning Goals

Understanding image as a collection of basis elements

A first step towards a “generative modelling” of images

Texture

What is **texture**?

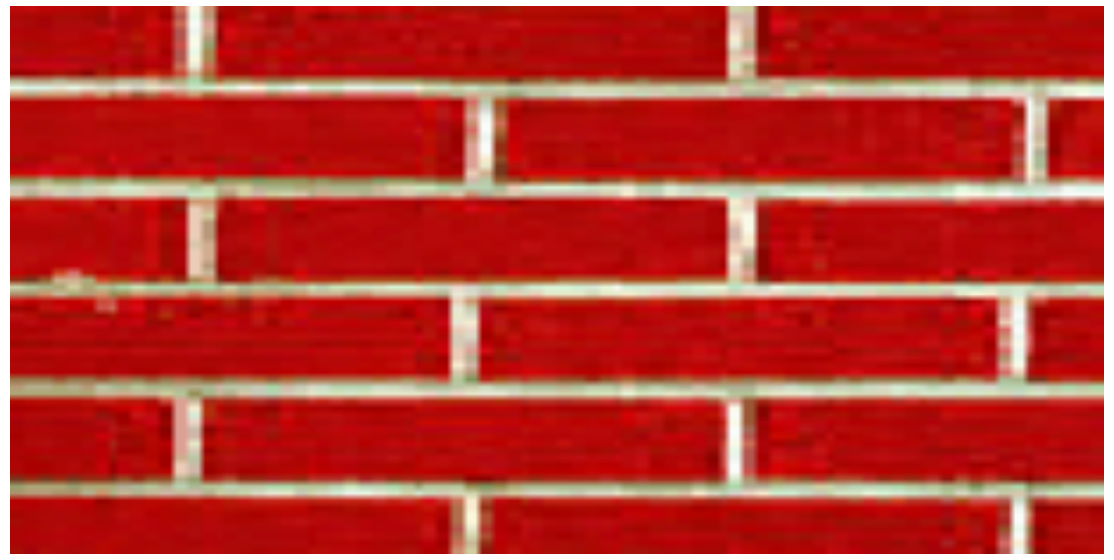
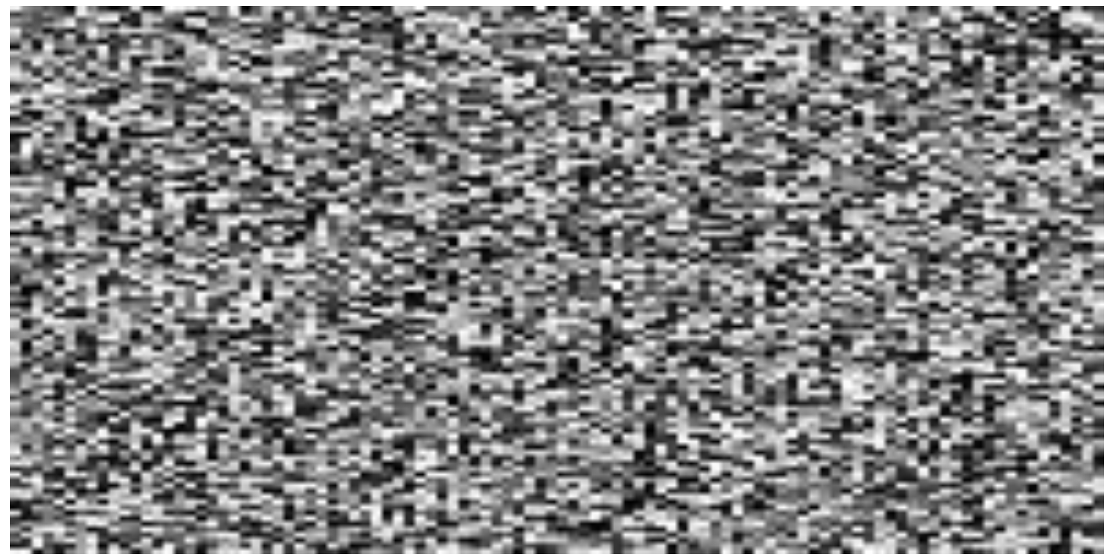
  

Figure Credit: Alexei Efros and Thomas Leung

Texture is widespread, easy to recognize, but hard to define

Views of large numbers of small objects are often considered textures
— e.g. grass, foliage, pebbles, hair

Patterned surface markings are considered textures
— e.g. patterns on wood

Definition of **Texture**

(Functional) **Definition**:

Texture is detail in an image that is at a scale too small to be resolved into its constituent elements and at a scale large enough to be apparent in the spatial distribution of image measurements

Uses of **Texture**

Texture can be a strong cue to **object identity** if the object has distinctive material properties

Texture can be a strong cue to an **object's shape** based on the deformation of the texture from point to point.

— Estimating surface orientation or shape from texture is known as "**shape from texture**"

Lecture 11: Re-cap **Texture**

We will look at two main questions:

1. How do we represent texture?
   → Texture **analysis**

2. How do we generate new examples of a texture?
   → Texture **synthesis**

We begin with texture synthesis to set up **Assignment 3**

Texture **Synthesis**

Why might we want to synthesize texture?

1. To fill holes in images (**inpainting**)

— Art directors might want to remove telephone wires. Restorers might want to remove scratches or marks.

— We need to find something to put in place of the pixels that were removed

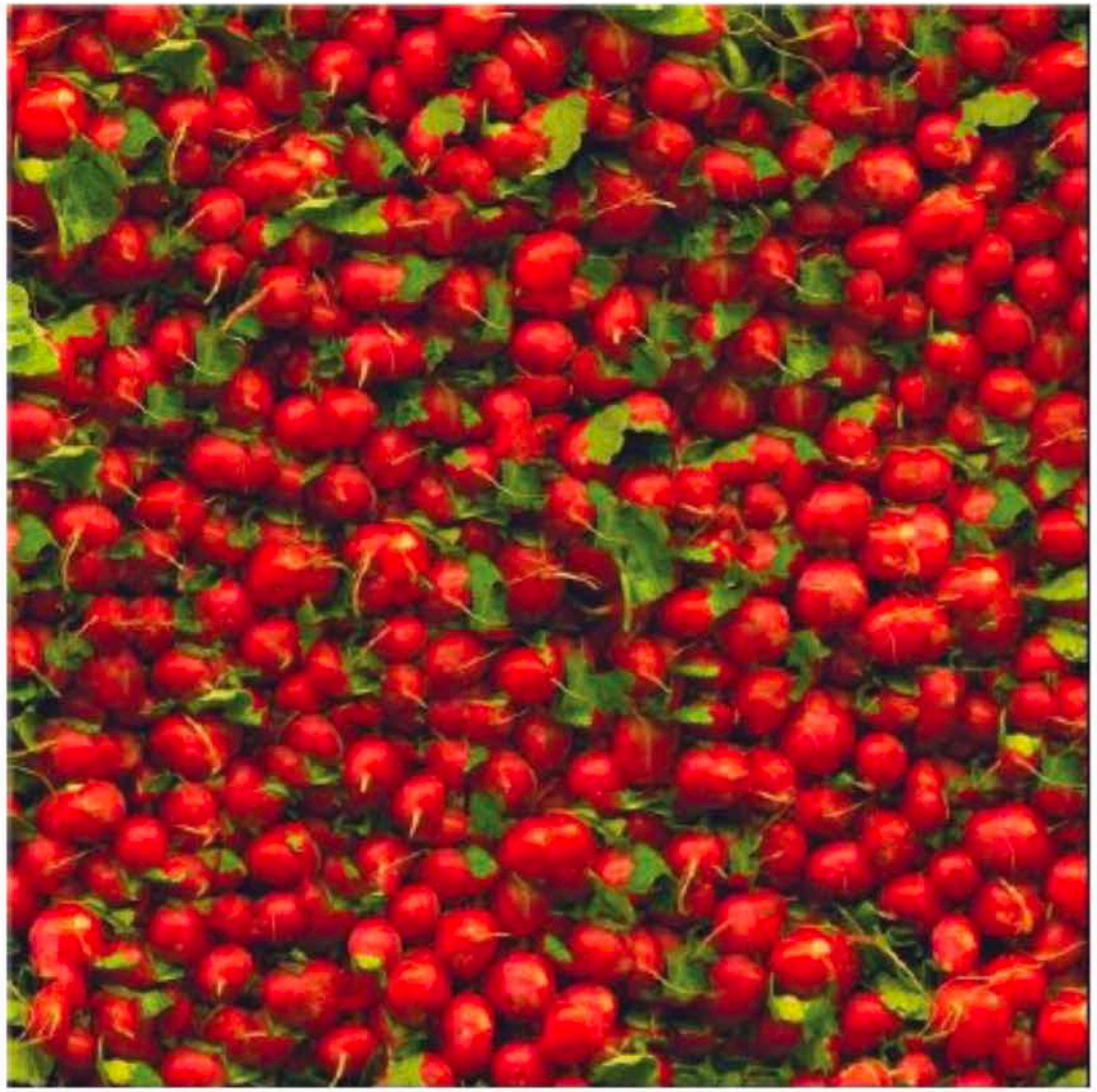
— We synthesize regions of texture that fit in and look convincing

2. To produce large quantities of texture for computer graphics

— Good textures make object models look more realistic

Texture **Synthesis**

radishes



lots more radishes

Szeliski, Fig. 10.49

Texture **Synthesis**

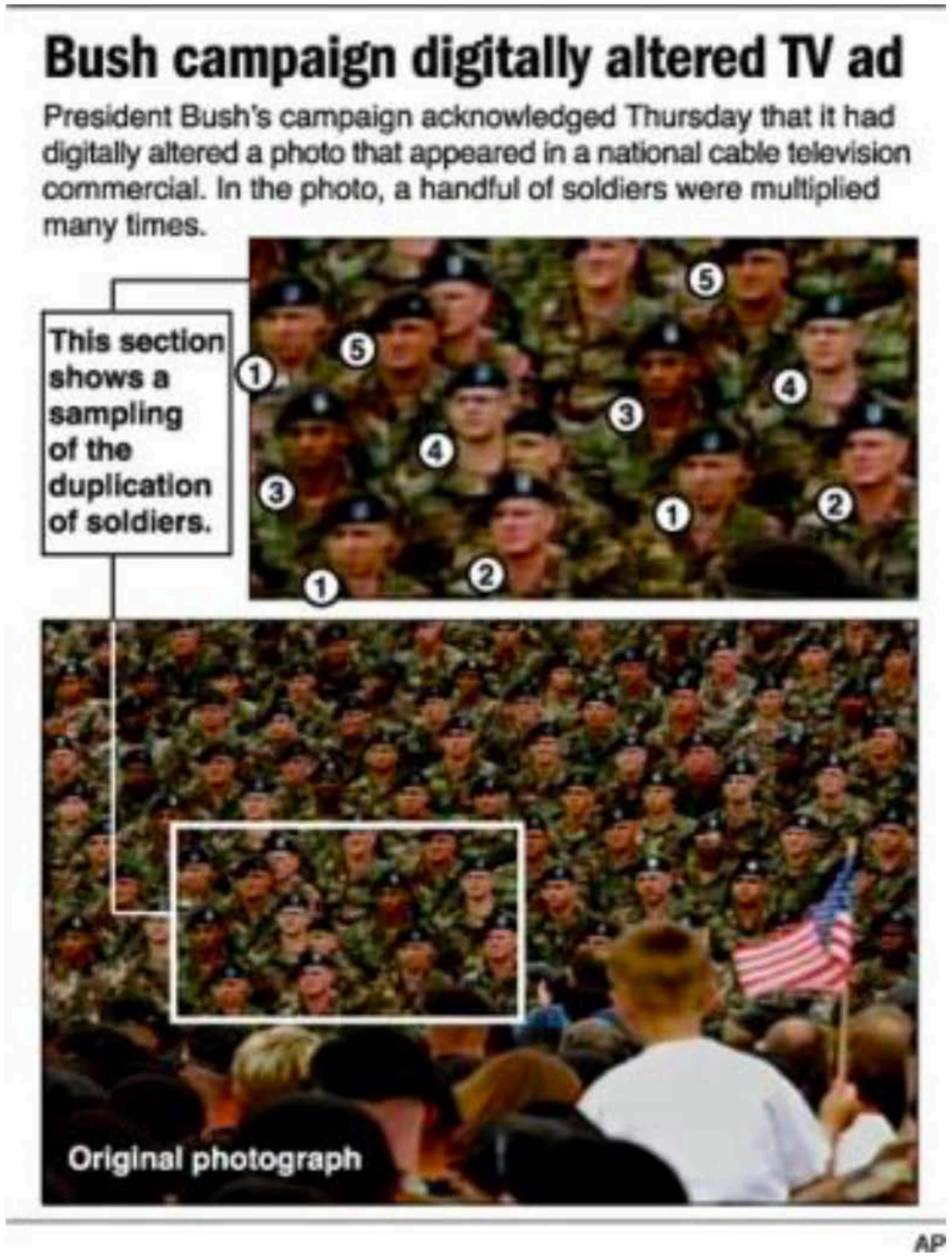


Photo Credit: Associated Pres

Texture **Synthesis**

Cover of "The Economist," June 19, 2010

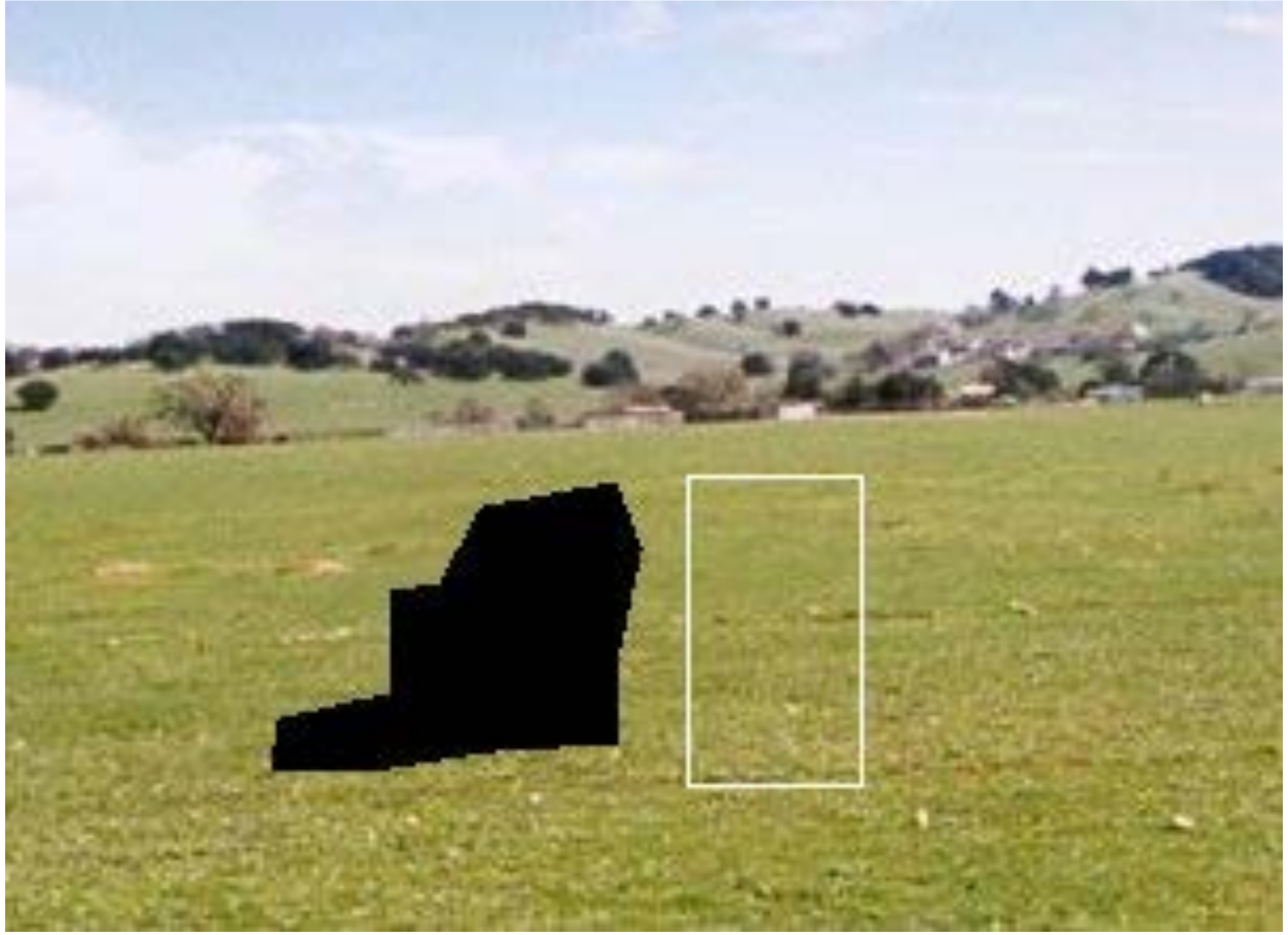
Photo Credit (right): Reuters/Larry Downing

Assignment 3 Preview: Texture Synthesis

Task: Make donkey vanish

Assignment 3 Preview: Texture Synthesis

Task: Make donkey vanish



Method: Fill-in regions using texture from the white box

Assignment 3 Preview: Texture Synthesis

Task: Make donkey vanish

Method: Fill-in regions using texture from the white box

Texture Synthesis

Objective: Generate new examples of a texture. We take a “data-driven" approach

Idea: Use an image of the texture as the source of a probability model

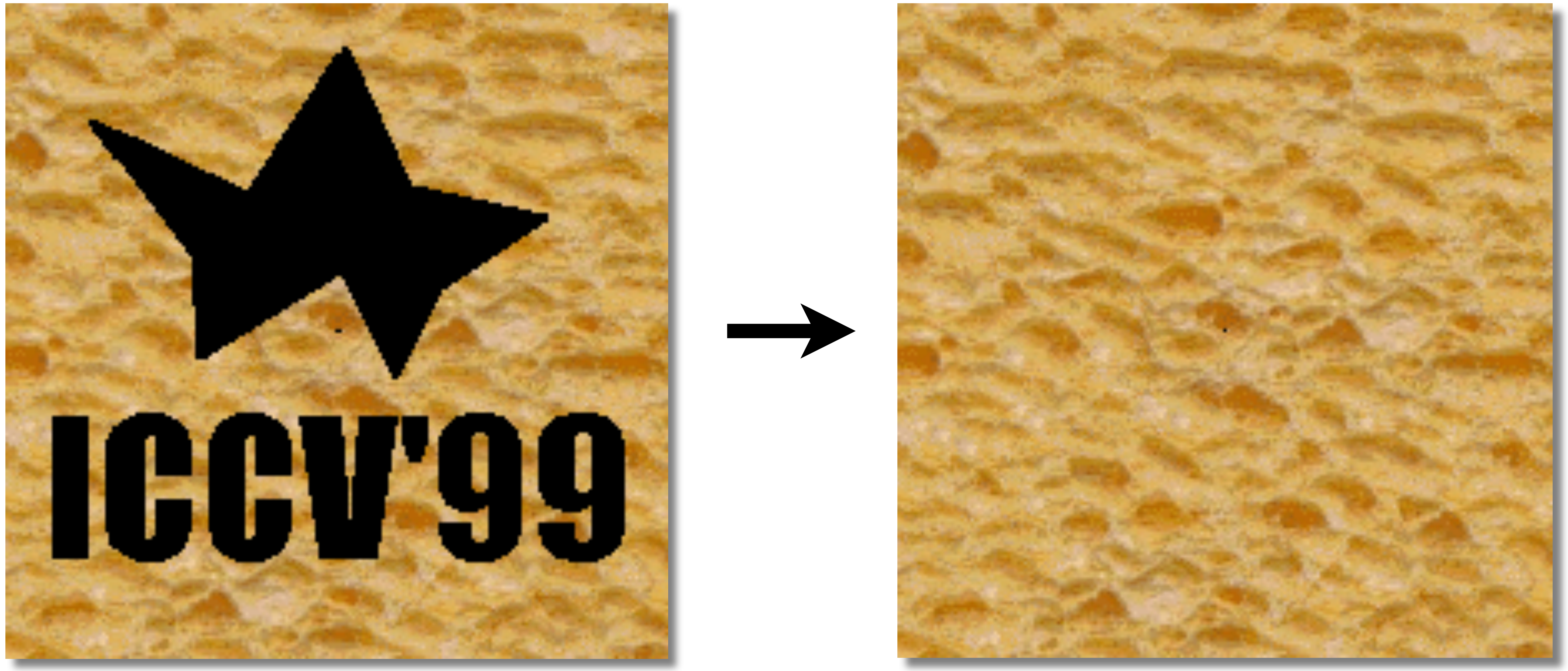
— Draw samples directly from the actual texture

— Can account for more types of structure

— Very simple to implement

— Success depends on choosing a correct “distance”

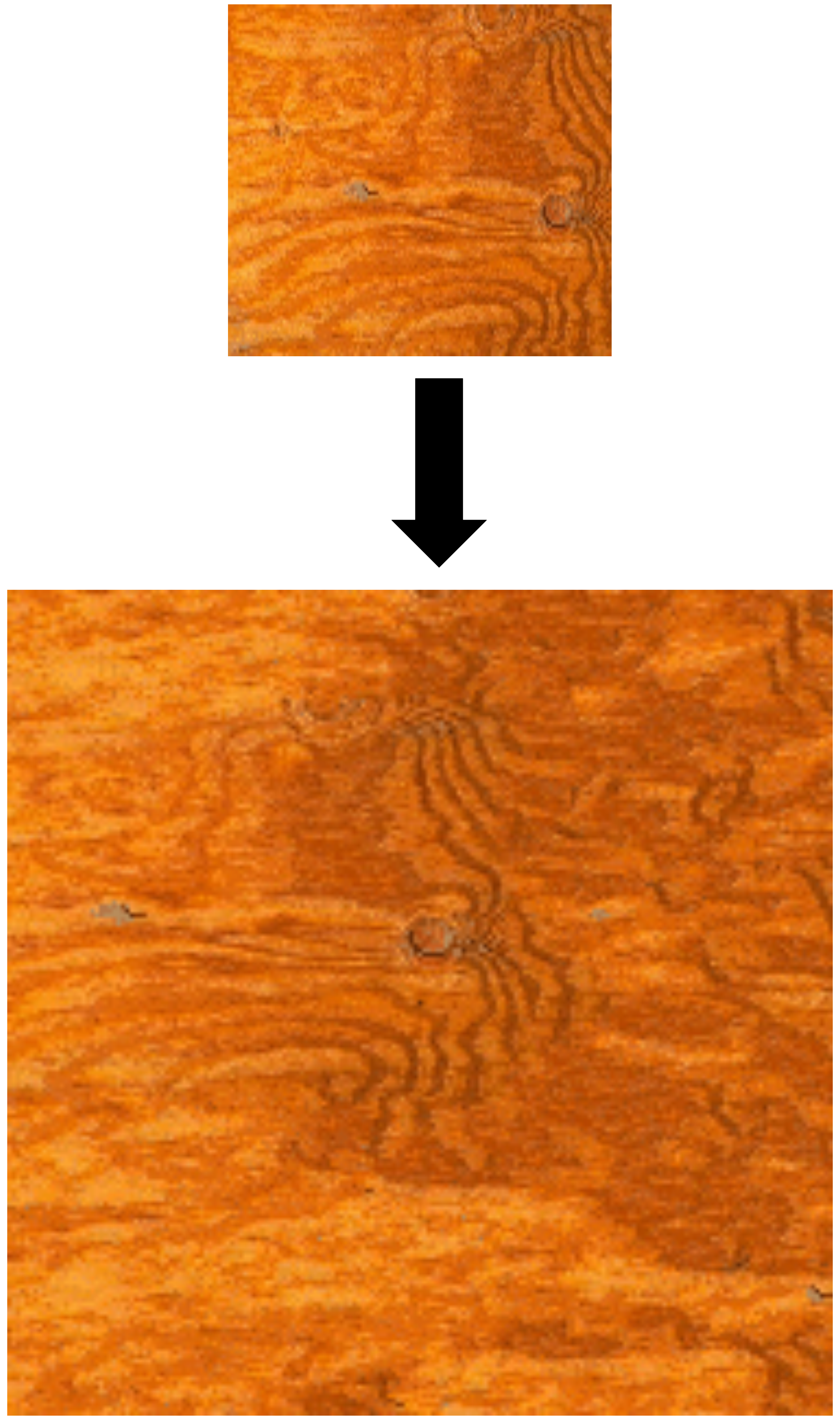
Texture Synthesis by Non-parametric Sampling



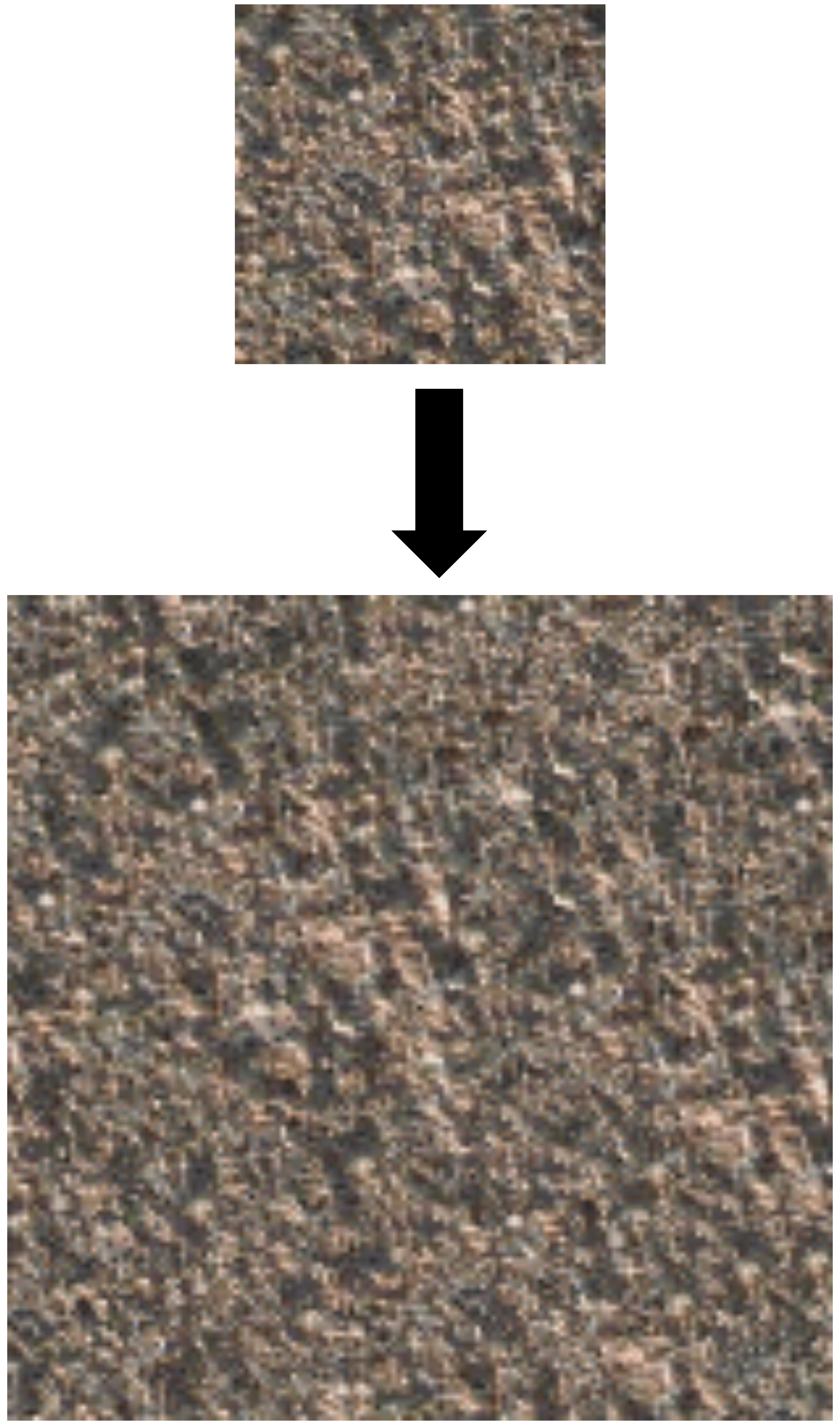
Alexei Efros and Thomas Leung
UC Berkeley

Slide Credit: http://graphics.cs.cmu.edu/people/efros/research/NPS/efros-iccv99.ppt

Efros and Leung

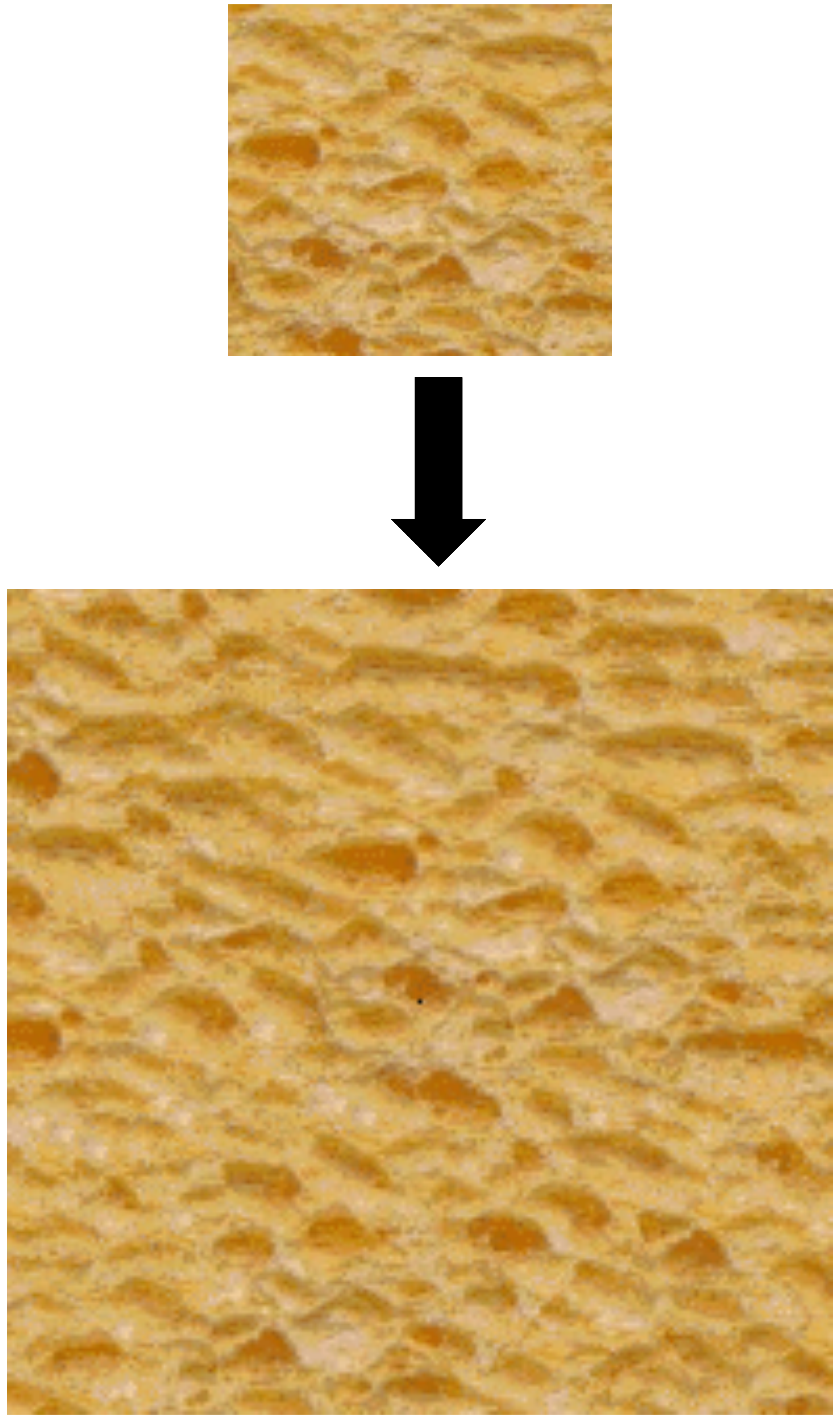


wood

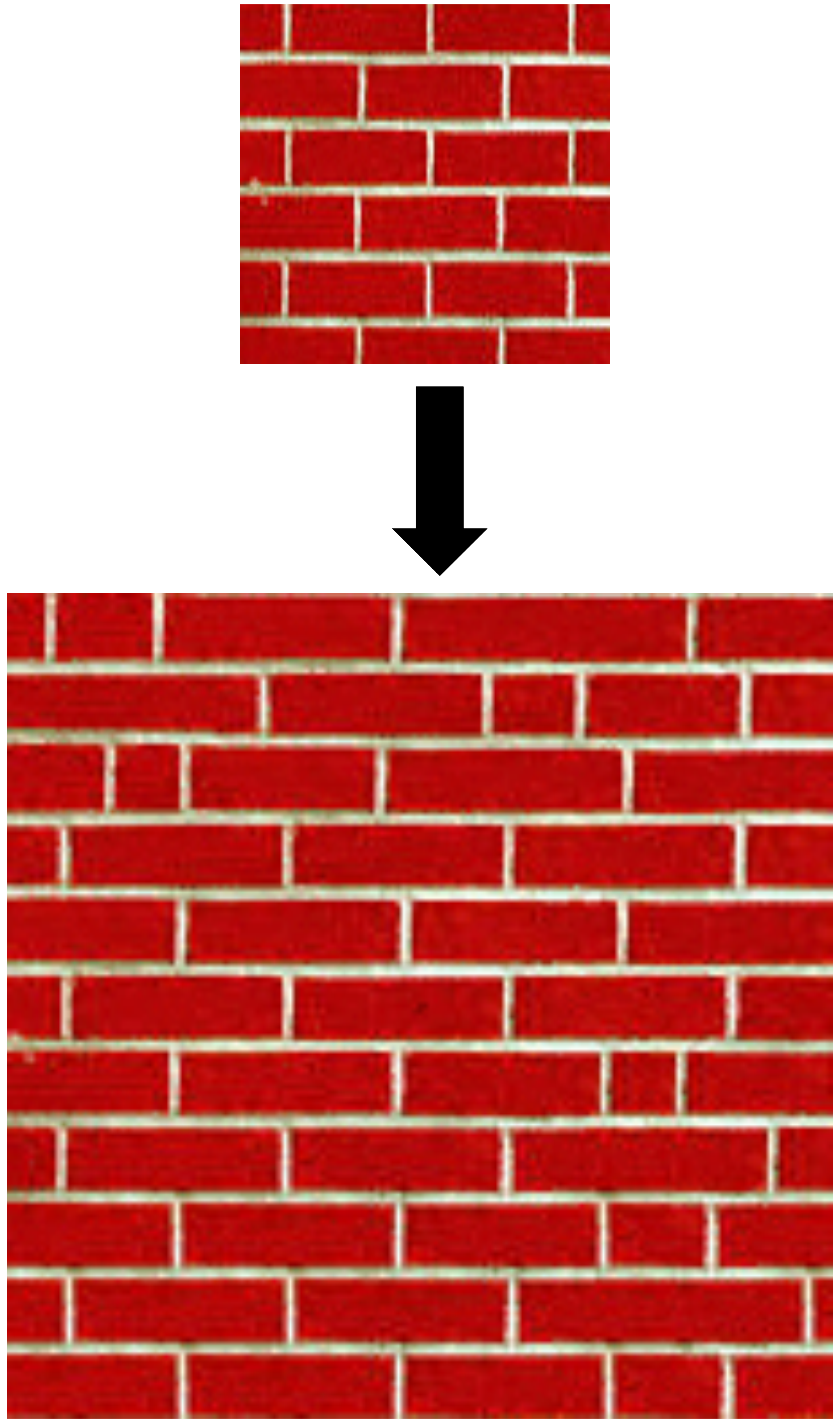


granite

Efros and Leung

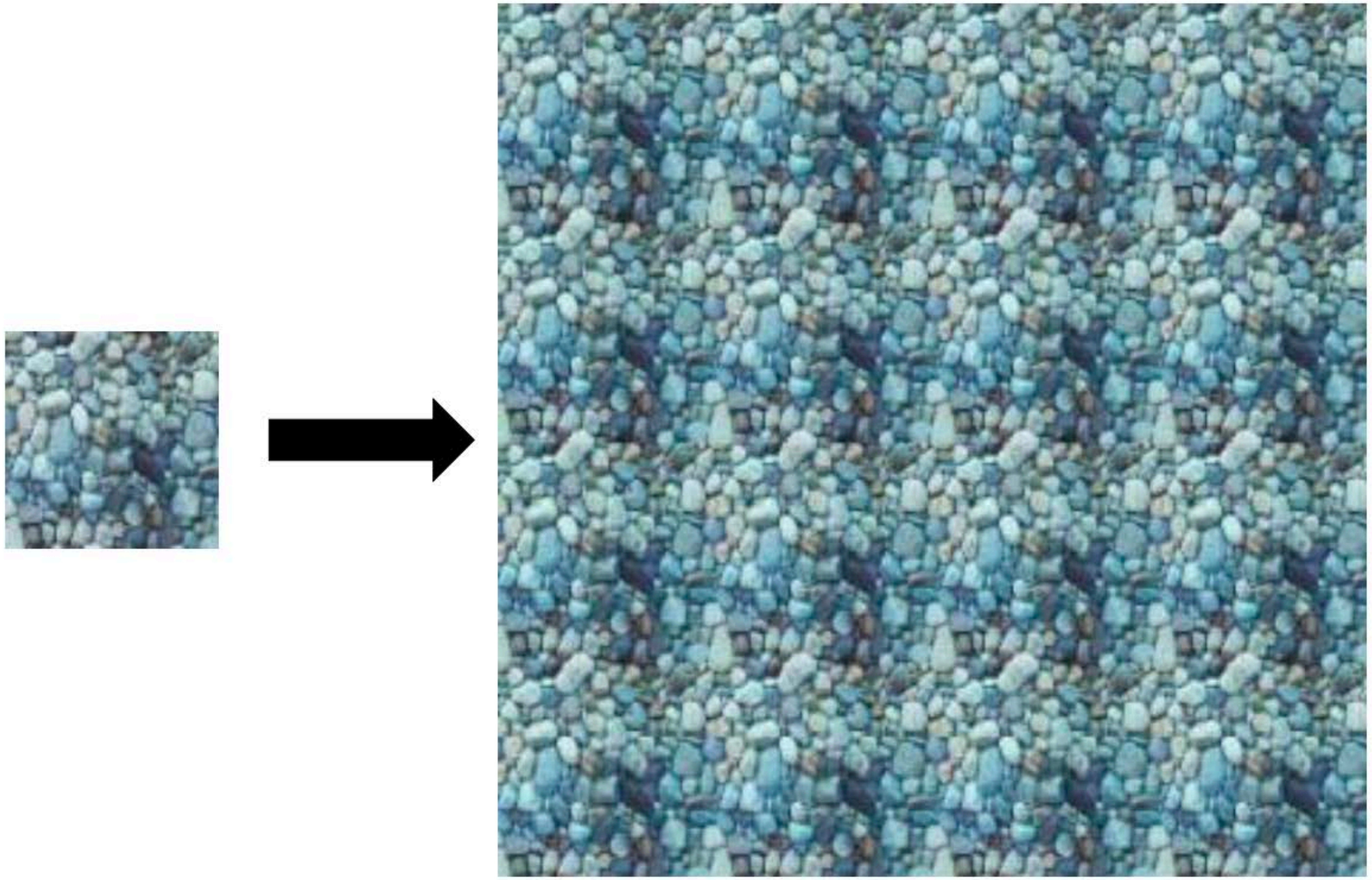


white bread

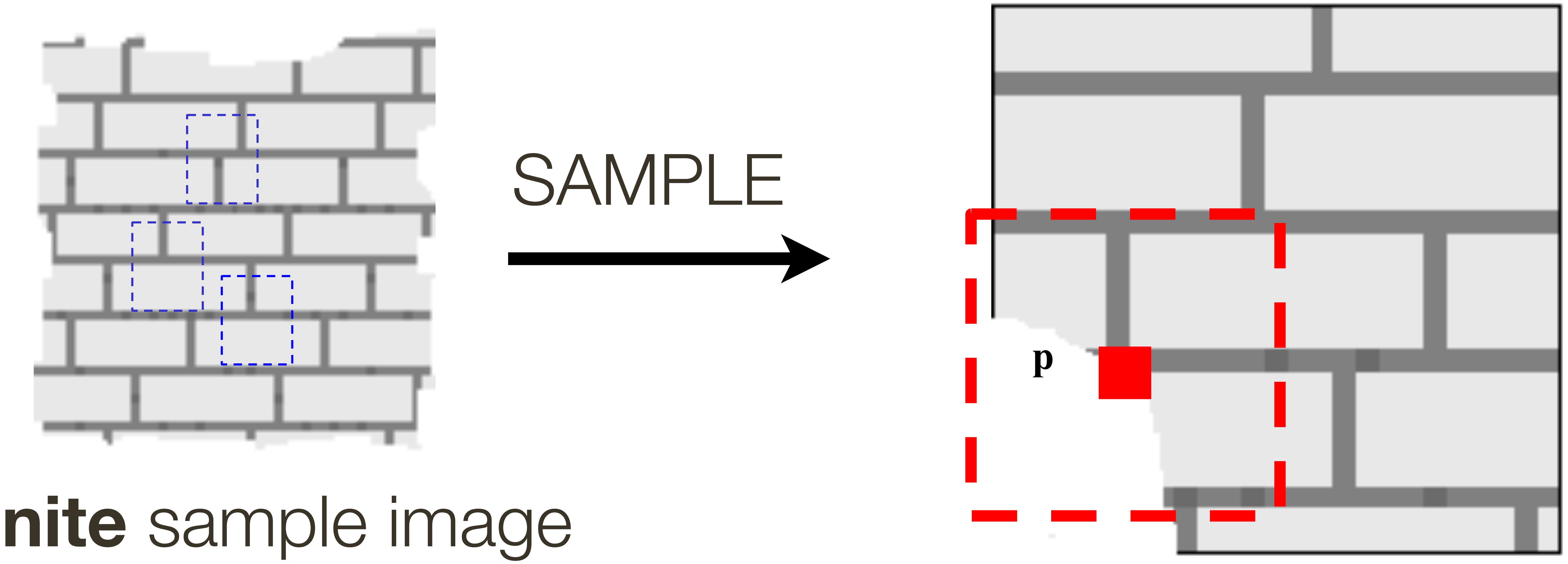


brick wall

Like **Copying**, But not Just Repetition



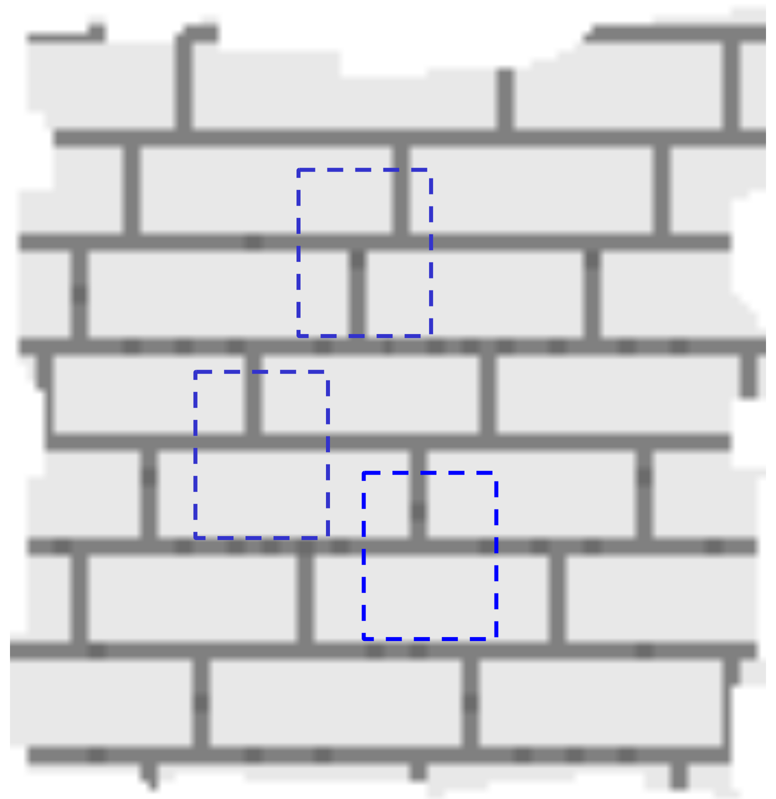
Efros and Leung: Synthesizing One Pixel



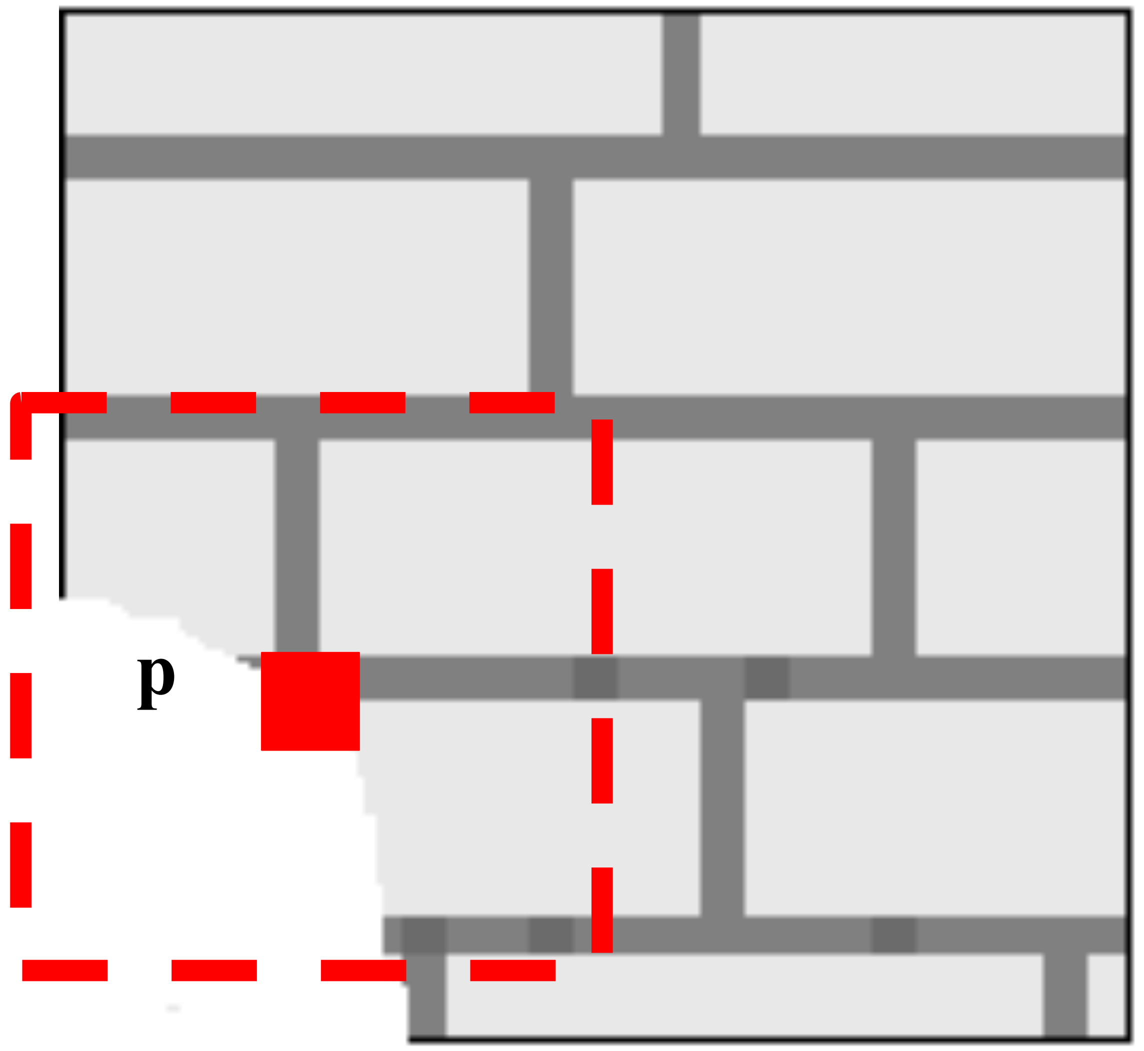
Infinite sample image

— What is **conditional** probability distribution of *p*, given the neighbourhood window?

Efros and Leung: Synthesizing One Pixel



SAMPLE

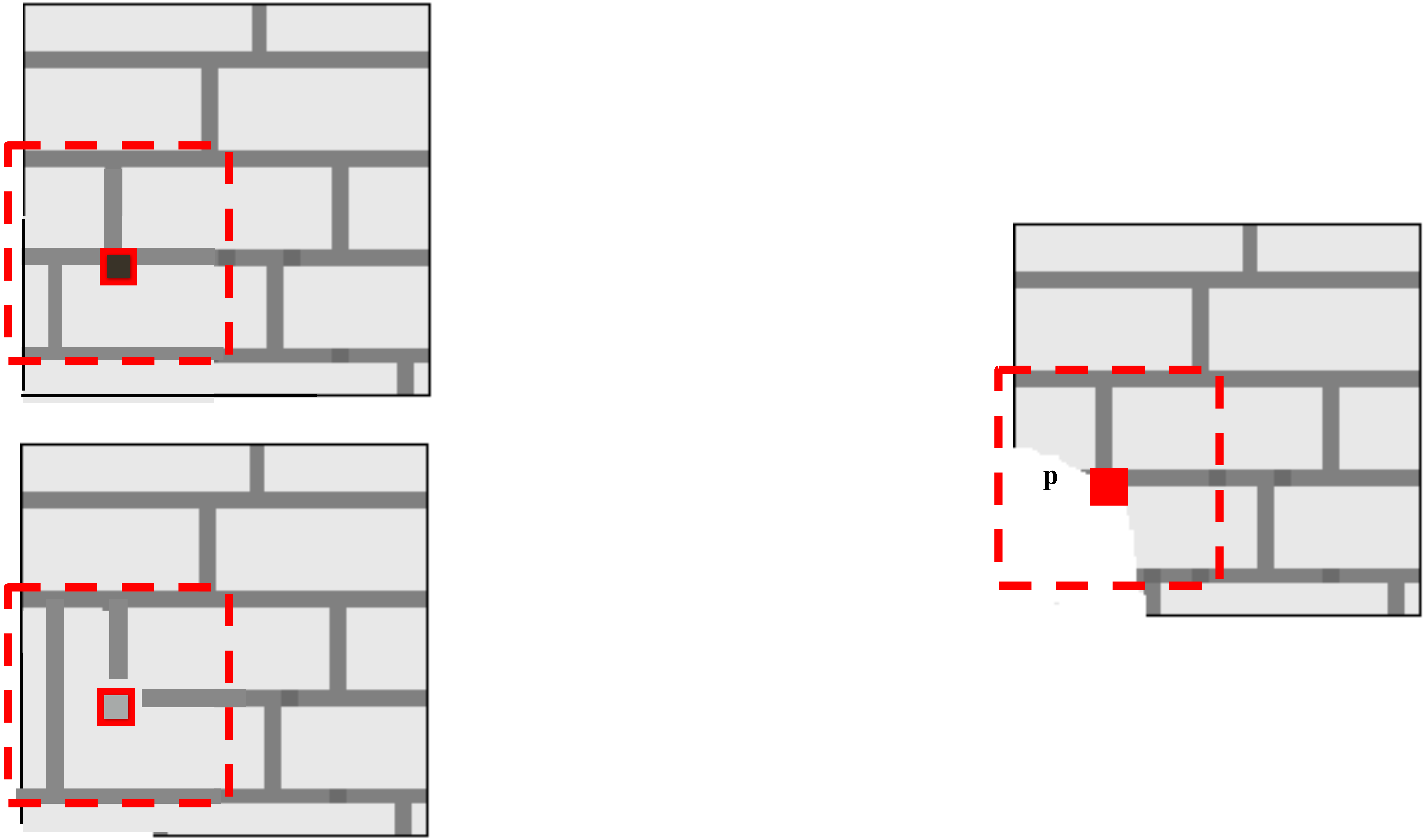


Infinite sample image

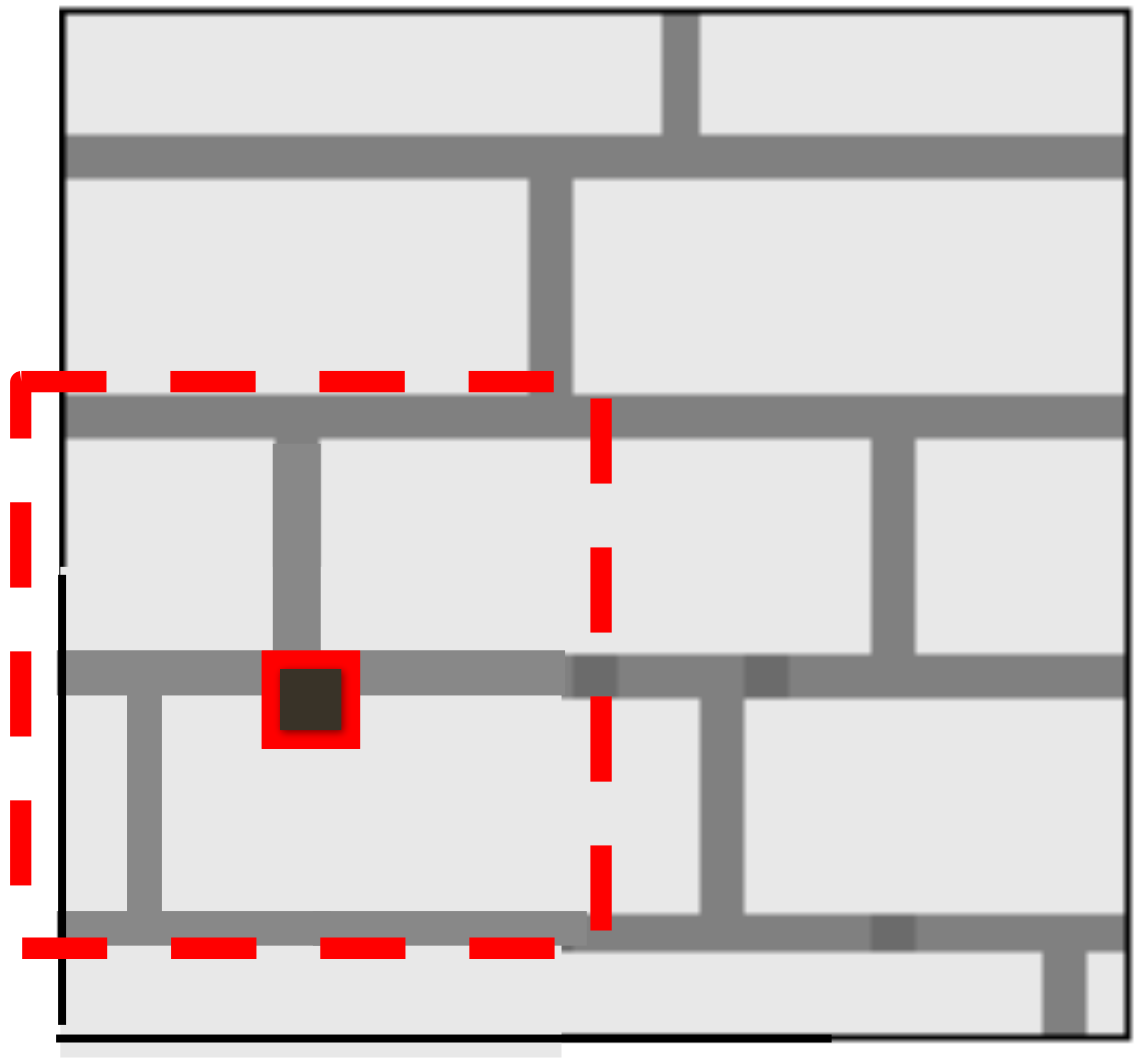
— What is **conditional** probability distribution of *p*, given the neighbourhood window?

— Directly search the input image for all such neighbourhoods to produce a **histogram** for *p*

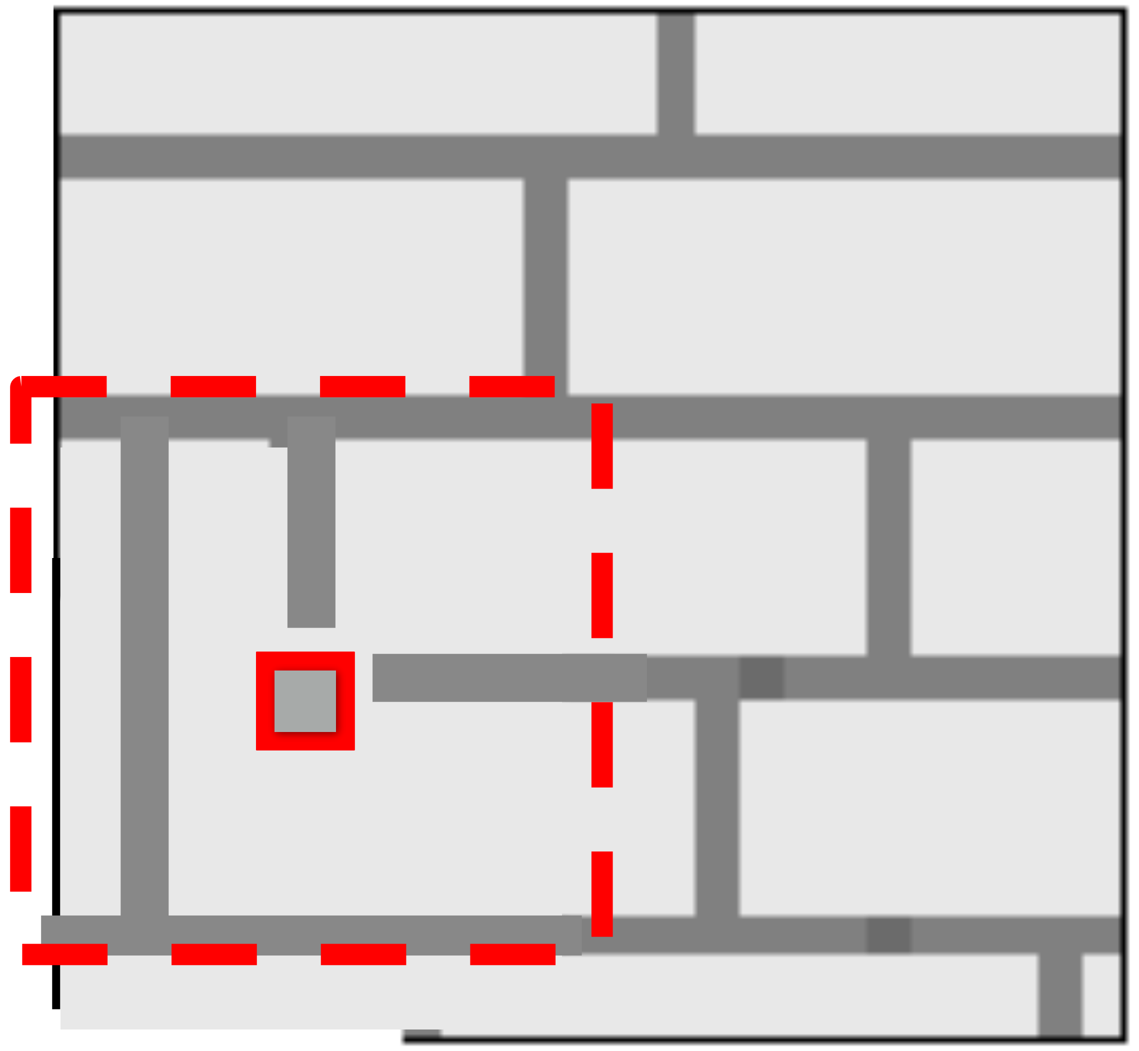
Efros and Leung: Synthesizing One Pixel



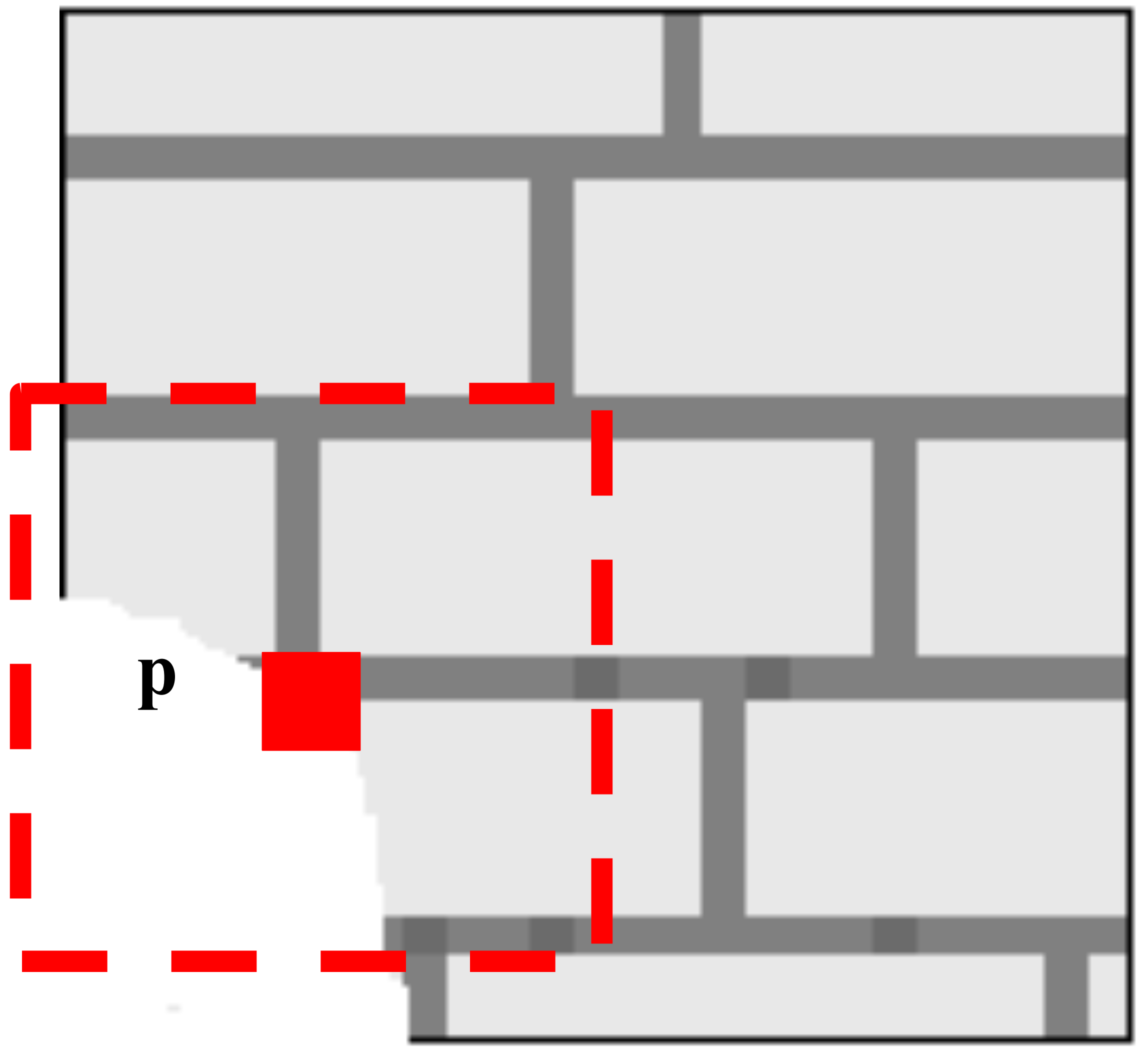
Efros and Leung: Synthesizing One Pixel



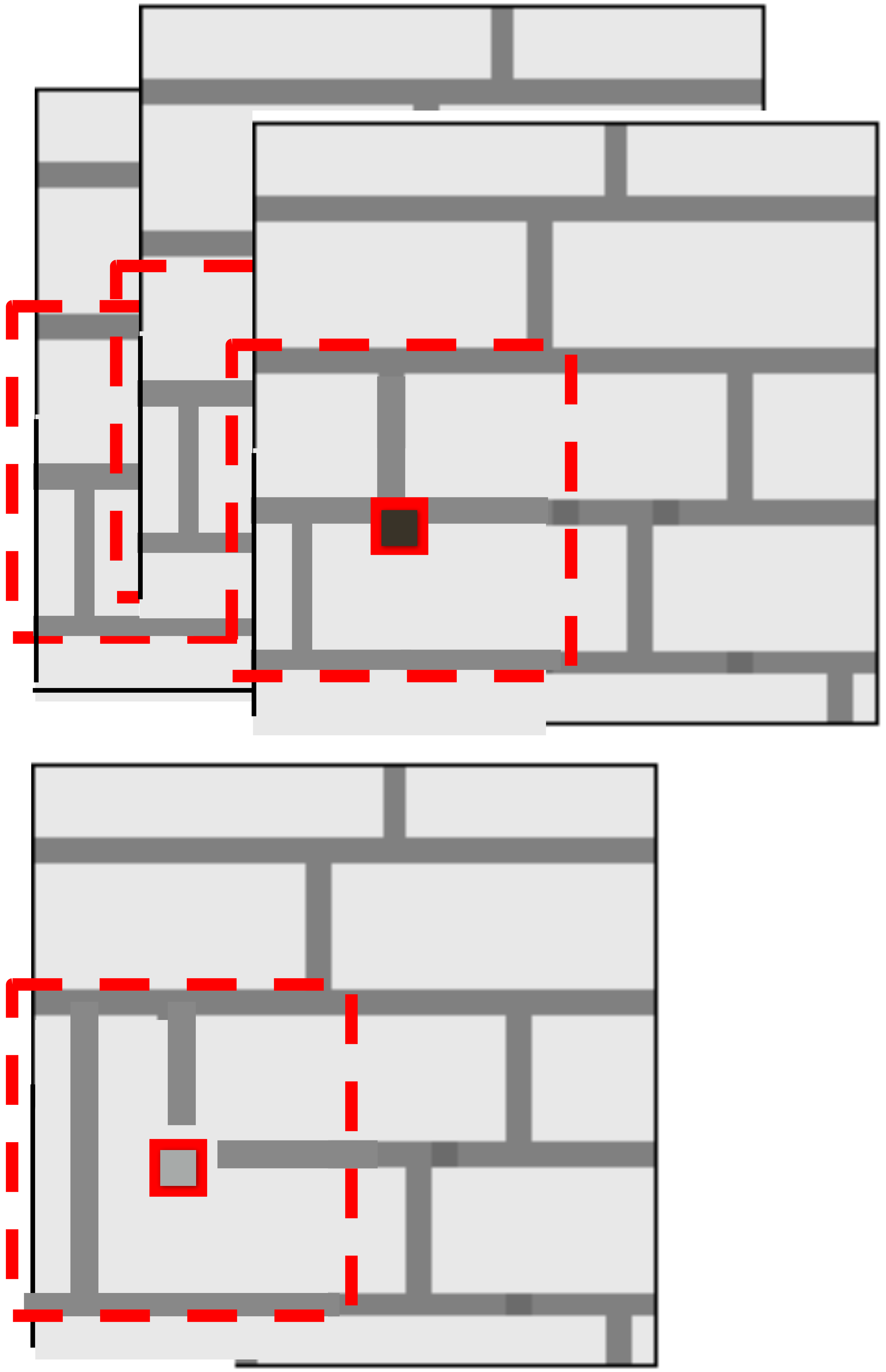
p(dark gray) = 0.5

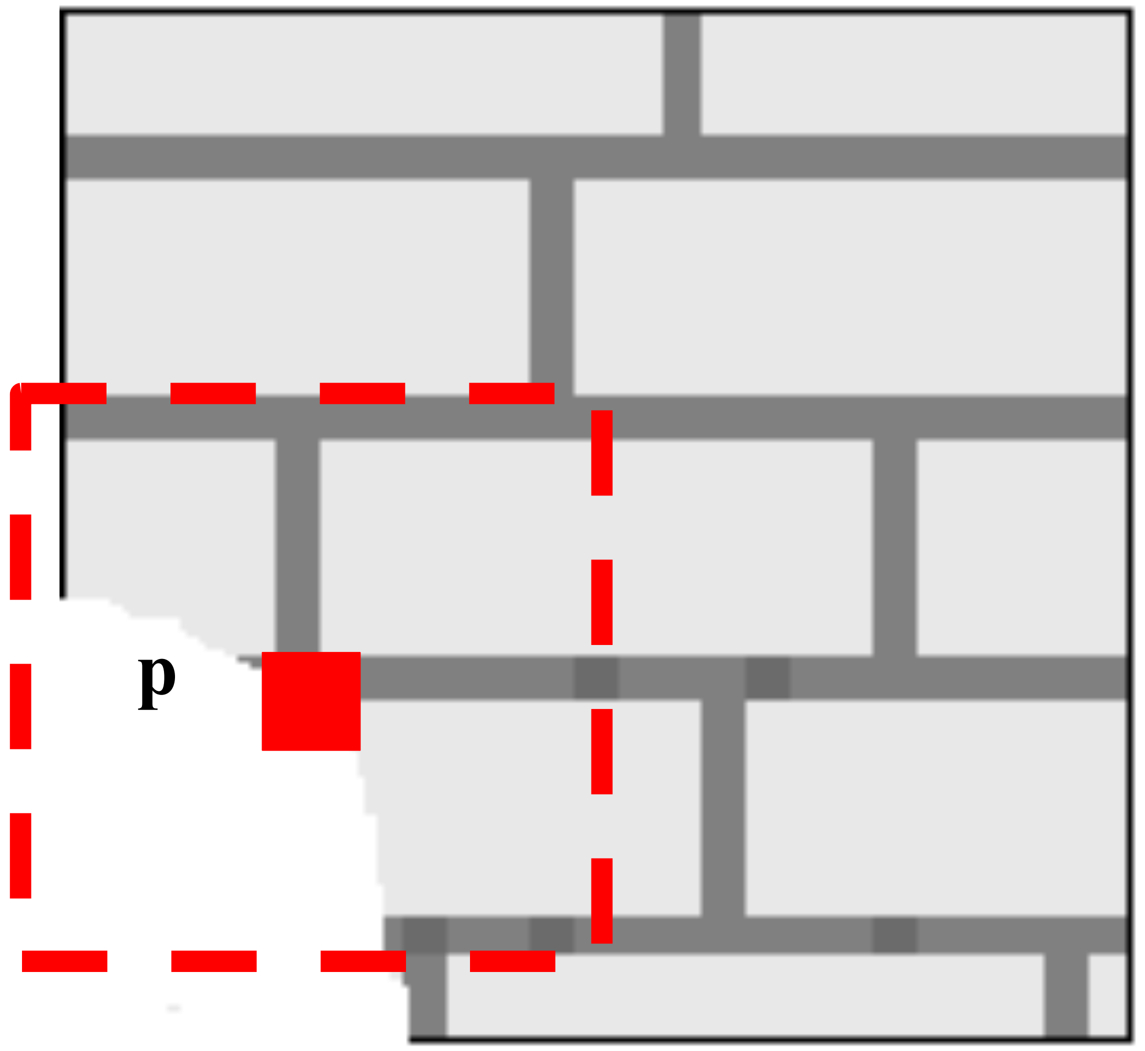


p(light gray) = 0.5

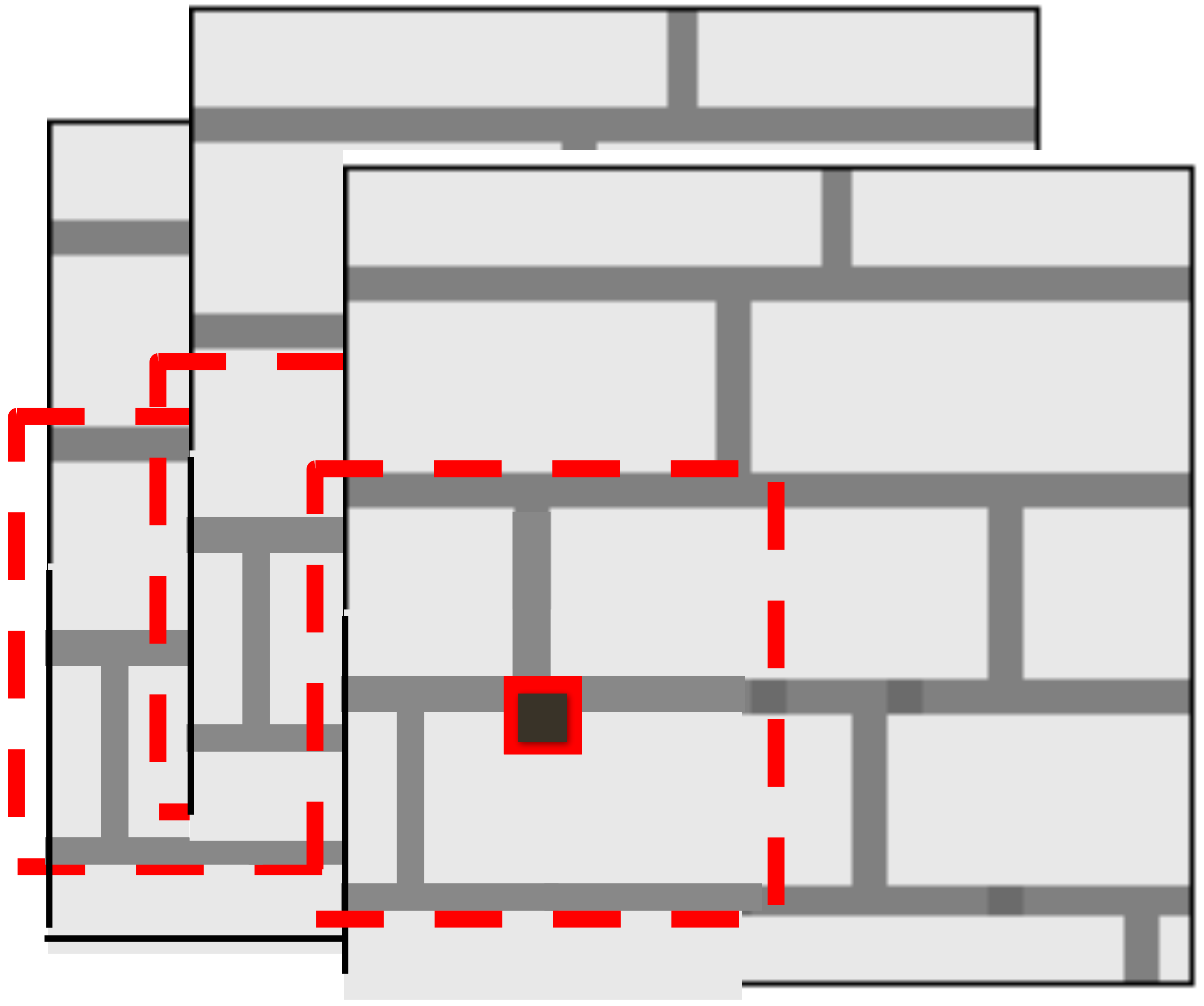


Efros and Leung: Synthesizing One Pixel

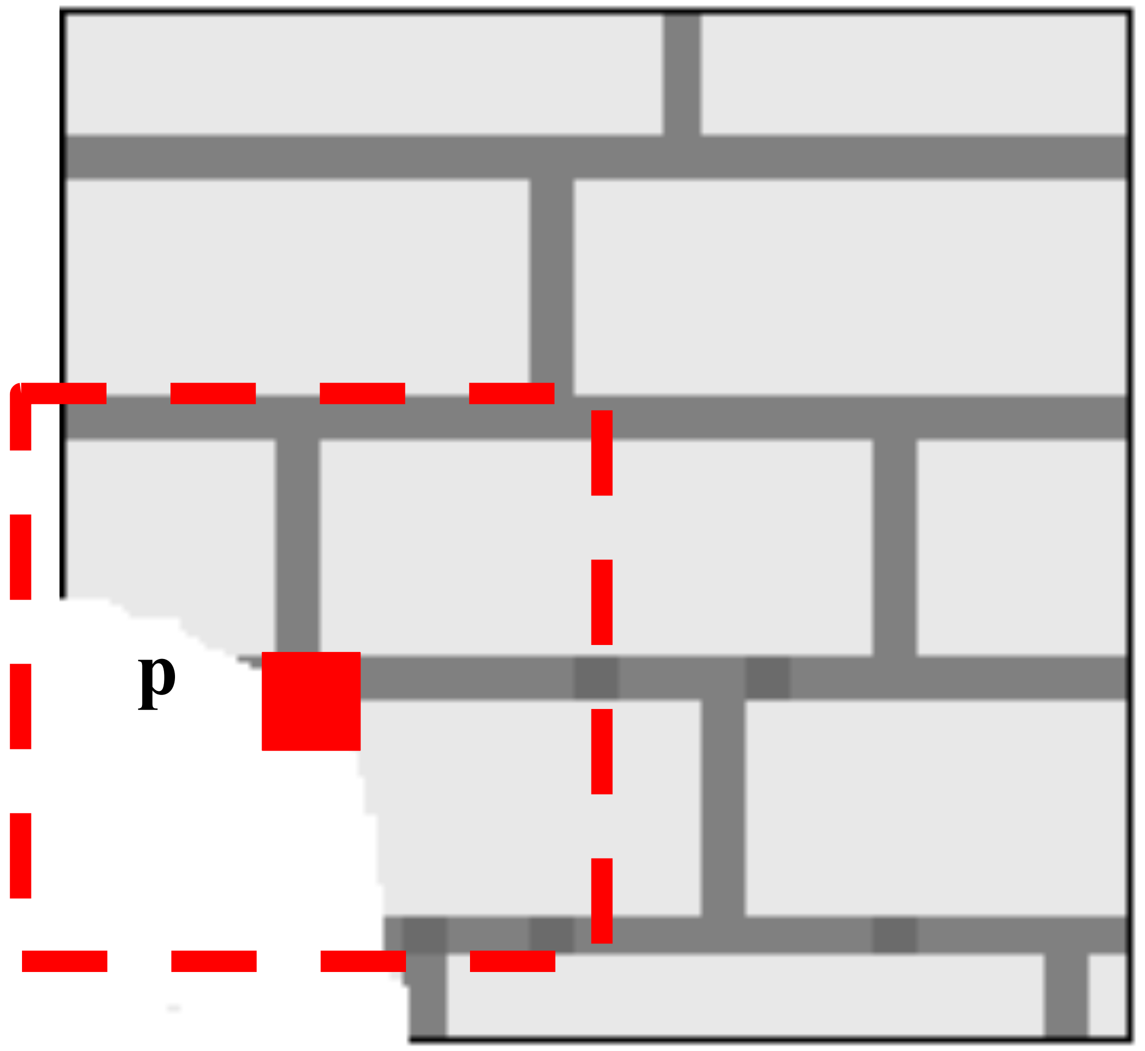


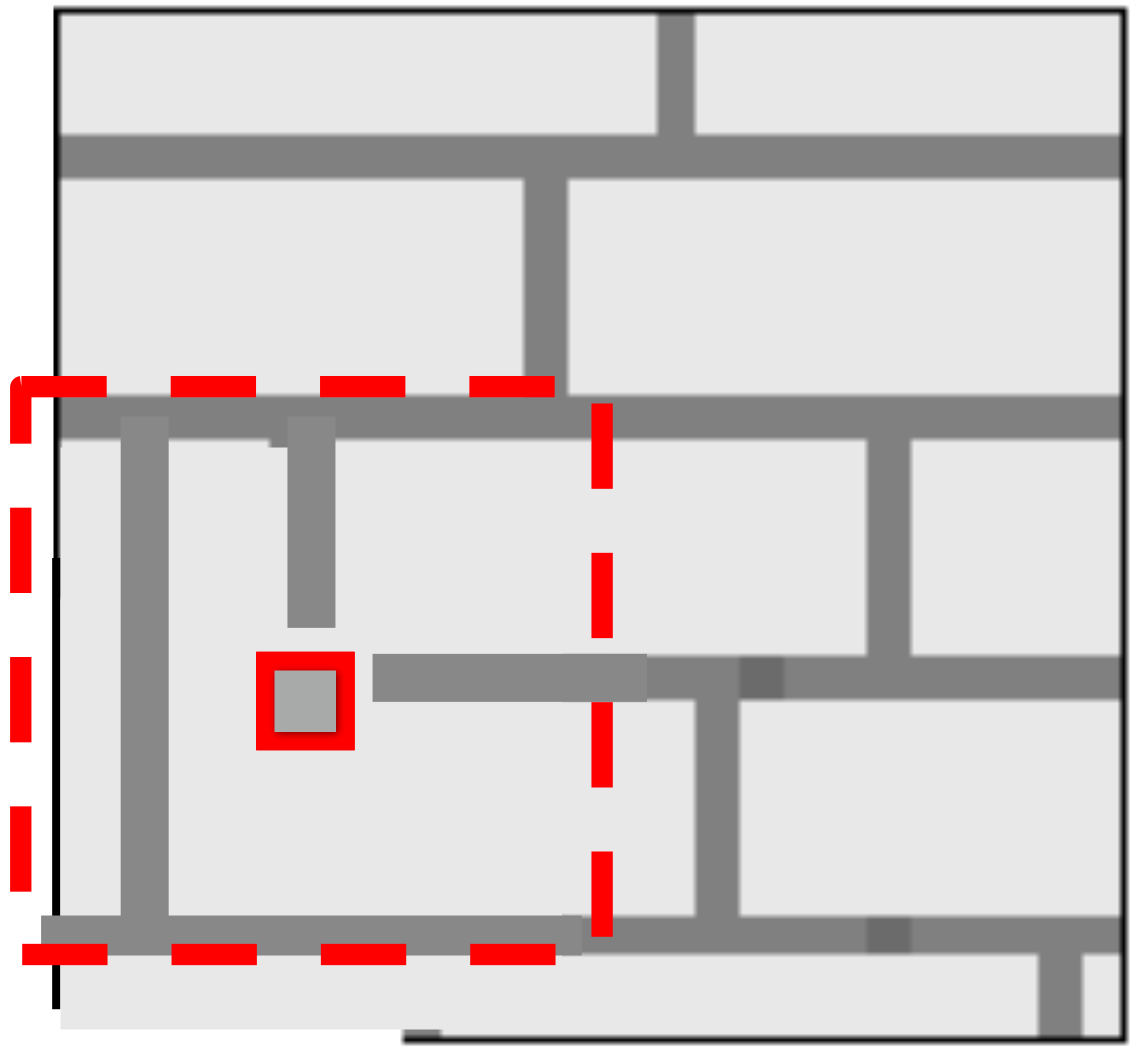


Efros and Leung: Synthesizing One Pixel



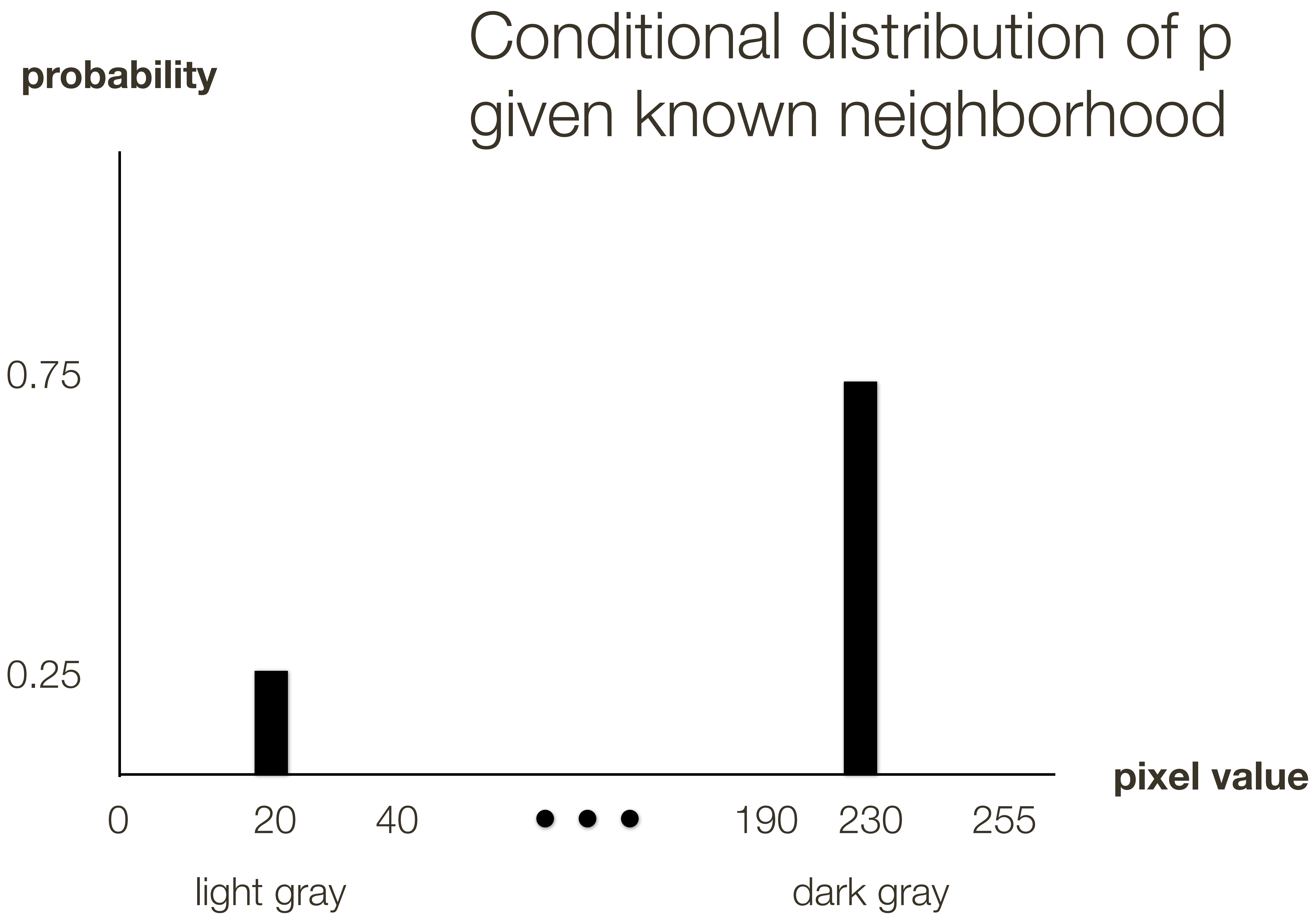
p(dark gray) = 0.75

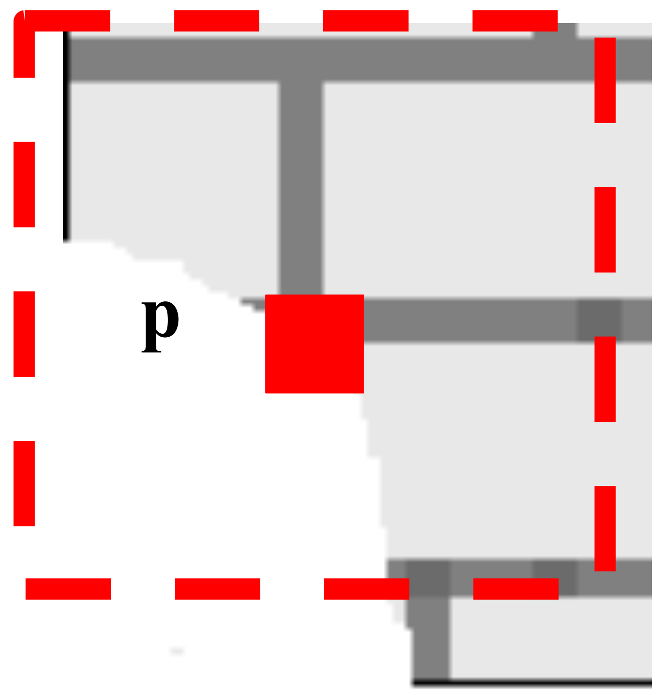




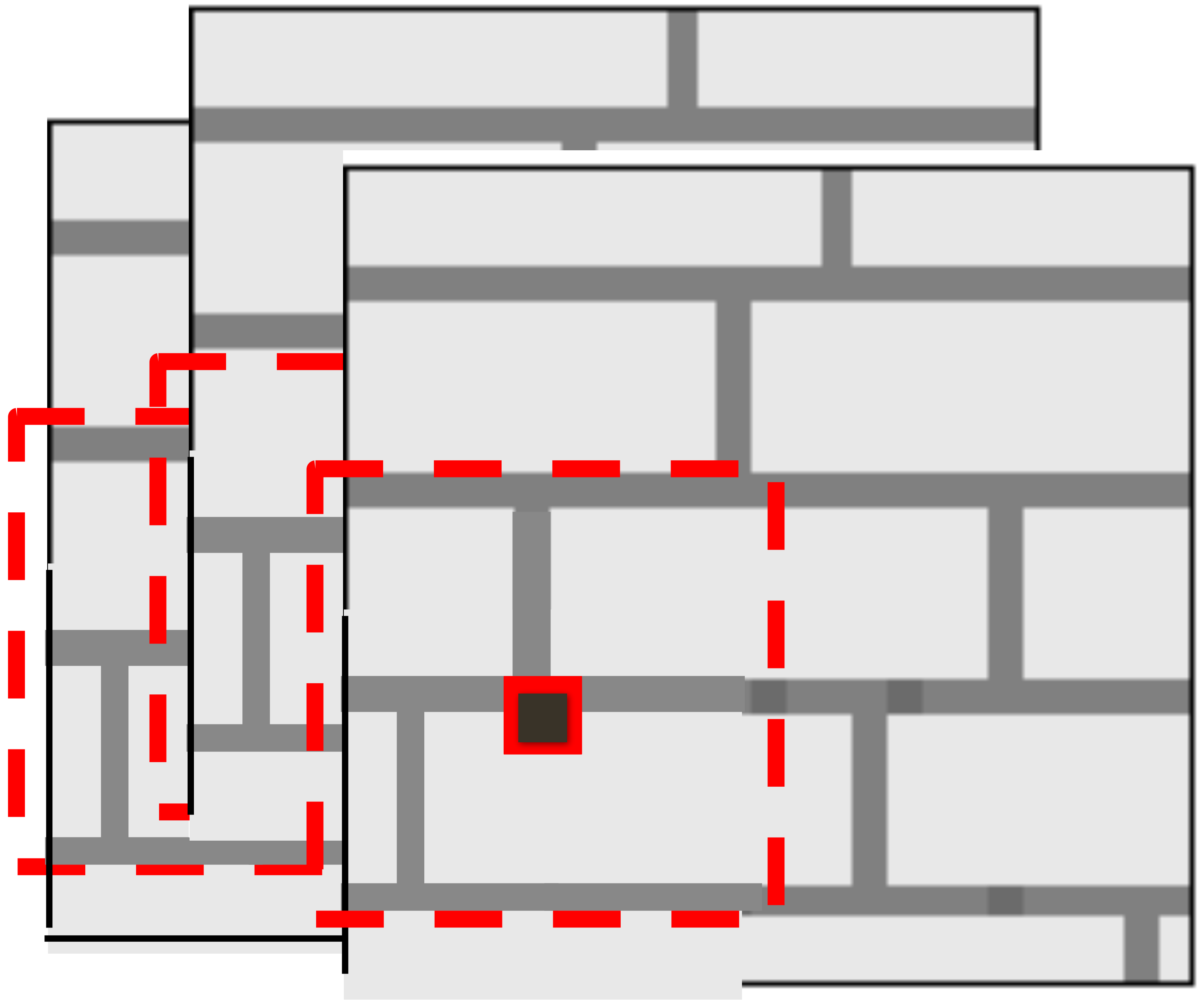
p(light gray) = 0.25

Efros and Leung: Synthesizing One Pixel

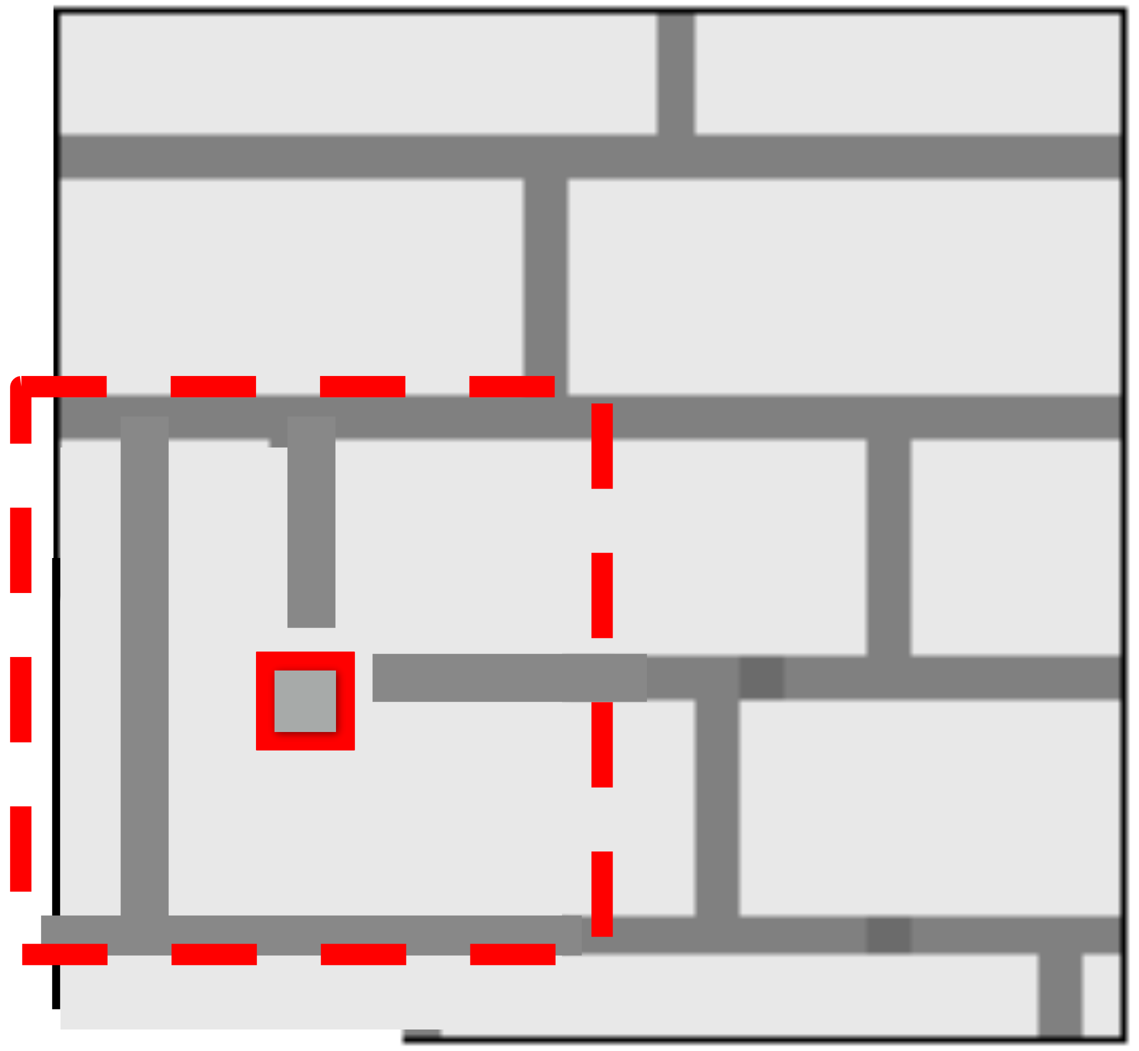




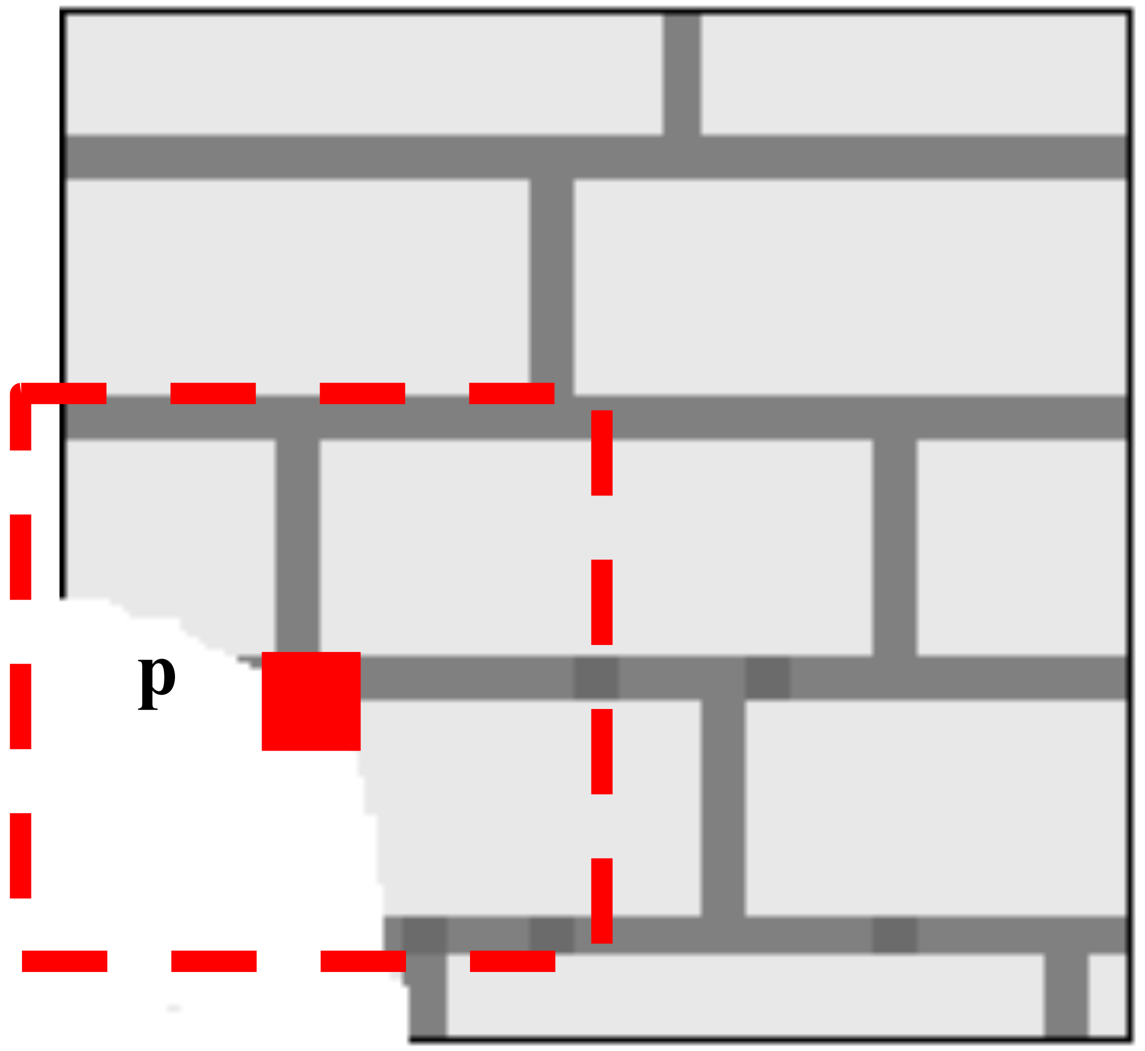
Efros and Leung: Synthesizing One Pixel



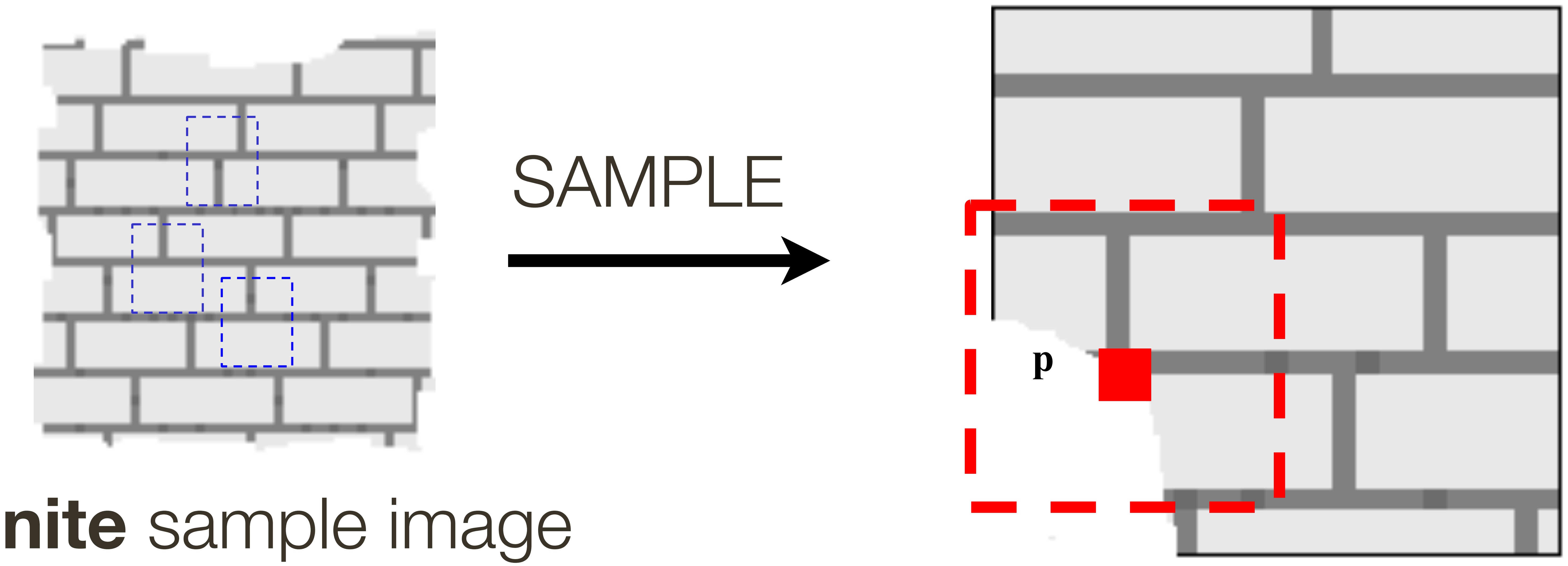
p(dark gray) = 0.75



p(light gray) = 0.25



Efros and Leung: Synthesizing One Pixel



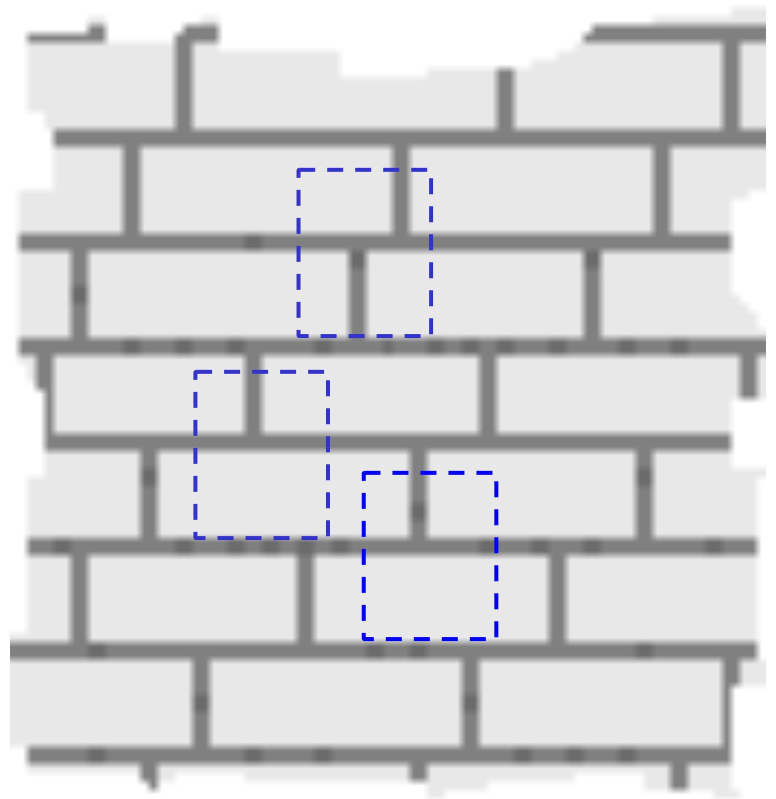
Infinite sample image

— What is **conditional** probability distribution of *p*, given the neighbourhood window?

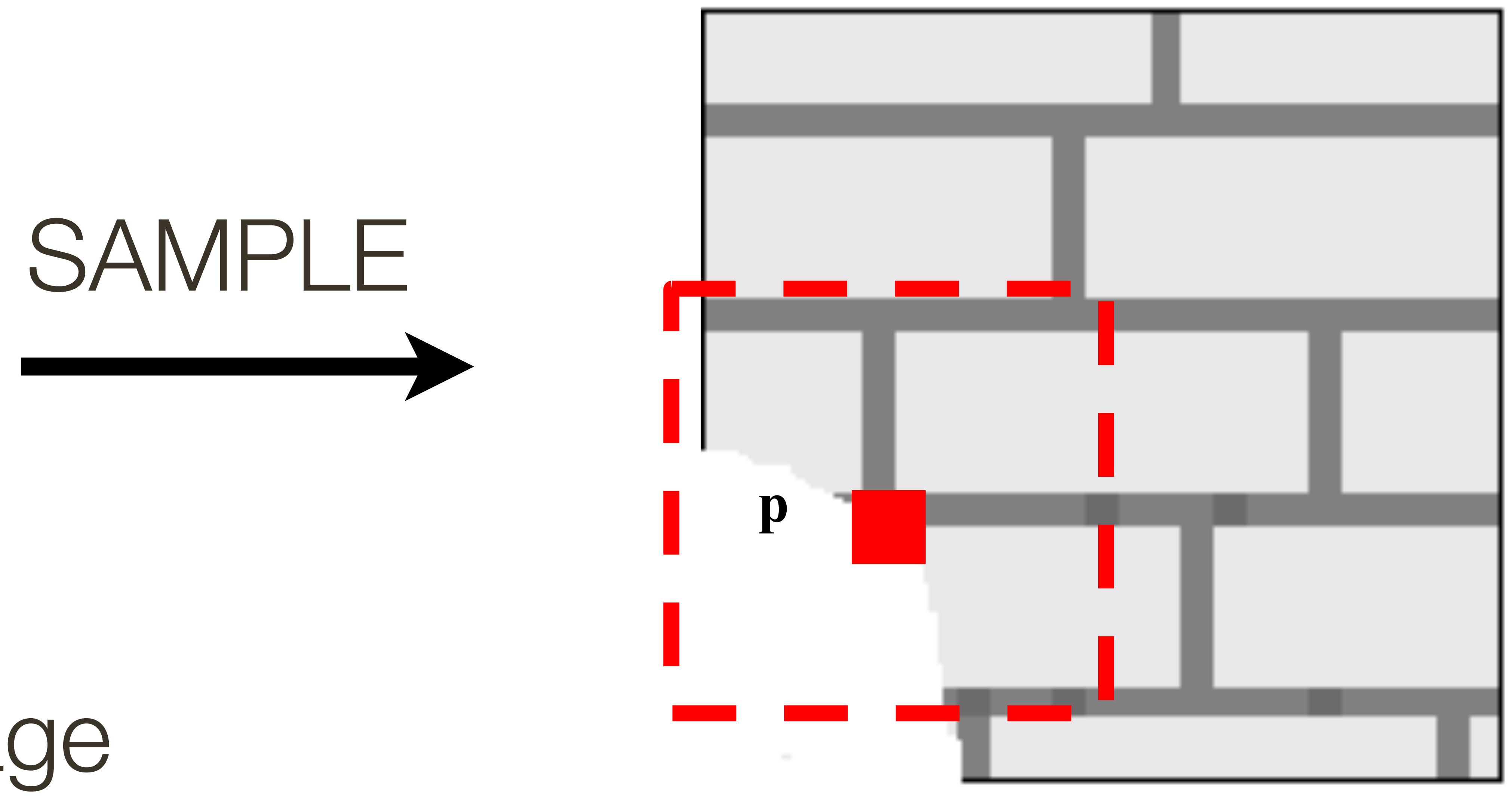
— Directly search the input image for all such neighbourhoods to produce a **histogram** for *p*

— To **synthesize** *p*, pick one match at random

Efros and Leung: Synthesizing One Pixel

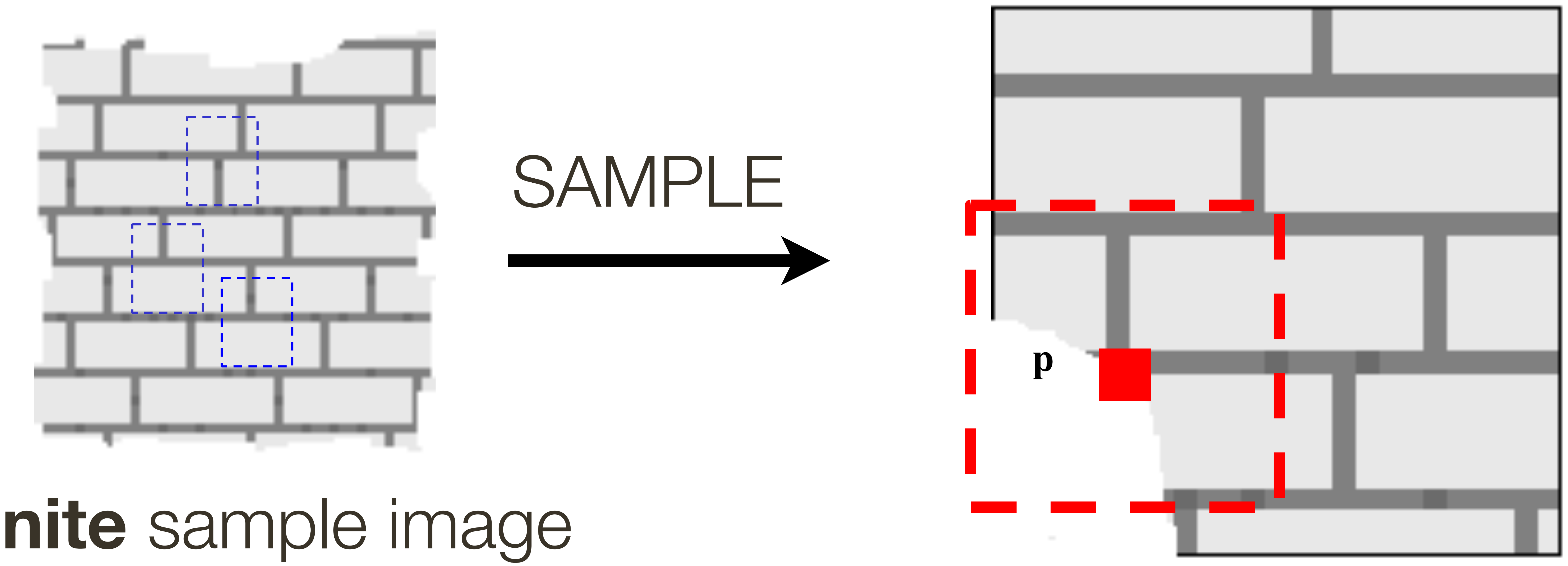


Infinite sample image



— Since the sample image is finite, an exact neighbourhood match might not be present

Efros and Leung: Synthesizing One Pixel

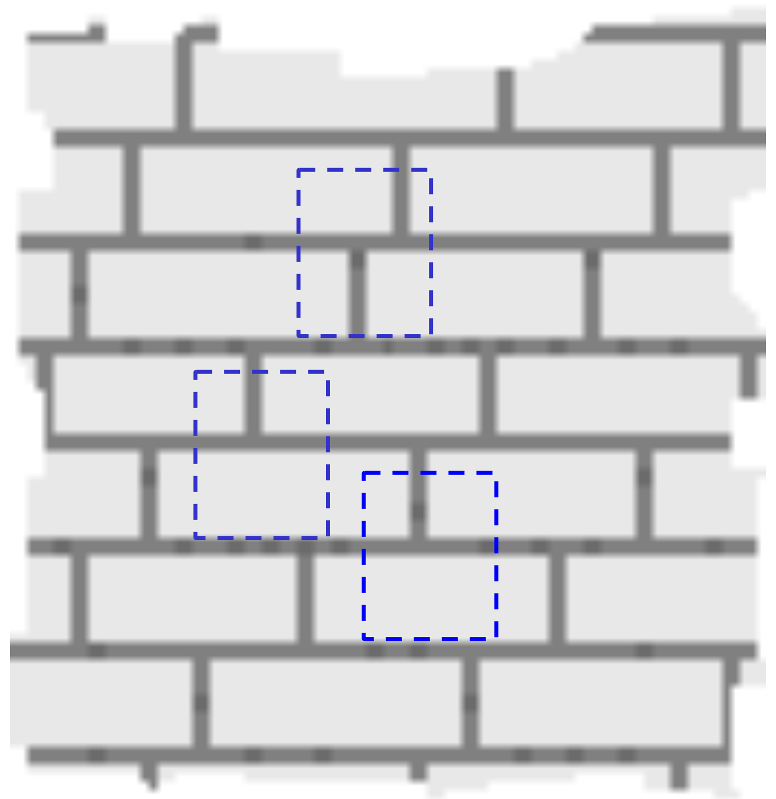


Infinite sample image

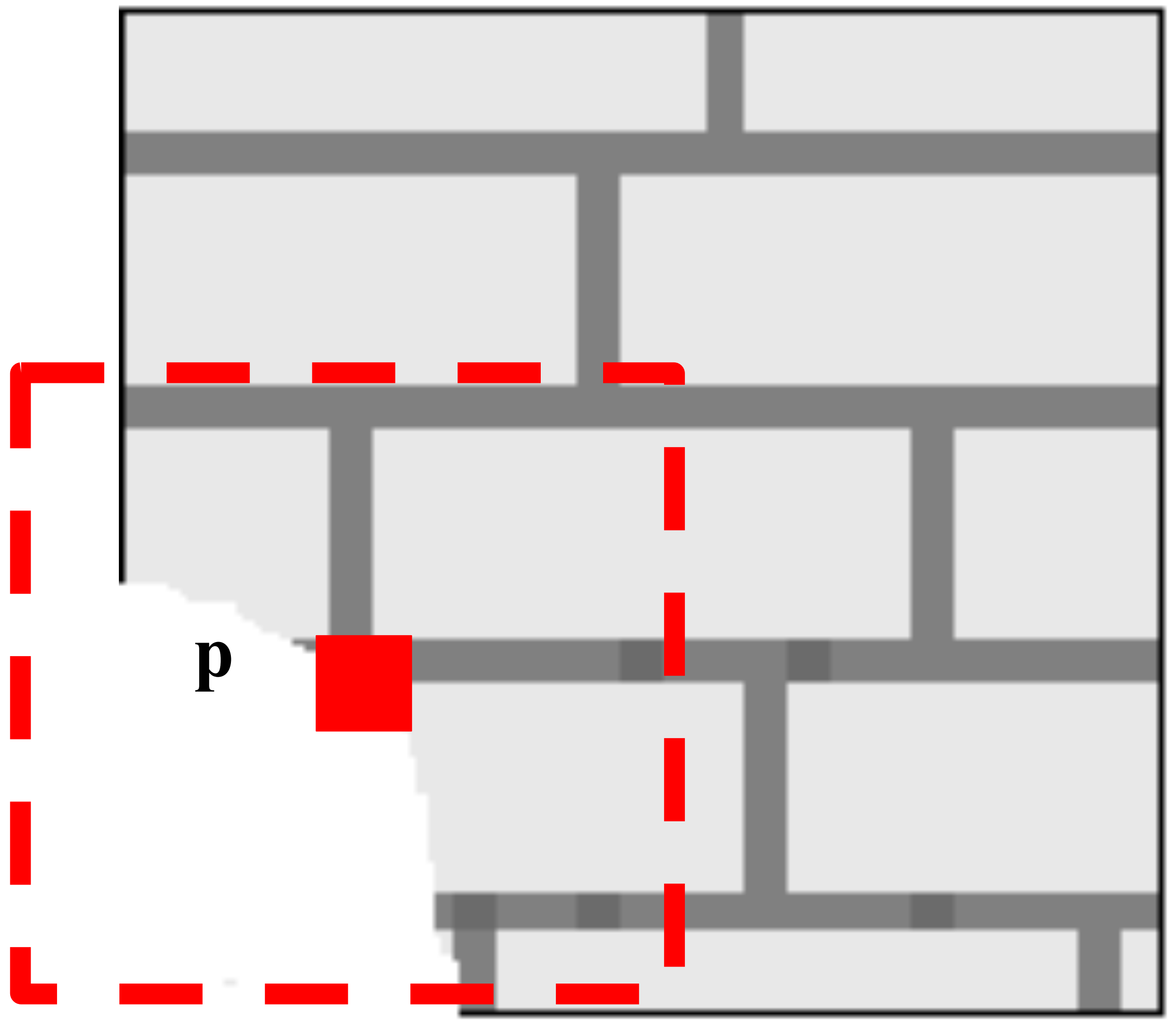
— Since the sample image is finite, an exact neighbourhood match might not be present

— Find the **best match** using SSD error, weighted by Gaussian to emphasize local structure, and take all samples within some distance from that match

Efros and Leung: Synthesizing One Pixel



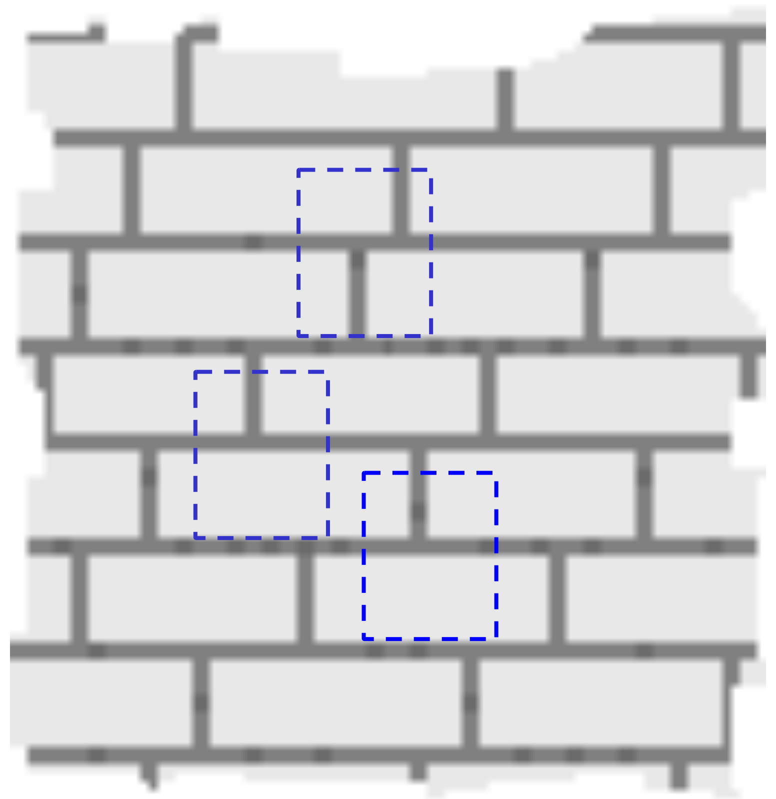
SAMPLE



Infinite sample image

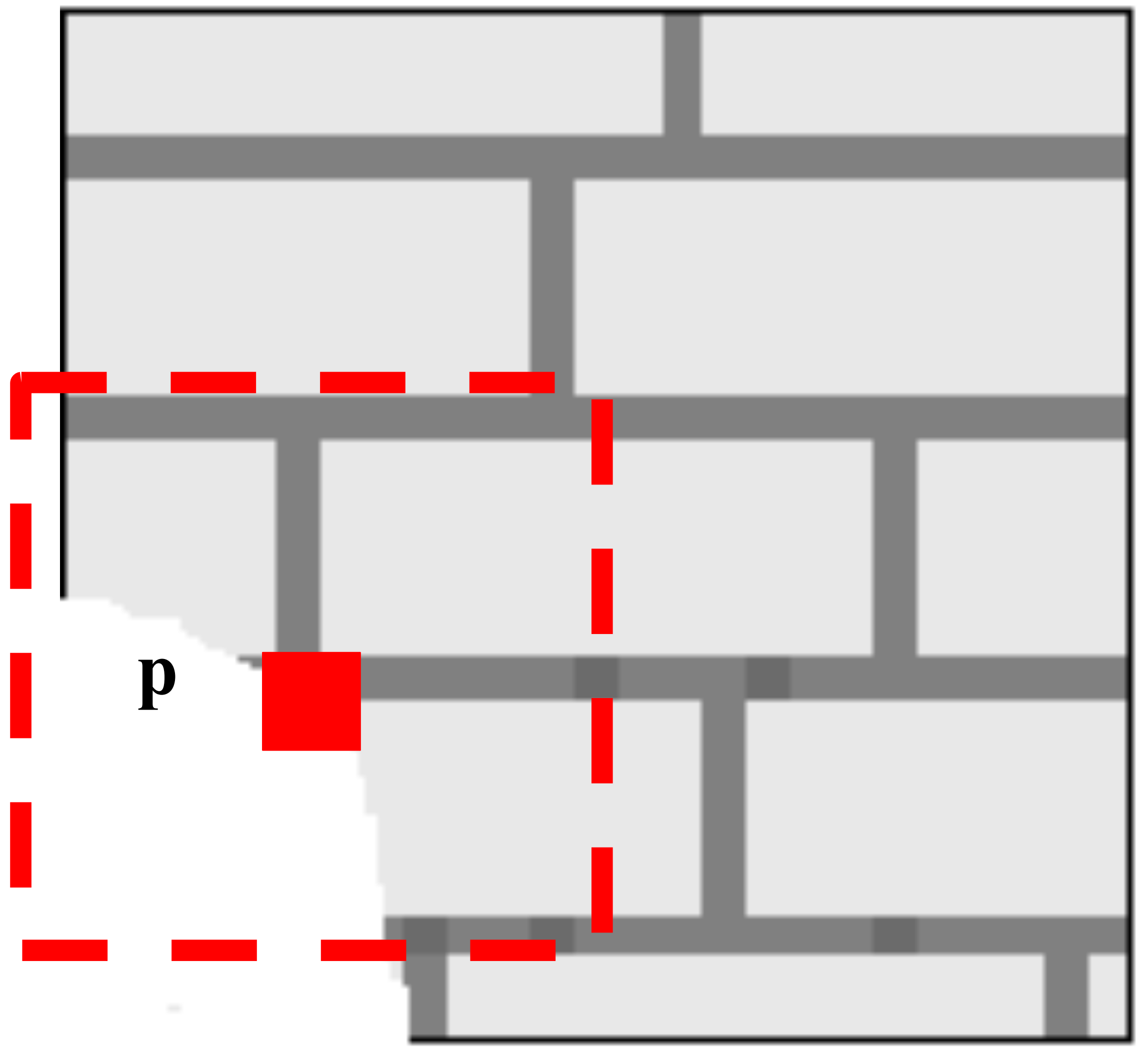
Ranked List	Similarity (cos)	
x = 5, y = 17	0.87	← best match
x = 63, y = 4	0.75	
x = 3, y = 44	0.72	
		threshold = best match * **0.8** = 0.696
x = 123, y = 54	0.64	
x = 4, y = 57	0.60	
⋮	⋮	

Efros and Leung: Synthesizing One Pixel



Infinite sample image

SAMPLE



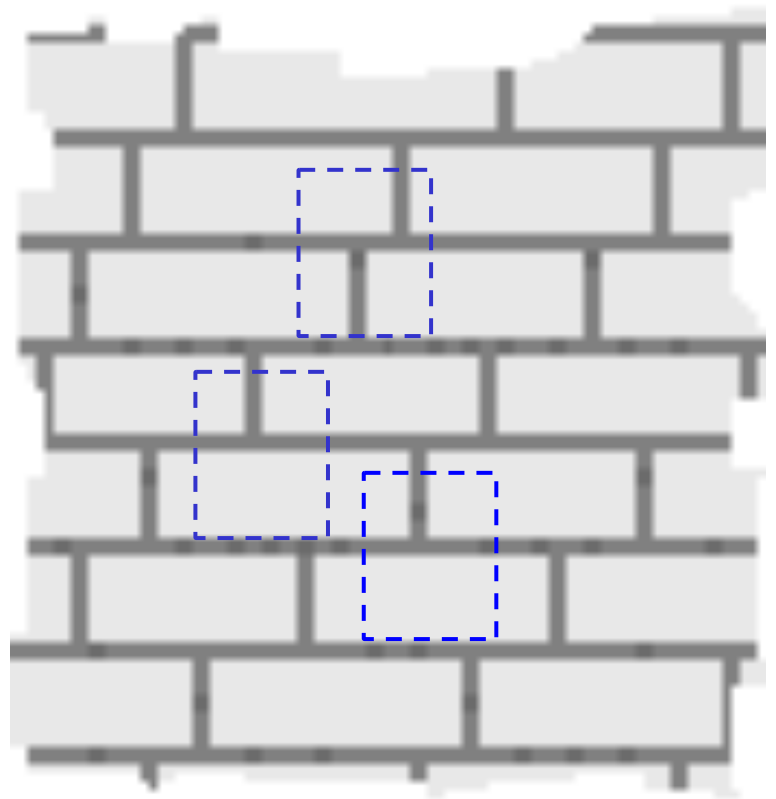
Ranked List	Similarity (cos)
x = 5, y = 17	**0.87**
x = 63, y = 4	**0.75**
x = 3, y = 44	**0.72**
x = 123, y = 54	0.64
x = 4, y = 57	0.60
⋮	⋮

best match

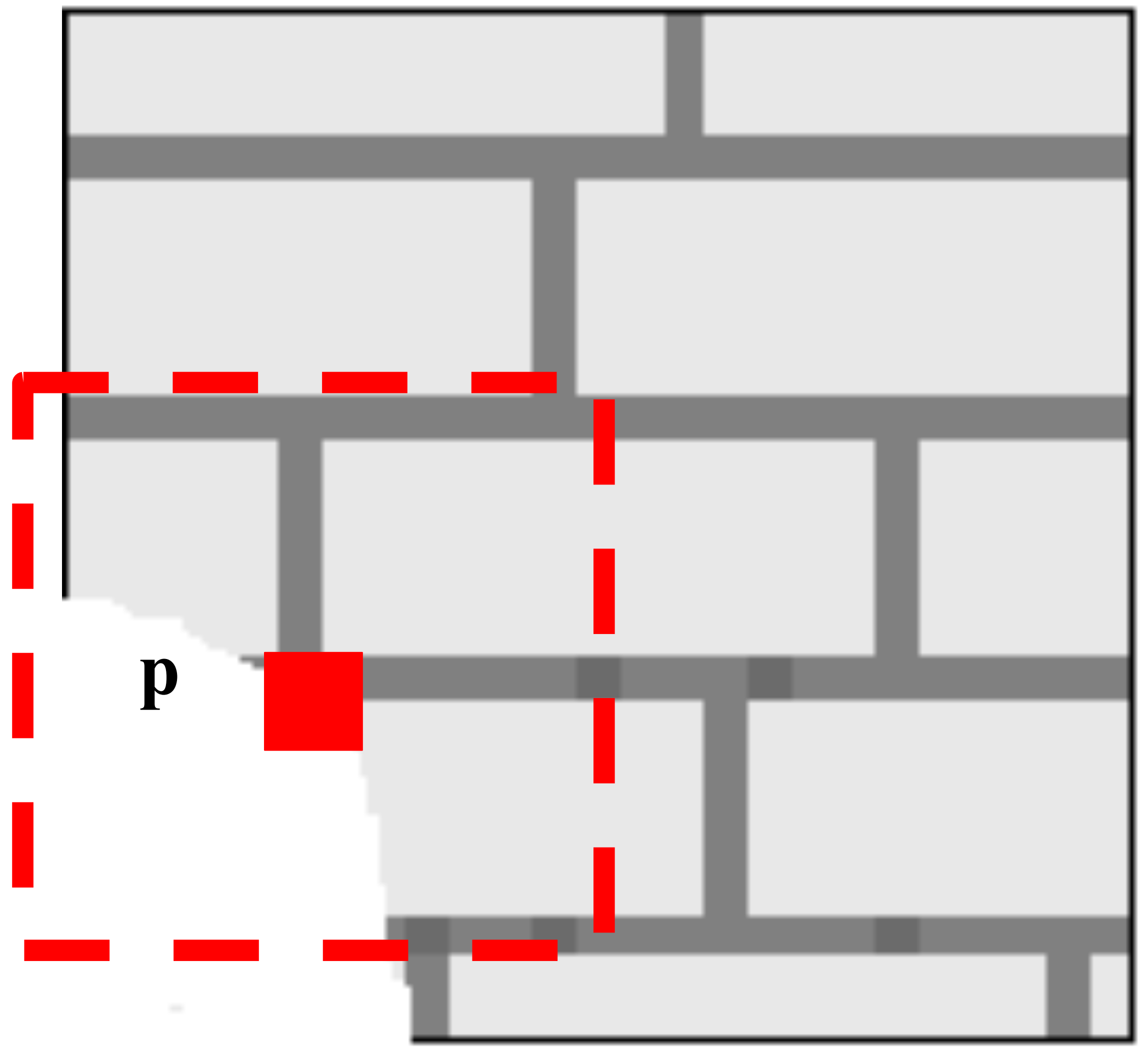
pick one at random and copy target pixel from it

threshold = best match * **0.8** = 0.696

Efros and Leung: Synthesizing One Pixel



SAMPLE



Infinite sample image

Ranked List	**Similarity (ssd)**
x = 5, y = 17	**0.13**
x = 63, y = 4	**0.25**
x = 3, y = 44	**0.28**
x = 123, y = 54	0.36
x = 4, y = 57	0.40
⋮	⋮

pick one at random and copy target pixel from it

threshold = best match * **2.5** = 0.325

Efros and Leung: Synthesizing Many Pixels

For multiple pixels, "grow" the texture in layers

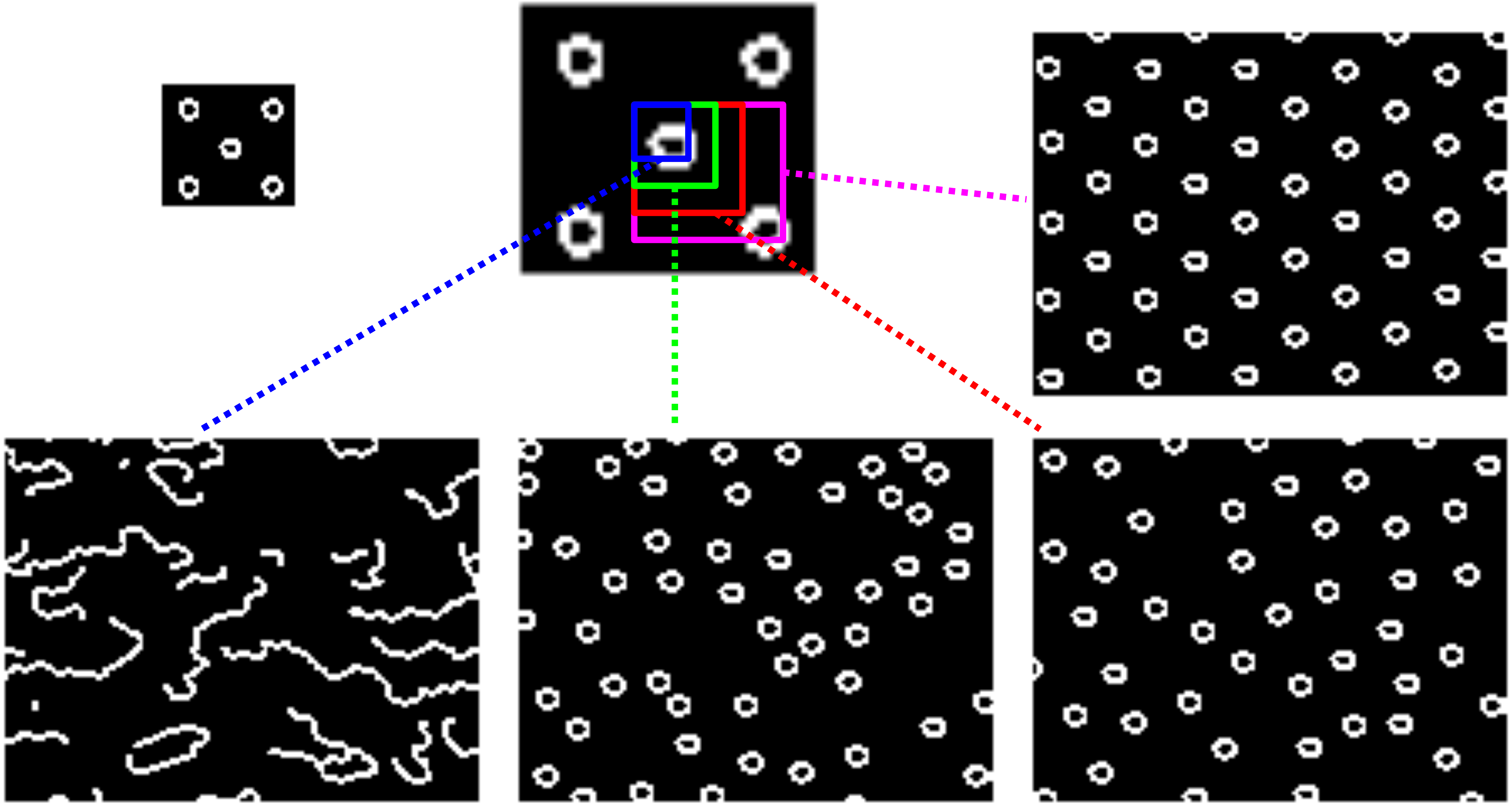
— In the case of hole-filling, start from the edges of the hole

For an interactive demo, see

https://una-dinosauria.github.io/efros-and-leung-js/

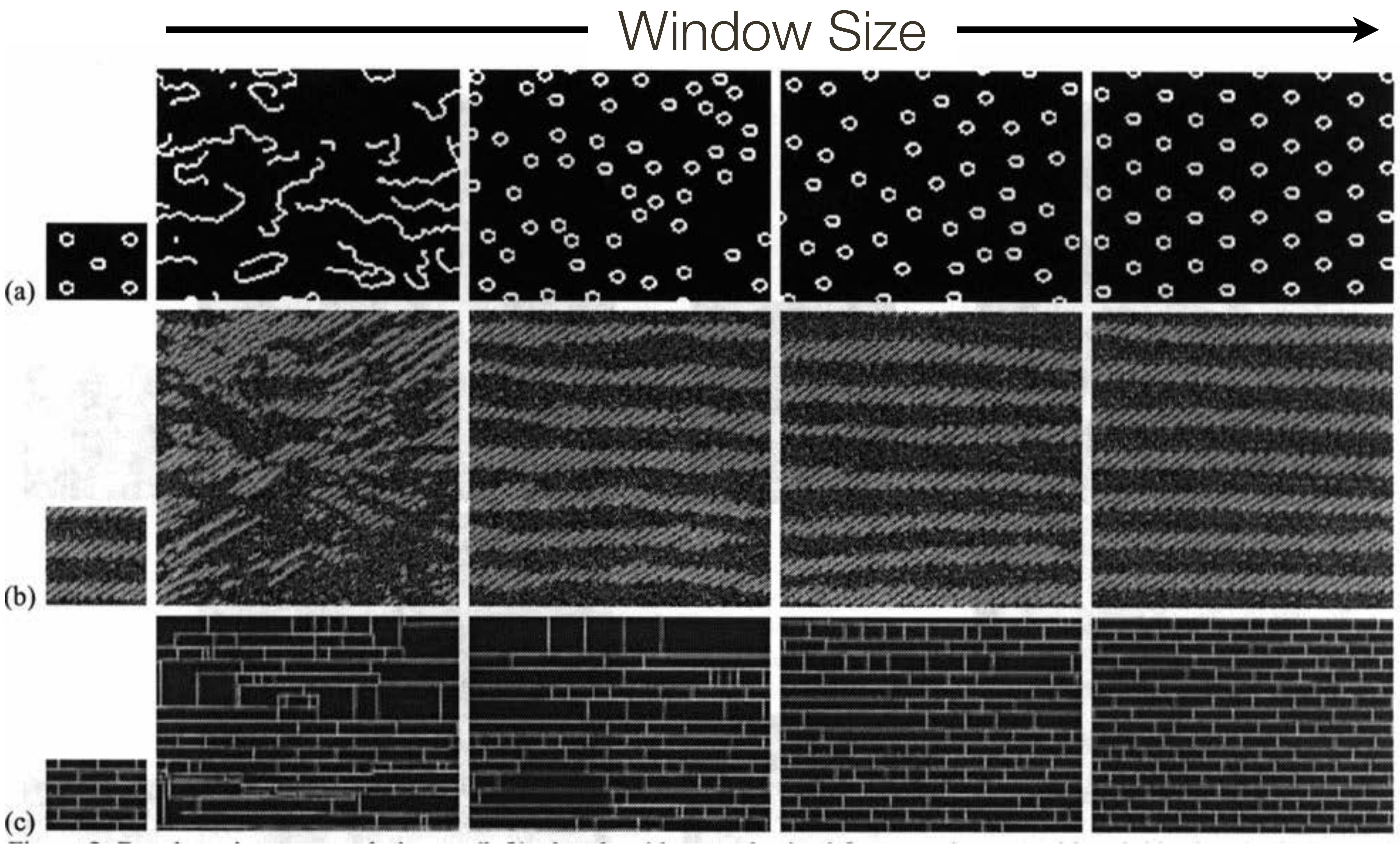
(written by Julieta Martinez, a previous CPSC 425 TA)

Randomness Parameter



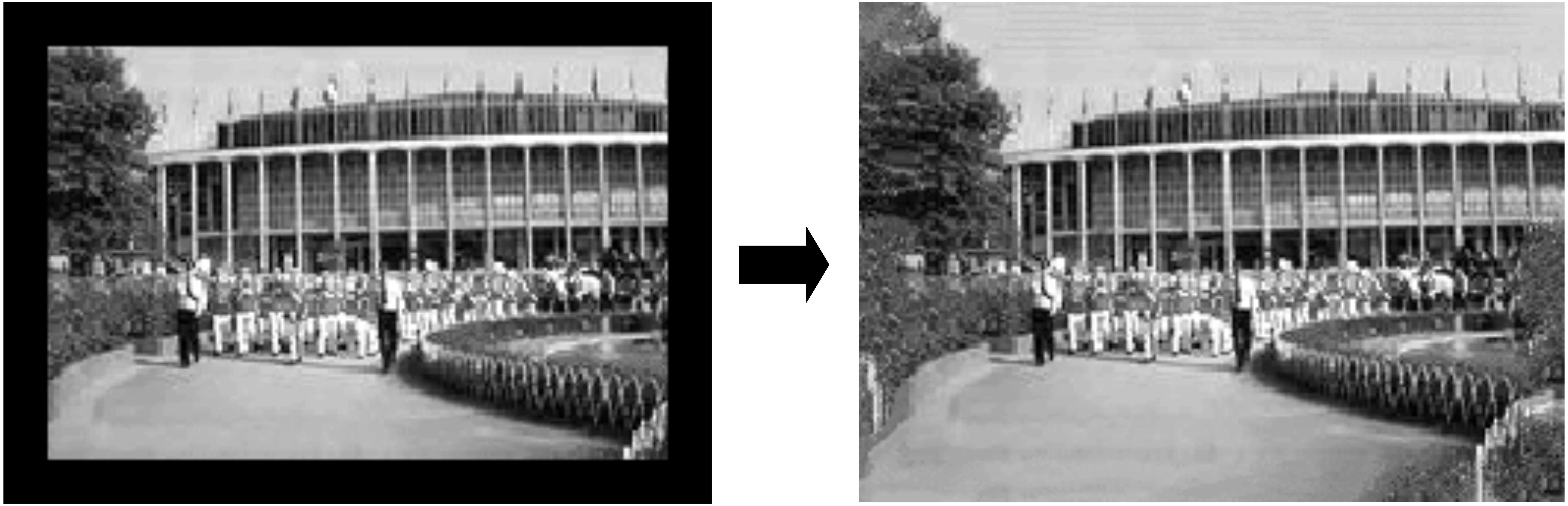
Slide Credit: http://graphics.cs.cmu.edu/people/efros/research/NPS/efros-iccv99.ppt

Efros and Leung: More Synthesis Results



Forsyth & Ponce (2nd ed.) Figure 6.12

Efros and Leung: Image Extrapolation



Slide Credit: http://graphics.cs.cmu.edu/people/efros/research/NPS/efros-iccv99.ppt

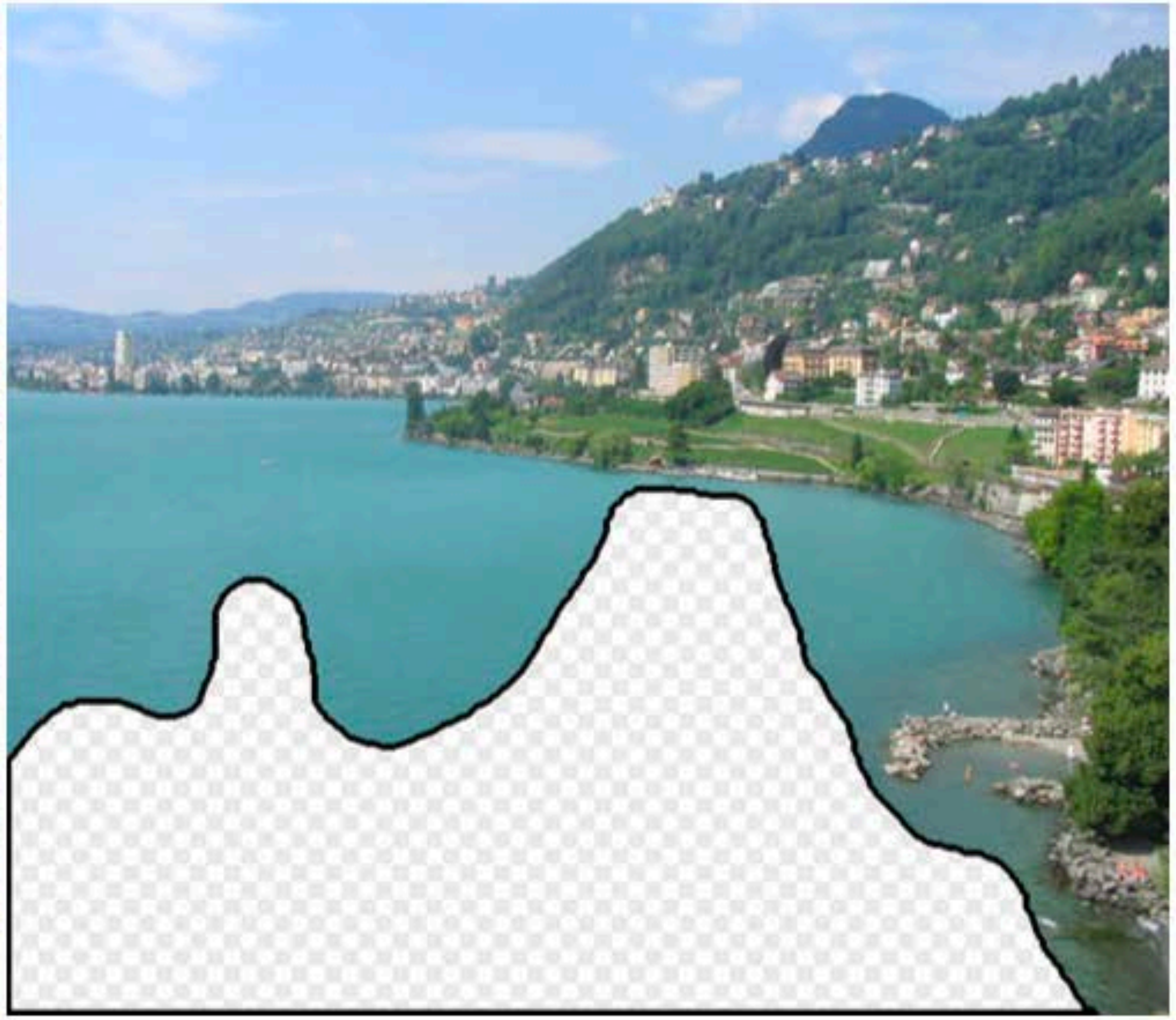
“**Big** Data” Meets Inpainting

“**Big** Data" enables surprisingly simple non-parametric, matching-based techniques to solve complex problems in computer graphics and vision.

Suppose instead of a single image, you had a massive database of a million images. What could you do?

“**Big** Data” Meets Inpainting

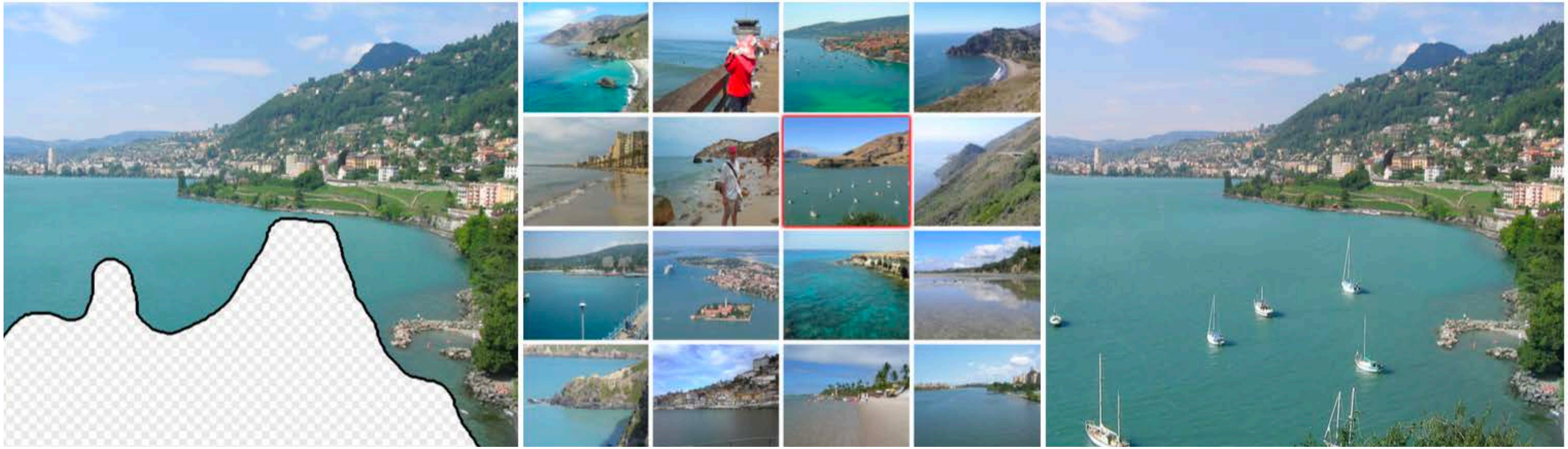
Original Image



Input

Figure Credit: Hays and Efros 2007

“**Big** Data” Meets Inpainting



Input Scene Matches Output

Figure Credit: Hays and Efros 2007

Effectiveness of "Big Data"

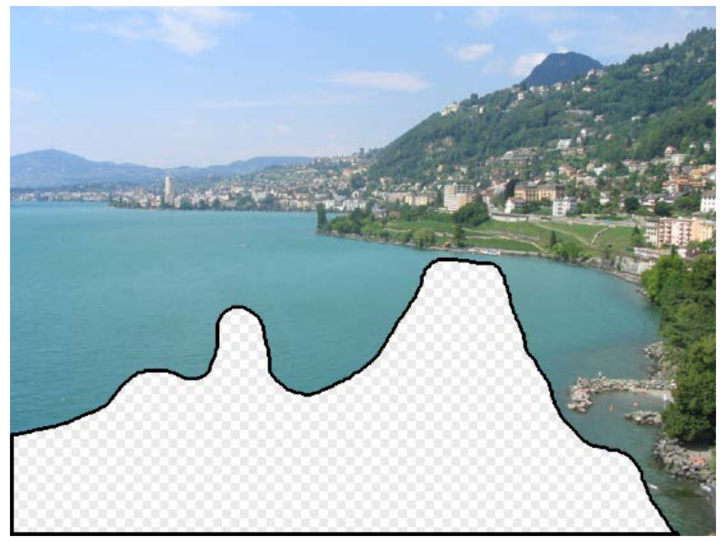
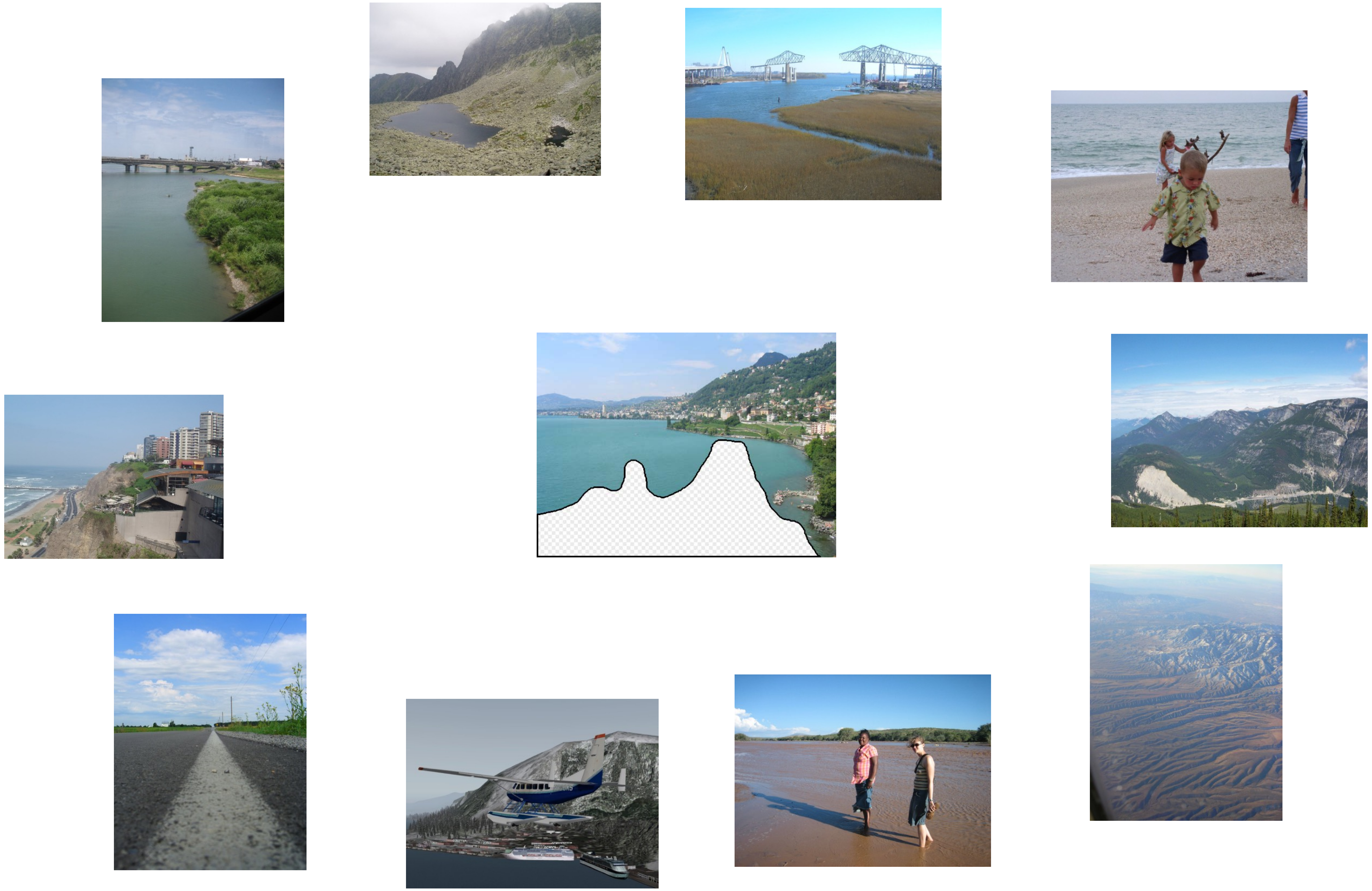


Figure Credit: Hays and Efros 2007

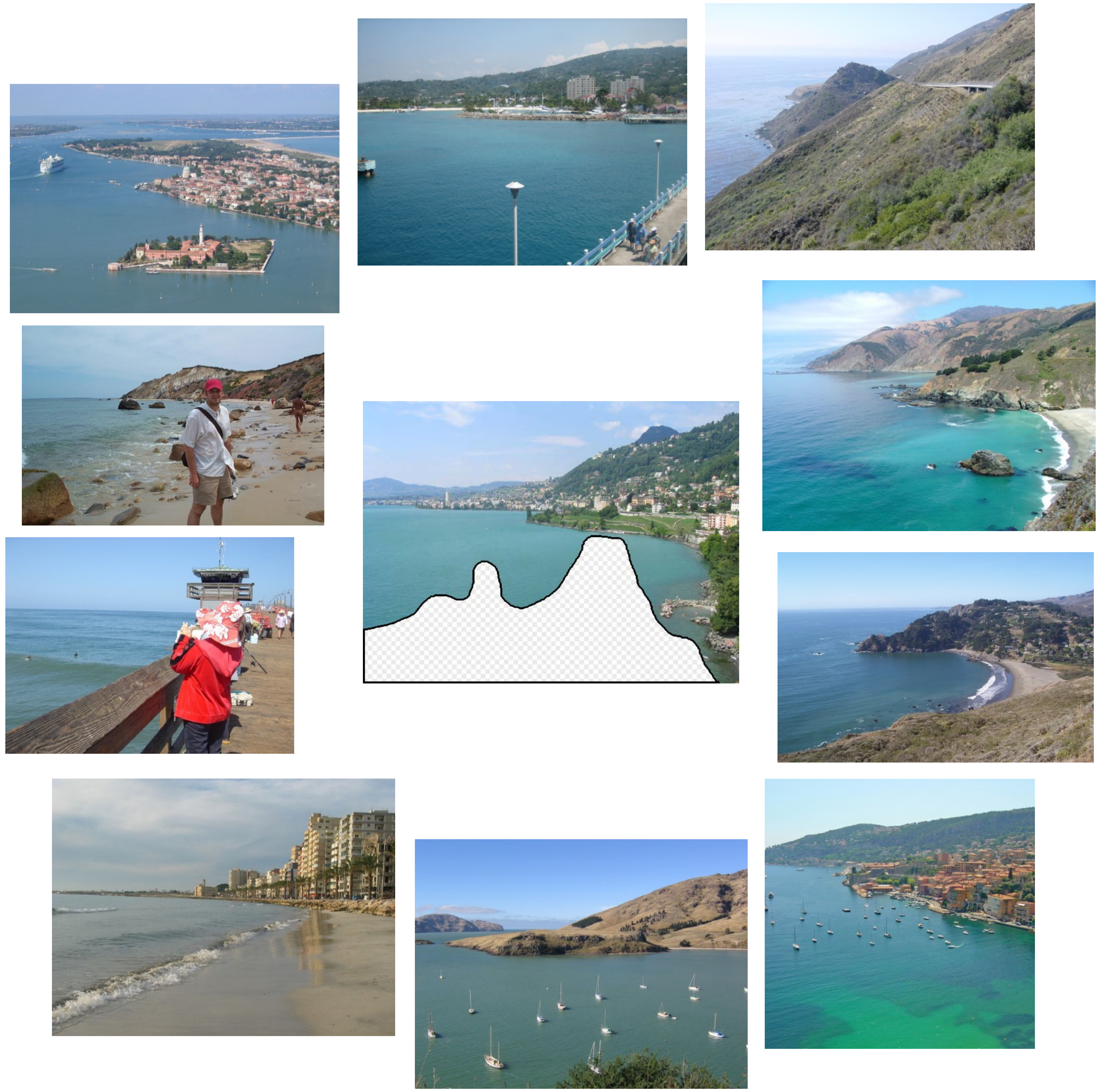
Effectiveness of “Big Data”



10 nearest neighbors from a collection of 20,000 images

Figure Credit: Hays and Efros 2007

Effectiveness of "Big Data"



10 nearest neighbors from a collection of 2 million images

Figure Credit: Hays and Efros 2007

“**Big** Data” Meets Inpainting

Figure Credit: Hays and Efros 2007

“**Big** Data” Meets Inpainting

Algorithm sketch (Hays and Efros 2007):

1. Create a short list of a few hundred “best matching" images based on global image statistics

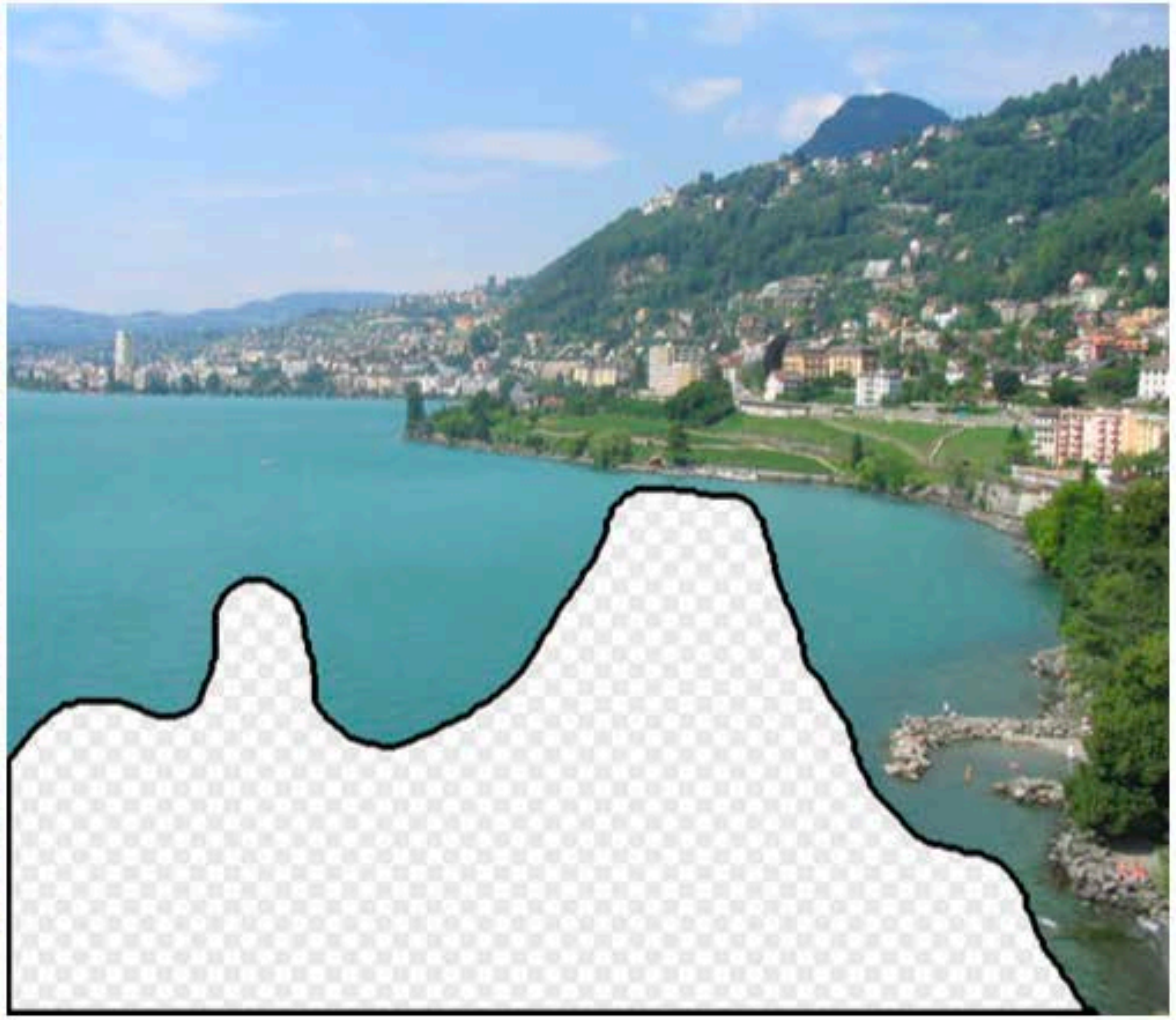
2. Find patches in the short list that match the context surrounding the image region we want to fill

3. Blend the match into the original image

Purely **data-driven**, requires no manual labeling of images

“**Big** Data” Meets Inpainting

Original Image



Input

Figure Credit: Hays and Efros 2007

“**Big** Data” Meets Inpainting

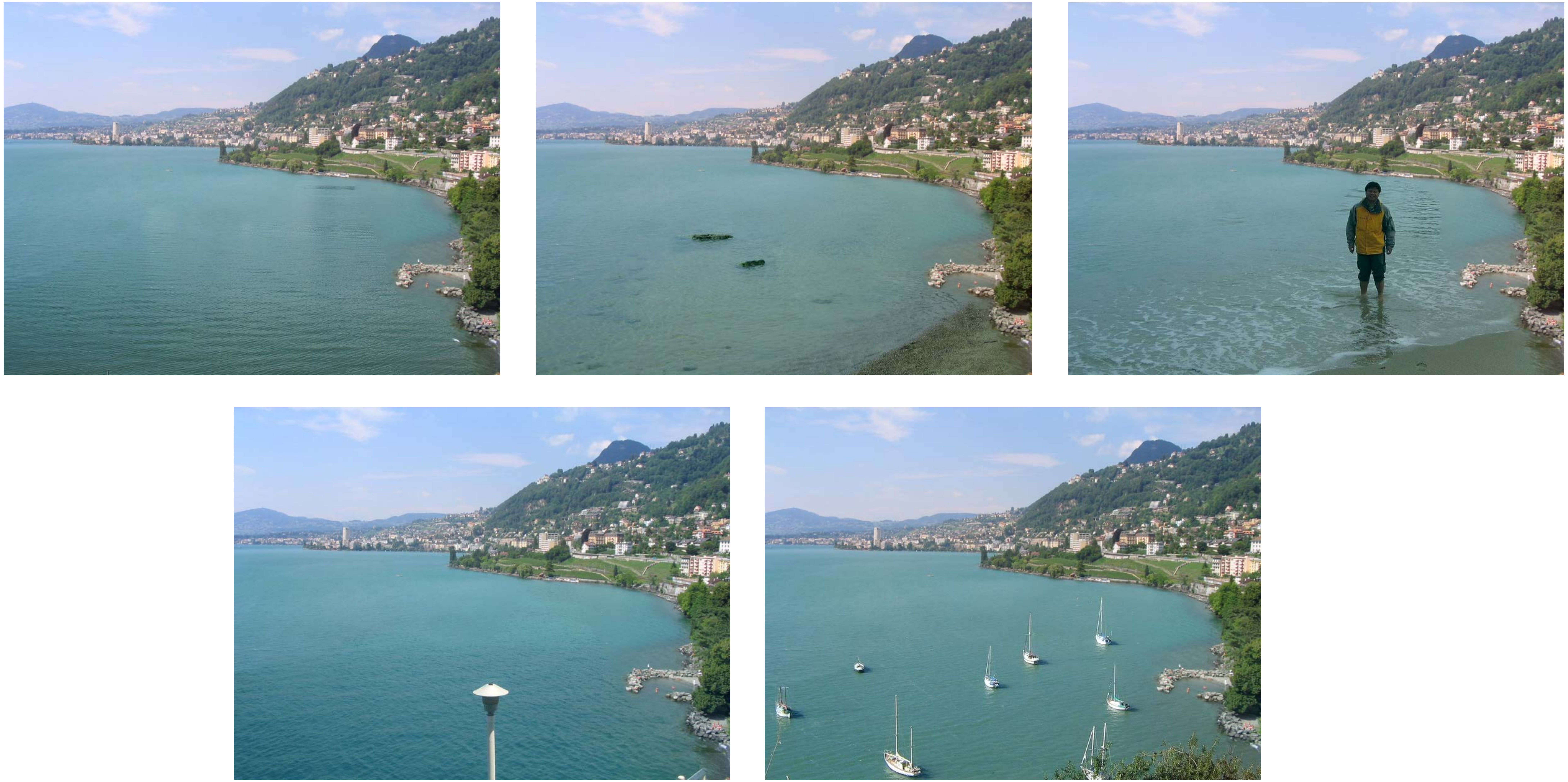


Figure Credit: Hays and Efros 2007

“**Big** Data” Meets Inpainting

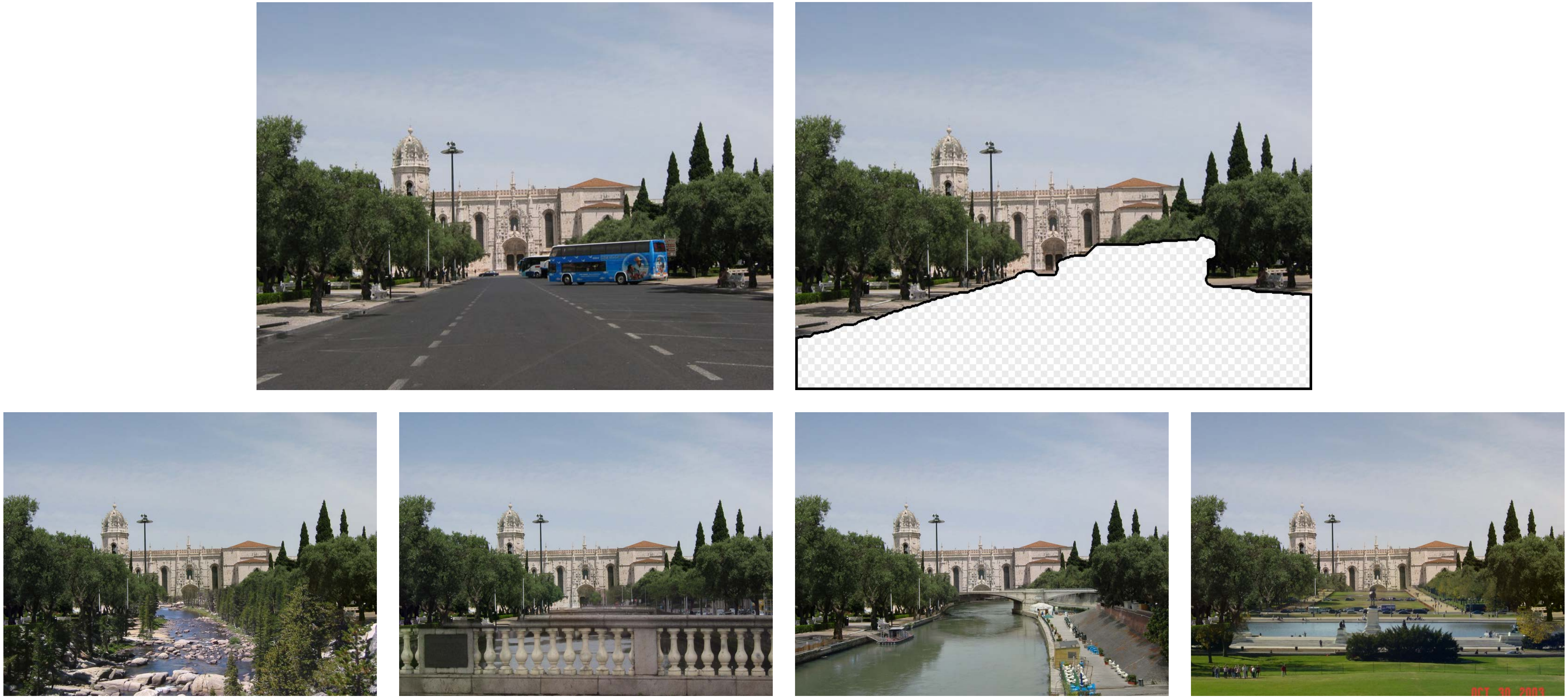


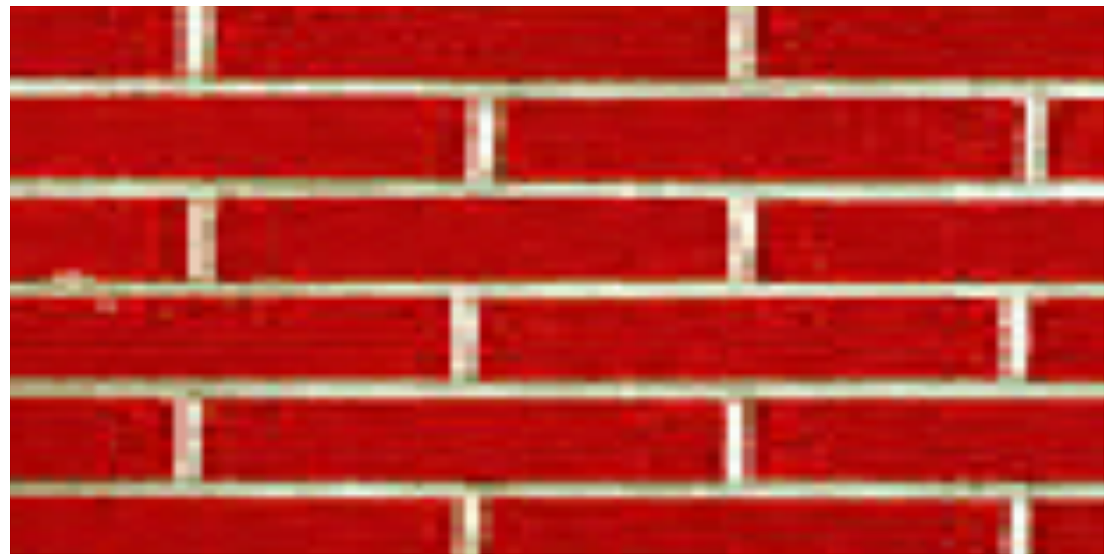
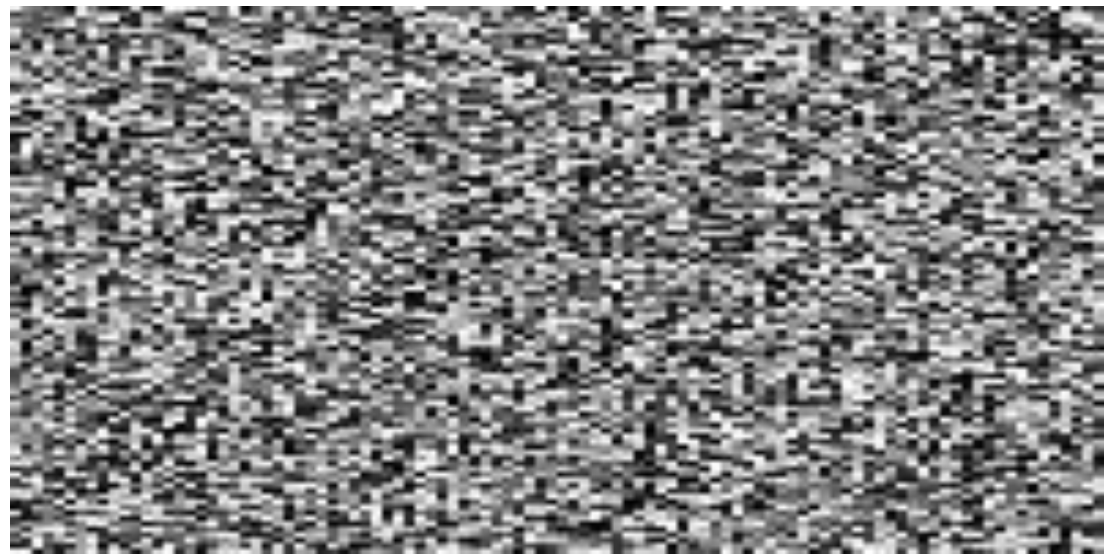
Figure Credit: Hays and Efros 2007

How do we **analyze** texture?

Texture **Representation**

Observation: Textures are made up of generic sub-elements, repeated over a region with similar statistical properties

Idea: Find the sub-elements with filters, then represent each point in the image with a summary of the pattern of sub-elements in the local region

Texture **Representation**

Observation: Textures are made up of generic sub-elements, repeated over a region with similar statistical properties

Idea: Find the sub-elements with filters, then represent each point in the image with a summary of the pattern of sub-elements in the local region

Question: What filters should we use?

Answer: Human vision suggests spots and oriented edge filters at a variety of different orientations and scales

Texture **Representation**

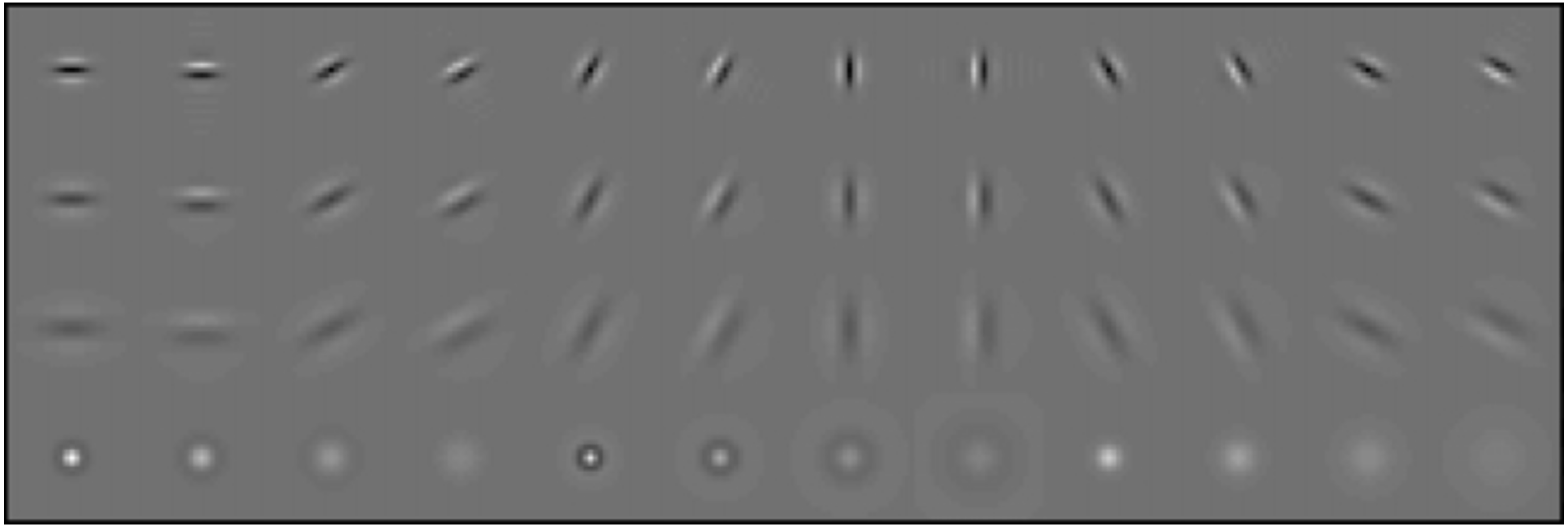


Figure Credit: Leung and Malik, 2001

Texture **Representation**

First derivative of Gaussian at 6 orientations and 3 scales

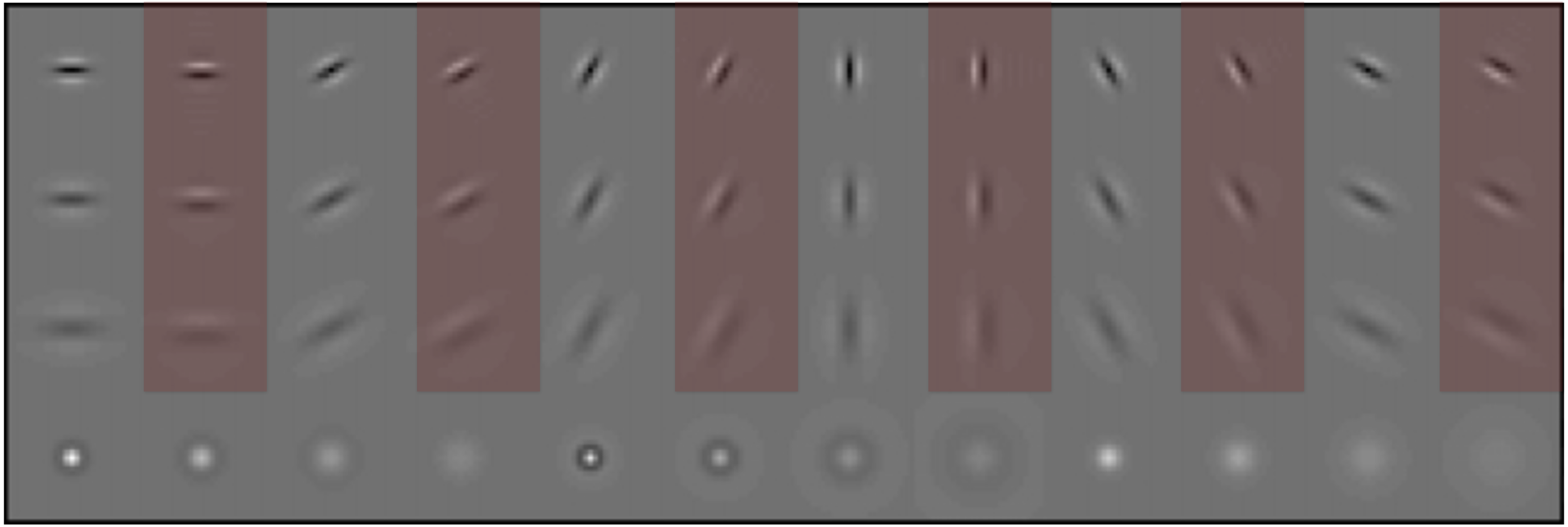


Figure Credit: Leung and Malik, 2001

Texture **Representation**

Second derivative of Gaussian at 6 orientations 3 scales

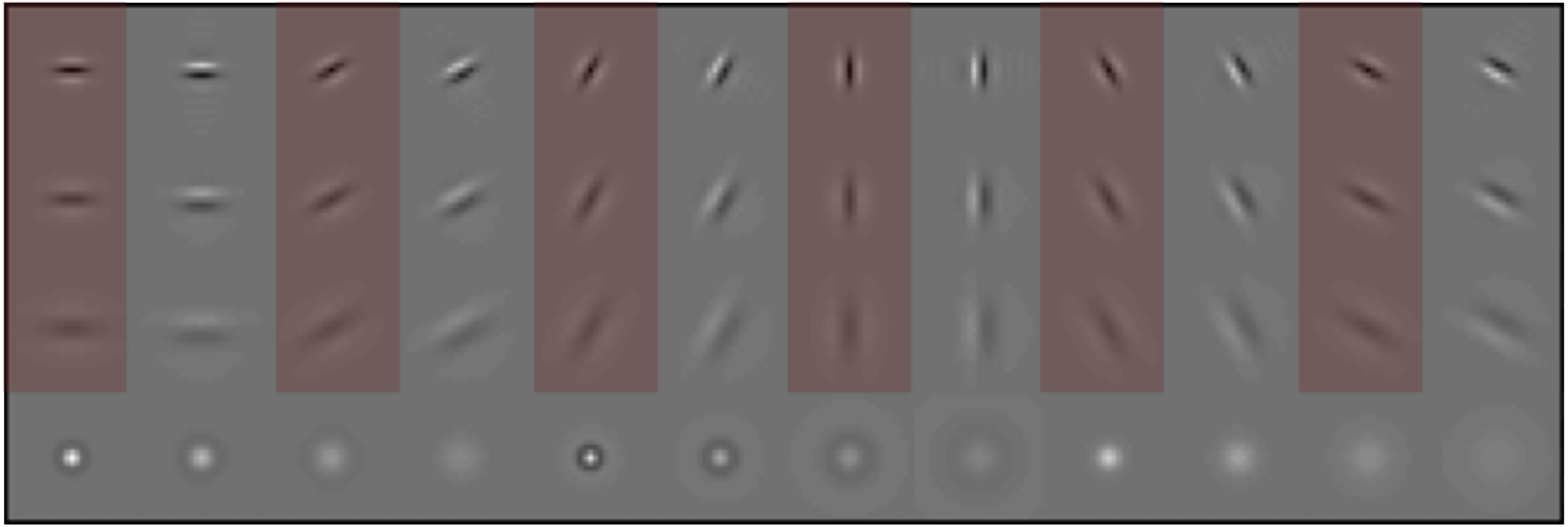


Figure Credit: Leung and Malik, 2001

Texture **Representation**

Laplacian of the Gaussian filters at different scales

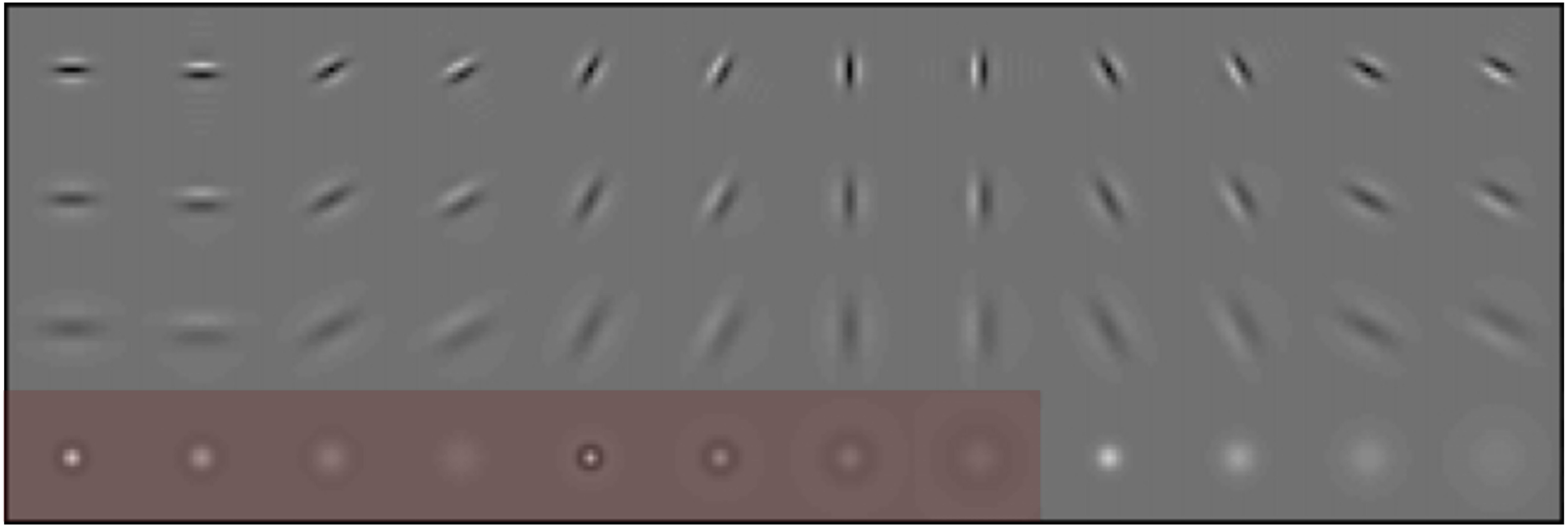


Figure Credit: Leung and Malik, 2001

Texture **Representation**

Gaussian filters at different scales

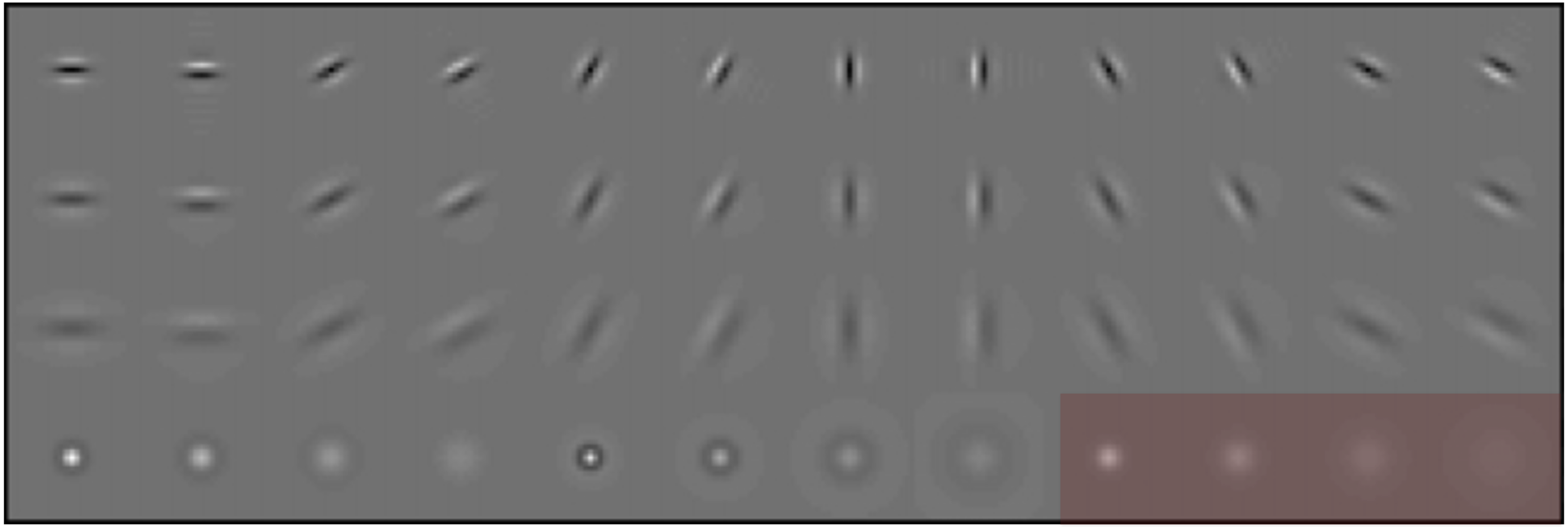
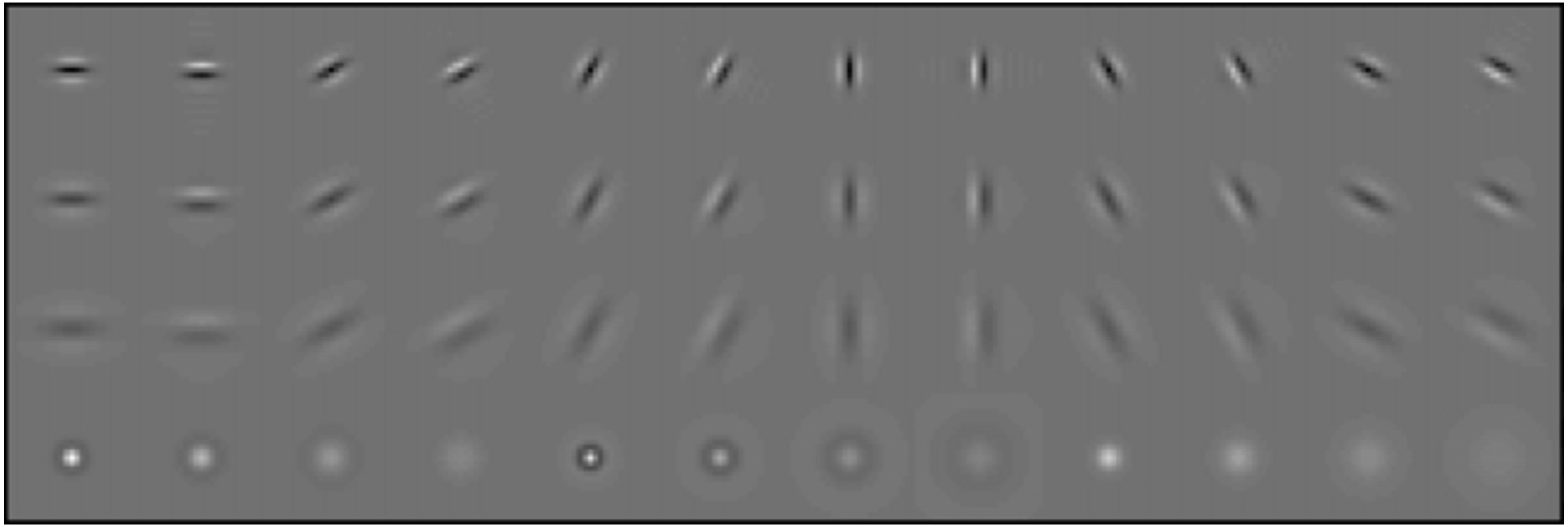


Figure Credit: Leung and Malik, 2001

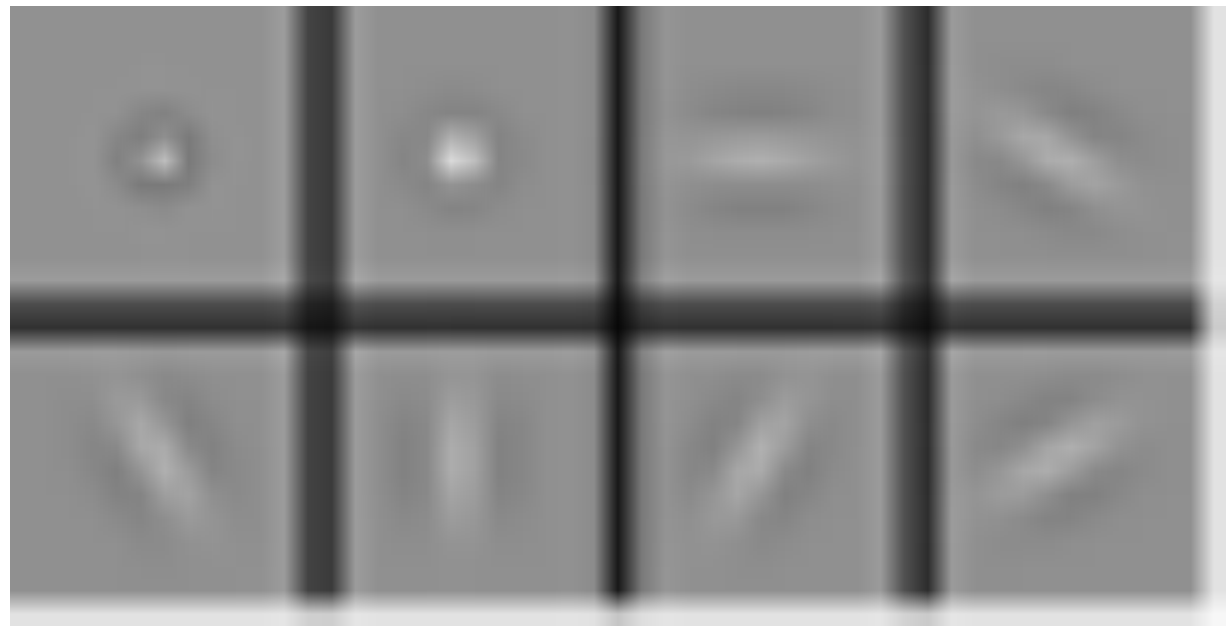
Texture **Representation**

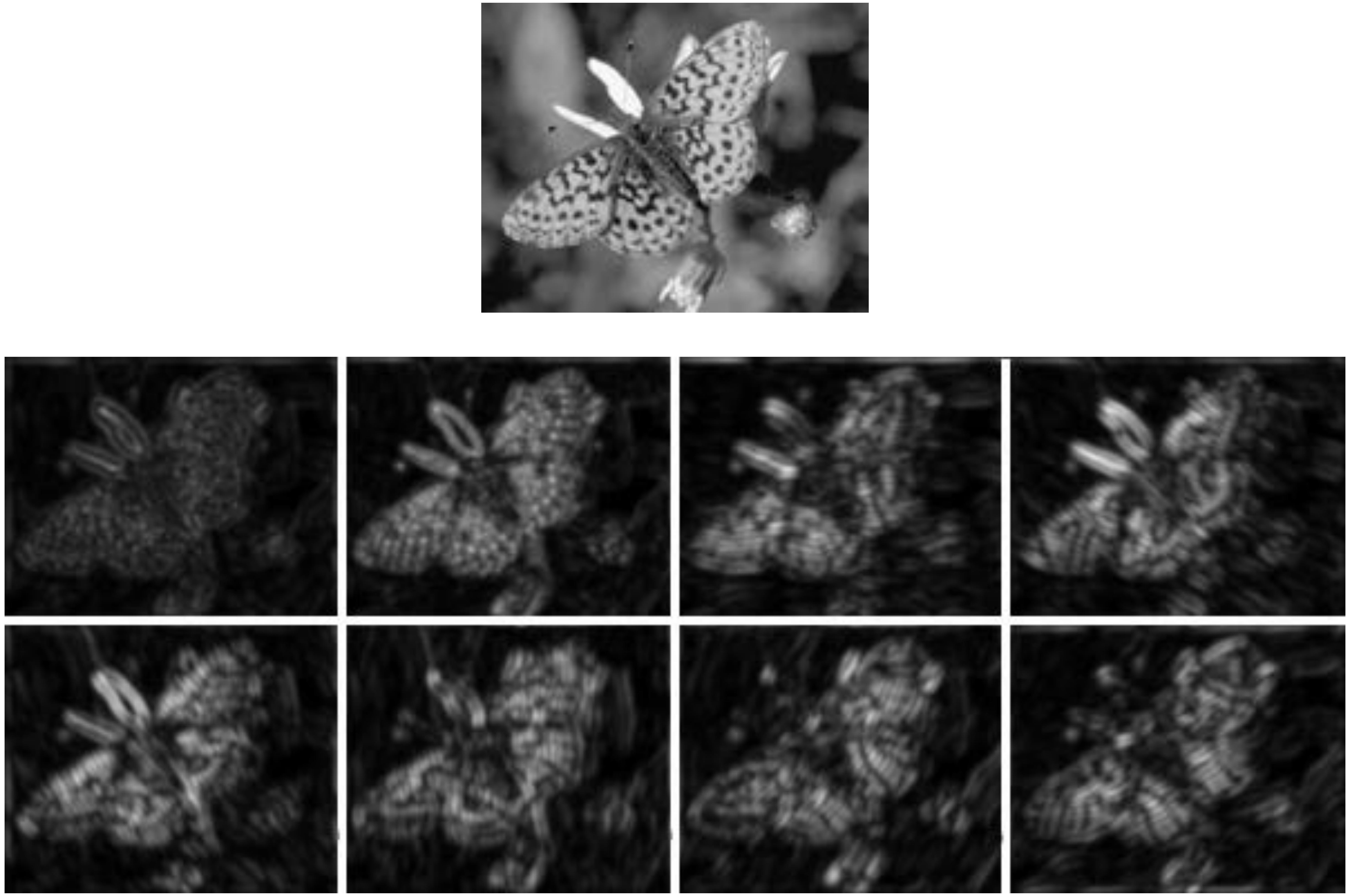


Result: 48-channel "image"

Figure Credit: Leung and Malik, 2001

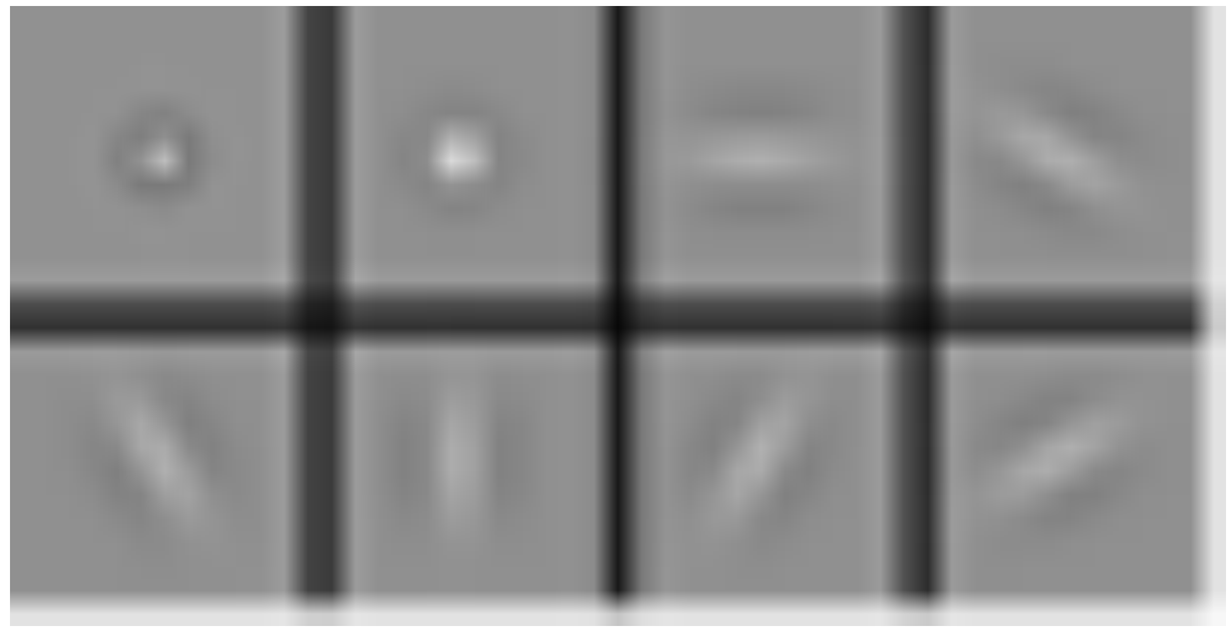
Spots and Bars (Fine Scale)

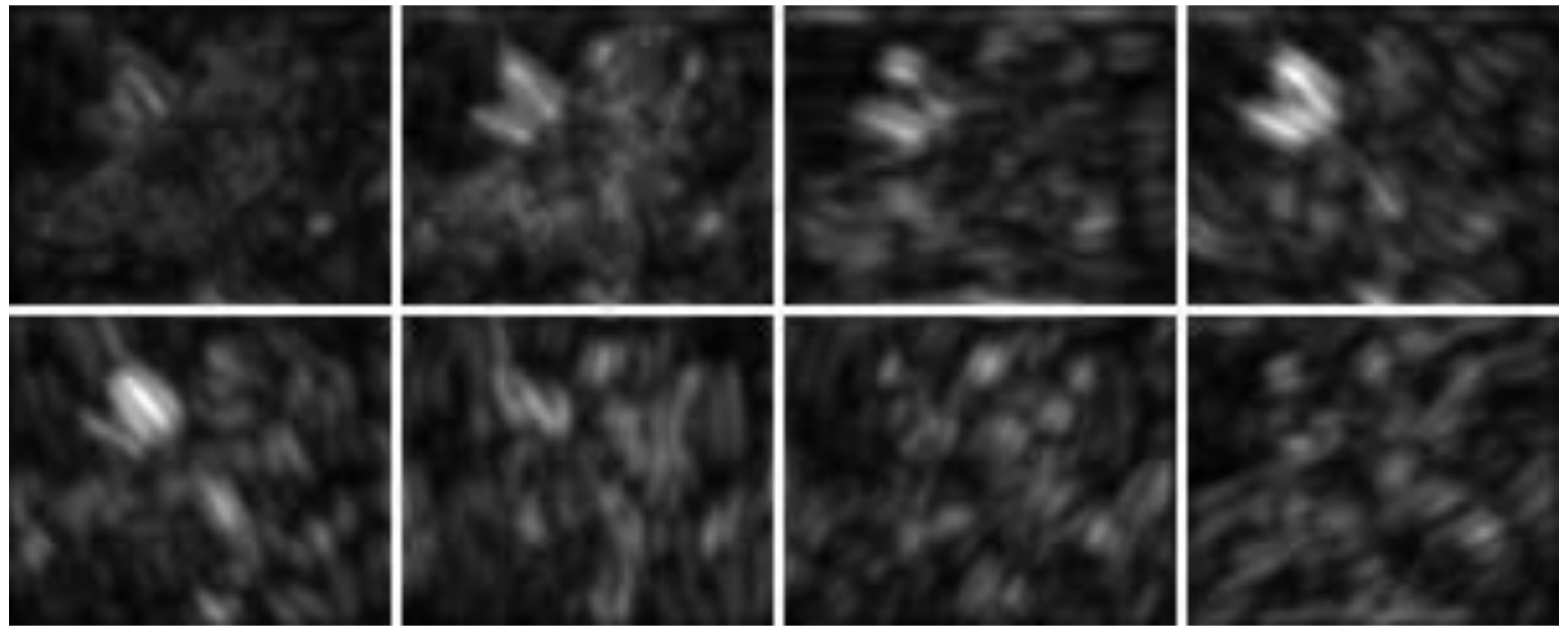




Forsyth & Ponce (1st ed.) Figures 9.3–9.4

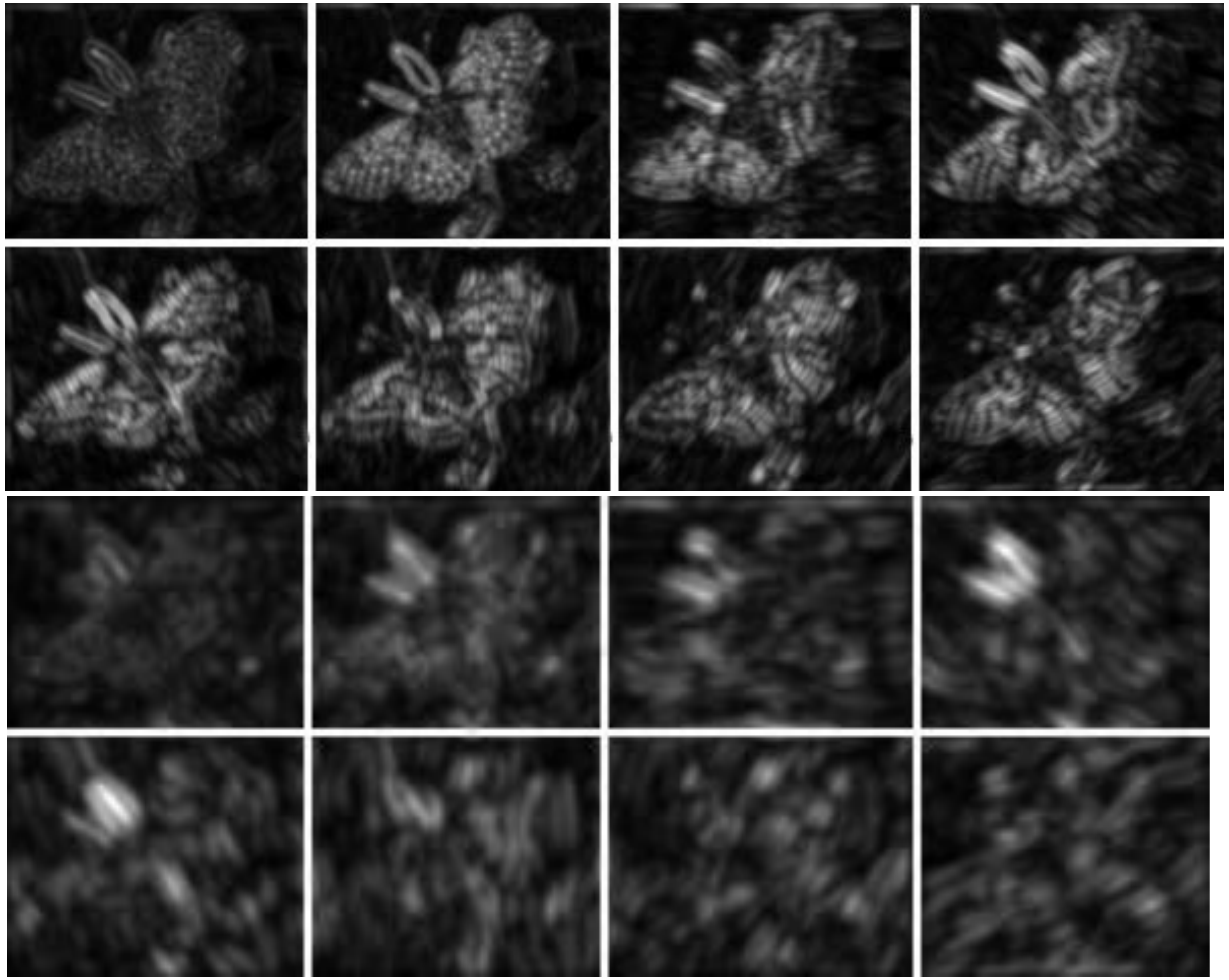
Spots and Bars (Coarse Scale)





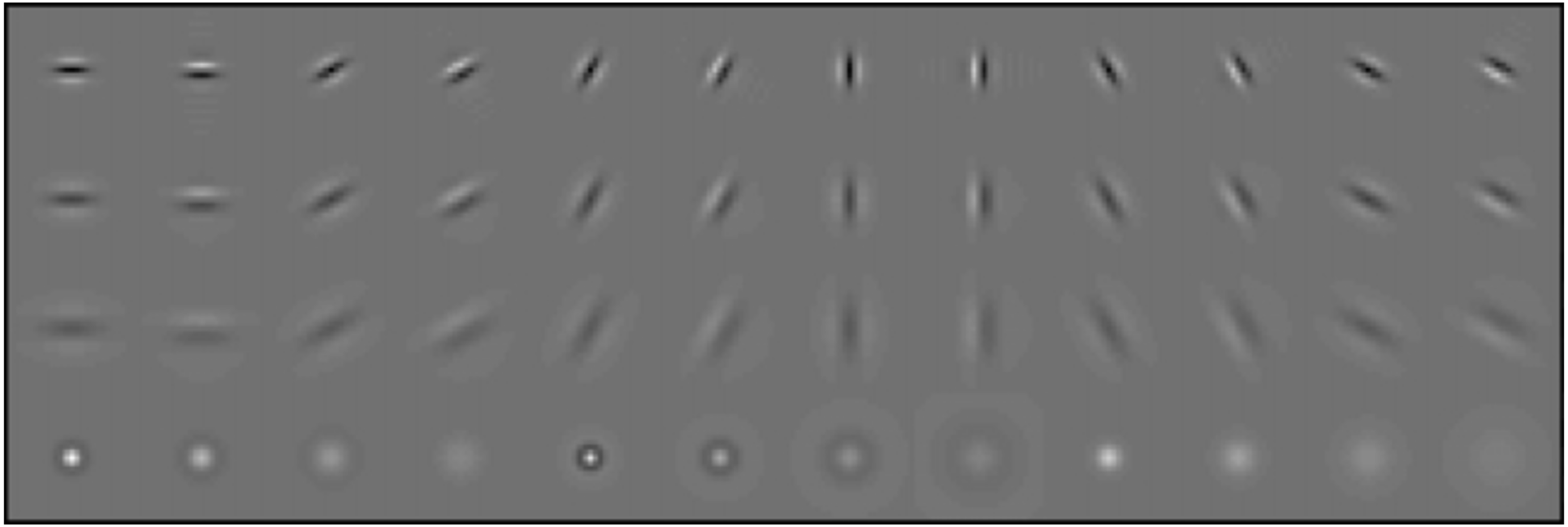
Forsyth & Ponce (1st ed.) Figures 9.3 and 9.5

Comparison of Results



Forsyth & Ponce (1st ed.) Figures 9.4–9.5

Texture **Representation**



Result: 48-channel “image”

Figure Credit: Leung and Malik, 2001

Texture **Representation**

Observation: Textures are made up of generic sub-elements, repeated over a region with similar statistical properties

Idea: Find the sub-elements with filters, then represent each point in the image with a summary of the pattern of sub-elements in the local region

Question: What filters should we use?

Answer: Human vision suggests spots and oriented edge filters at a variety of different orientations and scales

Texture **Representation**

Observation: Textures are made up of generic sub-elements, repeated over a region with similar statistical properties

Idea: Find the sub-elements with filters, then represent each point in the image with a summary of the pattern of sub-elements in the local region

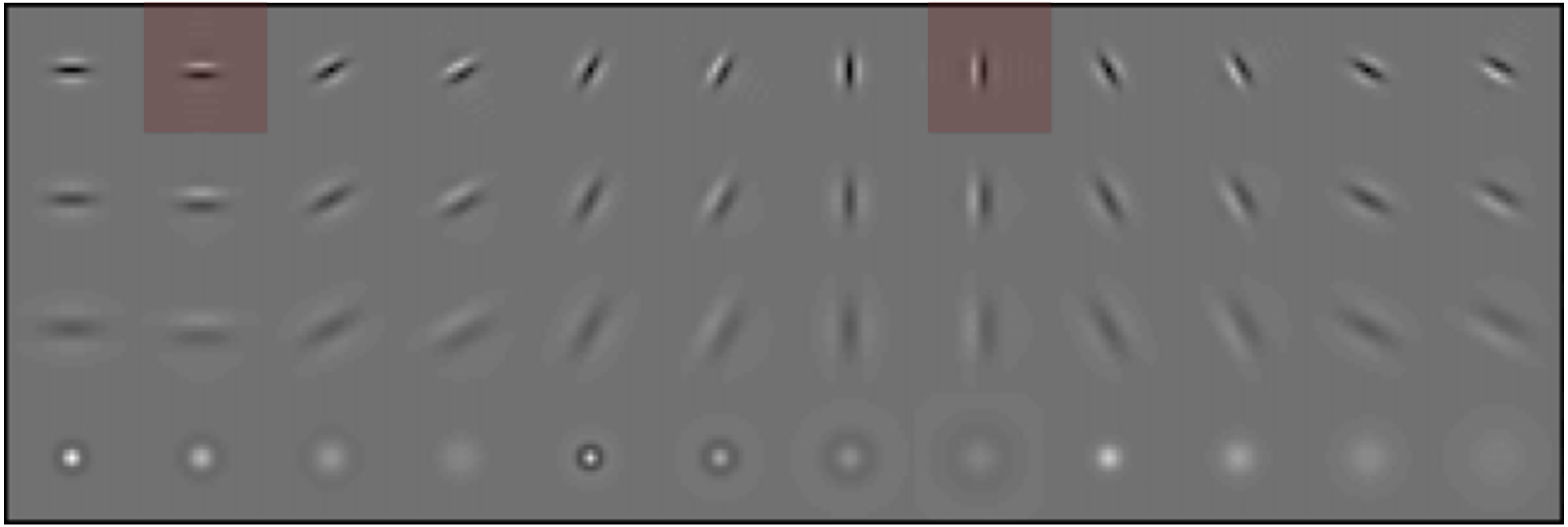
Question: What filters should we use?

Answer: Human vision suggests spots and oriented edge filters at a variety of different orientations and scales

Question: How do we “summarize”?

Answer: Compute the mean or maximum of each filter response over the region
— Other statistics can also be useful

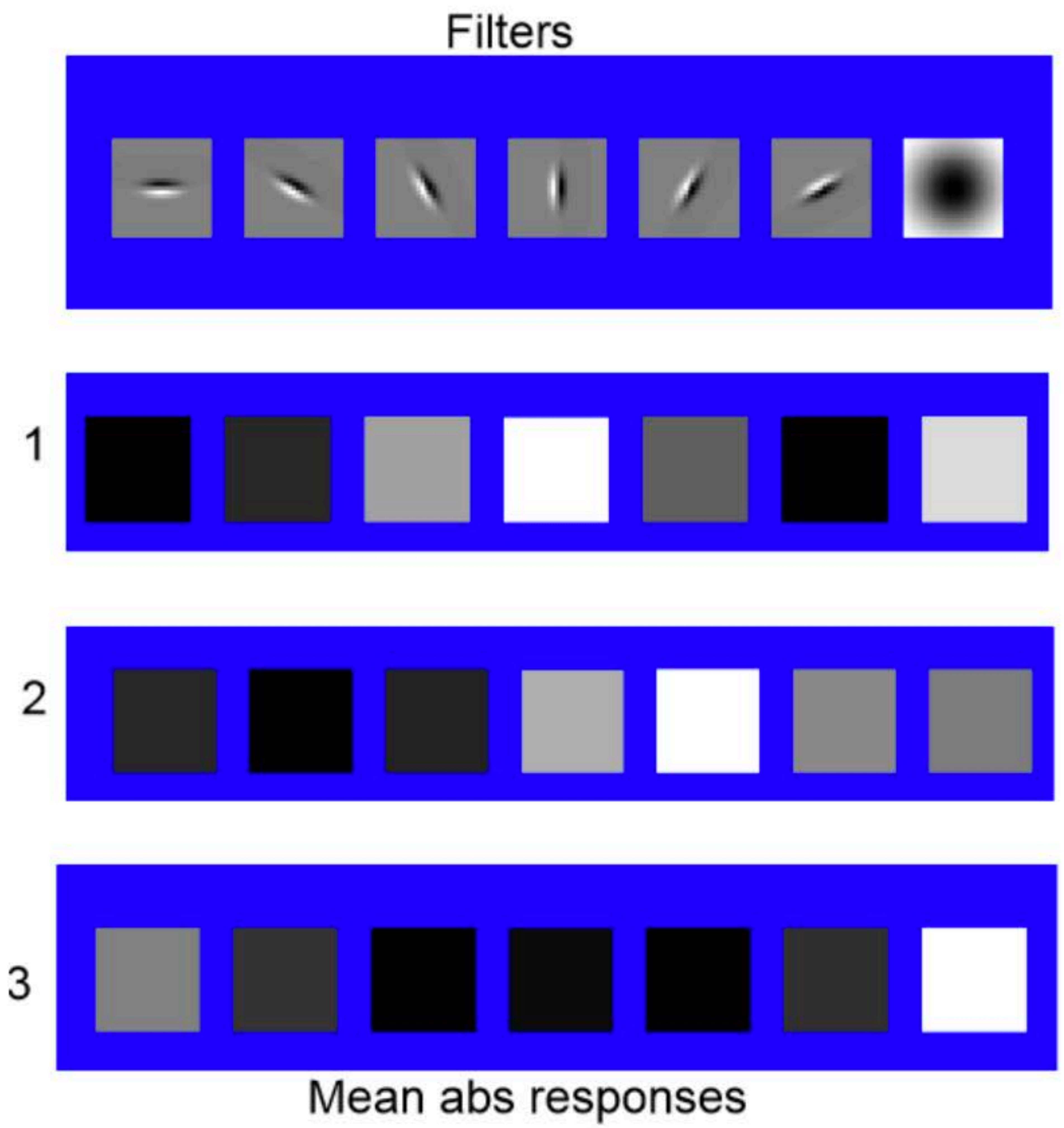
Texture **Representation**

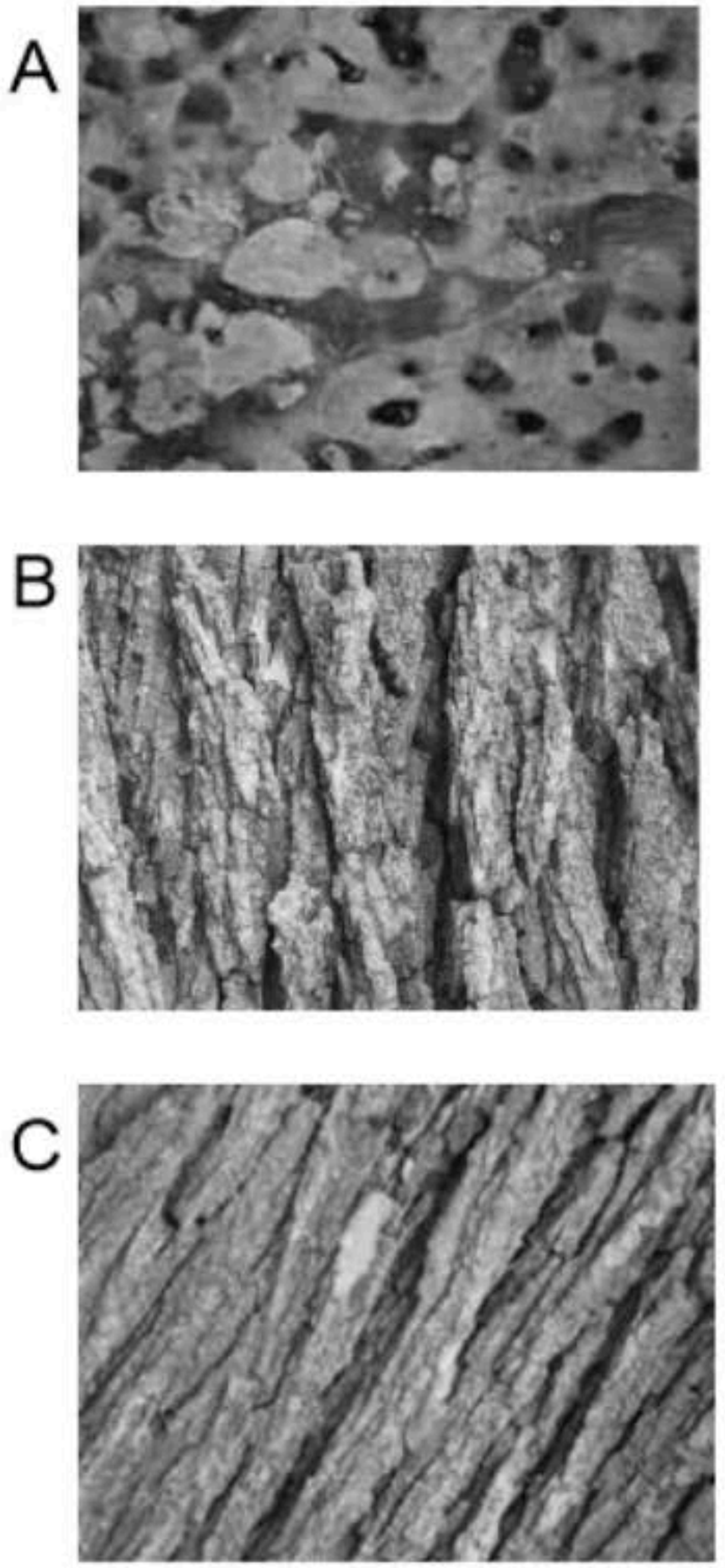


Result: 48-channel “image”

Figure Credit: Leung and Malik, 2001

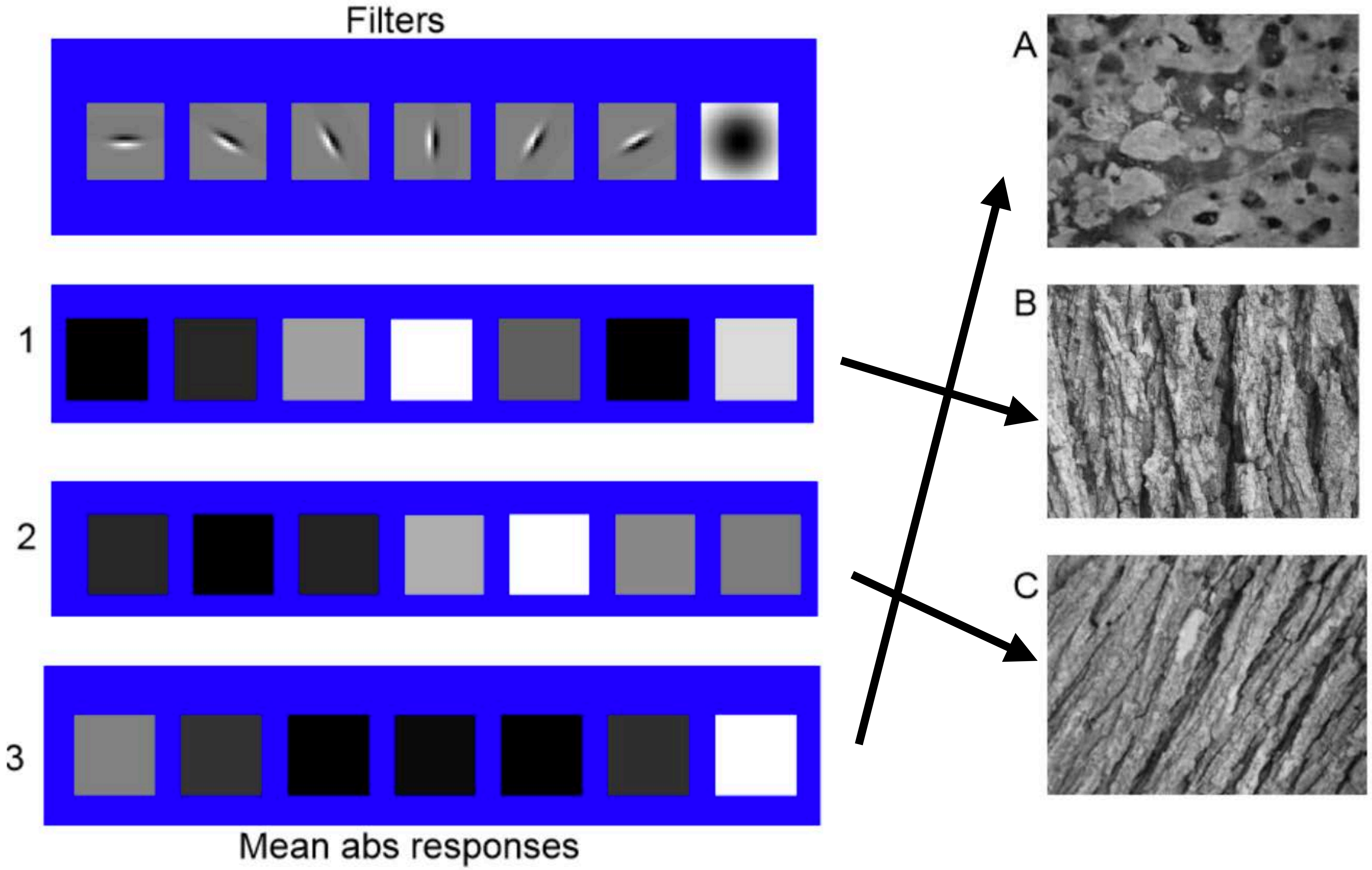
A Short **Exercise**: Match the texture to the response





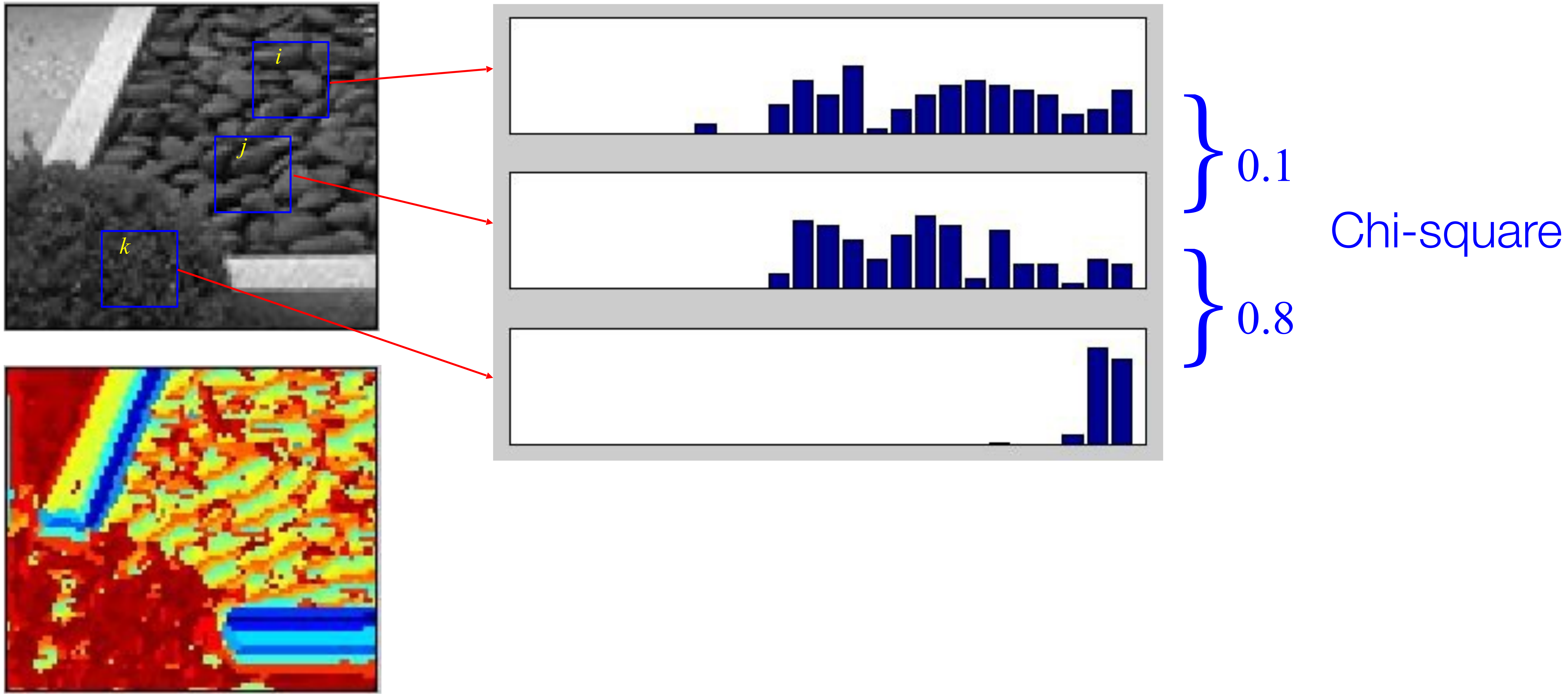
Slide Credit: James Hays

A Short **Exercise**: Match the texture to the response



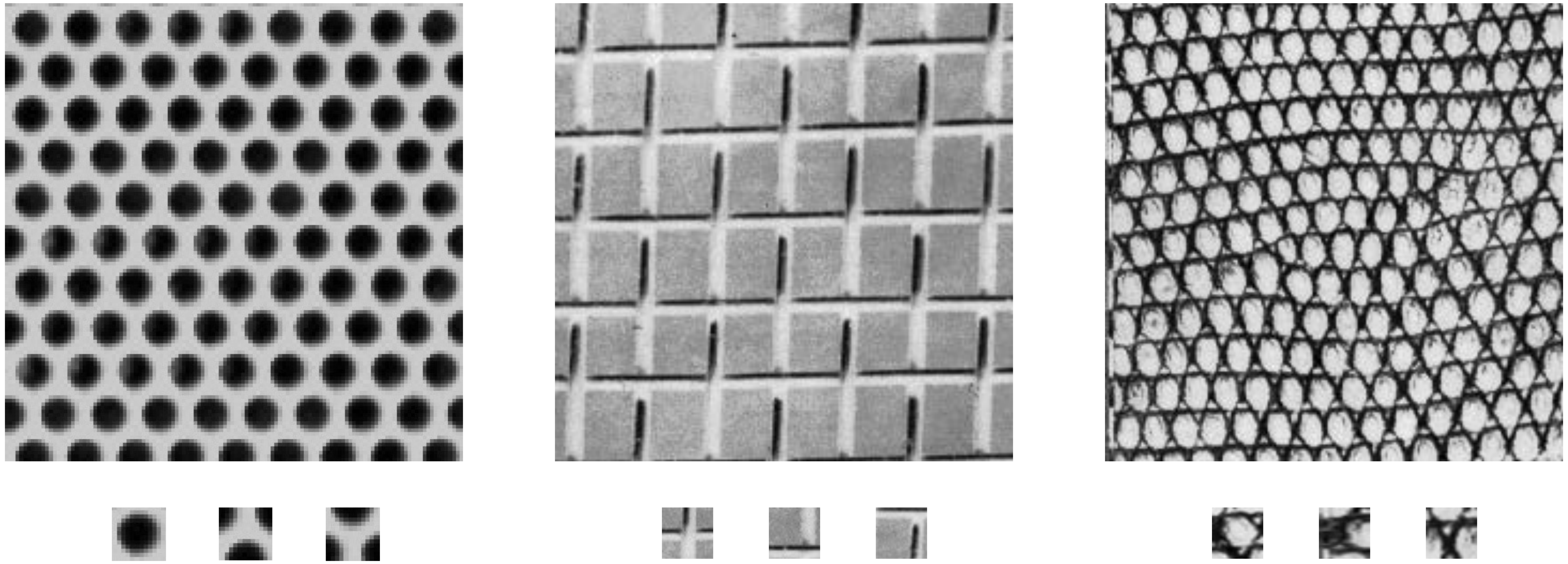
Slide Credit: James Hays

Texture **Representation**



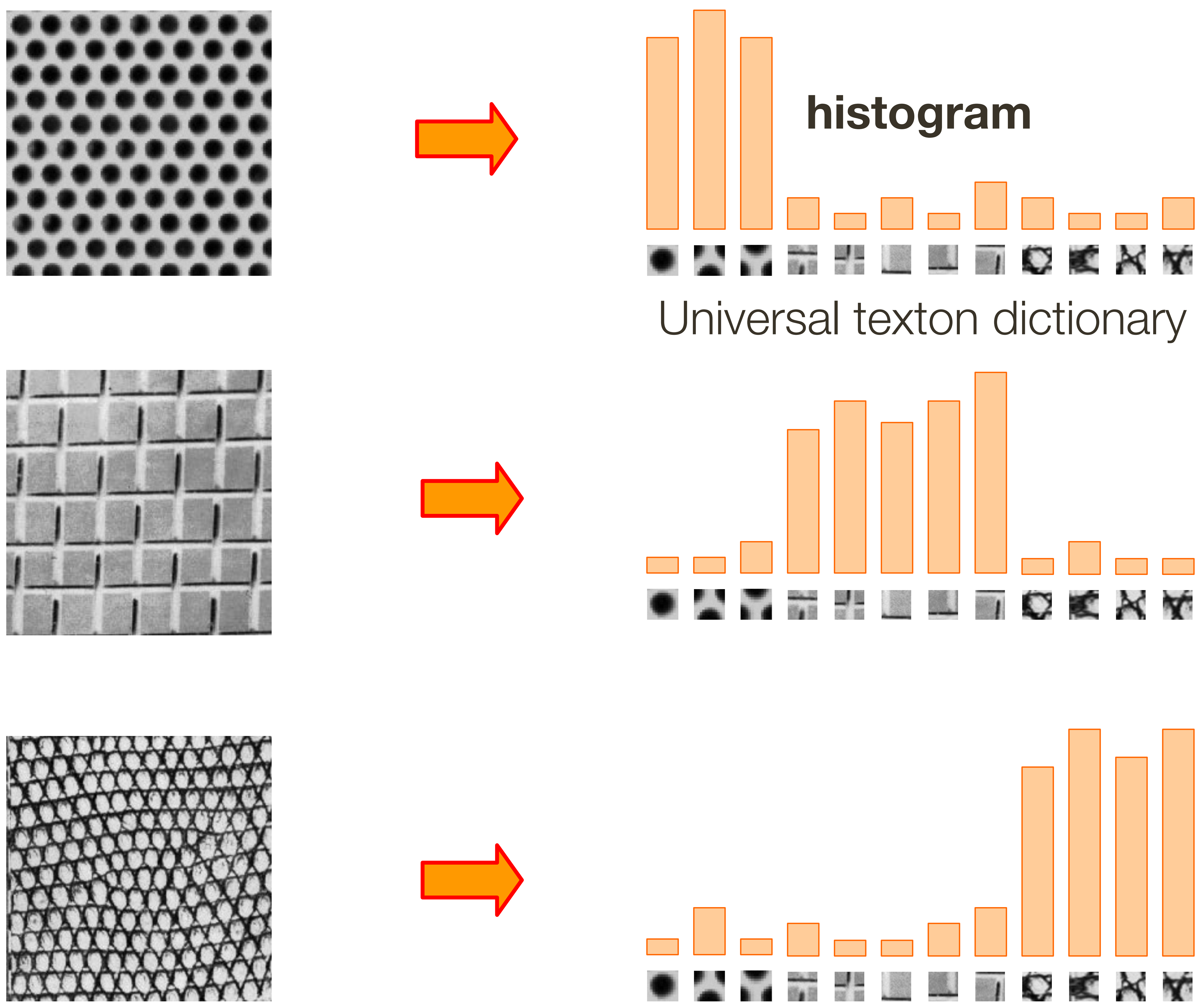
Texture representation and recognition

- Texture is characterized by the repetition of basic elements or **textons**
- For stochastic textures, it is the **identity of the textons**, not their spatial arrangement, that matters



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Texture representation and recognition



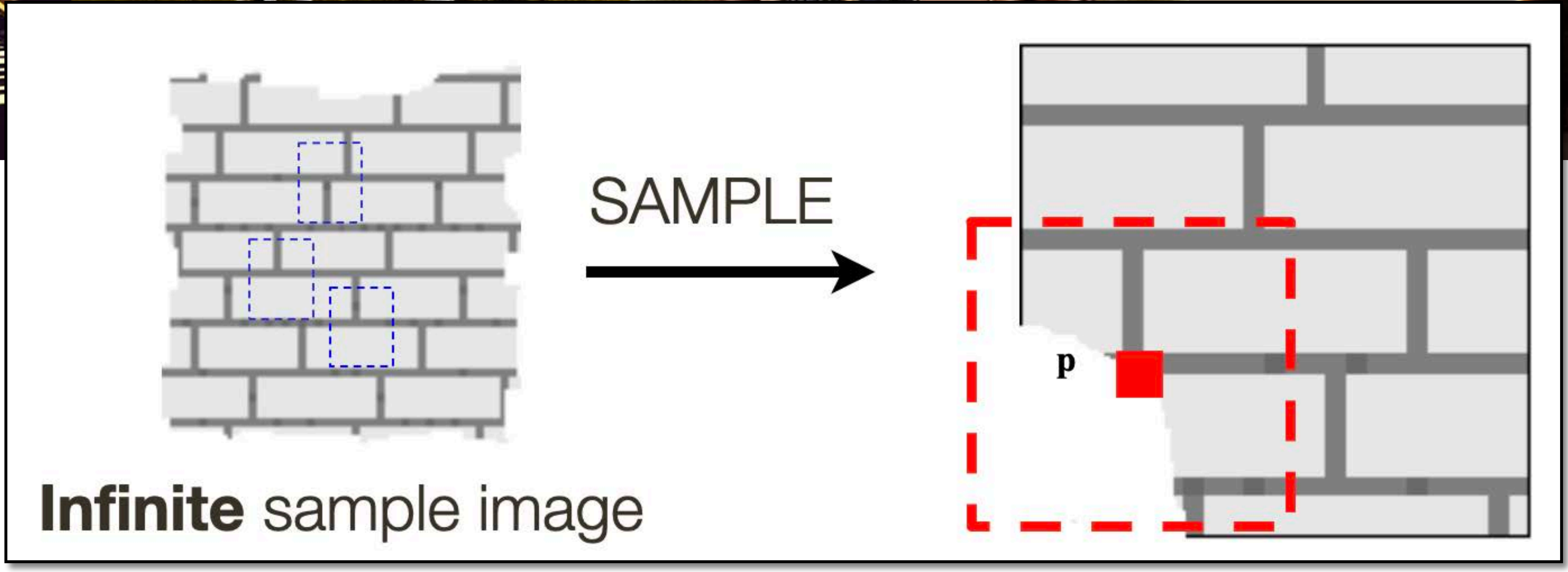
Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Relevant **modern** Computer Vision example

[Rombach et al., 2022] — https://github.com/CompVis/stable-diffusion

Relevant **modern** Computer Vision example

[Rombach et al., 2022] —



Summary

Texture representation is hard

— difficult to define, to analyze

— texture synthesis appears more tractable

Objective of texture **synthesis** is to generate new examples of a texture

— Efros and Leung: Draw samples directly from the texture to generate one pixel at a time. A “data-driven" approach.

Approaches to texture embed assumptions related to human perception