KECE471 Computer Vision

Texture

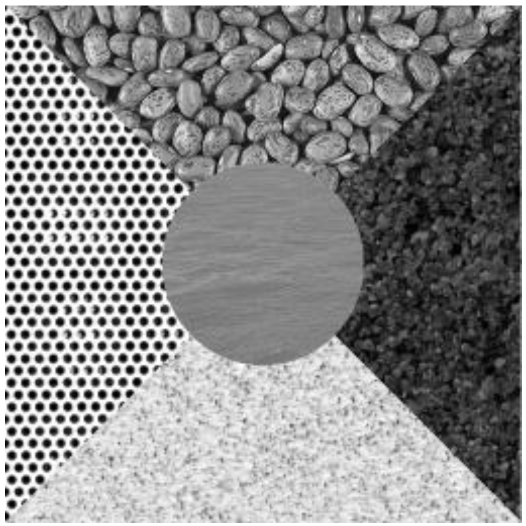
Chang-Su Kim

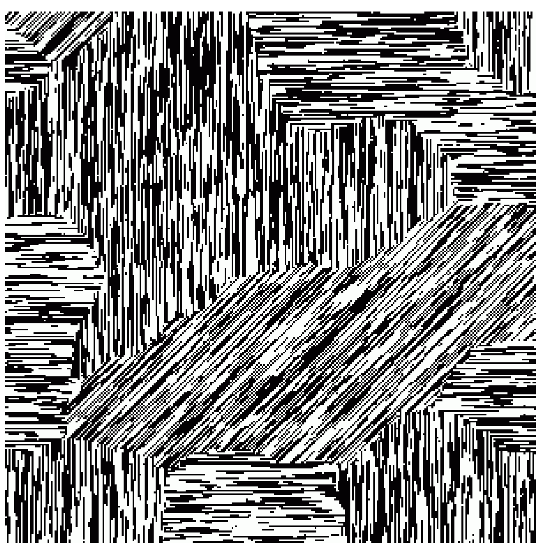
Chapter 9, Computer Vision by Forsyth and Ponce
Note: Most figures were copied from the lecture notes of Dr. Forsyth

Texture

- Topics include
  - Texture segmentation
    - A key issue: how to represent texture?
  - Texture synthesis
    - How to construct large regions of texture from small example images
  - Shape from texture
    - How to approximate the shape of a surface, assuming that textures are identical over the surface

Texture Segmentation

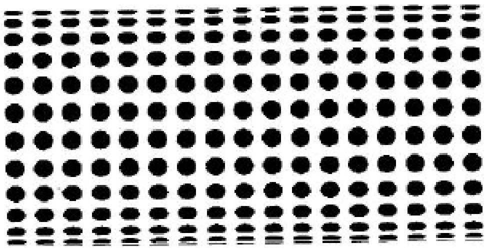




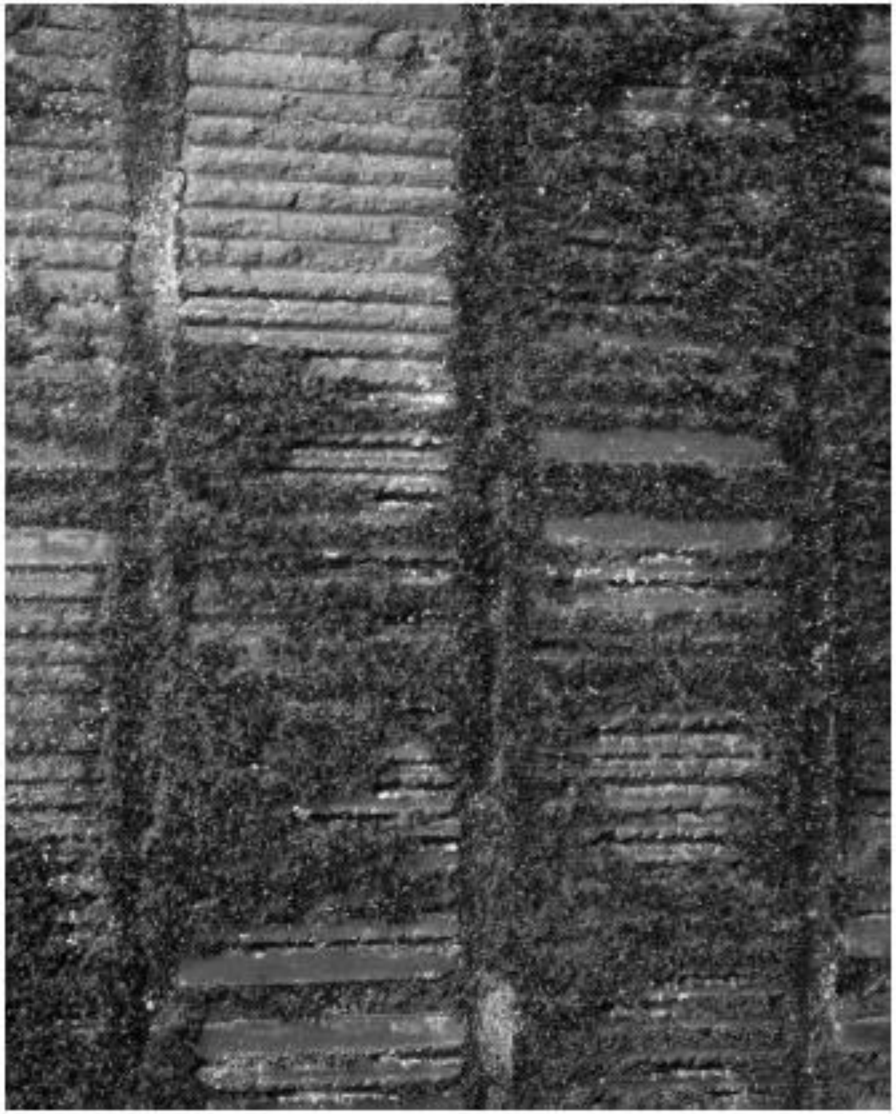
a

b

Shape from Texture



Typical Textured Images



Representing textures

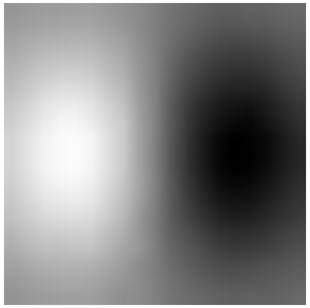
- Textures are made up of stylized subelements, repeated in meaningful ways
- Representation:
  - Find the subelements, and represent their statistics
- But what subelements?
  - Spots and oriented bars at a variety of different scales
- How do we find the subelements?
  - Find subelements by applying filters, looking at the magnitude of the response
- What statistics?
  - Mean, standard deviation, and etc

Filters as Templates

- A filter responds most strongly to pattern elements that look like the filter

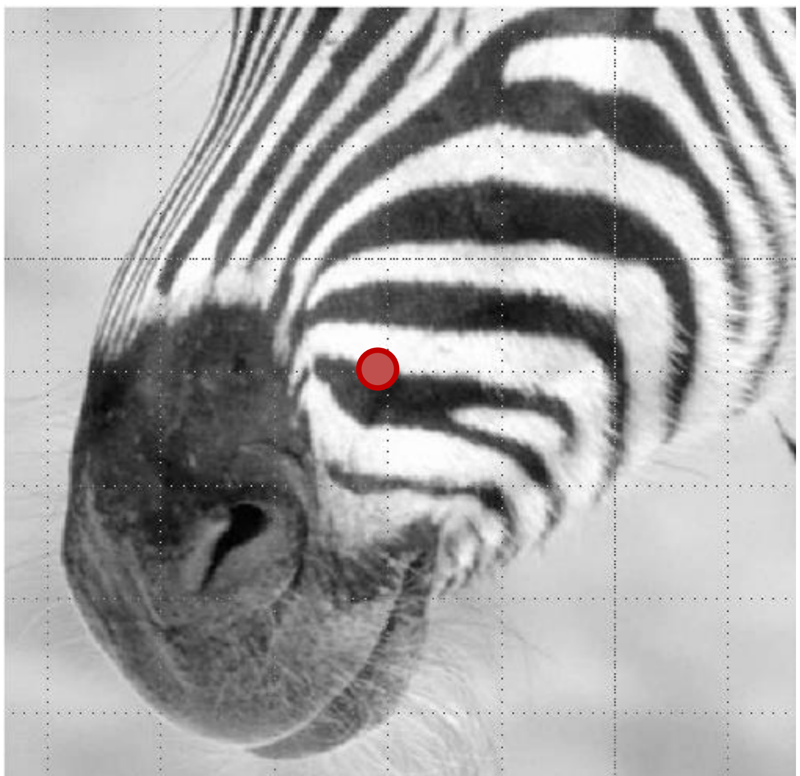
- $g(x, y) = f(x, y) * h(x, y)$

$*$



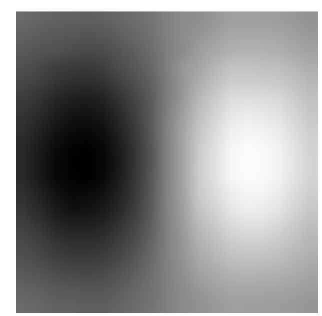
Filters as Templates

- A filter responds most strongly to pattern elements that look like the filter



- $$g(x,y) = f(x,y) * h(x,y)$$
$$= \sum_{m,n} f(x-m, y-m)h(m,n)$$
$$= \sum_{m,n} f(x+m, y+m)h(-m,-n)$$

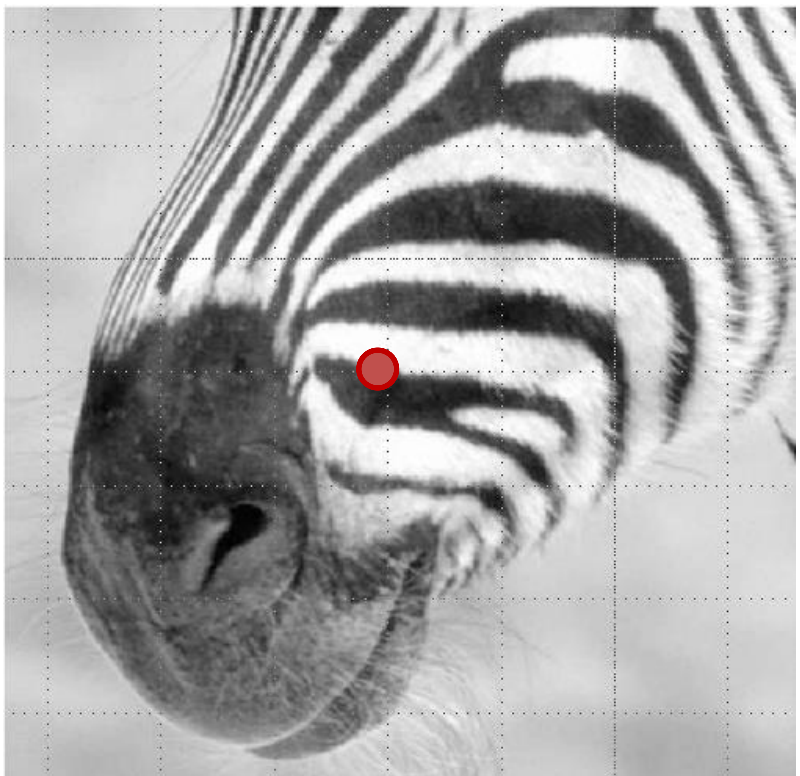
masking with $h(-x,-y)$

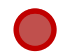


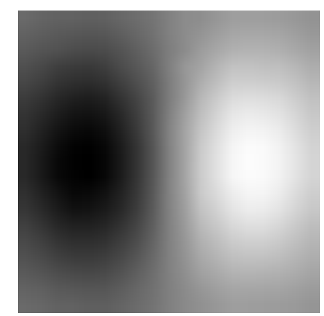
The value of g at the marked circle is obtained by multiplying the corresponding pixels on the mask and the input image and then summing up the products

Filters as Templates

- A filter responds most strongly to pattern elements that look like the filter



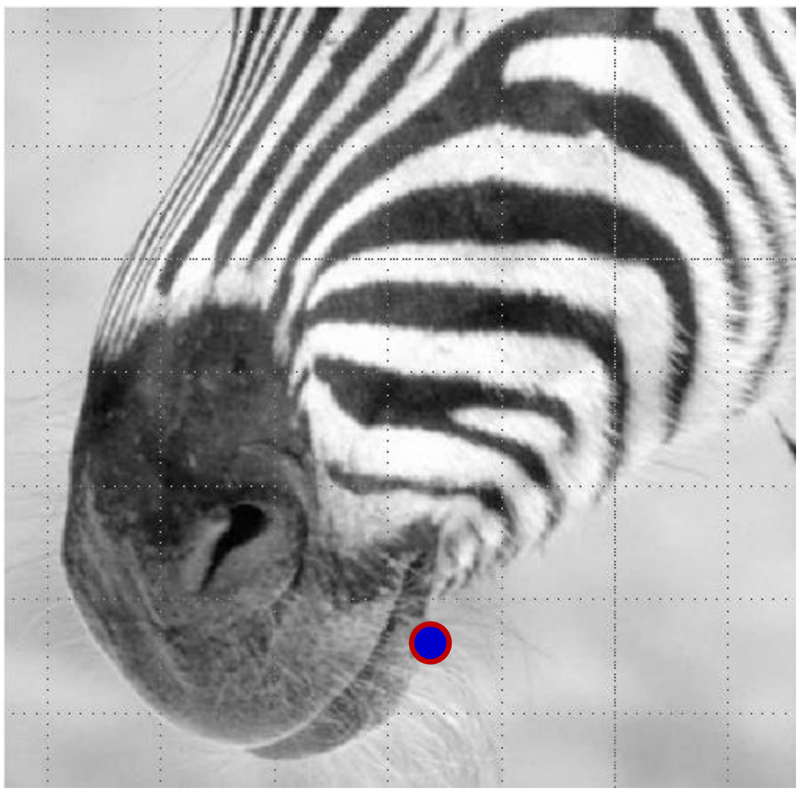
 = ($\mathbf{a}$, $\mathbf{b}$)
= inner product of **a** and **b**

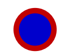


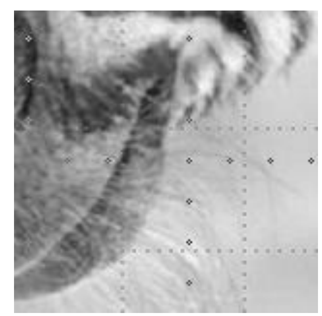
The value of g at the marked circle is obtained by multiplying the corresponding pixels on the mask and the input image and then summing up the products

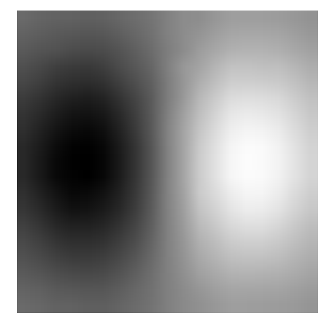
Filters as Templates

- A filter responds most strongly to pattern elements that look like the filter



 = (**a**, **b**)
= inner product of **a** and **b**

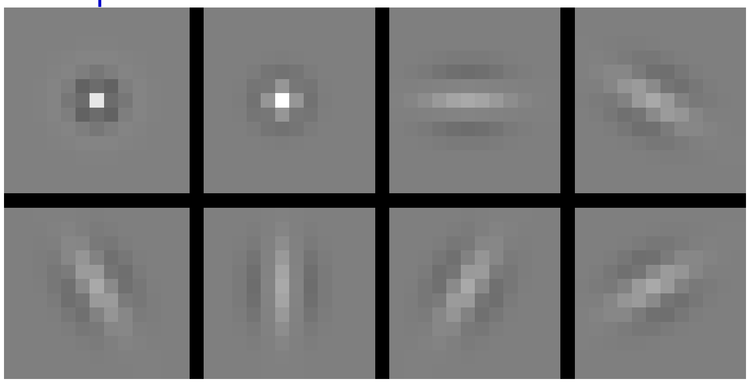




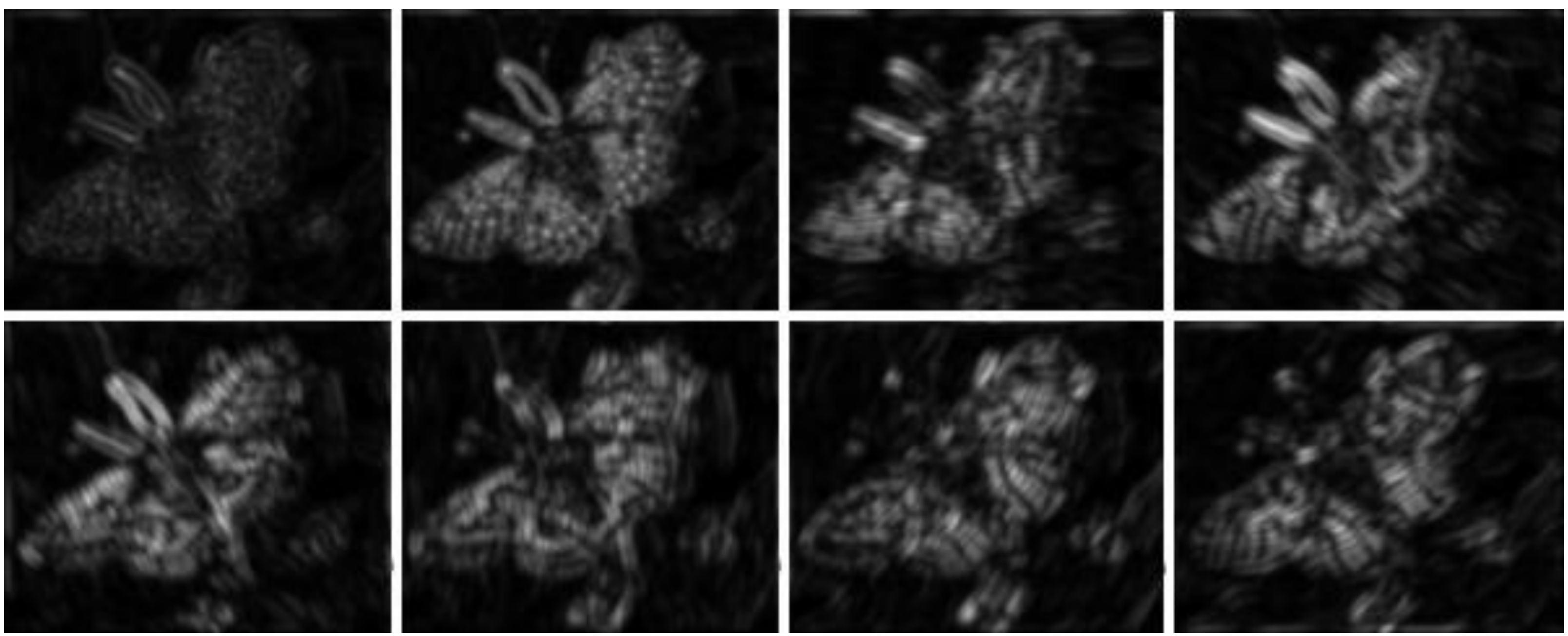
The value of g at the marked circle is obtained by multiplying the corresponding pixels on the mask and the input image and then summing up the products

Extracting Image Structure with Filter Banks at a Fine Scale

two spot filters and six bar filters

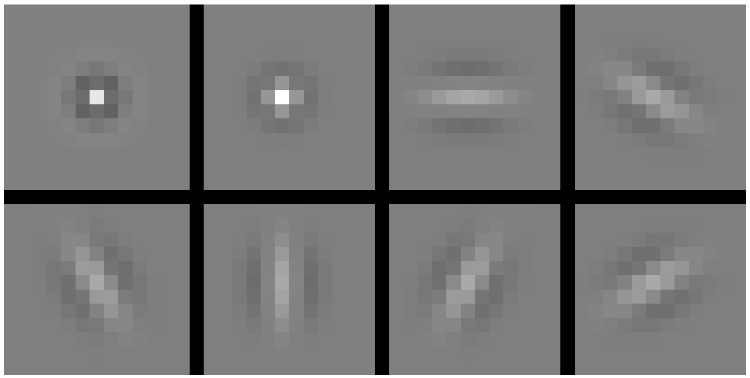


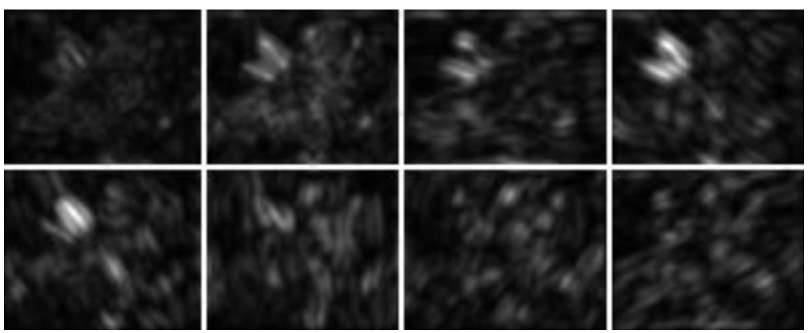
input



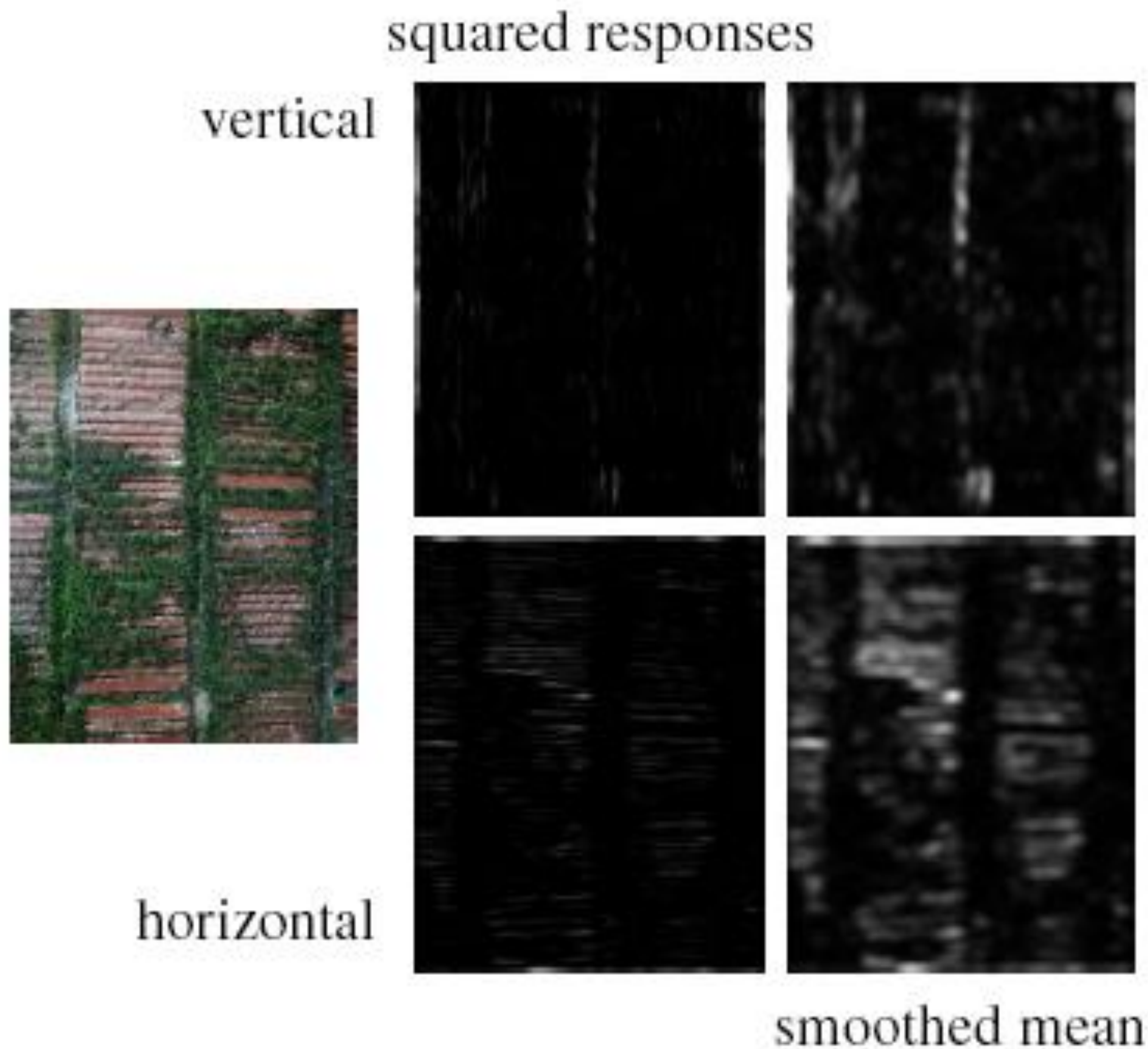
output

Extracting Image Structure with Filter Banks at a Coarse Scale



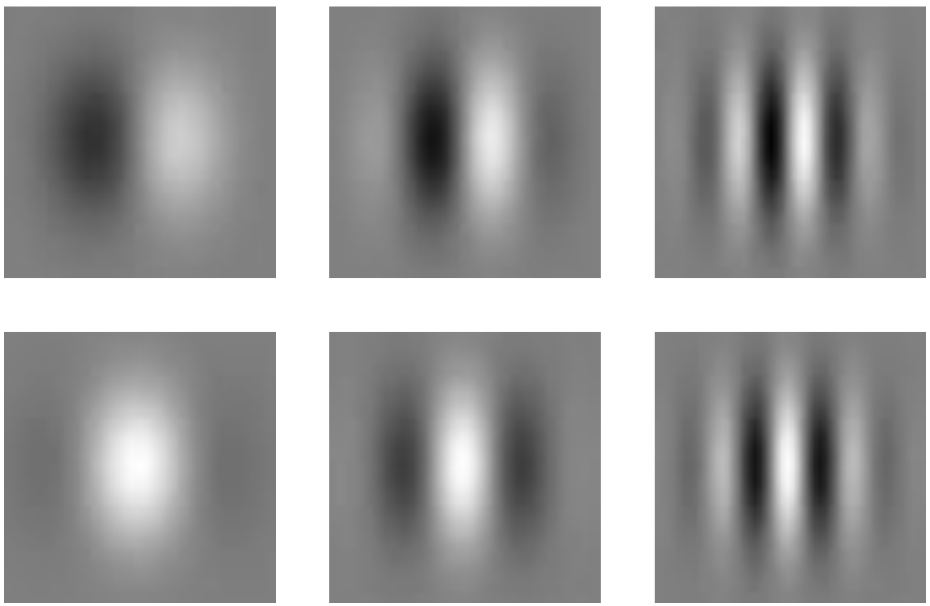


A Texture Classification System



black: neither horizontal
nor vertical
dark gray: horizontal
bright gray: vertical
white: both

Gabor Filter Kernels



$$G_{\text{antisymmetric}}(x,y) = \sin(\alpha x + \beta y)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$G_{\text{symmetric}}(x,y) = \cos(\alpha x + \beta y)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Texture Representation

- In the last classification system,

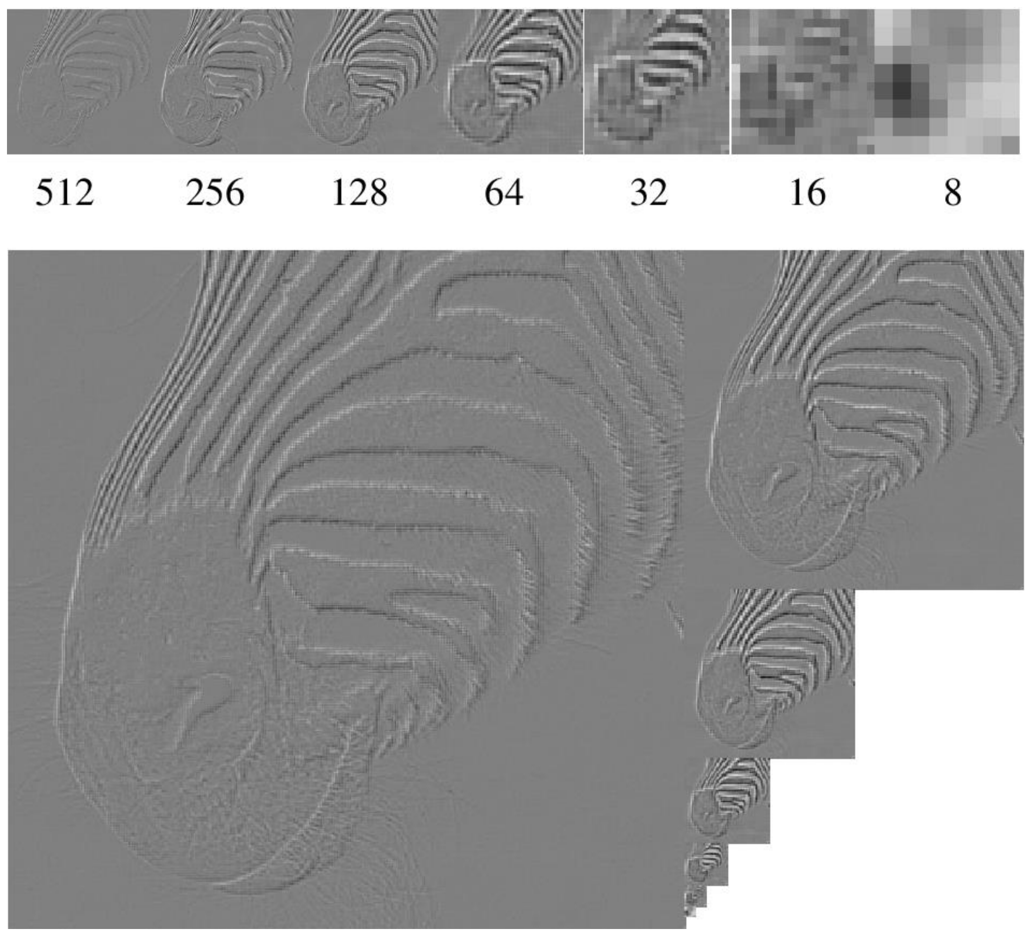
$$\begin{bmatrix} \text{Average of sqaured responses of the vertical bar filter over a window} \\ \text{Average of sqaured responses of the horizontal bar filter over a window} \end{bmatrix}$$

- Of course, other forms are also possible

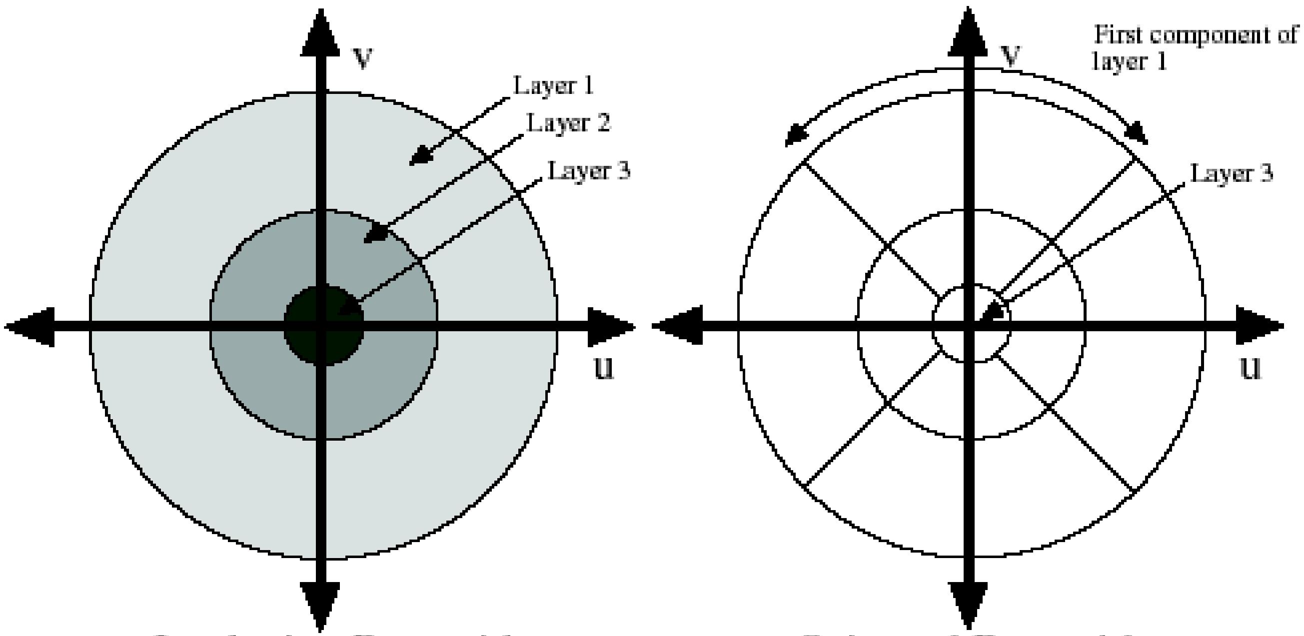
$$\begin{bmatrix} \text{Mean(responses of filter 1)} \\ \text{STD(responses of filter 1)} \\ \text{Mean(responses of filter 2)} \\ \text{STD(responses of filter 2)} \\ \vdots \end{bmatrix}$$

Laplacian Pyramid

- Each level can be interpreted as a bandpass output



Laplacian Pyramid and Oriented Pyramid

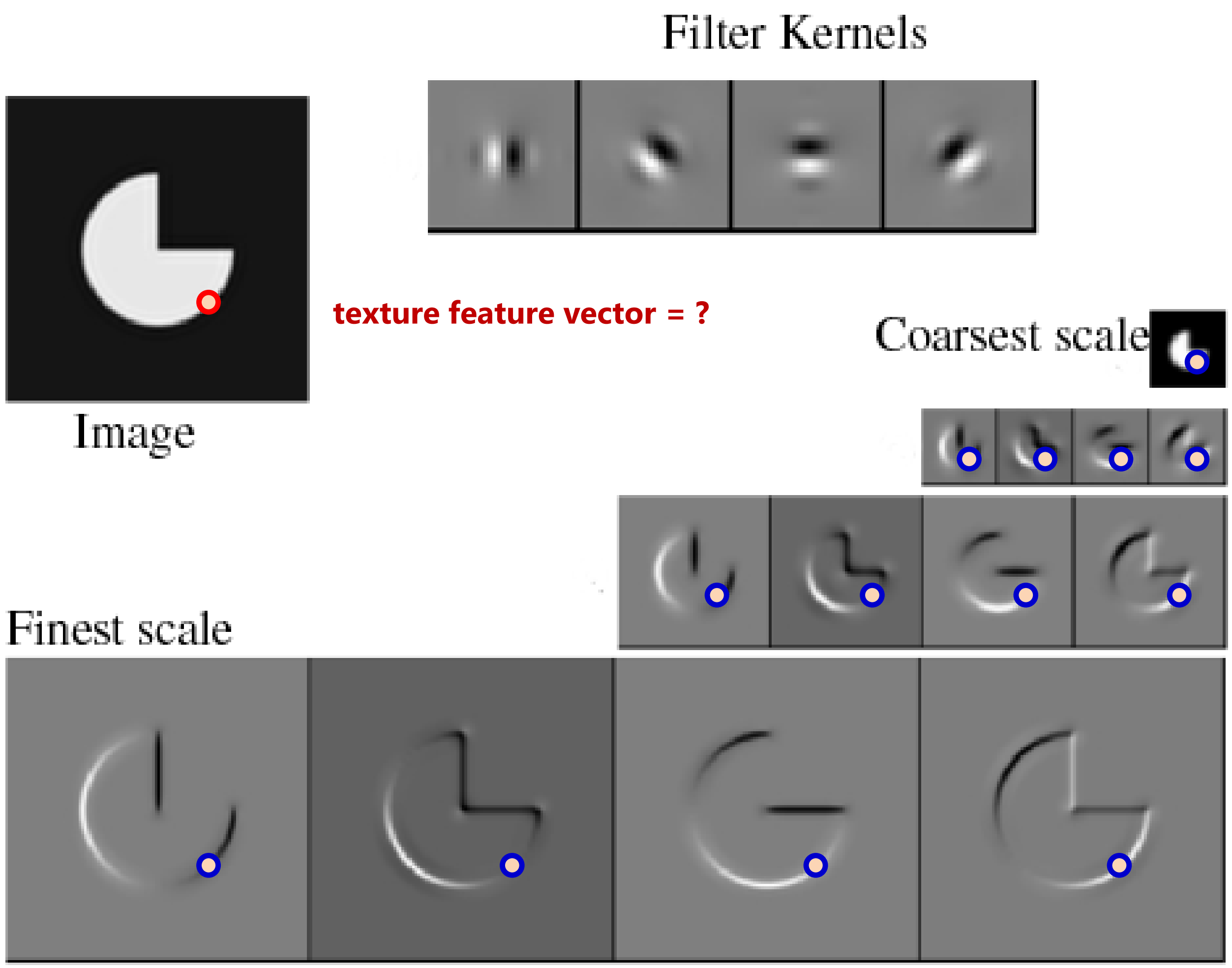


Laplacian Pyramid Oriented Pyramid

Oriented Pyramids

- Laplacian pyramid is orientation independent
- Apply an oriented filter to analyze orientations at each layer
  - this represents image information at a particular scale and orientation

Oriented Pyramids



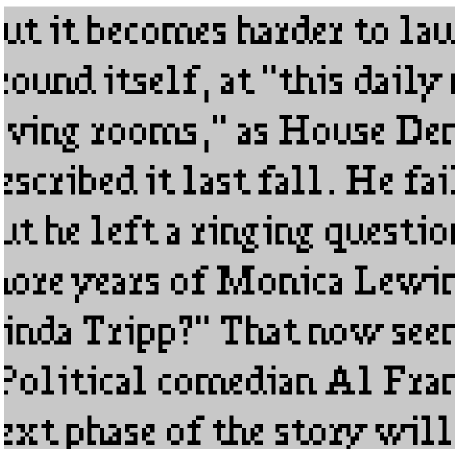
Reprinted from "Shiftable MultiScale Transforms," by Simoncelli et al., IEEE Transactions on Information Theory, 1992, copyright 1992, IEEE

Texture Synthesis

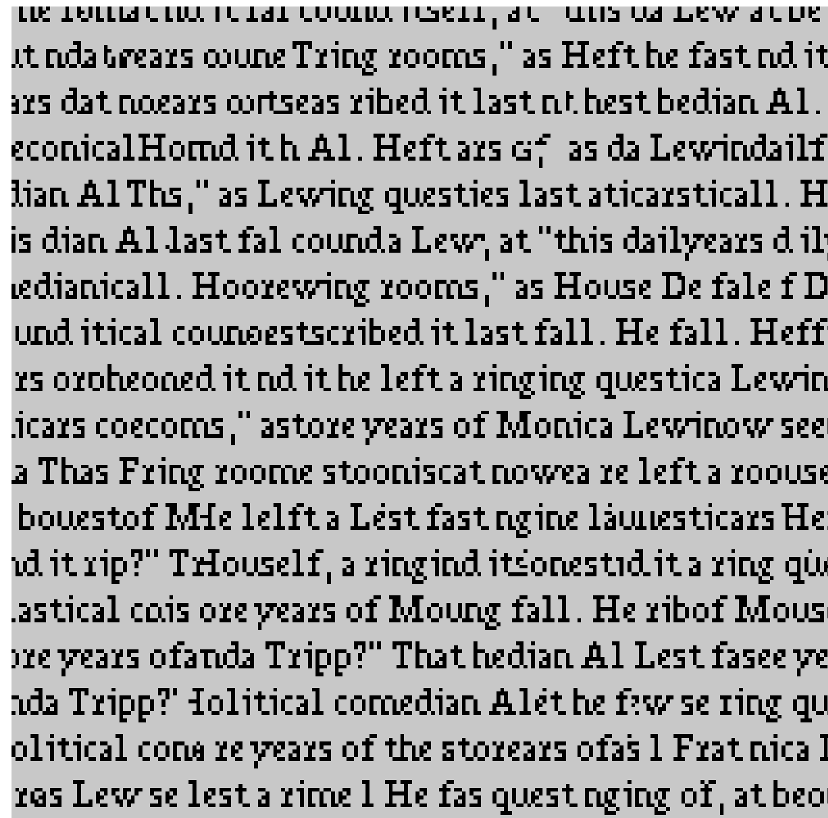
- Use an example image as a source of probability model
- Procedure
  1. Choose pixel values by matching neighborhood
  2. Filling in the new pixel
- Matching process
  – Look at pixel differences
  – Count only synthesized pixels

Texture Synthesis

Example



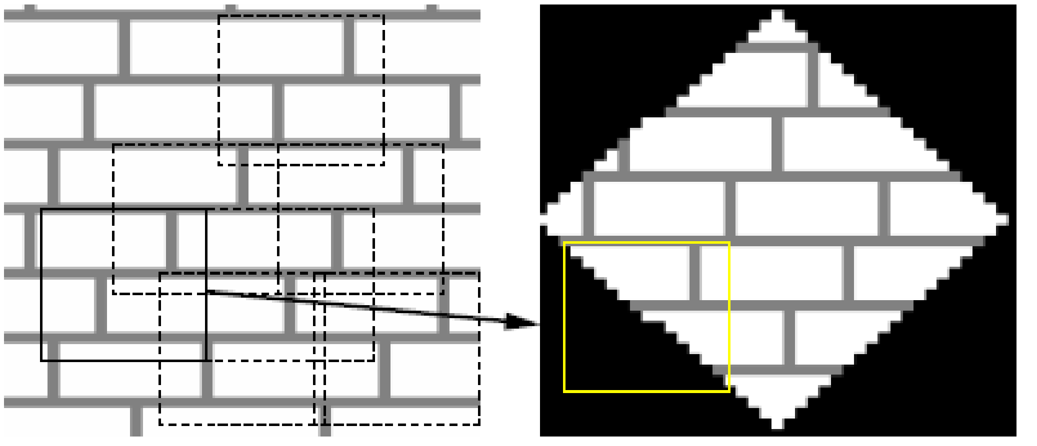
Synthesized Texture



Texture Synthesis Algorithm 1

- Alexei A. Efros and Thomas K. Leung
  - Texture Synthesis by Non-parametric Sampling
  - IEEE International Conference on Computer Vision, Sept. 1999

Overview



- Given a sample texture image (left), a new image is synthesized one pixel at a time (right)
- To synthesize a pixel,
  - Find all neighborhoods in the sample image that are similar to the pixel's neighborhood
  - Choose randomly one neighborhood and take its center to be the newly synthesized pixel

Finding Candidates

- All neighborhoods similar to the pixel's neighborhood
  - Compute d_{min}
  - Find neighborhoods satisfying $d<(1+\varepsilon)d_{min}$

- Distance measure
  - Gaussian weighted SSD (sum of squared differences)

Examples

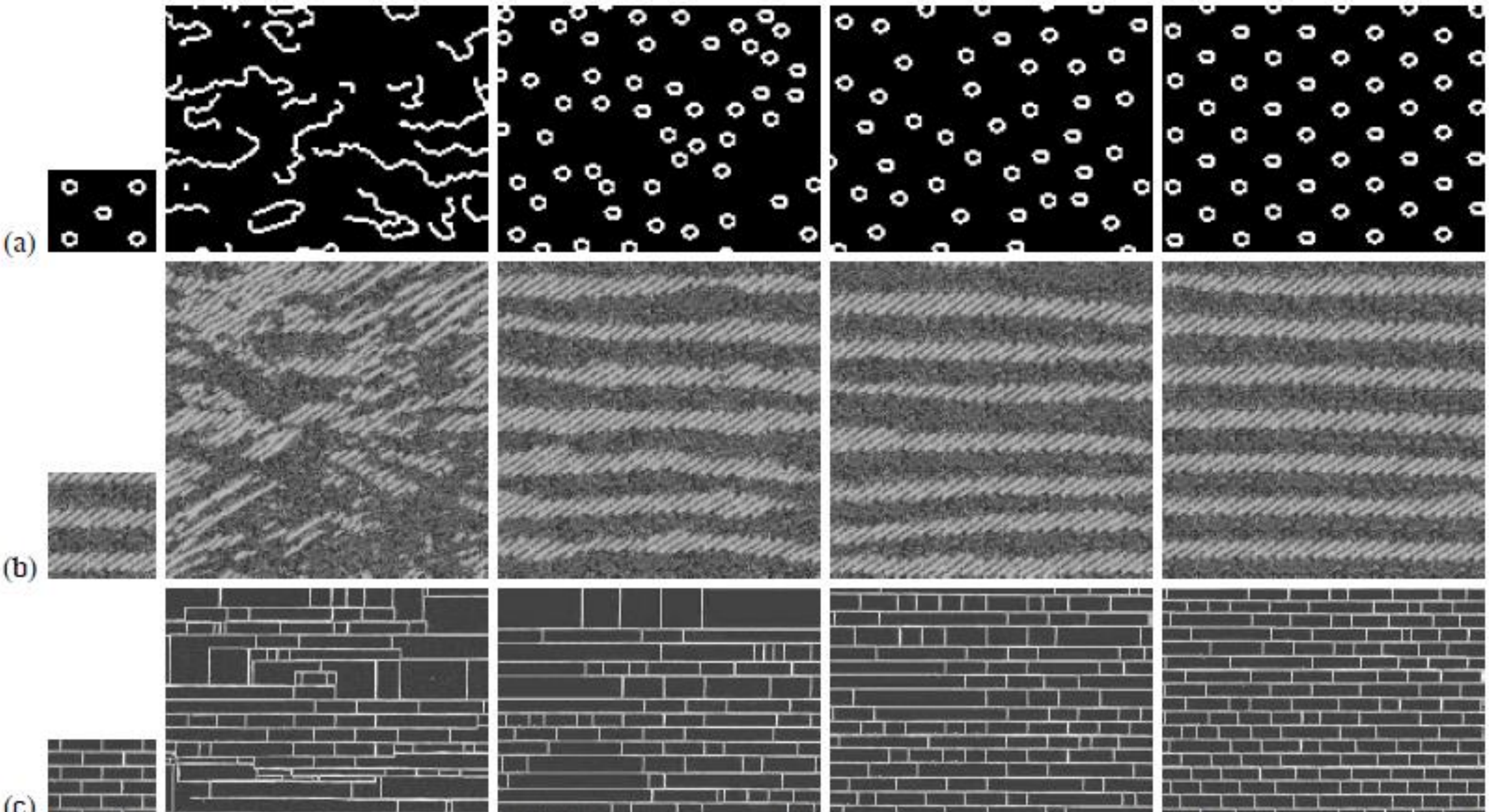
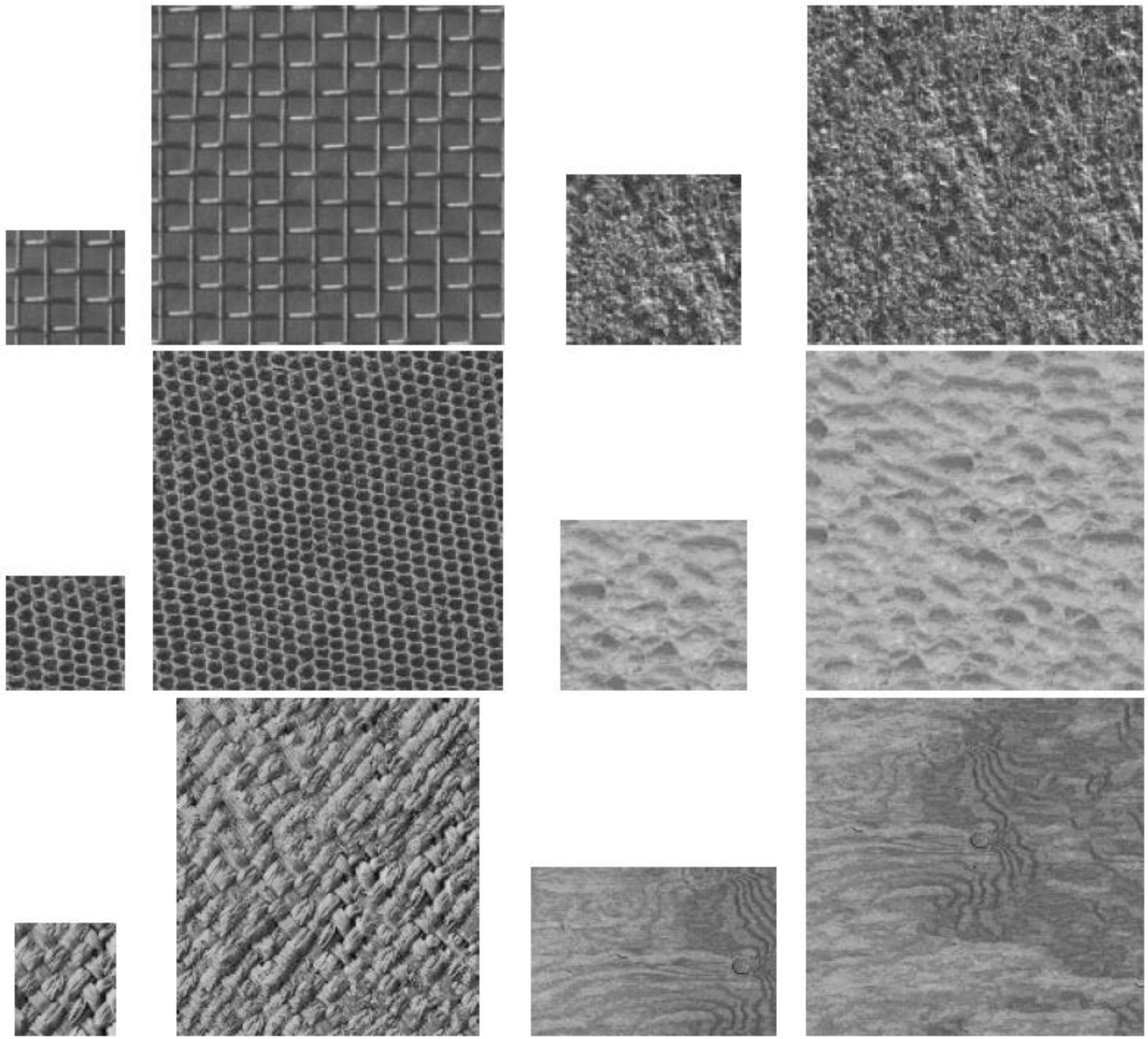
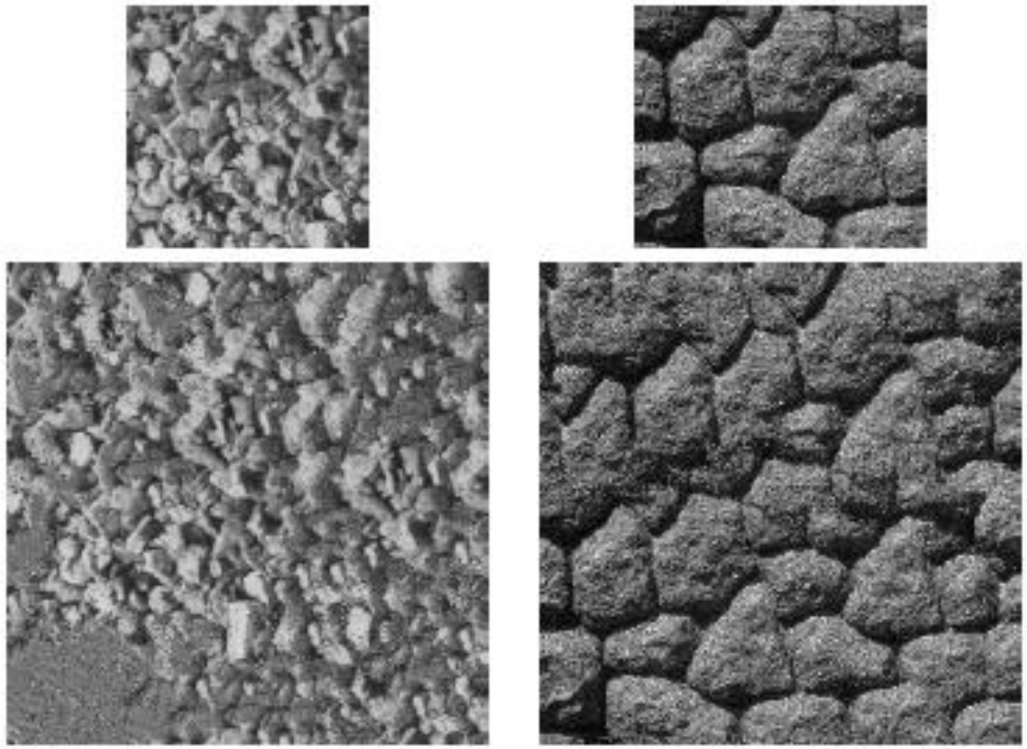


Figure 2. Results: given a sample image (left), the algorithm synthesized four new images with neighborhood windows of width 5, 11, 15, and 23 pixels respectively. Notice how perceptually intuitively the window size corresponds to the degree of randomness in the resulting textures. Input images are: (a) synthetic rings, (b) Brodatz texture D11, (c) brick wall.

Examples



Failure Examples



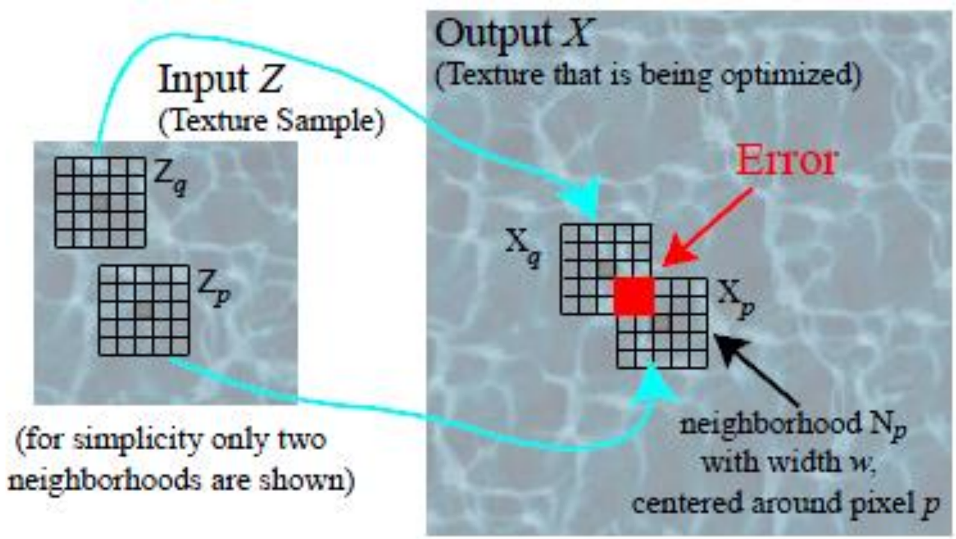
(a) (b)

Figure 5. Failure examples. Sometimes the growing algorithm "slips" into a wrong part of the search space and starts growing garbage (a), or gets stuck at a particular place in the sample image and starts verbatim copying (b).

Texture Synthesis Algorithm 2

- Vivek Kwatra, Irfan Essa, Aaron Bobick, Nipun Kwatra
  - Texture Optimization for Example-based Synthesis
  - ACM SIGGRAPH 2005

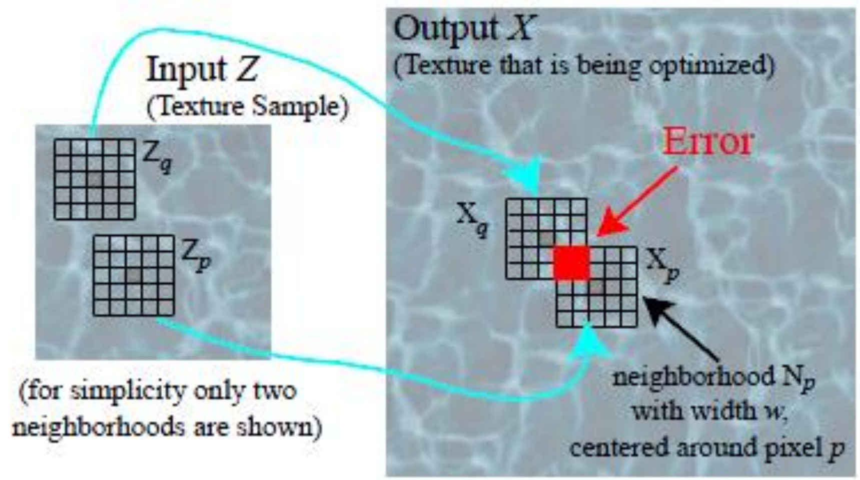
Cost Function



$$E = \sum_{p \in X} \left\| \mathbf{x}_p - \mathbf{z}_p \right\|^2$$

- X: synthesized texture
- Z: sample texture
- $\mathbf{x}_p$: block around p
- $\mathbf{z}_p$: most similar block to $\mathbf{x}_p$

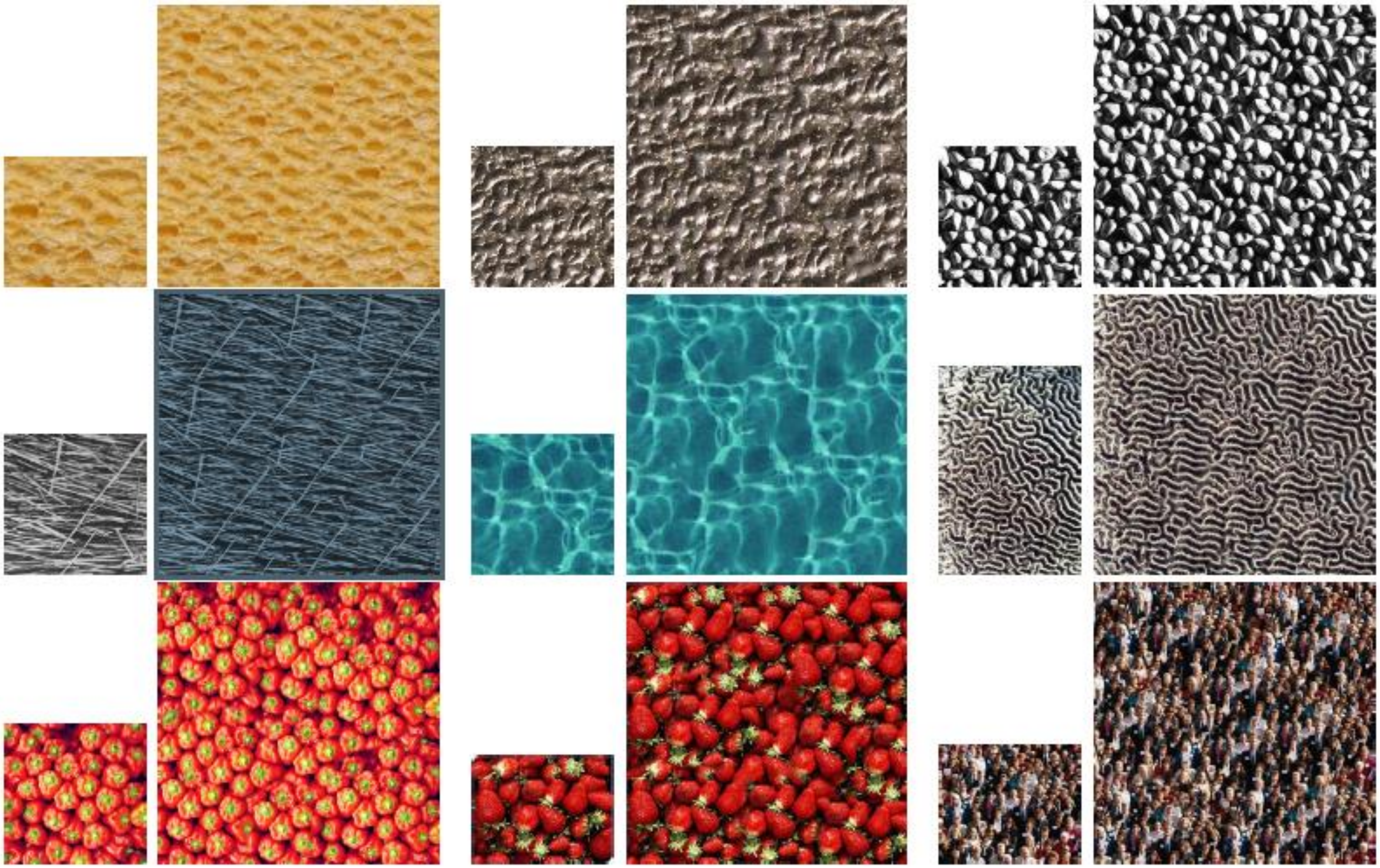
Two-Step Minimization



- E-step
  - Each pixel in X is assigned the average of pixels in the related patches $\mathbf{z}_q$
- M-step
  - For each $\mathbf{x}_p$, find the most similar $\mathbf{z}_p$

Examples

Examples



Texture Synthesis Algorithm 3

- Michael Ashikhmin
  - Synthesizing Natural Textures
  - ACM Symposium on Interactive 3D Graphics, 2001

Examples

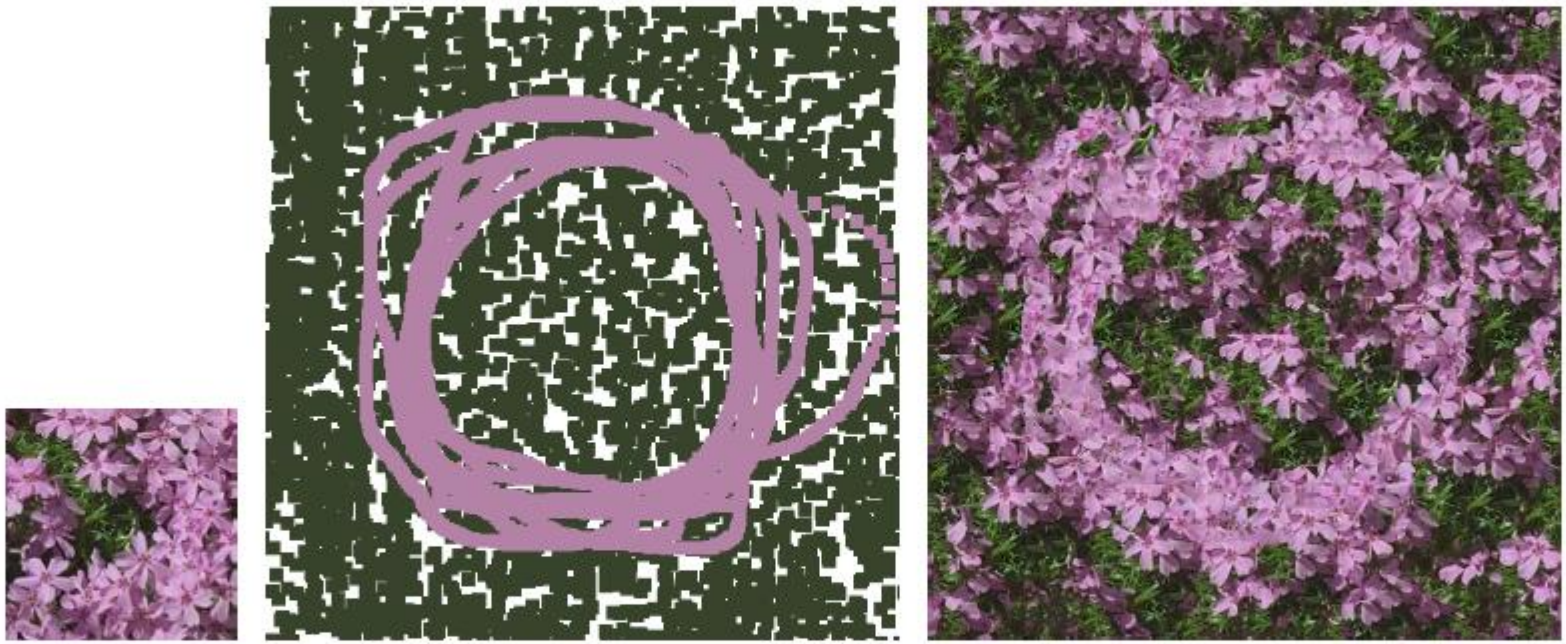


Figure 1: *An example of user controlled synthesis done by a first time user of the system. Left: input sample, Center: user-drawn target image, Right: synthesized result created in time short enough for an interactive loop. This particular result was subjectively chosen by the user from a few trials with the same target image.*

Examples

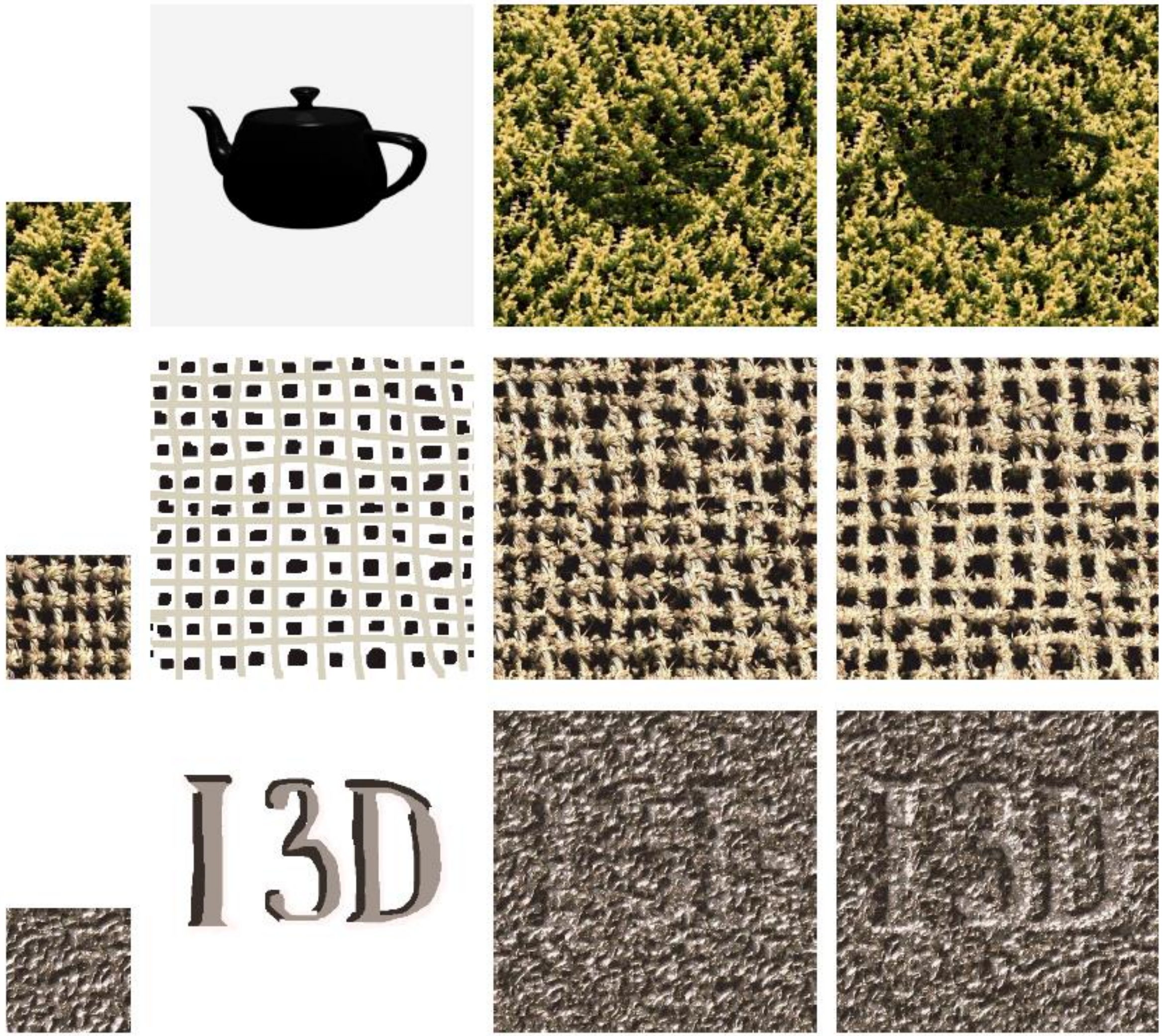


Figure 10: *User controlled texture synthesis results. All input textures are from VisTex dataset [1], 192x192 pixels. All other images are 500x500 pixels. White regions in target images did not receive user input. Left to right: input sample, target image, single pass result, result after more iterations (top to bottom: 5,5,8,20 total passes)*