Texture

Semester 2, 2021

Kris Ehinger

Shape and texture

Images: https://polyhaven.com

Inpainting demo

- https://www.nvidia.com/research/inpainting/index.html

Outline

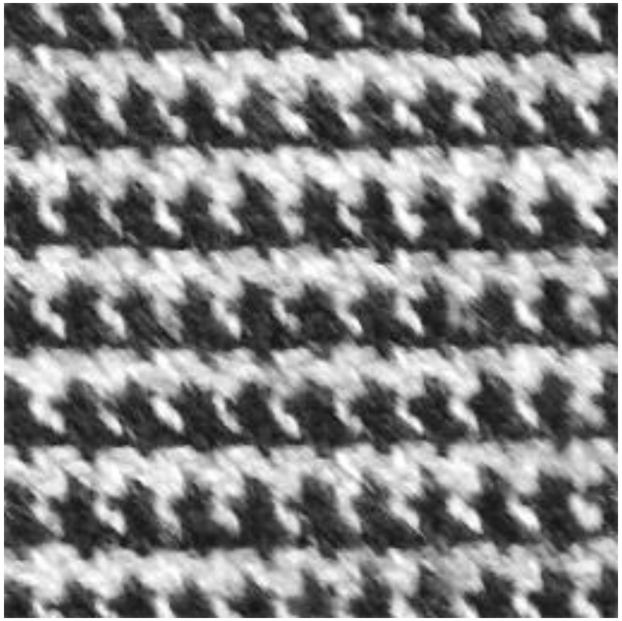
- What is texture?
- Texture synthesis
- Texture transfer

Learning outcomes

- Explain parametric and non-parametric methods for representing and synthesizing texture
- Explain common applications of texture synthesis
- Explain an algorithm for texture transfer (Neural Style Transfer)

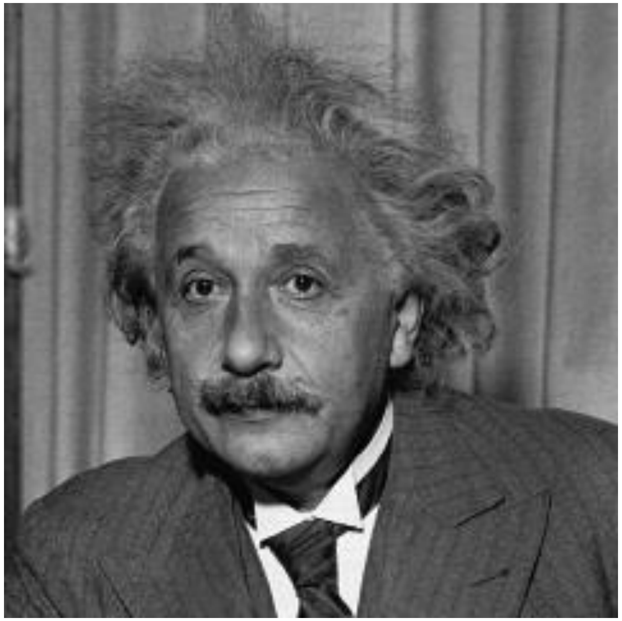
What is texture?

What is texture?



A

B



C

What is texture?

- A definition from image processing:

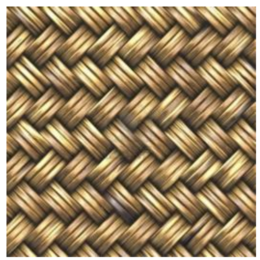
  Texture is an region with **spatial stationarity** (same statistical properties everywhere in the region)

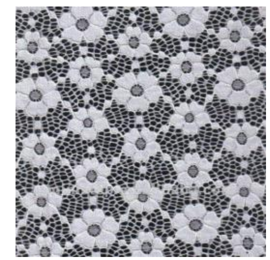
- A definition from computer graphics:

  Texture is a 2D surface applied to a 3D model

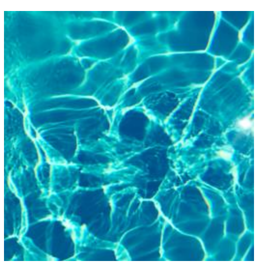
Types of texture

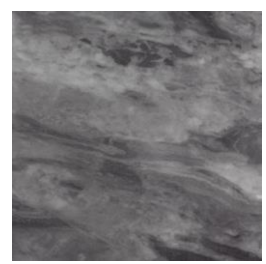
- Periodic texture – has a subregion that repeats in a regular pattern



- Stochastic (aperiodic) texture – generated by a random process

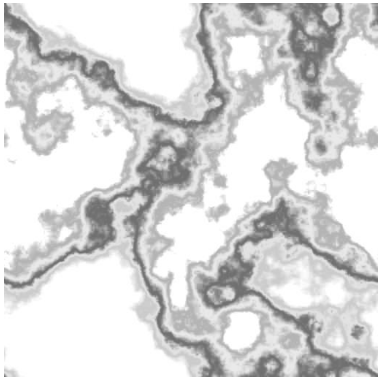


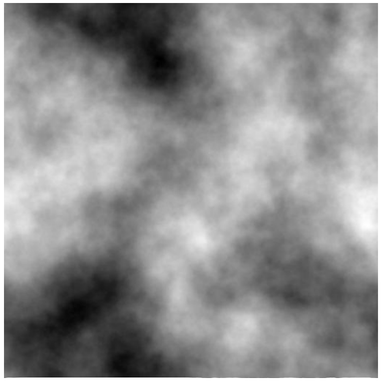




Images: Describable Textures Dataset (DTD), Efros & Freeman (2001)

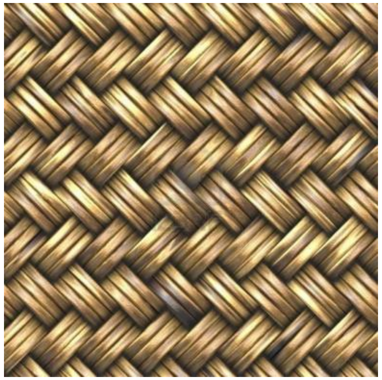
Texture models

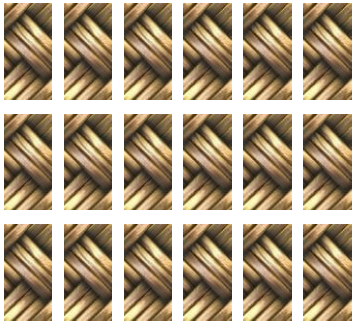
- **Parametric** models: represent texture with a set of adjustable parameters





- **Non-parametric** (stitching) models: represent texture as image patches



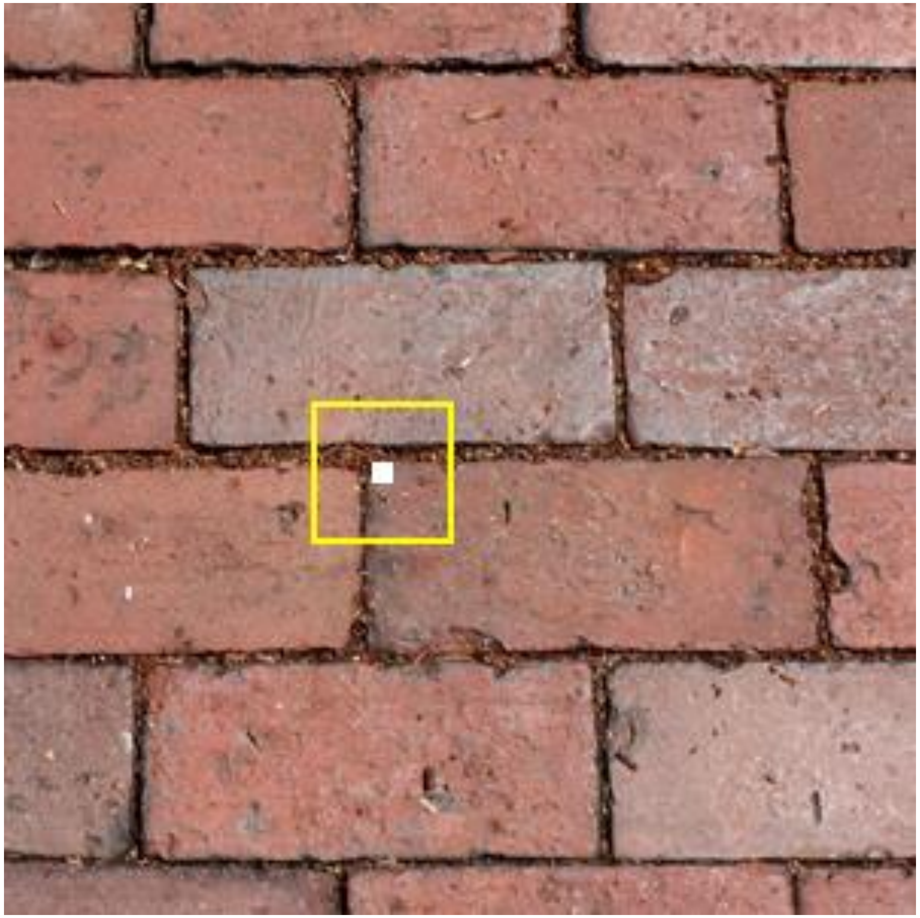


Why model texture?

- Texture synthesis – create more of a texture
  - Textures for computer graphics, video games, etc.
  - Image inpainting
- Texture transfer
  - Artistic effects
  - Online shopping

Texture synthesis

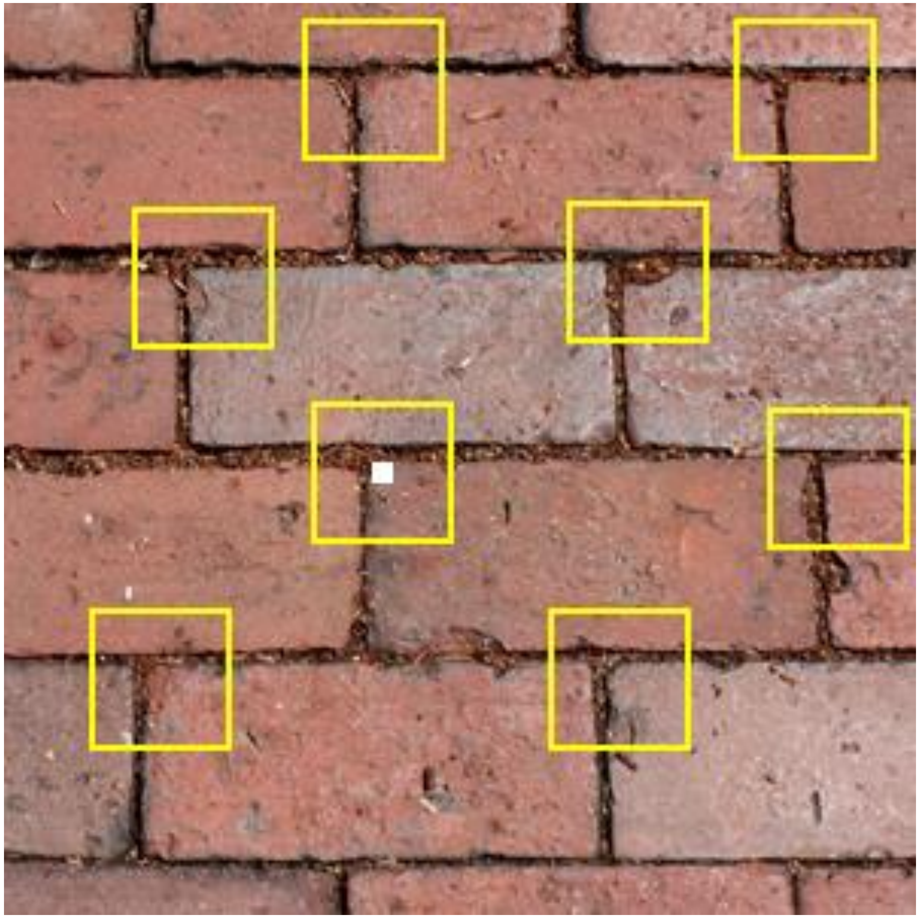
Image stitching approach



How do you fill in the missing data?
Look at a neighbourhood around this patch…

Image: Ehinger & Rosenholtz (2012)

Image stitching approach

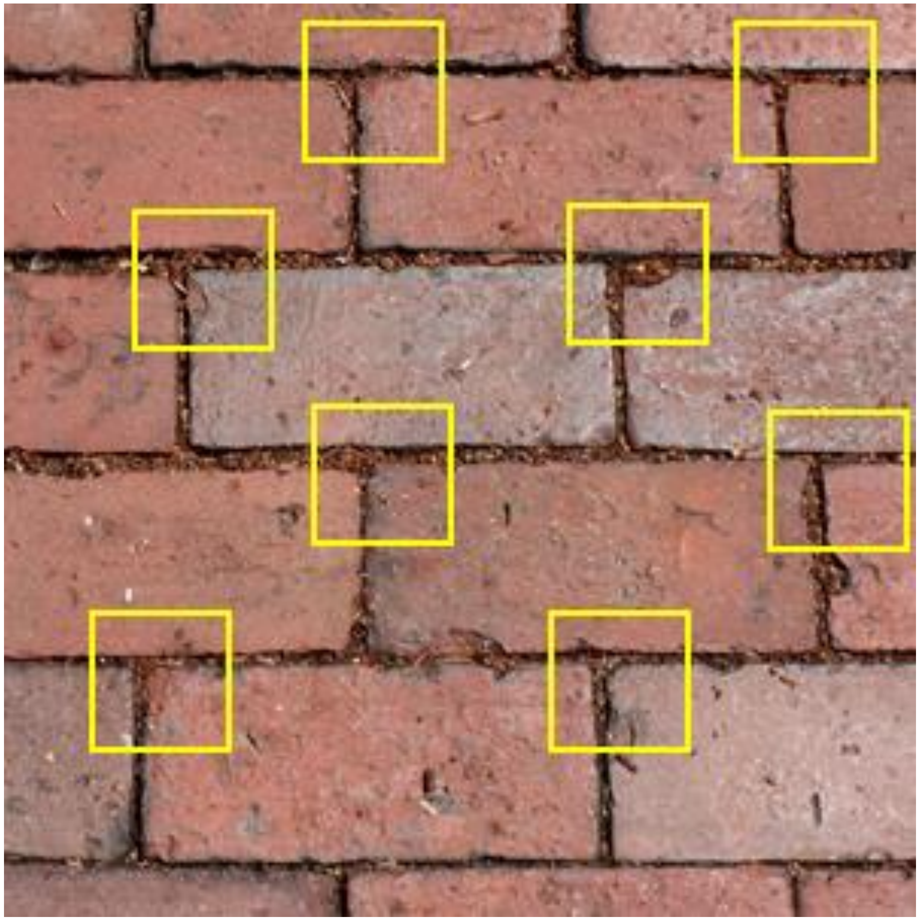


How do you fill in the missing data?
Look at a neighbourhood around this patch…

and find similar neighbourhoods in other parts of the texture.

Image: Ehinger & Rosenholtz (2012)

Image stitching approach



Original patch:

Similar patches:

Select probable value for the missing pixel, based on the similar patches:

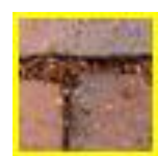
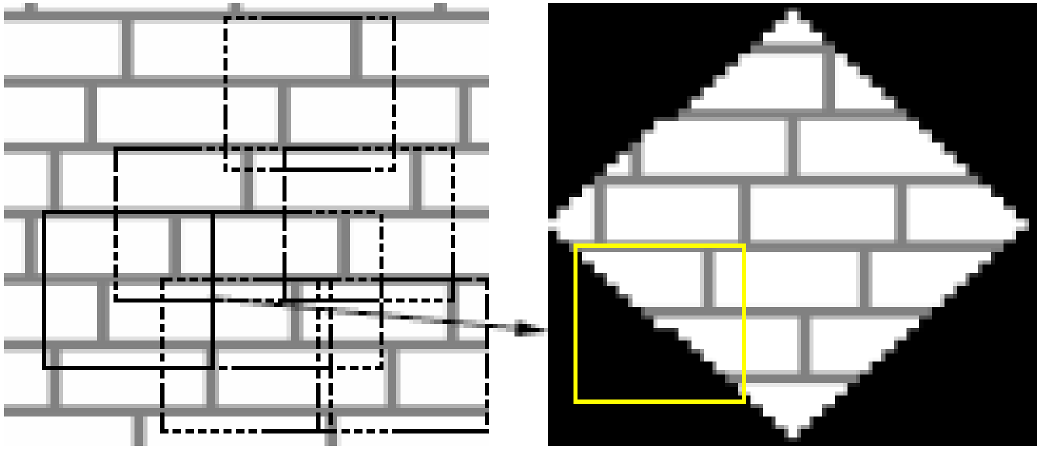


Image: Ehinger & Rosenholtz (2012)

Non-parametric texture synthesis

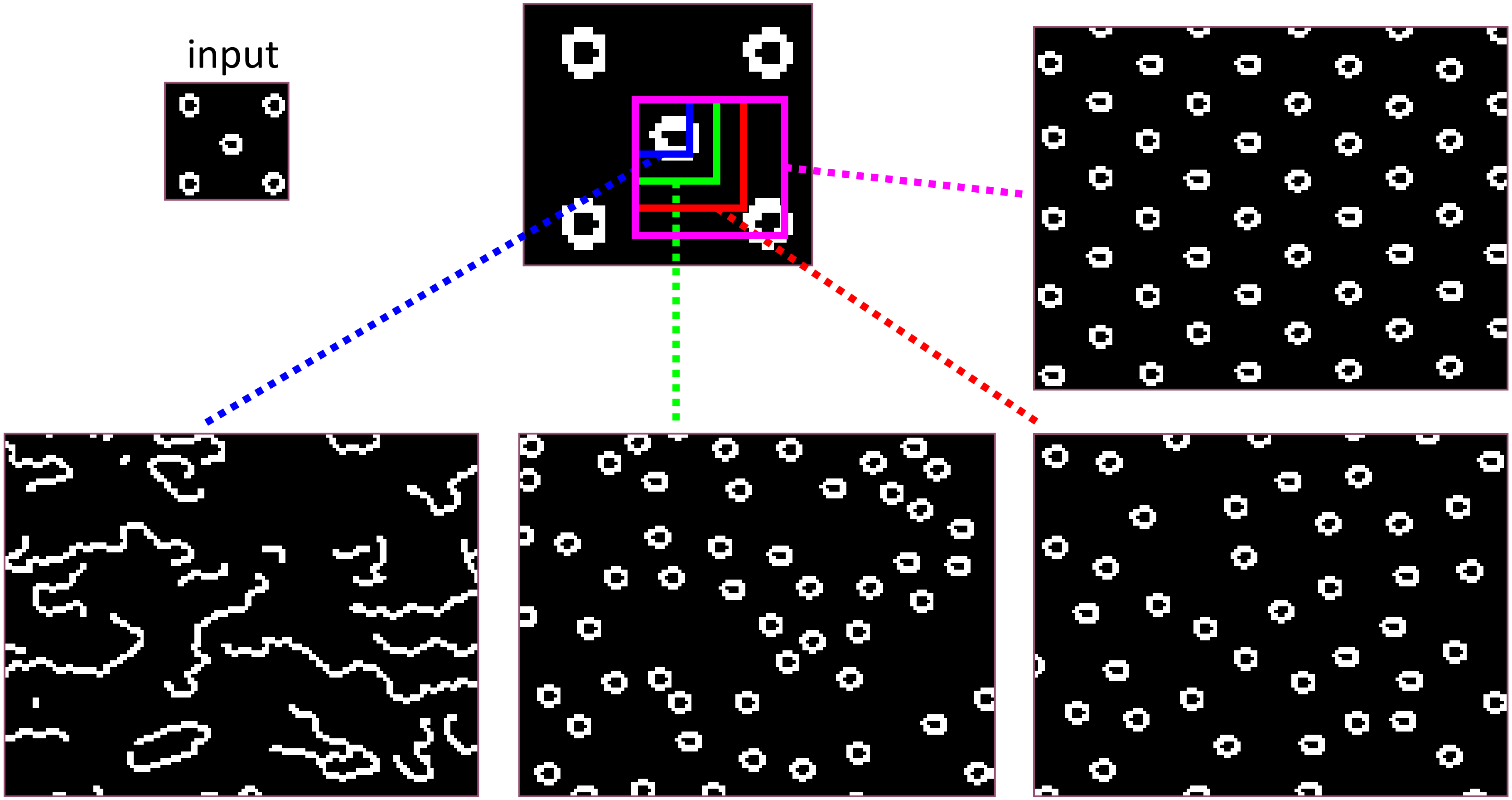


1. Randomly sample a small (e.g., 3 x 3 pixel) patch from the original image
2. Spiral outward, filling in missing pixels by finding similar neighborhoods in the original texture

(Neighbourhood size is a free parameter that specifies how stochastic the texture is)

Efros & Leung (1999)

Neighbourhood size



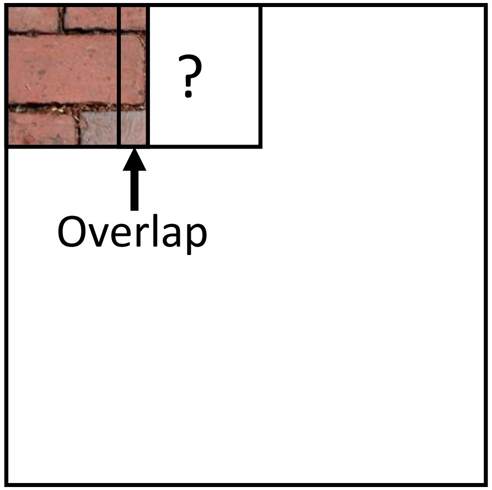
Slide: A. Efros

Image quilting

- Efficient patch-based texture synthesis (Efros & Freeman, 2001)
- Use existing patches of texture to synthesis more texture; main problem is connecting them together without visible artefacts/seams
- “Corrupt Professor’s Algorithm”
  - Plagiarize as much of the source image as you can
  - Then try to cover up the evidence

Image quilting algorithm

- Choose patch and overlap size
- Initialize with a random patch
- For each subsequent patch:
  - Find a patch in the original texture that is most similar to this region, considering only the pixels in the overlap region
  - Seamlessly paste in patch by cutting along a path with minimum overlap error



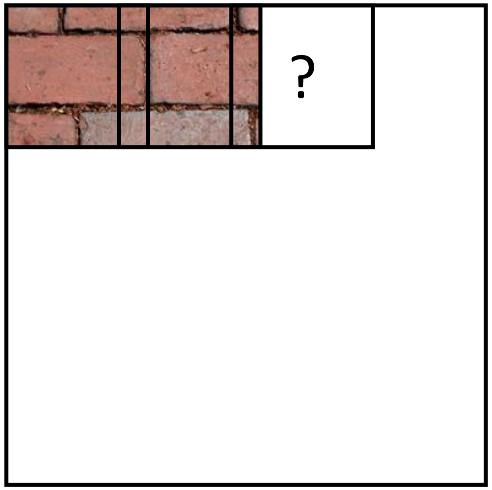
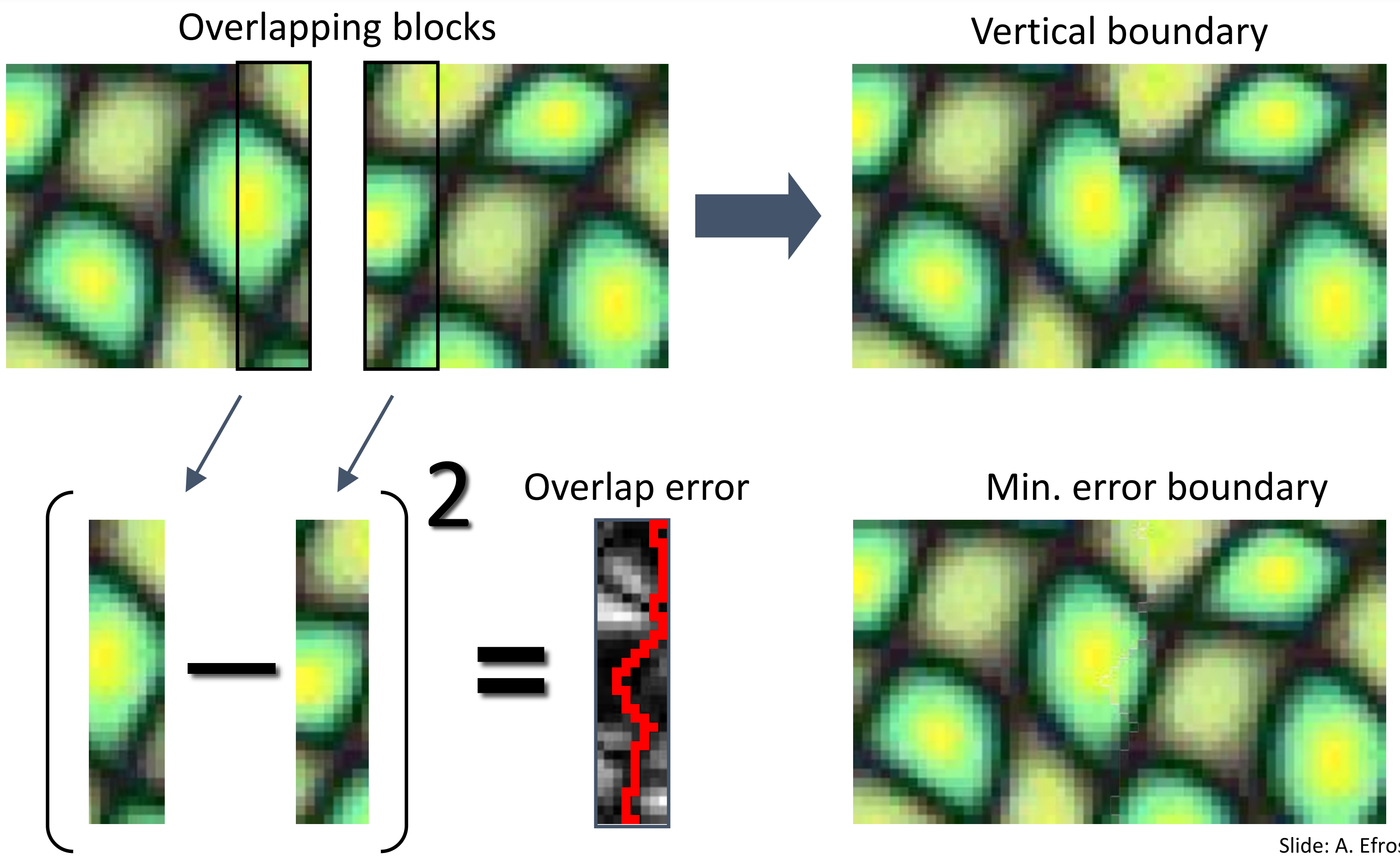


Image quilting algorithm

Animation: D. Lanman

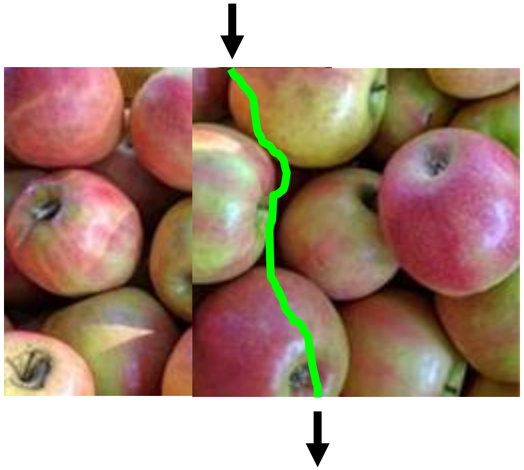
Image quilting



Slide: A. Efros

Graph cuts

- Represent neighbouring pixels as a graph
- Edge weight = overlap error
- Problem: Find path through graph with minimum total overlap error



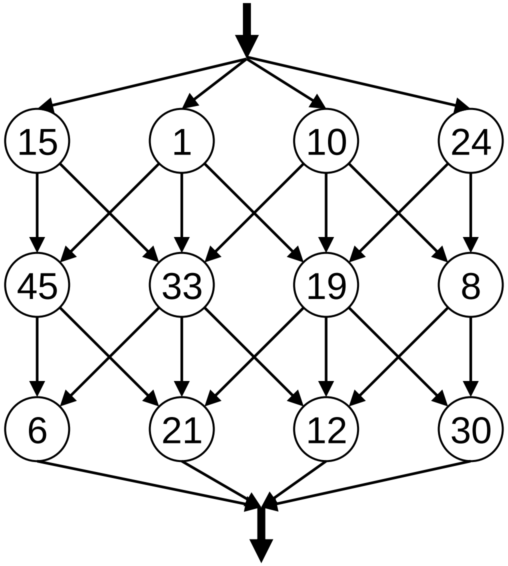
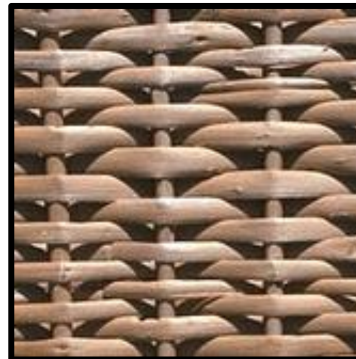


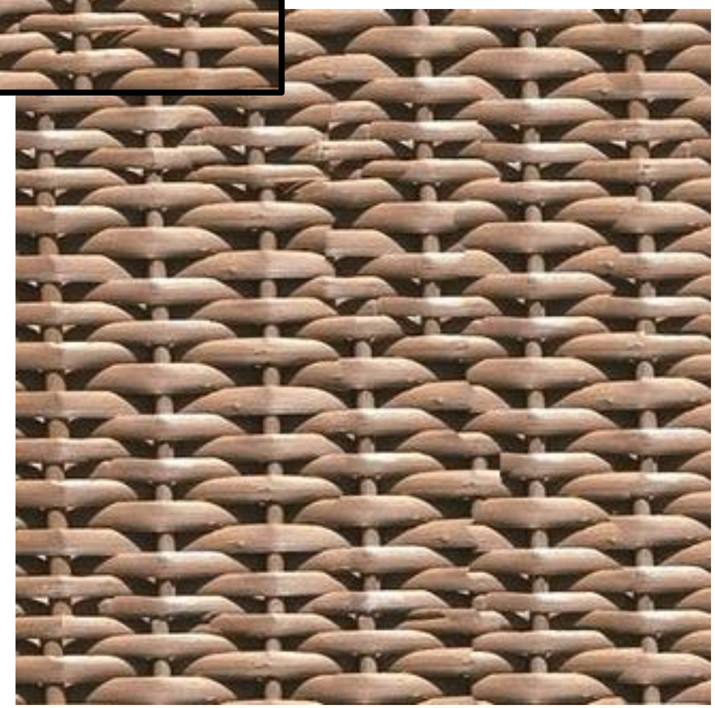
Image quilting results


Original

Synthesis

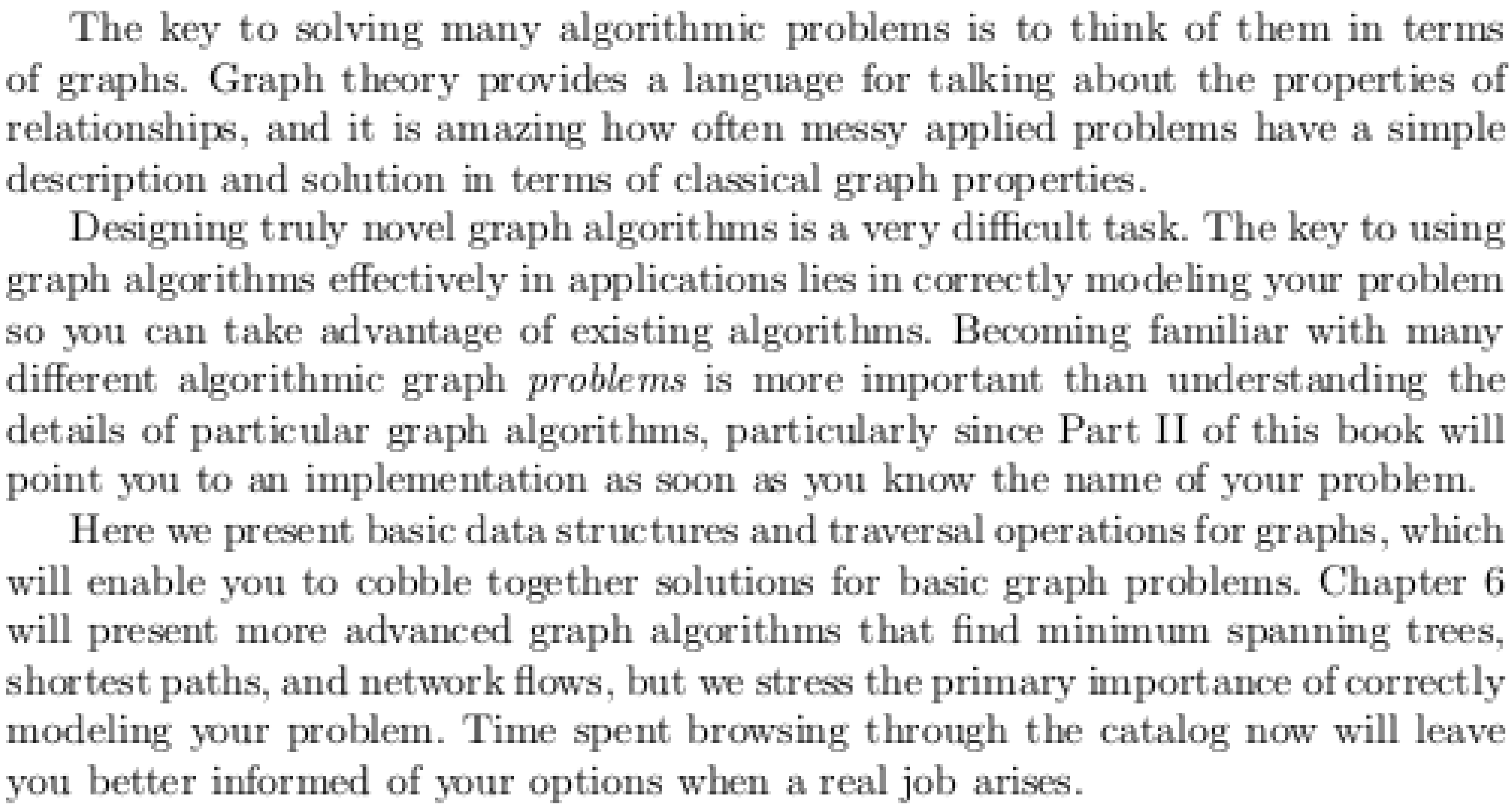

Original

Synthesis



Images: http://people.eecs.berkeley.edu/~efros/research/quilting/results.html

Image quilting results


The key to solving many algorithmic problems is to think of them in terms of graphs. Graph theory provides a language for talking about the properties of relationships, and it is amazing how often messy applied problems have a simple description and solution in terms of classical graph properties.

Designing truly novel graph algorithms is a very difficult task. The key to using graph algorithms effectively in applications lies in correctly modeling your problem so you can take advantage of existing algorithms. Becoming familiar with many different algorithmic graph *problems* is more important than understanding the details of particular graph algorithms, particularly since Part II of this book will point you to an implementation as soon as you know the name of your problem.

Here we present basic data structures and traversal operations for graphs, which will enable you to cobble together solutions for basic graph problems. Chapter 6 will present more advanced graph algorithms that find minimum spanning trees, shortest paths, and network flows, but we stress the primary importance of correctly modeling your problem. Time spent browsing through the catalog now will leave you better informed of your options when a real job arises.

S. S. Skiena. The Algorithm Design Manual.

ding s, particularly since Part II of with manyaph algorithms, particularly since Part II of this
ook won as you know the name of ycanding thenentation as soon as you know the name of your p
oblemes and traversal operations fois book wisic data structures and traversal operations for grap
s, whiutions for basic graph probler problem.bble together solutions for basic graph problems. C
hapt gorithms that find minimum saphs, whivanced graph algorithms that find minimum spann
ur pran take advantage of existing algorithms. Becoming familiar with mams for basic graph p
with algorithmic graph *problems* is more important than understanding thms that find minim
andiof particular graph algorithms, particularly since Part II of this book wiress the primary imp
s boot to an implementation as soon as you know the name of your problem. ing through the ca
probwe present basic data structures and traversal operations for graphs, whiof this bomiliar with
aphs ble you to cobble together solutions for basic graph problems. Chapter your prolnderstandi
Chapent more advanced graph algorithms that find minimum spanning treer graphs, of this bo
lving aths, and network flows, but we stress the primary important tof correctlems. Chapyour pro
theory provides ions lies ins lies in correctly mophs, whichrly since Part II of spanning for graphs
it is amazing lg algorithalgorithms. Becoming Chapter know the name of yance of coplems. Cha
lution in terms is more imore important thaning treeaversal operations for , now will spanning
novel graph algorithms is a very difficult task. The key r basic graph problems. Chapteble toge
effectively in applications lies in correctly modeling your that find minimum spanning tivanced gr
dvantage of existing algorithms. Becoming familiar wi the primary importance of correctwork flo
nic graph *problems* is more important than understan through the catalog now will lem. Time
ar graph algorithms, particularly since Part II of this bs.
plementation as soon as you know the name of your prask. The key to usingey to usiney to usin
t basic data structures and traversal operations for graphsodeling your problemur problemr probler
cobble together solutions for basic graph problems. Clg familiar with mar with manyith man
advanced graph algorithms that find minimum spannin understanding thtanding thanding th
d network flows, but we stress the primary importance of crt II of this book wils book wil book wi

Image quilting results

Image: https://commons.wikimedia.org/wiki/File:Franklin%27s_Gull_Flock_(30055312510).jpg

Image inpainting

- Similar idea to fill in missing regions of an image:
  - Find a similar patch in *another* image
  - Paste in patch with an error-minimizing cut

Hays & Efros (2007)

Parametric texture synthesis

- Alternative to stitching approaches: represent texture with a number of parameters
- To synthesize texture, coerce a noise image to match the required parameters (usually through gradient descent)
- What parameters are needed to define a texture?

Fourier magnitude?

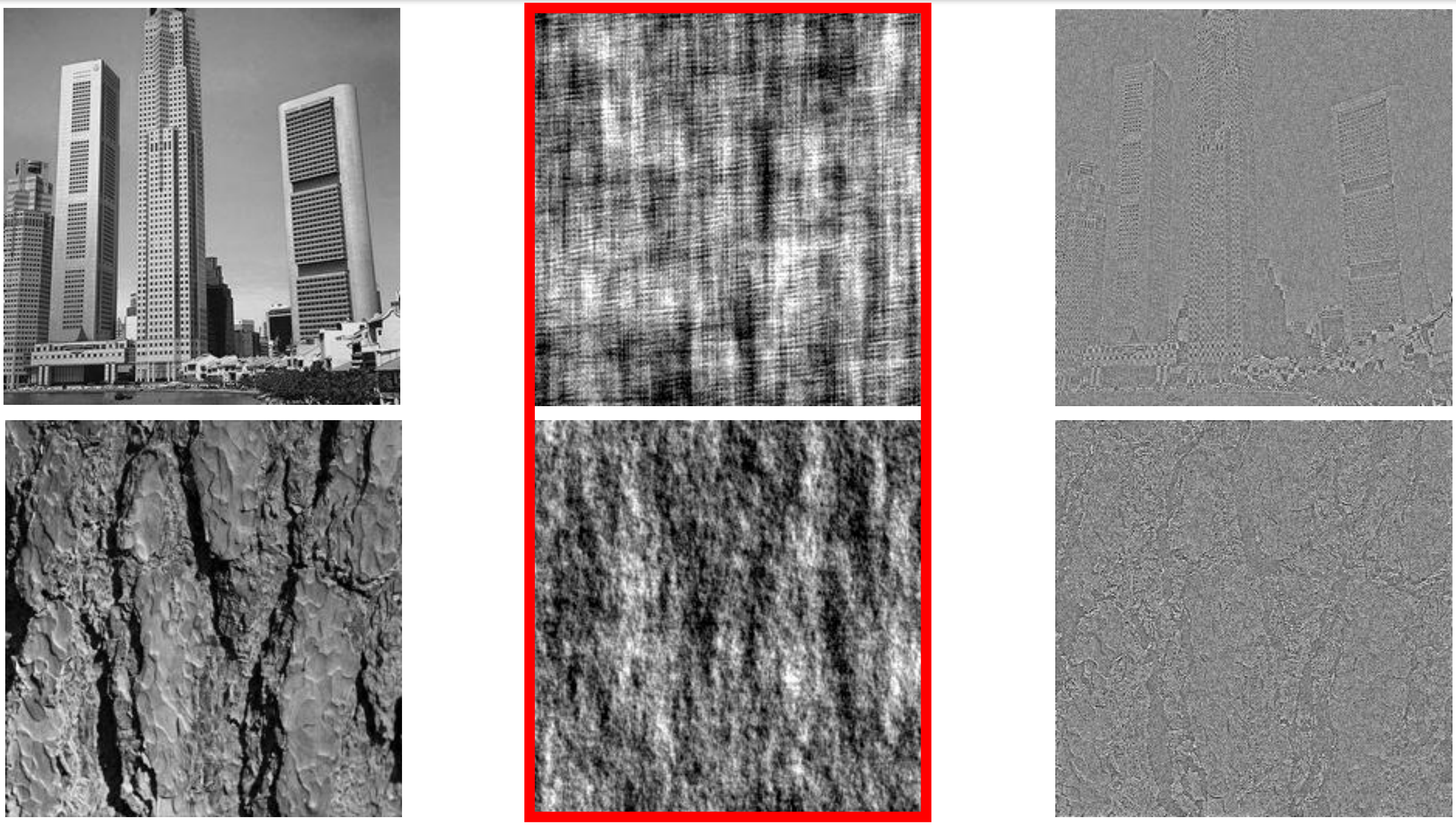
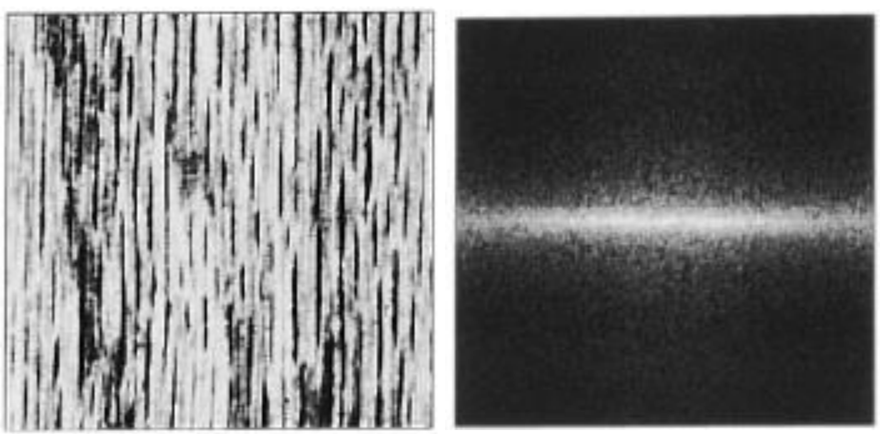


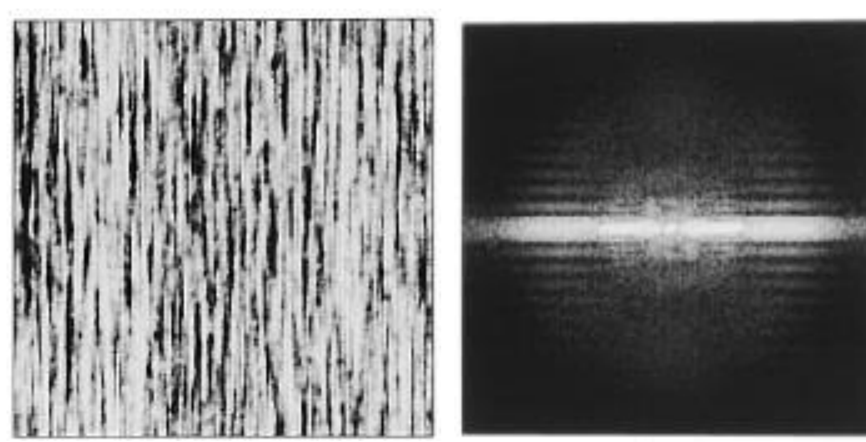
Image | Magnitude + random phase | Phase + random magnitude

Fourier texture synthesis

- Synthesize texture by matching Fourier magnitude
- Okay results for some simple textures, but doesn’t work well in general



Original image and power spectrum

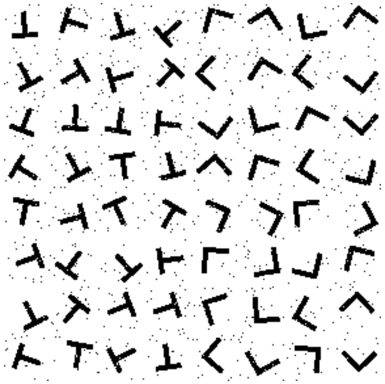


Synthesized image and power spectrum

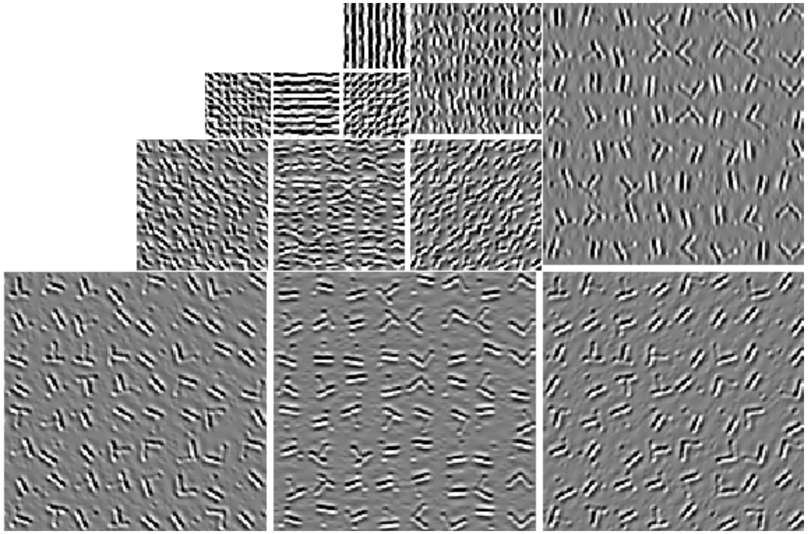
Images: Portilla, Navarro, Nestares, & Tabernero (1996)

Colour and edges?

- Textures could be defined as a distribution over simple features, like colour and edge orientation at various scales
- Synthesize texture by matching the distribution

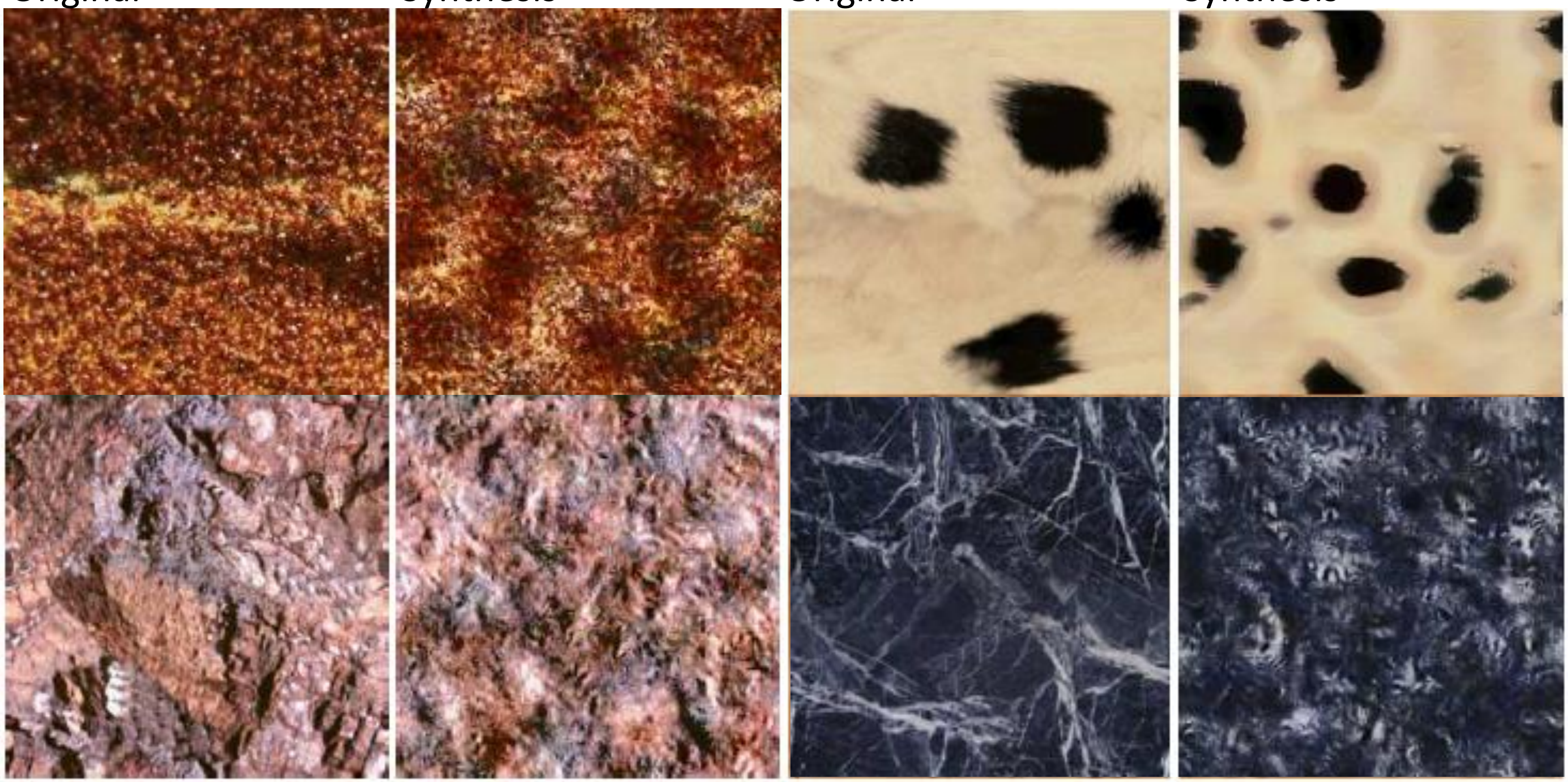


Image



Edge detector response at various orientations/scales

Distribution-matching results

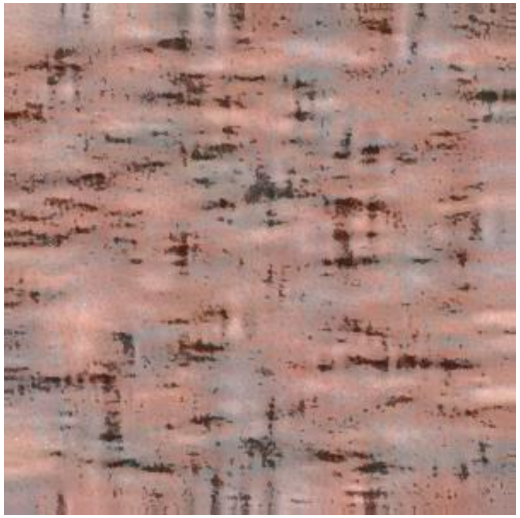


Images: Heeger & Bergen (1995)

More complex statistics?

- Simple distributions of features are not sufficient
- Also need to represent feature co-occurrence

Image

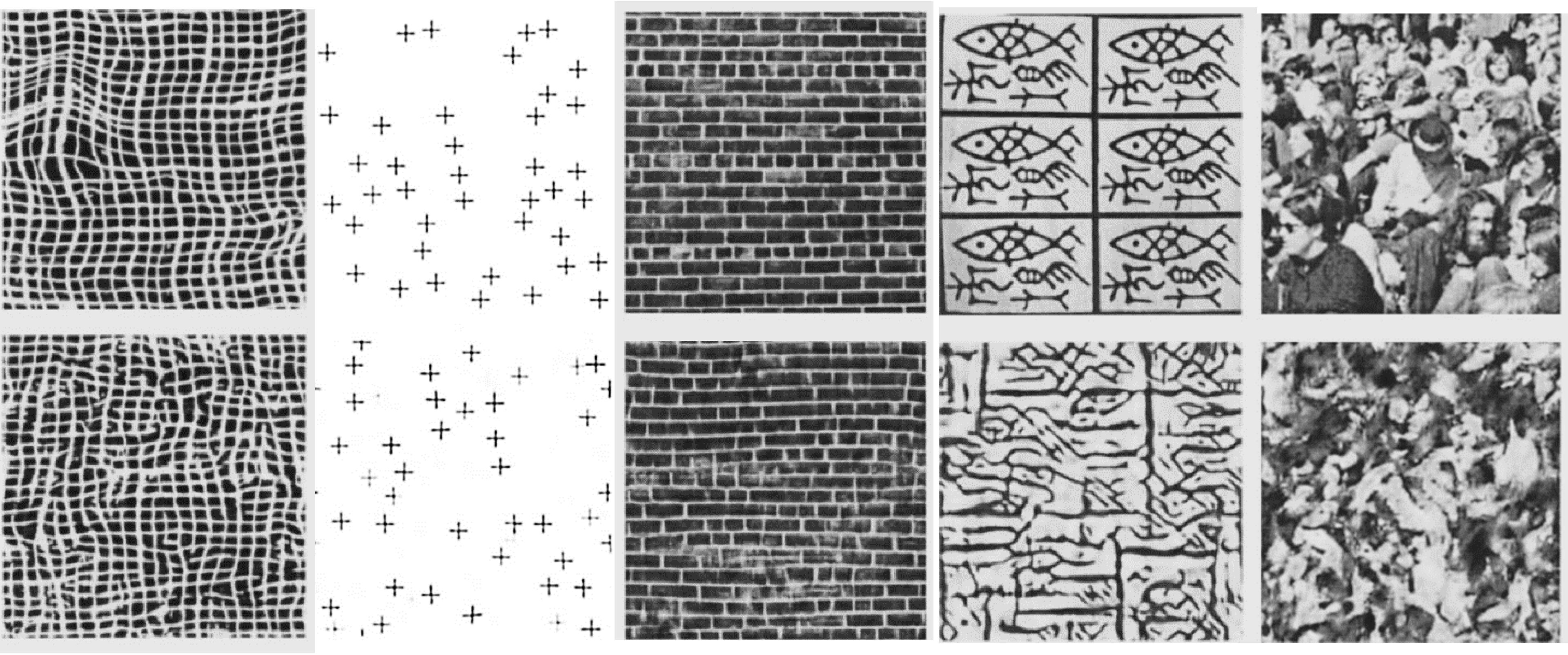


Matching feature distributions
(Heeger & Bergen, 1996)

Matching feature distributions
and correlations
(Portilla & Simoncelli, 2000)

Texture synthesis results

Original image



Synthesis

Images: Portilla & Simoncelli (2000)

Even more complex statistics?

- The set of statistics needed to represent real images may be very complex
- Instead of modelling statistics by hand, represent texture as the feature response in the layers of a neural network trained on ImageNet classification

Image: Gatys, Ecker, & Bethge (2015)

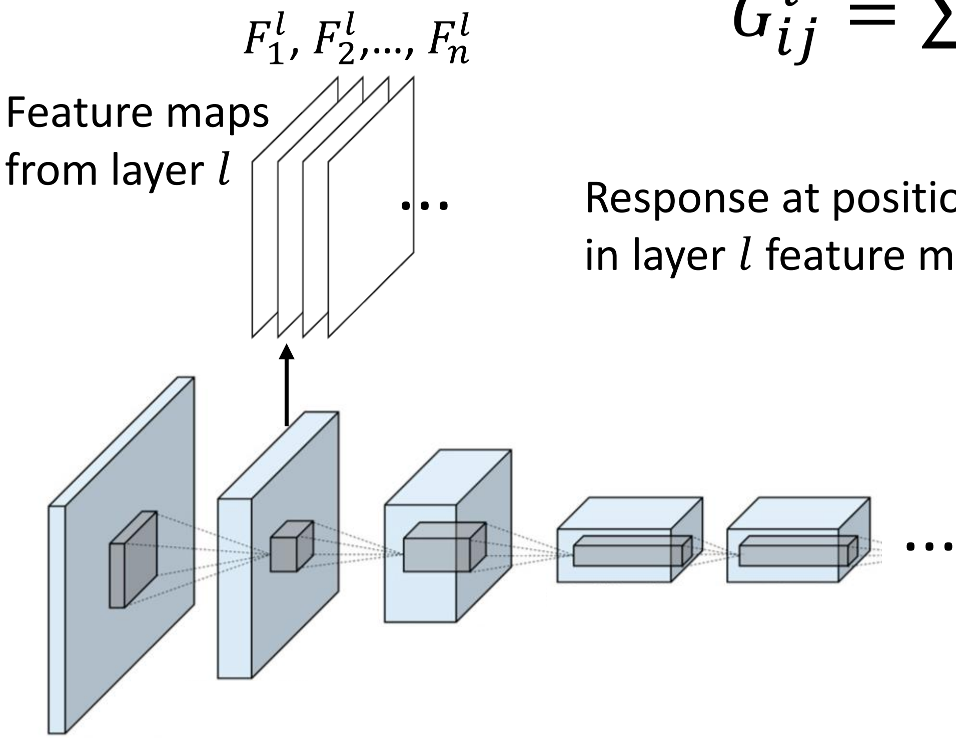
Feature correlations

- Texture is represented as the correlations between feature maps at a layer of the neural network:

$$G_{ij}^{l} = \sum_k F_{ik}^{l} F_{jk}^{l}$$

Response at position k in layer l feature map i

Response at position k in layer l feature map j



Multiple layers are included in the texture representation

Image: Han, Zhong, Cao, & Zhang (2017)

Texture synthesis results

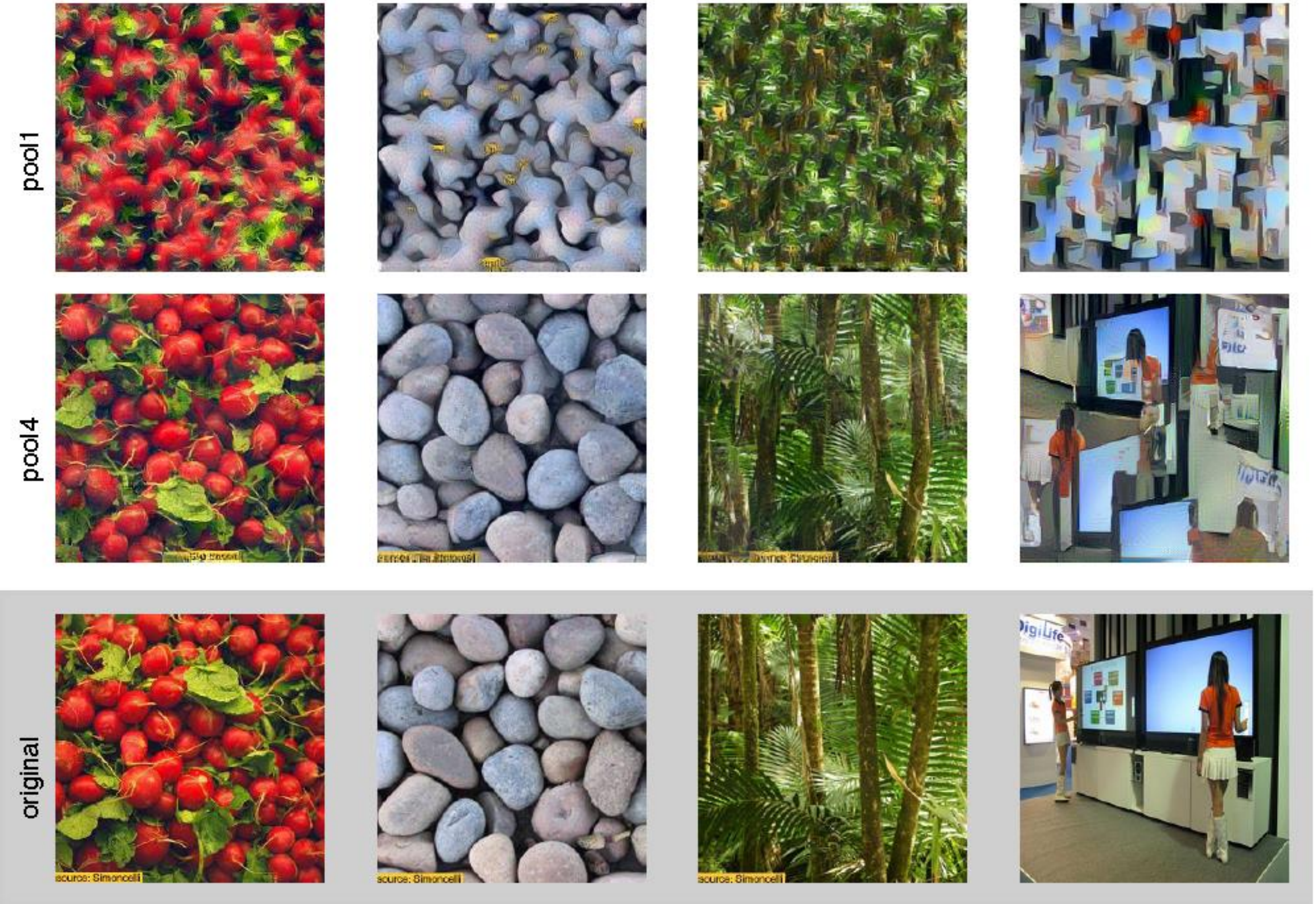
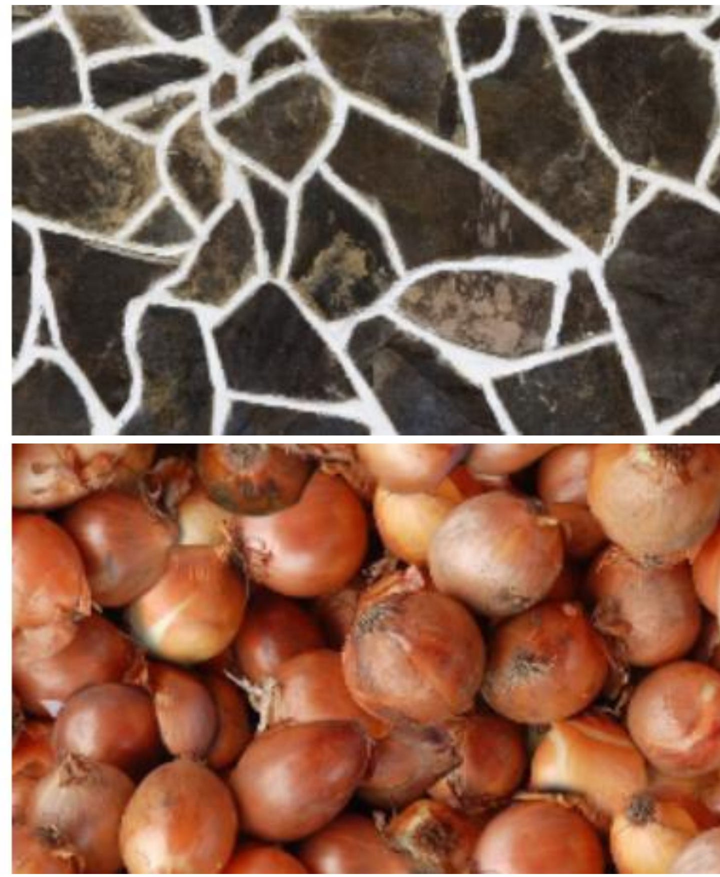


Image: Gatys, Ecker, & Bethge (2015)

Texture synthesis results

Synthesis

Original image



Images: http://bethgelab.org/deeptextures/

Summary

- Non-parametric texture synthesis is based on copying texture patches
  - Works very well on periodic textures
  - Disadvantage: No model of texture parameters
- Parametric texture synthesis represents textures in terms of a set of parameters
  - Most methods work better on stochastic textures
  - Disadvantage: Even very complex models (e.g., based on neural networks) may be incomplete

Texture transfer

Texture transfer

- Render an image in the style of another image:

Content image

+

Style image

Image: Gatys, Ecker, & Bethge (2016)

Neural style transfer algorithm

- Both images (content, style) are run through a VGG network trained on ImageNet
- Content is represented as the responses from a layer of the neural network
- Style is represented as the correlations between feature maps at a layer of the neural network
- Use gradient descent to find an image that matches both content and style

Neural style transfer

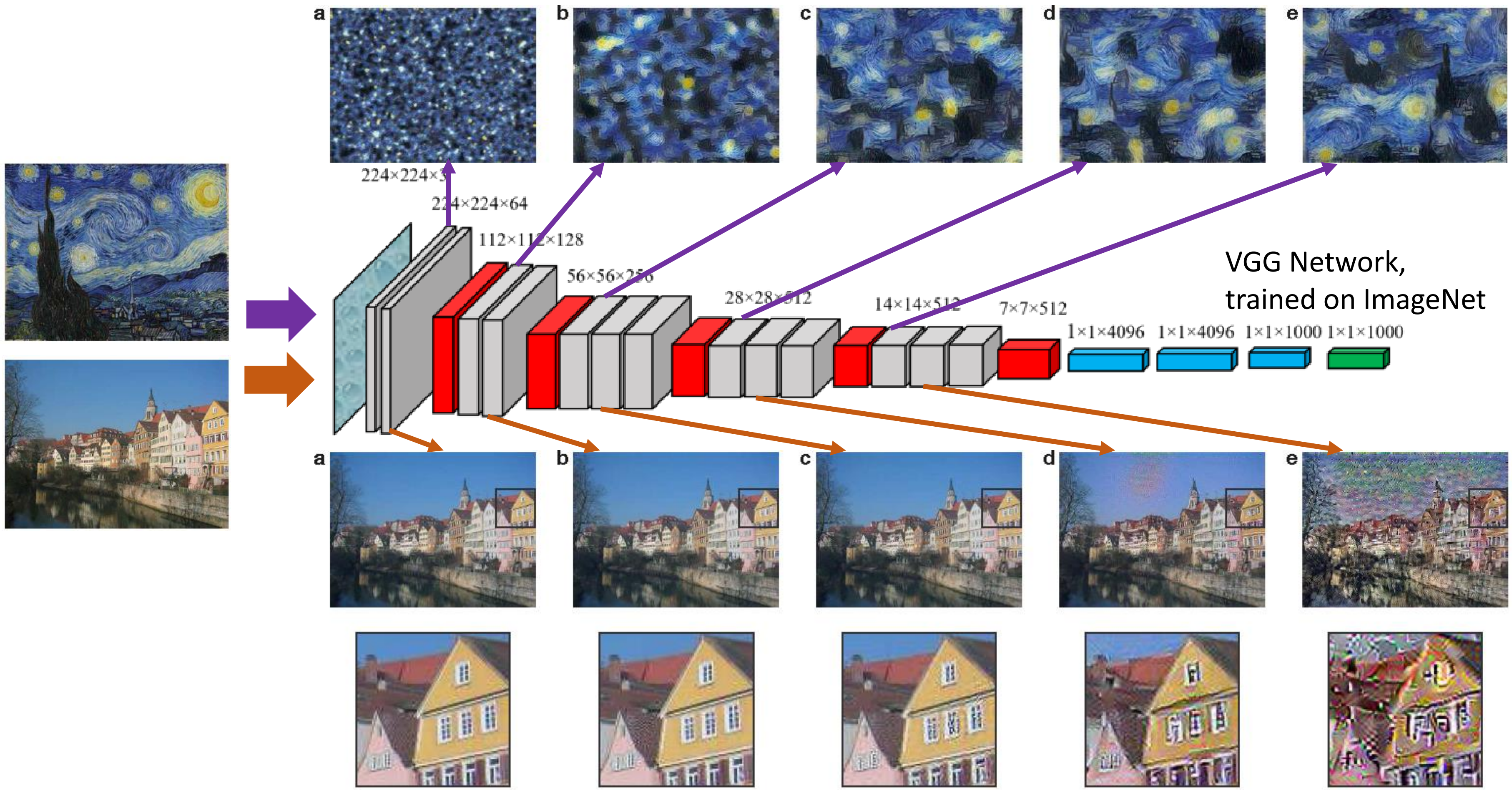


Image: Gatys, Ecker, & Bethge (2016)

Style transfer parameters

- Loss is sum of loss from content reconstruction and style reconstruction
- Relative weight of content vs. style is a free parameter:

Style image

More weight on style More weight on content

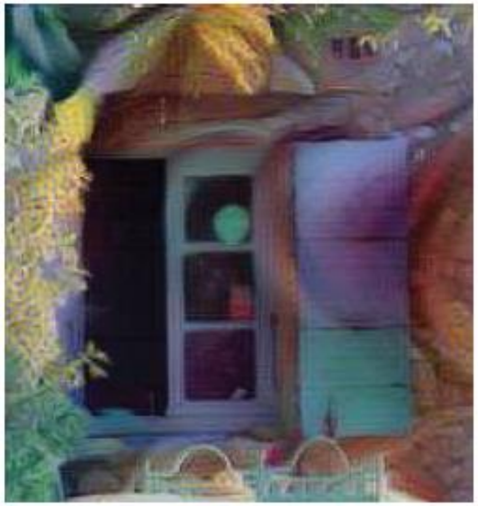
Image: Gatys, Ecker, & Bethge (2016)

Style transfer parameters

- Content and style can be matched at any combination of layers
- Generally, match content at higher layers, and style across all layers

Style image

Original



Matched at layer conv2_2

Matched at layer conv4_2

Image: Gatys, Ecker, & Bethge (2016)

Texture transfer results

Image: Gatys, Ecker, & Bethge (2016)

Summary

- Texture transfer – render an image in the style of another image
- Image content represented by neural network responses
- Texture represented by correlations of feature maps across multiple layers of a neural network

Summary

- Texture can be defined in different ways, but generally captures 2D/surface aspects of an image
- Texture representations are useful for texture synthesis and texture transfer
- Applications:
    - Image inpainting
    - Computer graphics
    - Art