Lecture 5 Multi-Scale Pyramids



6.8300/6.8301 Advances in Computer Vision Vincent Sitzmann, Mina Konaković Luković, Bill Freeman

Spring 2023





We need translation invariance









We need translation and scale invariance











Consider all patch location and sizes

Multi-Scale Pyramids

Template

Multi-Scale Pyramids

Multiscale image pyramid

Template

A multiscale image pyramid provides an alternative image representation to achieve translation and scale invariance

Gaussian Pyramid

Subsampling and aliasing

103×128

1/2

52×64

26×32

The Gaussian pyramid For each level 1. Blur input image with a Gaussian filter

$\bigcirc [1, 4, 6, 4, 1] \bigcirc 6 \rightarrow$

(The Gaussian filter is approximated with a binomial filter)

[1, 4, 6, 4, 1]

For each level1. Blur input image with a Gaussian filter2. Downsample image

$\bigcirc [1, 4, 6, 4, 1] \bigcirc 6 \rightarrow$

11

512×512

(original image)

256×256 128×128 64×64 32×32

Convolution as matrix multiplication

In the 1D case, it helps to make explicit the structure of the matrix:

0	0	0	90	90	90	90	90	0	0	ο	1/3
---	---	---	----	----	----	----	----	---	---	---	-----

In the 1D case, it helps to make explicit the structure of the matrix:

Reminder from Lecture 3

(The arrays shown here are for 1D signals)

For each level1. Blur input image with a Gaussian filter2. Downsample image

What about the opposite of blurring?

level k+1 and Gaussian pyramid level k.

Compute the difference between upsampled Gaussian pyramid

Gaussian pyramid

Gaussian pyramid

Laplacian pyramid

Blurring and downsampling:

Upsampling and blurring:

 $F_0 =$

 l_k

 g_k

 $G_k \mapsto g_{k+1}$

 F_k

Upsampling

64x64

Upsampling

1	0	1	0	1		
0	0	0	0	0		
1	0	1	0	1	0.25 0.5 0.25	C
0	0	0	0	0		
1	0	1	0	1		

0.5 0.25 1 0.5 0.5 0.25 **= ?**

Upsampling

1	0	1	0	1		
0	0	0	0	0		
1	0	1	0	1	0.25 0.5 0.25	C
0	0	0	0	0		
1	0	1	0	1		

Blurring and downsan

Upsampling and blur

mp	ling	:						6	4	1	0	0	0	0
0	0	0	0	0	0	0]		4	0	4	1	1	0	0
0	1	0	0	0	0	0	1		1	4	6	4	1	0
0	0	0	1	0	0	0	$\frac{16}{16}$	0	0	1	4	6	4	1
0	0	0	$\hat{0}$	0	1	0	10	0	0	0	1	4	6	4
Ŭ	ʹϽϙ៴	vnsa	molin	a by	2)	°_		0	0	0	0	1	4	6
		VIICa		909	-)			0	0	0	0	0	1	4
rin	g:										()	olur)		
2	4	1	0	0	0	0	0	[1	0	0	0	Γ		
(6	4	1	0	0	0	0	0	0	0	0			
2	4	6	4	1	0	0	0	0	1	0	0			
	1	4	6	4	1	0	0	0	0	0	0			
(0	1	4	6	4	1	0	0	0	1	0			
()	0	1	4	6	4	1	0	0	0	0			
()	0	0	1	4	6	4	0	0	0	1			
(0	0	0	0	1	4	6	0	0	0	0			
			(bl	ur)				L (Up	samp	oling	by 2			
			-		-									
			$ l_0 =$	= (I	0 —	F_0	$G_0)$	g_0						
								_						

$l_0 = (I_0 - F_0 G_0)g_0$

	182	-56	-24	-8	-2	0	0
	-56	192	-56	-32	-8	0	0
	-24	-56	180	-56	-24	-8	-2
_ 1	-8	-32	-56	192	-56	-32	-8
-256	-2	-8	-24	-56	180	-56	-24
	0	0	-8	-32	-56	192	-56
	0	0	-2	-8	-24	-56	182
	0	0	0	0	-8	-32	-48

and the second se

Laplacian pyramid

Gaussian residual

Can we invert the Laplacian Pyramid?

The Laplacian Pyramid n. F_0 G_2 F_2 ' F_1 G_1 g_2 g_3 F_2 $l\mathfrak{I}$

Analysis/Encoder

Synthesis/Decoder

Laplacian pyramid

Laplacian pyramid

Gaussian residual

Gaussian residual

Laplacian pyramid applications

- Image compression
- Noise removal
- Computing image features (e.g., SIFT)

Image Blending

Image Blending



IA

JB

Image Blending



$I = m * I^A + (1 - m) * I^B$





Image Blending with the Laplacian Pyramid











 $l_k = l_k^A * m_k + l_i^B * (1 - m_k)$







Image Blending with the Laplacian Pyramid







Image Blending with the Laplacian Pyramid

- Build Laplacian pyramid for both images: LA, LB
- Build Gaussian pyramid for mask: G
- Build a combined Laplacian pyramid:
- Collapse L to obtain the blended image

532

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, AF

The Laplacian Pyramid as a Compact Image Code

PETER J. BURT, MEMBER, IEEE, AND EDWARD H. ADELSON





http://persci.mit.edu/pub_pdfs/pyramid83.pdf











Gaussian Pyr

Image pyramids





Laplacian Pyr And many more: QMF, steerable, ... Convnets!

Orientations















Orientations























Low pass residual











 $\blacktriangleright b_{0,0}$

 $\blacktriangleright b_{0,1}$

 $\blacktriangleright b_{0,n}$

 $\blacktriangleright b_{1,0}$

 $\blacktriangleright b_{1,1}$













3 scales, 4 orientations



Analysis/Encoder

Synthesis/Decoder



Visual summary: Linear Image Transforms



b)







Steerable pyramid applications

- Texture recognition
- Image compression
- Noise removal
- Computing image features (e.g., SIFT)
- Object recognition





An astronaut riding a horse in a photorealistic style

Dall-E 2



https://openai.com/dall-e-2/



Making textures







Stochastic



Textures

















REVIEW ARTICLES

Textons, the elements of texture perception, and their interactions

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

Pre-attentive texture discrimination



Pre-attentive texture discrimination

- **K Y Y A A A A Y Y D A A A B B B B B B B B** ARABRERARA AAKKADKRJKA AKAAKARAKA ARGREDDAGESAR AKKKKKAAAA **KAKKKKKKA** 6 9 6 6 A A 6 9 9 F **K & D A K & D K & A A** A A D D D & & A K & A K

Pre-attentive texture discrimination

9 Ŀ 8 ¢, ጽ 9

This texture pair is pre-attentively indistinguishable. Why?





Julesz - Textons Textons: fundamental texture elements.

such as terminators, corners, and intersections within the patterns...



Textons might be represented by features



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,



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

Observation: the Xs look smaller than the Ls.

Early vision and texture perception

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

Ls 25% larger contrast adjusted to keep mean constant







Ls 25% shorter

> と フ コ ド マ レ マ イ ド レ ビ イ イ コトレンシック マイレント インレレンコレレアレイハフス グレンイ シング ビッフイグレア ^^~ > 7 + + + + + + × - 1 ^ 1 5 77 > + × × × × × + < < > ~> ~ ~ + + + + + + + 7 ~ (r < 1 1 + x + x + * + n < r 7L < L × × + + × + × r r r ^ 77 F + + + + + × × ~ ~ < トレマレメ メ チメ チメキレンレ コレイ マフフヘ コフイ ヘト フシ > < ¬ ^ 」 7 7 7 ~ ~ ~ ~ ~ ^ / () JL J ((V) ^ (V J 7

* ** * * * * * * * * * * * ANT TANA ANT AT AT

Pyramid-Based Texture Analysis/Synthesis

David J. Heeger* Stanford University

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

Abstract

This paper describes a method for synthesizing images that match the texture appearance of a given digitized sample. This synthesis is completely automatic and requires only the "target" texture as input. It allows generation of as much texture as desired so that any object can be covered. It can be used to produce solid textures for creating textured 3-d objects without the distortions inherent in texture mapping. It can also be used to synthesize texture mixtures, images that look a bit like each of several digitized samples. The approach is based on a model of human texture perception, and has potential to be a practically useful tool for graphics applications.

Introduction

Computer renderings of objects with surface texture are more interesting and realistic than those without texture. Texture mapping [15] is a technique for adding the appearance of surface detail by wrapping or projecting a digitized texture image onto a surface. Digitized textures can be obtained from a variety of sources, e.g., cropped from a photoCD image, but the resulting texture chip may not have the desired size or shape. To cover a large object you may need to repeat the texture; this can lead to unacceptable artifacts either in the form of visible seams, visible repetition, or both.

Texture mapping suffers from an additional fundamental problem: often there is no natural map from the (planar) texture image to the geometry/topology of the surface, so the texture may be distorted unnaturally when mapped. There are some partial solutions to this distortion problem [15] but there is no universal solution for mapping an image onto an arbitrarily shaped surface.

An alternative to texture mapping is to create (paint) textures by hand directly onto the 3-d surface model [14], but this process is both very labor intensive and requires considerable artistic skill.

Another alternative is to use computer-synthesized textures so that as much texture can be generated as needed. Furthermore, some of the synthesis techniques produce textures that tile seamlessly.

Using synthetic textures, the distortion problem has been solved in two different ways. First, some techniques work by synthesizing texture directly on the object surface (e.g., [31]). The second solution is to use solid textures [19, 23, 24]. A solid texture is a 3-d array of color values. A point on the surface of an object is colored by the value of the solid texture at the corresponding 3-d point. Solid texturing can be a very natural solution to the distortion problem:

there is no distortion because there is no mapping. However, existing techniques for synthesizing solid textures can be quite cumbersome. One must learn how to tweak the parameters or procedures of the texture synthesizer to get a desired effect.

This paper presents a technique for synthesizing an image (or solid texture) that matches the appearance of a given texture sample. The key advantage of this technique is that it works entirely from the example texture, requiring no additional information or adjustment. The technique starts with a digitized image and analyzes it to compute a number of texture parameter values. Those parameter values are then used to synthesize a new image (of any size) that looks (in its color and texture properties) like the original. The analysis phase is inherently two-dimensional since the input digitized images are 2-d. The synthesis phase, however, may be either two- or threedimensional. For the 3-d case, the output is a solid texture such that planar slices through the solid look like the original scanned image. In either case, the (2-d or 3-d) texture is synthesized so that it tiles seamlessly.

2 Texture Models

Textures have often been classified into two categories, deterministic textures and stochastic textures. A deterministic texture is characterized by a set of primitives and a placement rule (e.g., a tile floor). A stochastic texture, on the other hand, does not have easily identifiable primitives (e.g., granite, bark, sand). Many real-world textures have some mixture of these two characteristics (e.g. woven fabric, woodgrain, plowed fields).

Much of the previous work on texture analysis and synthesis can be classified according to what type of texture model was used. Some of the successful texture models include reactiondiffusion [31, 34], frequency domain [17], fractal [9, 18], and statistical/random field [1, 6, 8, 10, 12, 13, 21, 26] models. Some (e.g., [10]) have used hybrid models that include a deterministic (or periodic) component and a stochastic component. In spite of all this work, scanned images and hand-drawn textures are still the principle source of texture maps in computer graphics.

This paper focuses on the synthesis of stochastic textures. Our approach is motivated by research on human texture perception. Current theories of texture discrimination are based on the fact that two textures are often difficult to discriminate when they produce a similar distribution of responses in a bank of (orientation and spatial-frequency selective) linear filters [2, 3, 7, 16, 20, 32]. The method described here, therefore, synthesizes textures by matching distributions (or histograms) of filter outputs. This approach depends on the principle (not entirely correct as we shall see) that all of the spatial information characterizing a texture image can be captured in the first order statistics of an appropriately chosen set of linear filter outputs. Nevertheless, this model (though incomplete) captures an interesting set of texture properties.



Figure 5: (Top Row) Original digitized sample textures: red granite, berry bush, figured maple, yellow coral. (Bottom Rows) Synthetic solid textured teapots.

https://www.cns.nyu.edu/heegerlab/content/publications/Heeger-siggraph95.pdf

SIGGRAPH 1995

^{*}Department of Psychology, Stanford University, Stanford, CA 94305. heeger@white.stanford.edu http://white.stanford.edu

[†]SRI David Sarnoff Research Center, Princeton, NJ 08544. jrb@sarnoff.com



Can we use this representation to generate more samples from the same texture?

Steerable Pyramid $- b_{0,0}$ B_0 $\blacktriangleright b_{0,1}$ $\blacktriangleright B_1$ $\blacktriangleright B_n \blacktriangleright$ $\blacktriangleright b_{0,n}$ $\rightarrow b_{1,0}$ \blacktriangleright B_1 \blacktriangleright $\blacktriangleright B_n \blacktriangleright$ $\blacktriangleright b_{1,n}$ $\blacktriangleright g_2$ Synthesis/Decoder







 $\blacktriangleright b_{0,0}$

 $\blacktriangleright g_2$



Representation = Histograms of pixel values for each subband

Representation









Representation



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



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

Histogram



75% of pixels have an intensity value 9% of pixels have an intensity value smaller than 0.5 within the range[0.37, 0.41]



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







Cumulative distribution



Z[n,m]







Y[n,m]





We look for a transformation of the image Y

Y' = f(Y)

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

There are infinitely many functions that can do this transformation.

A natural choice is to use **f** being:

- point-wise non linearity
- stationary
- monotonic (most of the time invertible)

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)













Another example: Matching histograms







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





In this example, f is a step function.

Matching histograms of a subband













Matching histograms of a subband





Texture analysis





Steerable pyramid

(histogram)



Heeger and Bergen, 1995

Texture synthesis



(histogram)

Why does it work? (sort of)







Why does it work? (sort of)



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

Why does it work? (sort of)

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





slag stone, figured yew wood.

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, 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?



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



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



How to identify the set of equivalent textures?



This does not reveal how poor the representation actually is.



How to identify the set of equivalent textures?



These trajectories are more perceptually salient







This set is huge







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...

