Textures and Inpainting

Digital Visual Effects

Yung-Yu Chuang

*with slides by Alex Efros, Li-Yi Wei, Arno Schedl and Paul Debevec*
Outline

• Texture synthesis
• Acceleration by multi-resolution and TSVQ
• Patch-based texture synthesis
• Image analogies
Texture synthesis
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"
The challenge

- How to capture the essence of texture?
- Need to model the whole spectrum: from repeated to stochastic texture
Markov property

• \( P(f_i | f_{i-1}, f_{i-2}, f_{i-3}, \ldots f_0) = P(f_i | f_{i-1}, f_{i-2}, \ldots f_{i-n}) \)

• \( P(f_i | f_{S-\{i\}}) = P(f_i | f_{N_i}) \)
Motivation from language

- [Shannon’48] proposed a way to generate English-looking text using N-grams:
  - Assume a generalized Markov model
  - Use a large text to compute probability distributions of each letter given N-1 previous letters
    - precompute or sample randomly
  - Starting from a seed repeatedly sample this Markov chain to generate new letters
  - One can use whole words instead of letters too.
• Results (using alt.singles corpus):
  - "One morning I shot an elephant in my arms and kissed him."
  - "I spent an interesting evening recently with a grain of salt"

• Notice how well local structure is preserved!
  - Now let’s try this for video and in 2D...
Video textures

Still photos
Video clips
Video textures
Problem statement

video clip

video texture
Approach

How do we find good transitions?
Finding good transitions

Compute $L_2$ distance $D_{i,j}$ between all frames

Similar frames make good transitions
Video textures
Ideally

- Assuming Markov property, what is conditional probability distribution of p, given the neighbourhood window?
- Instead of constructing a model, let’s directly search the input image for all such neighbourhoods to produce a histogram for p
- To synthesize p, just pick one match at random
In reality

- However, since our sample image is finite, an exact neighbourhood match might not be present.
- So we find the best match using SSD error (weighted by a Gaussian to emphasize local structure), and take all samples within some distance from that match.
- Using Gaussian-weighted SSD is very important.
Neighborhood size matters
More results

Increasing window size
More results

french canvas

rafia weave
More results

wood

brick wall
Failure cases

Growing garbage

Verbatim copying
Summary of the basic algorithm

- Exhaustively search neighborhoods
Neighborhood

- Neighborhood size determines the quality & cost

- $3 \times 3$  
  - 423 s

- $5 \times 5$  
  - 528 s

- $7 \times 7$  
  - 739 s

- $9 \times 9$  
  - 1020 s

- $11 \times 11$  
  - 1445 s

- $41 \times 41$  
  - 24350 s
Summary

- Advantages:
  - conceptually simple
  - models a wide range of real-world textures
  - naturally does hole-filling

- Disadvantages:
  - it’s slow
  - it’s a heuristic
Acceleration by Wei & Levoy

- Multi-resolution
- Tree-structure
Multi-resolution pyramid

High resolution

Low resolution
Multi-resolution algorithm
Benefits

- Better image quality & faster computation (by using smaller windows)
Results

Random

Oriented

Regular

Semi-regular
Failures

- Non-planar structures
- Global information
Acceleration

- Computation bottleneck: neighborhood search
Nearest point search

- Treat neighborhoods as high dimensional points

Neighborhood

High dimensional point/vector
Tree-Structured Vector Quantization
Timing

- Time complexity: $O(\log N)$ instead of $O(N)$

Efros 99: 1941 secs
Full searching: 503 secs
TSVQ: 12 secs
Results

Input  Exhaustive: 360 s  TSVQ: 7.5 s
Patch-based methods

**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
Algorithm

- Pick size of block and size of overlap
- Synthesize blocks in raster order

- Search input texture for block that satisfies overlap constraints (above and left)
- Paste new block into resulting texture
  - blending
  - use dynamic programming to compute minimal error boundary cut
Input texture

Random placement of blocks

Neighboring blocks constrained by overlap

Minimal error boundary cut
Minimal error boundary

overlapping blocks = vertical boundary

overlap error = min. error boundary
Results
Results
Failure cases
Texture transfer

- Take the texture from one object and “paint” it onto another object

Then, just add another constraint when sampling: similarity to underlying image at that spot
GraphCut textures

Overlap

Patch A

Patch B

1 → 4

7

cut

2

5

6

8

9

Patch A

Patch B

Input

Image Quilting

Graph cut
GraphCut textures
GraphCut textures

Graphcut Textures: Image and Video Synthesis Using Graph Cuts

Vivek Kwatra
Arno Schödl
Irfan Essa
Greg Turk
Aaron Bobick

GVU Center / College of Computing
Georgia Institute of Technology
http://www.cc.gatech.edu/cpl/projects/graphcuttextures
Inpainting

- Growing is in “onion peeling” order
  - within each “layer”, pixels with most neighbors are synthesized first
Image extrapolation
Inpainting

\[ P(p) = C(p)D(p) \]

\[ C(p) = \frac{\sum_{q \in \psi_p \cap (I-\Omega)} C(q)}{|\psi_p|} , \quad D(p) = \frac{\nabla I_p^\perp \cdot n_p}{\alpha} \]
Results
Results
Results

http://research.microsoft.com/vision/cambridge/i3l/patchworks.htm
Structure propagation
Structure propagation

Image Completion with Structure Propagation

Jian Sun  Lu Yuan
Jiaya Jia  Heung-Yeung Shum

SIGGRAPH 2005
Image Analogies

A : A' :: B :: ?
Image Analogies Implementation
Image Analogies Implementation

unfiltered target image

unfiltered training image

filtered training image
Image Analogies Implementation

unfiltered target image

unfiltered training image

filtered target image

filtered training image

A

A'
Balance between approximate and coherence searches

\[ \text{function} \; \text{BestMatch}(A, A', B, B', s, \ell, q): \]
\[ p_{\text{app}} \leftarrow \text{BestApproximateMatch}(A, A', B, B', \ell, q) \]
\[ p_{\text{coh}} \leftarrow \text{BestCoherenceMatch}(A, A', B, B', s, \ell, q) \]
\[ d_{\text{app}} \leftarrow \|F_\ell(p_{\text{app}}) - F_\ell(q)\|^2 \]
\[ d_{\text{coh}} \leftarrow \|F_\ell(p_{\text{coh}}) - F_\ell(q)\|^2 \]
\[ \text{if } d_{\text{coh}} \leq d_{\text{app}}(1 + 2^{\ell - L} \kappa) \text{ then} \]
\[ \text{return } p_{\text{coh}} \]
\[ \text{else} \]
\[ \text{return } p_{\text{app}} \]
Coherence search

input image

completed portion (grey)

output image
Learn to blur

Unfiltered source ($A$)  Filtered source ($A'$)

Unfiltered target ($B$)  Filtered target ($B'$)
Super-resolution
Colorization

Unfiltered source (A) and Filtered source (A')

Unfiltered target (B) and Filtered target (B')
Artistic filters

Unfiltered source (A)  Filtered source (A')
Texture by numbers

Unfiltered source ($A$)

Filtered source ($A'$)

Unfiltered ($B$)

Filtered ($B'$)
Texture by numbers

Image Analogies

Aaron Hertzmann
Charles Jacobs
Nuria Oliver
Brian Curless
David Salesin
The end!