Today

- Texture synthesis
- Image analogies
- Video textures

Texture

- What is texture?
  - Easy to recognize, hard to define
  - Deterministic/regular textures (“thing-like”)
  - Stochastic textures (“stuff-like”)

- Tasks
  - Discrimination / Segmentation
  - Classification
  - Texture synthesis
  - Shape from texture
  - Texture transfer
  - Video textures
Modeling Texture

• What is texture?
  – An image obeying some statistical properties
  – Similar structures ("textons") repeated over and over again according to some placement rule
  – Must separate what repeats and what stays the same
  – Often has some degree of randomness
  – Often modeled as repeated trials of a random process

Texture Synthesis

• **Goal of Texture Synthesis**: create new samples of a given texture
• Many applications: virtual environments, hole-filling, texturing surfaces

The Challenge

• Need to model the whole spectrum: from repeated to stochastic texture

Method 1: Copy Block(s) of Pixels

• Visual artifacts at block boundaries!
  – Photo
  – Pattern repeated
One idea: Build Probability Distributions

Basic idea
1. Compute statistics of input texture (e.g., histogram of edge filter responses)
2. Generate a new texture that keeps those same statistics


Statistical Modeling of Texture
- Assume stochastic model of texture (Markov Random Field)
- Stationarity: the stochastic model is the same regardless of position

Markov Chain
- Markov Chain
  - a sequence of random variables $x_1, x_2, \ldots, x_n$
  - $x_t$ is the state of the model at time $t$
  \[
  x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow x_5
  \]
- Markov assumption: Each state is dependent only on the previous one
  - dependency given by a conditional probability:
  \[
p(x_t|x_{t-1})
  \]
  - The above is actually a first-order Markov chain
  - An $n^{th}$-order Markov chain: $p(x_t|x_{t-1}, \ldots, x_{t-N})$
**Markov Random Field**

A Markov Random Field (MRF)

- Generalization of Markov chains to two or more dimensions

**First-order MRF:**

- Probability that pixel $X$ takes a certain value given the values of neighbors $A$, $B$, $C$, and $D$:

  $P(X|A, B, C, D)$

- Higher-order MRFs have larger neighborhoods

**Motivation from Language**

- Shannon (1948) proposed a way to generate English-looking text using $n$-grams
  - Assume a Markov model
  - Use a large text corpus to compute probability distributions of each letter given $N-1$ previous letters
  - Starting from a seed, repeatedly sample the conditional probabilities to generate new letters
  - Can use whole words instead of letters too

**Statistical Modeling of Texture**

- Assume stochastic model of texture (Markov Random Field)
- Stationarity: the stochastic model is the same regardless of position
- Markov (i.e., local) property:
  $p(\text{pixel} | \text{rest of image}) = p(\text{pixel} | \text{neighborhood})$

**Mark V. Shaney (Bell Labs)**

- Results (using alt.singles corpus):
  - “As I’ve commented before, really relating to someone involves standing next to impossible.”
  - “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 in 2D using pixels
Efros & Leung Algorithm

• Assuming Markov property, compute $P(p|N(p))$
  – Building explicit probability tables infeasible
  – Instead, we search the input image for all similar neighborhoods — that’s our pdf for $p$
  – To sample from this pdf, just pick one match at random

Finding matches

• Sum of squared differences (SSD)

$$\|\begin{pmatrix} \text{pixel} \\ \text{neighborhood} \end{pmatrix} \|^2$$

Details

• How to match patches?
  – Gaussian-weighted SSD (more emphasis on nearby pixels)

• What order to fill in new pixels?
  – “Onion skin” order: pixels with most neighbors are synthesized first
  – To synthesize from scratch, start with a randomly selected small patch from the source texture

• How big should the patches be?

Finding matches

• Sum of squared differences (SSD)
  – Gaussian-weighted to make sure closer neighbors are in better agreement

$$\|\begin{pmatrix} \text{pixel} \\ \text{neighborhood} \end{pmatrix} * \begin{pmatrix} \text{match} \end{pmatrix} \|_2$$
Texture synthesis algorithm

- While image not filled
  1. Get unfilled pixels with filled neighbors, sorted by number of filled neighbors
  2. For each pixel, get top $N$ matches based on visible neighbors
     - Patch Distance: Gaussian-weighted SSD
  3. Randomly select one of the matches and copy pixel from it
More Results

white bread

brick wall

Summary

• The Efros & Leung algorithm
  – Very simple
  – Surprisingly good results
  – Synthesis is easier than analysis!
  – …but very slow

• How can you improve it? Any ideas?

Image Quilting [Efros & Freeman]

• 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
  • Not the same as multi-scale!
Texture synthesis with graph cuts

Accounting for existing seams
Large-scale Structures:

- Challenges: texture synthesis often fails when dealing with large-scale or near-regular structures.
- Solution: Our method suggests using guidance channels to improve synthesis results.

Guidance channels are computed using a new initialization strategy that bootstraps the synthesis process.

**Self Tuning Texture Optimization**

- We present a new method that automatically generates textures.
- The optimization process is fully automatic and suitable for a wide variety of applications.

**Failures**

- Chernobyl Harvest:
  - Excessive repetition
  - Mismatched or distorted boundaries

**Self Tuning Texture Optimization**

- Our method extends previous work and includes several key improvements.
- It successfully synthesizes structured textures.

Our improvements result from analyzing the shortcomings of previous methods, particularly those dealing with large-scale or near-regular structures.

- We pay particular attention to making the synthesis process fully automatic and suitable for a wide variety of applications.

**Results**

- Image Quilting / Our Result
- Resynthesis / Our Result
- Graphical Textures / Our Result
- Image Melding / Our Result

**Resources**

Political Texture Synthesis!

Fill Order

Exemplar-based Inpainting demo

- In what order should we fill the pixels?
  - choose pixels that have more neighbors filled
  - choose pixels that are continuations of lines/curves/edges

Application: Texture Transfer

• Try to explain one object with bits and pieces of another object:

Texture Transfer

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

Same as texture synthesis, except an additional constraint:

1. Consistency of texture
2. Patches from texture should correspond to patches from constraint in some way. Typical example: blur luminance, use SSD for distance
Today

- Texture synthesis
- Image analogies
- Video textures

Image Analogies

Hertzmann et al. SIGGRAPH 2001
Image Analogies

- Define a similarity between A and B
- For each patch in B:
  - Find a matching patch in A, whose corresponding A' also fits in well with existing patches in B'
  - Copy the patch in A' to B'
- Algorithm is done iteratively, coarse-to-fine
The Approach

The Approach

function CREATEIMAGEANALOGY(A, A', B):
Compute Gaussian pyramids for A, A', and B
Compute features for A, A', and B
Initialize the search structures (e.g., for ANN)
for each level \( \ell \), from coarsest to finest, do:
  for each pixel \( q \in B'_\ell \), in scan-line order, do:
    \( p \leftarrow \text{BESTMATCH}(A, A', B, B', s, \ell, q) \)
    \( B'_\ell(q) \leftarrow A'_\ell(p) \)
    \( s(q) \leftarrow p \)
return \( B'_L \)

Implementation Details

• Use approximate nearest neighbor search and Ashikhmin’s coherence search heuristic
• Use feature vectors instead of pixel values
  – Feature vector can consist of RGB values plus additional “channels” such as luminance, outputs of derivative filters
• Luminance remapping to align color histograms of source and target images
Blur Filter

Unfiltered source (A)  Filtered source (A')

Unfiltered target (B)  Filtered target (B')

Edge Filter

Unfiltered source (A)  Filtered source (A')

Unfiltered target (B)  Filtered target (B')

Artistic Filters

A  A'

B  B'

Artistic Filters

A  A'

B  B'
Colorization

Unfiltered source (A)  Filtered source (A')
Unfiltered target (B)  Filtered target (B')

Texture-by-numbers

A  B
A'  B'

Super-resolution

A  A'

Super-resolution (result!)

B  B'
Super-resolution

Super-resolution result

Today

- Texture synthesis
- Image analogies
- Video textures

Still photos
Video clips

Problem statement

Video Textures Approach

- How do we find good transitions?
Finding good transitions
- Compute $L_2$ distance $D_{i,j}$ between all frames

Markov chain representation

Transition costs
- Transition from $i$ to $j$ if successor of $i$ is similar to $j$
  - Cost function: $C_{i\rightarrow j} = D_{i+1,j}$

Transition probabilities
Probability for transition $P_{i\rightarrow j}$ inversely related to cost:
$$P_{i\rightarrow j} \sim \exp\left(-\frac{C_{i\rightarrow j}}{\sigma^2}\right)$$
Transition probabilities

Preserving dynamics

Preserving dynamics

- Cost for transition $i \rightarrow j$: $C_{i,j} = \sum_{k=1}^{N} w_k D_{i,k,j,k}$
Preserving dynamics – effect

- Cost for transition \( i \to j \): \( C_{i,j} = \sum_{k=1}^{N} w_k D_{i,k,j,k} \)

Preserving dynamics

\[
D'_{i,j} = \sum_{k=\pm m} w_k D_{i+k,j+k}
\]

- Filter with diagonal kernel, weights \( w \).

Dead ends

- No good transition at the end of sequence

Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

\[
F_{i \to j} = C_{i \to j} + \alpha \min_k F_{j \to k}
\]
Future cost

• Propagate future transition costs backward
• Iteratively compute new cost

\[ F_{i \rightarrow j} = C_{i \rightarrow j} + \alpha \min_k F_{j \rightarrow k} \]
Future cost – effect

Finding good loops

- Alternative to random transitions
- Precompute set of loops up front using Dynamic Programming

Visual discontinuities

- Problem: Visible “Jumps”

Crossfading

- Solution: Crossfade from one sequence to the other.
Morphing

• Interpolation task:
  \[
  \frac{2}{5} A + \frac{2}{5} B + \frac{1}{5} C
  \]

• Compute correspondence between pixels of all frames

Results – crossfading/morphing

• Interpolation task:
  \[
  \frac{2}{5} A + \frac{2}{5} B + \frac{1}{5} C
  \]

• Compute correspondence between pixels of all frames

• Interpolate pixel position and color in morphed frame

• based on [Shum 2000]
Results – crossfading/morphing

Jump Cut  Crossfade  Morph

A final example

The Chemical Brothers - Star Guitar
(Director: Michel Gondry)

https://www.youtube.com/watch?v=0S43lwBF0uM