Weather Forecast for Dummies

- Let’s predict weather:
  - Given today’s weather only, we want to know tomorrow’s
  - Suppose weather can only be {Sunny, Cloudy, Raining}

- The “Weather Channel” algorithm:
  - Over a long period of time, record:
    - How often S followed by R
    - How often S followed by S
    - Etc.
  - Compute percentages for each state:
    - \( P(R|S) \), \( P(S|S) \), etc.
  - Predict the state with highest probability!
  - It’s a Markov Chain

Markov Chain

\[
\begin{pmatrix}
0.3 & 0.6 & 0.1 \\
0.4 & 0.3 & 0.3 \\
0.2 & 0.4 & 0.4
\end{pmatrix}
\]

What if we know today and yesterday’s weather?

Text Synthesis

- [Shannon, ’48] proposed a way to generate English-looking text using N-grams:
  - Assume a generalized Markov model
  - Use a large text to compute prob. distributions of each letter given N-1 previous letters
  - Starting from a seed repeatedly sample this Markov chain to generate new letters
  - Also works for whole words

WE NEED TO EAT CAKE

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”
Texture

- Texture depicts spatially repeating patterns
- Many natural phenomena are textures

![Textures: Radishes, Rocks, Yogurt]

Texture Synthesis

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

![Texture Synthesis Example]

The Challenge

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

![Texture Challenge Diagram]

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

![Efros & Leung Algorithm Diagram]

Some Details

- Growing is in “onion skin” order
  - Within each “layer”, pixels with most neighbors are synthesized first
  - If no close match can be found, the pixel is not synthesized until the end
- Using Gaussian-weighted SSD is very important
  - to make sure the new pixel agrees with its closest neighbors
  - Approximates reduction to a smaller neighborhood window if data is too sparse

![Some Details Diagram]

Neighborhood Window

- Input image
- Non-parametric sampling
- Synthesizing a pixel
- Using Gaussian-weighted SSD is very important
  - To sample from this pdf, just pick one match at random

![Neighborhood Window Diagram]
Varying Window Size

Increasing window size

Synthesis Results

french canvas
rafa weave

More Results

white bread
brick wall

Homage to Shannon

Hole Filling

Extrapolation
Summary

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

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!

Our Philosophy

- The “Corrupt Professor’s Algorithm”:
  - Plagiarize as much of the source image as you can
  - Then try to cover up the evidence
- Rationale:
  - Texture blocks are by definition correct samples of texture so problem only connecting them together
Failures (Chernobyl Harvest)
In what order should we fill the pixels?

- choose pixels that have more neighbors filled
- choose pixels that are continuations of lines/crises/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. Similarity to the image being "explained"

Image Analogies

Aaron Hertzmann\textsuperscript{1,2}
Chuck Jacobs\textsuperscript{2}
Nuria Oliver\textsuperscript{2}
Brian Curless\textsuperscript{3}
David Salesin\textsuperscript{2,3}

\textsuperscript{1}New York University
\textsuperscript{2}Microsoft Research
\textsuperscript{3}University of Washington
Image Analogies

Blur Filter

Edge Filter

Artistic Filters

Colorization
Texture-by-numbers

Super-resolution

Super-resolution (result!)

Video Textures

Arno Schödl
Richard Szeliski
David Salesin
Irfan Essa
Microsoft Research, Georgia Tech

Problem statement

Our approach

- How do we find good transitions?
Finding good transitions

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

$D_{i,j}$

vs.

frame $i$

frame $j$

Similar frames make good transitions

Markov chain representation

Similar frames make good transitions

Transition costs

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

Transition probabilities

- Probability for transition $P_{i,j}$ inversely related to cost:
  - $P_{i,j} \sim \exp (-C_{i,j} / \sigma^2)$

Example

Example