

Image Analogies

Aaron Hertzman Charles E. Jacobs Nuria Oliver
Brian Curless David H. Salesin

To be presented at : SIGGraph 2001

Presented By : Sameer Agarwal
Course : Learning Algorithms Seminar
Date : May 23, 2001

Image analogies

- **Goal: Transform an image by example**



A

Specify Analogy

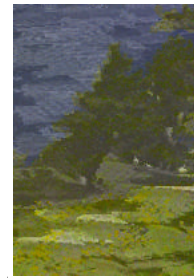


A'



B

Apply Analogy



B'

Why do analogy making ?

- **Image Processing is full of BAD interfaces. Analogies are a natural way of specifying a task.**
- **Some transformations are hard to describe explicitly.**
- **Analogy making could possibly provide a single general purpose mechanism for a broad variety of effects.**

Previous Work

No Previous work on Image Analogy Making exists.

The new tool presented is similar to the *Clone Brush* in Photoshop and *Image Hose* in Fractal Designer

Builds upon recent work on Texture Synthesis by

Ashikmin [2001]

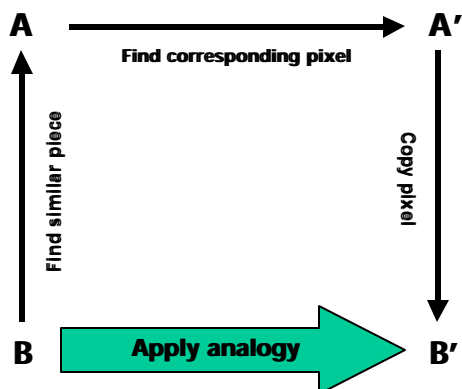
Wei and Levoy [2000]

Efros and Leung [1999]

Contributions of this paper

1. **A new framework for transforming images by example**
2. **Generalizes the task of texture synthesis to making image analogies.**
3. **Demonstrates how a variety of image processing tasks can be accomplished within this framework.**

How do you make analogies ?



So where is the learning ?

- The figure implicitly assumes a nearest neighbor learner.
- Pixels in A' serve as labels for neighborhoods in A
- An alternate algorithm can learn a function that maps the image A to A' and then use image B as input to produce B' .
- e.g. a Neural network trained to map 3×3 pieces from A to the pixel intensity of the center pixel in A' .

Principal Task

Given a structure in B find similar structure in A

Two subtasks

1. **Measure similarity of two structures**
 1. **Choose the feature space**
 2. **Choose a distance metric**
2. **Perform search for similar structures in A**

Organization of the rest of the talk

- **Underlying assumptions of the algorithm**
- **Design of the algorithm**
 - **Gaussian Pyramids**
 - **Similarity measures**
 - **Search algorithm**
 - **Multi-scale rendering**
- **Applications**

Assumption

The image can be modeled as a Markov Random Field

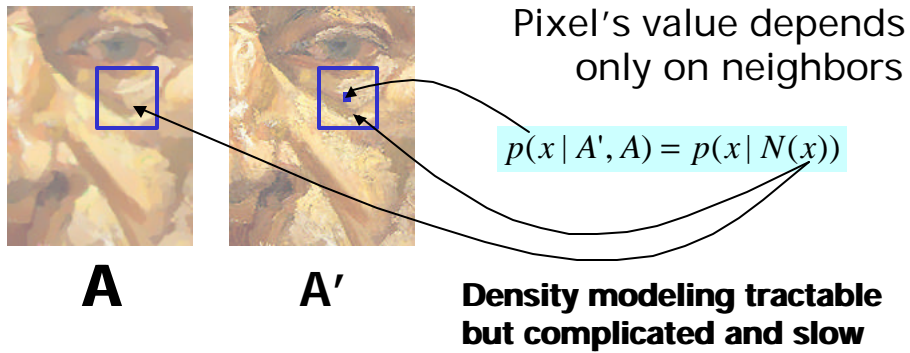
Markov Chain : A random process which defines a random variable X_t for each value of a discrete parameter t (usually corresponding to time), in which the value of X_{t+1} depends only on X_t , and is independent of all the earlier values.

Markov Random Field (MRF) : A two-dimensional version of a Markov chain. It is a conditional probability model where the probability of a pixel depends on its neighborhood. (The shape and size of the neighborhood is fixed across the image).

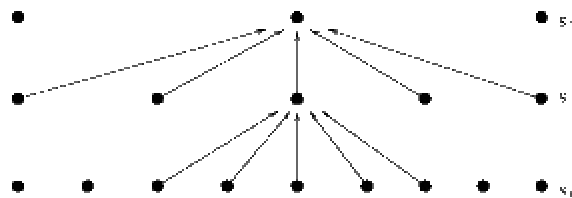
Any particular ordering of the pixels (e.g. scanline) can serve as time t .

Markov Random Fields

- **Problem: Model probability $p(A'|A)$**
- **Locality assumption**



Gaussian Pyramids



The Gaussian Pyramid is a hierarchy of low-pass filtered (high frequency removed) versions of the original image, such that successive levels correspond to lower frequencies.

Notation

A , A' B and B' are arrays of the form

Array : level x point \longrightarrow feature

A[l,p] denotes the feature vector associated with pixel p at level l.

s is an array of the form

Array : level x point \longrightarrow point

s[l,p] denotes the pixel in the image B at level l that is copied into the image B' at position p at the same level.

CreateImageAnalogy(A,A',B)

Computer Gaussian Pyramids for A, A', B

Compute features for A, A' and B

Initialize the search structures (Approx. NN)

For l = 1 to L do:

For each pixel $q \in B'_l$ in scanline order do :

p = BestMatchingPixel(A,A',B,B',s,l,q)

B'[l,q] = A'[l,p]

s[l,q] = p

Return B'

BestMatchingPixel(A,A',B,A',s,l,q)

a = BestApproxMatchingPixel(A,A',B,B',l,q)

c = MostCoherentPixel(A,A',B,B',s,l,q)

da = $|F[l,a] - F[l,q]|^2$

dc = $|F[l,c] - F[l,q]|^2$

if $dc \leq da \cdot (1 + k \cdot 2^{l-L})$ return c else return a

- $F[l,q]$ is the feature vector associated with the pixel q at level l .
- L is the total number of layers in the image pyramid
- k is scaling factor to compensate for the distance between neighboring pixels at different levels.
- The norm $|F[l,a] - F[l,q]|^2$ is calculated using a weighted distance over the feature vectors to emphasize nearer pixels.

Best Approximate Match

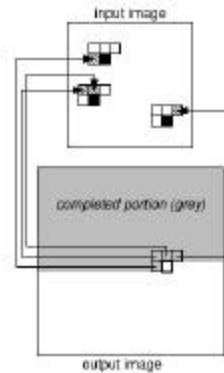
- PCA for reducing the dimension of the feature vectors (99% variance retained)
- Naïve method – Ordinary Nearest Neighbor
 - Complexity $O(\text{size of } A = N)$
- Use Approximate Nearest Neighbor (ANN)
 - Complexity $O(C \cdot \log^3 N)$ where C is large
- Tree Structured Vector Quantization (tree of clusters of similar points)
 - Complexity $O(\log N)$

Coherence Matching

Based on Ashikhmin (2000)

Intuition:

Pixels from the input sample that are appropriately "forward shifted" with respect to pixels already used in synthesis are well suited to fill in the current pixel.



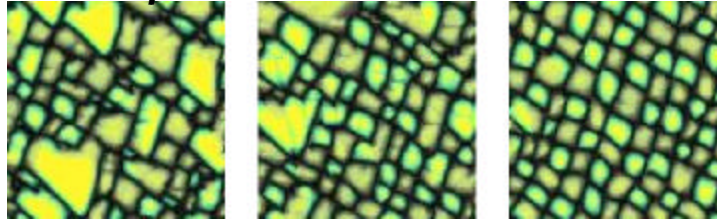
$$r = \arg \min_{r \in N(q)} |F[l, s[l, r] + (q - r)] - F[l, q]|^2$$

Features

- **RGB Channel information**
 - Suffers from the curse of dimensionality.
- **Luminance channel information (Channel Y from the YIQ colorspace) is used when the colors of the source and target images are very different.**
 - Color dependencies are lost.
 - Requires histogram matching.
- **Pyramids of filter responses corresponding to the image pyramids. Useful for providing derivative information which used in synthesizing lineart exampled.**

Neighborhood size

- **Larger neighborhoods give better results, but are slower**



5x5

7x7

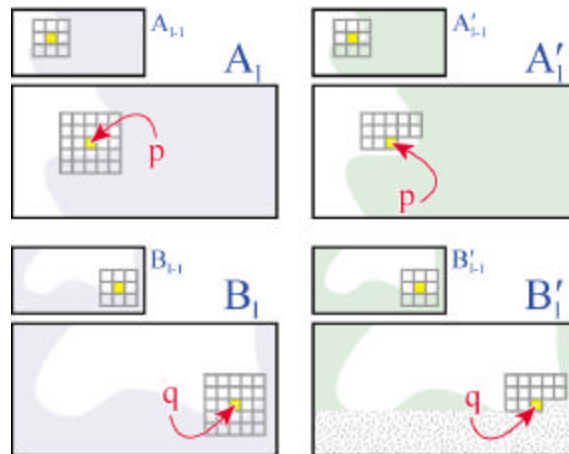
9x9

Images from Wei-Levoy 2000

Multiresolution Synthesis

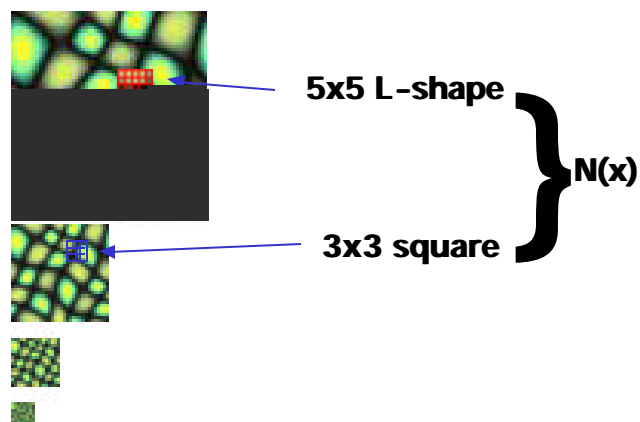
- **Single resolution captures textures by using adequately sized neighborhoods.**
- **Large neighborhood = More computation**
- **Multi-scale synthesis works because we can represent large scale structures by a few pixels in a low resolution image**
- **A neighborhood in level $l-1$ (previous) is attached with each pixel at level l as additional feature vector components. The point (x,y) at level l is associated with the 3×3 neighborhood centered at $(x/2, y/2)$ in the level $l-1$ image.**
- **This insures that as the synthesis proceeds, the high frequency details added in each layer are consistent with the already synthesized low frequency structures.**

Scanline Neighborhood Matching



Coarse-to-fine

- Neighborhood includes coarser image
- Wei and Levoy, SIGGRAPH 2000



Application : Blur



A



A'



B



B'

Application : Superresolution



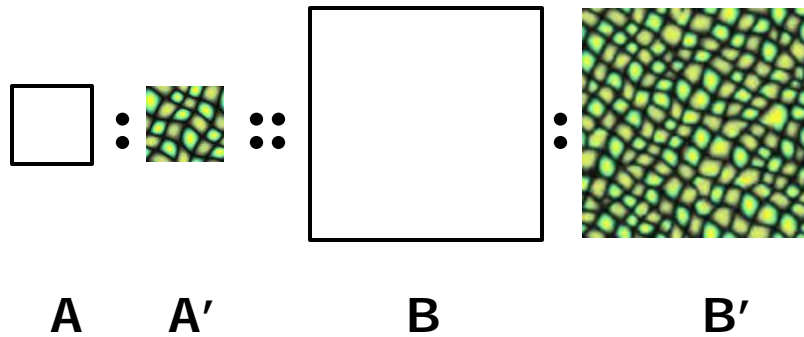
Training Pairs

Application : Superresolution

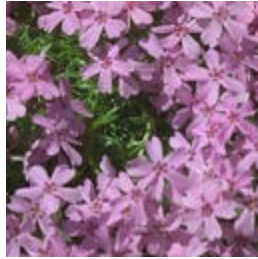


Application : Texture Synthesis

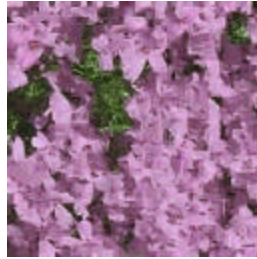
- Image analogy: constant A and B



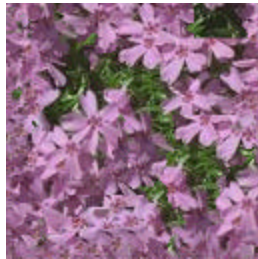
Improved Synthesis



Example texture



Wei-Levoy



Ashikhmin



Our algorithm

Application : Texture transfer



A



A'

(same texture)



B



Closer to texture

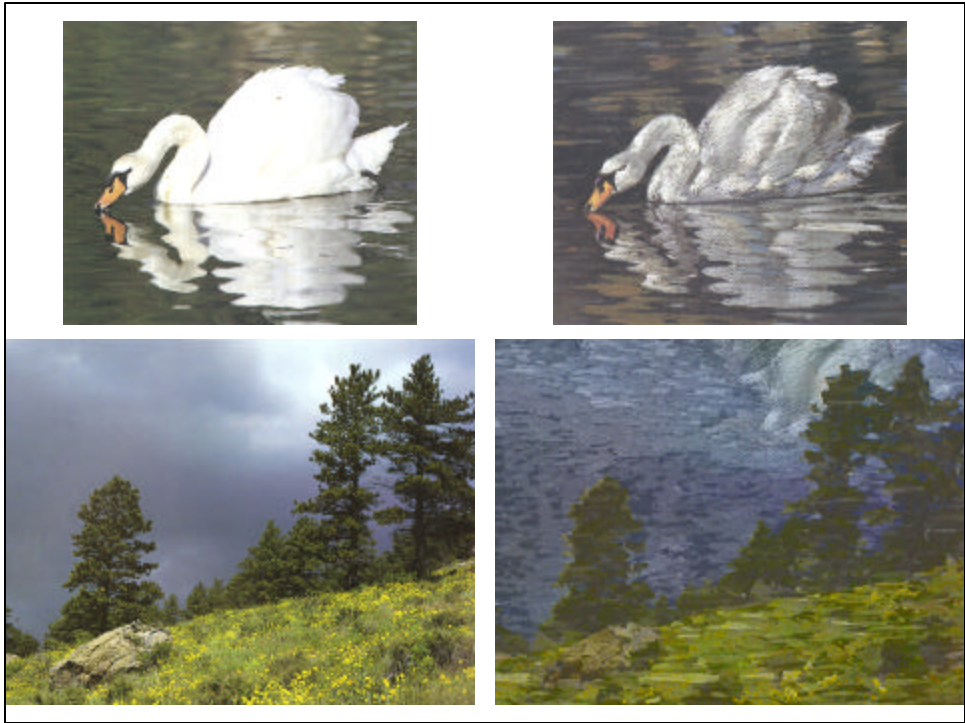


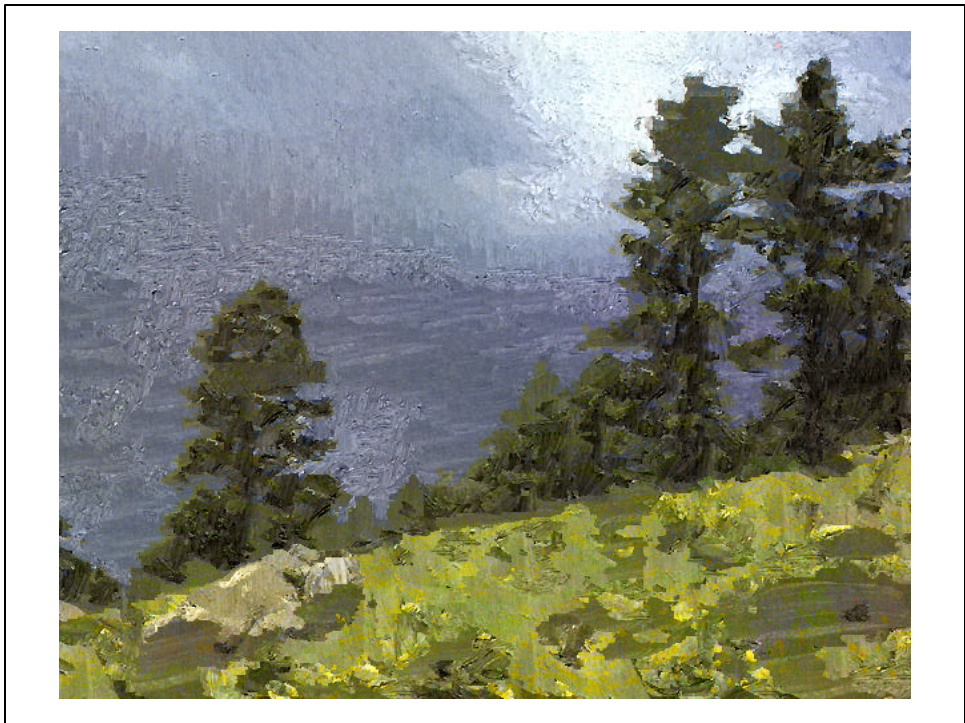
Closer to photo

B's

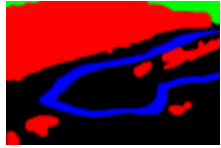
Application : Painting and Drawing







Application : Texture-by-Numbers

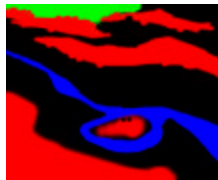


A



A'

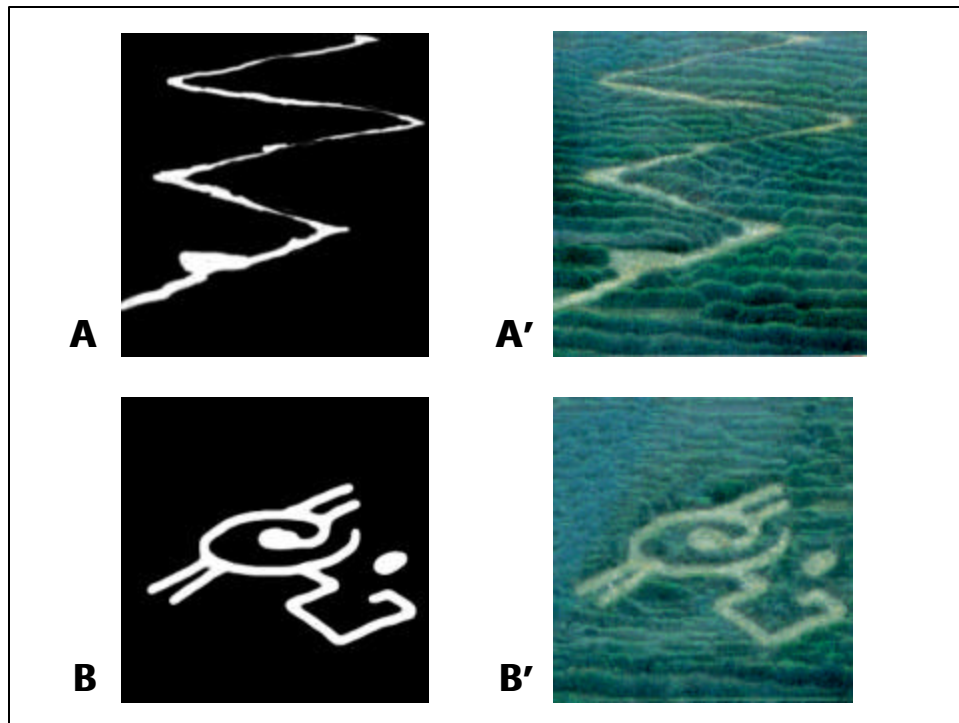
Application : Texture-by-Numbers



B



Result (B')



Limitations and Future Work

- **Speed**
- **Better perceptual matching and feature selection**
- **Color Handling – Many of the images analogies only used the luminosity channel.**
- **Image registration**
- **Extension to 3D and other domains.**
- **Estimation of alternate image statistics e.g. brush stroke structure.**

Conclusions

- **A new method for example based rendering is introduced.**
- **A texture similarity measure that considers feature similarity as well as coherence is presented.**
- **Many image filters and synthesis tasks can be performed within this framework.**

Acknowledgements

- Aaron Hertzman for sharing his slides, and pointing to sources of explanation
- Active Web for lending a laptop
- The music of Vitalij Kuprij for keeping me company.

Thank You

