Lecture 21:

PatchMatch +

Course-so-far Review
(in-class review, no slides)

Visual Computing Systems
CMU 15-869, Fall 2013
Today’s theme

- Image manipulation by example
Data-driven texture synthesis

- Input: low resolution texture image
- Want: high resolution texture that appears “like” the input

Source texture (low resolution) → High resolution texture generated by tiling
Non-parametric texture synthesis

Increasing neighborhood search window

Source textures

Synthesized Textures

5x5 11x11 15x15 23x23

[Efros and Leung 99]
Algorithm: non-parametric texture synthesis

Main idea: given NxN neighborhood $w(p)$ around unknown pixel $p$, want probability distribution function for value of $p$, given $w(p)$.

For each pixel $p$ to synthesize:

1. Find other patches in the image that are similar to the NxN neighborhood around $p$ (use gaussian weighted SSD as the patch distance function)

2. Center pixel of patches are candidates for $p$

3. Randomly sample from candidates weighted by distance $d$

[Efros and Leung 99]
More texture synthesis examples

Source textures

Synthesized Textures

Naive tiling solution
Image completion example

Original Image

Masked Region

Completion Result

Image credit: [Barnes et al. 2009]
Problem: low performance

- Large patch windows + full image search = slow
  - Large windows: preserve structure
  - Full-image search: highly relevant examples are rare

- Must repeat search process for all pixels to complete

- Possible accelerations
  - Limit search window
  - Use acceleration structure for search (e.g., k-d tree)
  - Dimensionality reduction of patches + approximate nearest neighbor search (ANN)
  - Exploit image coherence
PatchMatch

- A randomized algorithm for rapidly finding correspondences between image patches

Problem definition:
- Given images A and B, for each overlapping patch in image A, compute the offset to the nearest neighbor patch in image B
- Overlapping patches: each patch defined by its center pixel (ignoring boundary conditions, each MxN image consists of MxN patches)
- PatchMatch computes nearest neighbor field (NNF)
  - NNF is function $f: A \rightarrow \mathbb{R}^2$ (maps patches in A to patches in B)
  - Example: if patch b in B is NN of patch a in A, then $f(a) = b$
Patch match: key idea one

- Law of large numbers: a non-trivial fraction of a large field of random offset assignments are likely to be good guesses
- Initialize $f$ with random values

Visualization of $f$:

Saturation = magnitude of match offset (gray is matching patch in B is at same pixel location as match patch in A)

Hue = direction of offset
  offset X = red-cyan axis
  offset Y = blue-yellow axis

Image credit: [Barnes et al. 2009]
PatchMatch key idea two: spatial coherence

- High coherence of nearest neighbors in natural images
- Nearest neighbor of patch at \((x,y)\) should be a strong hint for where to find nearest neighbor of patch at \((x+1,y)\)

How this graph was made:
1. Compute NNF for collection of images
2. For select pixels \((x,y)\), compare NN offset to NN offsets of adjacent pixels \((x-1,y), (x+1,y), (x,y-1), (x,y+1)\)

Image credit: [Barnes et al. 2009]
Propagation: improving the NNF estimate

- The NNF estimate provides a “best-so-far” NN for each patch in A
  - NN patch: \( f(a) \)
  - NN distance = \( d(a,b) \)  (where \( b = f(a) \))

- Try to improve NNF estimate by exploiting spatial coherence with left and top neighbor:
  - Let \( a = (x,y) \), then candidate matches for \( a \) are:
    - \( f(x-1, y) + (1,0) \)
    - \( f(x, y-1) + (0,1) \)
  - If candidate patch is better match than \( f(a) \), then replace \( f(a) \) with candidate
    - Replace \( f(a) \) with candidate patch if \( d(a, f(x,y-1)+(0,1)) < d(a, f(a)) \)

- Next iteration, use bottom and right neighbors as candidates
PatchMatch iterative improvement

Image A

Image B (source of patches)

Experiment:
Reconstruct A using patches from B

Random init: 1/4 through iter 1

End of iter 1

Iter 2

Iter 5

Image credit: [Barnes et al. 2009]
Random search: avoiding local minima

- Propagation can cause PatchMatch to get stuck in local minima

- Sample random sequence of candidates from exponential distribution
  - Let $a=(x,y)$, then candidate matches for $a$ are: $(x,y) + w \alpha^i R^i$
  - $R^i$ is uniform random offset in $[-1,1] \times [-1,1]$
  - $w$ is maximum search radius (e.g., width of entire image)
  - $\alpha$ is typically $1/2$
  - Check all candidates where $w \alpha^i \geq 1$
Optimization: enrichment

- Propagation step propagates good matches across spatial dimensions of image
- Can also propagate good matches across space of matches itself
- Idea: if $f(a) = b$, and $f(b) = c$, then $c$ is a good candidate match for $a$
  - If you think of the NNF as a graph, then enrichment looks for nodes reachable in two steps
  - Note: assumes we’re searching for matches in the same image as the image we are trying to complete
Example applications

Photoshop's Content Aware Fill

Object Manipulation

Building segment marked by user

Building scaled up, preserving texture

Image retargeting (changing aspect ratio)

Original image (with user-provided search constraints)

Retargeted (without constraints)

Retargeted (with constraints)

Image credits: [Barnes et al. 2009]
PatchMatch summary

- Randomized algorithm
  - Converges rapidly in practice

- Main idea: coherence (largely spatial) of nearest neighbors

- Propagation step is inherently serial, but good parallel approximations exist
  - PatchMatch has been implemented efficiently on GPUs

- Data access caches well, but is unpredictable
  - Different from many other image processing algorithms we have discussed