Texture Synthesis

COS 526

Tom Funkhouser
Fall 2012

Slides from Efros, Freeman, Lazebnik, Wei

Texture

- Texture has spatially repeating patterns
- It lacks the full range of complexity of photographic imagery, but makes a good starting point for study of image-based techniques

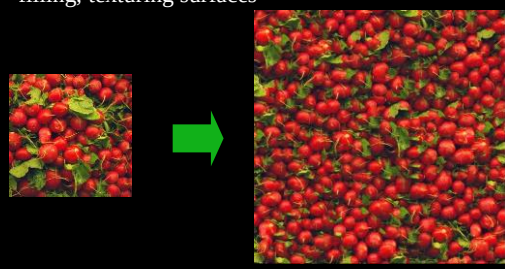

radishes


rocks


yogurt

Texture Synthesis

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

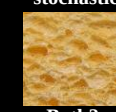


The Challenge

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


repeated


stochastic


Both?

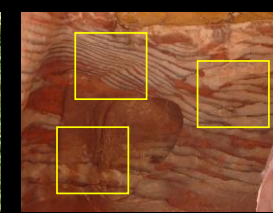
Some History

- Stochastic textures
  - [Heeger & Bergen,'95]
  - [DeBonet,'97]
  - [Portilla & Simoncelli,'98]
- Structured textures
  - [Liu, '04]
- Both
  - [Efros & Leung,'99]
  - [Efros & Freeman,'01]
  - [Kwatra, `05]

Statistical modeling of texture

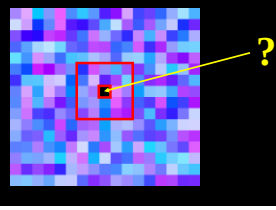
- Assume stochastic model of texture (*Markov Random Field*)
- *Stationarity*: the stochastic model is the same regardless of position


stationary texture


non-stationary texture

Statistical modeling of texture

- Assume stochastic model of texture (*Markov Random Field*)
- *Stationarity*: the stochastic model is the same regardless of position
- *Markov property*:
  p(pixel | rest of image) = p(pixel | neighborhood)



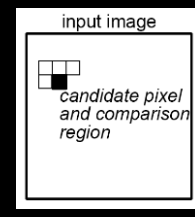
Motivation from Language

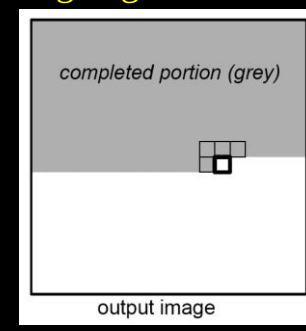
- Shannon (1948) proposed a way to generate English-looking text using *N-grams*
  - Assume a Markov model
  - Use a large text to compute probability distributions of each letter given N–1 previous letters
  - Starting from a seed repeatedly sample the conditional probabilities to generate new letters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

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...

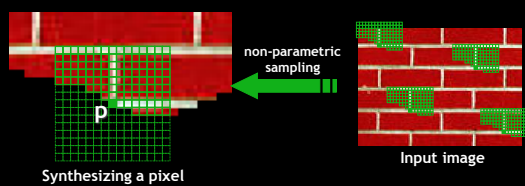
Efros & Leung Algorithm





Idea initially proposed in 1981 (Garber '81), but dismissed as too computationally expensive!

Efros & Leung Algorithm



- Assume Markov property, sample from P(**p**|N(**p**))
  - Building explicit probability tables infeasible
  - Instead, we *search the input image* for all sufficiently similar neighborhoods and pick one match at random

Finding matches

- Sum of squared differences (SSD)

Finding matches

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

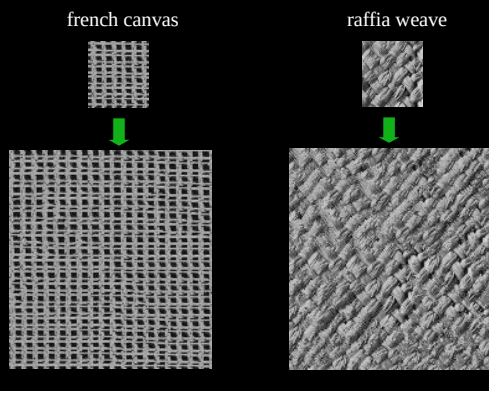
Efros

Implementation Details

- Initialization
  - Start with a few rows of white noise and grow in scanline order
  - Start with a "seed" in the middle and grow outward in layers
- Sampling
  - Random sampling from the set of candidates vs. picking the best candidate

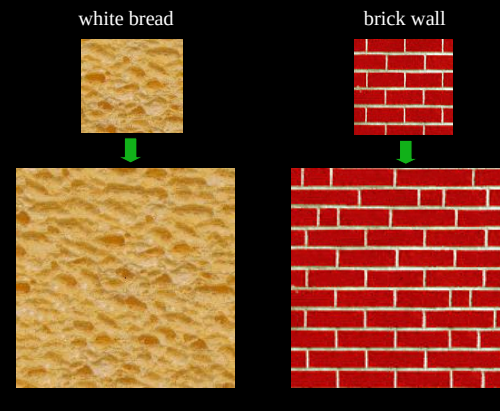
Efros

Synthesis Results



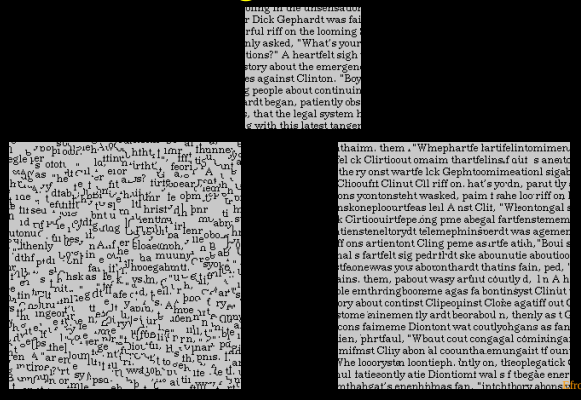
Efros

More Results



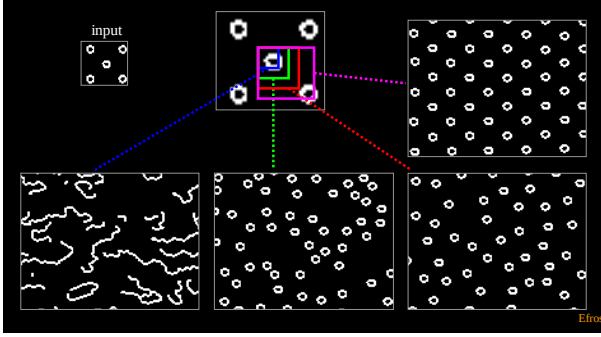
Efros

Homage to Shannon



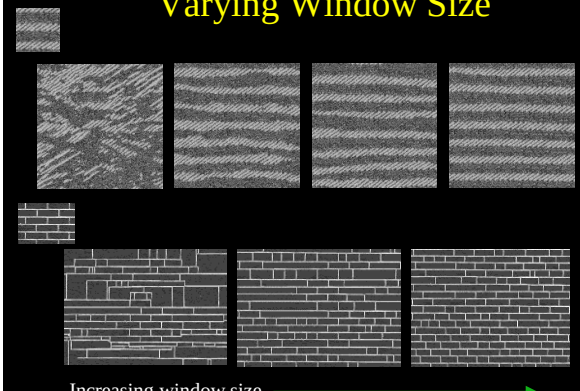
Efros

Varying Window Size



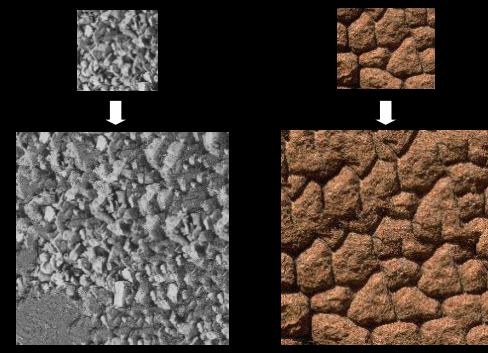
Efros

Varying Window Size



Efros

Failure Cases



Growing garbage **Verbatim copying**

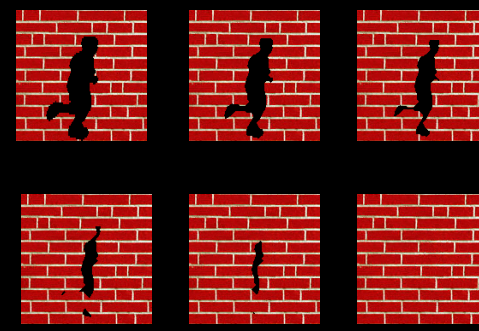
Efros

Hole Filling and Extrapolation

- Grow in "onion skin" order
  - Within each "layer", pixels with most neighbors are synthesized first
  - Normalize error by the number of known pixels
  - If no close match can be found, the pixel is not synthesized until the end

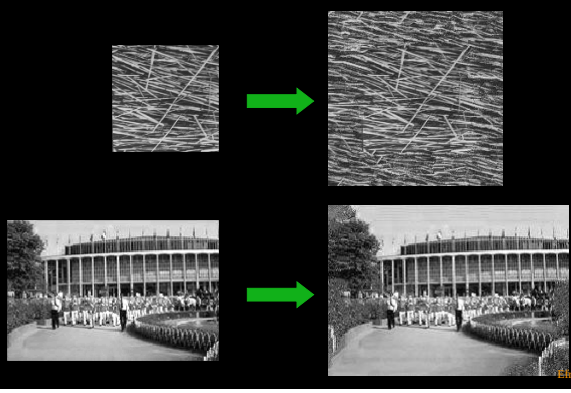
Efros

Hole Filling



Efros

Extrapolation



Efros

Summary

- The Efros & Leung algorithm
  - Very simple
  - Surprisingly good results
- Problems?

Accelerating texture synthesis

- Indexed similarity search
- Coherence
- Multiresolution
- Patches

Indexed Similarity Search

- Perform fast approximate nearest neighbor search using spatial search structure
  - *tree-structured vector quantization (TSVQ)*
  - *kd-tree*

Indexed Similarity Search

- Perform fast approximate nearest neighbor search using e.g. *tree-structured vector quantization*
  - Use all neighborhoods of the exemplar texture to build a tree-structured codebook
  - To find a match for a new neighborhood, follow the tree in best-first order (at each level, choose child codeword closest to the query)
  - Example running times from the paper:
    - Exhaustive search: 360 sec
    - Building codebook: 22 sec, synthesis: 7.5 sec
  - Shortcomings?

Wei00

Indexed Similarity Search

- Can degrade quality (blur)


original

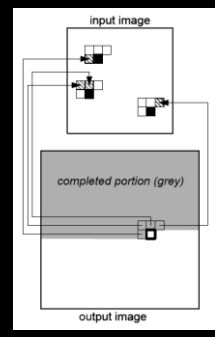

Full search


TSVQ

Wei00

Coherence

- Use original position of already synthesized neighborhood pixels to create a "short list" of candidates for the current pixel



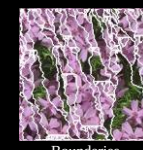
Ashikhmin01

Coherence


Original sample


Wei & Levoy

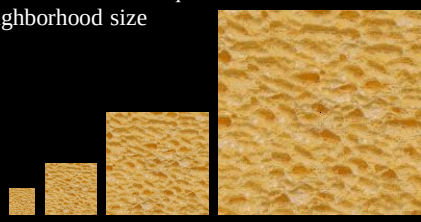

Ashikhmin


Boundaries

Ashikhmin01

Multiresolution

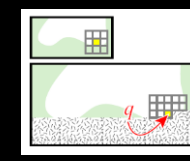
- For textures with large-scale structures, use a *Gaussian pyramid* to reduce required neighborhood size



Wei00

Multiresolution

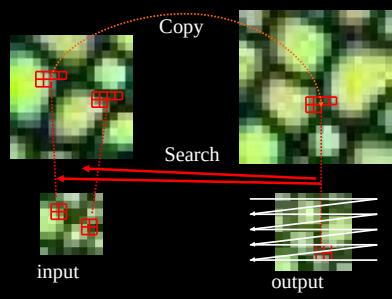
- For textures with large-scale structures, use a *Gaussian pyramid* to reduce required neighborhood size
  - Low-resolution image is synthesized first
  - For synthesis at a given pyramid level, the neighborhood consists of already generated pixels at this level plus all neighboring pixels at the lower level



Wei00

Multiresolution

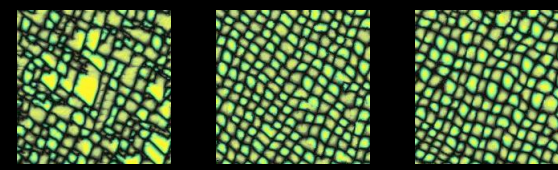
- Example:



Wei00

Multiresolution

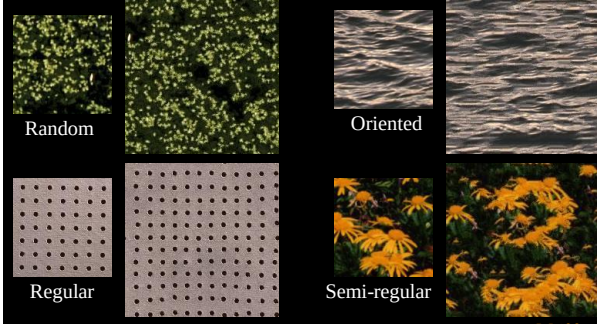
- Results



1 level 5×5 | 1 level 11×11 | 3 levels 5×5

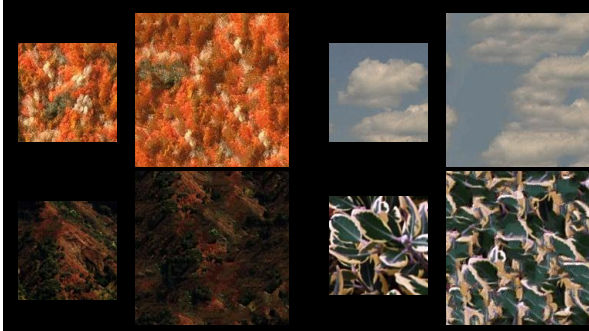
Wei00

Multiresolution



Wei00

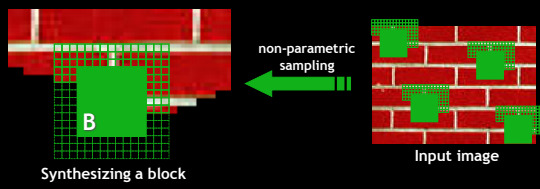
Multiresolution



Wei00

Patch-Based Synthesis

- Copy patches of pixels rather than pixels



- Observation: neighbor pixels are highly correlated
  - Exactly the same as Efros & Leung but P(**B**|N(**B**))
  - Much faster: synthesize all pixels in a block at once

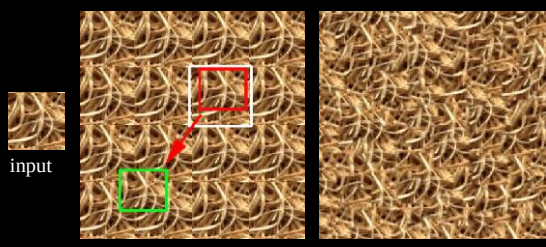
Efros01

Patch-Based Synthesis

- General approach:
  - Copy large blocks from input image
  - Then hide the seams
- Rationale:
  - Texture blocks are by definition correct samples of texture so problem only connecting them together

Efros01

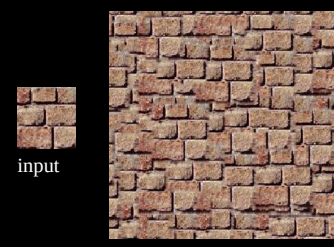
Chaos Mosaic



- Process: 1) tile input image; 2) pick random blocks and place them in random locations 3) Smooth edges

Xu00

Chaos Mosaic

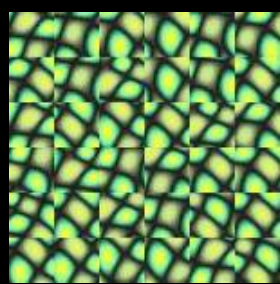


- Of course, doesn't work for structured textures

Xu00

Image Quilting [Efros & Freeman]

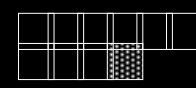
- Regularly arranged patches



Efros01

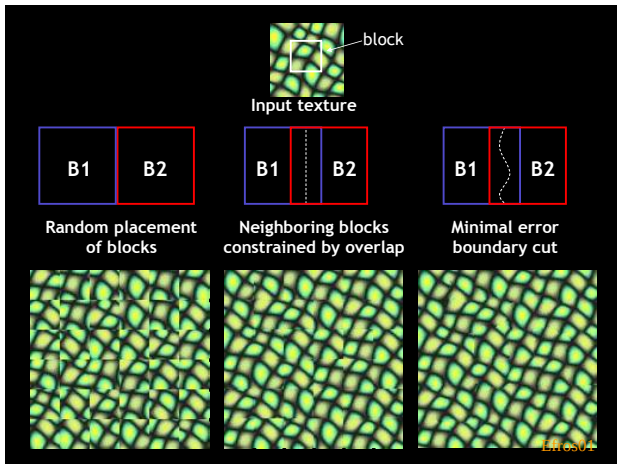
Algorithm

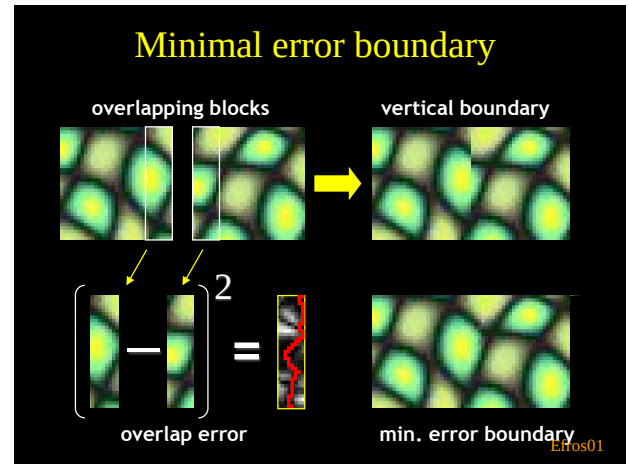
- Pick size of block and size of overlap
- Synthesize blocks in raster order

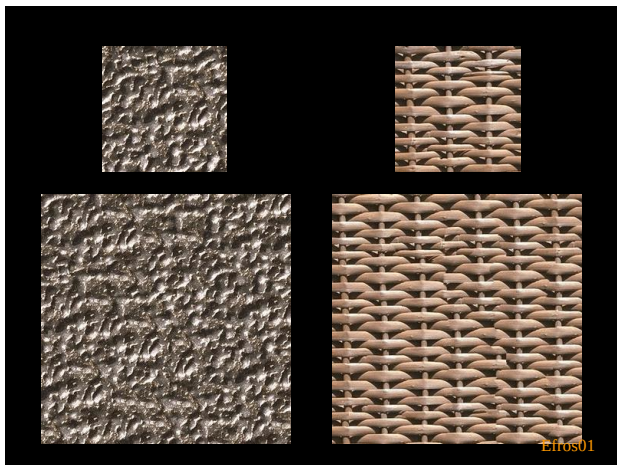


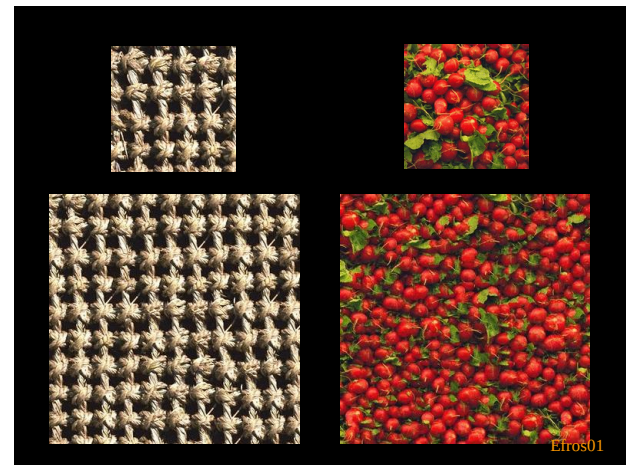
- Search input texture for block that satisfies overlap constraints (above and left)
- Paste new block into resulting texture
  - use dynamic programming to compute minimal error boundary cut

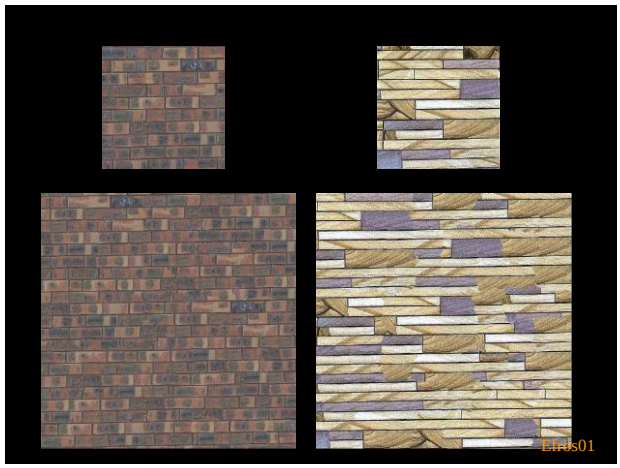
Efros01

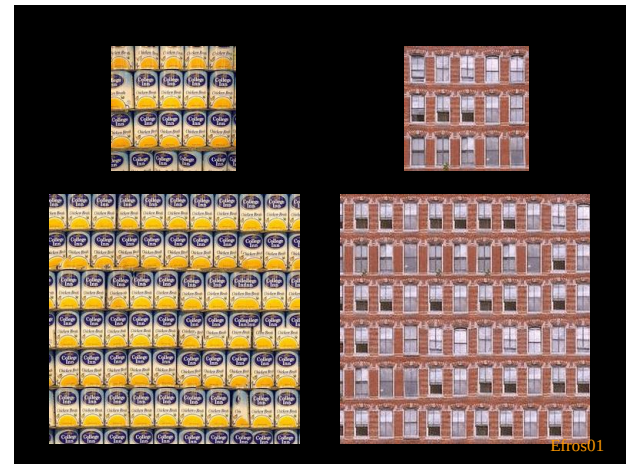

block
Input texture
B1
B2
B1
B2
B1
B2
Random placement of blocks
Neighboring blocks constrained by overlap
Minimal error boundary cut
Efros01

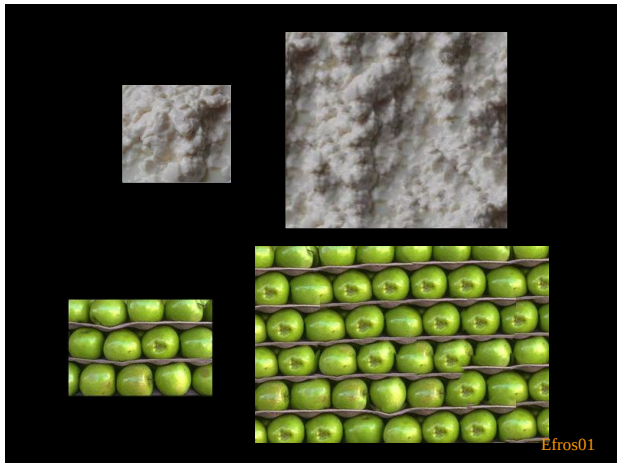

Minimal error boundary
overlapping blocks
vertical boundary
2
overlap error
min. error boundary
Efros01

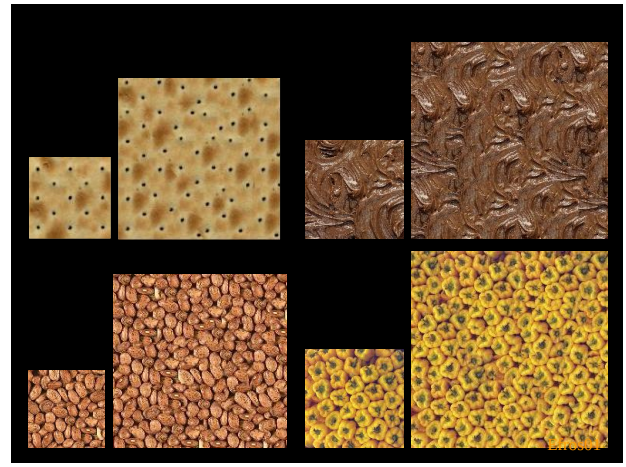

Efros01


Efros01


Efros01


Efros01





Summary

- Texture synthesis
  - create new samples of a given texture
- Non-parametric methods
  - Copy samples from input based on neighborhood similarity
- Acceleration techniques
  - Multiresolution
  - Indexing
  - Coherence
  - Patches