Textures and Texture Synthesis

Martin Hatka

Outline

- motivation
- textures
- bidirectional texture function
- texture synthesis, texture generating

- texture synthesis algorithms
- BTF roller

Motivation

- to enhance realism in virtual scene
  - textures, bump maps, BTF
- texture mapping
  - an efficient way to include detail on the surfaces of objects in the scene
  - smaller texture than object surface to be covered
    - larger texture
    - tiling
    - **texture synthesis**

Image Texture

Texture is a structure which is made of a large ensemble of elements that resemble each other “very much”, with some kind of an “order” in their locations, so that there is no one element which attracts the viewer’s eye in any special way. The human viewer gets an impression of uniformity when he looks at a “**texture**”.

Textures and BTF Textures

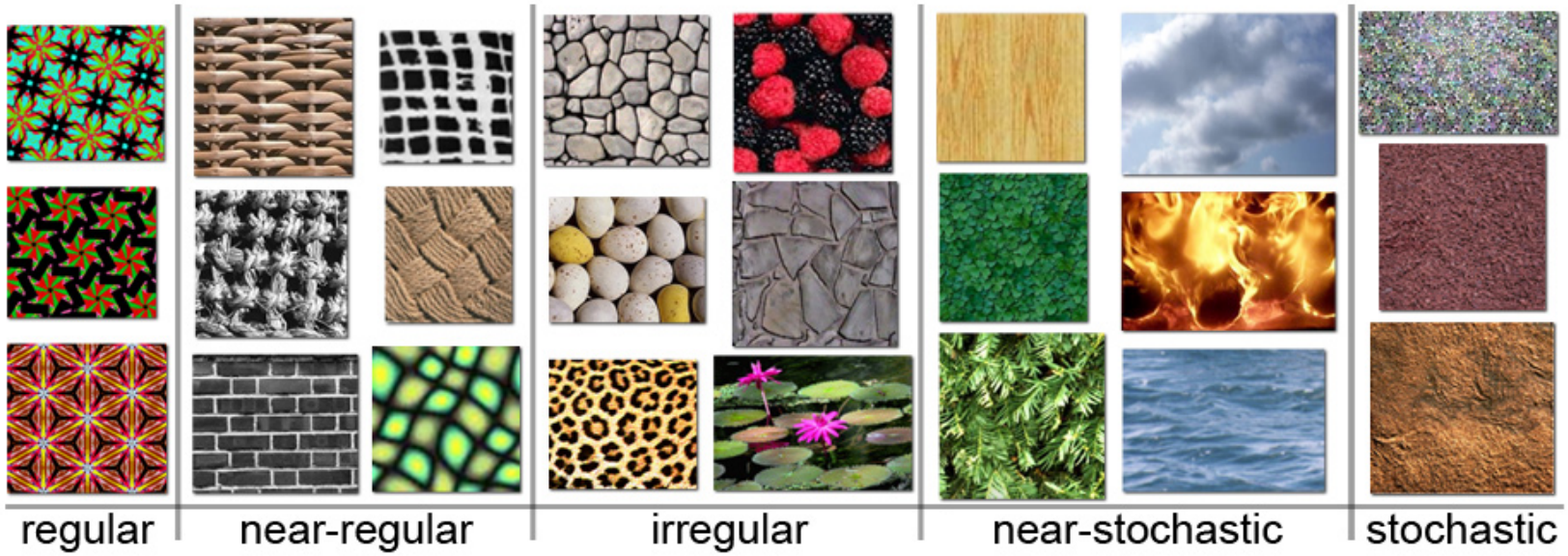
Image Texture

- there is no definition of texture
- properties
  - set of elements
  - homogenity, uniformity

Texture Categorization

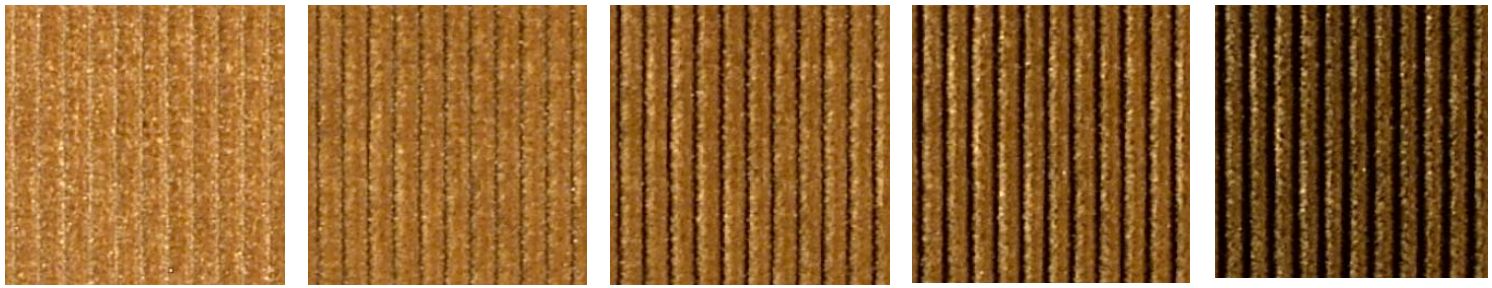
- origin – artificial, natural
- spectral – mono-spectral, multi-spectral
- time dependence – static, dynamic
- surface – smooth, rough

Texture Types



Bidirectional Texture Function

- view and illumination angle dependent
- realistic appearance



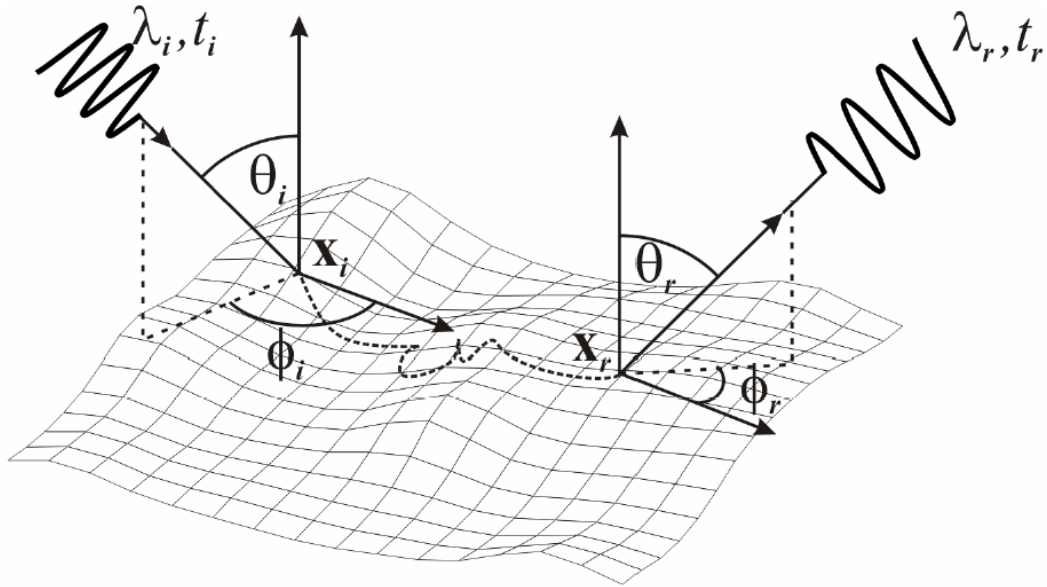
Bidirectional Texture Function

Acquisition of BTF

- 2D texture
  - simple process – 2D scanner, digital camera
- BTF texture
  - complex and controlled measurement environment
  - physical measurement of real-world reflection
  - device calibration, image registration

Reflectance Measurement

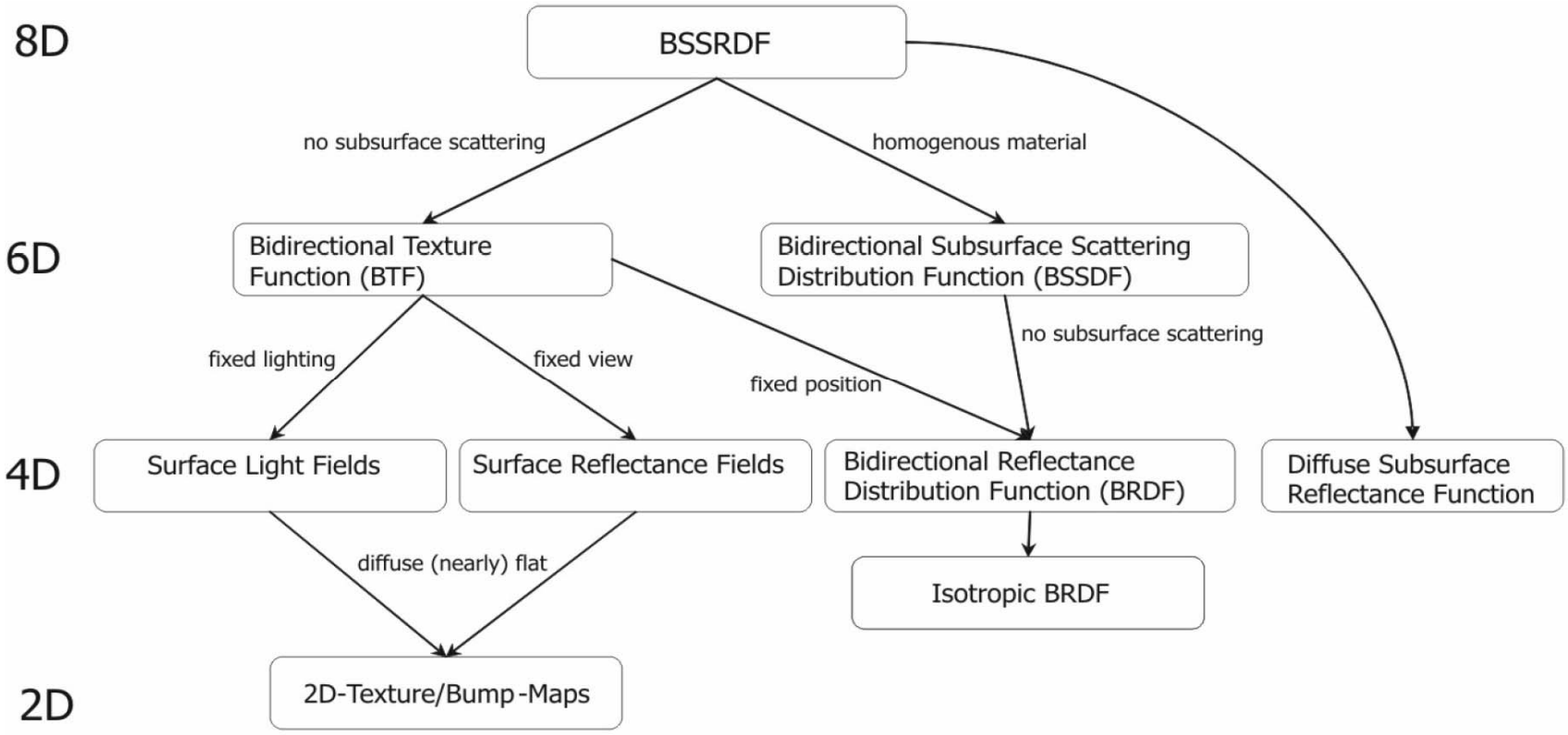
- path of photon
- parameters of general light-material interaction
- 12 parameters



Reflectance Measurement

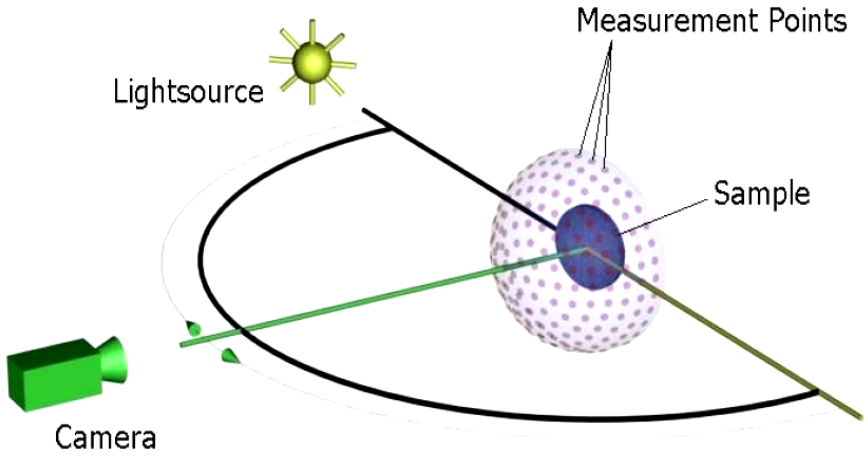
- 12 parameters → 8 parameters
  - light transport takes zero time
  - time invariant reflectance behavior of surface
  - interaction does not change wavelength
  - wavelength → 3 color bands, RGB
- 8D function, Bidirectional Surface Scattering Distribution Function (BSSRDF)
- BTF, BRDF, …

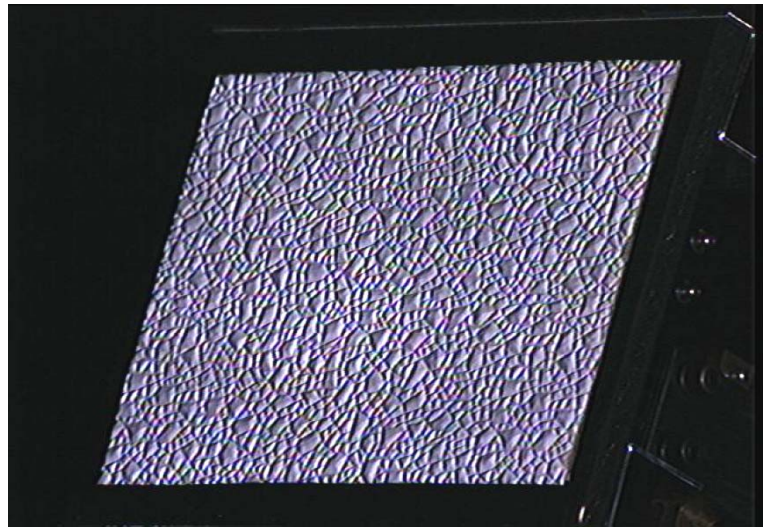
Hierarchy of Reflectance Functions



BTF Measurement

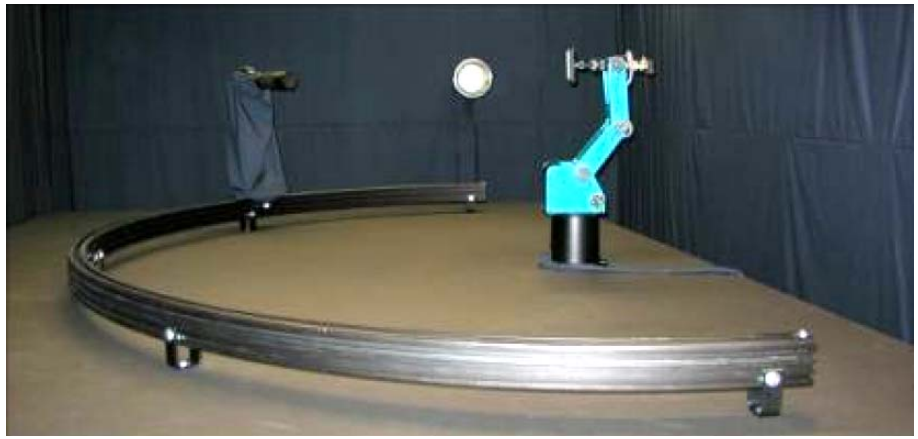
$$BTF_{RGB}(\mathbf{x},\theta_i,\phi_i,\theta_r,\phi_r) = \int_S BSSRDF_{RGB}(\mathbf{x}_i,\mathbf{x},\theta_i,\phi_i,\theta_r,\phi_r)\,d\mathbf{x}_i$$

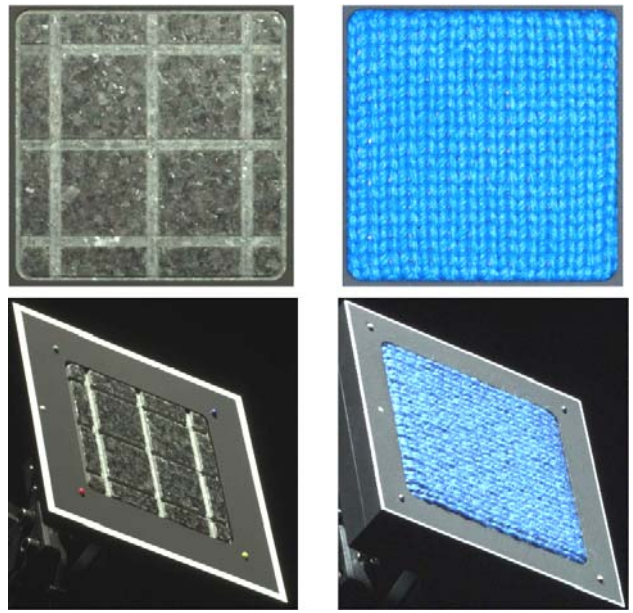




BTF Measurement

- measurement setup of the Bonn-System
  - HMI lamp, CCD camera, robot with a sample holder



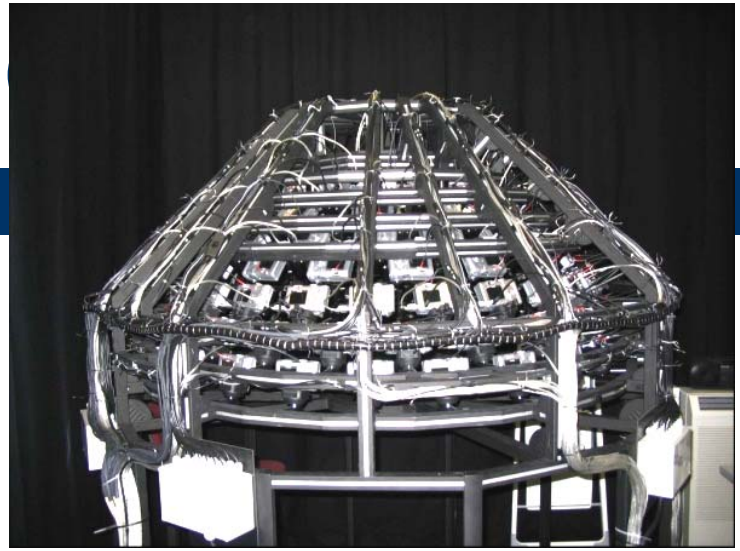


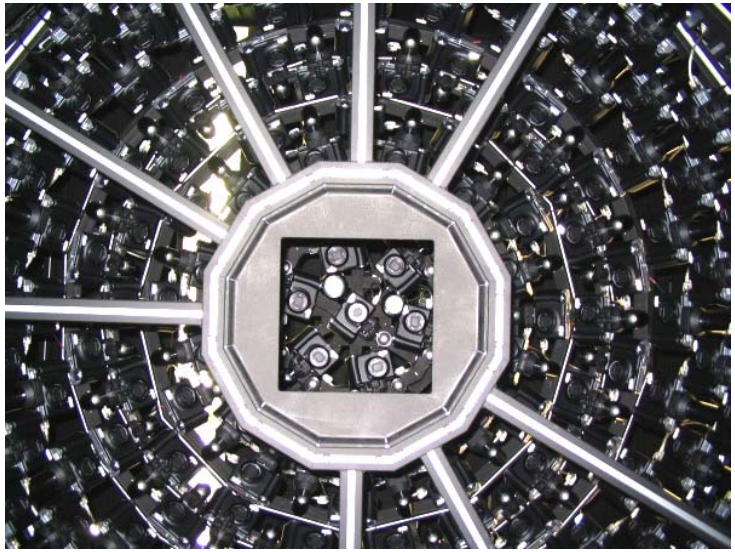
BTF Measurement

- measurement setup of the Bonn-System
  - 10x10 cm maximum sample size
  - 81 positions for camera, 81 positions for light
  - raw data - 4500x3000 pixels, 12MB per image
  - calibration (light-camera, aperture, geometric distortion)
  - data postprocessing (rectification, registration, resize) → 1.2GB per material

BTF Measurement

- camera array
  - fast high-quality acquisition of BTF
  - 151 digital cameras (22801 images)
  - no moving parts
  - known region of interest, no time-consuming detection of ROI
  - angular resolution – 0,04161 steradians
  - spatial resolution – 280DPI for BTF 1024x1024

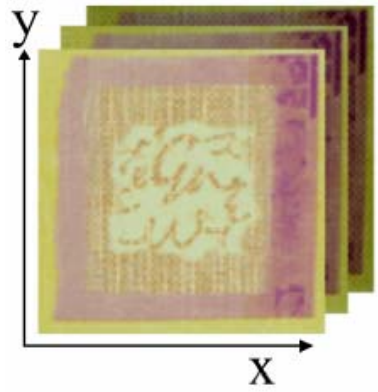




BTF Compression

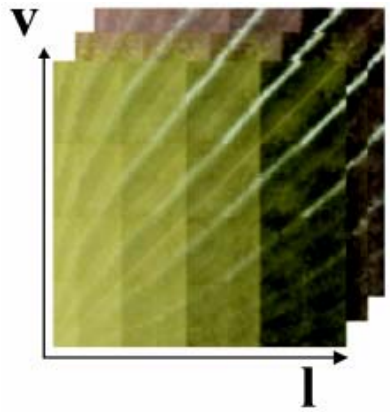
- collection of discrete textures

$$\left\{T_{(\mathbf{v},\mathbf{l})}\right\}_{(\mathbf{v},\mathbf{l})\in M}$$



- set of spatially varying apparent BRDF

$$\left\{B_{\mathbf{x}}\right\}_{\mathbf{x}\in I\subset \mathbf{N}^2}$$



BTF Compression

- fitting analytical BRDF-models
  - Lafortune lobes

$$s(\mathbf{v},\mathbf{l}) = (\mathbf{v}^T \cdot \mathbf{M} \cdot \mathbf{l})^n$$

  - reflectance fields

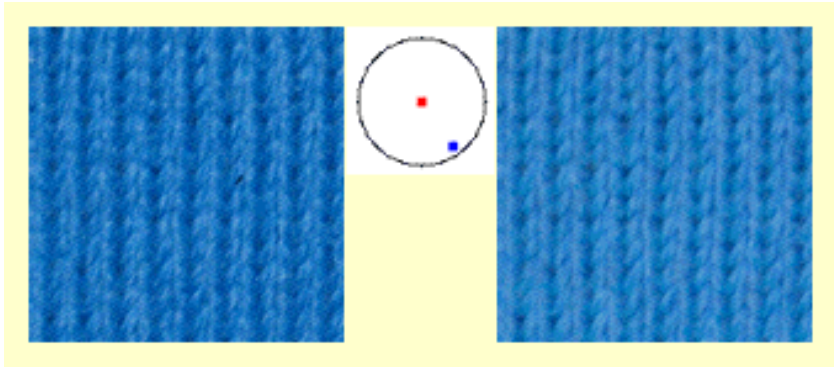
$$BTF(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \sum_{v \in N(\mathbf{v})} w_{\mathbf{x},v} RF_{\mathbf{x},v}(\mathbf{l})$$

$$RF_{\mathbf{x},v}(\mathbf{l}) \approx \rho_{d,\mathbf{x}} + \rho_{s,\mathbf{x},v}(\mathbf{l}) \cdot \sum_{i=1}^{k} s_{\mathbf{x},v}(\mathbf{l})$$

BTF Compression

- fitting analytical BRDF-models
  - Lafortune lobes

$$s(\mathbf{v},\mathbf{l}) = (\mathbf{v}^T \cdot \mathbf{M} \cdot \mathbf{l})^n$$



  - reflectance fields

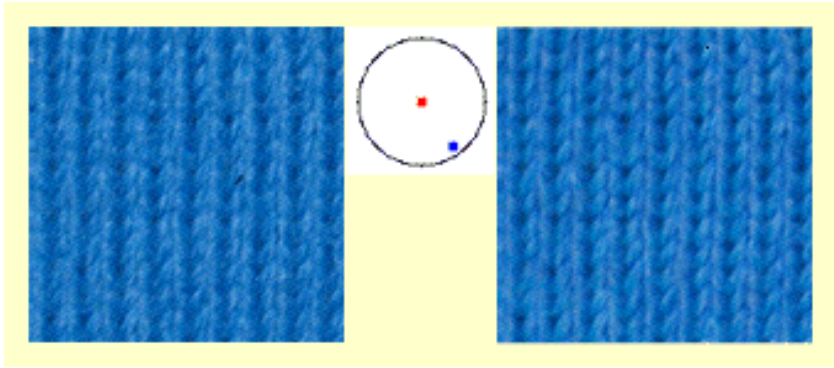
$$BTF(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \sum_{v \in N(\mathbf{v})} w_{\mathbf{x},v} RF_{\mathbf{x},v}(\mathbf{l})$$

$$RF_{\mathbf{x},v}(\mathbf{l}) \approx \rho_{d,\mathbf{x}} + \rho_{s,\mathbf{x},v}(\mathbf{l}) \cdot \sum_{i=1}^{k} s_{\mathbf{x},v}(\mathbf{l})$$

BTF Compression

- fitting analytical BRDF-models
  - Lafortune lobes

$$s(\mathbf{v},\mathbf{l}) = (\mathbf{v}^T \cdot \mathbf{M} \cdot \mathbf{l})^n$$



  - reflectance fields

$$BTF(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \sum_{v \in N(\mathbf{v})} w_{\mathbf{x},v} RF_{\mathbf{x},v}(\mathbf{l})$$

$$RF_{\mathbf{x},v}(\mathbf{l}) \approx \rho_{d,\mathbf{x}} + \rho_{s,\mathbf{x},v}(\mathbf{l}) \cdot \sum_{i=1}^{k} s_{\mathbf{x},v}(\mathbf{l})$$

BTF Compression

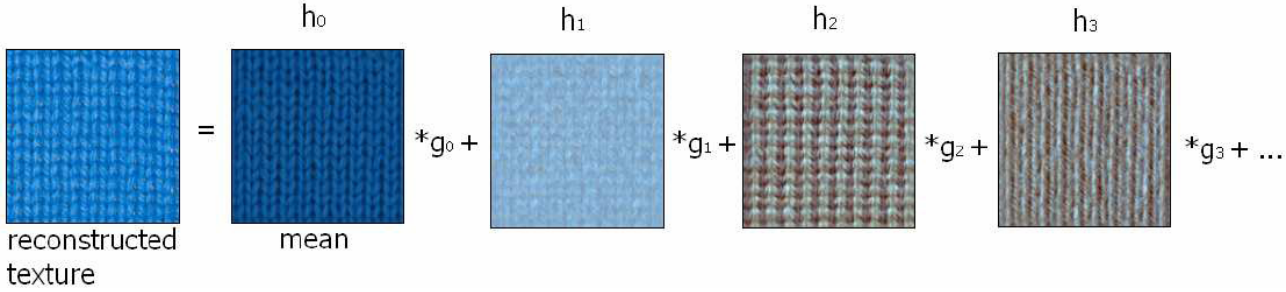
- linear basis decomposition (PCA)
  - per-texel matrix factorization

$$B_{\mathbf{x}}(\mathbf{v},\mathbf{l}) \approx \sum_{j}^{c} g_{\mathbf{x},j}(\pi_1(\mathbf{v},\mathbf{l}))h_{\mathbf{x},j}(\pi_2(\mathbf{v},\mathbf{l}))$$

  - full BTF-matrix factorization

$$BTF(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \sum_{j}^{c} g_j(\mathbf{x})h_j(\mathbf{v},\mathbf{l})$$

  - per-view factorization



BTF Rendering

- rendering

$$L_r(\mathbf{x},\mathbf{v}) = \int_{\Omega_i} \rho_{\mathbf{x}}(\mathbf{v},\mathbf{l}) L_i(\mathbf{x},\mathbf{l})(n_{\mathbf{x}} \cdot \mathbf{l}) d\mathbf{l}$$

- rendering including measured BTF

$$L_r(\mathbf{x},\mathbf{v}) = \int_{\Omega_i} BTF(\mathbf{x},\mathbf{v},\mathbf{l}) L_i(\mathbf{x},\mathbf{l})(n_{\mathbf{x}} \cdot \mathbf{l}) d\mathbf{l}$$

  - **x** is simply looked up from the BTF
  - assumption: mapping from 3D-surface to 2D spatial texture domain exists

Texture Synthesis

Texture Generating

- goal: generate texture of arbitrary size

- texture modelling, mathematical modelling
  - procedural texture modelling
  - adaptive texture models
- texture sampling
  - generating from a given texture sample

Procedural Textures

- analytical scalar function of (x,y,z) coordinates
- texturing – evaluation of function on object surface
  - ray-tracing - 3D intersection point with surface
- textures of natural objects
  - similarity between different patches
    - repetitiveness, coherence
  - similarity on different resolution scales
  - not completely identical
    - additional disturbances, turbulence, noise
- mimics statistical properties of natural textures
- empirical approach

Examples of Procedural Textures

- marble

$$f(x,y,z) = \sin^n(x + t(x,y,z))$$

- wood

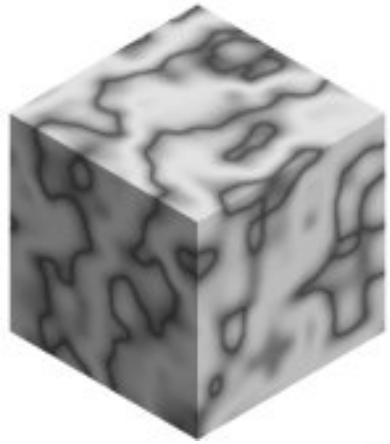
$$f(x,y,z) = x^2 + y^2 + n(x,y,z)$$

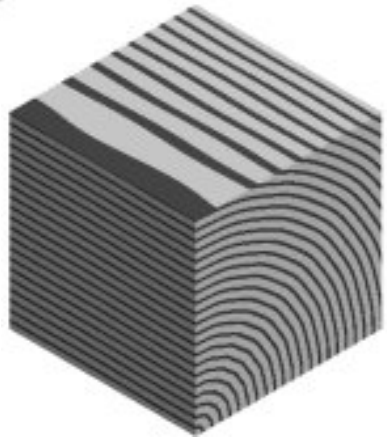
- clouds

$$f(x,y,z) = \sum_{i=1}^{4} 2^{-i} n(2^i x, 2^i y, 2^i z)$$

- fire

$$f(x,y,z) = z + \sum_{i=1}^{4} 2^{-i} n(2^i x, 0, 2^i z + \phi)$$





Adaptive Texture Models

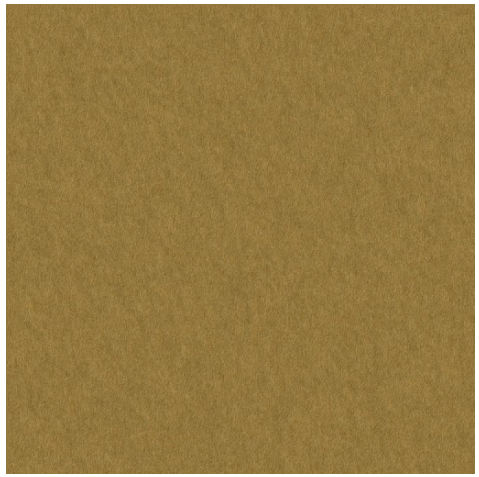
- procedural form
- Gaussian Markov Random Fields
  - flexible for natural or artificial texture simulation
  - represents high frequencies well
- extreme compression – only few parameters have to be stored
- unknown relation between parameters and visual appearance of texture

Adaptive Texture Models Results

- 3D Causal Autoregressive Model

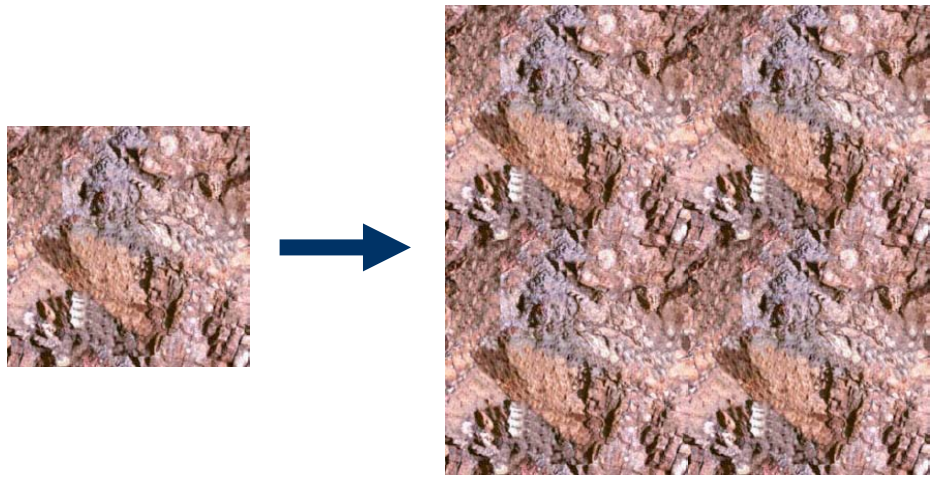







Sampling-based Texture Synthesis

- given a small texture patch, fill an arbitrary region



- tiling
  - discontinuities along edges?
  - artificial appearance – completely regular

Sampling Versus Modelling

- sampling
  - given texture
  - realistic synthetic texture
  - limited resolution, aliasing
  - high memory consumption
- adaptive models
  - procedural form
- modelling
  - unlimited resolution
  - low memory consumption
  - cannot model given texture
  - unknown relation between parameters and texture appearance

Sampling-based Texture Synthesis

- given texture *I*, generate a texture *J* which
  - looks like the same texture
  - has no obvious copying or tiling from *I*
  - difference between *I* and *J* should be the same as *I* “differs from itself”
- intelligent sampling approaches
  - preserve stochastic properties, no regularity (tiling), no visible seams
  - analytical part, synthesis part
    - not separated, partially separated of fully separated
    - if separated, real-time synthesis is possible

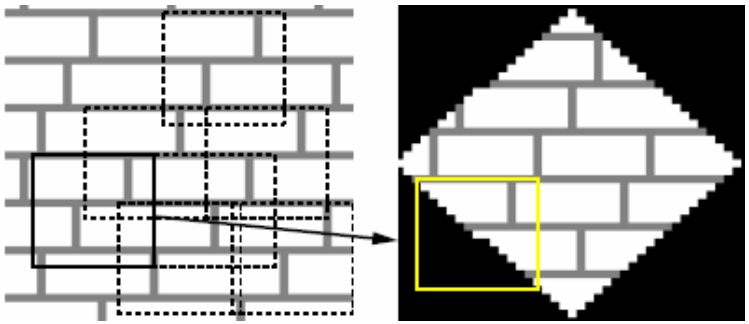
Intelligent Sampling Approaches

- different approaches
  - multiresolution pyramids
  - pixel-by-pixel synthesis
  - multiresolution pixel-by-pixel
  - patch-based synthesis
  - advanced tiling techniques
- no universal method exists
  - stochastic textures
  - near-periodic textures
  - BTF textures

Texture Synthesis Algorithms

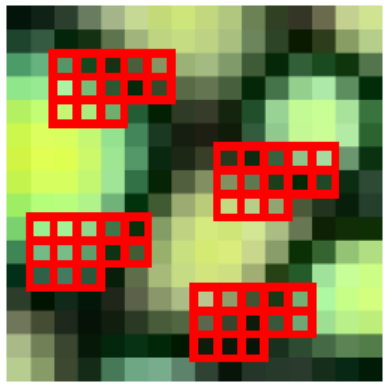
Non-parametric Sampling

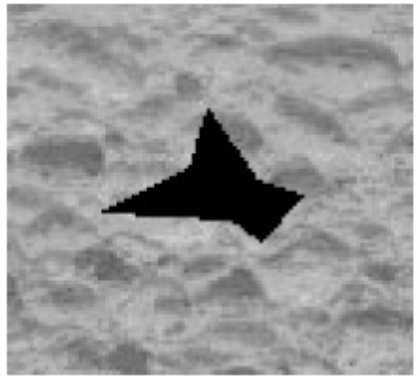
- Efros, Leung, 1999
- properties
  - MRF: probability of synthesized pixel depends on its neighborhood; independent of the rest of image
  - neighborhood
    - square window, size of window – free parameter
    - causal



Non-parametric Sampling

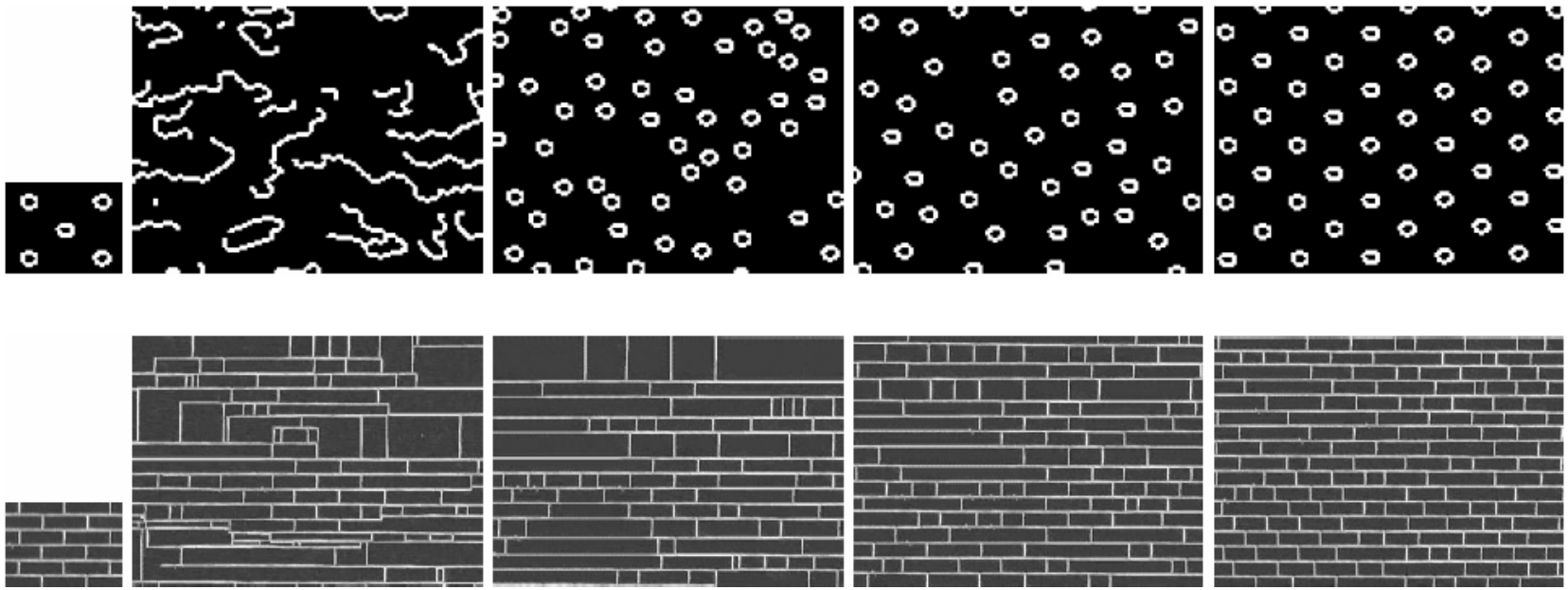
- neighborhood distance: $d=d_{SSD}*G$
- possible pixels: $dist<(1+\varepsilon)*d(N,N_{best})$
- synthesis: outward from 3-by-3 seed randomly taken from input sample
- hole filling: from the edges of the hole



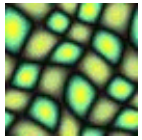


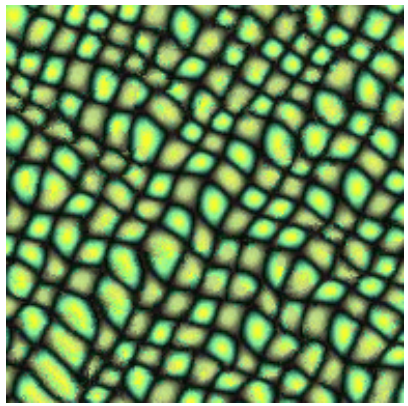
Non-parametric Sampling Results

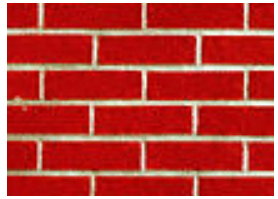
- size of neighborhood – 5, 11, 15, 23 pixels

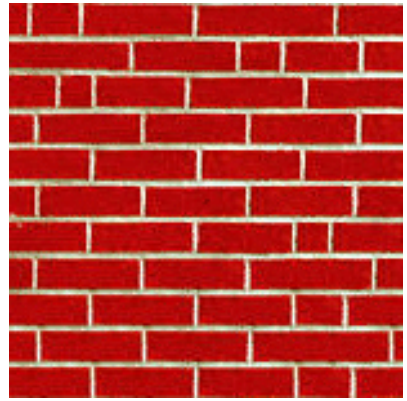


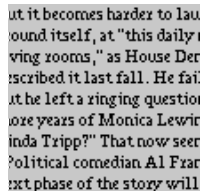
Non-parametric Sampling Results





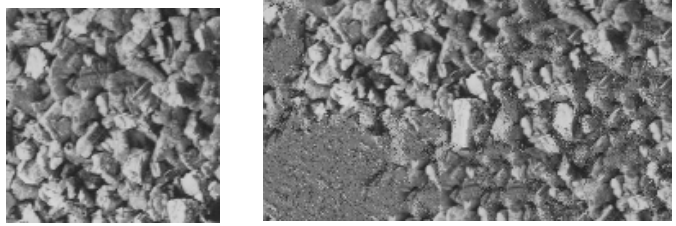






Non-parametric Sampling Failures

growing garbage



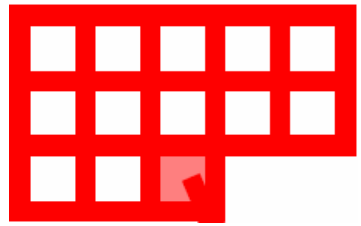
verbatim copying

Non-parametric Sampling

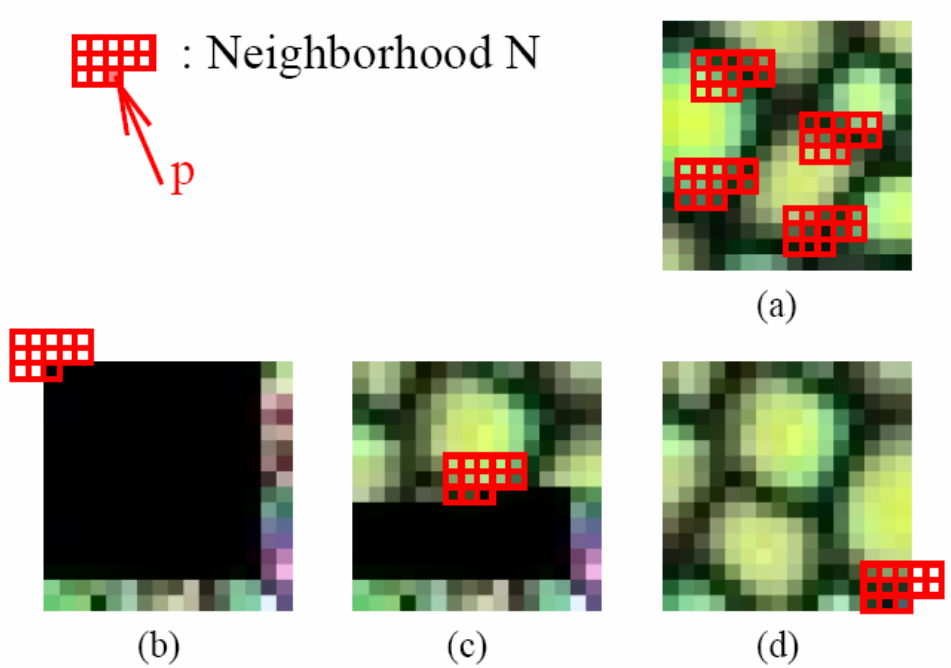
- size of neighborhood
  - largest feature size
  - the only parameter specified by user
- not separated analytical part from synthesis step
  - each pixel – full search
  - k-NN search in $3*(w^2/2-1)$ dimensions
  - very slow synthesis

Fast Texture Synthesis using TSVQ

- Wei, Levoy, 2001
- algorithm
  - initialize *J* to noise
  - synthesize new pixels in scanline order
  - select new pixel from *I* that has the closest matching neighborhood to *J*
  - in multiresolution case, look at feature vectors instead of neighborhood



Fast Texture Synthesis using TSVQ

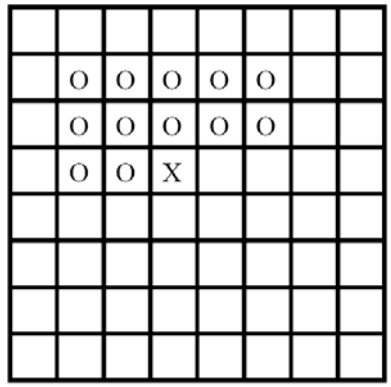


- scanline synthesis with neighborhoods
  - (a) input texture
  - (b) start
  - (c) midway point
  - (d) end

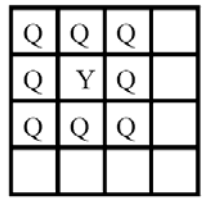
Fast Texture Synthesis using TSVQ

- multiresolution approach
  - search for X – neighborhood vector is constructed from O's, Q's and Y
  - small neighborhoods can be used
  - 5x5 top level,

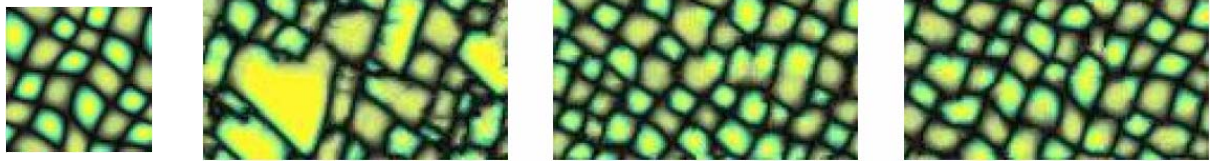
    1, 2, 3 levels



L

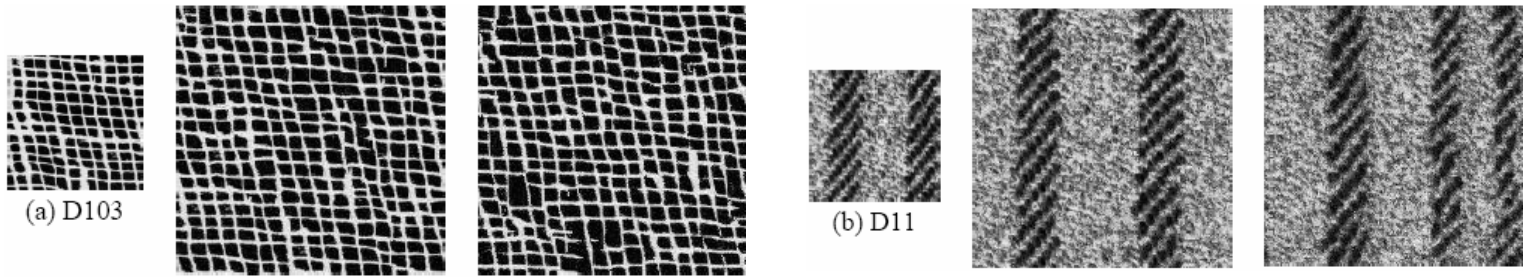


L+1



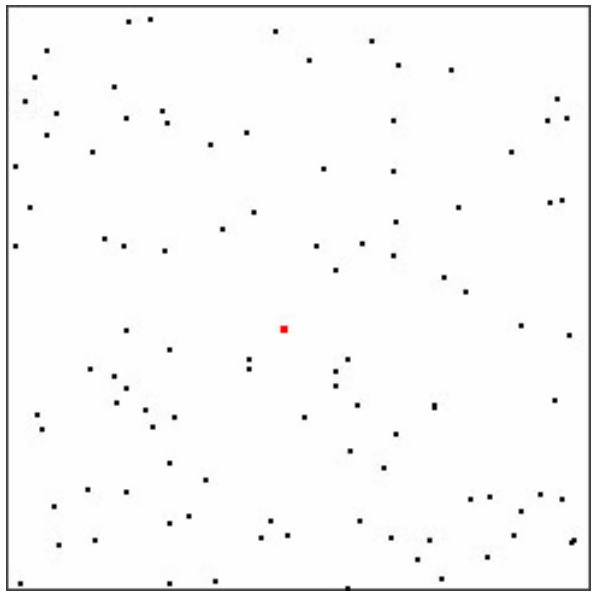
Fast Texture Synthesis using TSVQ

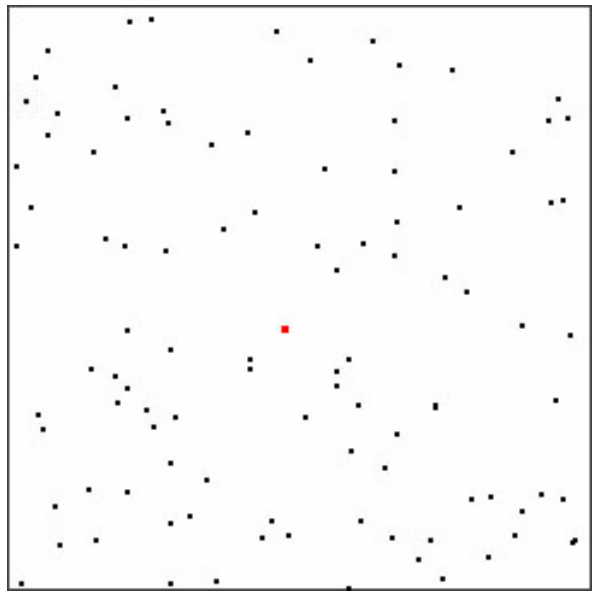
- TSVQ acceleration
  - exhaustive search – very slow
  - N(p) – point in multi-dimensional space
    - neighborhood matching = nearest point search problem
    - $\{N(p_i)\}$ from each $G_a(L)$ as the training data
    - generate tree structure codebooks
    - synthesis – time complexity $O(\log N_L)$ of search
  - adds additional unpredictability



Fast Texture Synthesis using TSVQ

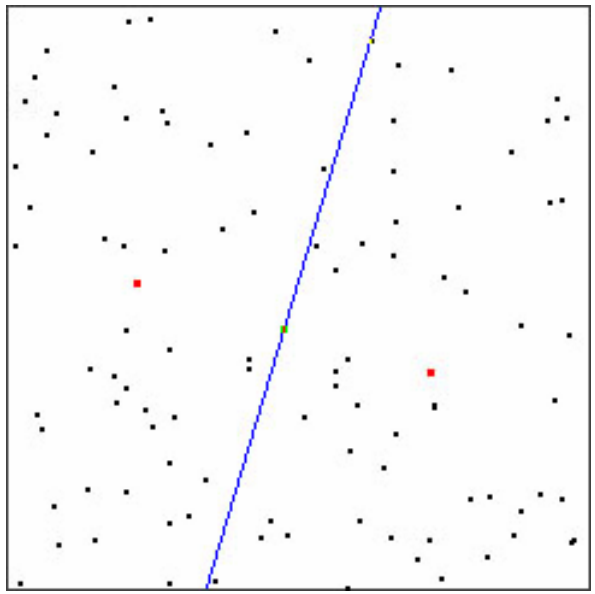
- TSVQ demonstration - initialization

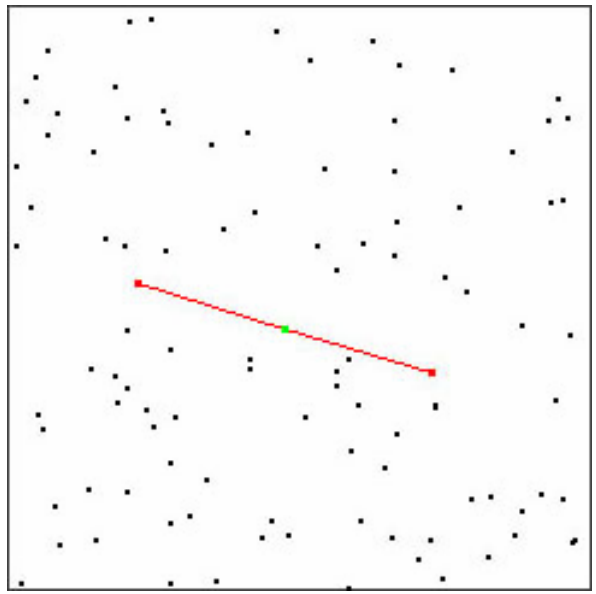




Fast Texture Synthesis using TSVQ

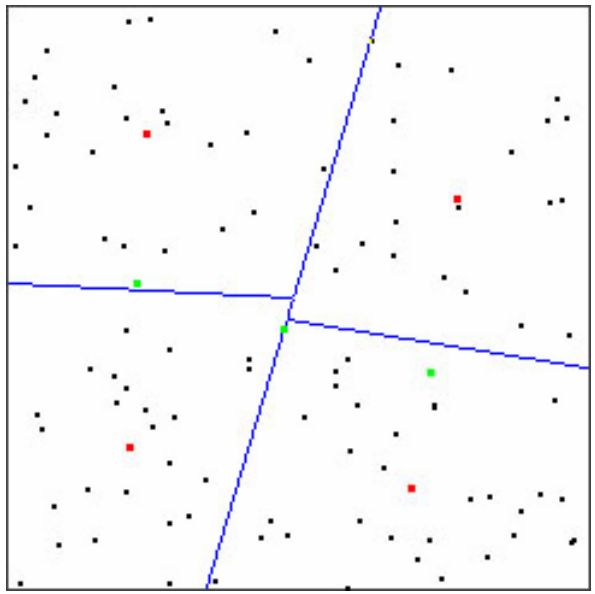
- TSVQ demonstration – 1. step

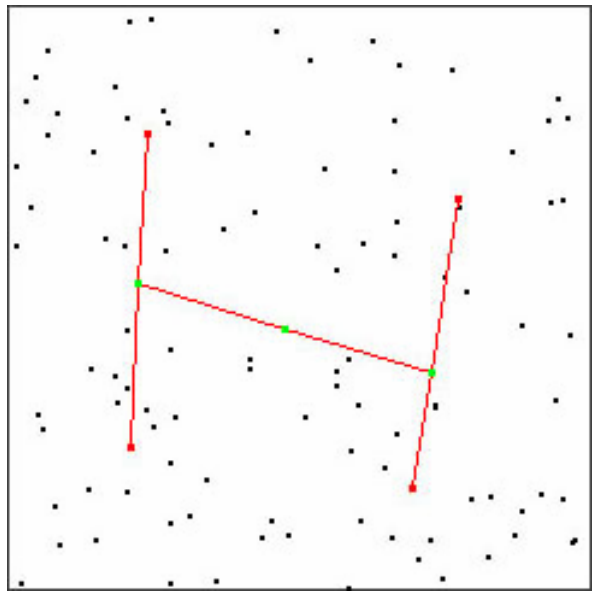




Fast Texture Synthesis using TSVQ

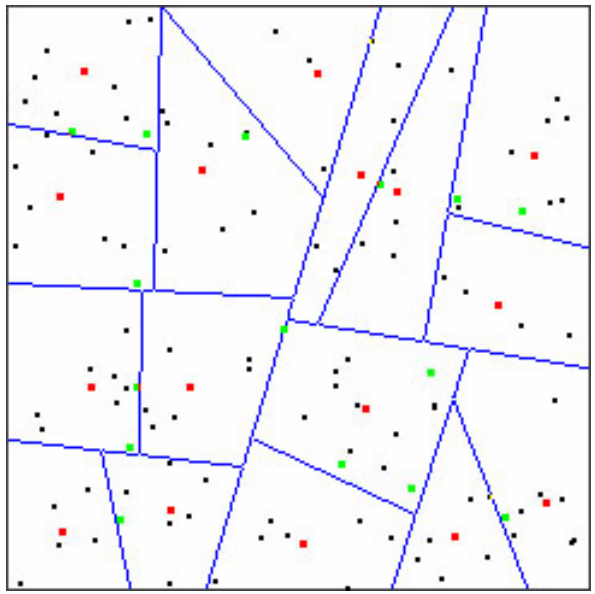
- TSVQ demonstration – 2. step

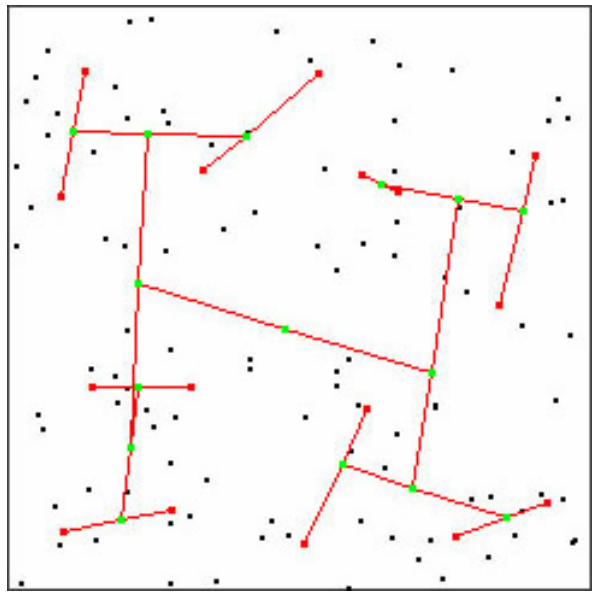




Fast Texture Synthesis using TSVQ

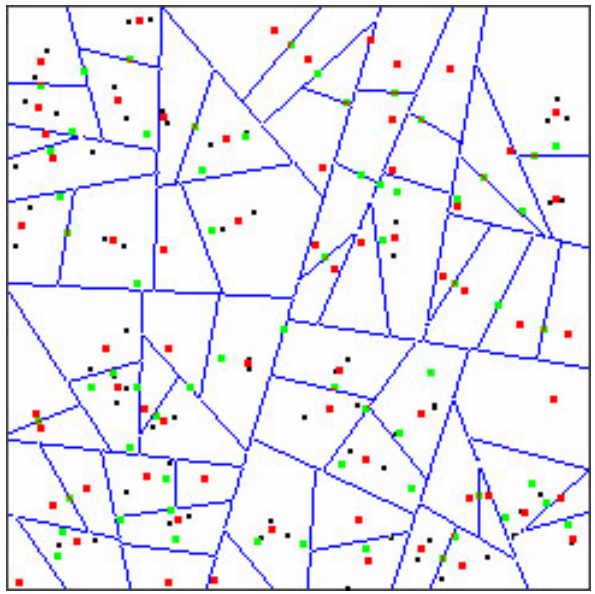
- TSVQ demonstration – 4. step

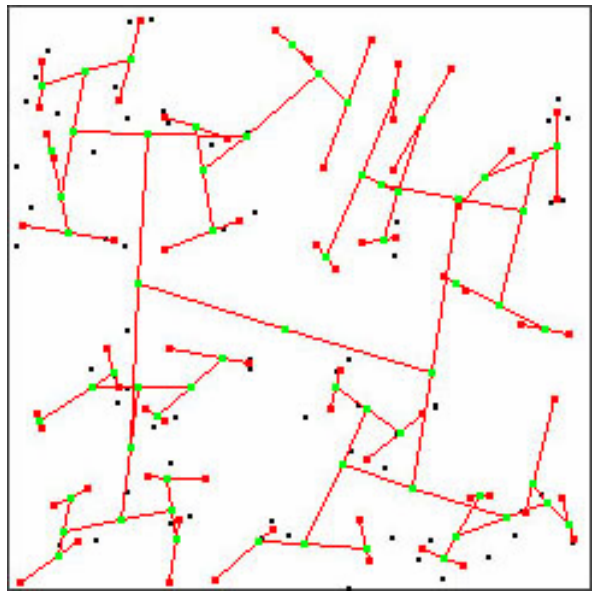




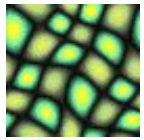
Fast Texture Synthesis using TSVQ

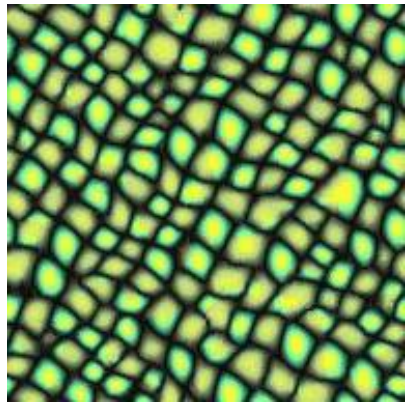
- TSVQ demonstration – 6. step





Fast Texture Synthesis using TSVQ Results





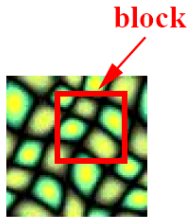
Fast Texture Synthesis using TSVQ

- parameters
  - neighborhood size and number of levels
  - TSVQ – number of codewords
- not fully separated analytical part from synthesis step
  - analysis
    - codebook construction
  - synthesis
    - each pixel – search using TSVQ – logaritmic complexity
    - much faster than NPS

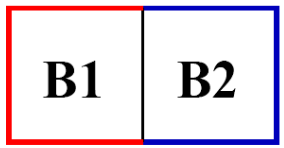
Image Quilting

- Efros, Freeman, 2001
- complex textures – very few pixels have a choice of values to assign
- pasting overlapping patches of the input texture into a rectangular grid and fixing the seams
- algorithm:
  - initialize *J* to empty
  - copy new patches of certain size in scanline order, with fixed overlap width
  - randomly select new patch with overlap error less than given threshold
  - crop patch by minimal error boundary cut

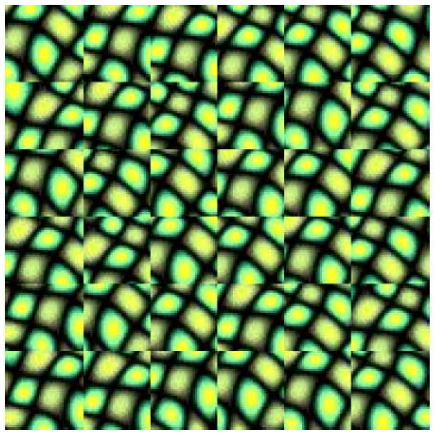
Image Quilting

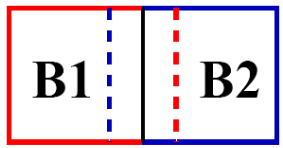


input texture

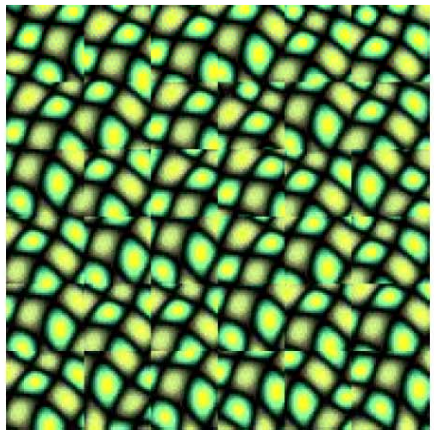


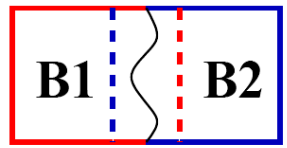
random placement of blocks





neighboring blocks constrained by overlap





minimum error boundary cut

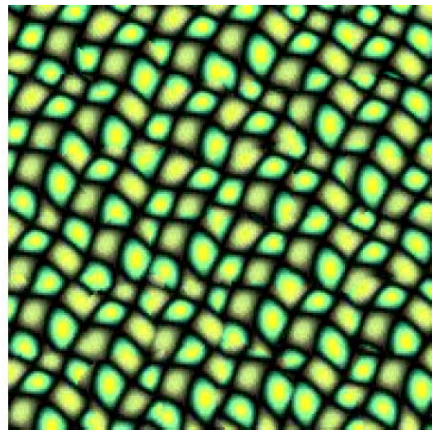


Image Quilting

- minimum error boundary cut

  $E_{i,j} = e_{i,j} + \min(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1})$

  – dynamic programming
- overlap error – L2 norm on pixel values
- width of overlap – 1/6 of the size of the block
- matching block tolerance – within 0.1 times error of the best matching block

Image Quilting

- demo of the method

Image Quilting Results

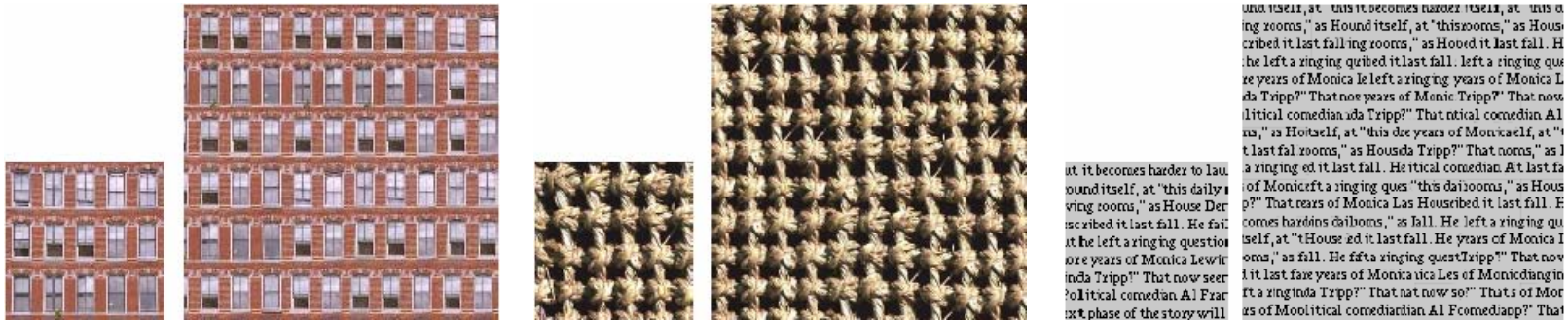


Image Quilting

- parameters
  - size of the square block
  - width of the overlap
- not separated analytical part from synthesis step
  - exhaustive search of best patch
  - very slow
- possible improvement
  - speed - search acceleration via TSVQ
  - quality – A* or Dijkstra's algorithm

General Failures

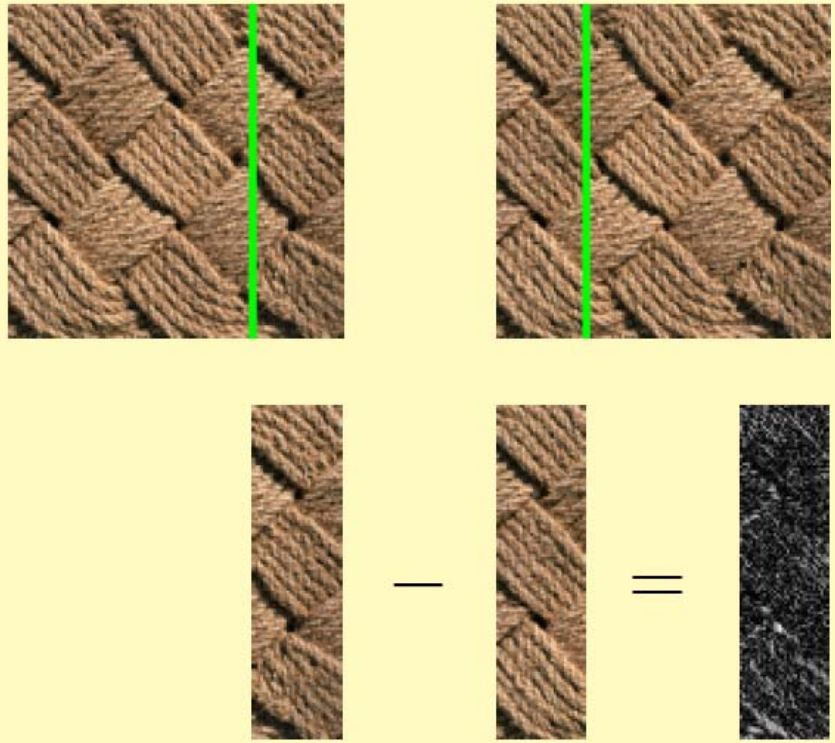
- non-uniform image intensity
  - algorithm slips into some region
- geometric distortion
  - perspective
- non-representative texture sample
  - low frequencies not captured

BTF Roller

- basic method
  - maximal toroidal tile
  - tiling
- advanced method
  - automatic tile size estimation
  - tiling of multiple tiles
  - BTF

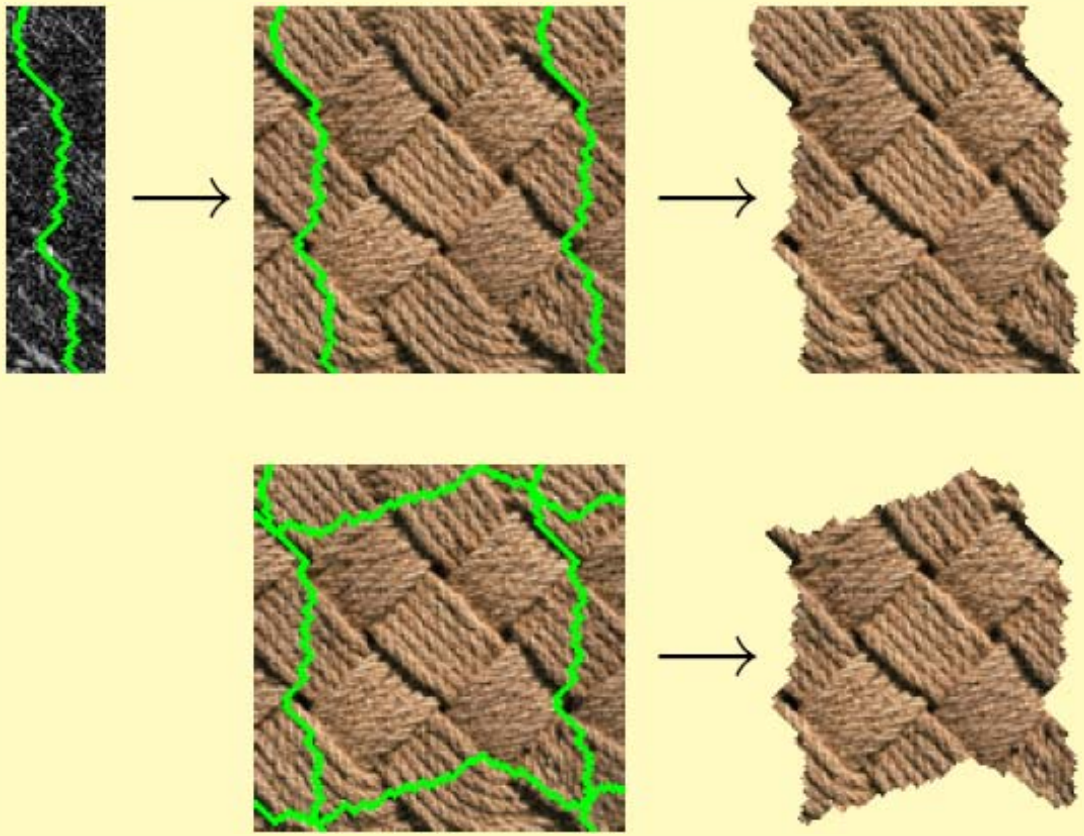
Toroidal Tile

- overlap



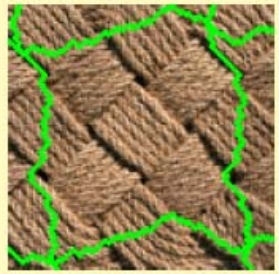
Toroidal Tile

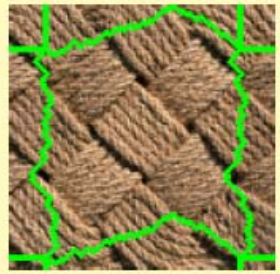
- overlap
- optimal cut

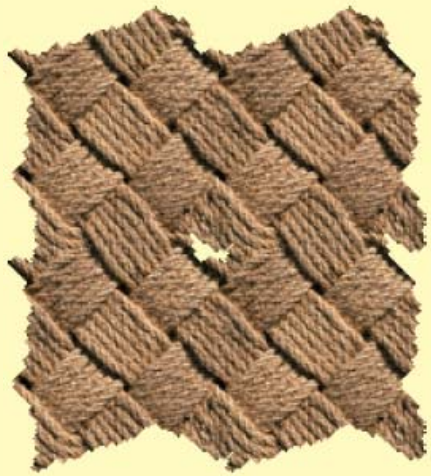


Toroidal Tile

- overlap
- optimal cut
- corner treatment

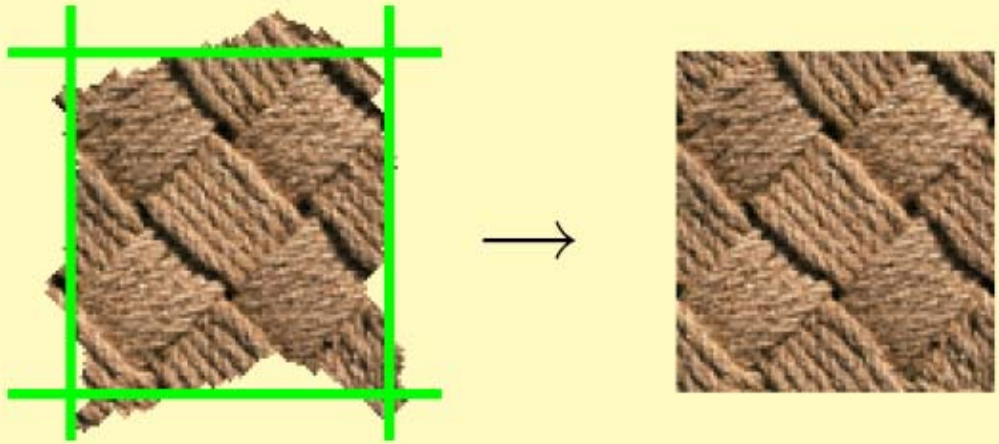






Toroidal Tile

- overlap
- optimal cut
- corner treatment
- rectangular tile

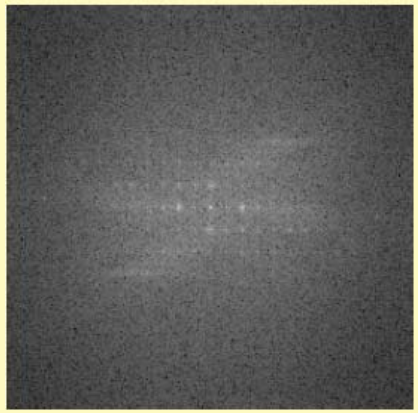


Multiple Tiles

- structure of synthetic texture
  - avoid artificial regularity
  - preserve original structure
- tile filling replacement
- optimal tile size estimation
  - number of tiles *k*
  - frequency content

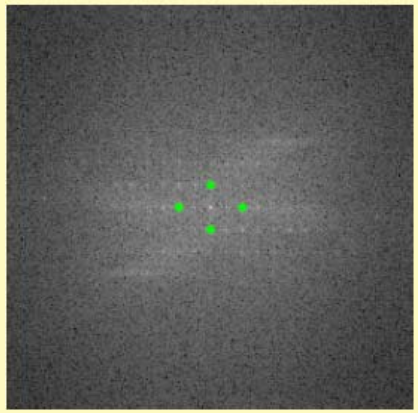
Optimal Tile Size Estimation

- Fourier transform
  - texture sample N × M



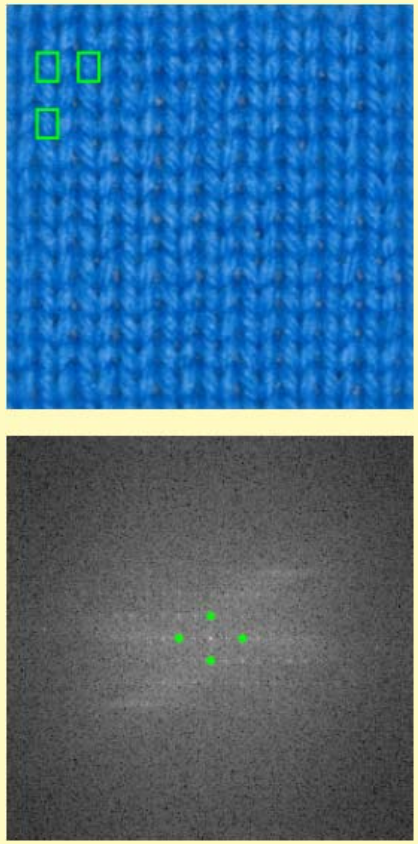
Optimal Tile Size Estimation

- Fourier transform
  - texture sample N × M
  - f_{row}, f_{col}



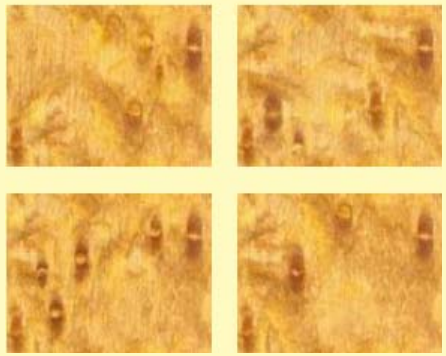
Optimal Tile Size Estimation

- Fourier transform
  - texture sample N × M
  - f_{row}, f_{col}
- texture analysis
  - n_{row}, n_{col}



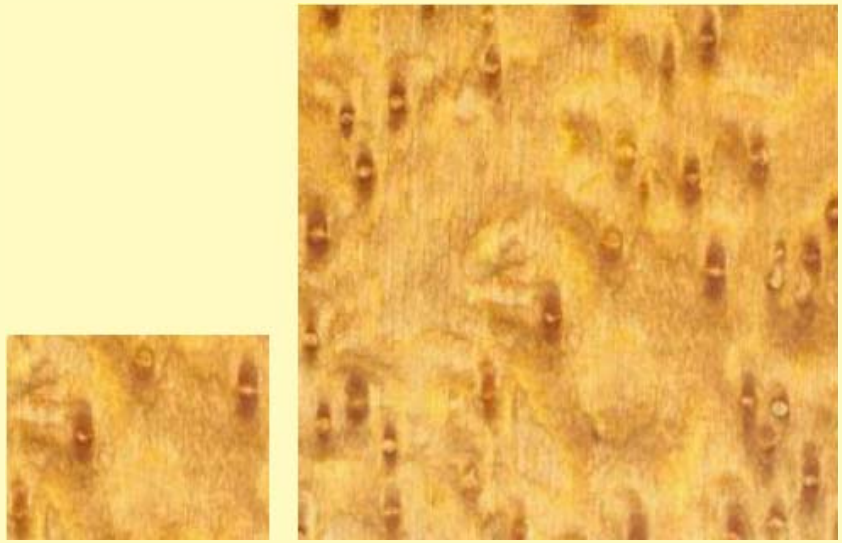
Optimal Tile Size Estimation

- tile N_t x M_t
  - as large as possible
  - $N_t \approx i\, n_{row}$
    $M_t \approx j\, n_{col}$
  - $N_t\, M_t \approx (1/k)NM$



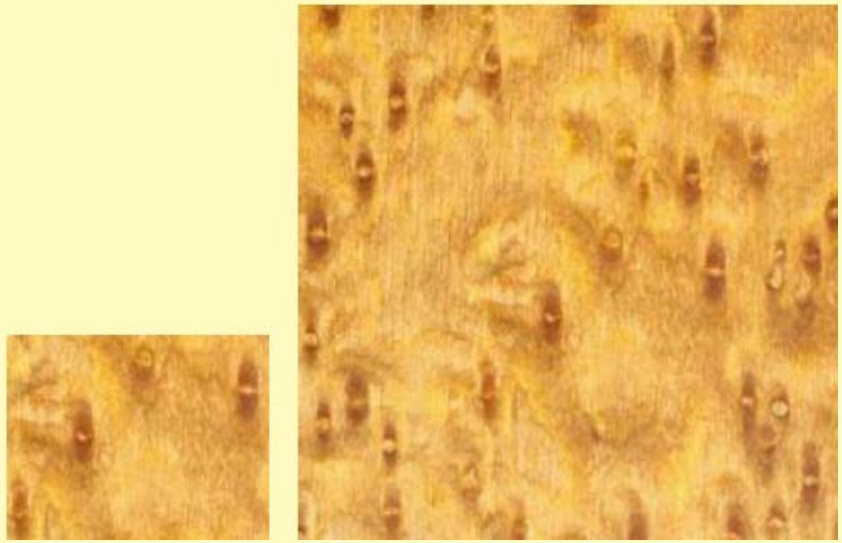
Tile Filling Replacement

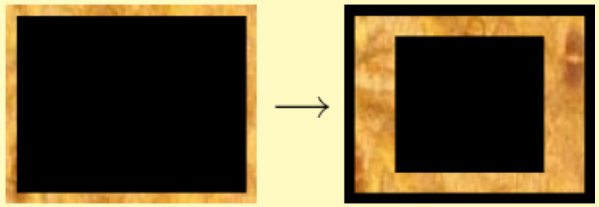
- preserve seamless elargement



Tile Filling Replacement

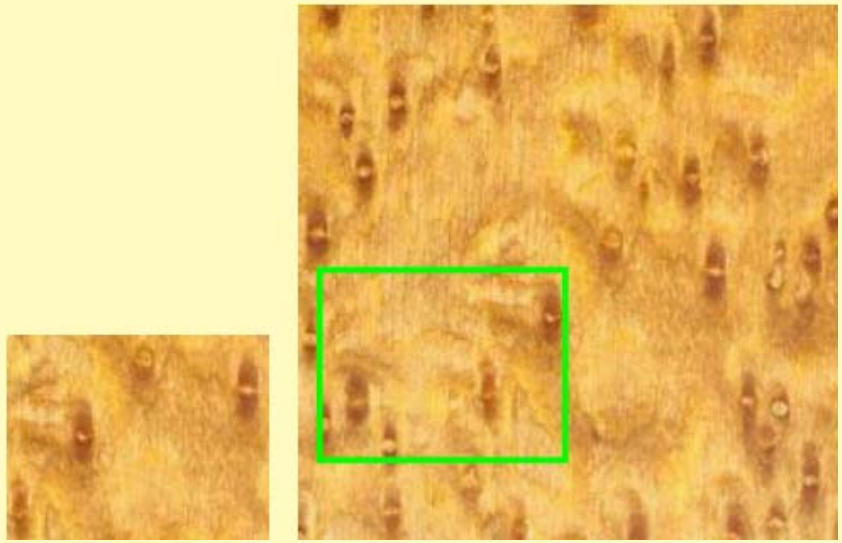
- preserve seamless elargement
- new tile filling
  - search





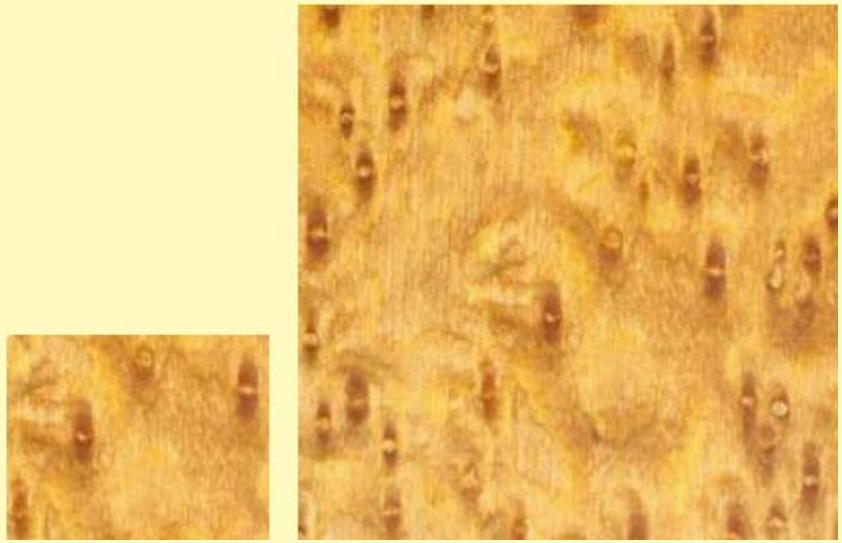
Tile Filling Replacement

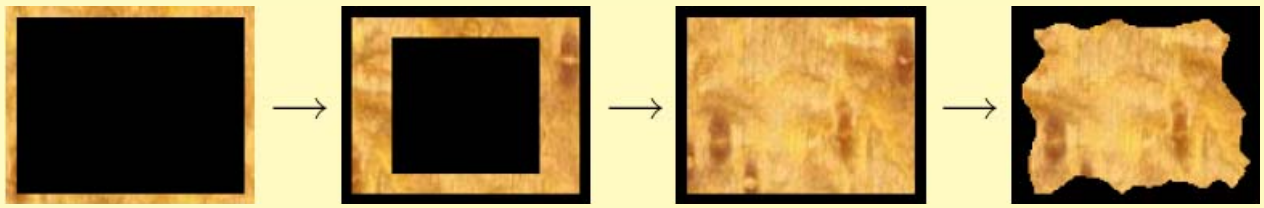
- preserve seamless elargement
- new tile filling
  - search



Tile Filling Replacement

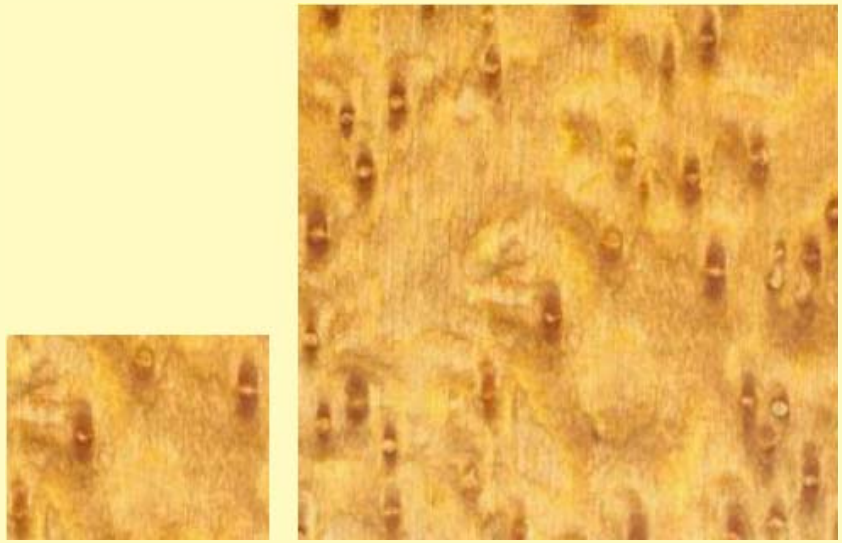
- preserve seamless elargement
- new tile filling
  - search
  - crop





Tile Filling Replacement

- preserve seamless elargement
- new tile filling
  - search
  - crop
- new tile



Roller Algorithm and BTF Synthesis

smooth textures

analysis

- estimates
- tile
- tile filling search
- new tiles

synthesis

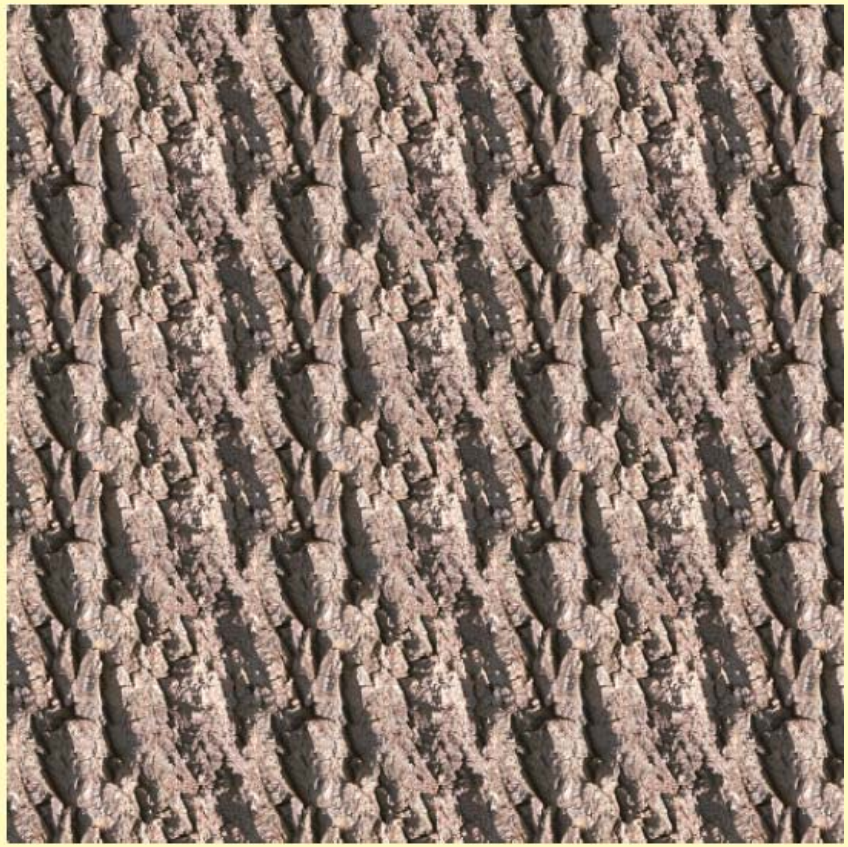
BTF textures

analysis

- estimates
  - perpendicular illumination
- tile
  - all illumination angles
- tile filling search
  - perpendicular illumination
- new tiles
  - all illumination angles

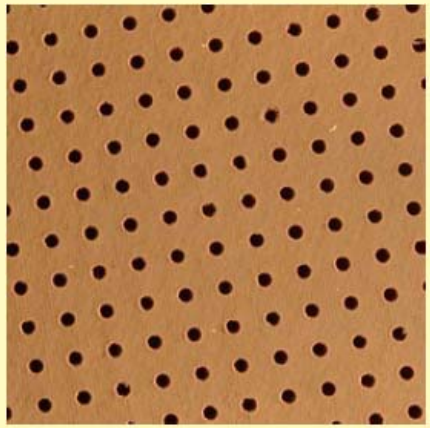
synthesis

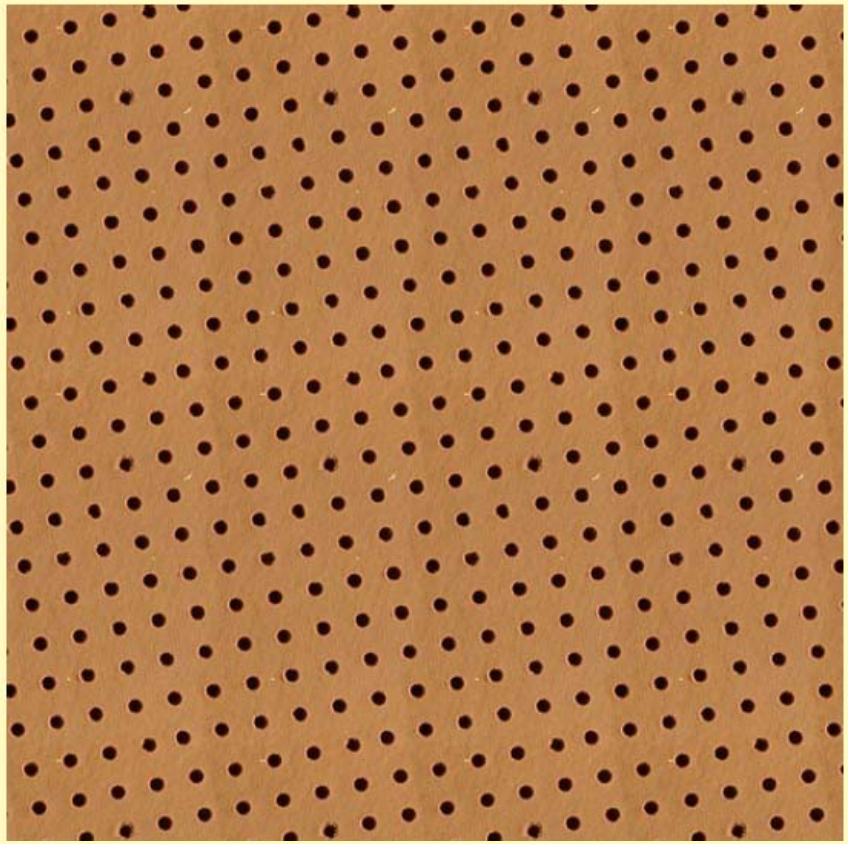
Results – 1 Tile



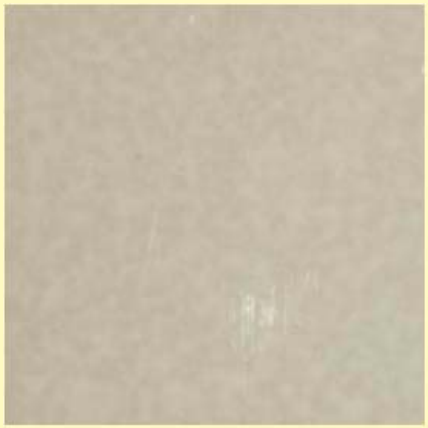
Results – 1 Tile

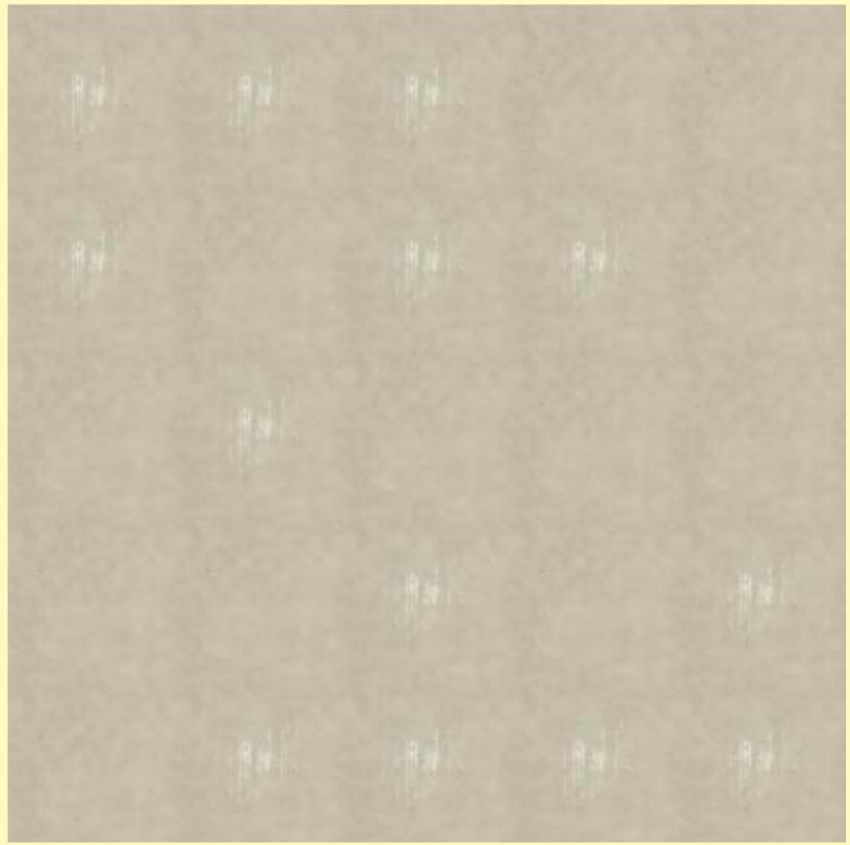
Results – 1 Tile



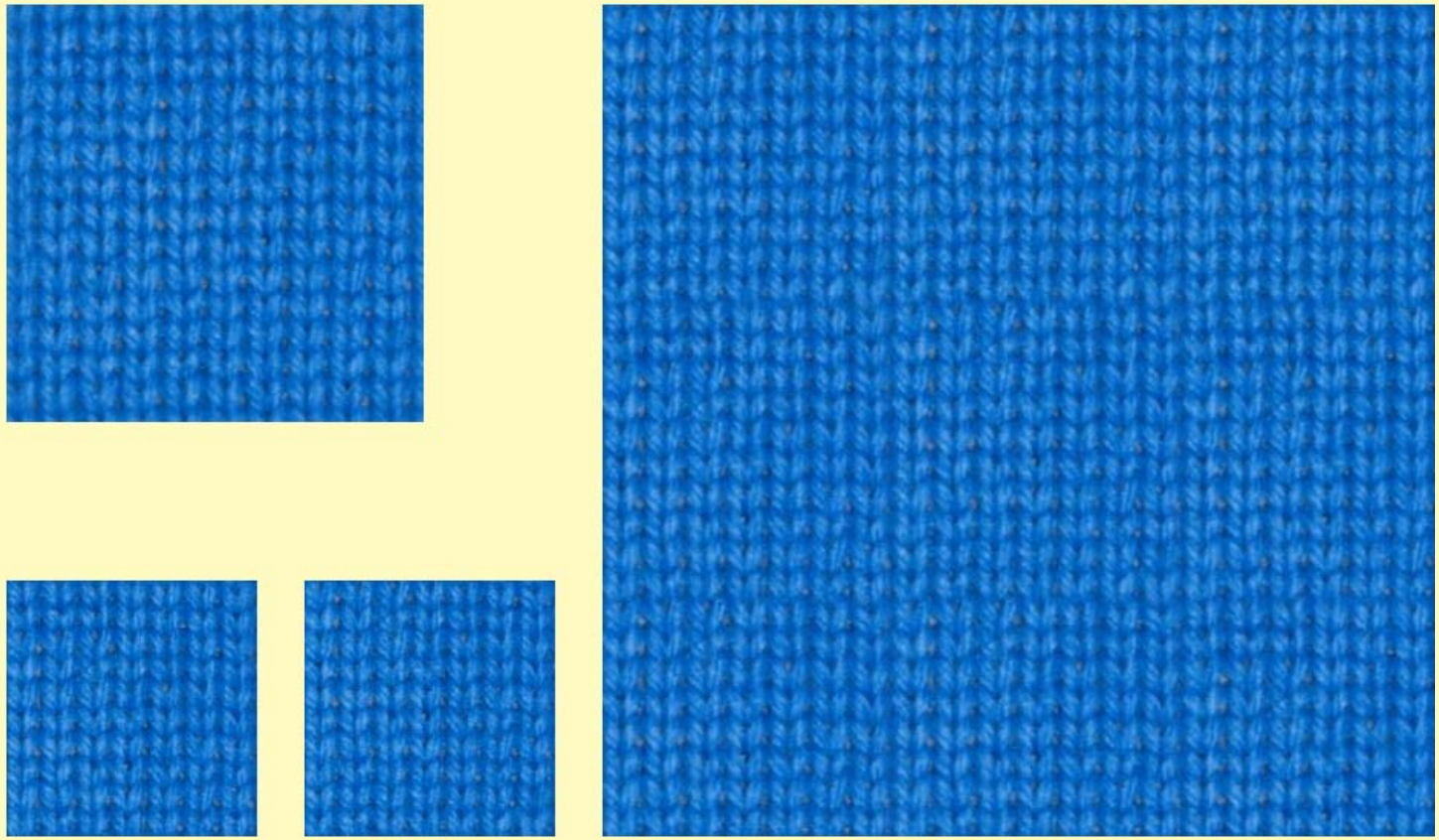


Results – 2 Tiles

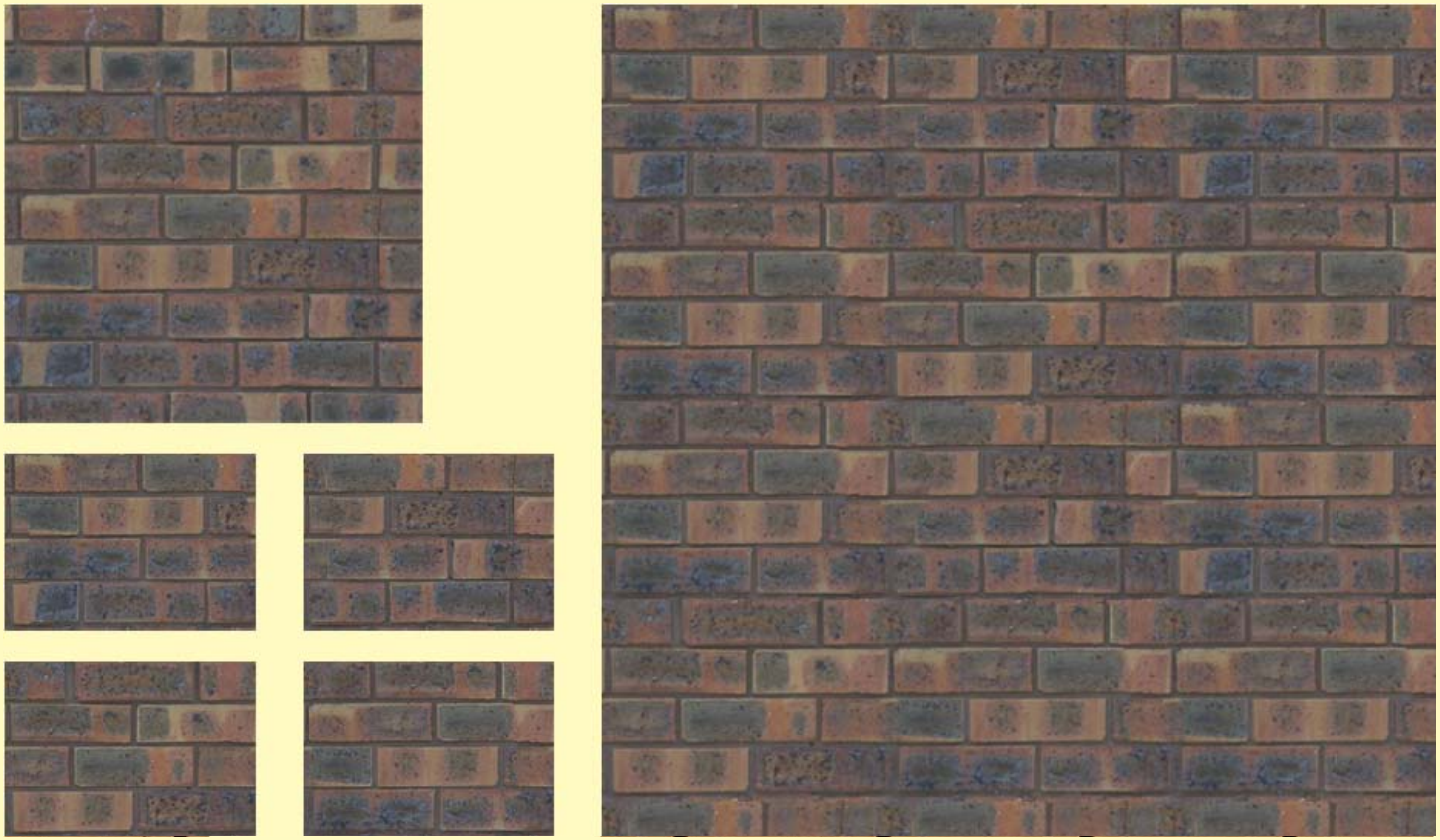




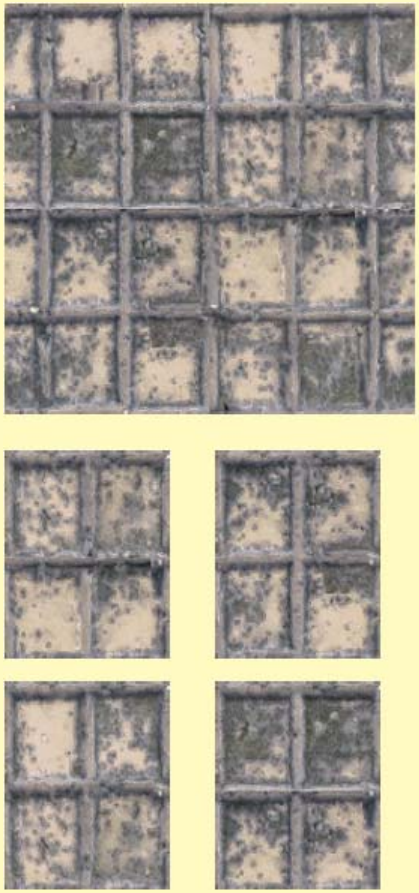
Results – 2 Tiles

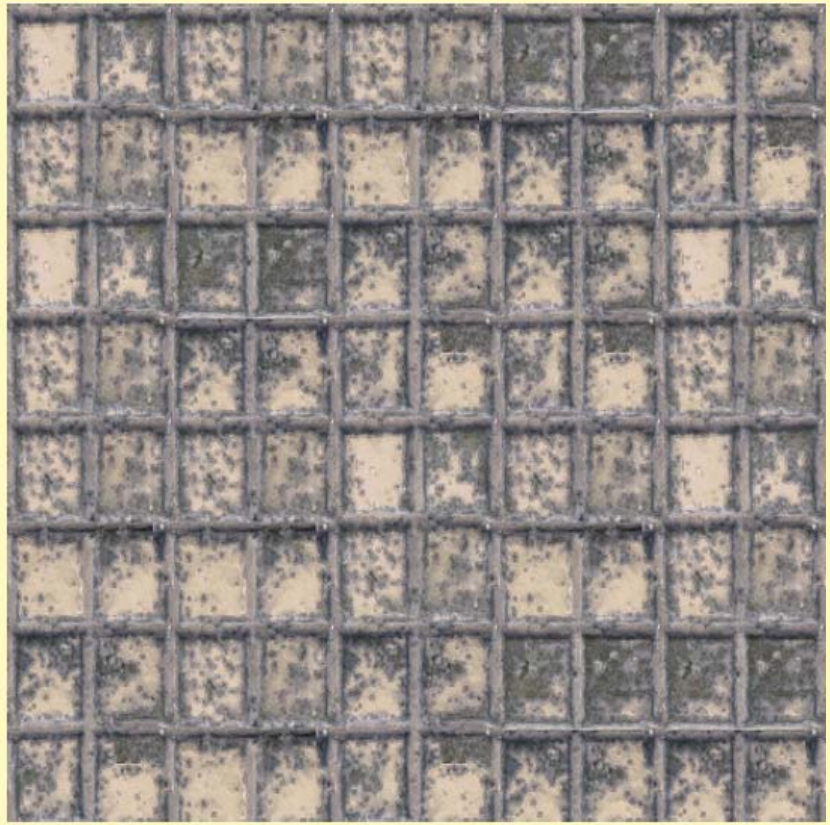


Results – 4 Tiles



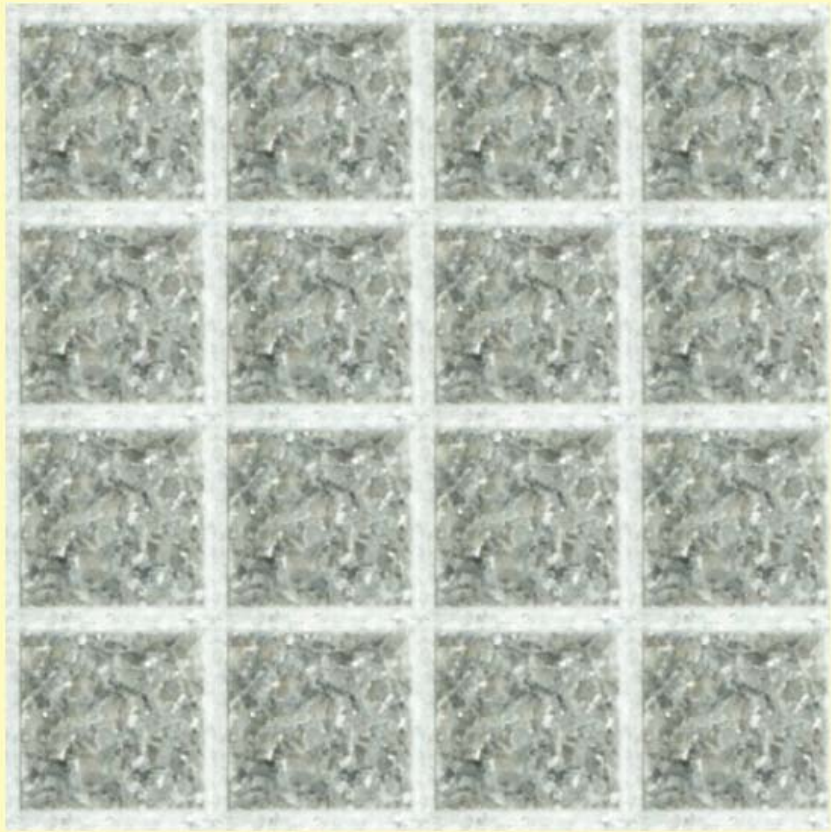
Results – 4 Tiles





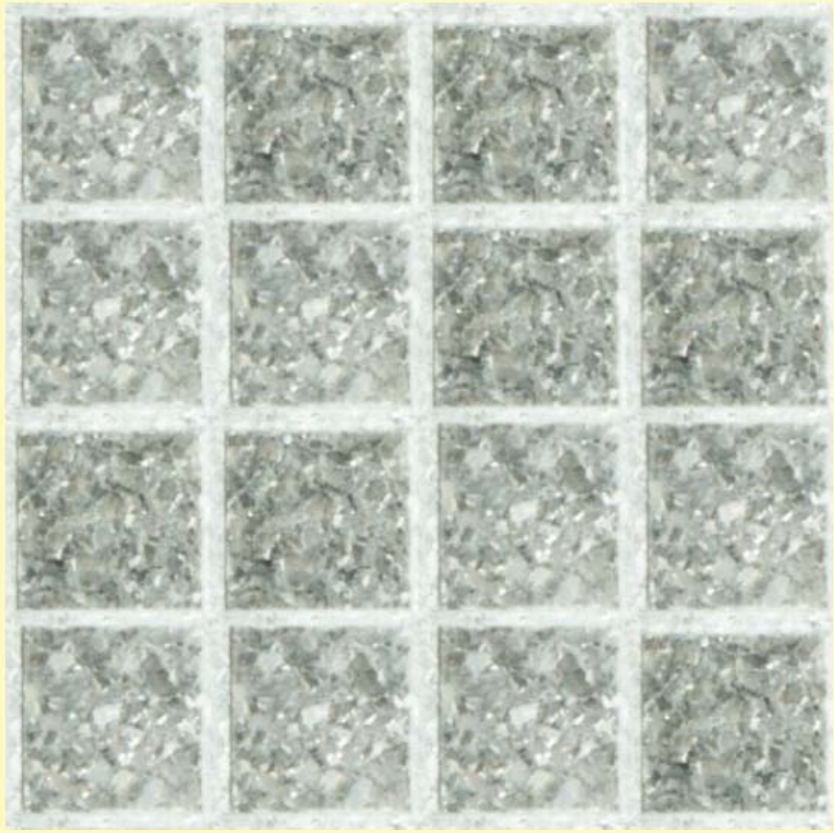
Results – 1, 2, 4 Tiles

1 tile

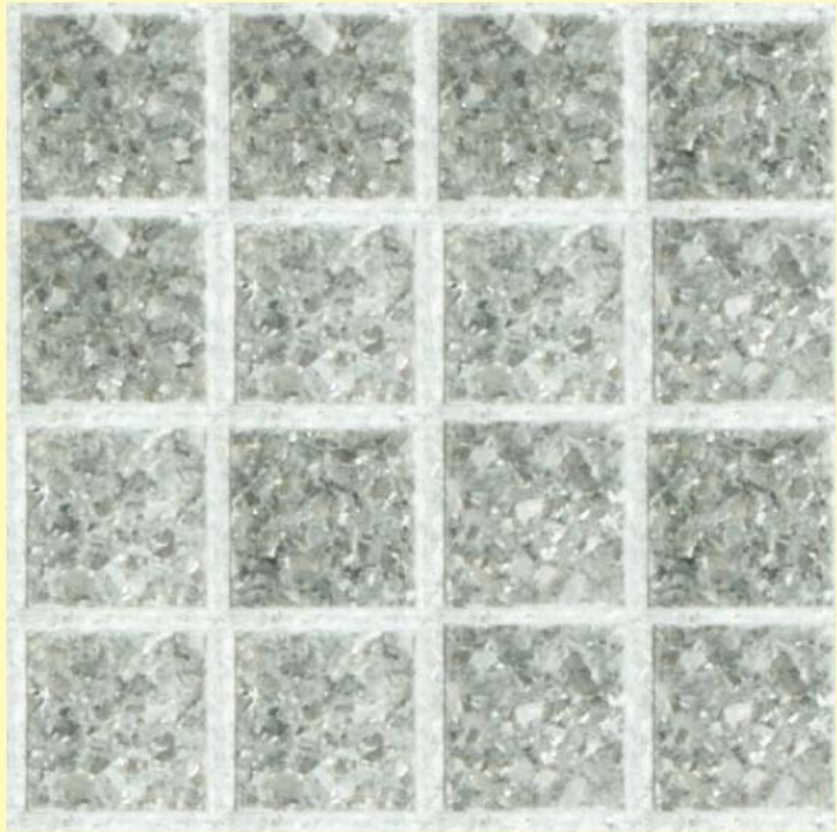


Results – 1, 2, 4 Tiles

2 tiles



Results – 1, 2, 4 Tiles



4 tiles

Results – 1, 2, 4 Tiles

1 tile

Results – 1, 2, 4 Tiles

2 tiles

Results – 1, 2, 4 Tiles

4 tiles

BTF Synthesis

- wool
- wood

Roller × Image Quilting

Roller

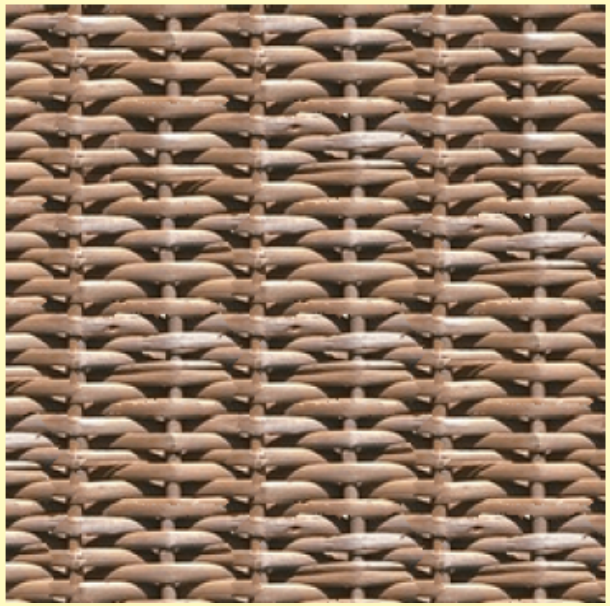
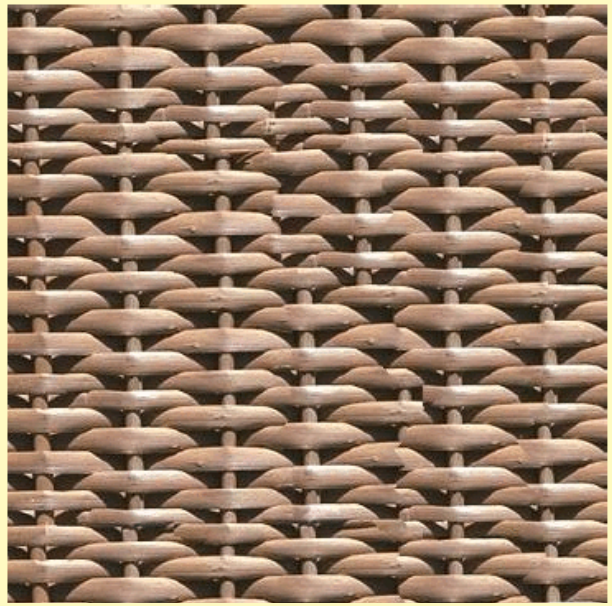


Image Quilting



Roller × Image Quilting

Roller

Image Quilting

Roller × Image Quilting

Roller

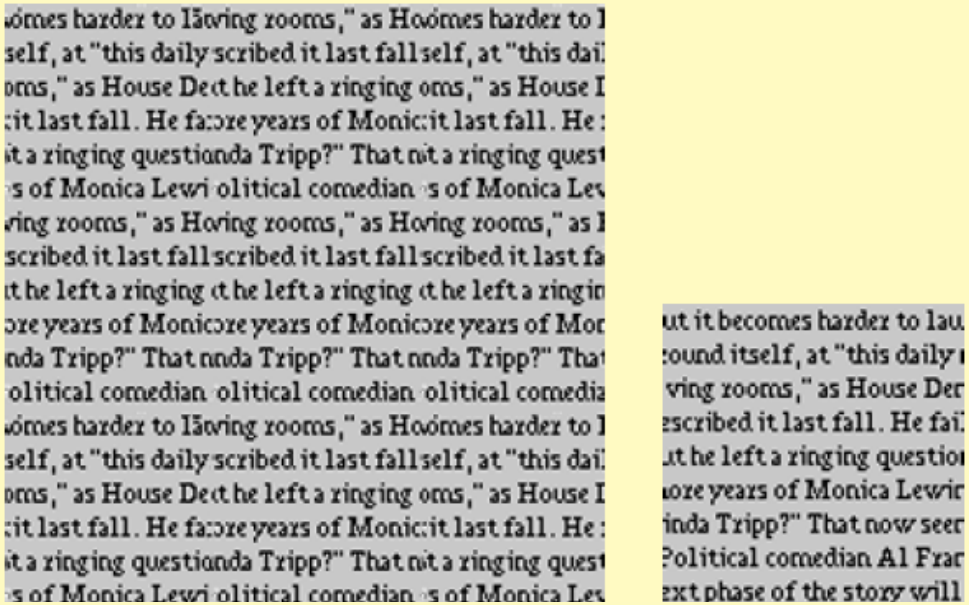
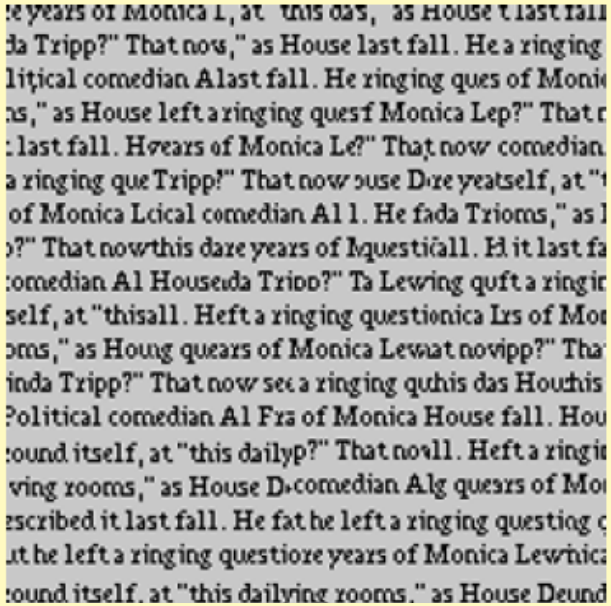


Image Quilting



Roller × Image Quilting

Roller

Image Quilting

Roller × Image Quilting

Roller

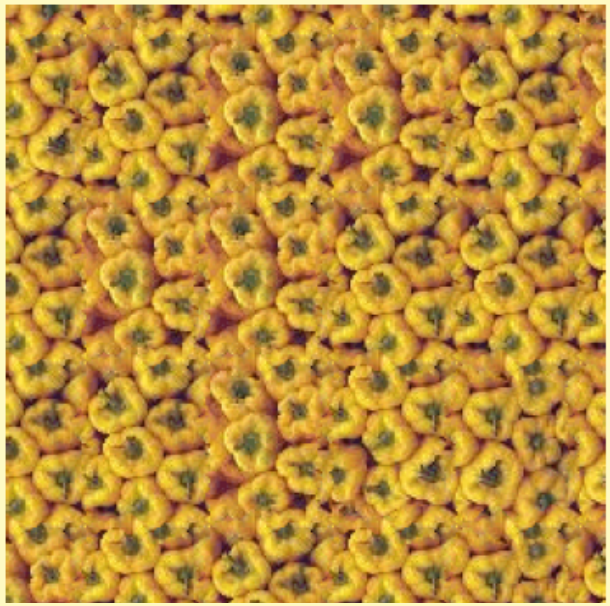
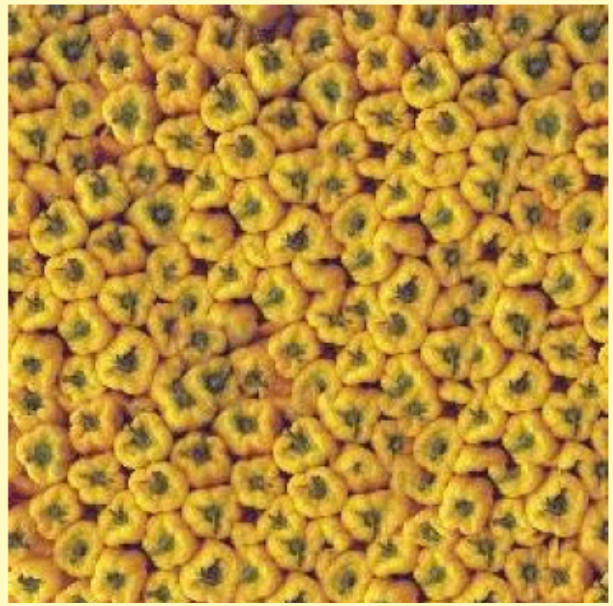


Image Quilting



Roller × Image Quilting

Roller

Image Quilting

Failures

- non-uniform image intensity

Roller

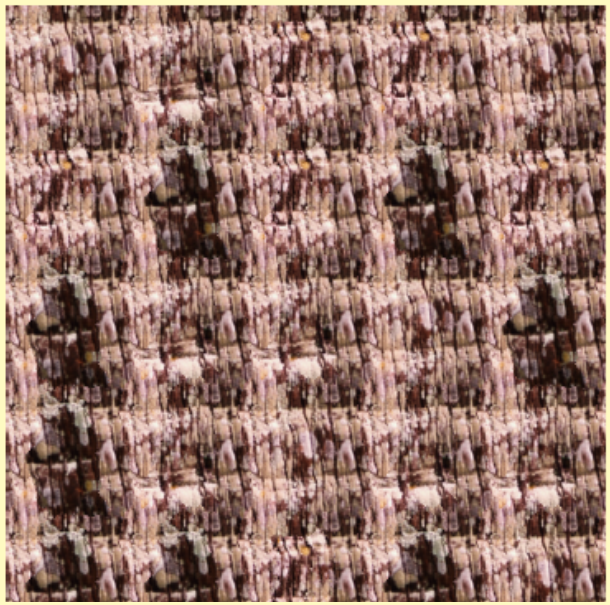


Image Quilting

Failures

- non-representative sample

Roller

Image Quilting

Failures

- perspective distortion

Roller

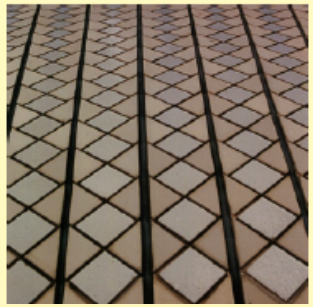
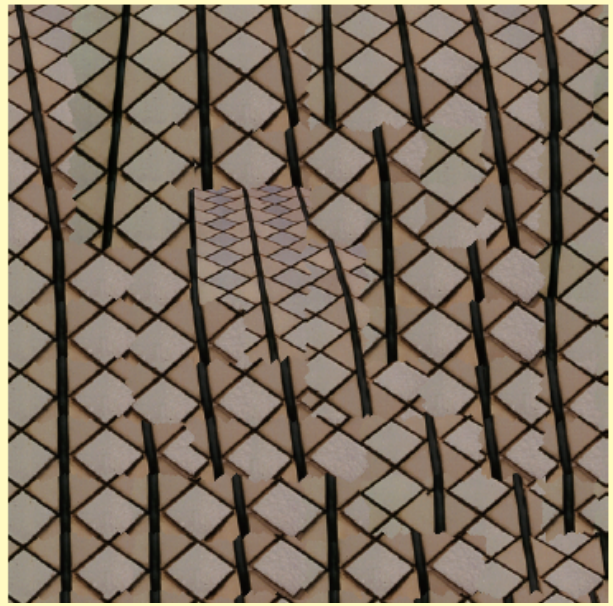


Image Quilting



Summary of Roller method

- automatic algorithm
  - the only parameter - *k* tiles
- real-time synthesis
- analysis time consumption
  - not important
  - 256×256, 1 tile … 36s
- very good results
- fast GPU implementation possible

Application

- smooth textures
- BTF textures
- alternative types of textures
  - eigen-textures
- parametric spaces of reflectance models