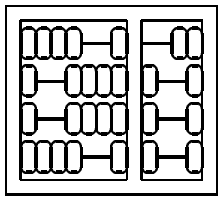
Texture Sythesis from Examples

Workshop Brasileiro de Morfologia Matemática – 1996



Jorge Stolfi

Departamento de Ciência da Computação (DCC–IMECC)

UNICAMP

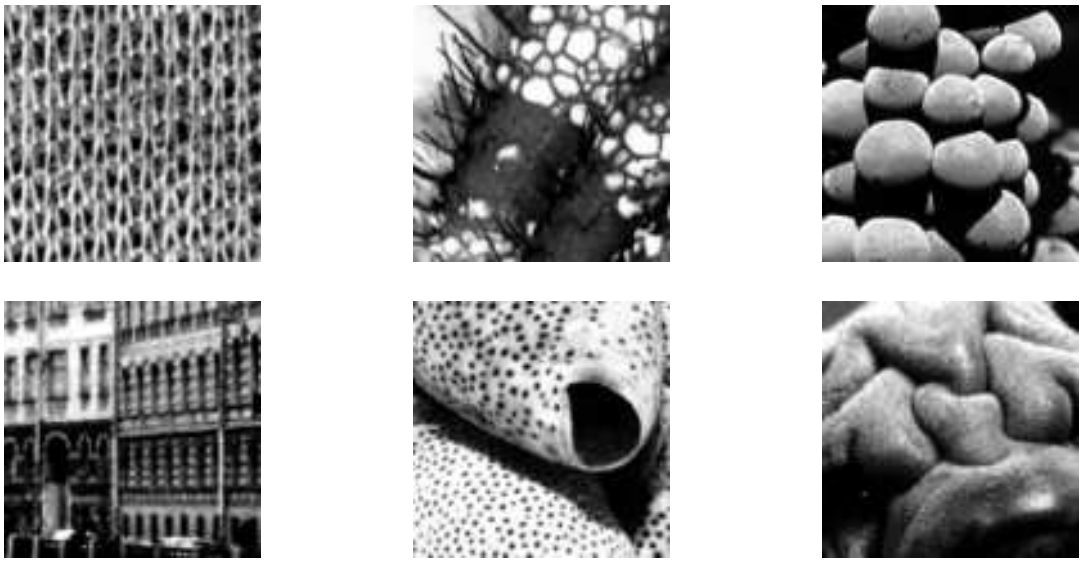
Caveats:

- This is work in progress! Please excuse the paucity of examples...
- This work is not directly related to Mathematical Morphology; but

This work builds upon a FAPESP undergraduate research project by Carlos H. Q. Forster (grant 93/4681-7) and was partly funded by CNPq (grant 301016/92-5).

What is a texture?

Informally, a em texture is an image that is perceived and recognized partly as a statistical distribution, rather than a definite geometric shape.



In texture perception, the interplay between statistics and geometry generally seems to go on at several scales, with different shapes and distributions involved at each level.

Why *synthetic* textures?

Why bother with synthesis? Why not just use scanned-in textures?

Texture synthesis may be desirable or necessary, because

- available texture samples are too small
- available textures aren't be homogeneous enough
- the surface to be textured has odd geometry or topology
- one needs extra knobs to customize or randomize

Approaches to texture synthesis

Some of the texture synthesis methods that have been considered:

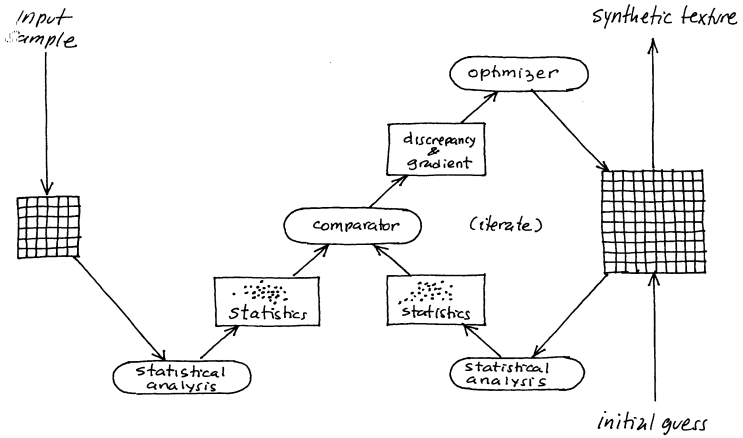
- Fourier transform
- modified noise
- distance correlation
- fractals
- reaction-diffusion systems
- particle systems
- gas-lattice systems
- ...

Each of these methods can produce only rather limited range of textures.

It is hard to select the free parameters so as to get the desired result.

Texture synthesis by statistical imitation

The basic algorithmm: given a sample texture S, computes an artificial texture A that "resembles" S

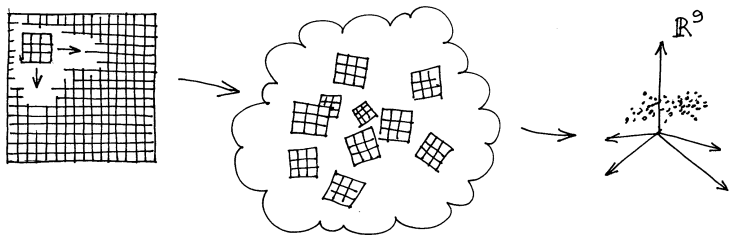


Local statistics

For the statistical analysis of a discrete image, we view it as a *set* of overlapping pixel neighborhoods.

A $k \times k$ neighborhood can be viewed as a point in $\mathbb{R}^{k^2}$.

A texture with N pixels becomes a cloud of N points in $\mathbb{R}^{k^2}$.
These points are assumed to be samples from some continuous distribution, which "defines" the texture (at this scale).



Statistics for isotropic textures

If the texture is supposed to be isotropic, it is desirable to reduce the neighborhood space $\mathbb{R}^{k^2}$ to a smaller space of local statistics that are rotationally invariant, for instance

$$\begin{aligned}
\tau_0 &= f \\
\tau_1 &= f_x^2 + f_y^2 \\
\tau_2 &= f_{xx} + f_{yy} \\
\tau_3 &= f_{xx} f_y^2 - 2 f_{xy} f_x f_y + f_{yy} f_x^2 \\
\tau_4 &= f_{xy}^2 - f_{xx} f_{yy}
\end{aligned}$$

Comparing two clouds

The statistical analysis transforms an image with N pixels into a cloud of N points in some parameter space $\mathbb{R}^d$.

The comparator must evaluate some *discrepancy* function of two clouds: the M points from the sample S, and the N points from the artificial texture A.

The discrepancy function had better be continuous and differentiable on the point positions.

The *electric* discrepancy: view the two distributions as particles with charge $+1/M$ and $-1/N$, respectively, and evaluate the electric potential

$$\mathrm{edist}(A, S) = \sum_{\substack{u,v \in A \cup S \\ u \neq v}} \frac{q_u q_v}{|u - v|}$$

(This formula needs some adjustments to avoid infinities.)

Electric formula speedups

The electric potential measure becomes quite expensive for large textures ($O(MN)$ time.).

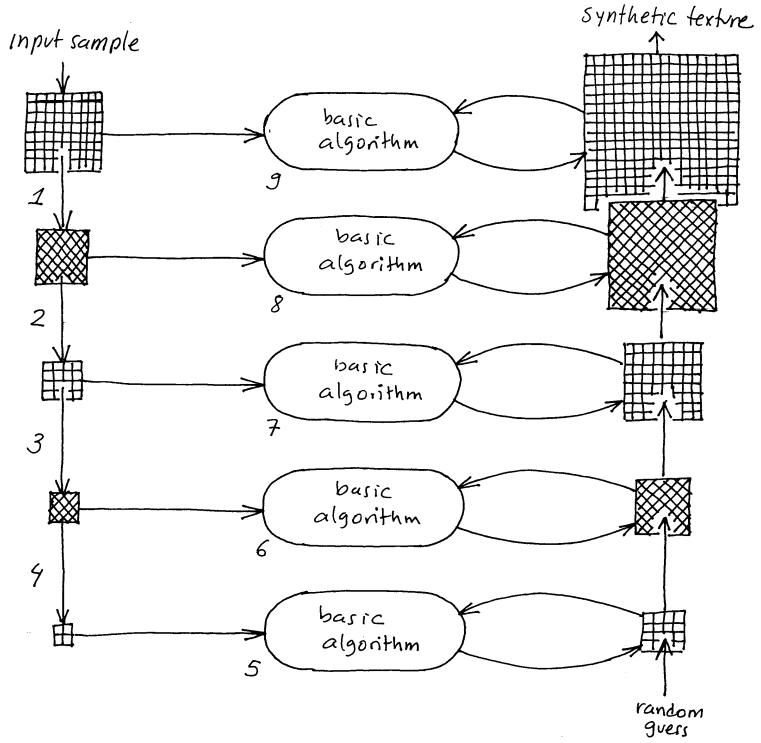
An obvious optimization would be to reduce each cloud to a d-dimensional histogram, and then compare the histograms. However, the points tend to be chustered along a twisted two-dimensional manifold sitting in $\mathbb{R}^d$; so the histogram would have to be an adaptive one. Also, one would have to use a smooth histogram, in order not to introduce discontinuities and plateaus in the comparison function.

Further along this line, there are fast (almost linear-time) algorithms for computing the electric potential of a set of charges. These algorithms might be of some help. However, care must be taken to get a reliable gradient estimate, too.

Texture synthesis by *multiscale* statistical imitation

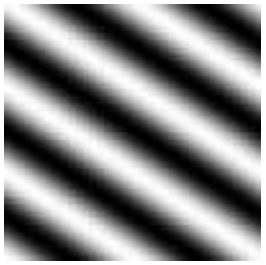
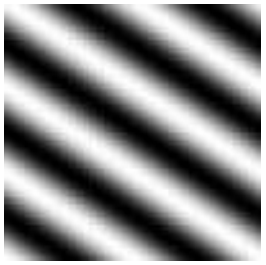
The basic algorithm alone doesn't work: it is sensitive only to local structure of the texture.

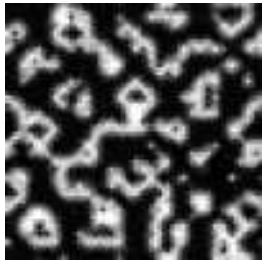
To imitate the sample texture at all scales, we use mutiscale techniques:



Passage from one level to the next requires shrinking the sample texture by a factor of 2 in area, and, some time later, expanding the synthetic texture by the same amount.

Some preliminary results

(a) (c)  (e) 

(b)  (d) (f)

Samples are at top, synthetic textures at bottom. The local statistics used were:

(a,b) the pixel value f and its 4 nearest neighbors.

(c,d) the pixel value f and the gradient $(\partial f/\partial x, \partial f/\partial y)$.

(e,f) the pixel value f and the slope squared $(\partial f/\partial x)^2 + (\partial f/\partial y)^2$ (isotropic).