Computational Vision
U. Minn. Psy 5036
Daniel Kersten
Lecture 21: Texture

Initialize

■ **Spell check off**

```
Off[General::spell1];
```

■ **Read in Statistical Add-in packages:**

```
Off[General::spell1];
<< Statistics`DescriptiveStatistics`
<< Statistics`DataManipulation`
<< Graphics`Graphics`
```

■ **Histogram**

```
histogram[image_, nbin_] := Module[{histx},
   Needs["Statistics`DataManipulation`"];
   histx = BinCounts[Flatten[image], {0, nbin - 1, 1}];
   Return[N[histx / Plus @@ histx]];
  ];
```

■ **Entropy**

```
entropy[probdist_] := Plus @@ (If[# == 0, 0, -# Log[2, #]] & /@ probdist)
```

Outline

Last time

Later, we'll pick up on motion again--namely structure from motion in the context of determining layout and computing heading

Surface material:

Surface properties, color, transparency, etc..

Reflectance & lightness constancy

Today

Generative models for texture classes

The "generic" natural image model

Is human vision "tuned" to natural image statistics?

Generative models for texture

Imagine an image ensemble consisting of all 256x256 images of "grass". This set is unimaginably large, yet there is a set of characteristic features that are common to all these images. Imagine we have an algorithm that from knowledge of these features generate random image samples from this imaginary ensemble. One kind of algorithm takes a white noise image as input , and produce as output image samples that resemble grass. The white noise input behaves like fair roll of a die.

We show several methods for generating textures.

And then we give an outline of one method for discovering the features from a small number of sample images.

There have been a number of studies that seek to extract the essential features of a texture class (such as "grass" or "fur" or "all natural images"...) and then use these to build a texture synthesizer that produces new samples from the same texture class. A generative model provides a test of the extent to which the model has capture the essential statistics or features. And as we show at the end of this notebook, a generative model can also be used to text theories of the kinds of information that human vision has about an image ensemble.

First-order intensity statistics. One of the simplest ways to do this would be to take what you've learned about intensity histograms, and then write a program that would produce new images by drawing pixel intensities from your model histo-

gram, assuming each pixel is independent of the others. In other words, make random draws without consideration of any other pixel values.

Make a random image generator that draws samples from an intensity histogram measured from an natural image

---

Random Fractals

Second-order intensity statistics. Recall that one way to characterize the second-order statistics of a natural image is in terms of its auto-correlation function. And also recall that the Fourier transform of the autocorrelation function is the spatial power spectrum of an image.

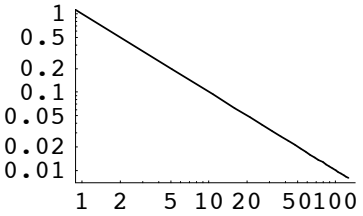
Natural images tend to have spatial frequency power spectra that fall off linearly with log spatial frequency (Simoncelli and Olshausen). When the slope of the fall-off is within a certain range, such images are called random fractals. The slope is related to the fractal dimension.

Random fractals can be characterized by the fractal dimension D (3<D<4) and amplitude spectrum, $1/(f_x^2 + f_y^2)$^(4-D). The amplitude spectrum is thus a straight line when plotted against frequency in log-log coordinates. The condition If[ ] is used to include a fudge term (1/(2)^(q)) to prevent blow up near zero in the Module[ ] routine below.

```
size = 256;
```

Random fractals have been suggested as good statistical models for the amplitude spectra natural images. Here is one way of generating them.

```
D1 = 3.5;
q = 4 - D1;
LogLogPlot[If[(i ≠ 0 || j ≠ 0), 1 / (i * i + 0 * 0) ^ (q), 1 / (2) ^ (q)],
  {i, 0, size / 2 - 1}];
```

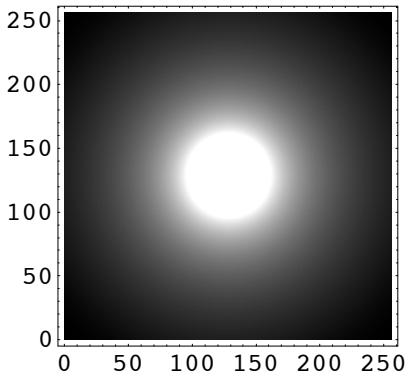


■ **Here is a function to make a low-pass filter with fractal dimension D. (D, here should be between 3 and 4). Note that we first make the filter centered in the middle, and then adjust it so that it is symmetric with respect to the four corners.**

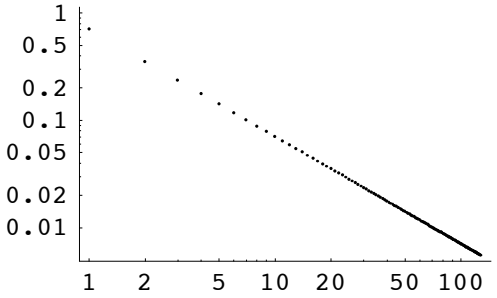
```
fractalfilter2[D_,size_] :=
Module[ {q,i,j,mat},
    q = 4 - D;
    mat = Table[If[(i != 0 || j!= 0),
        1.0/(i^2 + j^2)^q, 1.0/(2)^(q)],
    {i,-size/2,(size/2) - 1},{j,-size/2,(size/2) - 1}];
    Return[mat];
    ];
```

```
ft = Table[N[Pi (2 Random[Real]-1)],{i,1,size},{j,1,size}];
ft = Fourier[ft];
randomphase = Arg[ft];
randomspectrum = Abs[ft];
```

```
ListDensityPlot[fractalfilterarray = fractalfilter2[3.5, size],
  Mesh → False];
```

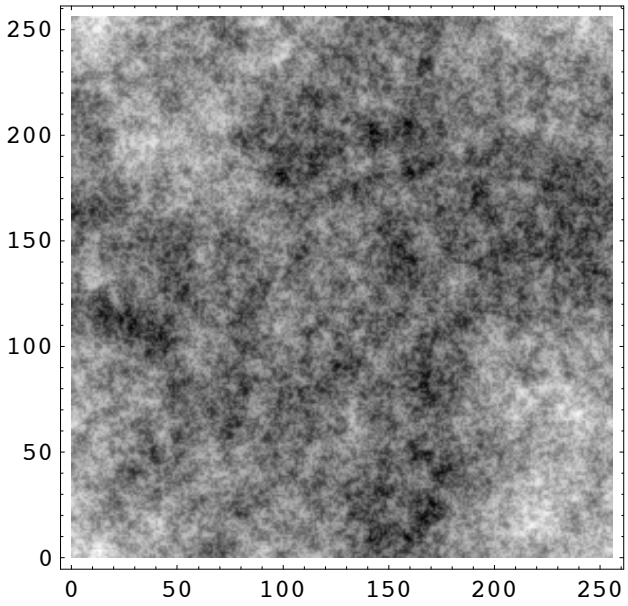


```
LogLogListPlot[
  Table[RotateLeft[fractalfilterarray, {size / 2 + 1, size / 2 + 1}][[i, i]],
   {i, 1, size / 2}]];
```



■ **Here is a random fractal image, with D = 3.5**

```
ListDensityPlot[Chop[
InverseFourier[RotateLeft[fractalfilterarray,{size/2,size/2}]
randomspectrum Exp[I randomphase]]],
Mesh->False];
```

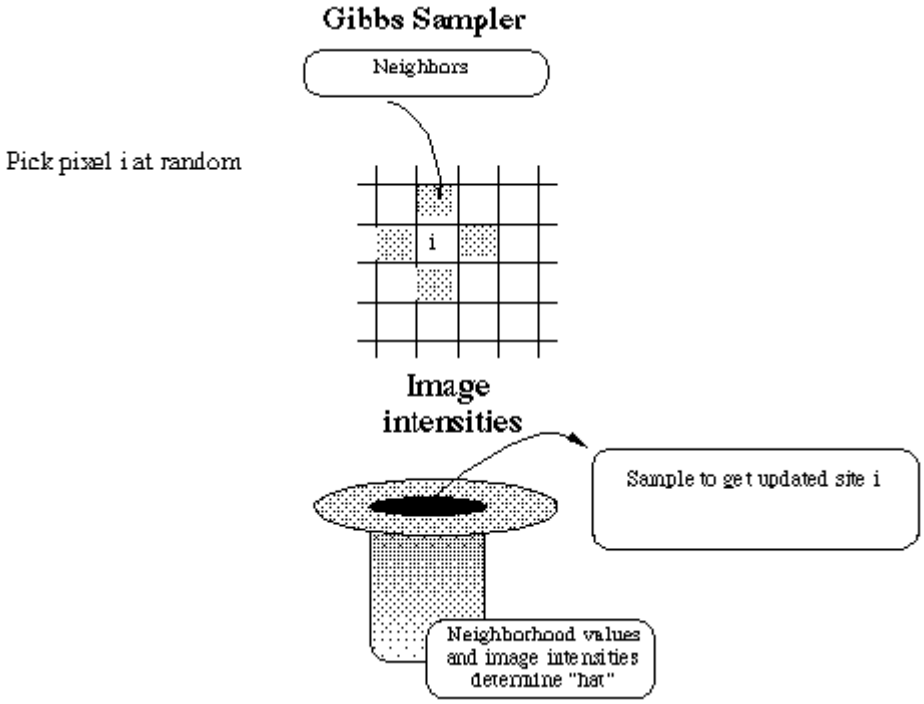


Texture synthesis using Markov Random Field models & Gibbs sampling

Samples from the fractal process modeled above are multi-variate Gaussian. A major limitation of Gaussian models is that they fail to capture phase structure, and in particular edges. This section shows one way to model textures that are piece-wise constant.

■ Modeling textures using Markov Random Fields

■ Sampling from textures using local updates



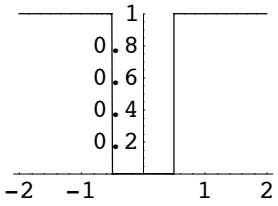
The Gibbs Sampler

■ Set up image arrays and useful functions

```
size = 32; T0 = 1.; ngray = 16.;
brown = N[Table[Random[Integer, {1, ngray}], {i, 1, size}, {i, 1, size}]];
next[x_] := Mod[x, size] + 1;
previous[x_] := Mod[x - 2, size] + 1;
Apply[Plus, Flatten[brown]] / Length[Flatten[brown]];
```

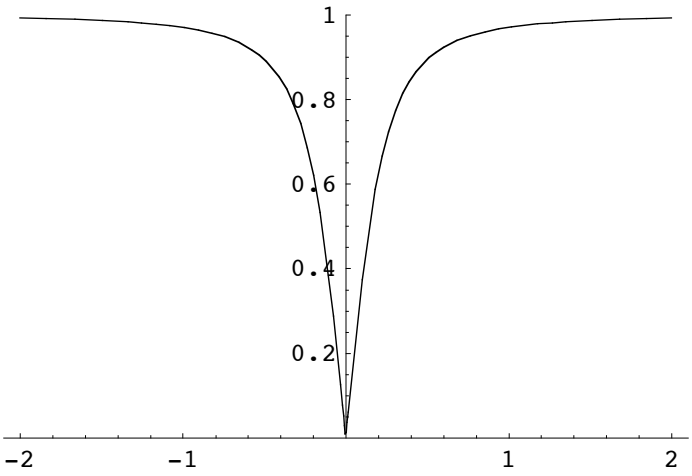
■ Ising potential

```
Clear[f]; (* Clear[f]; f[x_,n_]:=x^2;*)
f[x_, s_, n_] := If[Abs[x] < .5, 0, 1];
(*f[x_,s_,n_]:=N[(x/s)^2];*)
s0 = 1.; n0 = 5;
Plot[f[x, s0, n0], {x, -2, 2}, PlotRange → {0, 1}];
```



■ Geman & Geman potential

```
Clear[f]; (* Clear[f]; f[x_,n_]:=x^2;*)
f[x_, s_, n_] := N[Sqrt[Abs[x / s] ^ n / (1 + Abs[x / s] ^ n)]];
(*f[x_,s_,n_]:=N[(x/s)^2];*)
s0 = .25; n0 = 2;
Plot[f[x, s0, n0], {x, -2, 2}, PlotRange → {0, 1}];
```



■ Define the potential function using nearest-neighbor pair-wise cliques

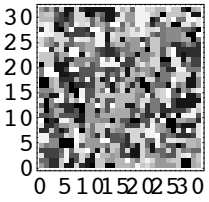
```
Clear[gibbspotential, gibbsdraw, tr];
gibbspotential[x_, avg_, T_] :=
  N[
   Exp[
    -(f[x - avg[[1]], s0, n0] + f[x - avg[[2]], s0, n0] + f[x - avg[[3]], s0, n0] +
        f[x - avg[[4]], s0, n0]) / T]];
```

■ **Define a function to draw a single pixel gray-level sample from a conditional distribution determined by pixels in neighborhood**

```
gibbsdraw[avg_, T_] := Module[{},
   temp =  Table[gibbspotential[x + 1, avg, T], {x, 0, ngray - 1}];
   temp2 = FoldList[Plus, temp[[1]], temp];
   temp10 = Table[{temp2[[i]], i - 1}, {i, 1, Dimensions[temp2][[1]]}];
   tr = Interpolation[temp10, InterpolationOrder → 0];
   maxtemp = Max[temp2];
   mintemp = Min[temp2];
   ri = Random[Real, {mintemp, maxtemp}];
   (*x=Round[Max[0,tr[ri]]];*)
   x = Floor[tr[ri]];
   Return[{x, temp2}];
  ];
```

■ **"Drawing" a texture sample**

```
For[iter = 1, iter ≤ 10, iter++,
  T = .25;
  For[j1 = 1, j1 <= size * size, j1++,
   {i, j} = {Random[Integer, {1, size}], Random[Integer, {1, size}]};
   avg = {brown[[next[i], j]], brown[[i, next[j]]] , brown[[i, previous[j]]] ,
     brown[[previous[i], j]]};
   brown[[i, j]] = gibbsdraw[avg, T][[1]];
  ];
  ListDensityPlot[brown, Mesh → False, PlotRange → {1, ngray}];
 ];
```

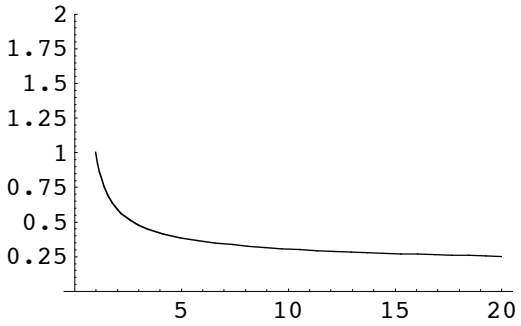


Was it a true sample? Drawing true samples means that we have to allow sufficient iterations so that we end up with images whose frequency corresponds to the model. How long is long enough?

Finding modes

■ **Define annealing schedule**

```
anneal[iter_, T0_, a_] := T0 * (1 / a) / (1 / a + Log[iter]);
Plot[anneal[iter, T0, 1], {iter, 1, 20}, PlotRange → {0, 2}];
```



■ **"Drawing" a texture sample with annealing**

```
For[iter = 1, iter ≤ 10, iter++,
  T = anneal[iter, T0, 1];
  For[j1 = 1, j1 <= size * size, j1++,
   {i, j} = {Random[Integer, {1, size}], Random[Integer, {1, size}]};
   avg = {brown[[next[i], j]], brown[[i, next[j]]] , brown[[i, previous[j]]] ,
     brown[[previous[i], j]]};
   brown[[i, j]] = gibbsdraw[avg, T][[1]];
  ];
  ListDensityPlot[brown, Mesh → False, PlotRange → {1, ngray}];
 ];
```

Learning textures

A fundamental problem in learning image statistics that are sufficient for generalization and random synthesis is that images have enormously high dimensionality compared with the size of a reasonable database. One method to deal with this is to seek out probability distributions that have the same statistics (i.e. a small finite set of statistical features) as those measured from an available database (e.g. "1000 pictures of grass"), but are minimally constraining in other dimensions. Suppose one has a collection of probability distributions that all have the same statistics. At one extreme, the original database itself defines a distribution--a random draw is just a pick of one of the pictures. But this distribution has no "creativity" and leaves out a huge set of grass images not in the database. However, at the other extreme, is the maximum entropy distribution (Cover and Thomas, 1991).

Minimax entropy learning: Zhu et al.

This section provides a brief outline of work by Zhu, S. C., Wu, Y., & Mumford, D. (1997). Minimax Entropy Principle and Its Applications to Texture Modeling. Neural Computation, 9(8), 1627-1660.

See the References for other work on texture learning and modeling.

■ **Maximum entropy to determine $p_M(\mathbf{I})$ which matches the measured statistics, but is "least committal"**

$$\{\phi_i(\mathbf{I}) : i = 1, \ldots, N\}$$

$$\sum_{\mathbf{I}} p_M(\mathbf{I})\phi_i(\mathbf{I}) = \psi_i, \quad \text{for} i = 1, \ldots, N$$

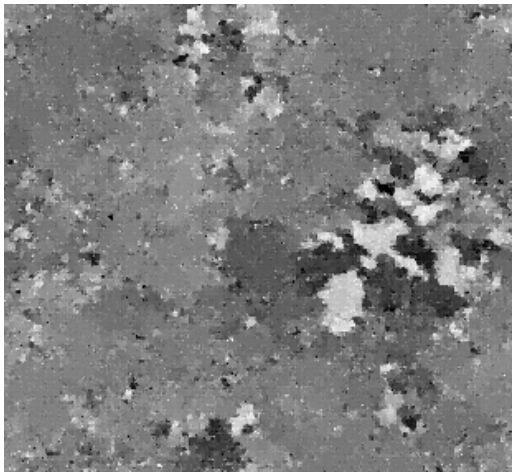
$$p_M(\mathbf{I}) = \frac{1}{Z[\lambda]} \exp\{-\sum_{i=1}^{N} \lambda_i \phi_i(\mathbf{I})\},$$

■ **Minimum entropy to determine statistics/features**

$$\sum_{\mathbf{I}} p(I) \log p_M(\mathbf{I}) = \sum_{\mathbf{I}} p_M(\mathbf{I}) \log p_M(\mathbf{I})$$

$$D(p(\mathbf{I}) \,|p_M(\mathbf{I})) = \text{entropy}(p_M(\mathbf{I})) - \text{entropy}(p(\mathbf{I}))$$

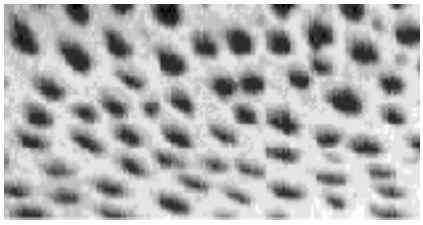
■ Sample from generic prior



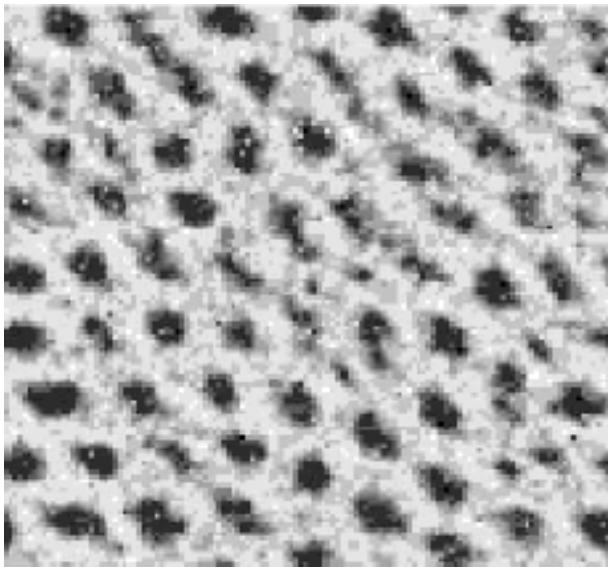
■ Sample from class-specific prior

Song Chun Zhu, Zhu & Mumford, IEEE PAMI, Zhu, Wu, Mumford, 1997

Original texture



Synthesized sample using Gibbs sampler



Texture processing: Human efficiency

One can use density estimation tools and texture synthesis models to test hypotheses of human image coding.

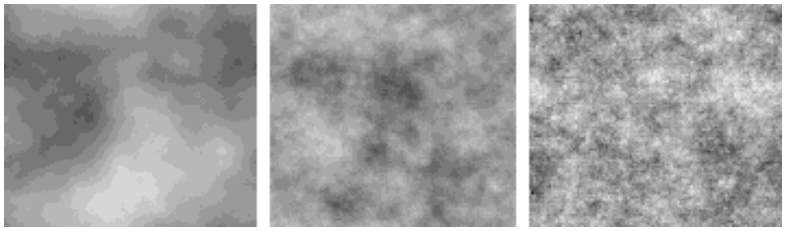
Efficiency of human processing of generic & class-specific textures. Knill, Field & Kersten, 1990

■ Fractal image discrimination

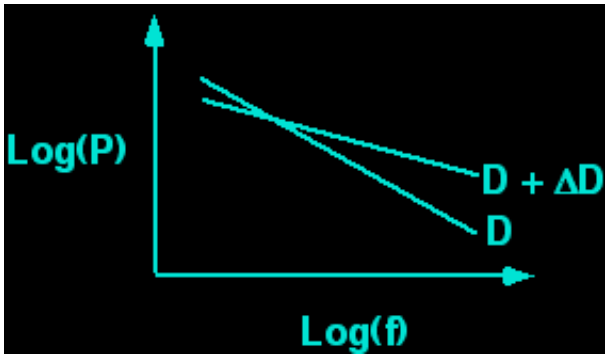
How well is the human visual system tuned to the correlational structure of images? Scale invariant subset of class of images defined by their correlation function, or equivalently by their spatial power spectrum.

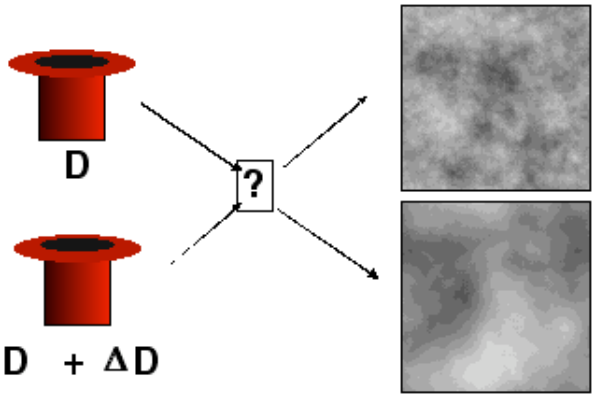
Random fractals: Log(power spectrum) = (2D - 8) Log (spatial frequency)

■ Samples of the stimuli

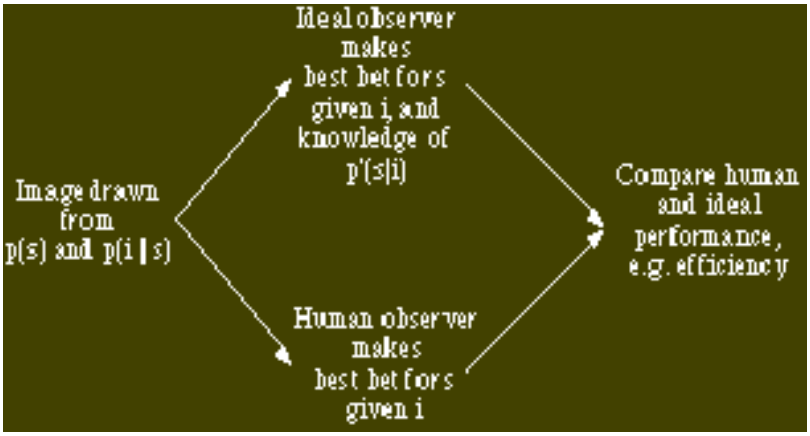


■ The psychophysical task





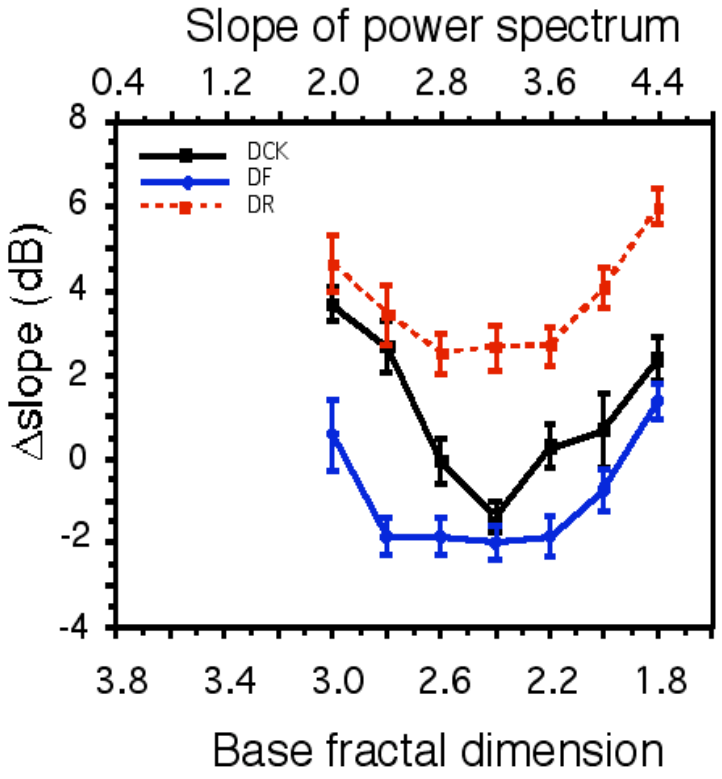
■ **The analysis**



Statistical efficiency

$$E = \frac{\Delta D_I^2}{\Delta D_H^2} \approx 10\%$$

■ **The results**


Slope of power spectrum
0.4
1.2
2.0
2.8
3.6
4.4
8
6
4
2
0
-2
-4
Δslope (dB)
DCK
DF
DR
3.8
3.4
3.0
2.6
2.2
1.8
Base fractal dimension

References

Besag, J. (1972). Spatial interaction and the statistical analysis of lattice systems. {\it Journal of the Royal Statistical Society B}, {\bf 34}, 75-83.

Clark, James J. \& Yuille, Alan L. (1990) Data fusion for sensory information processing systems. Kluwer Academic Press, Norwell, Massachusetts.

Cross, G. C., & Jain, A. K. (1983). Markov Random Field Texture Models. IEEE Trans. Pattern Anal. Mach. Intel., 5, 25-39.

Cover TM, Thomas J, A. (1991) Elements of Information Theory. New York: John Wiley & Sons, Inc.

Geiger, D., & Girosi. (1991). Parallel and Deterministic Algorithms from MRF's: Surface Reconstruction. I.E.E.E PAMI, 13(5).

D. Geiger, H-K. Pao, and N. Rubin (1998). Organization of Multiple Illusory Surfaces. Proc. of the IEEE Comp. Vision and Pattern Recognition, Santa Barbara.

De Bonet JS, Viola PA (1998) A Non-Parametric Multi-Scale Statistical Model for Natural Images. In: Advances in Neural Information Processing Systems (Jordan MI, Kearns MJ, Solla SA, eds): The MIT Press.

Geman, S., \& Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. {\it Transactions Pattern Analysis and Machine Intelligence}, {\bf PAMI-6}, 721-741.

Kersten, D. (1991) Transparency and the cooperative computation of scene attributes. In {\bf Computation Models of Visual Processing}, Landy M., \& Movshon, A. (Eds.), M.I.T. Press, Cambridge, Massachusetts.

Kersten, D., & Madarasmi, S. (1995). The Visual Perception of Surfaces, their Properties, and Relationships. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 19, 373-389.

Madarasmi, S., Kersten, D., & Pong, T.-C. (1993). The computation of stereo disparity for transparent and for opaque surfaces. In C. L. Giles & S. J. Hanson & J. D. Cowan (Eds.), Advances in Neural Information Processing Systems 5. San Mateo, CA: Morgan Kaufmann Publishers.

Stephen R. Marschner, Stephen H. Westin, Eric P. F. Lafortune, Kenneth E. Torrance, and Donald P. Greenberg. *Presented at Eurographics Workshop on Rendering*, 1999.

Marroquin, J. L. (1985). Probabilistic solution of inverse problems. M. I. T. A.I. Technical Report 860.

Mumford, D., & Shah, J. (1985). Boundary detection by minimizing functionals. Proc. IEEE Conf. on Comp. Vis. and Patt. Recog., 22-26.

Lee, T. S., Mumford, D., & Yuille, A. Texture Segmentation by Minimizing Vector-Valued Energy Functionals: The Coupled-Membrane Model.: Harvard Robotics Laboratory, Division of Applied Sciences, Harvard University.

Poggio, T., Gamble, E. B., \& Little, J. J. (1988). Parallel integration of vision modules. {\it Science}, {\bf 242}, 436-440.

Terzopoulos, D. (1986). Integrating Visual Information from Multiple Sources. In Pentland, A. (Ed.), {\it From Pixels to Predicates}, 111-142. Norwood, NH: Ablex Publishing Corporation.

Yuille, A. L. (1987). Energy Functions for Early Vision and Analog Networks. M.I.T. A.I. Memo 987.

Y. Q. Xu, S. C. Zhu, B. N. Guo, and H. Y. Shum**,** "Asymptotically Admissible Texture Synthesis", *Int'l workshop. on*

Statistical and Computational Theories of Vision, Vancouver, Canada, July 2001.

Q. Xu, B. N. Guo, and H.Y. Shum, "Chaos Mosaic: Fast and Memory Efficient Texture Synthesis", M*SR TR-2000-32*, April, 2000.

Zhu, S. C., Wu, Y., & Mumford, D. (1997). Minimax Entropy Principle and Its Applications to Texture Modeling. Neural Computation, 9(8), 1627-1660.

Zhu, S. C., & Mumford, D. (1997). Prior Learning and Gibbs Reaction-Diffusion. IEEE Trans. on PAMI, 19(11).