Computational Vision
U. Minn. Psy 5036
Daniel Kersten
Lecture 12: Coding Efficiency, Spatial statistics

Initialize

■ Read in Statistical Add-in packages:

In[40]:=
```
Off[General::spell1];
<< Statistics`DescriptiveStatistics`
<< Statistics`DataManipulation`
```

In[43]:=
```
<< Statistics`MultiDescriptiveStatistics`
<< Statistics`ContinuousDistributions`
<< Statistics`MultinormalDistribution`
```

■ Histogram

In[46]:=
```
histogram[image_] := Module[{histx},
   Needs["Statistics`DataManipulation`"];
   histx = BinCounts[Flatten[image], {0, 255, 1}];
   Return[N[histx / Plus @@ histx]];
  ];
```

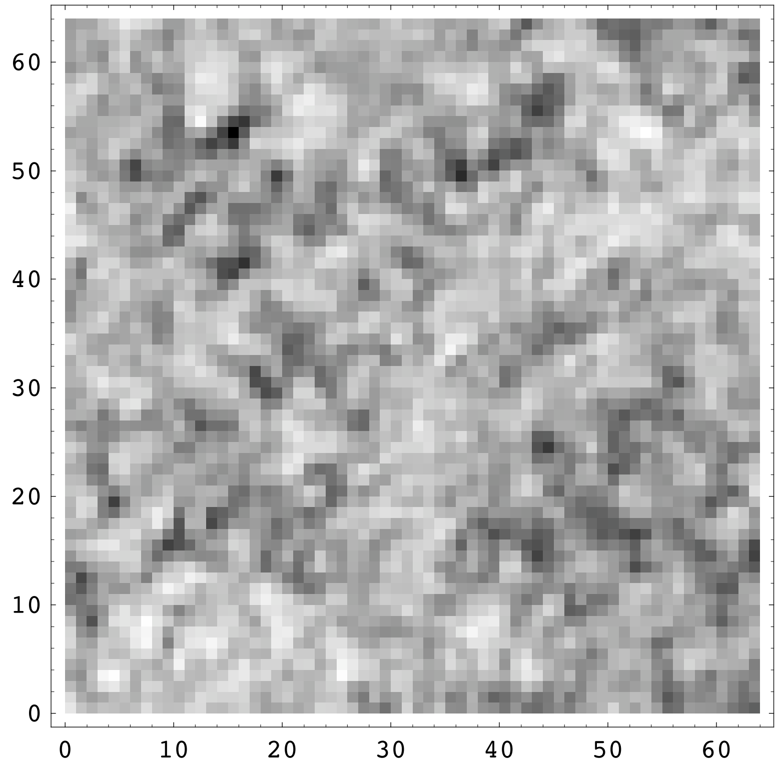
■ Entropy

In[47]:=
```
entropy[probdist_] := Plus @@ (If[# == 0, 0, -# Log[2, #]] & /@ probdist)
```

Import image file

In[48]:=
```
granite = Import[Experimental`FileBrowse[False]];
```

In[49]:=
```
granite = granite /. Graphics → List;
granite = granite[[1, 1]];
ListDensityPlot[granite, Mesh → False];
```



In[52]:=
```
N[Mean[Flatten[granite]]]
N[StandardDeviation[Flatten[granite]]]
width = Dimensions[granite][[1]]
```

Out[52]=
```
129.524
```

Out[53]=
```
19.7279
```

Out[54]=
```
64
```

Outline

Last time

- **Understanding intensity statistics, and point non-linearities in terms of efficient coding of natural images**

Today

- **Form and function: overview of visual pathway**

- **Spatial statistics and efficient coding**

We've learned about localized spatial frequency filters in early vision. We now ask: Why?

Retina to V1: Review of form & function

(There are a number of web-based overviews, for example:
http://www.sumanasinc.com/webcontent/anisamples/neurobiology/visualpathways.html).

Overview of pathways from eye-to-cortex

Roughly ten million retinal measurements are sent to the brain each second, where they are processed by some billion cortical neurons.

The primate retina has about 10^7 cones that send visual signals to the optic nerve via about 10^6 ganglion cells. The optic nerves from the two eyes meet at the optic chiasm where about half of the fibers cross over and the other half remain on the same side of the underside of the brain. Before synapsing in the lateral geniculate nucleus, about 20% of these fibers that make up the optic tract branch off to the superior colliculus--a structure involved with eye movements. The rest of the optic tract fibers synapse on cells in the lateral geniculate nucleus. Cells in the lateral geniculate nucleus send their axons in a bundle called the optic radiation to layer IV (one of six layers) of primary visual cortex. A schematic representation of these pathways was shown in notes for an earlier lecture.

Retina

Earlier we noted that retinal ganglion cells have a characteristic center-surround organization with excitatory centers and inhibitory surrounds (or inhibitory centers and excitatory surrounds). We modeled the spatial output of the retina as a linear filter that convolves the input image with a kernel determined by the center-surround receptive field weights--a so-called single channel model, because the kernel is assumed to be the same shape and size at different locations. The spatial frequency bandpass characteristics of the retina are determined by just one kernel.

The retina's temporal processing can also be thought of as differentiation, but in time rather than space, and can be modeled as a band-pass temporal frequency filter (see Enroth-Cugell and Robson, 1966). Analogous to the spatial frequency selectivity, retinal ganglion cells pass the contrast of medium temporal frequencies more effectively than either low or high frequencies. For a retinal ganglion cell, contrast sensitivity as a function of temporal frequency is an inverted U, qualitatively similar to the spatial CSF. Humans are insensitive to temporal frequencies higher than the temporal cut-off (for humans about 50-80 Hz, depending on the mean light level). That is why TV frames (60 Hz interlaced) or computer displays (now usually >70 Hz) are not seen to be flickering. An extreme consequence of the low temporal frequency attenuation, is that an image that is held stationary on the retina dissappears. A VLSI retina having similar spatial and temporal filtering properties was first built at Caltech by Mead and colleagues (Mead, 1989).

At the retina, one begins to see evidence for multiple visual pathways for spatio-temporal information. In cats, ganglion X-cells have smaller receptive fields and poorer temporal resolution than Y-cells, suggesting that the X channel carries information important for fine spatial detail, and the Y-cell channel conveys coarse-scale spatial information quickly. There is a similar distinction in primates, the, so-called magno-cellular (homologous to Y-cells) and parvo-cellular (homologous to X-cells) cells and pathways.

Human temporal contrast sensitivity functions.

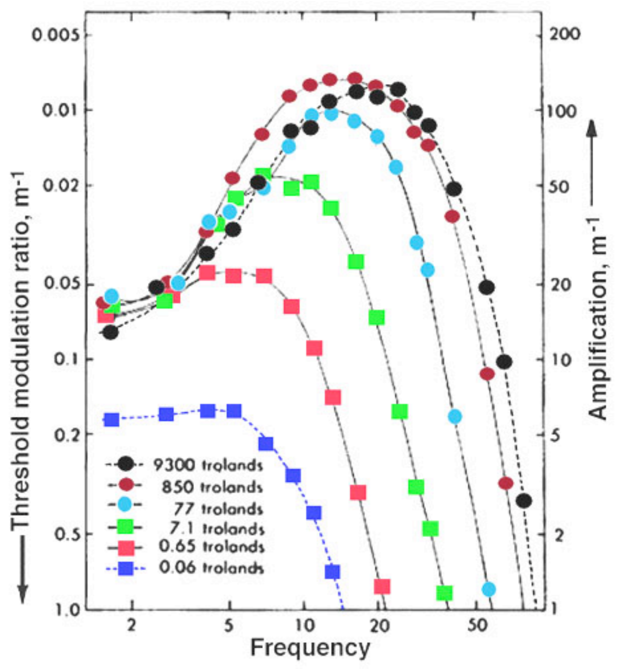


Fig. 11. Temporal Contrast Sensitivity Function (TSF) for various adapting fields. Kelly's data from Hart Jr, W. M., The temporal responsiveness of vision. In: Moses, R. A. and Hart, W. M. (ed) Adler's Physiology of the eye, Clinical Application. St. Louis: The C. V. Mosby Company, 1987.

Functions of the Chiasm and LGN

The optic chiasm routes neuronal information so that information from corresponding points on the left and right eyes can come together at cortex for binocular vision, and in particular stereo vision. Typically animals with frontal vision have nearly complete cross-over, and animals with lateral eyes (e.g. fish) have little or no cross-over. The nervous system has gone to considerable length to bring information from the two eyes together early on. This suggests that certain kinds of cortical computations cannot easily be done "remotely", but require close connectivity between neurons, and the resulting topographic maps.

The neurons of lateral geniculate nucleus do more band-pass filtering, and the cells are characterized by fairly symmetrical center-surround organization like the ganglion cells. They show even less response to uniform illumination than ganglion cells. Despite the fact that neurons from the two eyes exist within the same nucleus, no binocular neurons are found in LGN. We have to wait until cortex to see binocular neurons. The X- and Y-cell division of labor continues with the so-called parvocellular (with corresponding retina input from P cells in monkeys, or X cells in cats), and the magnocellular (Y cells or M cells) pathways. Again the experimental measurements are consistent with the idea the the M pathway carries a fast, but coarse spatial representation of the image to the cortex, while the P pathway carries finer spatial detail but more slowly.

Although the LGN is often considered a relay station, feedback from cortex suggests possible role of attention mechanisms (see Crick, 1984 for a speculative neural network theory of LGN and reticular function; Mumford, 1991; Sillito et al., 1994). Although we will bypass a treatment of the superior colliculus, it has an important role is in the control of eye movements--a highly non-trivial problem requiring coordination of head and eye movements in the context of a constantly changing environment.

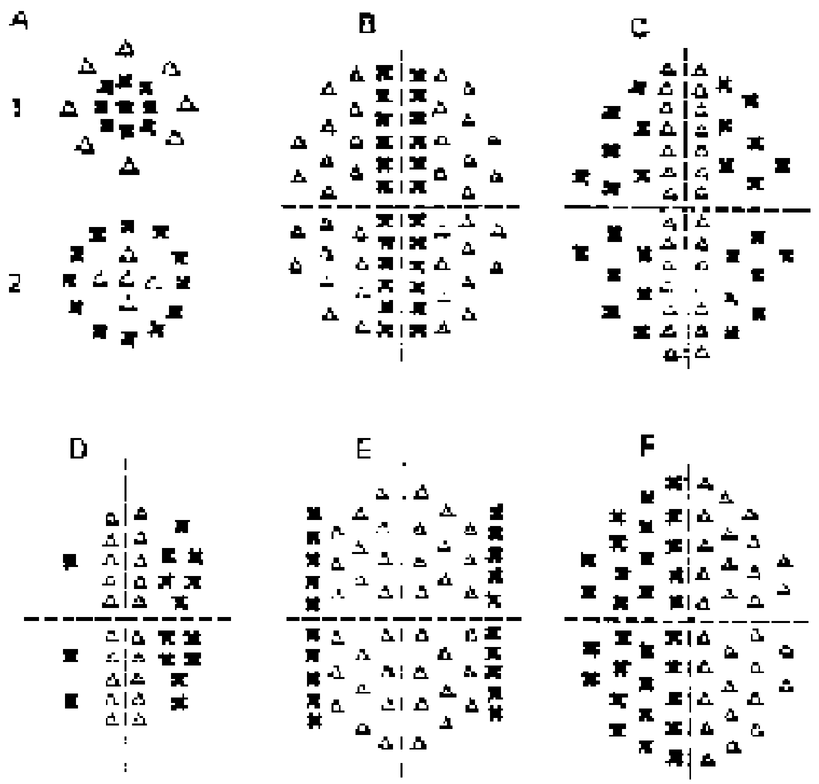
Anatomy and physiology of primary visual cortex

Neurons in the LGn send their axons (the optic radiation) to synapse on layer IV neurons of the primary visual cortex (also known as area 17 in cat, striate cortex or V1in monkeys and humans). Cortex is anatomically structured in layers, numbered from I (superficial) to VI (deep). The striate cortex is laid out as non-linear topographic map with 80% of cortical area devoted to about 20% of visual field, reflecting the higher acuity of foveal vision. Because of the cross-over at the optic chiasm, the left visual field (right retina) maps to right hemisphere. In monkey, many of the neurons in layer IV have receptive field properties similar to those in LGN. However, in striking contrast with receptive field characteristics of earlier neurons, most cortical cells (other layers of V1) show:

- orientation selectivity
- spatial frequency selectivity, some with quite narrow tuning
- spatial phase selectivity (simple cells)
- binocularity
- motion selectivity

Apart from the spatial frequency selectivity, these properties were discovered in large part by the work over a couple of decades by Hubel, D. H., & Wiesel, T. N. (see 1968 reference). Hubel and Wiesel won the Nobel prize for this work.

■ Receptive field structure



29–7 Comparison of the receptive fields of neurons in the retina and in the lateral geniculate nucleus with those of simple cortical cells in area 17. A. Cells of the retina and lateral geniculate fall into two classes: on-center (1) and off-center (2). B–F. Neurons of the primary visual cortex also fall into two major classes: simple and complex. Each of these classes, moreover, has several subclasses. This is illustrated here for simple cells. Despite this variety, however, all simple cells are characterized by three features: (1) specific retinal position, (2) their discrete excitatory (x) and inhibitory (Δ) zones, and (3) specific axis of orientation. For simplicity, only receptive fields with a vertical axis of orientation from 12 to 6 o'clock are shown in this figure; each has a rectilinear configuration. In fact, each region of the retina is represented in area 17, not only for this but for all axes of orientation—vertical, horizontal, and various obliques. (Adapted from Hubel and Wiesel, 1962.)

Figures from Kandel & Schwartz

There are two main types of cells. The **simple cells** are roughly linear except for rectification, are spatially and temporally band-pass, and show spatial phase sensitivity. A first approximation model for simple cell response firing rate (in impulses/sec) is:

$\sigma(\boldsymbol{w}.\boldsymbol{g})$, where **g** is the image vector, **w** the receptive field weighting function, and $\sigma(\cdot)$ is a rectifying function (e.g **If[#>0,#,0]&**).

Both the psychophysical and neurophysiological data could be accounted for, in part, by assuming the visual system performs a quasi-Fourier analysis of the image, the exact form determined by the receptive field weighting function **w**.

We've seen how one possible model assumes that the visual system computes the coefficients (or spectrum) of an image with respect to the following basis set, called a Gabor set (Daugman, 1988). The set $\{w_i\}$ is modeled as:

$\{e^{-\frac{(x^2+y^2)}{2\sigma^2}} \cos(2\pi(f_x\, x + f_y\, y + \phi))\}$, where $i \to (f_x, f_y, \phi)$.

We will return to a more detailed discussion of the receptive field models of simple cells later in the section of functions of the visual cortex. The half-wave rectification operation, σ, sets negative values to zero, and is linear for positive values. The spectrum coefficients are represented by the firing rates of cells whose receptive field weights are represented by the above basis functions. In actuality, because simple cells behave more like linear filters followed by half-wave rectification, there should be two cells for each coefficient-- "on" and "off" cells). One difference between this basis set, and the Fourier basis set (i.e. the optical eigenfunctions) is that this set has a local spatial restriction because of the Gaussian envelope. A second difference, which has major implications for computation, is that the basis functions are, in general, not orthogonal.

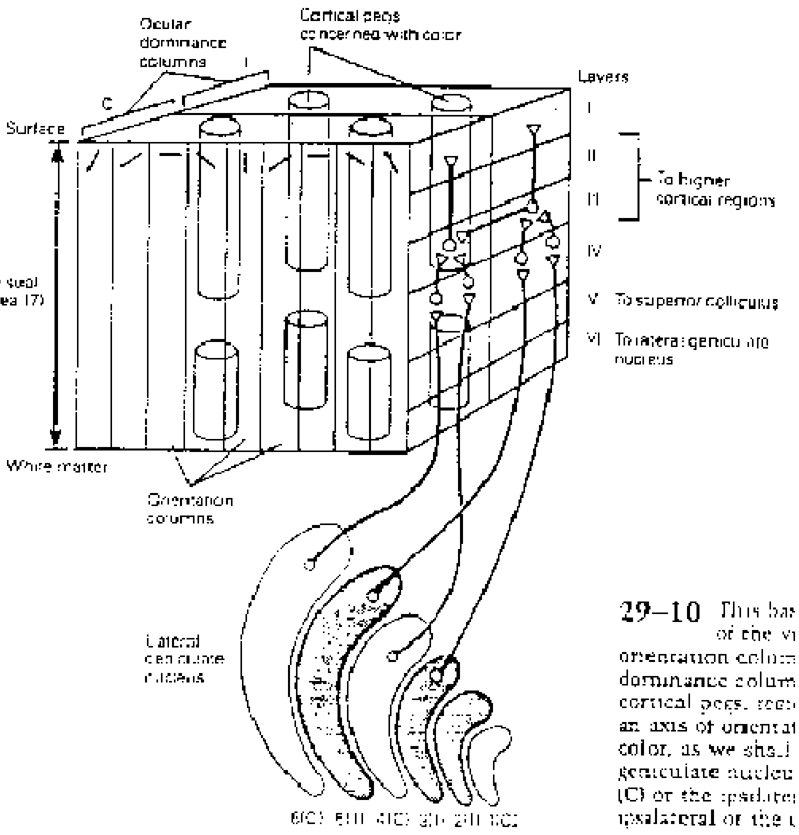
The second major class of neurons is that of **complex cells**. Like simple cells, complex cells are spatially and temporally band-pass, show orientation and motion direction selectivity, but are insensitive to the phase of a stimulus such as a sine-wave grating. Rather than half-wave rectification, they show full-wave rectification. A model for complex cells would resemble the sum of the outputs of several subunits positioned at several nearby spatial locations. Each subunit would resemble simple cell with a linear spatial filter followed by a threshold non-linearity. One way of obtaining the phase insensitivity would be to use subunits with cosine and sine phase receptive fields. The motion selectivity could be built in with appropriate inhibitory connections between subunits. Full-wave rectification could be built with subunit pairs that have excitatory and inhibitory receptive fields centers. Both simple and complex cells show *contrast normalization*--an important feature not included in the above simple model. For a discussion of models of simple and complex cells, see: Heeger, D. J. (1991). Nonlinear model of neural responses in cat visual cortex. In M. &. M. Landy A. (Ed.), Computational Models of Visual Processing (pp. 119-133). Cambridge, Massachusetts: M.I.T. Press.

A third class of cells are the end-stopped (or "hyper-complex") cells that have an optimal orientation for a bar or edge stimulus, but fire most actively if the bar or edge terminates within the receptive field, rather than extending beyond it. It has been suggested that these cells act as "curvature" detectors. (Dobbins, A., Zucker, S. W., & Cynader, M. S., 1987).

But things aren't as necessarily as neat as they at first seem. "Hyper-complex" is seen as less of class, and instead cells can show "end-stopping". Further, see: Melcher and Ringach (2002) for a discussion of the simple/complex cell distinction.

■ Columnar structure

In the cortex, we see for the first time binocular cells. The cells of the primary cortex are organized into columns running roughly perpendicular to the surface in which cells tend to have the same orientation preference and degree of binocularity. A "hypercolumn" is a group of columns spanning all orientations and both eyes


Ocular dominance columns
C
I
Cortical pegs concerned with color
Layers
I
II
III
IV
V
VI
To higher cortical regions
To superior colliculus
To lateral geniculate nucleus
Surface
Primary visual cortex (area 17)
White matter
Orientation columns
Lateral geniculate nucleus
6(C) 5(I) 4(C) 3(I) 2(I) 1(C)

29–10 This bas
of the vi
orientation colu
dominance colum
cortical pegs, seen
an axis of orientat
color, as we shal
geniculate nuclei
(C) or the ipsilate
ipsalateral or the

The receptive field organization of cortical cells is modifiable by experience. A number of models of self-organizing neural networks have been developed to account for this (Von der Malsburg, 1973; Bienenstock et al., 1982; Kohonen, 1981; and Linsker, 1988). Below we consider how efficient coding of natural image predicts how receptive field structure (Olshausen and Field, 1996; 2004).

Embedded in the cortical hypercolumns are cytochrome oxidase blobs in which are found opponent color cells that seem to lack strong orientation selectivity (Livingstone, M. S., & Hubel, D. H., 1984; Livingstone, M. S., & Hubel, D. H., 1987).

Functions of Primary Cortex

■ Local measurements

Basic idea:

V1 cortical cells measure local orientation-specific image contrast differences, that are correlated with spatial changes in surface/object depth, material (texture) and view-object and object-object changes (motion). Our challenge in the second half of the course will be to understand how local measurements can be used for global inference--e.g. as in object recognition.

■ Stereo, or disparity measurements

As mentioned earlier, primary cortex brings together information from the two eyes in single neurons. This information is important for coordinated eye movements and stereo vision. Stereovision depends on the slight image differences, called disparities, that occur as a consequence of the two eyes having different views of the 3D world. Cells can be binocular without being sensitive to disparity. Although V1 cells are predominantly binocular, it was at first thought that disparity selectivity did not arise until V2 (Hubel and Wiesel, 1970). However, there is evidence for disparity selective cells in V1 and V2 (Poggio, G., F., & Poggio, T. ,1984). Disparity selectivity is a trivial task for single bar stimulus (in a uniform background), and it wasn't until relatively recently that neurons were found that effectively solve the problem of false matching (Poggio and Talbot, 1981). One possible algorithm for stereo vision is discussed here: Poggio, T. (1984). Vision by Man and Machine. Scientific American, 250, 106-115. Stereo vision has received a lot of attention in both computer and biological vision over the past several decades (Cumming, B. G., & DeAngelis, G. C. , 2001).

■ Motion measurements

The directional selectivity of cells in striate cortex provide a form of early motion detection, akin to that described for invertebrate and rabbit peripheral vision. This detection is only local and thus ambiguous. Cortical cells suffer from the "aperture problem", and further computation is required to disambiguate object motion. Cortical cells are also selective for speed (Orban et al., 1983).

Both the motion selectivity and binocularity suggest a general hypothesis for cortical function: it links information likely to have a single environmental cause for subsequent extra-striate processing. We will return to the computational theory of motion detection later.

■ Spatial frequency filtering: Psychophysics and physiology

Earlier, we looked at the psychophysical evidence for spatial frequency filtering in the experiment of Campbell and Robson, and the evidence for scale-invariance of the filters in the ideal-observer experiment of Kersten. These two studies represent a small fraction of the psychophysics that has explored the properties of spatial frequency channels in human vision. Both adaptation and masking studies have also been used to infer properties of human spatial filters. The results of masking, adaptation, and other psychophysical studies of spatial and orientation frequency selectivity in human vision are surprisingly consistent in suggesting the basic form for a cortical basis set for images.

The basis set has to be discretized, and leaves several free parameters. Most models of detection and masking get by with about 6 spatial frequencies, about 12 orientations (specified by the ratio of horizontal and vertical spatial frequencies), and two phases (cosine and sine) at each retinal location. A subset of neurons representing a particular spatial frequency bandwidth makes up a spatial frequency channel. (Although there is neurophysiological evidence for pairs of V1 neurons having receptive fields with 90 deg phase shifted relative to each other, there is evidence against absolute phase--i.e. there is not a predominance of edge or bar type receptive fields. See Field and Tolhurst). One parameter still left unspecified is the standard deviation or spread of the Gaussian envelope. If large, this basis set approaches that of regular and global Fourier analysis. The psychophysical data suggest that the standard deviation be such that the Gaussian envelope is about one cycle (at the 1/e point) of the sine wave. One cycle corresponds to about 1.5 octaves spatial frequency bandwidth (an octave measure of width is: log to the base two of the ratio of the high to low frequencies.)

Why would the visual system have such a representation that combines orientation and spatial frequency selectivity? We have two types of explanations. One is that encoding over multiple spatial scales is important for subsequent processing that may involve edge detection, texture measurements, or stereoscopic matching, and so forth. Analogous pyramid schemes have been developed for computer vision. (See Adelson, E. H., Simoncelli, E., & Hingorani, R., 1987). The second explanation is in terms of economical or efficient encoding which we return to below (Simoncelli and Olshausen, 1999).

In short, the image processing functions from eye to cortex are:

Retina

spatio-temporal filtering attenuates low frequencies, wavelength/color coding

Chiasm

begins grouping information from nearby points in the world to nearby biological locations

Lateral geniculate nucleus (lgn)

more spatio-temporal filtering. groups, but doesn't combine information from two eyes.

Primary visual cortex (V1, striate, 17)

- Brings together local image measurements--information that belongs together
    - columnar structure
    - binocular vision and stereopsis
    - motion
    - edge & bar detectors
- Spatial filtering by: Simple, complex, end-stopped cells
- Why spatial filtering?
    - cortical basis set and efficient image representations
    - edge detection

Efficient representation of information & neural networks

We'll first consider the single-channel model and retinal coding.

Lateral inhibition is pervasive in early visual coding across many species of animals, from invertebrates like the horseshoe crab to primates. We would like to know why, and thus come up with a computational theory for lateral inhibition. We already saw an argument for lateral inhibition as a front-end for edge detection. It is also a means to reduce the dynamic range--but is there a principled way of reducing the dynamic range while avoiding discarding information? Let’s look at possible explanation is in terms of efficient encoding.

The retina needs to encode a large number of levels of light intensities into a small number of effective neuronal levels. A quick calculation based on Poisson statistics shows that in about a 1/5 second, *there are about 200 reliably distinguishable light levels* given a potential (huge) range of between 10^10 and 10^(-2) photons/sec/receptor at 555 nm.

A similar calculation based on Poisson statistics for neural discharge indicates *only about 14-16 levels can be encoded in 1/5 of a second.* (Ganglion cell discharge is in general modeled by a Gamma distribution on inter-spike intervals, and Poisson statistics are a convenient approximation that corresponds to a first order gamma distribution; Gerstein, 1966; Robson and Troy, 1987.) We can make a calculation based on a first order Poisson approximation:

$$p(k \text{ spikes in } \Delta t) = \frac{e^{-\lambda \Delta t} (\lambda \Delta t)^k}{k!}$$

(λ=average rate, $\lambda(t)=f$(intensity or contrast)

Because of the refractory period, the maximum rate is less than 1000 Hz. In general, it is much lower for ganglion cells, and 250 would be a liberal upper bound.

250 Hz => 50 spikes in 1/5 sec.

Working down in steps of 1 standard deviation produces about 14 levels. The big challenge is to go from 200 levels to 14

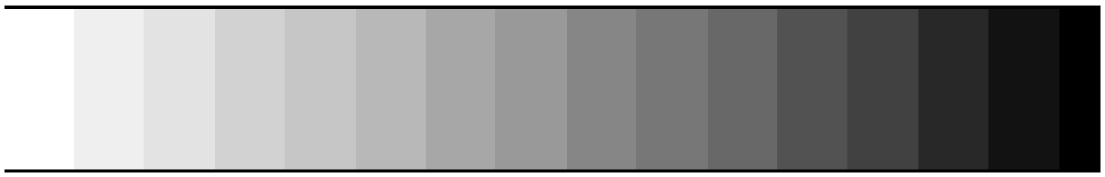
$\text{Log}_2 200$ -> $\text{Log}_2 14$, with minimal loss of information.

This would require squeezing 7.6 to 3.8 bits/cone. Of course, we don't have to handle this whole range for a given scene and using a single mechanism. We've seen how a duplex receptor system helps, and the role of a sigmoidal non-linearity. We've also noted, this is not simply a matter of introducing a non-linearity--this will not work because the variability is the ultimate limit to resolution and it would still remain.

What tricks that could be used to handle the range problem?

It turns out that for an arbitrary image ensemble, one cannot construct a reversible coding scheme that could squeeze the number of bits down. But for an image ensemble with some statistical structure or redundancy, there is hope. What is meant by statistical structure or redundancy?

In a 128 x 128 x 4 bit graphics display, there are 2^(128*128*4) or about 10^19,728 possible pictures. Imagine a machine that started iterating through them. The vast majority would appear unnatural and look like TV "snow" or visual noise. Only a near infinitesmal small fraction would correspond to natural images...i.e. are likely to occur. So what is this fraction? Some years ago, I estimated an upper bound on this fraction using theoretical results from Claude Shannon's famous guessing game for the predictability of written English text (Kersten, 1987). The result was that number of possible meaningful images < 10^6905 . If you could sit for multiple eons of time and view all the 10^19,728 pictures on your 128 x 128 x 4 bit computer display, about one out of every 10^12,823 pictures and your brain would "click" and you would say "aha, that one looks natural." Why is this? One fundamental reason is that there are correlations between neighboring pixel intensities. Correlations are one simple and basic measure of redundancy in images.



We need tools for measuring correlations, and redundancy in images.

2nd order statistics

Example of the idea: a non-isotropic "1-D random-walk" image ensemble

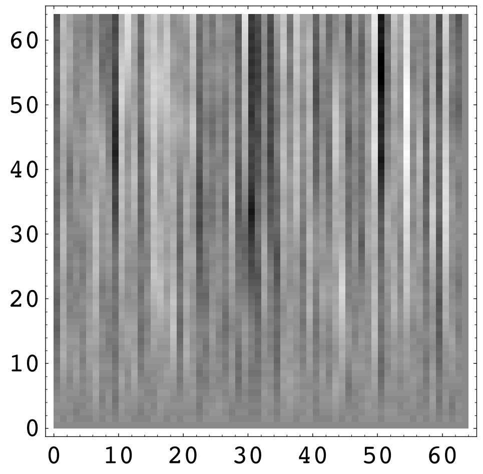
■ 1-D Brownian

In[55]:=
```
step := 2 (Random[Integer, 1] - 1 / 2);
next[x_] := Mod[x, size] + 1;
```

In[57]:=
```
size = 64;
brown = N[Table[128, {i, 1, size }, {i, 1, size }]];
```

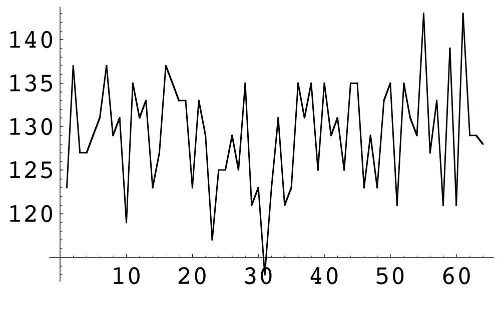
In[59]:=
```
For[j = 1, j < size, j ++,
  For[i = 1, i < size, i ++,
    If[Random[] > 0.5, brown[[next[i], j]] = brown[[i, j]] + step,
     brown[[next[i], j]] = brown[[i, j]] - step];
    If[brown[[i, j]] > 255, 255];
    If[brown[[i, j]] < 1, 0];
   ];
 ];
```

In[60]:=
```
ListDensityPlot[brown, Mesh → False];
```



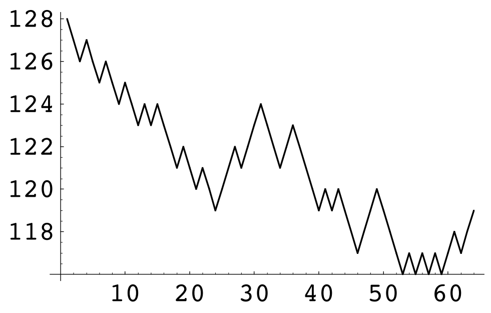
Along a horizontal line, the intensities are quite random--the samples were drawn independently. The gray-levels from pixel to pixel are not correlated:

In[61]:=
```
ListPlot[brown[[32]], PlotJoined → True];
```

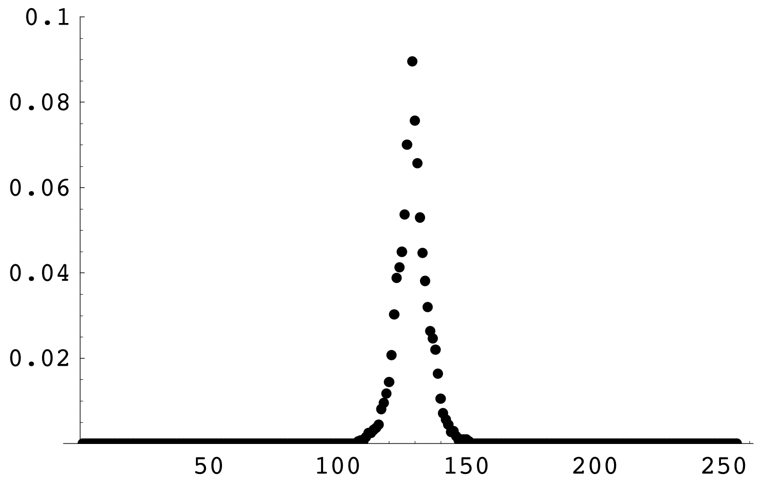


In contrast, vertical lines show a degree of regularity:

In[62]:=
```
ListPlot[Transpose[brown][[32]], PlotJoined → True];
```



In[63]:=
```
histobrown = histogram[brown];
ListPlot[histobrown, PlotStyle → PointSize[0.015], PlotRange → {0, 0.1}];
entropy[histobrown]
```

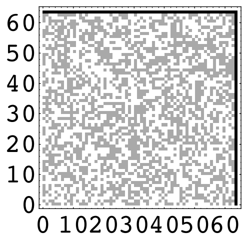


Out[65]=
```
4.54018
```

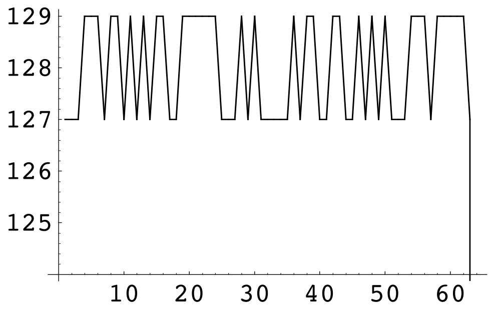
■ Efficient encryption code for 1-D brownian images

In[66]:=
```
codebrown = Table[0, {size}, {size}];
For[j = 1, j < size, j++,
  For[i = 1, i < size, i++,
    codebrown[[i, j]] = brown[[next[i], j]] - brown[[i, j]] + 128;
   ];
 ];
```

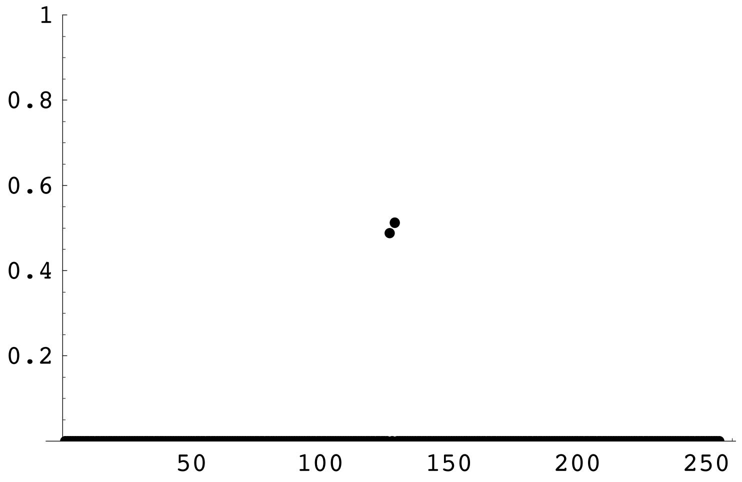
In[68]:=
```
ListDensityPlot[codebrown, Mesh → False];
```



In[69]:=
```
ListPlot[codebrown[[32]], PlotJoined → True];
```



In[70]:=
```
histocodebrown = histogram[codebrown];
ListPlot[histocodebrown, PlotStyle → PointSize[0.015],
  PlotRange → {0, 1}];
entropy[histocodebrown]
```



Out[72]=
```
0.999569
```

Second order statistics

■ Autocorrelation function

`ListCorrelate[`*ker*`,` *list*`]` computes $\sum_r K_r a_{s+r}$. Autocorrelation corresponds to $K_r \to a_r$: $\sum_r a_r a_{s+r}$.

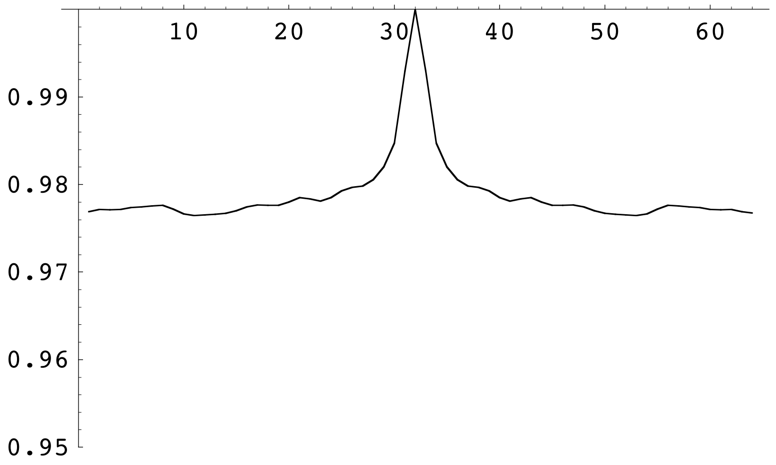
Analyze the correlation between pixel gray levels for each line, and then average them:

■ Read in an image, say face256

In[73]:=
```
face256 = granite;
```

In[74]:=
```
autoface2 = Table[0, {width}];
For[i = 1, i < width + 1, i ++,
  autoface2 += ListCorrelate[face256[[i]], face256[[i]], width / 2]];
```

In[99]:=
```
ListPlot[autoface2 / Max[autoface2], PlotJoined → True,
  PlotRange -> {.95, 1}];
```



■ Covariance matrices, and the outer product

Recall that the covariance is: $\text{Cov}[X, Y] = E[[X - \mu_X]\,[Y - \mu_Y]]$. The correlation gives a dimensionless measure of covariation: $\rho[X, Y] = \frac{\text{Cov}[X,Y]}{\sigma_X \sigma_Y}$.

Let **X**={x_1 ...} and **Y** = {y_j ...} be vectors. The average of the products $x_i\, y_j$ or $(x_i - \mu_{x_i})\,(y_j - \mu_{y_j})$ give measures of how well x_i and y_j predict each other. The latter collection of average products is called the covariance matrix:

$$\text{Cov}[X, Y] = E\left[(x_i - \mu_{x_i})\left(y_j - \mu_{y_j}\right)\right] = E[[X - \boldsymbol{\mu_X}]\,[Y - \boldsymbol{\mu_Y}]^T]$$

where XY^T is the notation for *outer product* of X and Y. *Mathematica* for the outer product is: **Outer[Times, X,Y]**. The outer product takes two vectors and produces the matrix whose entries are all possible pair-wise products of the elements of the two vectors. Contrast the outer with the inner (or dot) product which returns a scalar given two input vectors. Given M samples {X^s, Y^s}, we can estimate the covariance matrix as: $\frac{1}{M}\sum_{s=1}^{M}\,[X^s - \boldsymbol{\mu_X}]\,[Y^s - \boldsymbol{\mu_Y}]^T$.When X=Y, an covariance matrix is called an autocovariance matrix, and similarly for autocorrelation. A covariance matrix is a symmetric matrix, and thus has orthogonal eigenvectors with real eigenvalues--a property that will become useful later.

■ Multivariate gaussian

The multivariate gaussian is a generalization of the gaussian distribution to higher dimensions, in which the standard deviation is replaced by the covariance matrix. The multivariate gaussian plays a central role in statistics, and provides a crude approximation as a generative model for natural images. The probability of vector x of dimension p is given by:

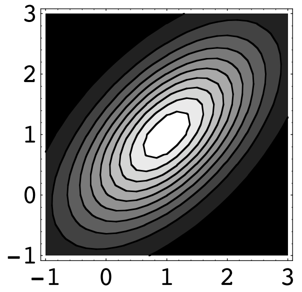
$$p(x) = \frac{1}{\sqrt{(2\pi)^p |\Sigma|}}\, e^{-\frac{1}{2}(x-\mu)^T\,\Sigma^{-1}(x-\mu)}, \text{ where } |\Sigma| = \text{Det}[\Sigma].$$

where μ is the vector mean, and Σ is the covariance matrix. *Mathematica* has an add-in package that extends the normal routines to the multivariate case:

A two-dimensional example.

```
In[91]:=  Σ = {{1, .6}, {.6, 1}};
          μ = {1, 1};
          ndist = MultinormalDistribution[μ , Σ ];
```

```
In[94]:=  ContourPlot[PDF[ndist, {x, y}], {x, -1, 3}, {y, -1, 3}];
```



Going to higher dimensions, an exponential drop-off in correlation can be modeled as a covariance matrix with diagonal elements equal to 1, and an exponential drop-off away from the diagonal. So the first row would be:

```
In[82]:=  row1[ρ_] := Table[ρ^i, {i, 0, 15}];
```

Later we show how the covariance matrix can be used to find a new basis set for images such that when we project images onto the basis elements, the projections are no longer correlated. One way to do this is through Principal Components Analysis or PCA.

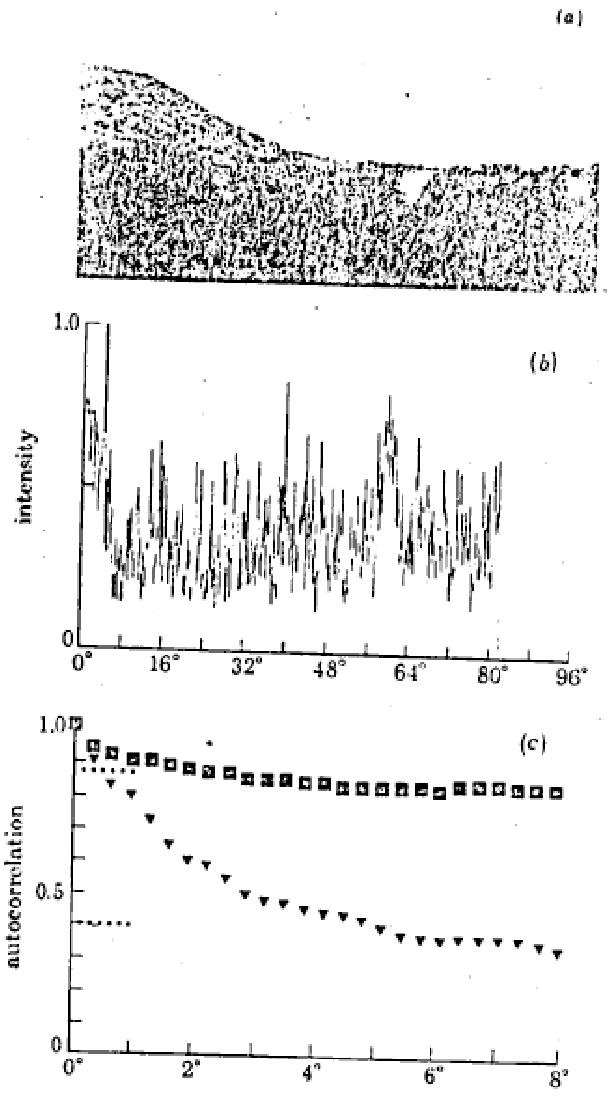
But first, let's look at some early and recent research that has sought to explain receptive field structure in terms of redundancy reduction.

Efficient coding by the retina and V1

Predictive coding & retina

Srinivasan et al. (1982) were the first to make quantitative predictions of how the retina makes use of inherent spatial and temporal correlations between light intensities found in natural images to reduce the output range required to send information about images. They measured the autocorrelation function and showed that it could be fit with an exponential curve.

■ Autocorrelation measurements & model

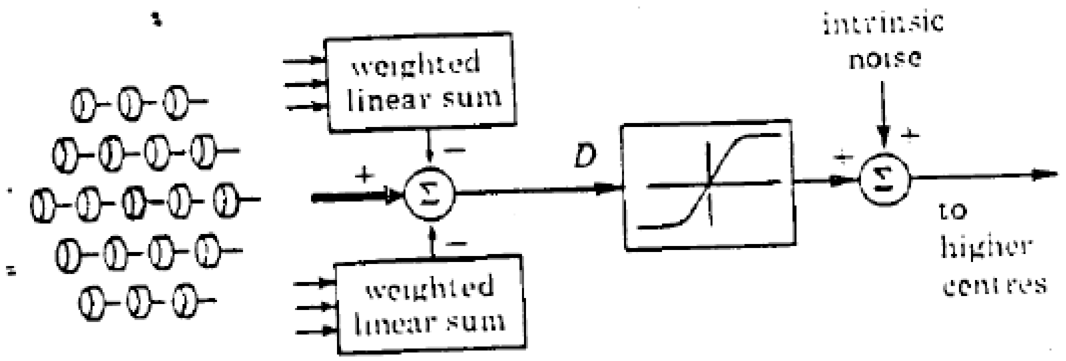


'he autocorrelation function of visual scenes. (a) The scene, a bed of reeds. (b) ty profile of scene, measured along a line joining the centres of the marker ty is plotted on a linear

$$E(L_i L_j) = \sigma^2 exp\left(-\frac{d_{ij}}{D}\right) + M^2 + N^2 \delta_{ij}$$

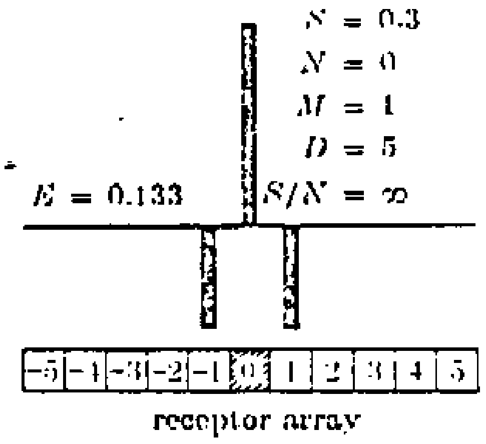
■ Linear neural network

They assumed a linear model:



■ The result

Given the autocorrelation function, and the linear model, $R_j = \sum_i w_{ji} L_i = L_j - \sum_{i \neq j}^{\square} H_{ji} L_i$, they were able to show that the receptive field weights that minimized $E(R_j^2)$ predicted a "center-surround" receptive field:



They also showed that one would expect the inhibitory side lobes to get smaller at low light levels.

V1

■ Olshausen & Field: Primary cortex

We might expect something like Fourier analysis of the image to result in efficient coding because of the close relationship between Fourier rotations and Karhunen- Loeve transformations (or Principal Components Analysis, see below) (e.g. Appendix A, Andrews, 1983). Fourier coefficients for natural images tend to be uncorrelated. Some work has been completed toward a functional explanation for the orientation and spatial frequency tuning properties of cortical receptive fields based on the statistics of natural images (Field, 1987; Snyder), but the story is far from complete. Barlow has argued that a decorrelated representation of sensory information is important for efficient learning (Barlow, 1990).

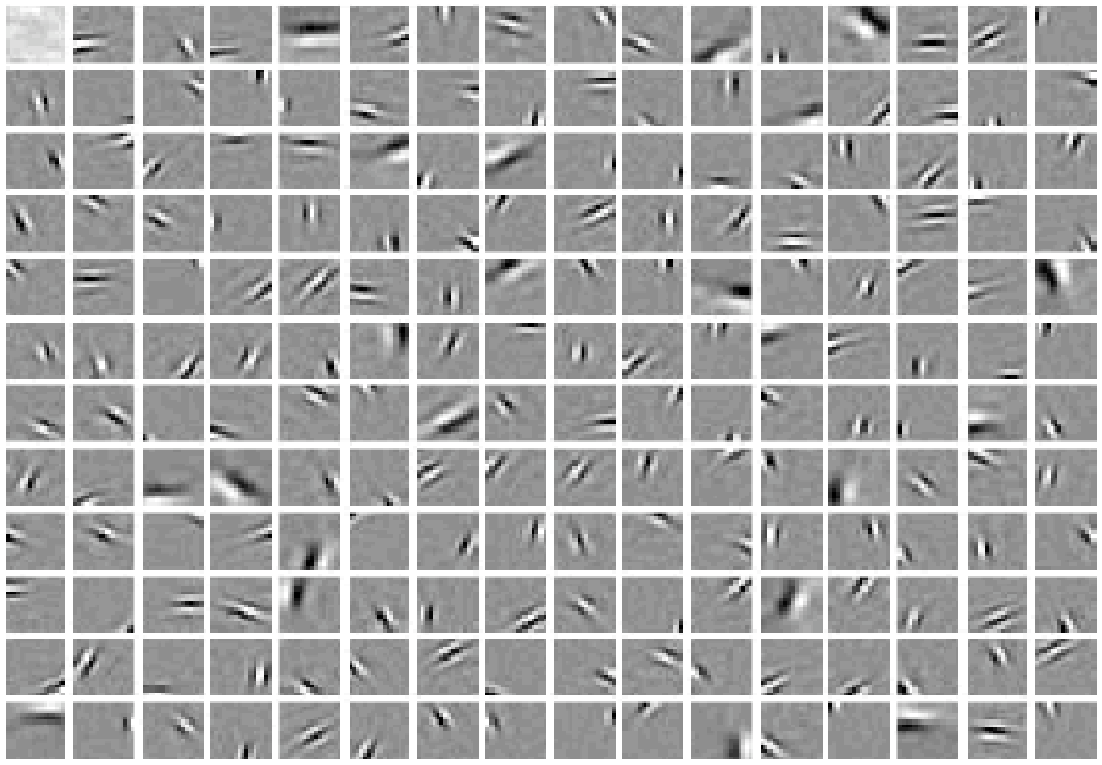
There has been progress studying the relationship between self-organizing models of visual cortex, and efficient coding of image information. For more on this, see: Linsker, R. (1990) and Barlow, H. B., & Foldiak, P. (1989). Linsker's computational studies show, for example, that orientation tuning, and band-pass properties of simple cells can emerge as a consequence of maximum information transfer (in terms of variance) given the constraint that the inputs are already band-pass, and the receptive field connectivity is a priori limited.

We will see in the next lecture that cells in the visual cortex send their visual information to an incredibly complex, and yet structured collection of extra-striate areas. Any hypothesized function of striate cortex must eventually take into account what the information is to be used for. In the next lecture, we will give a quick overview of extra-striate visual cortex, and introduce the computational problem of estimating scene properties from image data.

In 1996, Olshausen and Field showed that one could derive a set of basis functions that have the same characteristics as the ensemble of visual simple cells in primary visual cortex by requiring two simple constraints:

1) One should be able to express the image as a weighted sum of the basis functions.

2) The total sum of activity across the ensemble should, on average, be small. This latter constraint is called "sparse coding". That is, a typical input image should activate a relatively small fraction of neurons in the ensemble.



$$\sum_{x,y}\left[I(x,y)-\sum_{i}\,a_i\,\phi_i(x,y)\right]^2 \;+\; \sum_{i} S\left(\frac{a_i}{\sigma}\right)$$

Principal components analysis

Introduction to PCA

Principal components analysis (PCA) is a statistical technique that is applied to an ensemble of n-dimensional measurements (vectors or in our case images). To do PCA, all one needs is the autocovariance matrix and a good PCA algorithm. Good because images are big enough (p=mxn), and the covariance is much bigger (p^2).

PCA finds a matrix that transforms the input vectors into output vectors, such that output elements are *no longer correlated* with each other. There is more than one matrix that will do this however, and PCA find the matrix which is a rigid *rotation* of the original coordinate axes, so it preserves orthogonality. (The Fourier transform is also a rotation.) Further, the new coordinates can be ordered in terms of variance. The new coordinates turn out to be eigenvectors of the covariance matrix. The directions or eigenvectors with the biggest variances are called the principal components. So the dominant principal component has the most variance, and so forth. For data that are highly redundant, PCA can be used to eliminate dimensions that account for little of the total variance.

PCA is important in computational models of visual processing (See Wandell, pages 254-258). For example, PCA has been used to account for and model:

opponent color processing

visual cortical cell development

efficient representation of human faces

face recognition given variability over illumination

internal model of objects for visual control of grasping

There has been considerable growth in the area of theoretical neural networks and PCA. An introduction to some of the ideas is given in the optional section below.

Standard computer statistical packages provide the tools for doing PCA on large data sets.Below we try to provide intuition and background into the computation of principal components.

Statistical model of a two-variable input ensemble

Consider a two variable system whose inputs are correlated. The random variable, **rv**, is a 2D vector. The scatter plot for this vector has a slope of Tan[theta] = 0.41. The variances along the axes are 4 and $.25^2$ (.0625). **`gprincipalaxes`** is a graph of the principal axes which we will use for later comparison with simulations. **`ContinuousDistributions.m`** is a *Mathematica* package that provides routines for sampling from a Gaussian (or Normal) distribution, rather than the standard uniform distribution that **Random[]** provides.

In[1]:=
```
ndist = NormalDistribution[0,1];theta = Pi/8;
bigvar = 4.0; smallvar = 0.25
alpha = N[Cos[theta]]; beta = N[Sin[theta]];
rv :=
{bigvar x1 alpha + smallvar y1 beta,
bigvar x1 beta - smallvar y1 alpha}
/.{x1-> Random[ndist],y1-> Random[ndist]};

gprincipalaxes = Plot[{x beta, x (-1/beta)}, {x,-4,4},
    PlotRange->{{-4,4},{-4,4}},
    PlotStyle->{RGBColor[1,0,0]},
    AspectRatio->1,DisplayFunction->Identity];
```

Out[2]=
```
0.25
```

General::spell1 :
Possible spelling error: new symbol name "beta" is similar to existing symbol "Beta". More…

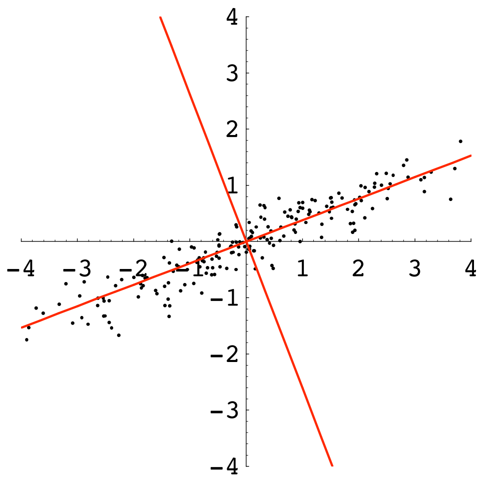
```
Syntax::sntxf: "/" cannot be followed by ".{x1-> Random[ndist],y1->
Random[ndist]};". More…
```

x1 and **y1** are correlated. Let's view a scatterplot of samples from these two correlated Gaussian random variables.

In[5]:=
```
npoints = 200;
rvsamples = Table[rv,{n,1,npoints}];
```

```
g1 = ListPlot[rvsamples,PlotRange->{{-4,4},{-4,4}},
    AspectRatio->1,
    DisplayFunction->Identity];
```

```
Show[g1,gprincipalaxes, DisplayFunction-> $DisplayFunction];
```



Standard Principal Components Analysis (PCA)

Let E[•] stand for the expected or average of a random variable, •. The covariance matrix of a of vector random variable, **x**, is: E[[**x**-E[**x**]][**x**-E[**x**]]^T]. Let's compute the autocovariance matrix for **rv**. The calculations are simpler because the average value of **rv** is zero. As we would expect, the matrix is symmetric:

```
autolist = Table[
    Outer[Times,rvsamples[[i]],rvsamples[[i]]],
        {i,Length[rvsamples]}];
MatrixForm[auto=
    Sum[autolist[[i]],
    {i,Length[autolist]}]/Length[autolist]]
Clear[autolist];
```

```
3.13451   1.24297
1.24297   0.5646
```

The variances of the two inputs (the diagonal elements) are due to the projections onto the horizontal and vertical axis of the generating random variable.

Now we will calculate the eigenvectors or the autocovariance matrix

```
MatrixForm[eigauto = Eigenvectors[auto]]
```

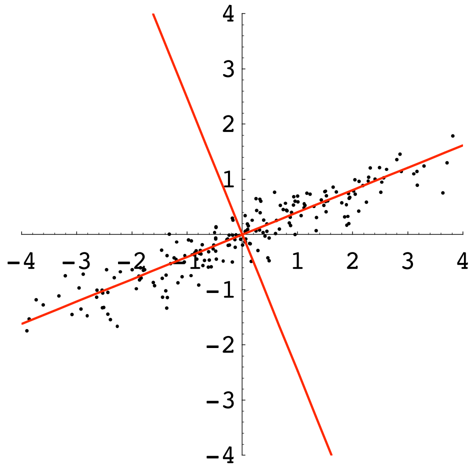
```
0.927025   0.375
-0.375     0.927025
```

Remember that the rows of a symmetric matrix are orthogonal. You can check that.

Let's graph the principal axes corresponding to the eigenvectors of the autocovariance matrix together with the scatterplot we plotted earlier.

```
gPCA =
Plot[{eigauto[[1,2]]/eigauto[[1,1]] x,
eigauto[[2,2]]/eigauto[[2,1]] x},
        {x,-4,4}, AspectRatio->1,
        DisplayFunction->Identity,
        PlotStyle->{RGBColor[1,0,0]}];
```

```
Show[g1,gPCA,DisplayFunction->$DisplayFunction];
```



The eigenvalues give the ratio of the variances of the projections of the random variables **rv[[1]]**, and **rv[[2]]** along the principal axes:

```
eigvalues = Eigenvalues[auto]
```

```
{3.63731, 0.0617931}
```

The projections along the principal axes are now **decorrelated**. We can see this by calculating the autocovariance matrix of the projected values:

```
autolist =
Table[
Outer[Times,eigauto.rvsamples[[i]],
            eigauto.rvsamples[[i]]],
            {i,Length[rvsamples]}];
MatrixForm[Chop[
    Sum[autolist[[i]],
    {i,Length[autolist]}]/Length[autolist]]]
Clear[autolist];
```

```
3.63731     0
0           0.0617931
```

Note that the off-diagonal elements (the terms that measure the covariation of the transformed random variables) are zero. Further, because the variance of one of the projections is near zero, one can in fact dispense with this component and achieve a good approximate coding of the data with just one coordinate.

PCA and natural images

■ Break a large image into a series of subimages.

The idea is that each subimage will be used as a statistical sample. We compute the outer product of each, and then average all 16 to get an estimate of the autocovariance matrix.

```
<< Statistics`MultiDescriptiveStatistics`
```

```
nregions = 16; swidth = width / nregions;
```

```
subface = Table[Take[granite, {i * swidth + 1, i * swidth + swidth},
    {j * swidth + 1, j * swidth + swidth}], {i, 0, nregions - 1},
   {j, 0, nregions - 1}];
```

```
subfacelist = Table[0.0, {256}];
Table[subfacelist[[i + 16 * (j - 1)]] = N[Flatten[subface[[i, j]]]],
 {i, 1, 16}, {j, 1, 16}];
```

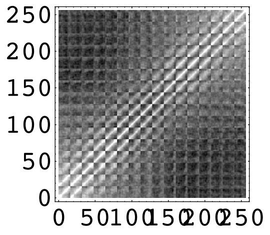
Subtract off the mean.

```
subfacelist2 = Table[subfacelist[[i]] - Mean[subfacelist[[i]]],
   {i, 1, 256}];
```

■ Calculate the autocovariance matrix

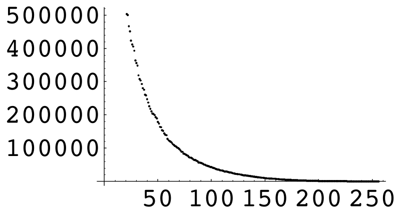
```
temp = Table[0.0, {256}, {256}];
For[i = 1, i < Dimensions[subfacelist][[1]], i++,
  temp = N[Outer[Times, subfacelist2[[i]], subfacelist2[[i]]]] + temp;
 ];
```

```
ListDensityPlot[temp, Mesh → False];
```



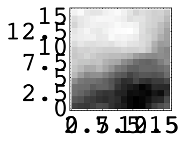
■ **Calculate the eigenvectors and eigenvalues of the autocovariance matrix**

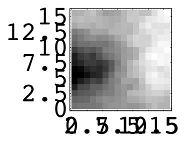
```
eigentemp = Eigenvectors[temp];
eigenvaluestemp = Eigenvalues[temp];
ListPlot[Chop[eigenvaluestemp]];
```

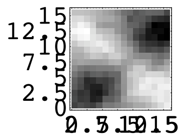


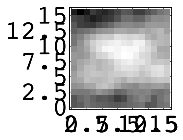
■ **Display the first 32 eigenvectors as "eigenpictures"**

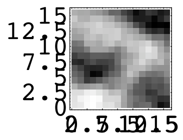
```
Table[ListDensityPlot[Partition[eigentemp[[i]], 16], Mesh → False],
  {i, 1, 32}];
```

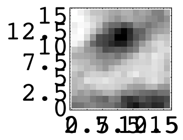


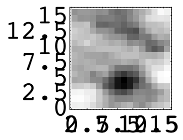


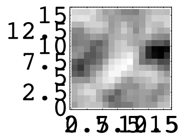


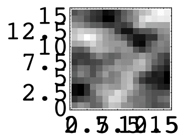


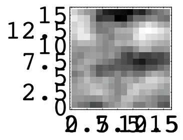


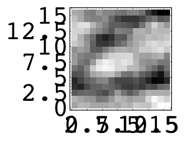


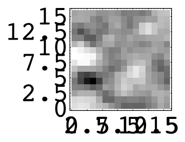


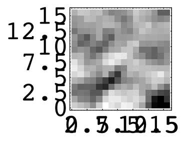


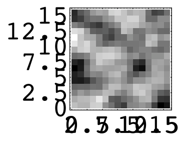


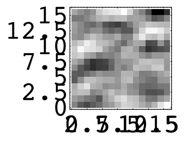


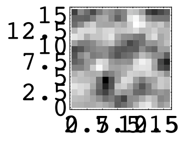


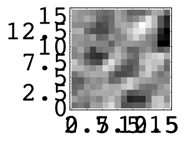


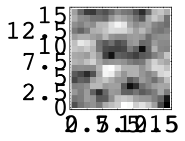


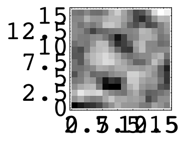


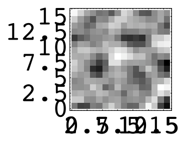


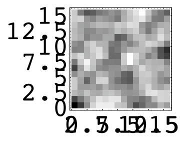


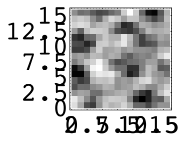


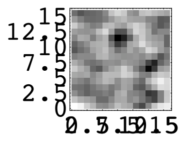


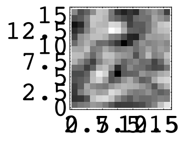


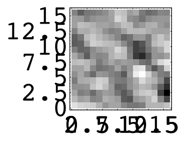


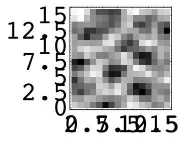


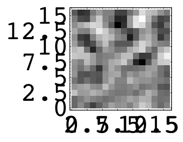


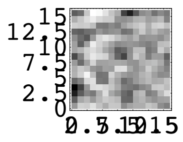


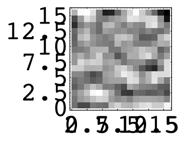


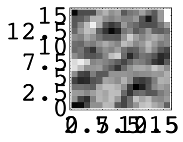


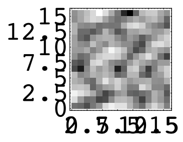


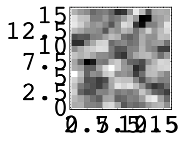












How good is a 2nd order model of natural images?

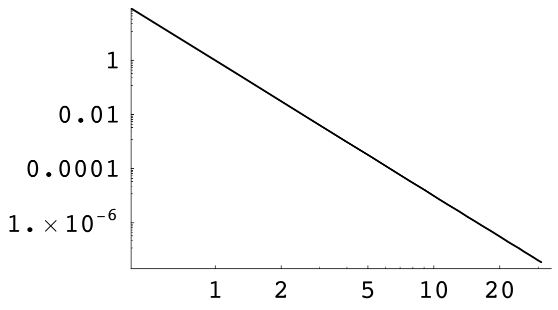
Let's construct a 2nd order generic generative statistical model of images and see what the samples look like.

Random Fractals

Random fractals are a crude but good statistical models for the amplitude spectra certain classes of natural images. 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 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 **Block[ ]** routine later.

```
size = 64;
hsize = size / 2;
fwidth = 2 * hsize; hfwidth = fwidth / 2;
```

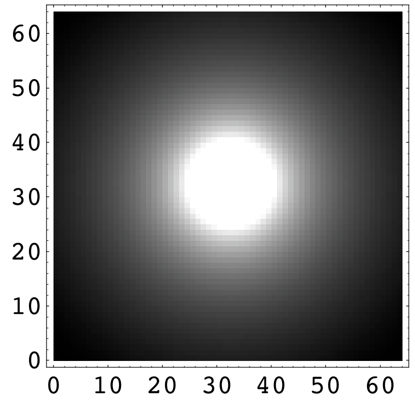
```
q = 2.5;
LogLogPlot[If[(i ≠ 0 || j ≠ 0), 1 / (i * i + 0 * 0) ^ (q), 1 / (2) ^ (q)],
  {i, 0, hfwidth - 1}];
```



■ **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.**

```
fractalfilter[D_] :=
Block[ {q,i,j,mat},
    q = 4 - D;
    mat = Table[If[(i != 0 || j!= 0),
        1/(i*i + j*j)^(q), 1/(2)^(q)],
    {i,-hfwidth,hfwidth-1},{j,-hfwidth,hfwidth-1}];
    mat = RotateRight[mat,{hfwidth,hfwidth}];
    Return[mat];
    ];
```

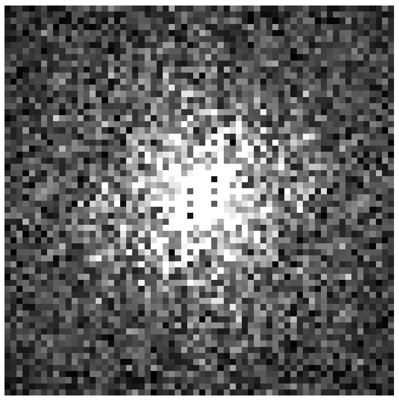
```
ListDensityPlot[RotateLeft[fractalfilter[3.5], {hfwidth, hfwidth}],
  Mesh → False];
```



Here is the amplitude spectrum plot for a random fractal image:

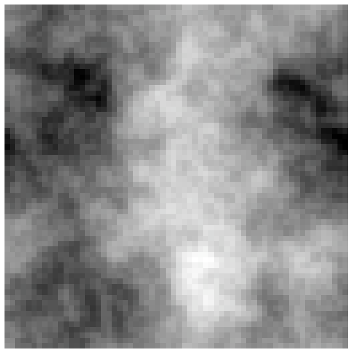
```
randomspectrum = Abs[temp = Fourier[Table[Random[], {width}, {width}]]];
randomphase = Arg[temp];
```

```
ffilt = fractalfilter[3.5] randomspectrum;
ListDensityPlot[RotateRight[ffilt,{hfwidth,hfwidth}], Mesh->False,
Frame->False];
```

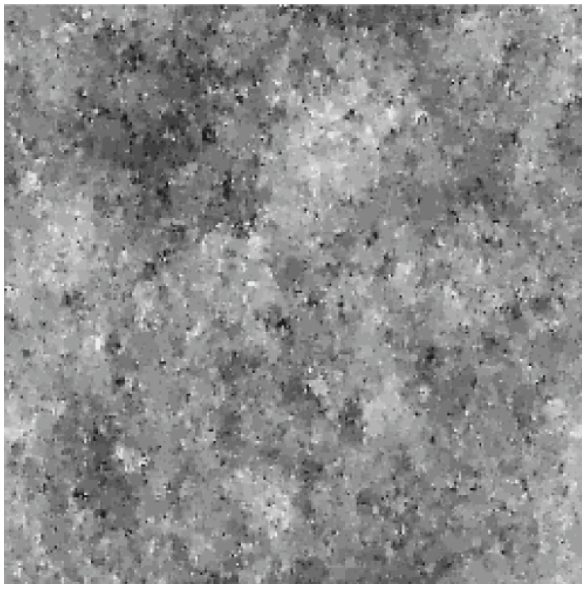


■ **Here is a random fractal image, with D = 3.2**

```
ListDensityPlot[Chop[
InverseFourier[
fractalfilter[3.2] randomspectrum Exp[I randomphase]]],
Mesh->False,Frame->False];
```



Can one do better? Yes. See the sample below from the paper by: Zhu, S. C., & Mumford, D. (1997). Prior Learning and Gibbs Reaction-Diffusion. *IEEE Trans. on PAMI, 19*(11), 1236-1250.



Next time

- **Edge detection**

Appendices

Some exercises on 1rst and 2nd order spatial filter statistics

For a discussion of 1rst and 2nd order spatial filter statistics, see: Simoncelli and Olshausen, 1999.

Initialization stuff

```
Off[General::spell1];
DeclarePackage["Statistics`DataManipulation`", {"BinCounts"}];
<< Statistics`DescriptiveStatistics`
```

Kurtosis

Some useful functions: scale256, histogram, argmax

```
scale256[image_] := Module[{α , β},
   α = 255 / (Max[image] - Min[image]);
   β = -α Min[image];
   Return[α image + β];];
```

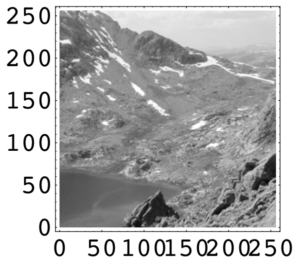
```
histogram[image_] := Module[{histx},
   histx = BinCounts[Flatten[image], {0, 255, 1}];
   Return[N[histx / Plus @@ histx]];
  ];
```

```
argmax[x_] := Position[x, Max[x]][[1, 1]];
```

Input image: " alpine.jpg" is a 256x256 array of gray-levels

```
 alpine = Import[Experimental`FileBrowse[False]];
```

```
 alpine =  alpine /. Graphics → List;
 alpine =  alpine[[1, 1]];
size = Dimensions[ alpine][[1]];
hsize = size / 2;
ListDensityPlot[ alpine, Mesh → False];
```



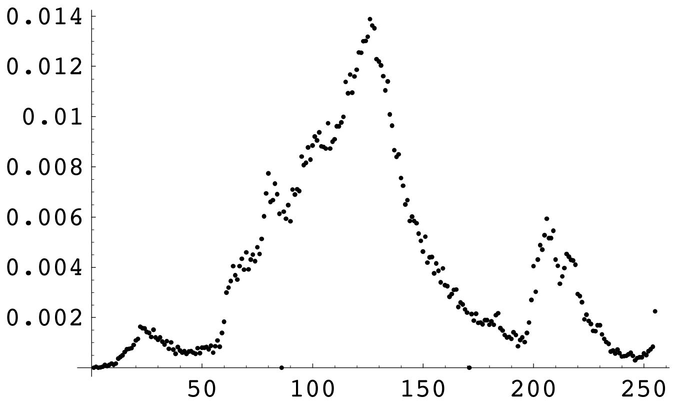
■ Problem 1: Histogram of alpine256

Expand alpine so that the minimum and maximum gray-levels are 0 and 255 respectively. Call it **alpine256**. Plot the histogram of **alpine256**.

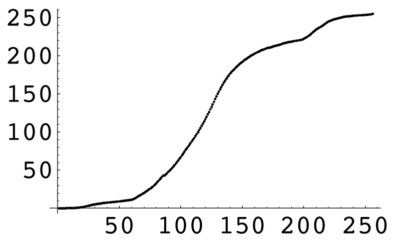
■ Answer 1:

```
 alpine256 = scale256[ alpine];
```

```
hist0 = histogram[ alpine256];
g0 = ListPlot[hist0];
```



```
cumulhist0 = FoldList[Plus, hist0[[1]], hist0];
g1 = ListPlot[255 * cumulhist0];
```



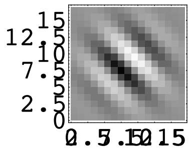
■ Problem 2: Histogram of the convolution of sgabor[x,y] with alpine256[x,y] = sgabor⊗alpine256

Define an 8x8 pixel sine-phase gabor filter:

sgabor[x_,y_,fx_,fy_,sig_]:=

N[Exp[(-x^2-y^2)/(2 sig*sig)] Sin[2 Pi (fx x+fy y)]]; ,

where fx = fy = 1/8. And sig = 4.



Convolve alpine256 with this filter. As with alpine, scale the resulting image so that [min,max] = [0,255]. Plot the histogram of the resulting "neural image".

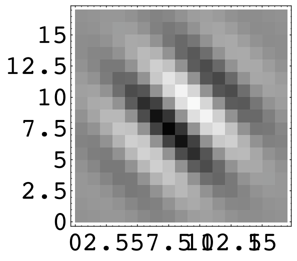
Answer 2:

Gabor filter

```
sgabor[x_, y_, fx_, fy_, sig_] :=
  N[Exp[(-x^2 - y^2) / (2 sig * sig)] Sin[2 Pi (fx x + fy y)]];
```

```
fsize = 16;
filter = Table[sgabor[(i - fsize / 2), (j - fsize / 2), 1 / 8, 1 / 8, 4],
  {i, 0, fsize}, {j, 0, fsize}];
filter = Chop[filter];
ListDensityPlot[filter, Mesh → False, PlotRange → {-1, 1}];
```

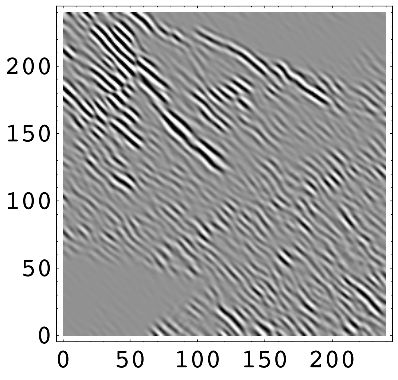
```
General::spell :
  Possible spelling error: new symbol name "fsize" is similar to existing symbols {hsize, size}. More…
```



Convolution

```
falpine = ListConvolve[filter, alpine256];
falpine256 = scale256[falpine];
```

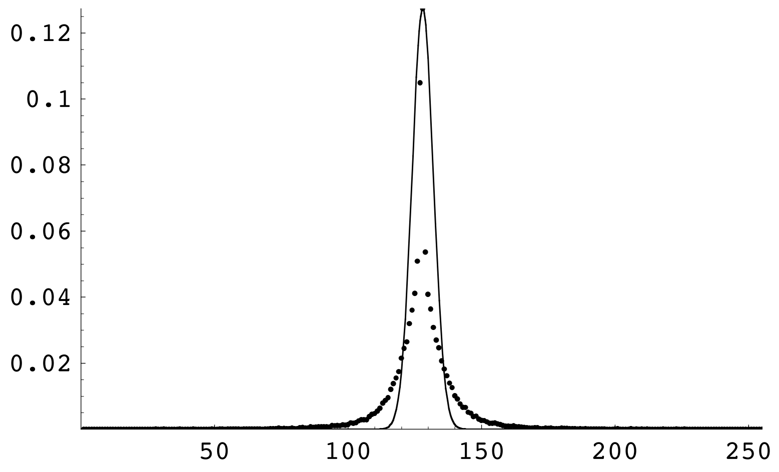
```
ListDensityPlot[falpine256, Mesh → False];
```



Histogram

```
hist1 = histogram[falpine256];
a = Max[hist1];
u = argmax[hist1];
s = 4;
g1 = ListPlot[hist1, PlotRange → {{0, 255}, {0, a}},
   DisplayFunction → Identity];
```

```
g2 = Plot[a Exp[- (x - u) ^2 / (2 s^2)], {x, 0, 255},
   PlotRange → {{0, 255}, {0, a}}, DisplayFunction → Identity];
Show[g1, g2, DisplayFunction → $DisplayFunction];
```



```
cumulhist1 = FoldList[Plus, hist1[[1]], hist1];
g11 = ListPlot[255 * cumulhist1];
```

■ Problem 3: Excess kurtosis: alpine256 vs. sgabor⊗alpine256

Skewness and kurtosis are statistics describing the shape of a distribution. Skewness is a measure of asymmetry. Kurtosis compares the concentration of data around the peak to the tails versus the concentration in the flanks.

`Kurtosis` is calculated by dividing the fourth central moment by the square of the variance of the data. `KurtosisExcess` is shifted so that it is zero for the normal distribution, positive for distributions with a prominent peak and heavy tails, and negative for distributions with prominent flanks.

Calculate excess kurtosis for alpine256 and falpine256.

Answer 3:

```
KurtosisExcess[Flatten[N[alpine256]]]
KurtosisExcess[Flatten[N[falpine256]]]
```

```
0.0326037
```

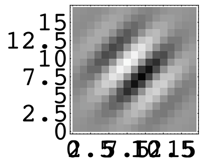
```
13.2636
```

2nd order statistics spatial filter

Problem 5: Joint histogram of two overlapping orthogonal filters

Answer 5:

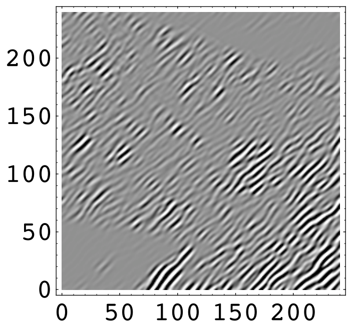
```
fsize = 16;
filter2 = Table[sgabor[(i - fsize / 2), (j - fsize / 2), 1 / 8, -1 / 8, 4],
   {i, 0, fsize}, {j, 0, fsize}];
filter2 = Chop[filter2];
ListDensityPlot[filter2, Mesh → False, PlotRange → {-1, 1}];
```



Convolution

```
falpineB = ListConvolve[filter2, alpine256];
falpine256B = scale256[falpineB];
```

```
ListDensityPlot[falpine256B, Mesh → False];
```

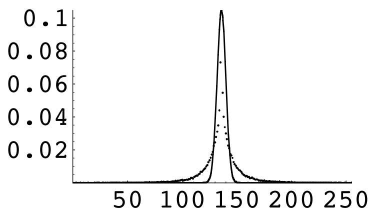


■ 1D Histogram

```
hist2 = histogram[falpine256B];

a = Max[hist2];
u = argmax[hist2];
s = 4;
g3 = ListPlot[hist2, PlotRange → {{0, 255}, {0, a}},
   DisplayFunction → Identity];
```

Non-gaussian

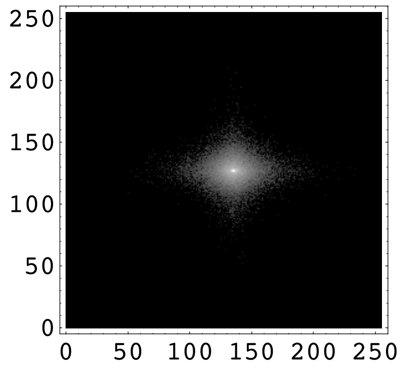
```
g2 = Plot[a Exp[-(x - u)^2 / (2 s^2)], {x, 0, 255},
   PlotRange → {{0, 255}, {0, a}}, DisplayFunction → Identity];
Show[g3, g2, DisplayFunction → $DisplayFunction];
```



■ 2D histogram

```
temp = Transpose[{Flatten[falpine256], Flatten[falpine256B]}];
twoDhist = BinCounts[temp, {0, 255, 1}, {0, 255, 1}];
twoDhist2 = Map[If[# == 0, 0, Log[#]] &, twoDhist, {2}];
```

```
ListDensityPlot[twoDhist2, Mesh → False];
```



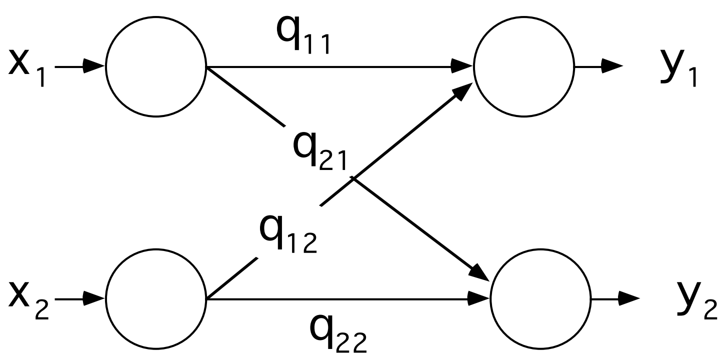
```
ListContourPlot[twoDhist2, ContourShading → False, Contours → 10];
```

Neural networks and principal components

■ Neural network model using Hebb together with Oja's rule for extracting the dominant principal component

Oja, E. (1982). A simplified neuron model as a principal component
analyzer. Journal of Mathematical Biology, 15, 267-273.

Consider the following linear neural network. The input and output values are represented by vectors **x**, and **y** respectively. The connection weights are represented by matrix **Q**.



We will combine the outer product form of Hebb's rule, together with Oja's modification. Without Oja's rule, the Hebb rule does not place a limit on the size of the weights. Recall that Oja's rule constrains the sum of the squares of the weights to approach 1. We will set the intial values of the weight matrix to random values between 0 and 1.

```
npoints = 400; p1 = {}; α = 0.01;
```

```
size = 2;
Q = Table[Random[], {size}, {size}];
```

Note that a space in *Mathematica* between two expressions does an element by element multiplication. We use this notation as economical way of writing Oja's rule. An example is:

```
MatrixForm[{{a,b},{c,d}} {x,y}]
```

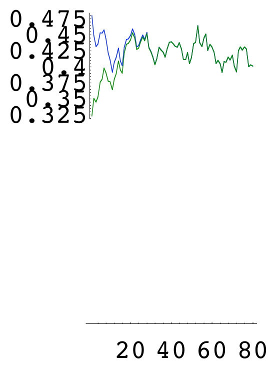
```
a x   b x
c y   d y
```

Note that this different from standard matrix multiplication.

```
For[i=1,i<=npoints,i++,
            x = rv; y = Q.x;
            Q = Q + α (Outer[Times,y,x] - Q y y);
            If[Mod[i,5]==0,
                p1 = Join[p1,{{Q[[1,2]]/Q[[1,1]], Q[[2,2]]/Q[[2,1]] }}]];

];
```

Let's plot the slopes of projection axes as a function of iterations. We've sampled every 5th value, using **Mod[i,5]**, and stored it in **p1**.

```
ListPlot[Map[#[[2]]&,p1], AxesOrigin->{0,0}, PlotJoined->True,
    DisplayFunction->Identity,
    PlotStyle->{RGBColor[0,.5,0]}];
ListPlot[Map[#[[1]]&,p1], AxesOrigin->{0,0}, PlotJoined->True,
    DisplayFunction->Identity,
    PlotStyle->{RGBColor[0,0,1]}];
Show[%,%%, DisplayFunction->$DisplayFunction];
```

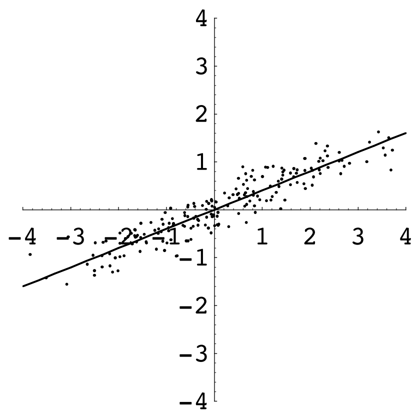


There is some random fluctuation in the weights. We can obtain more stability by having a timeconstant over which the Hebbian term and the variance of y are averaged.

We can see how well the coordinate transformation fits the principal axes of a sample scatter plot:

```
gnetwork = Plot[
    {s p1[[Length[p1]]][[1]], s p1[[Length[p1]]][[2]]},
    {s, -4, 4}, PlotRange->{{-4,4},{-4,4}},AspectRatio->1,
    DisplayFunction->Identity];
```

```
Show[gnetwork,g1,DisplayFunction->$DisplayFunction];
```



You can verify that the network does a good job of extracting the principal component. Recall that the slope for the population distribution is **Tan[theta]**:

```
N[Tan[theta]]
```

```
0.414214
```

The only problem with this network is that having two output neurons is redundant--they both pull out the same principal component--the dominant axis. The slopes for both are:

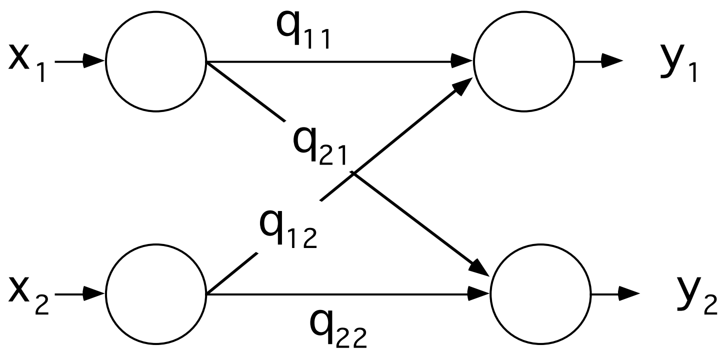
```
p1[[Length[p1]]]
```

```
{0.400705, 0.400705}
```

■ A generalization of Oja's rule for extracting all of the principal components with a "Neural network" (Sanger, 1989)

Sanger, T. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural Networks, 2, 459-473.

We will use the same network as in the above example. However, the learning rule will be asymmetric. The generalization of Oja's term is given by: **`LT Outer[Times,y,y]).Q,`** where **LT** is a lower triangular matrix. The entries above the diagonal are all zero, and the entries below and including the diagonal are one.



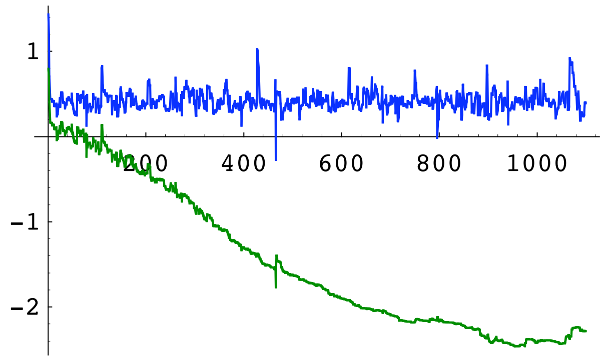
```
size = 2;
LT = Table[If[i>=j,1,0],{i,size},{j,size}];
```

```
npoints = 1200;
p1 = {}; α = 0.1;
```

```
Q = Table[Random[], {size}, {size}];
```

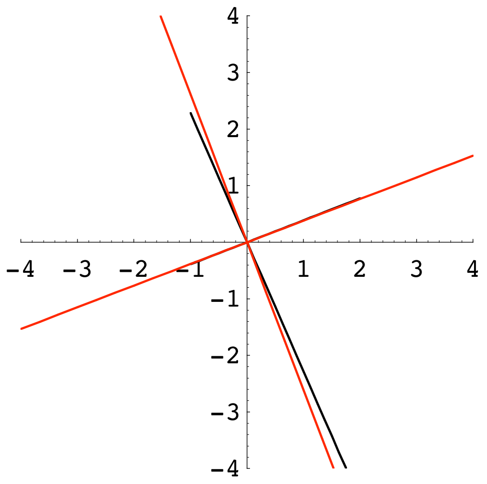
```
For[i=1,i<=npoints,i++,
    x = rv; y = Q.x;
    deltaQ = (Outer[Times,y,x] - (LT Outer[Times,y,y]).Q);
    Q = Q + α deltaQ;
    If[Mod[i,1]==0,
                p1 = Join[p1,{{Q[[1,2]]/Q[[1,1]], Q[[2,2]]/Q[[2,1]] }}]];

];
```

```
ListPlot[Map[#[[2]]&,p1], AxesOrigin->{0,0}, PlotJoined->True,
    DisplayFunction->Identity,PlotStyle->{RGBColor[0,.5,0]}];
ListPlot[Map[#[[1]]&,p1], AxesOrigin->{0,0}, PlotJoined->True,
    DisplayFunction->Identity,PlotStyle->{RGBColor[0,0,1]}];
Show[%,%%, DisplayFunction->$DisplayFunction];
```



Let's plot up the transformation axes of the network and compare them with the principal component axes:

```
gnetwork = Plot[
{s p1[[Length[p1]]][[1]], s p1[[Length[p1]]][[2]]},
    {s, -1, 2}, PlotRange->{{-4,4},{-4,4}},AspectRatio->1,
    DisplayFunction->Identity];
Show[gnetwork,gprincipalaxes,DisplayFunction->$DisplayFunction];
```



References

Adelson, E. H., Simoncelli, E., & Hingorani, R. (1987). Orthogonal Pyramid Transforms for Image Coding. Proc. SPIE - Visual Communication & Image Proc. II, Cambridge, MA.

Barlow, H. B., & Foldiak, P. (1989). Adaptation and decorrelation in the cortex. In C. Miall, R. M. Durban, & G. J. Mitchison (Ed.), The Computing Neuron Addison-Wesley.

Barlow, H. (1990). Conditions for versatile learning, Helmholtz's unconscious inference, and the task of perception. Vision Research, 30(11), 1561-1572.

Belhumeur, P. N., & Mumford, D. (1992). *A Bayesian Treatment of the Stereo Correspondence Problem Using Half-Occluded Regions*. Paper presented at the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Champaign, Illinois.

Campbell, F. W., & Robson, J. R. (1968). Application of Fourier Analysis to the Visibility of Gratings. 197, 551-566.

Crick, F. (1984). Function of the Thalamic Reticular Complex: The Searchlight Hypothesis. 81, 4586-4590.

Cumming, B. (1997). Stereopsis: how the brain sees depth. *Curr Biol, 7*(10), R645-647.

Cumming, B. G., & DeAngelis, G. C. (2001). The physiology of stereopsis. *Annu Rev Neurosci, 24*, 203-238.

Cumming, B. (2002). Stereopsis: where depth is seen. *Curr Biol, 12*(3), R93-95.

Daugman, J. G. (1988). An information-theoretic view of analog representation in striate cortex. In Computational Neuroscience Cambridge, Massachusetts: M.I.T. Press.

DeValois, R., Albrecht, D. G., & Thorell, L. G. (1982). Spatial frequency selectivity of cells in macaque visual cortex. Vision Research, 22, 545-559)

Dobbins, A., Zucker, S. W., & Cynader, M. S. (1987). Endstopped neurons in the visual cortex as a substrate for calculating curvature. Nature, 329(6138), 438-441.

Heeger, D. J. (1991). Nonlinear model of neural responses in cat visual cortex. In M. &. M. Landy A. (Ed.), Computational Models of Visual Processing (pp. 119-133). Cambridge, Massachusetts: M.I.T. Press.

Hubel, D. H., & Wiesel, T. N. (1968). Receptive Fields and Functional Architecture of Monkey Striate Cortex. London: 215-243.

Koenderink, J. J., & van Doorn, A. J. (1990). Receptive field families. Biol. Cybern., 63, 291-297.

Linsker, R. (1990). Perceptual neural organization: some approaches based on network models and information theory. Annual Review of Neuroscience, 13, 257-281.

Livingstone, M. S., & Hubel, D. H. (1984). Anatomy and Physiology of a Color System in the Primate Visual Cortex. 4(1), 309-356;

Livingstone, M. S., & Hubel, D. H. (1987). Psychophysical Evidence for Separate Channels for the Perception of Form, Color, Movement and Depth. The Journal of Neuroscience, 7(11), 3416-3468).

Mechler, F., & Ringach, D. L. (2002). On the classification of simple and complex cells. *Vision Res, 42*(8), 1017-1033.

Morrone, M. C., & Burr, D. (1988). Feature detection in human vision: a phase dependent energy model. Proceedings of the Royal Society, London, 235, 221-245.

Mumford, D. (1991). On the computational architecture of the neo-cortex: I. The role of the thalamo-cortical loop. Biological Cybernetics, 65, 135-145.

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature, 381, 607-609.

Olshausen, B. A., & Field, D. J. (2004). Sparse coding of sensory inputs. *Curr Opin Neurobiol, 14*(4), 481-487.

Poggio, G., F., & Poggio, T. ,1984. The Analysis of Stereopsis. Annual Review of Neuroscience, 7, 379-412).

Poggio, T. (1984). Vision by Man and Machine. Scientific American, 250, 106-115.

Poggio, G. F., & Talbot, W. H. (1981). Mechanisms of Static and Dynamic Stereopsis in Foveal Cortex of the Rhesus Monkey. 315, 469-492.

Sillito, A. M., Jones, H. E., Gerstein, G. L., & West, D. C. (1994). Feature-lined synchronization of thalamic relay cell firing induced by feedback from the visual cortex. Nature, 369, N. 9, 479-482.

Simoncelli, E. P., & Olshausen, B. A. (2001). Natural image statistics and neural representation. Annu Rev Neurosci, 24, 1193-1216

van der Schaaf, A., & van Hateren, J. H. (1996). Modelling the power spectra of natural images: statistics and information. *Vision Res, 36*(17), 2759-2770.

von der Heydt, R., Zhou, H., & Friedman, H. S. (2000). Representation of stereoscopic edges in monkey visual cortex. *Vision Research, 40*(15), 1955-1967.

Yuille, A. L., Geiger, D., & Bülthoff, H. H. (1991). Stereo integration, mean field theory and psychophysics. *Network, 2*, 423-442.

Zhu, S. C., & Mumford, D. (1997). Prior Learning and Gibbs Reaction-Diffusion. *IEEE Trans. on PAMI, 19*(11), 1236-1250.