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

Initialize

In[45]:=
```
Off[General::spell1];
SetOptions[ArrayPlot, ColorFunction → "GrayTones", DataReversed → True,
  Frame → False, AspectRatio → Automatic, Mesh → False,
  PixelConstrained → True, ImageSize → Small];
SetOptions[ListPlot, ImageSize → Small];
```

In[48]:=
```
nbinfo = NotebookInformation[EvaluationNotebook[]];
dir =
 ("FileName" /. nbinfo /. FrontEnd`FileName[d_List, nam_, ___] :>
    ToFileName[d]);
```

■ Histogram

In[49]:=
```
myhistogram[image_] := Module[{histx},
   histx = BinCounts[Flatten[image], {0, 255, 1}];
   Return[N[histx / Plus @@ histx]];
  ];
```

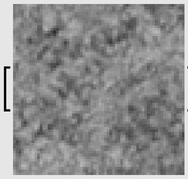
■ Entropy

In[50]:=
```
entropy[probdist_] := Plus @@ (If[# == 0, 0, -# Log[2, #]] & /@ probdist)
```

■ Image data

In[51]:=

```
granite = ImageData[
```



```
];

{N[Mean[Flatten[granite]]]
 , N[StandardDeviation[Flatten[granite]]],
 width = Dimensions[granite][[1]]}
```

Out[52]=

```
{0.507939, 0.0773643, 64}
```

Outline

Last time

■ First-order intensity statistics. Explain point non-linearities in terms of histogram equalization natural image intensities

The assumption was that the cell’s output range was effectively fixed to encode N levels. This doesn’t mean that the output is digitize, but rather than noise limits the number of resolvable levels.

And that an efficient use of those levels was to not favor one over another.

The question then was how best to allocate input contrasts to those levels, so that probability of using an output value was equal across all N levels.

The answer was to use the cumulative distribution function of the input histogram as the mapping function. If the input is bell-shaped, a sigmoidal non-linearity results.

■ Introduction to 2nd order statistics

A difference histogram is a “marginal” distribution, which means the histogram you get when you project higher-dimensional data points onto a lower-dimensional axis. If natural images were gaussian, the marginals would be too. But in general, they are not. The differences concentrate near zero (“blue sky effect”), but also get spread out in the tails (big jumps at edges). Difference histograms for natural images are said to have “excess kurtosis”--i.e. a normal distribution has kurtosis of 3, but natural images tend to have larger values.

This doesn’t just show up with simple differences between nearby pixels, but more generally with any derivative-type filter in which uniform values map to zero. (i.e. because the excitatory and inhibitory contributions cancel out). For example, consider the $\nabla^2 G$ filter:

In[26]:=

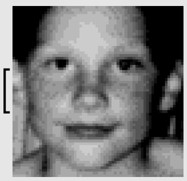
```
LaplacianGaussianFilter[
```

```
, 2] // ImageAdjust
```

Out[26]=

In[11]:=

```
Kurtosis[Flatten[ImageData[LaplacianGaussianFilter[
```



```
, 2]]]]
```

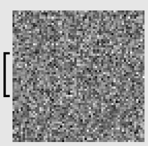
Out[11]=

```
5.69827
```

compared with the kurtosis of gaussian noise:

```
igaussiannoise = RandomImage[NormalDistribution[0, .2], {64, 64}] //
   ImageAdjust;

Kurtosis[Flatten[ImageData[LaplacianGaussianFilter[
```



```
, 2]]]]
```

Out[24]=

```
2.97776
```

Today

■ 2nd order spatial statistics and efficient coding

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

Efficient representation of information: the range problem

When we considered the rationale for a point-wise sigmoidal non-linearity, we assumed a fixed output range. But what if we could code the input efficiently so that a smaller range would do. This could lead to metabolic savings. First, let's see why there is a range problem.

We'll first consider the single-channel spatial filtering 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. There is the huge range of physical light energy, ranging from 10^{-6} to 10^{7} candelas/m^2 (a measure of luminance of a surface)--from the just visible to painfully bright. But the number of distinguishable levels can be much smaller over a large range of intensities.

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 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:

$$p\,(k\text{ spikes in }\Delta t) = \frac{e^{-\lambda\Delta t}\,\lambda\Delta t^{k}}{k!} \tag{1}$$

(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 then is to go from 200 levels to 14

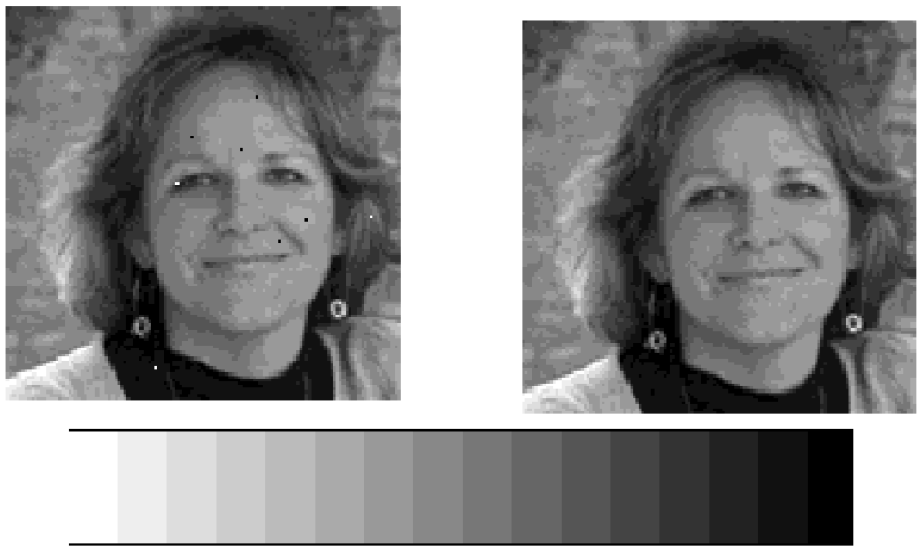
$\text{Log}_2 200 \to \text{Log}_2 14$, with minimal loss of information.

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. A duplex receptor system (rods and cones) helps, and sigmoidal compression. 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 where there are no spatial (or temporal) dependencies, 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 $10^{19,728}$ 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

We can build our intuitions be considering a space of 1-D images that, like natural images, is constrained to have similar nearby pixels. We start with a gray-level of 128, and then flip a coin to decide whether to increase or decrease the intensity of the next pixel. So nearest-neighbor pixels are close, but not identical in intensity.

■ 1-D Brownian images

In[53]:=

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

In[55]:=

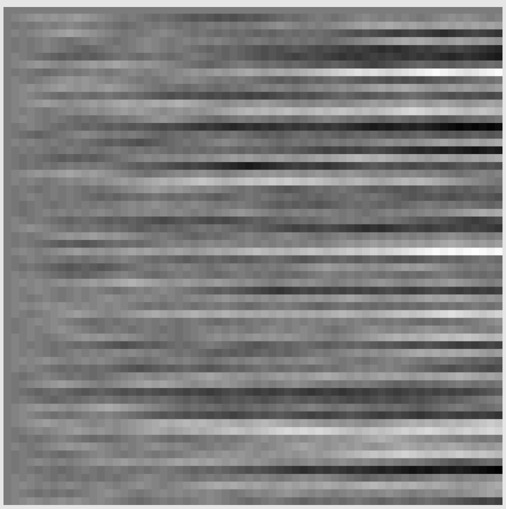
```
size = 64;
brown = N[Table[128, {i, 1, size }, {i, 1, size }]];
```

In[57]:=
```
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];
   ];
 ];
```

Visual each 1-D image using **Image[]**. Let's stack the images horizontally, one on top of the other:

In[58]:=
```
ImageRotate[Image[brown, ImageSize → Small] // ImageAdjust]
```

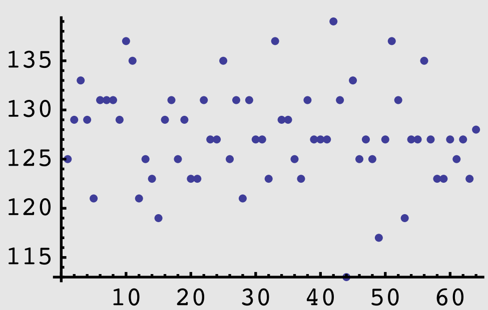
Out[58]=



Once we get away from the starting value of 128, along a vertical line, the intensities are quite random--the samples were drawn independently. The gray-levels from pixel to pixel are not correlated:

In[59]:=
```
ListPlot[brown[[32]], ImageSize → Small]
```

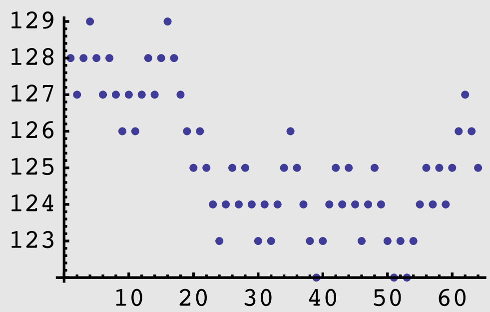
Out[59]=



In contrast, vertical lines show a degree of regularity:

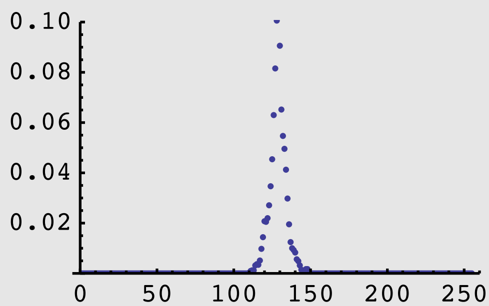
In[60]:=
```
ListPlot[Transpose[brown][[32]], ImageSize → Small]
```

Out[60]=



In[61]:=
```
histobrown = myhistogram[brown];
ListPlot[histobrown, PlotStyle → PointSize[0.015], PlotRange → {0, 0.1}]
entropy[histobrown]
```

Out[62]=



Out[63]=
```
4.34351
```

■ Efficient encryption code for 1-D brownian images

Let's encode the brownian images as the difference between neighboring pixel values:

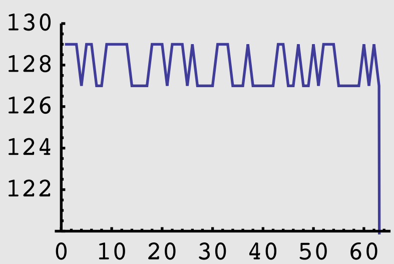
In[64]:=
```
codebrown = Table[0, {size}, {size}];
For[j = 1, j < size, j++,
  For[i = 1, i < size, i++,
    codebrown[[i, j]] = brown[[next[i], j]] - brown[[i, j]] + 128;
   ];
 ];
```

In[74]:=
```
ArrayPlot[codebrown, Mesh → False, PlotRange → {126, 130}]
```

Out[74]=

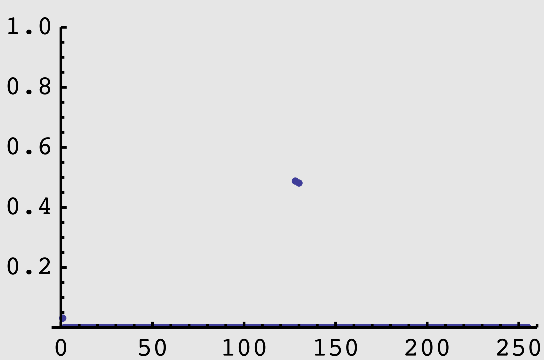
In[71]:=
```
ListPlot[codebrown[[32]], PlotRange → {120, 130}, Joined → True,
 ImageSize → Small]
```

Out[71]=



In[68]:=
```
histocodebrown = myhistogram[Flatten[codebrown]];
ListPlot[histocodebrown, PlotStyle → PointSize[0.015],
 PlotRange → {0, 1}]
entropy[histocodebrown]
```

Out[69]=



Out[70]=
```
1.16837
```

Second order statistics in natural images

■ Autocorrelation function

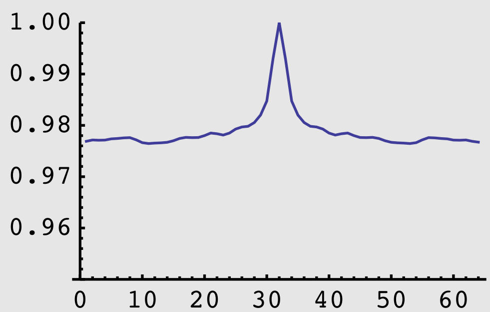
`ListCorrelate[`*ker*, *list*`]` computes $\sum_r K_r\, a_{s+r}$. Autocorrelation corresponds to $K_r \to a_r$: $\sum_r a_r\, a_{s+r}$.

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

■ Example with granite image

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

```
ListPlot[autogranite / Max[autogranite], Joined → True,
 PlotRange -> {.95, 1}, ImageSize → Small]
```



■ Covariance matrices, and the outer product

Recall that the covariance is: $\text{Cov}[X, Y] = E[[X - \mu_X]\ [Y - \mu_Y]]$, where X,Y are scalar random variables. The correlation gives a dimensionless measure of covariation, relative to the standard deviations: $\rho[X, Y] = \frac{\text{Cov}[X,Y]}{\sigma_X\, \sigma_Y}$.

Now let **X**={x_1 ...} and **Y** = {y_j ...} be vectors, and the lower case represent the scalar random variable elements. 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:

$$\text{Cov}[X, Y] = E[[X - \mu_X]\ [Y - \mu_Y]^T]$$

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 vector samples indexed by s, {X^s, Y^s}, we can estimate the covariance matrix as:

$$\frac{1}{M} \sum_{s=1}^{M} [X^s - \mu_X]\, [Y^s - \mu_Y]^T.$$

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 (See ProbabilityOverview.nb, Lecture 3)

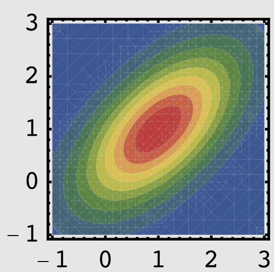
If the distribution is assumed to be multivariate gaussian, then the vector mean and covariance matrix fully determines the distribution. The multivariate gaussian is a generalization of the gaussian distribution to higher dimensions, in which the standard deviation is replaced by the covariance matrix. 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.

```
Σ = {{1, .6}, {.6, 1}};
μ = {1, 1};
ndist = MultinormalDistribution[μ , Σ ];
ContourPlot[PDF[ndist, {x, y}], {x, -1, 3}, {y, -1, 3},
 ImageSize → Tiny, ColorFunction → "DarkRainbow", ContourStyle → None]
```



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:

```
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 the classical statistical technique called Principal Components Analysis or PCA.

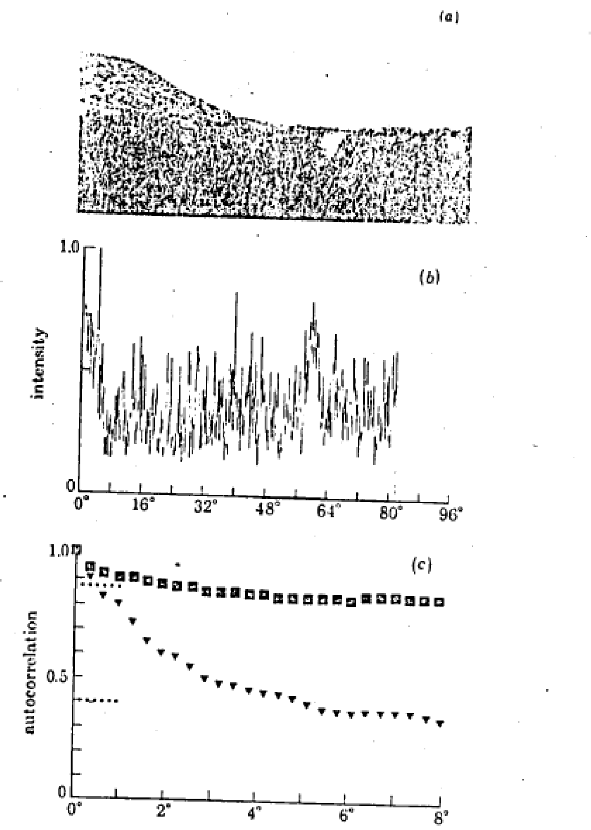
But first, let's look at some early and recent research that has sought to explain receptive field structure in terms of redundancy reduction.

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

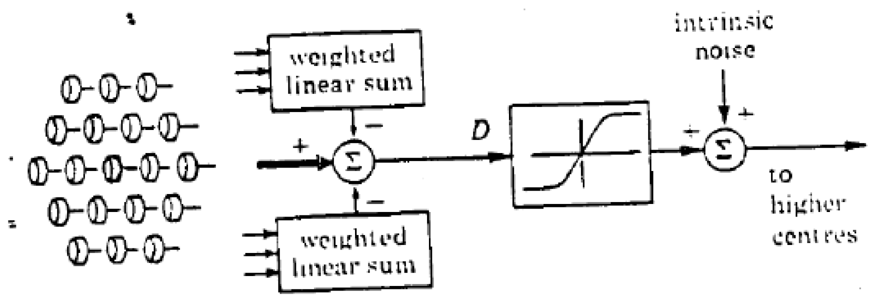


'he autocorrelation function of visual scenes. (a) The scene, a bed of reeds. (b) S
ty profile of scene, measured along a line joining the centres of the marker

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

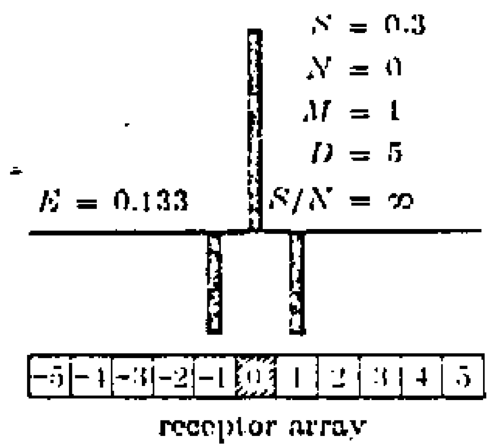
Linear neural network

They assumed a linear model:

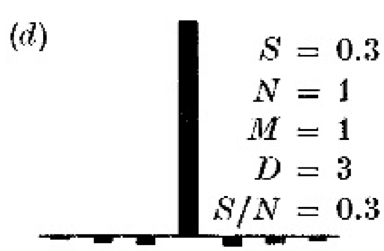


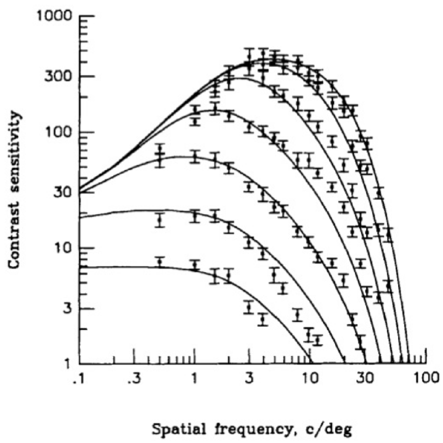
■ The result

Given the autocorrelation function, and the linear model, $R_j = \sum_i w_{ji} L_i = L_j - \sum_{i \neq j} H_{ji} L_i$, 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. Compare with the CSF functions for various light levels below. Srinivasan et al. was a "proof of concept". Atick & Redlich (1992) showed how the exact shape of the CSF as a function of mean light level could be accounted for in terms of efficient coding given the statistics of natural images (see below).





The left figure shows contrast thresholds for various light levels (from van Nes, & Bouman, M. A. (1967). Spatio modulation transfer in the human eye. J Opt Soc Am, 57(3), 401-406). The right figure is a replot of the left figure from: Atick, J. J., & Redlich, A. N. (1992). What does the retina know about natural scenes? Neural Computation, 4(2), 196-210. The solid lines show fits by Atick & Redlich based on an efficient coding model.

Efficient, sparse coding in 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.

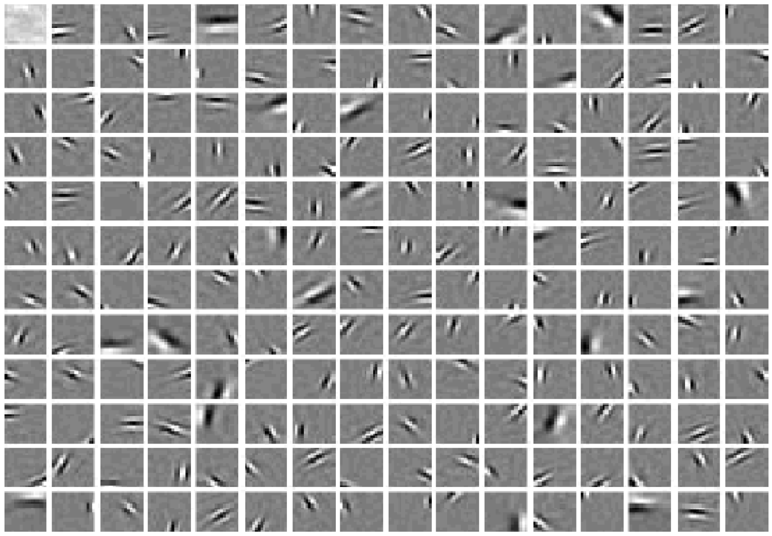
We will see later that cells in the visual cortex send their visual information to an incredibly complex, and yet structured collection of extra-striate areas. Any hypothesized function of striate cortex must eventually take into account what the information is to be used for. In the next lecture, we 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 I(x,y) as a weighted sum of the basis functions, $\{\phi_i\}$

2) The total 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. S() for example could be the absolute value of the activity a_i :

$$\sum_{x,y}\left[I(x,y)-\sum_i a_i\,\phi_i(x,y)\right]^2+\sum_i S(a_i) \tag{2}$$



■ Adaptation

Human orientation and spatial frequency selectivity changes with adaptation. Adaptation has been interpreted as an optimal change to new conditions in the input image statistics. (e.g. see, Wainwright, M. J. (1999). Visual adaptation as optimal information transmission. Vision Research, 39, 3960--3974.)

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];
```

Out[2]=

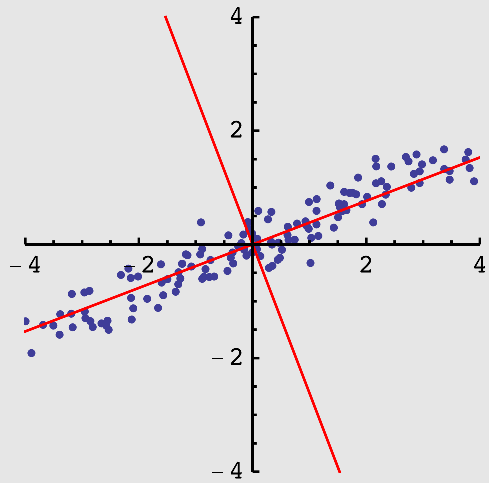
```
0.25
```

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

In[8]:=

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

```
g1 = ListPlot[rvsamples,PlotRange->{{-4,4},{-4,4}},
    AspectRatio->1];
Show[g1,gprincipalaxes,ImageSize→Small]
```



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];
```

$$\begin{pmatrix} 12.2737 & 5.09813 \\ 5.09813 & 2.18568 \end{pmatrix}$$

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]]
```

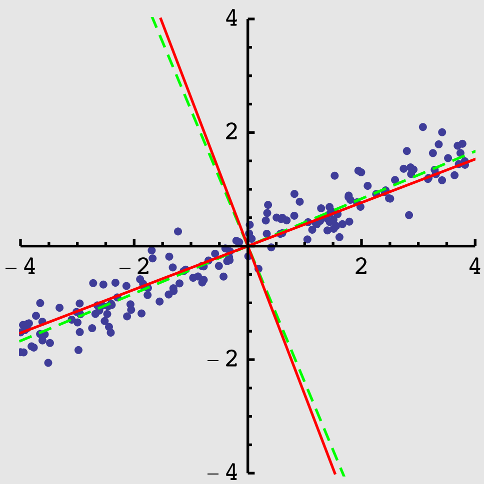
$$\begin{pmatrix} -0.922855 & -0.385148 \\ 0.385148 & -0.922855 \end{pmatrix}$$

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,
        PlotStyle->{{RGBColor[0,1,0],Dashed},{RGBColor[0,1,0],Dashed}}];
```

```
Show[g1, gPCA, gprincipalaxes, ImageSize → Small]
```



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]
```

```
{14.4013, 0.0580088}
```

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];
```

$$\begin{pmatrix} 14.4013 & 0 \\ 0 & 0.0580088 \end{pmatrix}$$

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.

```
alpine = ImageData[
```

```
];

awidth = Dimensions[alpine][[1]];
nregions = 16; swidth = awidth / nregions;
```

```
subface = Table[Take[alpine, {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.

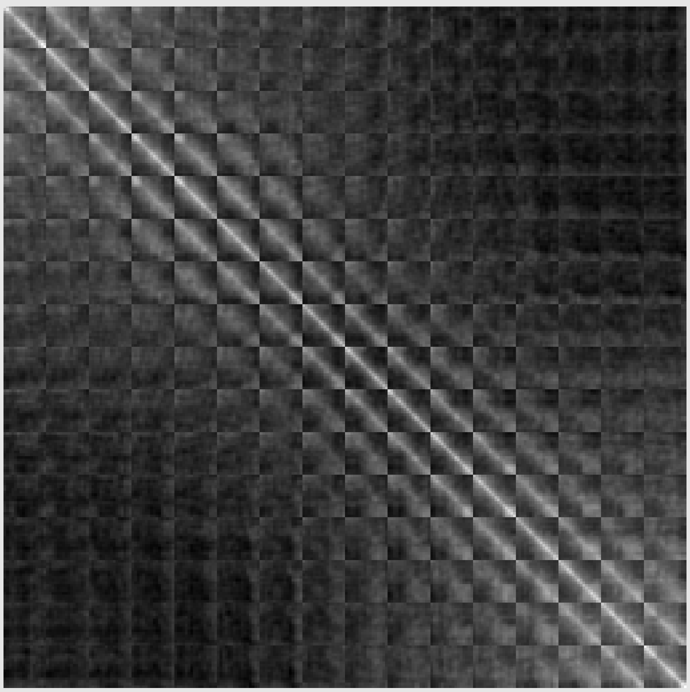
```
subfacelist2 = Table[subfacelist[[i]] - Mean[subfacelist[[i]]],
   {i, 1, 256}];
```

■ Calculate the autocovariance matrix

```
{Dimensions[subfacelist2],
 Dimensions[Outer[Times, subfacelist2[[1]], subfacelist2[[1]]]]}
```

```
{{256, 256}, {256, 256}}
```

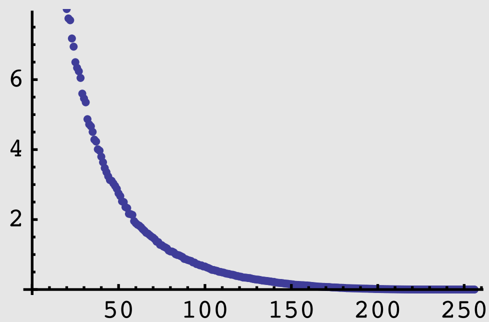
```
temp = Table[0.0, {256}, {256}];
For[i = 1, i < Dimensions[subfacelist][[1]], i++,
  temp = N[Outer[Times, subfacelist2[[i]], subfacelist2[[i]]]] + temp;
 ];
Image[temp] // ImageAdjust
```



■ Calculate the eigenvectors and eigenvalues of the autocovariance matrix

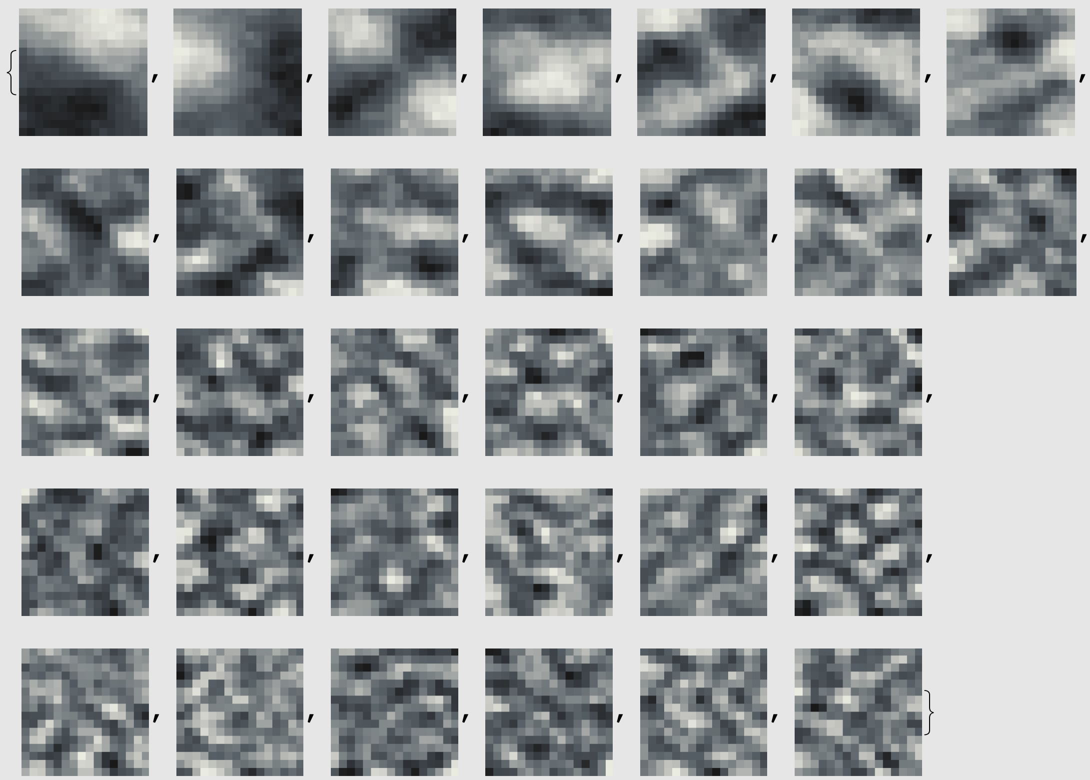
■ Calculate the eigenvectors and eigenvalues of the autocovariance matrix

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



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

```
Table[ArrayPlot[Partition[eigentemp[[i]], 16], Mesh → False,
  ImageSize → Tiny, PixelConstrained → {3, 3}], {i, 1, 32}]
```



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

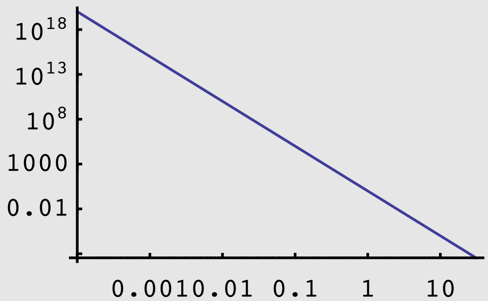
Let's construct a 2nd order generic generative statistical model of images and see what the samples look like.

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, $\mathbf{1/(}f_x^{\mathbf{2}} \mathbf{+} f_y^{\mathbf{2}}\mathbf{)\verb|^|(4\text{-}D)}$. 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;
```

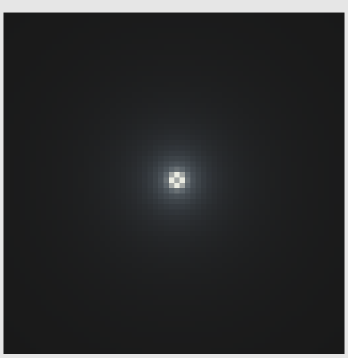
```
q = 2.5;
LogLogPlot[If[(i ≠ 0 || j ≠ 0), 1 / (i * i + 0 * 0) ^ (q), 1 / (2) ^ (q)],
 {i, 0.0001, hfwidth - 1}, ImageSize → Small]
```



- **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];
    ];
```

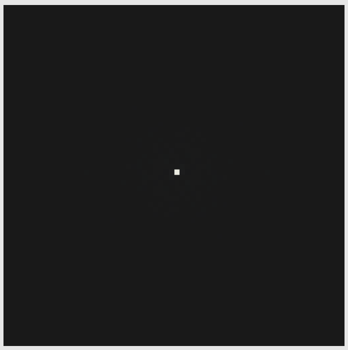
```
ArrayPlot[RotateLeft[fractalfilter[3.5], {hfwidth, hfwidth}],
 Mesh → False]
```



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

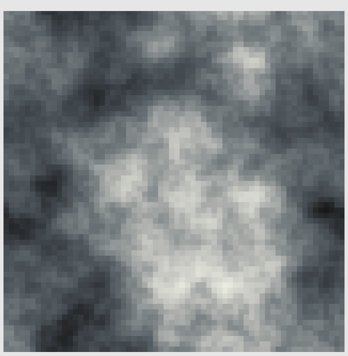
```
randomspectrum = Abs[temp = Fourier[Table[Random[], {size}, {size}]]];
randomphase = Arg[temp];
```

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

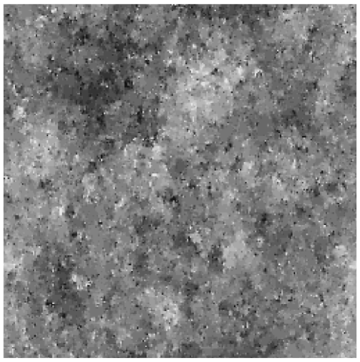


Here is a random fractal image, with D = 3.2

```
ArrayPlot[Chop[
InverseFourier[
fractalfilter[3.2] randomspectrum Exp[I randomphase]]]]
```



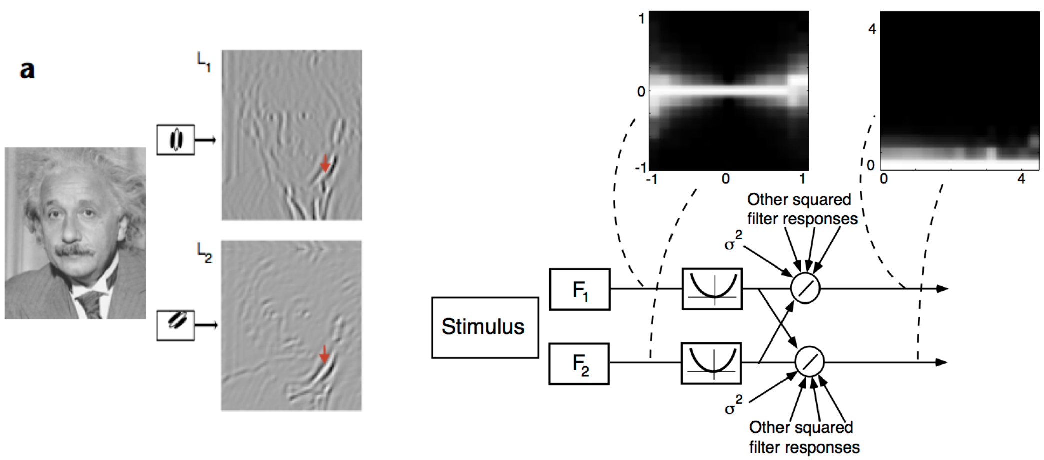
The above gaussian fractal is low-pass, and natural images have tend to have edges, and somewhat sharp patches over a range of scales. Can one do better? Yes. See the sample below from the paper by: Zhu, S. C., & Mumford, D. (1997). Prior Learning and Gibbs Reaction-Diffusion. IEEE Trans. on PAMI, 19(11), 1236-1250.



Higher order redundancies & contrast normalization

Contrast normalization

There are many higher order redundancies in natural images, and a major challenge is to characterize them, and understand how the visual system exploits these redundancies. For example, the figure below shows that the output response of one spatial filter (receptive field (RF) responses) influences the variability in a second spatial filter. Odelia Schwartzt and Eero Simoncelli have shown how a non-linearity called "contrast normalization" serves to remove this redundancy; explaining a number of non-linearities observed in Vq cortical neurons.



See Figure 8 in Simoncelli, E. P., & Olshausen, B. A. (2001). Natural image statistics and neural representation. Annu Rev Neurosci, 24, 1193-1216.(pdf).

See also Figure 5 in Geisler, W. S. (2008). Visual perception and the statistical properties of natural scenes. Annu Rev Psychol, 59, 167-192. (pdf).

Next time

- **Edge detection**

Appendices

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.

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.

Schwartz, O., & Simoncelli, E. P. (2001). Natural signal statistics and sensory gain control. Nature Neuroscience, 4(8), 819–825.

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.

Wainwright, M. J. (1999). Visual adaptation as optimal information transmission. Vision Research, 39, 3960--3974.

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),