

Computational Vision

U. Minn. Psy 5036

Daniel Kersten

Lecture 8: Linear Systems & optics

Outline

Last time

Generative models: intensity-based (e.g. 2D images) and scene-based (e.g. 3D graphics)

Focus on linear, intensity-based models.

When we look at a point of light, like a star, what is the image on the retina? It is slightly blurred image called the “Point Spread Function” or PSF.

(in other contexts, the “impulse response function”)

The PSF of the human eye is determined by optical aberrations, defocus, and in the limit of small pupil size, by diffraction due to the wave nature of light.

For a one dimensional image or signal, the principle of superposition says that the intensity of the output, r_i , corresponding to pixel location i can be expressed as:

$$r_i = \sum_{i'=0} w_{i+i'} l_{i'}$$

where $l_{i'}$ is the intensity of the i'^{th} pixel, and $w_{i+i'}$ is the value of PSF at the $(i+i')$ th location. We've numbered the pixel indices starting at 0. In general, any filtering function such as the PSF is called a “kernel”, represented by a set of weights $\{w_i\}$.

This operation is the same as going to a pixel location, taking the dot product of the kernel with the input image, and putting the dot product into the corresponding pixel location of r , and then moving on to the next location. This operation is called (discrete) **cross-correlation**. So to model the eye's optical transformation of an input image, we treat each incoming pixel as a point, determine its PSF, and add all of these up to form the output image.

We can also do the filtering by first reversing the order of the kernel values followed by the same operation. This is called **convolution**. Why this is useful will become clearer later. If the kernel is symmetric (as in our PSF example), then we get the same results either way. Discrete convolution can be written as:

$$r_i = \sum_{i'} w_{i-i'} l_{i'}$$

If the variables are continuous, then


$$r(x) = \int_{-\infty}^{\infty} w(x-x') l(x') dx' = w(x) \otimes l(x) = l(x) \otimes w(x)$$

\otimes is short-hand for the convolution operation, and the equation: $w(x) \otimes l(x) = l(x) \otimes w(x)$ expresses the commutation property.

In two dimensions,

$$r(x, y) = \int_{-\infty}^{\infty} l(x-x', y-y') w(x', y') dx' dy'$$

The previous lecture used `ListConvolve[]` which operates on lists to do this. *Mathematica* also provides a version that takes an `Image` type as argument. Once adjusted for scale, it gives the same result. Here's a convolution model with a 18x18 square-shaped PSF as the blurring function:

`ImageConvolve[`  `, Table[` $\frac{1}{64}$ `, {i, 1, 18}, {j, 1, 18}]] // ImageAdjust`

Understanding the convolution operation is basic to understanding a wide range of pattern transformations from optical blur to, spatial filtering by neurons in the retina, to “deep convolutional neural networks”.

Today

Today we will see how convolutions are a special case of a linear systems filtering operation, and how convolutions are related to the Fourier decomposition of an image.

Linear systems analysis: Linear intensity-based image models + linear systems

Fourier synthesis/analysis, and the spatial frequency analysis of images

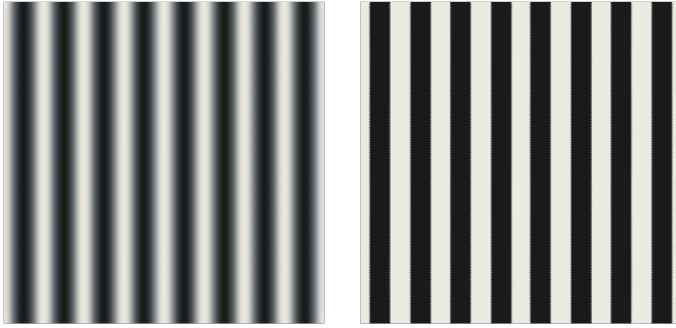
Vector/matrix modeling of optical image transformation

Eigenfunctions of linear shift-invariant systems

Spatial frequency analysis

Application: Modulation transfer functions and the human eye

Preview: Sinewaves are a special type of pattern. We'll try to understand why, and why they are useful.



Linear systems analysis

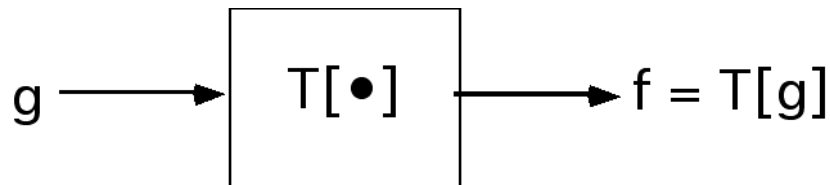
The world of input/output systems can be divided up into linear and non-linear systems. Linear systems are nice because the mathematics that describes them is not only well-known, but also has a mature elegance. On the other hand, it is a fair statement to say that most real-world systems are not linear, and thus hard to analyze...but fascinating if for that reason alone. That nature is usually non-linear doesn't mean one shouldn't familiarize oneself with the basics of linear system theory. Many times a non-linear system has a sufficiently smooth mapping that it can be approximated by a linear one over restricted ranges of parameter values. The assumption of linearity is an excellent starting point--but must be tested. The optics of the eye can be approximated as a linear system over small patches. The responses of some populations of neurons in the retina and primary visual cortex can also be approximated as linear systems.

So how should we define a "linear system"?

The notion of a "linear system" is a generalization of the input/output properties of a straight line passing through zero. For any matrix \mathbf{W} , the equation $\mathbf{W}\cdot\mathbf{g} = \mathbf{f}$ represents a linear system. If \mathbf{W} is a matrix, \mathbf{g}_1 and \mathbf{g}_2 are vectors, and a and b are scalars, it is easy to show that:

$$\mathbf{W}\cdot(a\mathbf{g}_1 + b\mathbf{g}_2) = a\mathbf{W}\cdot\mathbf{g}_1 + b\mathbf{W}\cdot\mathbf{g}_2$$

This is a consequence of the laws of matrix algebra. The idea of a linear system has been generalized beyond matrix algebra to include continuous functions, not just vectors. Imagine we have a box that takes inputs \mathbf{g} , and outputs $\mathbf{f} = \mathbf{T}[\mathbf{g}]$.



The general definition of a linear system is that it satisfies:

$$\mathbf{T}[a\mathbf{g}_1 + b\mathbf{g}_2] = a\mathbf{T}[\mathbf{g}_1] + b\mathbf{T}[\mathbf{g}_2]$$

where T is the transformation that takes the sum of scaled inputs g_1, g_2 (which can be functions or vectors) to the sum of the scaled transformation of g_1 and g_2 . The property, that the output of a sum is the sum of the outputs, is called the *superposition principle* for linear systems--the assumption we needed to add up all the PSF contributions in the previous lecture. The property that a scaled version of the input results in an output scaled by the same amount (i.e. by a or by b) is called the *homogeneity principle*. The fact that linear systems show superposition is good for doing theory, but as we will see later, it limits the kind of input/output computations that can be done with linear systems, and thus with linear neural network models.

When encountering a new input output system, it makes sense to first test for superposition. This was done, for example, in the early days of functional magnetic resonance imaging responses to visual input (Boynton et al., 1996), which showed that linearity held to a first approximation--a very useful finding.

Shift-invariance means that if:

$$g(x,y) \rightarrow f(x,y)$$

then if you shift the input image over by (a,b) and run it through the transformation, you get the same result as just shifting over the original output image by the same amount:

$$g(x-a, y-b) \rightarrow f(x-a, y-b)$$

Shift-invariance is a reasonable assumption over small optical regions.

But once the image gets transduced, shift-invariance breaks down for neural processing of spatial patterns because receptor sampling density isn't constant--it grows with eccentricity.

Vector/matrix modeling of image transformations: Linear intensity-based

Suppose that $T[]$ is a linear system, with g_1 and g_2 input images, (e.g. on a high-resolution computer screen that you are viewing), and f_1, f_2 output images (e.g. on your retina). Usually we will think of the values of the images in terms of contrast, rather than intensity. (Note that if we add lots of images, the intensities would tend to get bigger and bigger. Thus it can be convenient to subtracted out the mean levels of images, then images are represented by contrast.)

By definition, if

$$g_1 \rightarrow f_1$$

and

$$g_2 \rightarrow f_2,$$

then a linear combination of the inputs maps to a linear combination of the outputs

$$a \mathbf{g}_1 + b \mathbf{g}_2 \rightarrow a \mathbf{f}_1 + b \mathbf{f}_2$$

where a and b are scalar weights, such as contrast scaling factors. We now want to understand how to characterize the transformation in terms of the properties of a matrix \mathbf{T} . But first let's see the ways in which one can represent the input vector as combinations of other vectors. We'll assume a linear intensity-based generative model of images.

Brief review of matrix, vector multiplication

We'll need to know how to multiply a vector by a matrix, multiply a vector by a scale, and add two vectors. We can review the rules using Mathematica.

Multiplying a vector times a matrix: replace the i^{th} row of the output vector by the inner product of the i^{th} row of the matrix with the vector.

```
Clear[M, v];
Mmat = Table[Mi,j, {i, 1, 3}, {j, 1, 4}];
% // MatrixForm
```

```
vvec = Table[vj, {j, 1, 4}];
% // MatrixForm
```

$$\begin{pmatrix} M_{1,1} & M_{1,2} & M_{1,3} & M_{1,4} \\ M_{2,1} & M_{2,2} & M_{2,3} & M_{2,4} \\ M_{3,1} & M_{3,2} & M_{3,3} & M_{3,4} \end{pmatrix}$$

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}$$

Use a dot ($.$), not $*$:

```
Mmat.vvec;
% // MatrixForm
```

$$\begin{pmatrix} v_1 M_{1,1} + v_2 M_{1,2} + v_3 M_{1,3} + v_4 M_{1,4} \\ v_1 M_{2,1} + v_2 M_{2,2} + v_3 M_{2,3} + v_4 M_{2,4} \\ v_1 M_{3,1} + v_2 M_{3,2} + v_3 M_{3,3} + v_4 M_{3,4} \end{pmatrix}$$

Note that the number of columns of $Mmat$ must equal the number of elements in v .

Multiple a vector times a scalar, k :

```
k * vvec
{k v1, k v2, k v3, k v4}
```

Adding two vectors:

```

Clear[v1, v2]
vvec1 = Table[v1j, {j, 1, 4}];
vvec2 = Table[v2j, {j, 1, 4}];
vvec1 + vvec2;
% // MatrixForm

$$\begin{pmatrix} v_{1_1} + v_{2_1} \\ v_{1_2} + v_{2_2} \\ v_{1_3} + v_{2_3} \\ v_{1_4} + v_{2_4} \end{pmatrix}$$


```

Linear system property of matrix transformations. Replace a and b, from above, by k1 and k2. And replace W, g1, and g2 by Mmat, vvec1, and vvec2.

```
Mmat.(k1 * vvec1 + k2 * vvec2) // Expand
```

$$\{k_1 v_{1_1} M_{1,1} + k_2 v_{2_1} M_{1,1} + k_1 v_{1_2} M_{1,2} + k_2 v_{2_2} M_{1,2} + k_1 v_{1_3} M_{1,3} + k_2 v_{2_3} M_{1,3} + k_1 v_{1_4} M_{1,4} + k_2 v_{2_4} M_{1,4}, k_1 v_{1_1} M_{2,1} + k_2 v_{2_1} M_{2,1} + k_1 v_{1_2} M_{2,2} + k_2 v_{2_2} M_{2,2} + k_1 v_{1_3} M_{2,3} + k_2 v_{2_3} M_{2,3} + k_1 v_{1_4} M_{2,4} + k_2 v_{2_4} M_{2,4}, k_1 v_{1_1} M_{3,1} + k_2 v_{2_1} M_{3,1} + k_1 v_{1_2} M_{3,2} + k_2 v_{2_2} M_{3,2} + k_1 v_{1_3} M_{3,3} + k_2 v_{2_3} M_{3,3} + k_1 v_{1_4} M_{3,4} + k_2 v_{2_4} M_{3,4}\}$$

```
k1 * Mmat.vvec1 + k2 * Mmat.vvec2 // Expand
```

$$\{k_1 v_{1_1} M_{1,1} + k_2 v_{2_1} M_{1,1} + k_1 v_{1_2} M_{1,2} + k_2 v_{2_2} M_{1,2} + k_1 v_{1_3} M_{1,3} + k_2 v_{2_3} M_{1,3} + k_1 v_{1_4} M_{1,4} + k_2 v_{2_4} M_{1,4}, k_1 v_{1_1} M_{2,1} + k_2 v_{2_1} M_{2,1} + k_1 v_{1_2} M_{2,2} + k_2 v_{2_2} M_{2,2} + k_1 v_{1_3} M_{2,3} + k_2 v_{2_3} M_{2,3} + k_1 v_{1_4} M_{2,4} + k_2 v_{2_4} M_{2,4}, k_1 v_{1_1} M_{3,1} + k_2 v_{2_1} M_{3,1} + k_1 v_{1_2} M_{3,2} + k_2 v_{2_2} M_{3,2} + k_1 v_{1_3} M_{3,3} + k_2 v_{2_3} M_{3,3} + k_1 v_{1_4} M_{3,4} + k_2 v_{2_4} M_{3,4}\}$$

Vector/matrix modeling of optical image transformation: Simple 1D case

Output image "response" to a single point of light (pixel)

For the time being, let's imagine our images are one-dimensional, and represented as vectors whose positions indicate pixel location, and whose values are intensities. Suppose that an arbitrary input image is given by $\mathbf{g} = \{g_1, g_2, \dots\}$. We'd like to get a deeper understanding of how to model its transformation to an output image.

Let an input *test* image (e.g. on a computer screen) be represented by $\mathbf{u}_i = \{0, 0, 0, \dots, 1, \dots\}$ where all pixels are black (zero) except for the *i*th one which is bright (1). If the 4th pixel is bright, for example, the test image is represented by a vector $\mathbf{u}_4 = \{0, 0, 0, 1, 0, 0, \dots\}$:

$$\mathbf{u4} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

We'll use **bold** for vectors and matrices, and plain for scalars.

Let **ti** represent the output image (e.g. retinal image), i.e. *point spread function (PSF)*, that results from **u1**.

Let's suppose that for $i=4$, **t4** = {t6, t7, t8, t1, ...} represents the output image (e.g. retinal image), i.e. *point spread function (PSF)*, that results from input **u4**

$$\mathbf{t4} = \begin{pmatrix} t6 \\ t7 \\ t8 \\ t1 \\ t2 \\ t3 \\ t4 \\ t5 \end{pmatrix}$$

For example, if there is a little local blurring, t4, our "PSF", might look like this:

$$\mathbf{t4} = \begin{pmatrix} 0 \\ 0 \\ 1/4 \\ 1/2 \\ 1/4 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

```
u4 = {0, 0, 0, 1, 0, 0, 0, 0};
```

```
t4 = {0, 0, 1/4, 1/2, 1/4, 0, 0, 0};
```

```
GraphicsRow[{ArrayPlot[{u4}, ColorFunction -> "GrayTones"],
  ArrayPlot[{t4}, ColorFunction -> "GrayTones"]}]
```



Output image "response" to a spatially shifted point of light

What if we now shift the input pattern over a position, so the bright pixel is at the (i+1)th position? E.g.

$$\mathbf{u}_5 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

For a system like the eye, we might expect that the output image would be basically the same form, except that it also gets shifted over:

$$\mathbf{t}_5 = \begin{pmatrix} X \\ t_6 \\ t_7 \\ t_8 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix}$$

We've stuck an "X" at the slot left open after the shift. There are various ways of modeling the boundaries. One way which is convenient mathematically is to use a circular boundary. This is OK for large images, and where the PSF is local and falls off to zero away from the i th position. So most of the "X"s are zeros.

The practical justification is that images are usually big, and the blur kernel is small relative to the size of the image, so the boundaries don't contribute much to the center of the image.

We wrap the entries around, so in the above case $X \rightarrow t_5$, and for our example $=0$.

$$\mathbf{t5} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1/4 \\ 1/2 \\ 1/4 \\ 0 \\ 0 \end{pmatrix}$$

In other words, the image of a point shifted over ($t5$) has basically the same form as $t4$ --it is just offset by the shift.

Output image "response" to an arbitrary image \mathbf{g}

Let's see how \mathbf{g} gets transformed to an output image \mathbf{f} , through a transformation matrix \mathbf{T} .

We'll generalize things a bit. Let $\mathbf{u1}, \mathbf{u2}, \mathbf{u3}, \dots$ be the shifted test images where the bright pixel is at location 1, 2, ...:

$$\mathbf{u1} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \mathbf{u2} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \dots$$

This is the familiar Cartesian basis set for vectors.

Using our linear intensity-based model of image patterns, we can express the (arbitrary) input image \mathbf{g} as a weighted linear sum of the test images:

$$\mathbf{g} = \begin{pmatrix} g1 \\ g2 \\ g3 \\ g4 \\ g5 \\ g6 \\ g7 \\ g8 \end{pmatrix} = g1 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + g2 \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \dots = g1 \mathbf{u1} + g2 \mathbf{u2} + \dots$$

The weights are just the intensities of the pixels.

Assuming that the transformation $\mathbf{T}[\]$ is linear, then the following holds by definition:

$$\begin{aligned}
 \mathbf{f} &= \\
 \mathbf{T}[\mathbf{g}] &= \mathbf{T}[g_1\mathbf{u}_1 + g_2\mathbf{u}_2 + \dots] = g_1\mathbf{T}[\mathbf{u}_1] + g_2\mathbf{T}[\mathbf{u}_2] + \dots = g_1\mathbf{t}_1 + g_2\mathbf{t}_2 + \dots \quad (1) \\
 &= g_1 \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{pmatrix} + g_2 \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{pmatrix} + \dots
 \end{aligned}$$

Each $\mathbf{T}[\mathbf{u}_i]$, $i = 1, 2, \dots$ is the blurry image of the input test image, \mathbf{u}_i .

The transformation \mathbf{T} can be represented by a matrix that multiplies \mathbf{g} . Assuming shift-invariance, we can line up all the \mathbf{t}_i vectors in columns to represent the output as matrix \mathbf{T} times input \mathbf{g} :

$$\mathbf{f} = \begin{pmatrix} t_1 & t_8 & t_7 & t_6 & t_5 & t_4 & t_3 & t_2 \\ t_2 & t_1 & t_8 & t_7 & t_6 & t_5 & t_4 & t_3 \\ t_3 & t_2 & t_1 & t_8 & t_7 & t_6 & t_5 & t_4 \\ t_4 & t_3 & t_2 & t_1 & t_8 & t_7 & t_6 & t_5 \\ t_5 & t_4 & t_3 & t_2 & t_1 & t_8 & t_7 & t_6 \\ t_6 & t_5 & t_4 & t_3 & t_2 & t_1 & t_8 & t_7 \\ t_7 & t_6 & t_5 & t_4 & t_3 & t_2 & t_1 & t_8 \\ t_8 & t_7 & t_6 & t_5 & t_4 & t_3 & t_2 & t_1 \end{pmatrix} \begin{pmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \\ g_8 \end{pmatrix} = \mathbf{T} \cdot \mathbf{g} \quad (2)$$

► 1. Prove that equation (2) is the same as (1)

As noted above, we are using "circulant" boundaries, i.e. that wrap around. The above matrix is said to be **circulant**. A matrix for which the values along diagonals are the same is called a Toeplitz matrix. A circulant matrix is a special case of a Toeplitz matrix in which each subsequent row is a copy of the previous row but shifted one element to the right.

Shift-invariance and symmetric transformations, T

Let's take our model one step further. In addition to shift-invariance, we might also expect an optical system like the eye to show symmetry to a first approximation. I.e. the PSF is radially symmetric.

Then our 1-D transformation matrix becomes:

$$\begin{pmatrix} t1 & t2 & t3 & t4 & t5 & t6 & t7 & t8 \\ t2 & t1 & t2 & t3 & t4 & t5 & t6 & t7 \\ t3 & t2 & t1 & t2 & t3 & t4 & t5 & t6 \\ t4 & t3 & t2 & t1 & t2 & t3 & t4 & t5 \\ t5 & t4 & t3 & t2 & t1 & t2 & t3 & t4 \\ t6 & t5 & t4 & t3 & t2 & t1 & t2 & t3 \\ t7 & t6 & t5 & t4 & t3 & t2 & t1 & t2 \\ t8 & t7 & t6 & t5 & t4 & t3 & t2 & t1 \end{pmatrix}$$

One of the benefits of the symmetric shift-invariant properties of the eye's optics is that we can model the transformation as a symmetric Toeplitz matrix. As we will see later, these properties suggest an alternative basis set to represent images. Rather than represent an image as a linear combination of points (i.e. the above u 's), images will be represented as a linear combination of *spatial sinewave gratings*. We'll see how the symmetric shift-invariant case provides a particularly simple example of Fourier analysis/synthesis.

So in standard *Mathematica* format, the above matrix calculation is written as:

```
Clear[g1, g2, t4]
```

Let's use Mathematica to multiply the input vector:

```
{g1, g2, g3, g4, g5, g6, g7, g8} // MatrixForm
```

$$\begin{pmatrix} g1 \\ g2 \\ g3 \\ g4 \\ g5 \\ g6 \\ g7 \\ g8 \end{pmatrix}$$

by our transformation matrix T with the circulant property:

```
{{t1, t8, t7, t6, t5, t4, t3, t2}, {t2, t1, t8, t7, t6, t5, t4, t3},
 {t3, t2, t1, t8, t7, t6, t5, t4}, {t4, t3, t2, t1, t8, t7, t6, t5},
 {t5, t4, t3, t2, t1, t8, t7, t6}, {t6, t5, t4, t3, t2, t1, t8, t7},
 {t7, t6, t5, t4, t3, t2, t1, t8}, {t8, t7, t6, t5, t4, t3, t2, t1}} // MatrixForm
```

$$\begin{pmatrix} t1 & t8 & t7 & t6 & t5 & t4 & t3 & t2 \\ t2 & t1 & t8 & t7 & t6 & t5 & t4 & t3 \\ t3 & t2 & t1 & t8 & t7 & t6 & t5 & t4 \\ t4 & t3 & t2 & t1 & t8 & t7 & t6 & t5 \\ t5 & t4 & t3 & t2 & t1 & t8 & t7 & t6 \\ t6 & t5 & t4 & t3 & t2 & t1 & t8 & t7 \\ t7 & t6 & t5 & t4 & t3 & t2 & t1 & t8 \\ t8 & t7 & t6 & t5 & t4 & t3 & t2 & t1 \end{pmatrix}$$

```

{{t1, t8, t7, t6, t5, t4, t3, t2}, {t2, t1, t8, t7, t6, t5, t4, t3},
 {t3, t2, t1, t8, t7, t6, t5, t4}, {t4, t3, t2, t1, t8, t7, t6, t5},
 {t5, t4, t3, t2, t1, t8, t7, t6}, {t6, t5, t4, t3, t2, t1, t8, t7},
 {t7, t6, t5, t4, t3, t2, t1, t8}, {t8, t7, t6, t5, t4, t3, t2, t1}}.
{g1, g2, g3, g4, g5, g6, g7, g8} // MatrixForm

```

$$\begin{pmatrix} g_1 t_1 + g_8 t_2 + g_7 t_3 + g_6 t_4 + g_5 t_5 + g_4 t_6 + g_3 t_7 + g_2 t_8 \\ g_2 t_1 + g_1 t_2 + g_8 t_3 + g_7 t_4 + g_6 t_5 + g_5 t_6 + g_4 t_7 + g_3 t_8 \\ g_3 t_1 + g_2 t_2 + g_1 t_3 + g_8 t_4 + g_7 t_5 + g_6 t_6 + g_5 t_7 + g_4 t_8 \\ g_4 t_1 + g_3 t_2 + g_2 t_3 + g_1 t_4 + g_8 t_5 + g_7 t_6 + g_6 t_7 + g_5 t_8 \\ g_5 t_1 + g_4 t_2 + g_3 t_3 + g_2 t_4 + g_1 t_5 + g_8 t_6 + g_7 t_7 + g_6 t_8 \\ g_6 t_1 + g_5 t_2 + g_4 t_3 + g_3 t_4 + g_2 t_5 + g_1 t_6 + g_8 t_7 + g_7 t_8 \\ g_7 t_1 + g_6 t_2 + g_5 t_3 + g_4 t_4 + g_3 t_5 + g_2 t_6 + g_1 t_7 + g_8 t_8 \\ g_8 t_1 + g_7 t_2 + g_6 t_3 + g_5 t_4 + g_4 t_5 + g_3 t_6 + g_2 t_7 + g_1 t_8 \end{pmatrix}$$

Our transformation matrix T is completely characterized by just one vector (which just gets repeated and shifted). In fact this filtering operation can be done using a convolution operator. `ListConvolve[]` gives us the same answer:

```

ListConvolve[{t1, t2, t3, t4, t5, t6, t7, t8},
 {g1, g2, g3, g4, g5, g6, g7, g8}, 1] // MatrixForm

```

$$\begin{pmatrix} g_1 t_1 + g_8 t_2 + g_7 t_3 + g_6 t_4 + g_5 t_5 + g_4 t_6 + g_3 t_7 + g_2 t_8 \\ g_2 t_1 + g_1 t_2 + g_8 t_3 + g_7 t_4 + g_6 t_5 + g_5 t_6 + g_4 t_7 + g_3 t_8 \\ g_3 t_1 + g_2 t_2 + g_1 t_3 + g_8 t_4 + g_7 t_5 + g_6 t_6 + g_5 t_7 + g_4 t_8 \\ g_4 t_1 + g_3 t_2 + g_2 t_3 + g_1 t_4 + g_8 t_5 + g_7 t_6 + g_6 t_7 + g_5 t_8 \\ g_5 t_1 + g_4 t_2 + g_3 t_3 + g_2 t_4 + g_1 t_5 + g_8 t_6 + g_7 t_7 + g_6 t_8 \\ g_6 t_1 + g_5 t_2 + g_4 t_3 + g_3 t_4 + g_2 t_5 + g_1 t_6 + g_8 t_7 + g_7 t_8 \\ g_7 t_1 + g_6 t_2 + g_5 t_3 + g_4 t_4 + g_3 t_5 + g_2 t_6 + g_1 t_7 + g_8 t_8 \\ g_8 t_1 + g_7 t_2 + g_6 t_3 + g_5 t_4 + g_4 t_5 + g_3 t_6 + g_2 t_7 + g_1 t_8 \end{pmatrix}$$

When we do a convolution (or a correlation, as in `ListCorrelate[kern,list,offset]`), we can specify that *Mathematica* use a circular boundary, and place *kern* in the first row, but aligned relative to *list* by *offset*.

So what have we accomplished? We've shown that we can interpret convolution as a matrix multiplication, where the matrix has a special structure in which each row is a shifted version of the previous one. We've also seen that we can represent a vector as a linear combination of more basic vectors. But these "basic" vectors were so basic as to seem trivial. What if we represent a vector by a linear combination of some other collection of "basic" vectors, i.e. a "basis set"?

Characterizing a linear system by its response to an orthonormal basis set

Orthonormal basis sets

Given an image, we can think of it as being composed of a weighted sum of basic or "basis" images that together make up a "basis set" (e.g. the u 's above). Each basis image is a vector of the same dimension-

ality as the image, but is presumed fixed. But with a suitable combination of weights, the same set can be used to represent any vector.

Recall the idea from the previous lecture to represent an image \vec{T} in terms of linear combinations or sums of other images \vec{T}_i :

$$\vec{T} = m_1 * \vec{T}_1 + m_2 * \vec{T}_2 + m_3 * \vec{T}_3 + \dots$$

where m_i 's are scalars. Each image is flattened into an N-dimensional vector, which can be thought of as a point in an N-dimensional space, where N is the number of pixels.

What is a "good" basis set? We have lots (∞ !) to choose from, and what is best will depend on what we are doing. The simplest basis set (the Cartesian set used above) corresponds to our standard pixel representation for images, where the i th pixel is represented by a vector that is all zero's except for the i th element which is 1.

Suppose we have a different basis set (e.g. a "Walsh" set defined below). Let's call the i th basis image $\mathbf{w}[i]$. Any 8-pixel image \mathbf{g} can be written:

$$\mathbf{g} = g[1] \begin{pmatrix} .3535 \\ .3535 \\ .3535 \\ .3535 \\ .3535 \\ .3535 \\ .3535 \\ .3535 \end{pmatrix} + g[2] \begin{pmatrix} +.3535 \\ -.3535 \\ -.3535 \\ +.3535 \\ +.3535 \\ -.3535 \\ -.3535 \\ +.3535 \end{pmatrix} + \dots = g[1] \mathbf{w}[1] + g[2] \mathbf{w}[2] + \dots$$

If the set $\{\mathbf{w}_i\} = \{\mathbf{w}[i]\}$ is *orthogonal*, then $\mathbf{w}[i] \cdot \mathbf{w}[j] = 0$ for $i \neq j$. Note that $\mathbf{w}[i]$ is the i th vector in the set-- not a scalar.

If it is *normal*, then $\mathbf{w}[i] \cdot \mathbf{w}[i] = 1$. (The vector length of each basis vector is 1).

If the set is *complete*, then it spans 8-space in such a way that we can express any 8-d vector as a linear sum of these basis vectors.

Orthonormal (orthogonal and normalized) basis sets make it easier to do calculations.

Are there other orthonormal sets than the Cartesian or Walsh set? Again, there are lots, and again which one we use will depend on the job at hand.

Let $\{\mathbf{w}_i\} = \{\mathbf{w}[i]\}$ be any orthonormal set. Then an arbitrary vector, \mathbf{g} can be represented as a weighted sum of the orthonormal vectors, each weighted by the amount \mathbf{g} projects onto \mathbf{w}_i i.e. $(\mathbf{g} \cdot \mathbf{w}_i)$.

The vector $\{g \cdot w_1, g \cdot w_2, g \cdot w_3 \dots\}$ is sometimes called the spectrum of \mathbf{g} . We have:

$$\mathbf{g} = \sum (\mathbf{g} \cdot \mathbf{w}_i) \mathbf{w}_i$$

In our first example (the above \mathbf{u}_i 's play the role of the \mathbf{w}_i s), the dot product is really simple and $\mathbf{g} \cdot \mathbf{w}_i = g[[i]]$, i.e. the i th element of vector \mathbf{g} .

Application to an "unknown" system

How can one go about measuring the matrix to characterize a linear system such as the optics of the eye? Or a neural system?

We won't presume symmetry or shift-invariance.

Suppose we have an unknown physical system, which we model as a linear system with matrix \mathbf{T} :

```
T = Table[RandomReal[], {i, 1, 8}, {j, 1, 8}];
```

Goal: We would like to make a simple set of measurements that could characterize \mathbf{T} in such a way that we could predict the output of \mathbf{T} to any input.

This is the sort of task that engineers face when wanting to characterize, say a stereo amplifier (as a model linear system), so that the output sound can be predicted for any input sound. What kind of measurements would tell us what \mathbf{T} is?

We could just "stimulate" the system with cartesian vectors $\{1,0,0,0,0,0,0,0\}$, $\{0,1,0,0,0,0,0,0\}$, and so forth and collect the responses which would be the columns of \mathbf{T} --just as we did above.

This has two practical problems: 1) for a real physical system, such as your stereo, or a neuron in the eye, to get a measurable response might require stimulating it with a high-intensity audio or light intensity spike, which could damage what you are trying to study; 2) Characterizing the linear system by a matrix \mathbf{T} , requires n^2 numbers, where n is the input signal vector length--and n can be pretty big for both audio and visual systems. Problem 2) can be solved when \mathbf{T} is symmetric and/or shift-invariant (because of the redundancy, $O(n)$ numbers are sufficient). Problem 1) can be addressed by showing that we can characterize \mathbf{T} with any basis set--so we can pick one that won't blow out the physical system being tested.

As an example, consider the orthonormal set (\mathbf{v} 's below) of Walsh functions. It has the advantage that the elements that contribute to the "energy", i.e. (the square of the length) are distributed across the vector.

```
VectorLength[x_] := N[Sqrt[x.x]]
```

```

v1 = {1, 1, 1, 1, 1, 1, 1, 1}; w1 = v1/Vectorlength[v1];
v2 = {1,-1,-1, 1, 1,-1,-1, 1}; w2 = v2/Vectorlength[v2];
v3 = {1, 1,-1,-1,-1,-1, 1, 1}; w3 = v3/Vectorlength[v3];
v4 = {1,-1, 1,-1,-1, 1,-1, 1}; w4 = v4/Vectorlength[v4];
v5 = {1, 1, 1, 1,-1,-1,-1,-1}; w5 = v5/Vectorlength[v5];
v6 = {1,-1,-1, 1,-1, 1, 1,-1}; w6 = v6/Vectorlength[v6];
v7 = {1, 1,-1,-1, 1, 1,-1,-1}; w7 = v7/Vectorlength[v7];
v8 = {1,-1, 1,-1, 1,-1, 1,-1}; w8 = v8/Vectorlength[v8];

```

Consider an arbitrary 1-D image **g**:

```
g = {2,6,1,7,11,4,13, 29};
```

g can be written as the sum of its own projections onto the basis set:

```

(g.w1) w1 + (g.w2) w2 + (g.w3) w3 + (g.w4) w4 +
(g.w5) w5 + (g.w6) w6 + (g.w7) w7 + (g.w8) w8
{2., 6., 1., 7., 11., 4., 13., 29.}

```

Suppose we now do an "experiment" to find out how **T** transforms the vectors of our basis set, and we put all of these transformed basis elements into a new set of vectors **newW[[i]]**.

newW is a matrix for which each column is the response of **T** to a basis vector.

```
newW = Transpose[{T.w1,T.w2,T.w3,T.w4,T.w5,T.w6,T.w7,T.w8}];
```

Note that **newW** is an 8x8 matrix. So how can we calculate the output of **T**, given input **g** without actually running the input through **T**?

By the principle of linearity, we can also calculate the output by finding the "spectrum" of **g**, and then scaling each of the transformed basis elements by the spectrum and adding them up:

$$T.g = T.\left\{\sum (g.w_i) w_i\right\} = \sum (g.w_i) T.w_i$$

```

(g.w1) T.w1 + (g.w2) T.w2 + (g.w3) T.w3 + (g.w4) T.w4 +
(g.w5) T.w5 + (g.w6) T.w6 + (g.w7) T.w7 + (g.w8) T.w8
{34.2402, 50.9029, 14.0481, 37.893, 45.0004, 43.6688, 32.8009, 38.1511}

```

..but we don't have to run the basis vectors through T each time. We have already done our "experiment in the lab", so we know what the transformed basis vectors **{T.w1, ...}** are, we stored them as columns of the matrix **newW**. Each of these columns is given by: **Transpose[newW][[i]]**. (The transpose of a matrix flips the rows into columns.)

So all we need to do is calculate what the spectrum (**g.wi**) is, and then combine this using the above equation, to produce the output of **T**. I.e. **T.g** is:

```

(g.w1) Transpose[newW][[1]] + (g.w2) Transpose[newW][[2]] + (g.w3) Transpose[newW][[3]] +
(g.w4) Transpose[newW][[4]] + (g.w5) Transpose[newW][[5]] + (g.w6) Transpose[newW][[6]] +
(g.w7) Transpose[newW][[7]] + (g.w8) Transpose[newW][[8]]
{34.2402, 50.9029, 14.0481, 37.893, 45.0004, 43.6688, 32.8009, 38.1511}

```

The point is that once we've measured the output to the complete set of basis images, we just store those images (just 8 of them in this simple example).

Then we can get the output to any arbitrary input image by computing the spectrum of the input and using those values as weights for the stored images.

The PSF was a special case of the "stored image", but was simpler because of shift-invariance. Here we are not assuming shift-invariance.

Check our answer: If we "go back to the lab" and run the input \mathbf{g} through \mathbf{T} we get:

$\mathbf{T} \cdot \mathbf{g}$

```
{34.2402, 50.9029, 14.0481, 37.893, 45.0004, 43.6688, 32.8009, 38.1511}
```

Same thing in more concise notation

Let the basis vectors be the rows of a matrix \mathbf{W} :

```
W = {w1, w2, w3, w4, w5, w6, w7, w8};
```

So again, we can project \mathbf{g} onto the rows of \mathbf{W} , and then reconstitute it in terms of \mathbf{W} to get \mathbf{g} back again:

$(\mathbf{W} \cdot \mathbf{g}) \cdot \mathbf{W}$

```
{2., 6., 1., 7., 11., 4., 13., 29.}
```

$\text{newW} \cdot \mathbf{W} \cdot \mathbf{g}$

```
{34.2402, 50.9029, 14.0481, 37.893, 45.0004, 43.6688, 32.8009, 38.1511}
```

What if the choice of basis set is the set of eigenvectors of \mathbf{T} ?

We are now going to introduce an elegant idea that will simplify our analysis of linear shift-invariant systems. Our illustration relies on a result from linear algebra that says the eigenvectors of a symmetric matrix are real and orthogonal. In general, one can relax the assumption of symmetry.

Eigenvectors and eigenvalues--short review

It is worth taking a little time to learn (or review) the idea of an eigenvector. It crops up in statistical data analysis (PCA), and in the statistics of natural images, which we will study later.

Eigenvectors

An eigenvector, \mathbf{x} , of a matrix, \mathbf{A} , is vector that when you multiply it by \mathbf{A} , you get an output vector that points in the same direction as \mathbf{x} :

$$\mathbf{Ax} = \lambda \mathbf{x}$$

where λ is a scalar that adjusts the length change of \mathbf{x} .

The *Mathematica* function **Eigenvectors[A]** returns the eigenvectors of matrix \mathbf{A} as the rows of a matrix, which we'll call **eig**:

```
In[840]:= A = {{1,2},{3,4}};
          eig = Eigenvectors[A]
```



```
Out[841]= {{1/6 (-3 + Sqrt[33]), 1}, {1/6 (-3 - Sqrt[33]), 1}}
```

We can verify that **eig[[1]]** and **A.eig[[1]]** lie along the same direction by taking the dot product of the unit vectors pointing in the directions of each:

```
normalize[x_] := x/Sqrt[x.x];
normalize[eig[[1]]].normalize[A.eig[[1]]];
N[%]
1.
```

Eigenvalues

The eigenvalues are given by:

```
Eigenvalues[A]
{1/2 (5 + Sqrt[33]), 1/2 (5 - Sqrt[33])}
```

Eigenvalues and eigenvector elements do not have to be real numbers. They can be complex, that is an element can be the sum of a real and imaginary number. In *Mathematica*, imaginary numbers are represented by multiples (or fractions) of **I**, the square root of -1:

```
Sqrt[-1]
Sqrt[-1]//StandardForm
i
i
```

```
B = {{1,2},{-3,4}};
Eigenvalues[B]
{1/2 (5 + i Sqrt[15]), 1/2 (5 - i Sqrt[15])}
```

We've seen how linearity provides us with a method for characterizing a linear system in terms of the responses of the system to the basis vectors. The problem is that if the input signals are long vectors, say with dimension 40,000, then this set of basis vector responses is really big-- 1.6×10^9 numbers.

Lets assume we have a symmetric matrix transformation, **T**, and apply it to its own eigenvectors **{w_i}**. Then,

$$\mathbf{T} \cdot \mathbf{w}_i = \lambda_i \mathbf{w}_i$$

The eigenvectors of a symmetric matrix are orthogonal and real-valued (very handy). Now we use those same eigenvectors of **T** to represent the input **g**. If the elements of the basis set are the eigenvectors of **T**, then the transformation of any arbitrary input vector **g** is given by:

$$\mathbf{T} \cdot \mathbf{g} = \mathbf{T} \cdot \left\{ \sum (\mathbf{g} \cdot \mathbf{w}_i) \mathbf{w}_i \right\} = \sum (\mathbf{g} \cdot \mathbf{w}_i) \mathbf{T} \cdot \mathbf{w}_i = \sum \alpha_i \lambda_i \mathbf{w}_i$$

Where the α_i are the projections of \mathbf{g} onto each eigenvector.

Having the eigenvectors of \mathbf{T} enables us to express the *input and output* of \mathbf{T} in terms of the same basis set--the eigenvectors.

All \mathbf{T} does to the input is to scale its projection onto each eigenvector by the eigenvalue for that eigenvector.

The set of these eigenvalues, $\{\lambda_i\}$ is sometimes called the *modulation transfer function* because it describes how the amplitude of the eigenvectors change as they pass through \mathbf{T} .

The α_i is the spectrum of the input signal.

Linear systems analysis is the foundation of Fourier analysis, and is why it makes sense to characterize your stereo amplifier in terms of frequency response. But your stereo isn't just any linear system--it has the special property that if you input a sound at time t and measure the response, and then you input the same sound again at a later time, you get the same response, except of course that it is shifted in time. It is a shift-invariant system. The eigenvectors of a shift-invariant system are sinusoids.

(The eigenvectors of the symmetric matrix are sinusoids, not just because the matrix was symmetric, but also because each row of the matrix was a shifted version of the previous row--the elements along any given diagonal are identical-- a symmetric Toeplitz matrix.)

Sinewave inputs are the eigenvectors of your stereo system. The dimensionality is much higher--if you are interested in frequencies up to 20,000 Hz, your eigenvector for this highest frequency would have least 40,000 elements--not just 8!

This kind of analysis has been applied not only to physical systems, but to a wide range of neural sensory systems. For the visual system alone, linear systems analysis has been used to study the cat retina (Enroth-Cugell and Robson, 1964), the monkey visual cortex, and the human contrast sensitivity system as a whole (Campbell and Robson, 1968).

Much empirical analysis has been done using linear systems theory to characterize neural sensory systems, and other neural systems such as those for eye movements. It works wonderfully as long as the linear system approximation holds. And it does do quite well for the lateral eye of the limulus, X-cells and P-cells of the mammalian visual system, over restricted ranges for so-called "simple" cells in the visual cortex, among others. The optics of the simple eye is another example of an approximately linear system. Many non-linear systems can be approximated as linear systems over smooth

subdomains.

Summary of frequency analysis

In summary, if \mathbf{T} has n distinct orthogonal eigenvectors, \mathbf{e}_i , and known eigenvalues, λ_i , then we have a particularly easy way to calculate the response to an input \mathbf{g} :

Step 1: Project \mathbf{g} onto eigenvectors of \mathbf{T} ($\mathbf{w}_i = \mathbf{e}_i$) using the dot product: $\mathbf{g} \cdot \mathbf{e}_i$. This gives us the spectrum of \mathbf{g} .

Step 2: Scale each $\mathbf{g} \cdot \mathbf{e}_i$ by the eigenvalue of \mathbf{e}_i : $\lambda_i \mathbf{g} \cdot \mathbf{e}_i$. λ says how much to change the gain or attenuation of the spectral components

Step 3: Scale each \mathbf{e}_i by the scalar $\lambda_i \mathbf{g} \cdot \mathbf{e}_i$

Step 4: Sum these up. That's the response of \mathbf{T} to \mathbf{g} : $\mathbf{f} = \sum_i (\lambda_i \mathbf{g} \cdot \mathbf{e}_i) \mathbf{e}_i$

The Modulation Transfer Function (MTF) of the human eye

Gratings: Eigenfunctions of linear shift-invariant optics

Now a point of light, a face, a square wave grating do not maintain the same form through optical transformation, because aberrations and diffraction blur the edges. However, for a spatially homogeneous optical system, a sine-wave grating does keep the same form (consistent with the shift-invariant properties of a linear system). The figure below shows pictures of a square-wave and a sine-wave (you need 256 gray-levels to see the sine-wave grating). What happens to the form of the two intensity patterns when you blur your eyes?

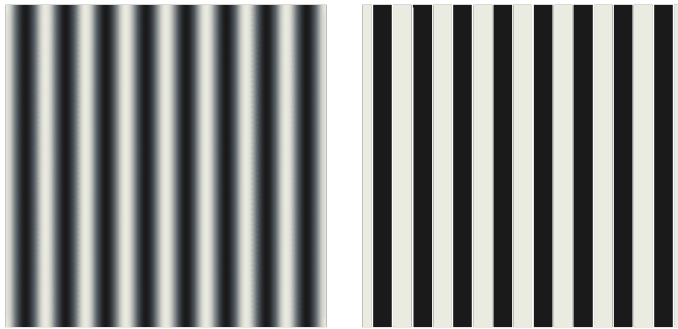
Blurring: Sine vs. square

```
Grating[x_, y_, fx_, fy_] := Cos[2 Pi (fx x + fy y)];
Square[x_, y_, fx_, fy_] := Sign[Grating[x, y, fx, fy]];

gsine = DensityPlot[0.25 * Grating[x, y, 4, 0], {x, -1, 1}, {y, -1, 1}, PlotPoints -> 64,
  Mesh -> False, Frame -> False, PlotRange -> {-1, 1}, ColorFunction -> "GrayTones"];

gsquare =
  DensityPlot[0.25 * Square[x, y, 4, 0], {x, -1, 1}, {y, -1, 1}, PlotPoints -> 64,
    Mesh -> False, Frame -> False, PlotRange -> {-1, 1}, ColorFunction -> "GrayTones"];
```

```
Show[GraphicsRow[{gsine, gsquare}]]
```



Sine-wave gratings are eigenfunctions of linear shift-invariant systems. Our complete basis set (a collection of eigenfunctions, $\{b_i\}$) will have to be built out of collection of gratings of various frequencies (combinations of f_x and f_y produce different orientations) and phases (ϕ). So if we can represent our images in terms of sums of sine-wave gratings, then we can model how a linear shift-invariant optical system distorts the image. This kind of representation of an image is called *Fourier Analysis*. Because we are talking about intensity as a function of space (we'll get to time later), this kind of Fourier analysis is called *spatial frequency analysis* of images.

Fourier analysis

Representing images: The formal case, continuous variable case -> Fourier transform

A complete formal analysis for continuous images is the analog to the discrete model we introduced above, where frequencies can take on continuous values. The basic formalism is similar, with summation replaced by integration, and complex variables are used (to compactly and efficiently deal with phase relationships between the gratings). In one spatial dimension, the forms are given by:

$$\text{Fourier transform : } F(f_x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \text{image}(x) e^{i2\pi f_x x} dx,$$

$$\text{Inverse fourier transform : } \text{image}(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(f_x) e^{-i2\pi f_x x} df_x$$

Mathematica has built-in functions `FourierTransform[]` and `InverseFourierTransform[]` to deal with symbolic manipulations for the continuous case. `Fourier[]` handles the discrete numerical calculations. One theoretical advantage of Fourier transforms is that, in contrast to Fourier series, one is not limited to periodic images.

Actual practice: hybrid discrete-continuous

But in actual practice, we might use some discrete collection of continuous gratings (Fourier series) to

approximate the image as:

$$\text{image} = \sum_i a_i \mathbf{b}_i + b_0$$

$$a_i \mathbf{b}_i = a_i \cos[2\pi (f_x x + f_y y) + \phi_i]$$

b_0 is the average background light level, and a_i ($= a_i^{\text{in}}$) is the amplitude of the grating. Often, we talk only about the *contrast* of the grating:

$$\text{contrast} = \frac{a_i}{b_0}.$$

The reason is psychophysical and physiological--the human visual system is largely invariant to average background level--contrast corresponds well to the relative variations in brightness that you see in an image pattern, as well as to the neural variations transmitted from the retina to the brain. So for convenience, we can drop the b_0 term which is constant, and if there is negligible absorption by the optics, remains unchanged anyway. The set $\{a_i\}$ is the amplitude spectrum of the image, and $\{\phi_i\}$ the phase spectrum. Both are usually plotted as a function of frequency (here indexed by i).

The Modulation Transfer Function (MTF)

Characterizing an optical system

In order to characterize an optical system, the idea is to measure how the amplitude (contrast) of the eigenfunctions (sine wave gratings) changes as a function of spatial frequency. The ratio of output grating amplitude to input amplitude (as a function of spatial frequency) is called the *modulation transfer function* of the optical system (e.g. eye):

$$a_x^{\text{in}} \sin 2\pi f_x x \qquad a_x^{\text{out}} \sin(2\pi f_x x + \phi)$$

$$MTF_i = \frac{a_i^{\text{out}}}{a_i^{\text{in}}}$$

Of course, the spatial scale will change with optical minification or magnification. We will assume the scale is the same. We can always put the right scale back in with a suitable mapping $x \rightarrow \text{scale factor} * x$. Suppose the input contrast amplitude is fixed, say $a_i^{\text{in}} = 1$. Then a_i^{out} is the MTF. In one dimension, we'll write $a(f) = a_i^{\text{out}}$.

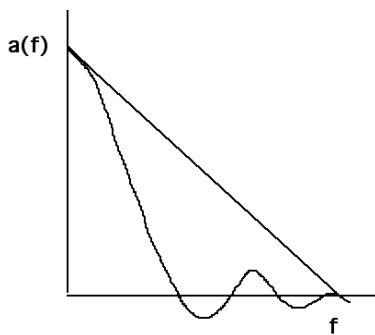
Can you guess what a typical modulation transfer function would do to the following gratings, going from low to high spatial frequencies?

```
g = Table[DensityPlot[0.5` Grating[x, y, fx, 0],
  {x, -1, 1}, {y, -1, 1}, PlotPoints -> 128, Mesh -> False, Frame -> False,
  PlotRange -> {-1, 1}, ColorFunction -> "GrayTones"], {fx, 1, 7, 2}];
Show[GraphicsRow[{g[[1]], g[[2]], g[[3]], g[[4]]}]]
```



Some answers

Here are two possible MTFs:



The form of these curves makes intuitive sense. The amplitudes of low spatial frequency gratings remains largely unchanged going through the optics. On the other hand, at high spatial frequencies, the light and dark bars are close together and get smeared out by aberrations. You might think that the negative part of the MTF is a little unlikely. In fact, it is quite common. An example is when your eye is out of focus. The contrast at those spatial frequencies corresponding to negative MTF undergo phase reversal. This is called spurious resolution. You can't really notice this with ordinary images, but can see it with sinewave grating images.

Visualizing your own contrast sensitivity function (CSF)

Warning--executing the next cell can take some time.

```
CSF[x_, y_] := 127.5 e- $\frac{y}{6.125}$  Cos[2  $\pi$  e $\frac{x}{4}$ ] + 127.5;
csfg = DensityPlot[CSF[x, y], {x, 0, 6}, {y, 0, 1}, PlotPoints -> 512, PlotRange ->
  {1, 254}, Mesh -> False, Axes -> False, Frame -> False, ColorFunction -> "GrayTones"]
```

The nominal physical contrast is constant across any horizontal straight line; however, the subjective appearance of the boundary between the easily visible and nearly invisible contrast transition appears like an upside-down U-shaped function, \cap .

There is an apparent fall-off in contrast sensitivity for high frequencies; however, there is also a drop-off at low frequencies. We'll focus on the high spatial frequency fall-off and return to the reasons for the low frequency loss later.

To re-cap

So the sine waves are the known basis functions, the MTF { t_i 's} is measured, and the spectrum (Li's) can be calculated from an image using a fourier transform subroutine. Then our discrete-hybrid analog of what we introduced at the very beginning of this lecture says that knowing the human eye's MTF, we can calculate the input image \mathbf{g} ,

$$\mathbf{g} = \sum_i (\mathbf{g} \cdot \mathbf{b}_i) \mathbf{b}_i = \sum_i a_i^{\text{in}} \mathbf{b}_i$$

retinal image f , as:

$$f = \sum_i a_i^{\text{in}} \lambda_i \mathbf{b}_i$$

where

$$\lambda_i = \frac{a_i^{\text{out}}}{a_i^{\text{in}}}, \text{ is the MTF.}$$

For the advanced...

For the continuous case, the image at the retina could be represented as an integral of the spectrum, the modulation transfer function, and the basis functions (or eigenfunctions):

$$r(x,y) = \int g(f_x, f_y) T(f_x, f_y) b(x, y; f_x, f_y) df_x df_y$$

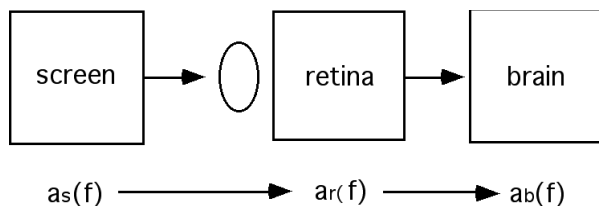
where $L(f_x, f_y)$ is the (complex-valued) spectrum, $T(f_x, f_y)$ the optical transfer function (takes into account phase shifts, that the MTF doesn't), and $\{b(x, y; f_x, f_y)\}$ are the complex-valued eigenfunctions, $\{e^{-2\pi i(xf_x + yf_y)}\}$.

The clever experiment of Campbell and Green

The idea--bypass the optics to measure the "eye-brain" response

How can we measure the MTF of a real eye? The first measurements were made in the late 1950's and 1960's using several techniques. One of the cleverest was developed by Campbell and Green in 1966.

Let us break the way contrast of a grating gets processed into two part



Our goal is to measure the MTF, but to do this would require getting access to $ar(f)$. This seems like a tough problem. Campbell and Green came up with the following solution. Rather than measuring the MTF directly, they measured two other functions. One is called the contrast sensitivity function (CSF) of the human eye. This is measured by having subjects adjust the contrast of a grating until it is just

disappearing, that is where the brain's "response" is always k . One then plots up the reciprocal of contrast (called sensitivity) as a function of spatial frequency. Such a graph is shown below. Using our terminology

$$CSF(f) = \frac{k}{a_s(f)}$$

The second function they measured, we will call the "brain's contrast sensitivity function", or BTF for short. The idea was to present a grating on the retina whose contrast was unaffected by the optics of the eye, and then measure the contrast sensitivity in the same way as for the CSF

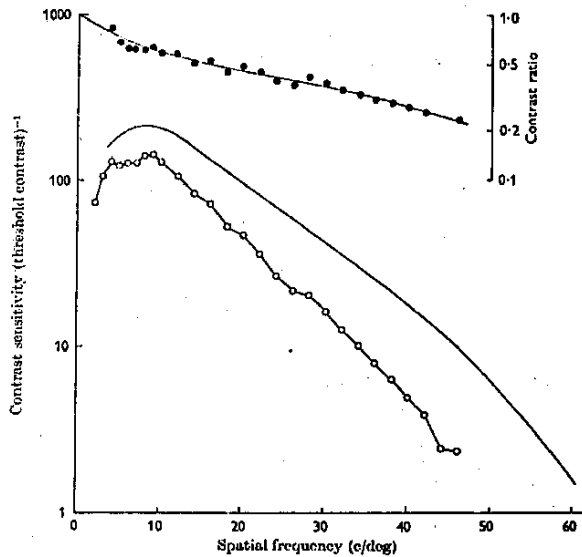
$$\text{"BTF"} = \frac{k}{a_r(f)}$$

$$CSF = MTF \times BTF = \frac{a_r}{a_s} \cdot \frac{k}{a_r} = \frac{k}{a_s}$$

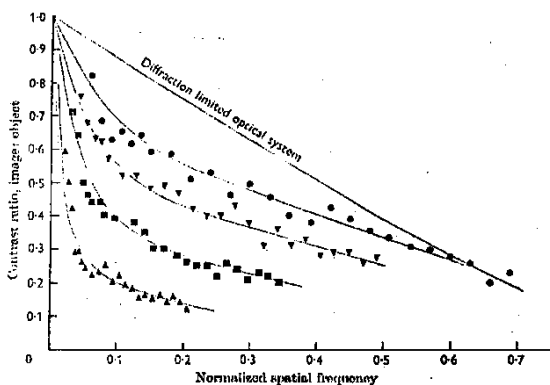
$$MTF = \frac{a_r}{a_s}$$

If we know the CSF and the BTF, we can get the MTF. But how can the BTF be measured? The solution was to image two points of coherent laser light in the pupil of the eye. These two point sources in the pupil produce a sinusoidal interference pattern on the retina of the eye. In fact, the pupil corresponds to the fourier plane of the retina--the fourier transform of a pair of delta functions is a sinusoid, because of constructive and destructive interference (see Appendix and the cosine/dirac delta transform pairs). Further, the shift theorem says that if the points are moved apart in the pupil, the frequency of the grating gets higher, because the retinal pattern shrinks as the pupil fourier pattern expands. Campbell and Green were able to have subjects modulate the contrast of the interference pattern to find the BTF, or contrast sensitivity as a function of spatial frequency with the optics effectively bypassed

The measured CSF and MTF of the human eye



The solid line shows the BTF--we require less contrast to see a grating if the optics are bypassed. The line with the data points shows the CSF. The MTF is the CSF/BTF and is shown in the upper panel

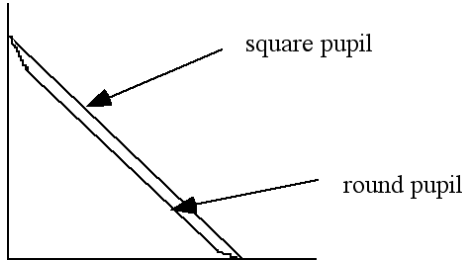


In this figure, the estimated MTFs are plotted for various pupil sizes (2, 2.8, 3.8, and 5.8 mm). The curves are normalized so that the diffraction limit would correspond to a frequency of 1. The solid line is the diffraction limit. You can see that for high frequencies, the eye with a 2mm pupil is essentially diffraction limited. However, aberrations (e.g. spherical) greatly affect contrast for big pupils.

The theoretical diffraction limit and receptor spacing

Diffraction limit

What is the theoretical diffraction limit? The MTF can be calculated by computing the fourier transform of the Airy disk function. For round and square pupils, the curves look roughly like



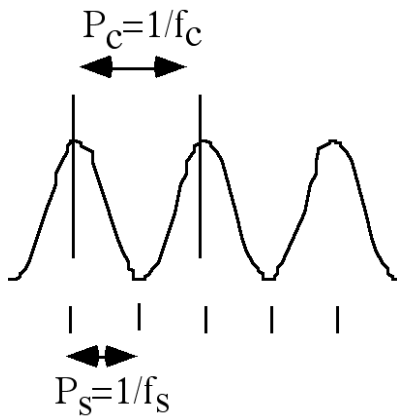
(Note: these are hand sketched, and are meant mainly to show the steady monotonic drop in contrast transfer).

The high frequency cut-off is:

$$f_c = \frac{a}{\lambda} \text{ cycles/radian} = 3,603.6 \text{ c/rad for } a = 2\text{mm, } \lambda = 555\text{nm} = 63 \text{ c/degree}$$

One of the advantages of spatial frequency analysis of the optics is that it gives us a precise description of the information that is lost--i.e. spatial frequencies higher than the cut-off frequency. (There have been efforts to recover this missing high frequency information in optics using analytic continuation, but here one is again thwarted by noise). The period $P_c = .016 \text{ deg} = 1/63$. Now recall that the cone spacing = .008 deg. How well does receptor sampling period, P_s , match the period of the highest frequency in a broad band image spectrum?

The Nyquist limit



This is actually a perfect match (perhaps too perfect?). The Whittaker-Shannon theorem says that one can perfectly reconstruct a continuous band-limited function if the discrete sampling rate, f_s , is at least twice that of the highest frequency in the spectrum

$$f_s > 2f_c \quad P_s < \frac{P_c}{2}$$

The smallest sampling frequency that one can get by with is twice the highest frequency in the spectrum of the signal. This smallest frequency is called the Nyquist rate

$$f_s = f_{Nyquist} = 2f_c$$

If the sampling frequency is less, then aliasing results. Aliasing produces moiré patterns. (See Williams, 1986; Coletta et al., 1990 for examples of the effects of aliasing in human vision). Is aliasing bad for neural processing of images? Allan Snyder once took me down in the basement of the Physiological Laboratories at the University of Cambridge to show me a Garter snake in a terrarium. With a simple ophthalmoscope, we peered into the tiny eye of the snake. We saw the photoreceptor mosaic in striking detail and regularity--the optics were clearly better than the sampling frequency.

We now have the tools to calculate an upper bound on the information capacity of the eye. The modulation transfer function effectively limits the size of a resolution cell, as specified by diffraction and the Whittaker-Shannon sampling theorem, and photon statistics limit the number of distinguishable levels. There are formulas that combine these two factors to give precise measures of the limits to optical capacity in terms of bits.

Why aren't the optics better?

Making the pupil bigger leads to problems with aberrations (spherical).

Making the pupil smaller leads to increasing diffraction blur.

If the pupil was bigger, and aberration could be reduced, we would have to pack the cones more tightly to adequately sample the higher spatial frequencies passed by the optics. It is thought that there may be a physical wave-guide limit of 1-2 micrometers to receptor diameter, across animals of varying acuity (eagle, hawk or human).

Next time

Linear systems and visual neurons

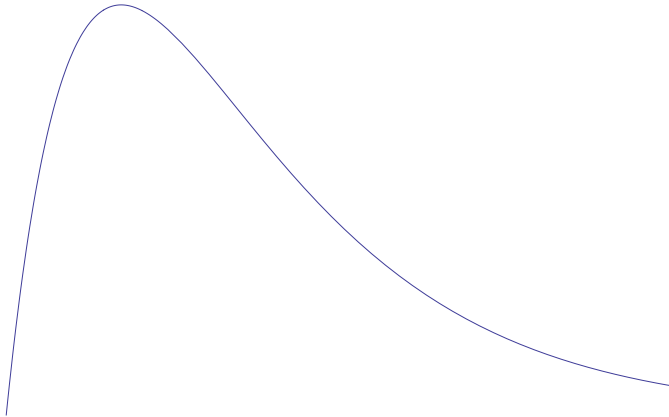
->Multi-resolution, wavelets

->A model of the spatial filtering properties of neurons in the primary visual cortex

Appendices

Model CSF

```
csfpg = Plot[-e-fx + e- $\frac{fx}{2}$ , {fx, 0, 8}, Axes → False]
```



Mathematica has built-in symbolic functions: `FourierTransform[]` and `InverseFourierTransform[]`

$$\text{Fourier transform: } F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt,$$

$$\text{Inverse fourier transform: } f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(\omega) e^{-i\omega t} d\omega$$

1. The fourier transform of the inverse fourier transform of $g[f]$ is $g[f]$.

$g[f]=$

```
FourierTransform[InverseFourierTransform[g[w], w, x], x, w]
```

```
$Aborted
```

2. Convolution theorem: $f * g =$

```
InverseFourierTransform[FourierTransform[f[x], x, w]
```

```
FourierTransform[g[x], x, w], w, x]
```

```
InverseFourierTransform[
```

```
FourierTransform[g[x], x, w] FourierTransform[h[x], x, w], w, x]
```

```
InverseFourierTransform[
```

```
FourierTransform[g[x], x, f] FourierTransform[h[x], x, f], f, x]
```

Fourier transform of a delta function is constant

FourierTransform[DiracDelta[x], x, f]

$$\frac{1}{\sqrt{2\pi}}$$

What is the Fourier transform of Cos[x]?

FourierTransform[Cos[x], x, f]

$$\sqrt{\frac{\pi}{2}} \text{DiracDelta}[-1 + f] + \sqrt{\frac{\pi}{2}} \text{DiracDelta}[1 + f]$$

Some functions have the same shape in Fourier domain as in the spatial domain

FourierTransform[Exp[-(x / σ) ^ 2], x, f]

$$\frac{e^{-\frac{1}{4} f^2 \sigma^2} \sqrt{\sigma^2}}{\sqrt{2}}$$

Note that as the gaussian gets narrower in space, it gets broader in frequency.

References

- Barlow, H. B. (1981). Critical Limiting Factors in the Design of the Eye and Visual Cortex. *Proceedings of the Royal Society London B*, 212, 1-34.
- Boynton, G. M., Engel, S. A., Glover, G. H., & Heeger, D. J. (1996). Linear systems analysis of functional magnetic resonance imaging in human V1. *The Journal of Neuroscience*, 16(13), 4207-4221.
- Coletta, N. J., Williams, D. R., & Tiana, C. L. M. (1990). Consequences of spatial sampling for human motion perception. *Vision Research*, 30(11), 1631-1648.
- Campbell, F. W., & Green, D. (1966). Optical and retinal factors affecting visual resolution. *Journal of Physiology (Lond.)*, 181, 576-593.
- Gaskill, J. D. (1978). *Linear Systems, Fourier Transforms, and Optics*. New York: John Wiley & Sons.
- He, S., & MacLeod, D. I. (1996). Local luminance nonlinearity and receptor aliasing in the detection of high-frequency gratings. *J Opt Soc Am A*, 13(6), 1139-1151.
- Liang, J., & Williams, D. R. (1997). Aberrations and retinal image quality of the normal human eye. *J Opt Soc Am A*, 14(11), 2873-2883.
- Smallman, H. S., MacLeod, D. I., He, S., & Kentridge, R. W. (1996). Fine grain of the neural representation of human spatial vision. *J Neurosci*, 16(5), 1852-1859.
- Williams, D. R. (1986). Seeing through the photoreceptor mosaic. *Trends in Neuroscience*, 9(5), 193-197.