Computational Vision
U. Minn. Psy 5036
Daniel Kersten
Lecture 7: Image Modeling, Linear Systems

Initialize standard library files:

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

Outline

Last time

Bayesian decision theory applied to perception

The task is important: Some causes of image features are important to estimate (primary variables, S_{prim}), and some are not S_{sec}.

Use the joint probability to characterize the "universe" of possibilities:

$$p\left(S_{prim},\ S_{sec},\ I\right) \qquad (1)$$

(Directed) Graphs provide a way of modeling what random variables influences other random variables, and correspondingly how to decompose the joint into a simpler set of factors (conditional probabilities).

Through conditioning on what is known, and integrating out what we don't care to estimate, we arrive at the posterior:

$$p\left(S_{prim} \mid I\right) \qquad (2)$$

And by Bayes' theorem the posterior probability is given by:

$$p[S_{prim} \mid I] = \frac{p[I \mid S_{prim}]\, p[S_{prim}]}{p[I]}$$

$p[I \mid S_{prim}]$ is the likelihood, and is determined by the generative model for I

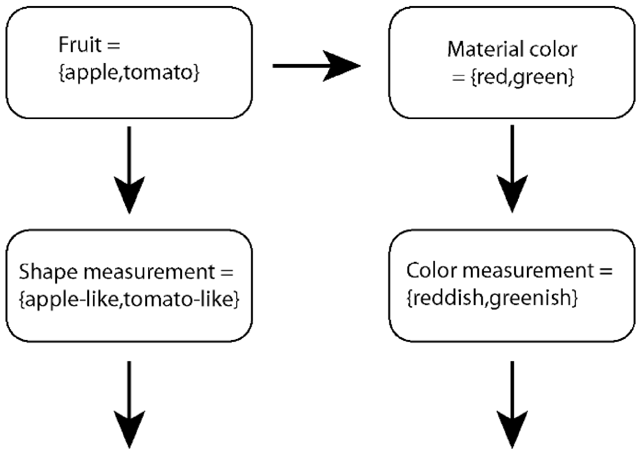
$p[S_{prim}]$ is the prior model,
which can also be considered part of the generative model

$p[I]$ is fixed

The task determines how to use the posterior.

Picking the most probable value of S_{prim} (MAP) results in the fewest errors on average

Counter-intuitive consequences:


Fruit = {apple,tomato}
Material color = {red,green}
Shape measurement = {apple-like,tomato-like}
Color measurement = {reddish,greenish}

Inference: Fruit classification example

Pick most probable color--Answer "red"

Pick most probable fruit--Answer "apple"

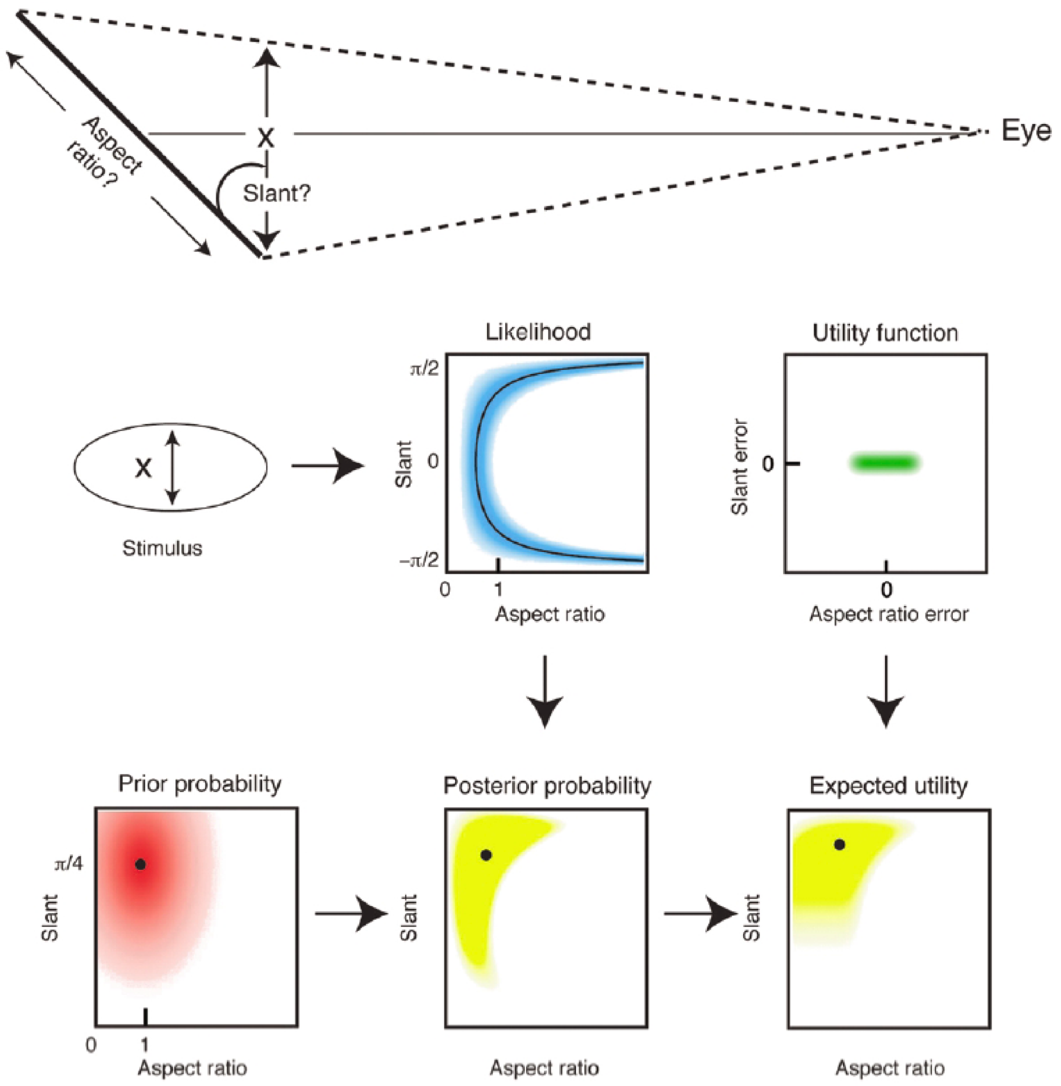
Pick most probable fruit AND color--Answer "red tomato", ***not*** "red apple"

Moral of the story: Optimal inference depends on the precise definition of the task

■ Generalization to degrees of desired precision leads to Bayesian *decision* theory

Slant estimation example illustrates how utility affects optimal estimates

Imagine the image of an ellipse arrives at your retina, and for simplicity that somehow you know the width = 1. Your visual system measures the height of the ellipse in the image (x=1/2) and using this single number must estimate two object properties: 1) the aspect ratio of the actual disk causing the elliptical image, (i.e. the physical dimension of the disk orthogonal to the width); 2) the angle of inclination (slant) away from the vertical.



Today

Generative modeling -- How can we characterize an image, or a collection of images?

Introduction to modeling image structure:

image-based vs. scene-based

Linear systems (linear image-intensity based modeling)

Optical Resolution: Diffraction limits to spatial resolution

Point spread function, convolutions and blur

Overview of image modeling

Generative models for images: rationale

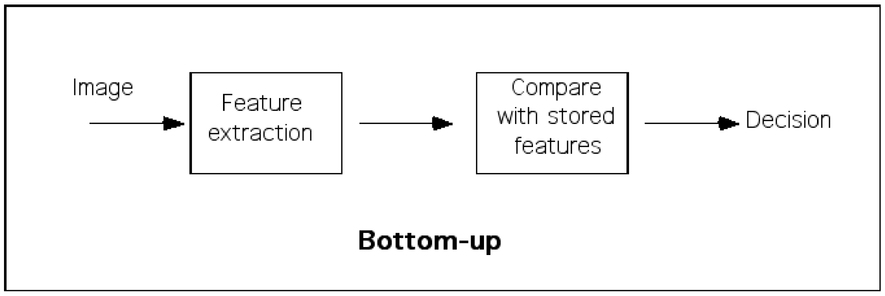
- **Provides practical tools to specify the variables that determine the properties of images, either in terms of image features or scene properties.**
  Important for characterizing the independent variables in psychophysics, and vision models
- **Easier to characterize information flow: Mapping is is many-to-one**

Can be used to model the likelihood of scene descriptions, p(I|S) (i.e. the probability of an image description given a scene description)

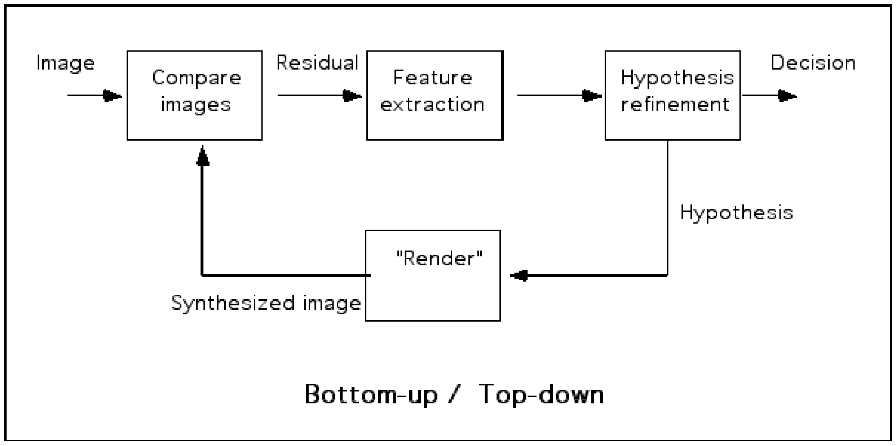
`Often easier to first specify p[I | S] and p[S] than p[S | I].`

- **Characterize the knowledge required for *inference mechanisms, e.g. neural networks for visual recognition***

Feedforward procedures:



Pattern theory perspective: "analysis by synthesis" (Grenander, Mumford)



- **Two basic concepts: Photometric (intensity, color) & geometric variation**
- **Two more basic concepts: Local and global representations**

E.g. an edge can be locally represented in terms of the contrast and orientation at a point of an image. But a long edge (or contour) can also be represented by function with a small number of parameters (e.g. slope and intercept of straight line, or if curved, as a polynomial curve).

An image can be represented in terms of a list of intensities at each location (local), or as we will see shortly, as a linear combination of global patterns (sine-wave gratings or other "basis functions").

- **Two fundamental classes of image modeling:**

  1) image- (or appearance) based

  2) scene based

As an example, ***image-based modeling*** tools are provided in software packages like:

Adobe Photoshop or GIMP, or Adobe Illustrator

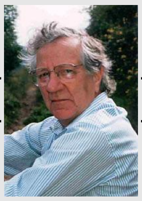
For an introduction to image manipulation using *Mathematica*, see:

http://library.wolfram.com/infocenter/Articles/1906/

Scene-based modeling tools are provided in 3D graphics packages like: Maya, 3DS Studio Max, Sketchup, or software development packages like OpenGL.

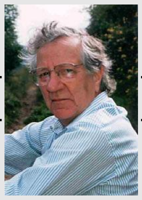
Digression: Getting images into Mathematica

The easiest way is to drag an image from your computer desktop into the argument slot for **Image[**<drop it here>**]**

```
In[26]:= Image[
```

```
]
```

You can find out the image type using **ImageType[]**. Use **ImageDimensions[]** to get the pixel dimensions of the raster.

Often we'll want the raw pixel data in raster format:

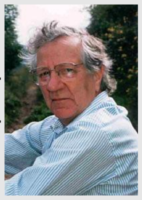
```
In[28]:= testimage = ImageData[
```

```
];
```

Remember if you execute **ImageData[< >]** without a semi-colon, *Mathematica* will output the whole array. We can use **Dimension[]** to see the dimensions of any list, including.

```
Dimensions[testimage]

Out[33]= {287, 200}
```

It is a 256x256 pixel array with 3 values for each pixel, one for each color channel. If we want the gray-level version

```
In[32]:= testimageg = ImageData[ColorConvert[
```

```
, "Grayscale"]];
```

Check the Dimensions of testimageg.

■ *Mathematica* database

Mathematica has a standard database library that includes images. You can get a list of testimages with:

```
In[41]:= ExampleData["TestImage"]
```

```
Out[41]= {{TestImage, Aerial}, {TestImage, Aerial2}, {TestImage, Airplane},
          {TestImage, Airplane2}, {TestImage, Airport}, {TestImage, APC},
          {TestImage, Boat}, {TestImage, Bridge}, {TestImage, CarAndAPC},
          {TestImage, CarAndAPC2}, {TestImage, ChemicalPlant},
          {TestImage, Clock}, {TestImage, Couple}, {TestImage, Couple2},
          {TestImage, Elaine}, {TestImage, F16}, {TestImage, Girl},
          {TestImage, Girl2}, {TestImage, Girl3}, {TestImage, Gray21},
          {TestImage, House}, {TestImage, House2}, {TestImage, JellyBeans},
          {TestImage, JellyBeans2}, {TestImage, Lena}, {TestImage, Man},
          {TestImage, Mandrill}, {TestImage, Moon}, {TestImage, Numbers},
          {TestImage, Peppers}, {TestImage, ResolutionChart}, {TestImage, Ruler},
          {TestImage, Sailboat}, {TestImage, Splash}, {TestImage, Tank},
          {TestImage, Tank2}, {TestImage, Tank3}, {TestImage, TestPattern},
          {TestImage, Tiffany}, {TestImage, Tree}, {TestImage, Truck},
          {TestImage, TruckAndAPC}, {TestImage, TruckAndAPC2}, {TestImage, U2}}
```

ExampleData[{"TestImage", "Peppers"}] can be used to display an image. The test image "Lena" was used extensively in the development of digital compression and other image processing algorithms. Check wikipedia for a quick overview of the controversy surrounding its use.

Image-based modeling

Four classes of image-based models

■ Linear intensity-based

(Photometric)

Basic idea is to represent an image $\vec{I}$ in terms of linear combinations or sums of other images $\vec{I_i}$:

$$\vec{I} = m_1 * \vec{I_1} + m_2 * \vec{I_2} + m_3 * \vec{I_3} + \ldots$$

where m_i's are scalars. You can think of each image as a point in an N-dimensional space, where N is the number of pixels, and the dimensions of the point correspond to the pixel intensities.

In[53]:=
```
rcolorpic1 = ImageData[
```

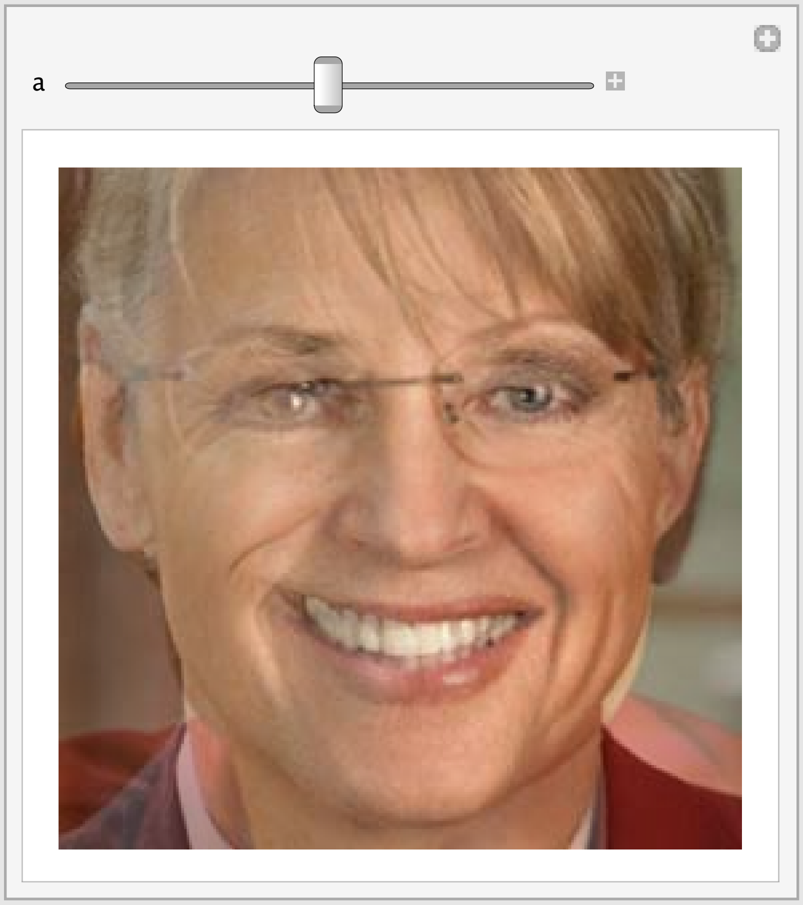
```
];
```

```
rcolorpic2 = ImageData[
```

```
];
```

In[57]:=
```
Manipulate[Image[a * rcolorpic1 + (1 - a) * rcolorpic2], {{a, .4}, 0, 1}]
```

Out[57]=



Linear models get mathematically interesting when we begin to talk about *linear transformations*. An clear image on a computer screen can get transformed by poor optics by the time the information lands on the retina. The powerful idea behind linear systems is that a transformed image can be represented by the sum of its transformed "sub-images" or "basis images". This is the principle of superposition that will crop up repeatedly later.

A quick preview of the idea of an *image basis set* { $\overrightarrow{I_i}$} :

Complete:

Means you have enough basis images that you can construct any image from a weighted sum of the basis images

$$\vec{I} = m_1 * \overrightarrow{I_1} + m_2 * \overrightarrow{I_2} + m_3 * \overrightarrow{I_3} + \ldots$$

Will lead us to linear shift-invariant systems--fourier synthesis

application: optics of the eye

efficient representation of natural images -- wavelets (sometimes "over-complete")

application: V1 cortical neuron representation, edge detection

V1, MT motion-selective neurons

Incomplete:

lossy compression, principal components analysis (PCA)

characterizing image variation in specialized domains

application: illumination variation for fixed views of an object

■ Non-linear intensity-based

(Photometric)

Point non-linearities (e.g. gamma correction for computer display): **intensity at pixel i <- function(intensity at pixel i)**

General non-linearities: **intensity at pixel i <- function(intensity at pixel j, j in some neighborhood of i)**

-- Polynomial expansions (a generalization of the linear case): the function is a polynomial function of the neighboring pixel intensities

-- Divisive normalization (used for gain control): function is a combination of linear and divisive terms

application: V1 cortical neurons

■ Linear geometry-based

Geometry-based models use operations on the pixel locations rather than pixel intensities.

Affine:

$\{x_i, y_i\} \rightarrow \{x'_i, y'_i\}$, $\{x'_i, y'_i\} = \mathbf{M}.\{x_i, y_i\}$, where **M** is a 2x2 matrix

rigid translations, rotations, scale and shear

(e.g. in 2D, all the transformations that result from applying the 2x2 M matrix

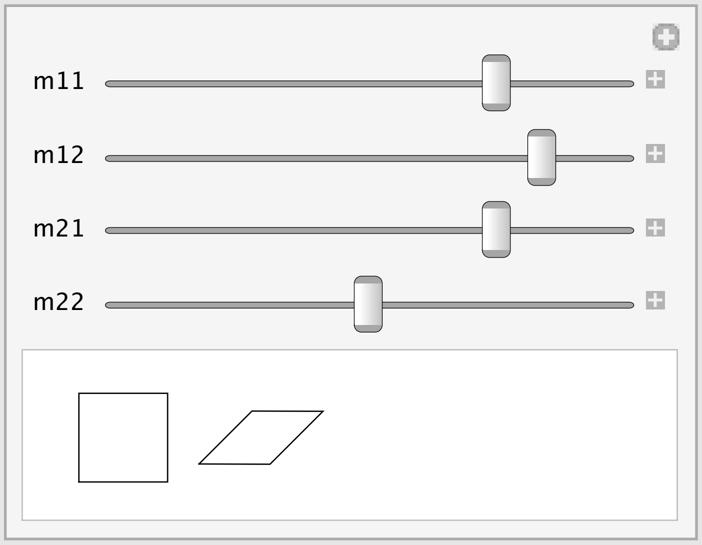
to each of the coordinates of points $\{x_i, y_i$ in an image)

Application: viewpoint variation

You can play with the values in the matrix M below to transform the rectangle x into new shapes x2:

In[58]:=
```
Manipulate[
 x = {{0, 0}, {0, 1}, {1, 1}, {1, 0}, {0, 0}};
M = {{m11, m12}, {m21, m22}};
x2 = (M.#1 &) /@ {{0, 0}, {0, 1}, {1, 1}, {1, 0}, {0, 0}};
GraphicsRow[{Graphics[Line[x]], Graphics[Line[x2]]},
  ImageSize → Tiny],
 {{m11, 1}, -2, 2}, {{m12, 2}, -2, 2}, {{m21, 2}, -2, 2},
 {{m22, 0}, -2, 2}]
```

Out[58]=



■ Non-linear geometry-based

Warps (Morphs)

$\{x_i,y_i\}\to\{x'_i,y'_i\}$, where the transformation is no longer a single matrix, and each point can get mapped in arbitrary ways. By warp or morph, we usually mean a smooth mapping.

Applications in vision:

within-category variation for an object, or objects

finding the "average" face

(e.g. Perrett, D. I., May, K. A., & Yoshikawa, S. (1994). Facial shape and judgments of female attractiveness. Nature, 368, 239-242.)

computing correspondence between pixel points in two images, such as optic flow, and stereo vision

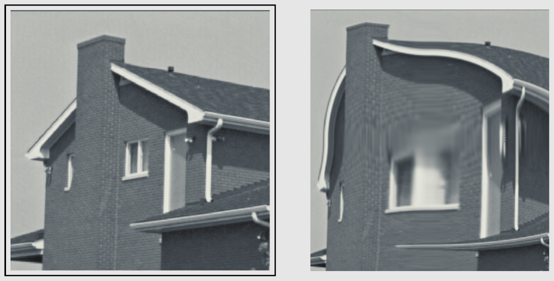
In[78]:=
```
grayscalepic =
  ImageData[ColorConvert[ExampleData[{"TestImage", "House"}],
    "Grayscale"]];
g1 = ArrayPlot[grayscalepic, Mesh → False, ColorFunction → "GrayTones",
   DataReversed → False, ImageSize → Tiny];
```

In[127]:=
```
grayscalepicFunction =
  ListInterpolation[Transpose[Reverse[grayscalepic, 1]],
   {{-1, 1}, {-1.19, 1.19}}];
```

(Transpose[] and Reverse[] were used so the image upright below would be upright.)

In[130]:=
```
g2 = DensityPlot[grayscalepicFunction[Sign[x] x^2, Sign[y] y^2],
   {x, -1, 1}, {y, -√1.19, √1.19}, PlotPoints → 100, Mesh → False,
   AspectRatio → Automatic, Frame → None, ColorFunction → "GrayTones",
   ImageSize → Tiny];
GraphicsRow[{g1, g2}]
```

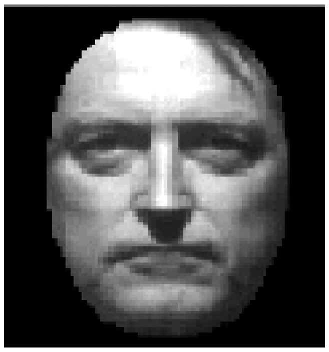
Out[131]=



Pattern theory perspective (Grenander, Mumford)

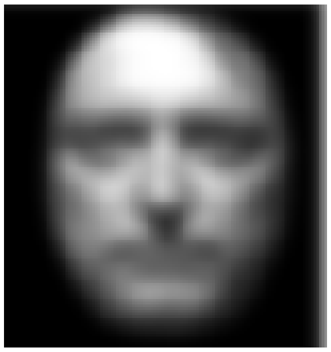
Is there a convenient set of operations to describe the kinds of transformations that are typically undergone by natural patterns such as images? Grenander's pattern theory:

■ Original

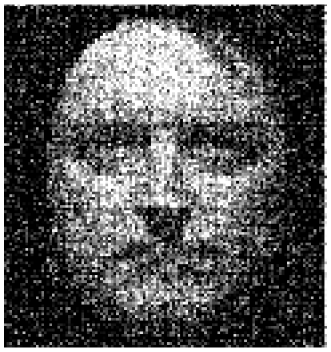


■ Superposition: blur, additive noise, transparency

photometric



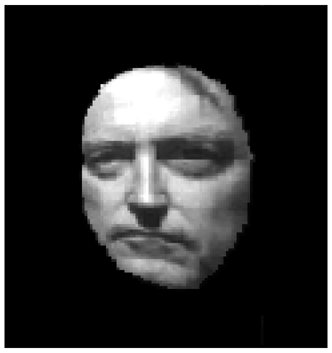
added noise



transparency: sum of two images

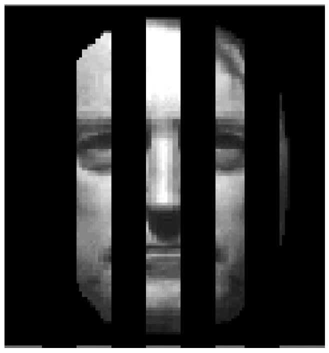
■ Domain warping (morphs)

geometric



■ Domain interruption (occlusion)

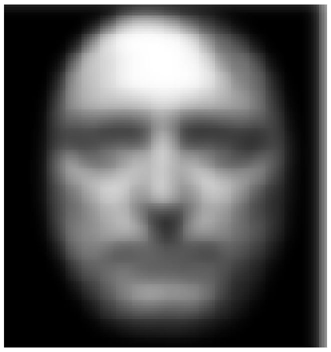
geometric



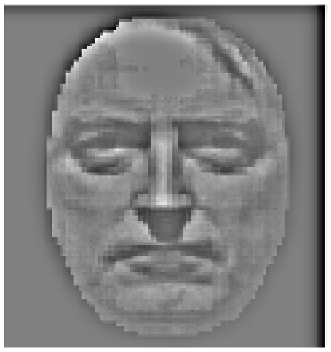
■ Processes occur over multiple scales

photometric and geometric

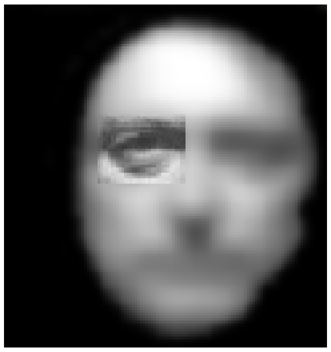
Low-pass



High-pass

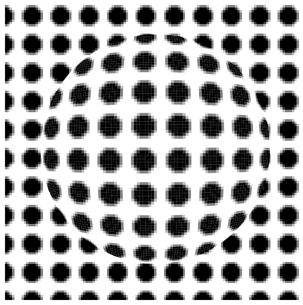


Large scale structures, and small scale structures



We'll see later how local edges can be seen as a kind of cooperation over multiple scales

Texture (regularities at small scales) can interact with regularities at larger scales (surface shape) to produce patterns like this:



■ Combinations

We need various combinations of transformations to account for the kind of variations that we'd see in other forms of the same face. For example, a 3D rotation produces both domain warping and occlusion. A beard is another form of domain interruption. Expressions are a combination of warping, occlusion, and dis-occlusion.

David Mumford

Ulf Grenander

(From: http://www.dam.brown.edu/ptg/ and http://www.dam.brown.edu/people/mumford/)

Scene-based modeling: 3D Computer graphics models

■ Scenes

Variable classes : extended surfaces, solid objects, diffuse "stuff"/particles, lights, and camera (eye).

■ Objects & surfaces

Shape: Lots of choices...parametric, polygon composites, hierarchical composites,...

Articulations: E.g. dynamic linkages between parts, as in the motion of an animal

Material & texture: E.g. pigmentation, shiny, reflectance...

■ Illumination

Points and extended light sources

Ray-tracing: Physics-based modeling of individual light rays and how they bounce from one object to the next

Radiosity: equilibrium solution of light ray bounces

■ Camera

Projection geometry:

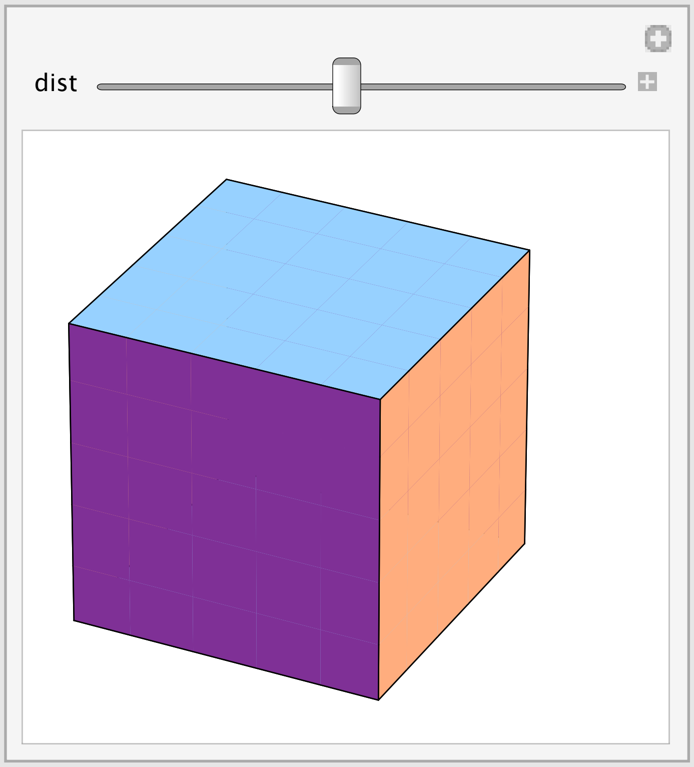
perspective, orthographic

focus -- can be modeled as image-based in one depth plane, but need knowledge of depth in 3D for depth-of-field effects

Demonstration of orthographic projection.

In[132]:=
```
Manipulate[Graphics3D[Cuboid[], ViewPoint → dist * {Pi, Pi / 2, 2},
  ImageSize → Small, Boxed → False], {{dist, 1}, 1, 10}]
```

Out[132]=



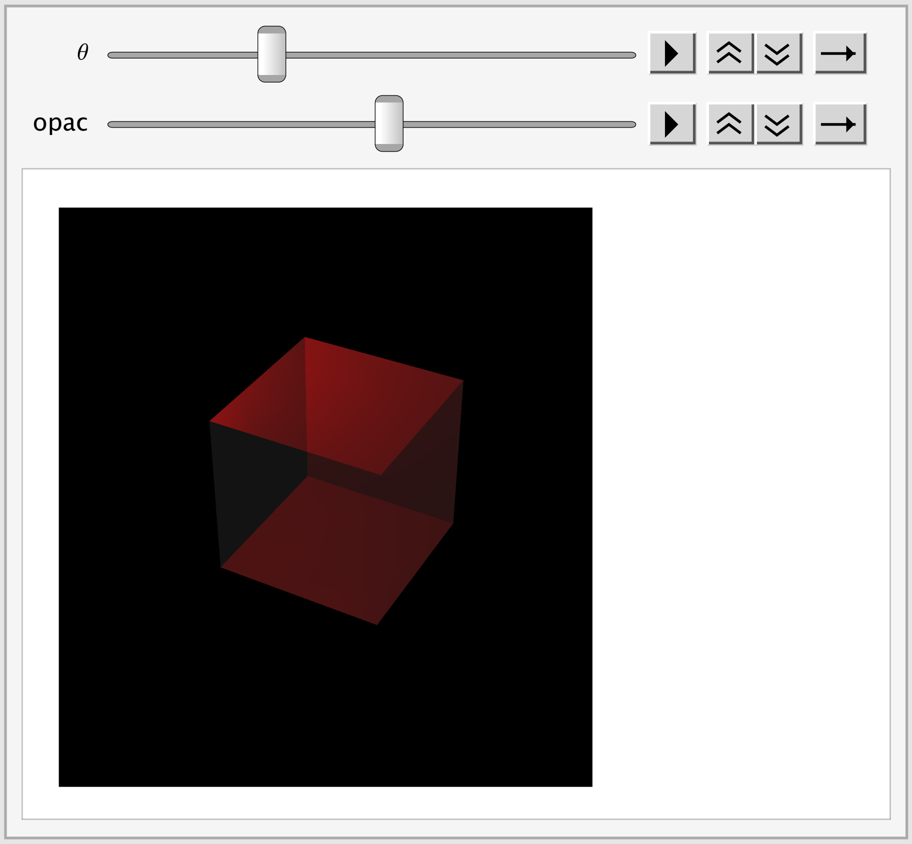
This demo also illustrates a visual illusion--what is the illusion?

■ *Mathematica* provides some primitive 3D modeling tools

See the *Mathematica* Documentation on **Lighting** for some examples.

In[133]:=
```
p1[θ_] := RotationTransform[θ, {0, 0, 1}][{0, 1, 1}];
Animate[
 Graphics3D[{Specularity[White, 30],
   {EdgeForm[], Opacity[opac], Cuboid[{-.5, -.5, -.5}]},
   Sphere[{.3, .3, .3} + p1[θ], .2]},
  Lighting → {{"Ambient", GrayLevel[.1]}, {"Point", Red, p1[θ]}},
  PlotRange → 1.0, Background → Black, Boxed → False, ImageSize → 200],
 {θ, 0, 2 Pi}, {{opac, .5}, 0, 1}, SaveDefinitions → True,
 AnimationRunning → False]
```

Out[133]=



Digression: Getting objects into Mathematica

Mathematica also maintains a library of standard 3D objects

In[44]:=
```
ExampleData[{"Geometry3D", "UtahVWBug"}]
```

Out[44]=

■ Preview of general inference issues for vision:

Edge ambiguity: What is the cause of an intensity change in the image?

Scene-based modeling prompts questions such as is an image intensity change due to: Depth change, orientation change, material change, illumination change?

Projection invariants--for inference

Pros and cons

Pros of image-based modeling:

closer in some sense to the image data representations, "features", that visual perceptual inference deals with.

The so-called "proximal" stimulus corresponds to image information or features directly tied to the image.

Cons of image-based modeling:

usually far from the kinds of representations useful for representing invariant object properties and planning actions

scene-based modeling specifies the properties of the "distal stimulus", like depth, shape, material.

Moral: Good to be familiar with both kinds of generative image modeling

In vision, historically the most common type of modeling begins with linear-intensity based models, which we treat next.

Linear systems & images: Motivation

Often when you look through a reflecting glass, you can see one image superimposed transparently on another, as in the above demo in which the intensities of two faces were averaged. But superposition is one of the image transformations of pattern theory that crops up in many different circumstances other than transparency, from early vision to high-level vision. The idea is that a particular image intensity pattern can be expressed as a weighted sum of other images. Being able to combine weighted sums of images can be useful in domains quite different from reflections in a window.

Here is a simple thing you could do by combining images. You could calculate the "average" face. This has been done, and the average face is rated as "beautiful", if tending to be a bit fuzzy even with careful registration. This raises the question: when is it appropriate to model image variation in terms of sums? It turns out that taking the average of a set of images, such as faces, is tricky because there is not only photometric variation, but also geometric variation. I.e. the faces need to be put in register, and geometric registration is a non-trivial problem, and one that has been studied in many domains such as biometrics.

Rather than adding images up, suppose we have an image, and want to decompose it as a sum of other sub-images. A natural question to ask is: what should the other "sub" images be? The answer will depend on the task. For example, below we will learn to model how the optics of the eye transform an input image to retinal image. A natural "sub-image" is the image of a point of light or a pixel. These sub-images are the extreme case of being spatially quite localized.

But we will see there are advantages to using sub-images that are not spatially localized, and are global, or somewhere between. An important class of global sub-images are sinusoidal gratings.

Later, when we study early neural mechanisms in vision, we will see that models of image coding can be understood as a decomposition of the visual image into image-like sub-components that make a local/global trade-off. We will see evidence for neural basis images that look like two-dimensional "wavelets" that if summed up (with appropriate weights) would give us back the orginal image.

As another example, it is possible to model the space of human faces in terms of a linear sum of "basis faces" whose number is carefully chosen to be as small as possible.

Much later in this course, we will see a less obvious consequence of superposition of light in a method to deal with illumination variation in object recognition. A good approximation to the space of all images of an object can be obtained by summing a small set of appropriate sub-images.

In order to understand how to use superposition, we need to understand the principles of linear systems analysis. And in order to introduce linear systems, we will start with a historical question we asked in lecture 2:

What are the physical limits to vision?

And in particular ask a question scientists have been asking for at least three centuries:

What are the physical/optical limits to spatial resolution?

Optical Resolution

Factors limiting spatial resolution

As we have noted earlier, the physical limits to visual light discrimination and resolution are a consquence of the quantum and wave nature of light. We have seen how the particle aspect of light limits resolution of intensity differences. The wave nature of light limits resolution of spatial differences.

As two identical small points of light get closer and closer, we eventually reach a point at which we can no longer tell whether we are looking at one or two spots of light. What are the limits to our ability to resolve spatial separation?

Physical/optical

Diffraction

(Recall Paul Dirac's famous statement about photons:

"*Each photon interferes only with itself. Interference between photons never occurs.*")

Optical aberrations

spherical, chromatic, focus, astigmatism

Biological

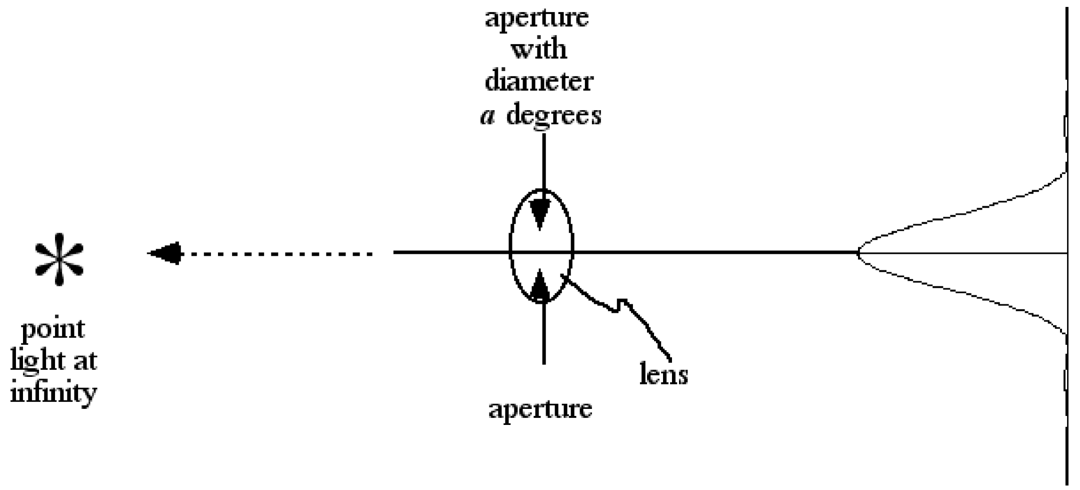
Sampling--retinal receptor array is discrete

Neural convergence (may lead to loss of resolution information)

Diffraction

As with limits to intensity discrimination, our understanding of the limits to spatial resolution begins with the physics of light. Diffraction effects are caused by the constructive and destructive interference of many wavefronts. (The word interference is a result of the same underlying causes, but typically is used when talking about wave phenomena involving only a small number of wavefronts).

It is possible to mathematically describe what the image of point of light looks like under ideal optical conditions. The image is not a point, because of diffraction. The many wavefront sources around the aperture produce a characteristic pattern for a circular pupil called an Airy disk (1834):



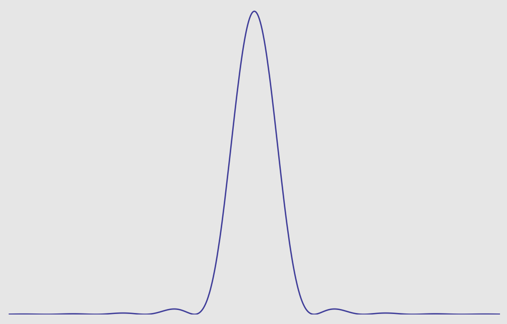
This distribution of light has a very precise form depending on the pupil shape. For a circular pupil, the light energy distribution as a function of distance x away from the center axis, is given by the Airy disk function:

$$\left[\frac{J_1(\pi x)}{\pi x}\right]^2$$

In[45]:=
```
Airy[x_, y_] := If[x == 0 && y == 0, 1/4, (BesselJ[1, π Sqrt[x^2 + y^2]] / (π Sqrt[x^2 + y^2]))^2];
Plot[Airy[x, 0], {x, -5, 5}, PlotRange → {0, 0.26}, Axes → False]
```

$$\text{Airy}[x\_, y\_] := \text{If}\left[x == 0 \,\&\&\, y == 0,\ \frac{1}{4},\ \left(\frac{\text{BesselJ}\left[1, \pi\sqrt{x^2+y^2}\right]}{\pi\sqrt{x^2+y^2}}\right)^2\right];$$

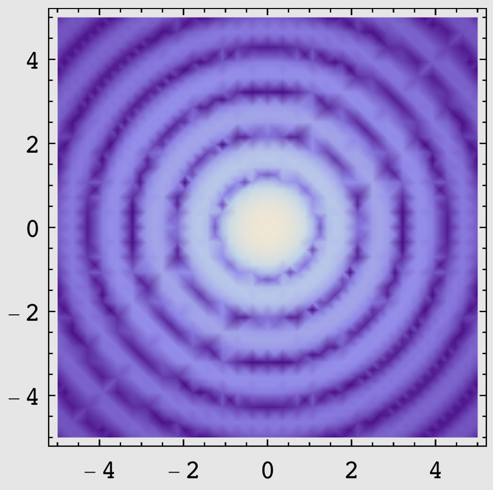
Out[46]=



where J_1 is a first order Bessel function--which if you haven't run into them before, is another one of a large class of "special functions" mathematicians, engineers, and physicists keep in their bag of tricks because they come in handy (See Gaskill, 1978 for details about imaging optics; Also look at section 6.9 in "Robot Vision" by Horn, 1986 for a discusion of the zero and first order Bessel functions in the context of Hankel transforms, and see the *Mathematica* Help Browser for the differential equation for which Bessel functions are the solution). We won't go into the details of the derivation. Here is a two dimensional density plot, where the intensity is squashed by a log function so you can see the ripples (which actually fade away rather quickly as you can check by playing with the PlotRange in the linear cross-section plot above):

In[47]:=
```
LogAiry[x_, y_] := Log[0.00001 + Airy[x, y]];
DensityPlot[LogAiry[x, y], {x, -5, 5}, {y, -5, 5}, Mesh → False,
  ImageSize → Small]
```

Out[48]=



Some of the earliest psychophysics on the limits to spatial resolution was done by astronomers. In 1705, Hooke determined that resolution by eye of two stars was pretty good around 3 min, but became difficult around 1 min of arc. Can we account for human spatial resolution in terms of diffraction limits?

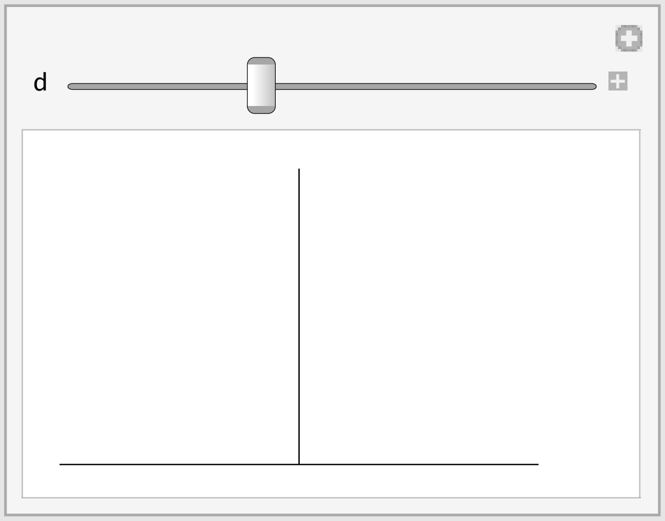
With 2 points of light, the diffraction patterns overlap. Imagine two of the above Airy functions superimposed exactly on top of each other

In[134]:=
```
twopoints[x_,d_] := Airy[x+d,0] + Airy[x-d,0];
```

Let's plot the intensity distribution of two points with several separations: d= 1, 0.5, and 0:

In[135]:=
```
Manipulate[Plot[twopoints[x, d], {x, -7, 7}, Ticks → None,
  ImageSize → Small, PlotRange → {{-3, 3}, {0, 0.5}}], {{d, .5}, .3, 1.5}]
```

Out[135]=



What if the light level was very low--is the above two-point plot representative of the signal?

■ A back-of-the-envelope calculation

When are the points distinguishable?

To answer these questions rigorously would require a signal detection theory model based on the noisiness of the light samples obtained to make a decision (see Geisler, 1984; 1989). This would require bringing what we learned about Poisson statistics into the picture. But let's ignore the photon noise for the moment, and assume that the temporal integration time is sufficient to absorb lots of photons, so that the image intensity profiles are smooth as in the above figure. Then we can make some "back of the envelope calculations just based on diffraction. Let's proceed with a little less rigor, and ask: when are two points separated far enough so that the first zeros of the 2 Airy disks coincide?

The distance from the peak to the first zero is given by: x = 1.22λ/a, where a is the pupil diameter, so the distance between the two peaks when separated by twice this distance is:

$$d = \frac{2 \times 1.22\lambda}{a}\ radians$$

$$a = 2 \times 10^{-3} meters,\ \lambda = 555 \times 10^{-9} meters$$

picking 2mm pupil, and the wavelength corresponding to peak photopic sensitivity.

We can then calculate the separation between the two points to be: **d = 0.04 degrees** of visual angle, or 2.3' of arc. (By the way, it is useful to bear in mind that, in the human visual system:

receptor spacing ~ 0.008 degrees in the fovea, or, about 0.48'.

■ ..but wait a minute...let's check our assumptions

Now let's how what we've just done is related to the observation made by Hooke in 1705. At first you might think that 2' of arc is close enough to explain the 3' that Robert Hooke reported. But there were a couple of things wrong with our choices for both pupil size and wavelength. These figures aren't really appropropriate for scotopic vision where the pupil would be larger, and we should use the peak sensitivity for dark adapted rod vision.

If the pupil is 8 mm in diameter, and the wavelength is 505 nm, the predicted d would be smaller--about 0.008 degrees, which corresponds to 0.5' of arc. This is too small to account for the observed limit to human resolution of between 1 and 3' of arc for spatial resolution of two points in the dark.

Further, we were actually quite liberal in allowing this big of a separation between the two points. In fact, if there were no intensity fluctuations to worry about (imagine averaging the image of the two points over a long period of time), then there would be no theoretical limit to the separation one could discriminate, if the observer had an exact model of the two images (two vs one point) to look for. Factors other than diffraction enter to limit spatial resolution of two points in the dark. The main limitation is the separation of the rods, and the degree of neural convergence.

Although our ability to resolve two points for scotopic vision is worse than we would calculate based on diffraction-blur alone, there are conditions under bright light were our spatial resolution comes very close to the diffraction limit. But we need some more powerful mathematical tools to make this point solid. Later we will introduce a technique--spatial frequency analysis (or 2D Fourier analysis)--to study spatial resolution in general, that will include diffraction as a special case. One of the consequences of spatial frequency analysis of imaging optics is that we will see that there is information which is lost when going through the optics, and noise averaging will not help to get it back.

But first let's generalize the idea of the diffraction function to a point source, and see what it could buy us. This will give us a preview of the kind of predictions we would like to be able to make about image quality, and introduce a simple mathematical notion that has broad uses.

Point spread functions, blur, and convolution

Going from two points to lots of points

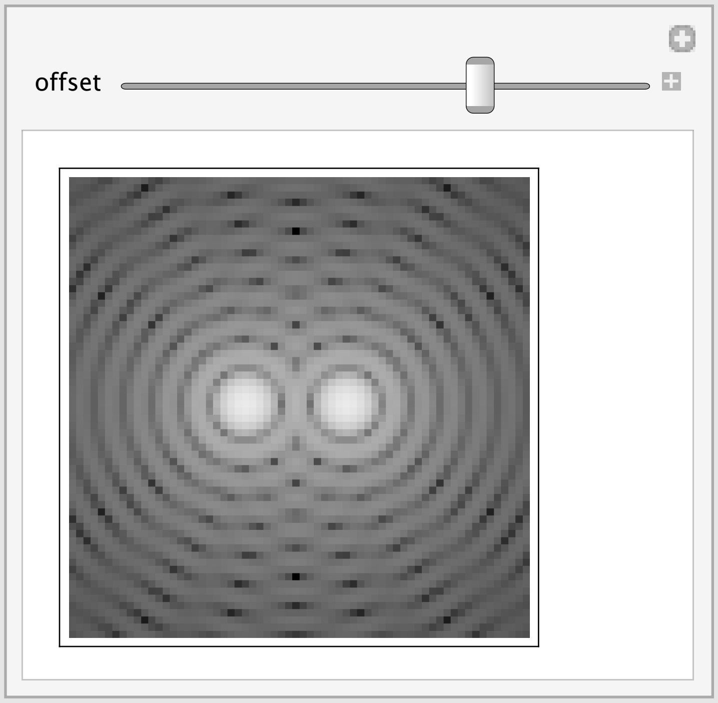
Incoherent light adds. So we can use the principle of superposition to predict what the image of two points of light will look by summing up the images corresponding to what each would look like alone. This is the key idea behind linear sysems analysis pursued in greater detail below.

In[136]:=
```
offset = 2;
size = 64;
onepoint = Table[0., {size}, {size}];
anotheronepoint = Table[0., {size}, {size}];

anotheronepoint⟦size / 2, size / 2 + offset⟧ = 1.;
onepoint⟦size / 2, size / 2 - offset⟧ = 1.;
```
$$\texttt{twopoints} = \frac{\texttt{onepoint + anotheronepoint}}{2};$$

In[143]:=
```
Manipulate[
 oneblurredpoint = Table[Airy[(x - size / 2)/4, 1/4 (y - (size / 2 - offset))],
   {x, 1, size}, {y, 1, size}];
 anotheroneblurredpoint =
  Table[Airy[(x - size / 2)/4, 1/4 (y - (size / 2 + offset))], {x, 1, size},
   {y, 1, size}];
 ArrayPlot[Log[oneblurredpoint + anotheroneblurredpoint],
  Mesh → False, ImageSize → Small], {offset, 0, 10, 1}]
```

Out[143]=



If we have N points, then we just add up the images of each of them. This kind of operation occurs so frequently in many domains, that we will now formalize and generalize what we've just illustrated.

■ Point spread function (impulse response function)

We see that a theory of spatial resolution based on diffraction alone is not enough to account for spatial resolution. One of the reasons is that the actual pattern at the back of the eye due to a point source of light, in general, is not an Airy function. This is because of other optical aberrations (e.g. spherical, chromatic, astigmatism) in addition to diffraction, that blur the image.

In general, an optical system's response to a delta function input (point of light) is called a point spread function (PSF):

$$\delta(x, y) \to PSF(x, y)$$

The Dirac delta function is an infinite spike at 0, but its area is 1:

```
Clear[x];
Integrate[DiracDelta[x], {x, -Infinity, Infinity}]
```

```
1
```

(Recall that we encountered the Dirac delta function earlier in the formalism to represent discrete probability distributions as densities.)

Below we will study convolutions. One example of a convolution is the following integral ("convolution of Dirac delta function with a function f1()). It may look sophisticated, but the net result is that it is equivalent to evaluating f1[x] at t:

```
Integrate[f1[t - x] DiracDelta[x], {x, -Infinity, Infinity}]
```

```
f1(t)
```

A delta function, for us, is a unit point source of light--it has no size, but the light intensity adds up to 1--this is a tricky definition, because it means that the intensity at (0,0 has to be infinite (the delta function is also called the unit impulse, e.g. "Robot Vision" by Horn, chapter 6). It is a useful idealization that can make calculations easier. The convolution property of Dirac delta functions is important and crops up frequently.

For the special case of diffraction limited imaging with a circular aperture, the PSF is the Airy disk function.

Suppose for the moment that we have measured the point spread function (or "impulse response") of an optical imaging system--e.g. your eye. With enough information we could calculate it too--this is a branch of Fourier Optics. In any case, if we know the point spread function of the eye's optics, could we predict the form of the image on the retina, given an arbitrary input image? Yes. It is fairly straightforward if we can assume that the optics are linear and spatially homogeneous (a fairly good approximation for the optics of the eye over small patches called isoplanatic patches). Given an input image l(x,y), the output r(x,y) is given by the convolution of l(x,y) with the point spread function, PSF(x,y):

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

The idea is to treat each point in the input image as generating a smeared-out pattern (i.e.the PSF) on the output image whose intensity value is scaled by the input intensity l(x,y).Then add up all the contributions over the whole image.The short hand for convolution is

$$r(x,y) = l(x,y) * PSF(x,y)$$

Among other properties, you can show that the convolution operation commutes: f*g = g*f.

Understanding convolutions is important, not only for understanding optical image formation, but also because the formalism is applied in a number of models of the neural processing of images. Convolution is often used as a step to describe how image information is transformed by neural networks. So, r(x,y) could also be "neural image" response due to some neural analog of the point spread function. We will see too that convolution or pre-blurring is an important step in edge detection algorithms in computer vision. But more on this later.

If we know the point spread function of the eye's optics, we could predict the image intensity pattern for an arbitrary input. Further, given a linear approximation to neural image transmission, we might be able to find a "neural point spread function", and thereby predict the form that neural images might take at various stages of early vision. In some cases, the neurophysiologist's analog for the neural point spread function is receptive field. One has to be careful, because this analogy is only good to the extent that neural networks are spatially homogeneous (the receptive field of a single neuron can be modeled by the same set of weights from each input; the "spread" function corresponds to the weights assigned multiple outputs given a single input unit), and behave as if they are linear--e.g. calculating the response as the area under the input pattern times the receptive field. Under a number of instances, these are reasonable approximations.

■ Convolutions with Mathematica

Suppose we have measured the PSF, let's see how various PSF patterns affect the input image. First, we will get a sample image file, Fourier128x128.jpeg.

Under the **Input** menu, you can use **Get File Path** to find the directory of a file that you'd like to read in, and then set the current working directory to that location.

You can use this combination of function calls to bring up a window to hunt for a file whose directory you want to be the default working directory. Once you've set your working directory, you can read in a jpeg file.

Or we can just drag it in:

```
In[155]:= fourier = ImageData[
```

```
];
```

In[157]:=
```
ArrayPlot[fourier, Frame → False, Mesh → False, AspectRatio → Automatic,
  ImageSize → Small, ColorFunction → "GrayTones", DataReversed → False]
```

Out[157]=

You can get the width and height using: **Dimensions[fourier]**.

We set up a simple kernel in which each element is 1/64 in an 8x8 array. Convolution replaces each pixel intensity with the average value of the 64 values in the square image region surrounding it:

In[169]:=
```
kernel = Table[1/64, {i, 1, 18}, {j, 1, 18}];
```

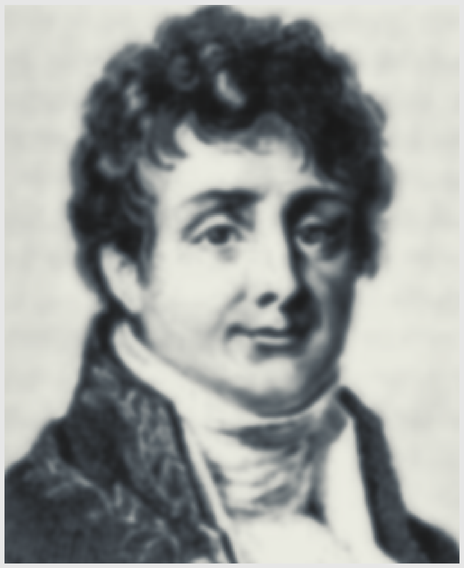
In[170]:=
```
blurFourier = ListConvolve[kernel, fourier];
ArrayPlot[blurFourier, Frame → False, Mesh → False,
 AspectRatio → Automatic, ColorFunction → "GrayTones",
 DataReversed → False, ImageSize → Small]
```

Out[171]=

In[166]:=
```
kernel = Table[Airy[x, y], {x, -2.3, 2.3, .3}, {y, -2.3, 2.3, .3}];
```

In[167]:=
```
blurFourier = ListConvolve[kernel, N[fourier]];
ArrayPlot[blurFourier, Frame → False, Mesh → False,
 AspectRatio → Automatic, ColorFunction → "GrayTones",
 DataReversed → False, ImageSize → Small]
```

Out[168]=



It is technically hard to measure the response to a delta function for most real (physical or biological) devices--infinite or near infinite values in the input fall outside any approximately linear range the device might have. An alternative is to measure the responses to other basic image patterns that are formally related to point spread functions. This problem leads us to a more general consideration of linear systems theory, and in particular to the spatial frequency or Fourier analysis of images. Spatial frequency has a number of other nice properties over point spread function characterizations. One, alluded to earlier, is that it will make explicit the kind of information that is lost, irrespective of the noise considerations.

Exercise: Astigmatism

Modify the above blurring kernel so that it blurs more in the horizontal than the vertical direction. This would simulate the effect of astigmatism.

Exercise:

Now try a kernel = {{0,1,0},{1,-4,1},{0,1,0}};

```
kernel = {{0, 1, 0}, {1, -4, 1}, {0, 1, 0}} // MatrixForm
```

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

- **Convolution is a special case of a a linear operator. The next section looks at the general discrete case.**

Next time

Linear shift-invariant systems: Fourier analysis of images & optics

- **Sinewave basis elements**
- **The Fourier Transform**

Appendices

Other ways of importing images

```
In[146]:= SetDirectory[DirectoryName[SystemDialogInput["Directory"]]]
```

```
{FileNameSetter[Dynamic[f]], Dynamic[f]}
```

```
{ Browse... , /Users/kersten2/Sites/kersten-lab/courses/Psy5036W2010/Lectures/7.ImageModelingLine
              Biden.jpg
```

```
colorpic = Import[f]
```

Gray scale plot

You can specify weights for how to mix the red, green, and blue channels:

```
grayscalepic = Map[(0.3 #[[1]] + 0.59 #[[2]] + 0.11 #[[3]])/255 &, colorpic;
  ArrayPlot[grayscalepic, DataReversed → True,
   ColorFunction → "Graytones"]
```

References

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.

Geisler, W. S. (1984). Physical limits of acuity and hyperacuity. *J. Opt. Soc. Am. A, 1,*, 775-782.

Geisler, W. (1989). Sequential Ideal-Observer analysis of visual discriminations. *Psychological Review, 96*(2), 267-314.

Grenander, U. (1996). *Elements of pattern theory*. Baltimore: Johns Hopkins University Press.

Horn, B. K. P. (1986). *Robot Vision*. Cambridge MA: MIT Press.

Kersten, D. (1997). Inverse 3D Graphics: A Metaphor for Visual Perception. *Behavior Research Methods, Instruments, & Computers, 29*(1), 37-46.

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.

Mumford, D. (1996). Pattern theory: A unifying perspective. In D. C. Knill & R. W. (Eds.), *Perception as Bayesian Inference* (pp. Chapter 2). Cambridge: Cambridge University Press.

Mumford, David (2002) *Pattern Theory: the Mathematics of Perception*. APPTS Report #02-10, August 2002. [pdf]