Computational Vision
U. Minn. Psy 5036
Daniel Kersten
Lecture 10: Image processing

Initialize

■ **Read in Add-in packages:**

In[14]:=
```
Off[General::"spell1"];
  << "BarCharts`"; << "Histograms`"; << "PieCharts`"
SetOptions[ArrayPlot, ColorFunction → "GrayTones", DataReversed → True,
  Frame → False, AspectRatio → Automatic, Mesh → False,
  PixelConstrained → True, ImageSize → Small];
```

■ **The input 64x64 image: face**

In[18]:=
```
width = Dimensions[face][[1]]; size = width;
hsize = width/2; hfwidth = hsize; height = Dimensions[face][[2]]; face;
gface = ArrayPlot[face]
```

Out[19]=

Outline

Last time

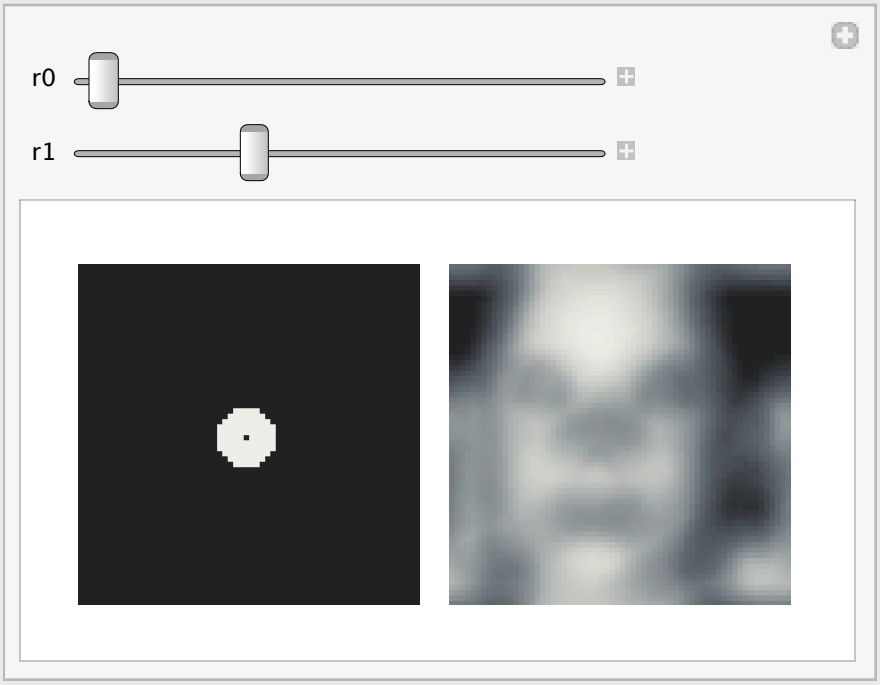
Single - channel spatial filtering

A fixed choice of lower and upper bound spatial frequencies, r0 and r1, respectively.

In[20]:=
```
filter[r0_, r1_] :=
  Table[If[((x - hsize) ^ 2 + (y - hsize) ^ 2 > r0) &&
     (x - hsize) ^ 2 + (y - hsize) ^ 2 < r1, 1, 0] , {x, 1, size}, {y, 1, size}];
```

In[21]:=
```
Manipulate[
 tt = Chop[InverseFourier[Fourier[filter[r0, r1]] Fourier[face]]];
 tt = RotateLeft[tt, {hsize, hsize}];
 GraphicsRow[{ArrayPlot[filter[r0, r1]], ArrayPlot[tt]}],
 {{r0, 0}, 0, r1}, {{r1, hsize}, 0, 100}]
```

Out[21]=



Multiple spatial frequency channels

Global vs. local:

Sinusoidal basis functions are global filters

Psychophysical experiments.

->Multi-resolution, but local filters

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

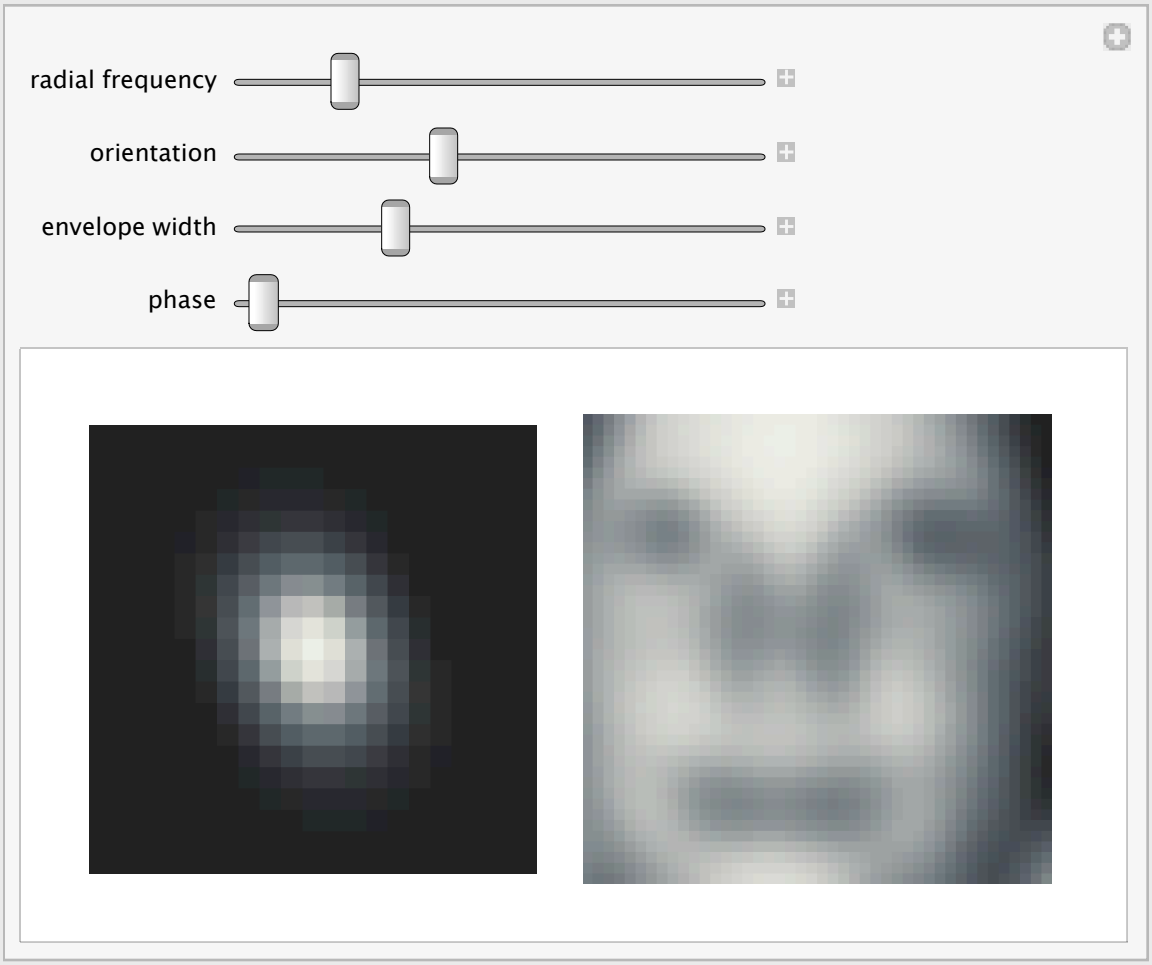
Multiresolution analysis with local filters

Instead of sinusoidal basis functions, we can filter with localized "gabor function" filters:

In[27]:=
```
Grating[x_,y_,fx_,fy_,phase_] := Cos[(2.0 Pi (fx x + fy y) + phase)];
GratingPatch[x_,y_,fx_,fy_,sig_,phase_] := Exp[-((x)^2 + (y)^2)/(2*sig^2)]*
kern[fx_, fy_, sig_,phase_] :=
  Table[GratingPatch[x, y, fx, fy, sig,phase], {x, -1, 1, .1}, {y, -1, 1, .
```

In[31]:=
```
Manipulate[
 GraphicsRow[{
   ArrayPlot[kern[fr * Cos[theta], fr * Sin[theta], sig, phase]],
   ArrayPlot[ListConvolve[kern[fr * Cos[theta], fr * Sin[theta],
      sig, phase], face]]}], {{fr, 1, "radial frequency"}, .1, 2},
 {{theta, .4, "orientation"}, 0, Pi},
 {{sig, .4, "envelope width"}, .001, 1}, {{phase, 0, "phase"}, .0, Pi / 2}]
```

Out[31]=



Self-similarity

But another restriction is that the filters could all be the same shape, but just scaled versions of each other.

The self - similar idea is important to vision because of the need for some kind of scale - invariance. Further, the self - similar aspect of these neural models bore a close resemblance to the emerging mathematical field of wavelet analysis. The emphases are different-- over - completeness may be important and vision does the projections in parallel (the serial algorithmic component of wavelet computation is integral to the mathematical interest).

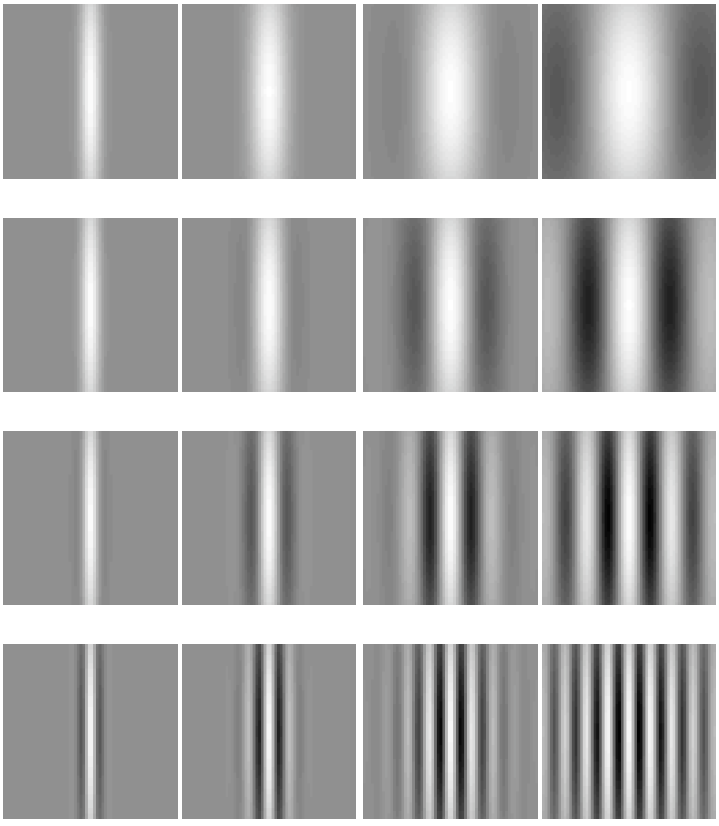
Human efficiency for detecting gabor patches

Burgess, Wagner, Jennings and Barlow (1981) combined the SKE observer and spatial frequency analysis of human vision to find out how efficiently humans detected patterns. They showed in a 1981 Science article that narrowly windowed sinusoids were detected with high efficiency (>70%) when added to static visual noise. Further, these targets were detected more efficiently than disks of light.

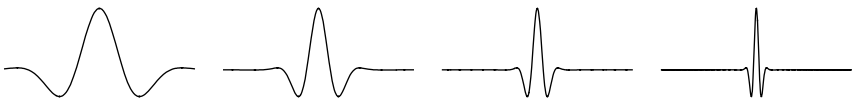
You basically have all the tools to replicate the experiment of Burgess et al. You can compute d' for the ideal observer for signal-known-exactly patterns. And you can generate Gaussian-windowed sinusoids and add them to gaussian white noise. If you measure the percent correct, and convert that to d' for the human observer, you can calculate the absolute efficiency for human detection--and contribute to answer the question of what the eye sees best.

Watson, Barlow & Robson (1983) found that that a 7 c/deg grating drifting at 4 Hz, (with a narrow gaussian envelope in space and time) was detected more efficiently than other patterns. Further, the quantum efficiency was very low (<0.05%).

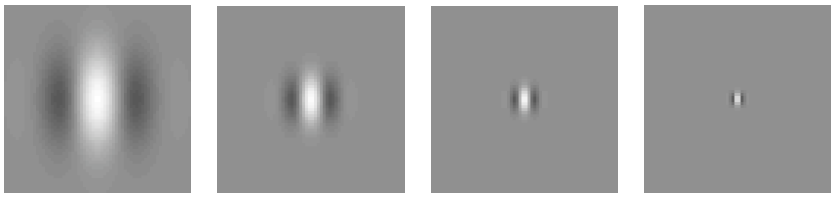
Kersten (1984) measured efficiency for 1-d gratings (i.e. vertical) in temporal (1-d spatial) visual noise for various spatial frequencies and widths.



Peak efficiency was found for patterns of about the same shape, regardless of spatial frequency. The cross-sectional profiles for high efficiency patterns corresponded to the diagonals in the above graph and looked like:



Psychophysical measurements across spatial scale haven't been made systematically yet for various vertical sizes. One prediction would be that images of the following type would be efficiently detected in noise:

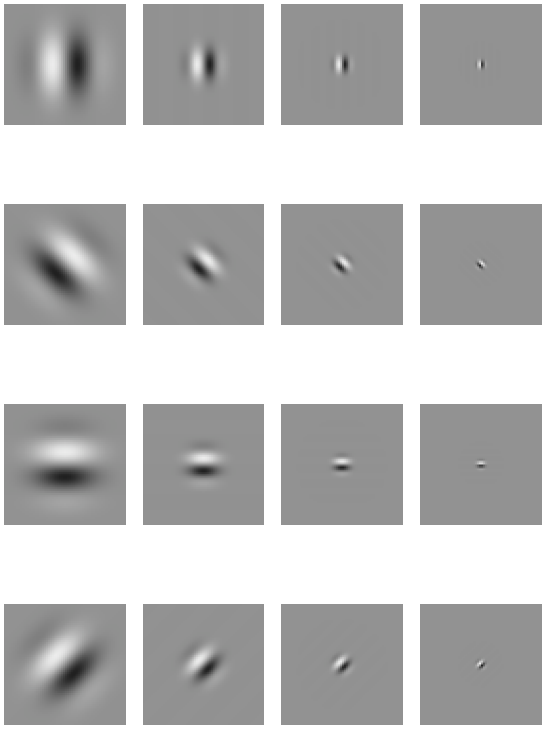


When the filters have the same shape except for a change of scale ($x \to \alpha x$), they are called *self-similar*.

■ Bottom-line: image coding in terms of scale and orientation:

A model for human spatial image representation

At each spatial location, project the image onto a collection of basis vectors (i.e. compute the dot product) that span a range of *spatial scales* and *orientations*:



In general, these neural models of basis functions may be over-complete, and non-orthogonal. And there may be a range of phases. Above we show only the "sine-phase" or "edge-detectors" of Hubel and Wiesel.

Neural image? Or neural image representation?

We can view the response activities of a family of receptive fields of neurons as representing a filtered neural image of the input image. Although useful, this view can be misleading when we start to think about function, for "who is looking at the image"?

Alternatively, thinking in terms of basis functions gives us another perspective. We can view the response activities of a family of receptive fields as a representation of the input image. If linear, an activity is the result of a projection of an image on to a basis function (receptive field weights). Given such a representation we can begin to ask questions like:

1. Is the neural basis set complete? Can any image be represented?

2. Are image representations unique? Is any information lost? I.e. we do the inverse transformation, can the original input be reconstructed?

Or are there "equivalent classes" of images--i.e. ones that all produce the same neural representation? Consider for example, a single-channel model with lateral inhibitory center-surround spatial filters--what is an equivalence class?

3. Maybe the neural basis set is "over-complete"?

4. Are the neural basis functions orthogonal? Are they normal?

What is a multiresolution scale/orientation representation good for?

What is the computational significance of a wavelet-like decomposition?

Efficient coding?

-> savings in neurons, or metabolic requirements?

-> representations for efficient learning?

-> analysis of natural image statistics

Analysis of what vision needs to recognize objects, etc..

-> Edge detection?

-> Edge detection at different spatial scales. Combining over spatial scale

Today

■ Upcoming dates:

Mid-term: Oct. 20th. Study-guide available by Wednesday this week.

3rd Assignment due Oct. 29th. Will involve variations on exercises in this and the next two notebooks.

Final project outlines due: Nov. 12th.

Final projects

- **Image manipulations**

Final projects

Format

Should be written like a scientific paper.

Might require most of the code to be put in appendices.

Can use modules you find elsewhere, but preserve copyrights, and reference

Will post final notebooks on the class web site.

Your "audience" will be your class peers.

Possible types of projects

Programming perceptual demonstrations

- **Motion illusions**

  e.g. stereograms, autostereograms, lightness illusions

  with interactive parameter variation

  http://viperlib.york.ac.uk/

- **Geometrical morphs to illustrate adaptation, perceptual hysteresis**
- **Instructional demonstrations**

E. g. Cortical magnification: use function interpolation to illustrate how the retintopic map maps the visual field onto primary visual cortex.

Programming visual psychophysics (quantitative measurements)

- **What does the eye see best?**
- **Data analysis/report of data collected elsewhere (by you)**

OK to complement other projects, but need to clarify how the work load is divided up.

Classification images

See: http://www.journalofvision.org/2/1/introduction.html

See: Olman and Kersten (2004)

Importance of the phase spectrum in visual recognition

See, Glass patterns, Barlow and Olshausen

Visual adaptation

See for example, Webster, M. A., Georgeson, M. A., & Webster, S. M. (2002). Neural adjustments to image blur. *Nat Neurosci, 5*(9), 839-840.

Computational models

- **Machine vision: but should have discussion/comparison of relevance to human vision.**
- **Orthogonal wavelet decomposition in Mathematica**

See: http://www.cns.nyu.edu/~eero/software.html for Matlab versions

- **Neural network models**

  e.g. adaptive receptive field development, visual attention,...

- **Models of human/biological vision**
- **Bayesian edge detection**
- **Geometrical morphs to register images to compute shape means, and other statistics**
- **Statistical analyses of images**

  Apply signal detection theory to analyzing how well an edge detector performs relative to ground truth.

  e.g. Bayesian edge detector, correlational analyses, ...

Image processing: Simple point manipulations

Point operations

- **Various contrast definitions**

(Imax - Imin)/(Imax + Imin): Called "Michelsen contrast". Particularly appropriate for gratings, or stimuli with primary peak and trough.

$\Delta I/I_{background}$: Used in psychophysics of small points/disks against a background.

$\Delta I/I_{mean}$: Used in psychophysics for simple stimuli. Gives same number as Michelsen for gratings.

$c(x,y) = (I(x, y) - I_{mean})/I_{mean}$: contrast at a point (x,y). Could be as function of time too, c(x,y,t).

$\sum_{x,y} \sqrt{(I(x, y) - I_{mean})^2} / I_{mean}$: r.m.s. contrast, a good summary measure for complex image. "contrast power", $\sum_{x,y} c^2(x,y)$, is the square of r.m.s. contrast. "Contrast energy" is *power x area x duration*. In psychophysics, area is measured in degrees of visual angle.

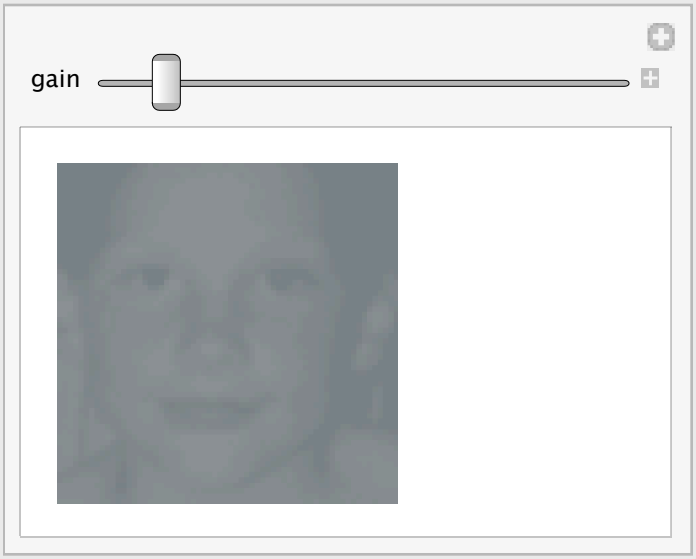
One needs to decide the region over which to calculate the mean. The default is the whole image.

- **Contrast manipulations**

Adjusting contrast (gain=1 leaves image unchanged, gain=0 reduces it to a uniform field):

In[32]:=
```
μ = Mean[Flatten[face]]; gain = 0.045;
Manipulate[
ArrayPlot[gain (face - μ) + μ, Mesh → False, Frame → False,
  PlotRange → {Min[face], Max[face]}],
 {{gain, 0.045}, 0, 1}]
```

Out[33]=



■ Psychophysics and contrast

When measuring human sensitivity, it is important to carefully measure and calibrate the image stimuli. Because standard computer displays can at best resolve 256 graylevels, it is useful to convert the stimuli into a range going from 0 to 255.

Scale so values are represented as graylevels between 0 and 255:

In[34]:=
```
α = 255 / (Max[face] - Min[face]);
β = -α Min[face];
```

In[36]:=
```
face256 = α face + β;
```

Exercise: Normalize face so that it has a mean level of zero, and an r.m.s. contrast of 1. Use ListPlot and Flatten[] to show a scatter plot of the values before and after the scaling.

Exercise: Given that human contrast sensitivity for a sinewave grating can be as high as 500, could you get a good measure of it using a typical computer graphics screen?

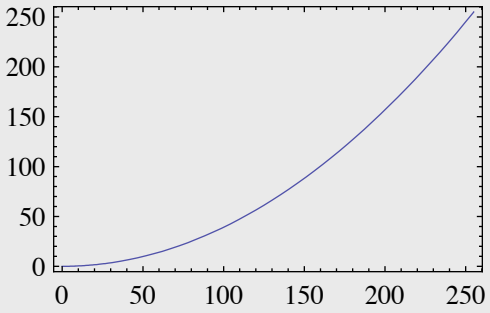
Exercise: Produce a negative image by reversing the contrast

■ Gamma correction

Computer operating systems allow one to adjust for various non-linearities between displays. A typical CRT has a non-linear relationship between measured screen intensity and the voltage supplied. A fairly standard way of summarizing the non-linear relationship is in terms of "gamma": *intensity = a x voltage^gamma*. Let's assume that we are using intensity units that range from 0 to 255 and voltage units also going from 0 to 255. intensity = 255^(1-gamma) x voltage^gamma:

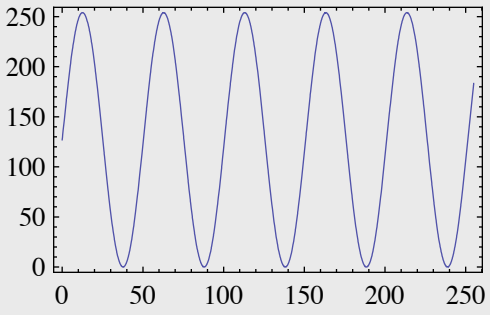
In[41]:=
```
output[input_, gamma_] := 255^(1-gamma) input^gamma;
Plot[output[x, 2], {x, 0, 255}, Frame → True, ImageSize → Small]
```

Out[42]=



In[43]:=
```
(output2[input_] := 127. Sin[input/8] + 127;);
Plot[output2[x], {x, 0, 255}, Frame → True, ImageSize → Small]
```

Out[44]=



You can do a many-to-one mapping:

```
ArrayPlot[output2[face256], PlotRange → {0, 255}]
```

Does the fact that you can still see a recognizable form in the above picture tell you anything about human vision?

The computer's display card has a look-up-table (LUT) that can be loaded with the inverse gamma function to linearize the display.

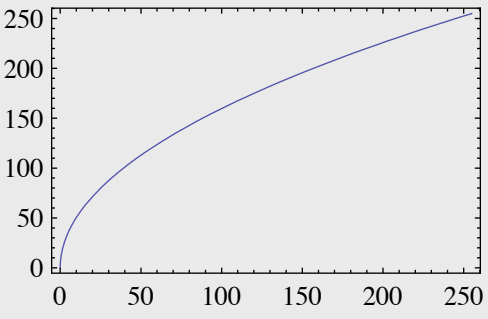
```
Solve[output == 255^(1 - gamma) input^gamma, input]
```

Solve::ifun : Inverse functions are being used by Solve, so some
solutions may not be found; use Reduce for complete solution information. ≫

$$\left\{\left\{\text{input} \to \left(255^{\text{gamma}-1}\ \text{output}\right)^{\frac{1}{\text{gamma}}}\right\}\right\}$$

In[50]:=
```
inverse[output_, gamma_] := (255^(-1+gamma) output)^(1/gamma);
Plot[inverse[x, 2], {x, 0, 255}, Frame → True, ImageSize → Small]
```

Out[51]=

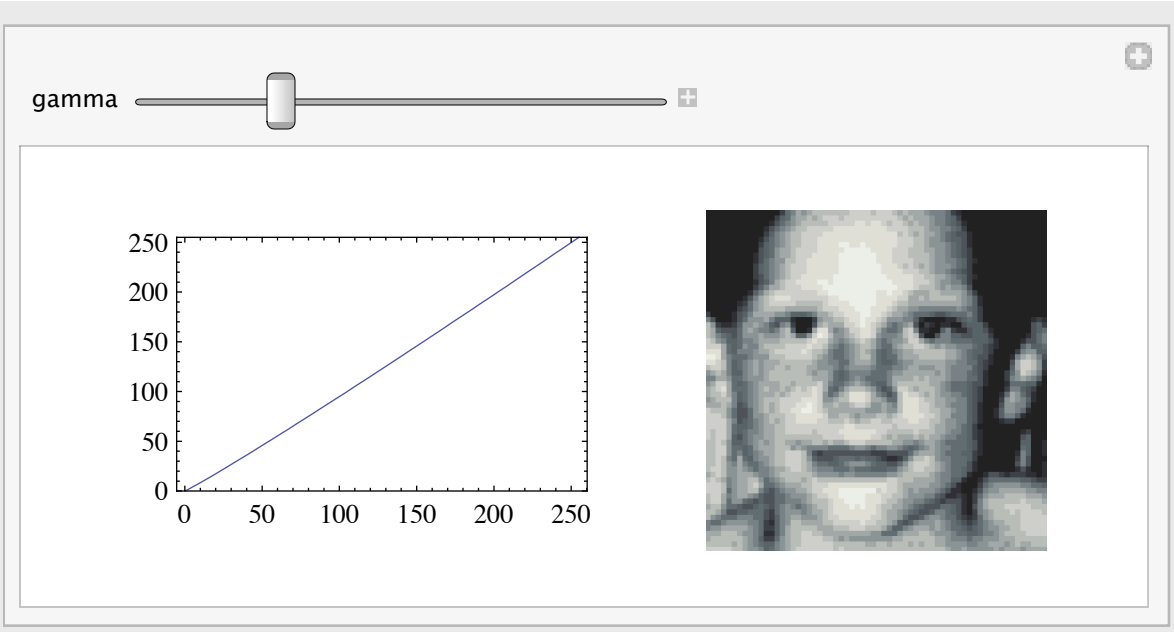


■ Using gamma to do point operations

You can also use the gamma transformation to do non-linear point operations on an image:

In[54]:=
```
Manipulate[
GraphicsRow[
  {Plot[output[x, gamma], {x, 0, 255}, Frame → True, ImageSize → Small,
    PlotRange → {0, 255}], ArrayPlot[output[face256, gamma],
    PlotRange → {0, 255}]}],
 {gamma, 0.1, 4}]
```

Out[54]=



■ Sigmoidal contrast manipulation

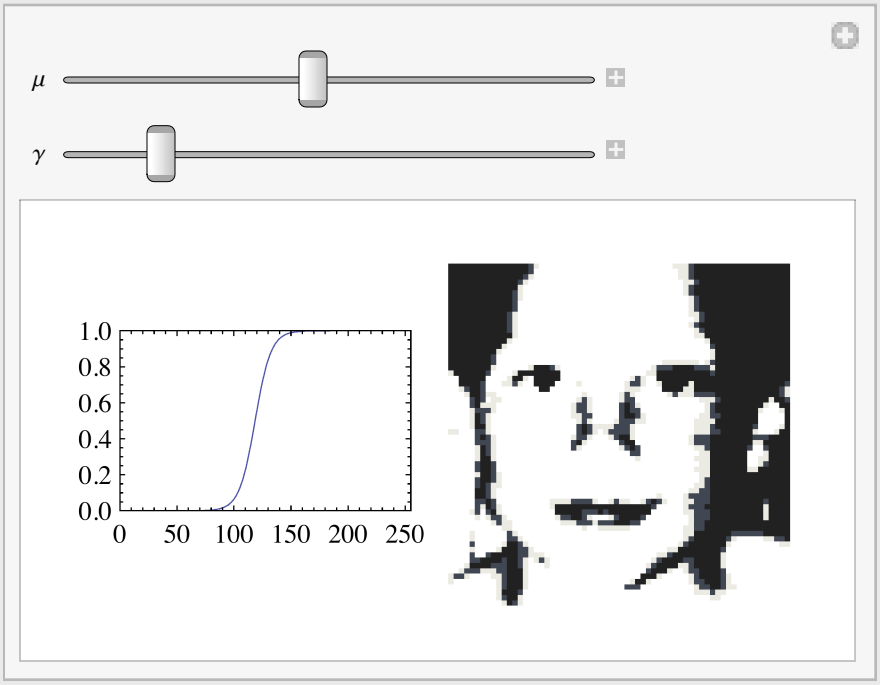
Here is a gain function (called the "logistic function") that manipulates contrast smoothly--a "soft" threshold:

In[55]:=
```
squash[x_, μ_, γ_] := N[1 / (1 + e^(-γ (x-μ)))];

Plot[squash[x, 128, 1], {x, 0, 255}, Frame → True, ImageSize → Small]
```

In[60]:=
```
gain = 0.045`; μ0 = Mean[Flatten[face256]];
Manipulate[
 GraphicsRow[
  {Plot[squash[x, μ, γ], {x, 0, 255}, Frame → True,
    PlotRange → {{0, 255}, {0, 1}}],
   ArrayPlot[squash[(face256 - μ0) + μ0, μ, γ],
    PlotRange → {Min[face], Max[face]}]}], {{μ, 128}, 0, 255},
 {{γ, .5}, 0, 1}]
```

Out[61]=



As γ grows, the sigmoid approaches a hard-threshold. Images that are forced to have only two values are called "Mooney images".

We can make Mooney images more directly using a function that takes an image and sets pixels bigger than τ to 255, and if less than (or equal to) τ, to 0:

```
Mooney[image_, τ_] := Map[If[# > τ, 255, 0] &, image, {2}];
```

(See Moore and Engel, 2001, for an application to studying brain responses to objects).

Exercise: Write a function that quantizes an image to a set of gray levels specified by a set of thresholds: $\tau_1, \tau_2, \tau_3, ..., \tau_{N-1}$. Set N=3. (Try using Which[]).

Simple statistics

First-order, i.e. don't take into account relations between pixels

■ Mean, variance, r.m.s. contrast

In[62]:=
```
μ = Mean[Flatten[face]];
σ = Sqrt[Variance[Flatten[face]]];
```

r.m.s. contrast can be calculated as:

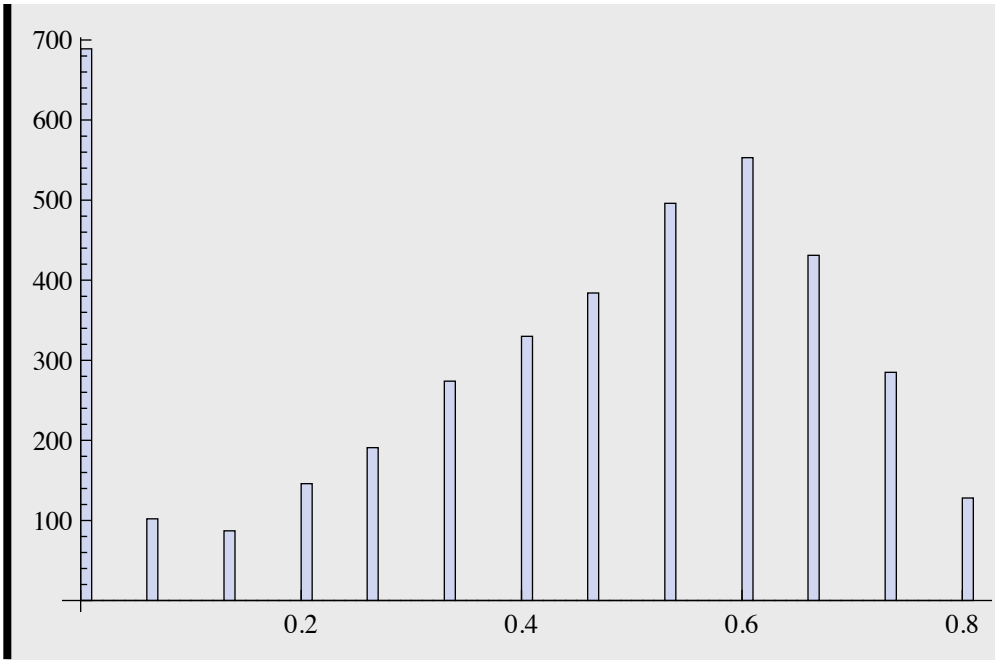
In[64]:=
```
Sqrt[Variance[Flatten[face]]] / Mean[Flatten[face]]
```

Out[64]=
```
0.600059
```

■ Histograms

In[65]:=
```
Histogram[Flatten[face]]
```

Out[65]=



You can tell that the image is quantized at a coarse level (less than 4 bits).

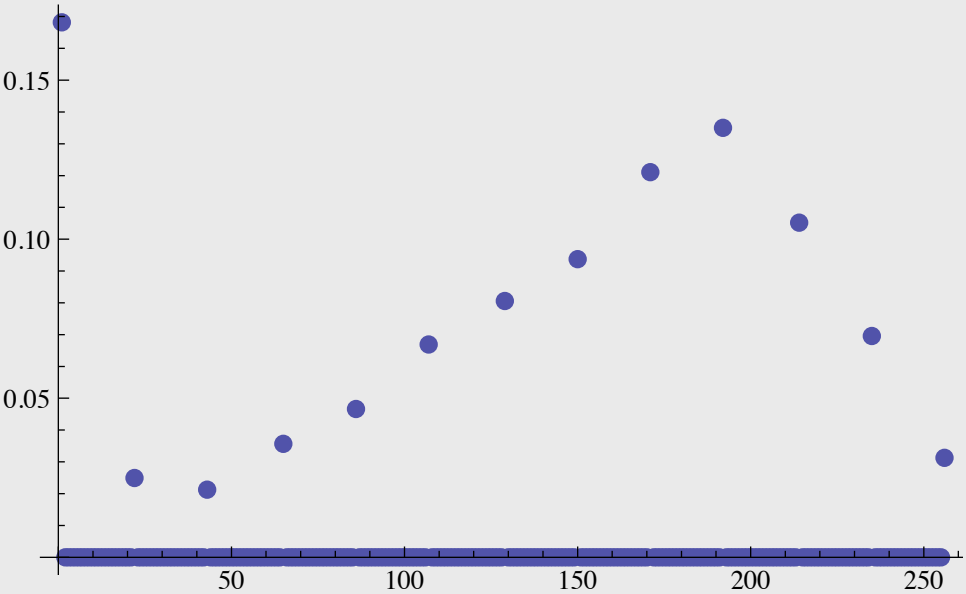
Alternatively, you could calculate the histogram with built-in functions. To do the pattern match below, the floating point numbers are first converted to integers using **Round[]**:

In[69]:=
```
domain = Range[0, 255];
Freq = Map[Count[Round[Flatten[face256]], #] &, domain];
```

If we normalize the histogram so that the sum is one, then we have a probability:

In[71]:=
```
ListPlot[Freq / Plus @@ Freq, PlotStyle → PointSize[0.02]]
```

Out[71]=



Getting regions of images

```
ArrayPlot[Take[face256, {1, 32}, {1, 64}]]
```

Or with *Mathematica* 6.0, you can use:

```
ArrayPlot[face256[[1 ;; 32, 1 ;; 64]]]
```

■ Getting coordinates

```
ArrayPlot[face256, PixelConstrained → {1, 1}]
```

Double-click on the image above to bring up the 2D Drawing tools. Now use the Get Coordinates tool to select the {x0,y0}, and {x1,y1} as the corners of the rectangular patch that you want. Do Save, and then do Paste in a cell below. Here are coordinates for diagonal points for the eyes.

```
{{9, 36}, {53, 51}}
Reverse[Transpose[%]]
ArrayPlot[Take[face256, %[[1]], %[[2]]]]
```

$$\begin{pmatrix} 9 & 36 \\ 53 & 51 \end{pmatrix}$$

$$\begin{pmatrix} 36 & 51 \\ 9 & 53 \end{pmatrix}$$

```
ArrayPlot[face256[[36 ;; 51, 9 ;; 53]]]
```

Geometric image manipulations using function interpolation

Compare the plots with InterpolationOrder → 0 and InterpolationOrder → 1.

```
faceFunction = ListInterpolation[Transpose[face], {{-1, 1}, {-1, 1}},
   InterpolationOrder → 0];
DensityPlot[faceFunction[x, y], {x, -1, 1}, {y, -1, 1}, PlotPoints → 256,
 Mesh → False, AspectRatio → Automatic, Frame → None,
 ColorFunction → "GrayTones"]
```

Morphing

```
faceFunction = ListInterpolation[Transpose[face], {{-1, 1}, {-1, 1}}];
```

```
DensityPlot[faceFunction[Sign[x] x^2, Sign[y] y^2], {x, -1, 1},
 {y, -1, 1}, PlotPoints → 100, Mesh → False, AspectRatio → Automatic,
 Frame → None, ColorFunction → "GrayTones"]
```

More filtering: Calculating the spatial gradient of an image using function interpolation

The gradient of an image intensity function f, ∇f, has a maximum value in the direction of greatest change.

$$|\nabla f| = \left|\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)\right| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2, \left(\frac{\partial f}{\partial y}\right)^2} \tag{1}$$

Let filterface =f, where we've blurred out face a little to reduce quantization artifacts:

In[99]:=
```
kernel = {{1, 1, 1}, {1, 1, 1}, {1, 1, 1}};
filterface = ListConvolve[kernel, face];
```

In[101]:=
```
faceFunction = ListInterpolation[Transpose[filterface],
   {{-1, 1}, {-1, 1}}];
```

In[102]:=
```
nx[x_, y_] := Evaluate[D[faceFunction[x, y], x]];
ny[x_, y_] := Evaluate[D[faceFunction[x, y], y]];

ImageGradient[x_, y_] :=
  Evaluate[Sqrt[D[nx[x, y], x]^2 + D[ny[x, y], y]^2] ];
```

Plot the rate of change in the x-direction:

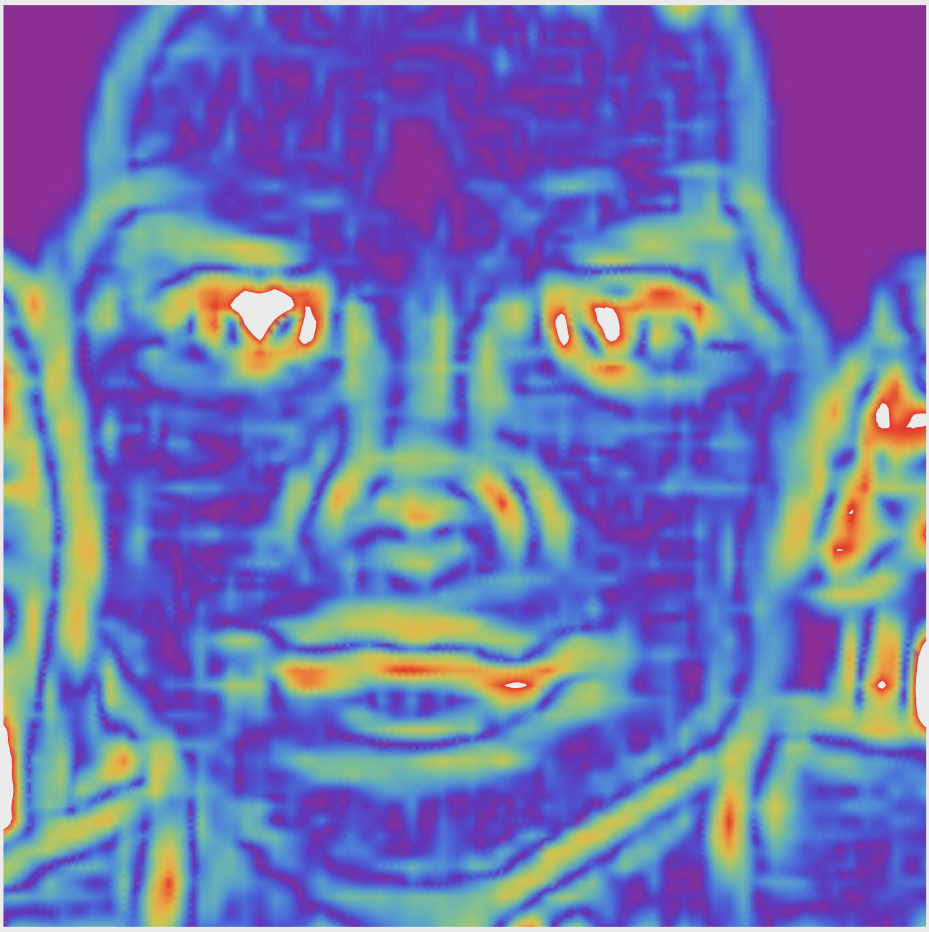
In[105]:=
```
temp = Table[nx[x, y], {x, -1, 1, .005}, {y, -1, 1, .005}];
ArrayPlot[Transpose[temp]]
```

Out[106]=

Plot the magnitude of the gradient to highlight regions of the image where contrast is changing the most rapidly:

In[107]:=
```
DensityPlot[ImageGradient[x, y], {x, -1, 1}, {y, -1, 1},
 PlotPoints → width, Mesh → False, Frame → False, ColorFunction → "Rainbow"]
```

Out[107]=



Manipulating color images

```
image = ExampleData[{"TestImage", "Mandrill"}]
```

```
RGBvalues = image[[1, 1]];
Dimensions[RGBvalues]
```

```
reds = Map[#[[1]] &, N[RGBvalues], {2}];
greens = Map[#[[2]] &, N[RGBvalues], {2}];
blues = Map[#[[3]] &, N[RGBvalues], {2}];
```

```
GraphicsRow[{ArrayPlot[reds], ArrayPlot[greens], ArrayPlot[blues]}]
```

■ **A weighted average of RGB values to produce a luminance image:**

```
grayscale = Map[(0.3 #[[1]] + 0.59 #[[2]] + 0.11 #[[3]])/255 &, N[RGBvalues], {2}];
ArrayPlot[grayscale]
```

Note the weights above are arbitrary, and the chosen values will depend on the color calibration.

■ **Putting the R, G, B images back together:**

```
r = reds;
g = greens;
b = blues;
temp2 = Partition[Transpose[{Flatten[r], Flatten[g], Flatten[b]}],
   Dimensions[r][[2]]];
```

```
Graphics[Raster[temp2 / 255., ColorFunction → RGBColor]]
```

Next time

Efficient coding

Science writing

References

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

Barlow, H. B., & Olshausen, B. A. (2004). Convergent evidence for the visual analysis of optic flow through anisotropic attenuation of high spatial frequencies. *J Vis, 4*(6), 415-426.

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

Engel, S. A., Glover, G. H., & Wandell, B. A. (1997). Retinotopic organization in human visual cortex and the spatial precision of functional MRI. *Cereb Cortex, 7*(2), 181-192.

Gold, J. M., Murray, R. F., Bennett, P. J., & Sekuler, A. B. (2000). Deriving behavioural receptive fields for visually completed contours. *Curr Biol, 10*(11), 663-666.

Konishi, S. M., Yuille, A. L., Coughlan, J. M., & Zhu, S. C. (2003). Statistical edge detection: Learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25*(1), 57-74.

Olman, C. A., & Kersten, D. (2004). Classification objects, ideal observers & generative models. *Cognitive Science, 28*, 227-239.

Moore, C., & Engel, S. A. (2001). Neural response to perception of volume in the lateral occipital complex. *Neuron, 29*(1), 277-286.

Schwartz, E. L. (1980). A quantitative model of the functional architecture of human striate cortex with application to visual illusion and cortical texture analysis. *Biol Cybern, 37*(2), 63-76.

http://library.wolfram.com/howtos/images/#histograms