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

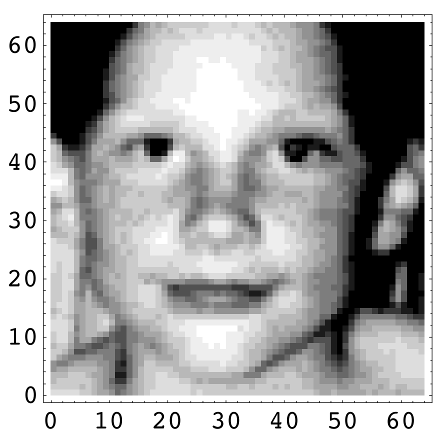
Initialize

■ Read in Statistical Add-in packages:

```
Off[General::spell1];
<< Statistics`DescriptiveStatistics`
<< Statistics`DataManipulation`
<< Graphics`Graphics`
```

■ The input 64x64 image: face

```
width = Dimensions[face][[1]]; hsize = width/2;
hfwidth=hsize;
height = Dimensions[face][[2]];
Short[face,12]; (* check out the first few lines*)
gface=ListDensityPlot[face,Mesh->False];
```



Outline

Last time

Single-channel spatial filtering

Multiple channel filters

Psychophysical experiments.

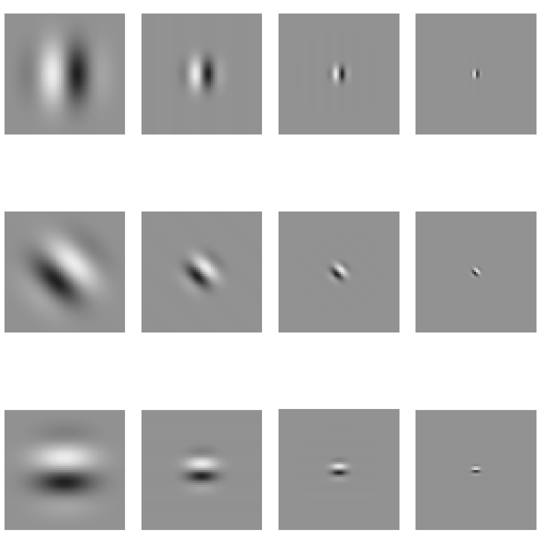
->Multi-resolution, and wavelet bases

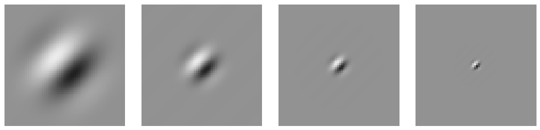
->A model of the spatial filtering properties of neurons in the primary visual cortex

■ Multiresolution cont'd

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

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).

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. A closely related question is: Is any information lost? I.e. we do the inverse transformation, can the original input be reconstructed?

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. 23th. Study-guide available by Monday next week.

3rd Assignment due Oct. 25th. Will involve variations on exercises in this and the next two notebooks.

Final project outlines due: Nov. 15th.

■ 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

Perceptual demonstrations

- **Motion illusions**

  e.g. stereograms, autostereograms, lightness illusions

  with interactive parameter variation

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

- **Instructional demonstrations**

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

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

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**

- **Statistical analyses of images**

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

Image processing: Simple point manipulations

Point operations

- **Contrast**

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

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

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

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

$\sum_{x,y} \sqrt{(I(x, y) - I_{\text{mean}})^2} / I_{\text{mean}}$: 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):

```
gain = 0.045; μ = Mean[Flatten[face]];
ListDensityPlot[gain (face - μ) + μ, Mesh → False, Frame → False,
  PlotRange → {Min[face], Max[face]}];
```

■ 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:

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

```
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.

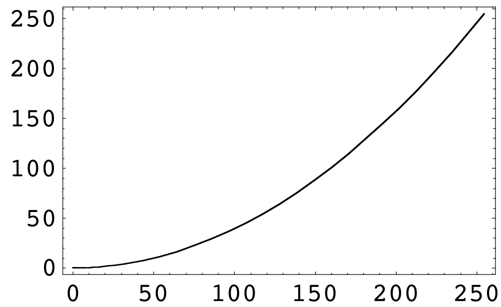
Exercise: Given that human contrast sensitivity for a sinewave grating can be as high as 500, could you get a good measure of it using a typical computer graphics screen?

Exercise: Produce a negative image by reversing the contrast

■ Gamma correction

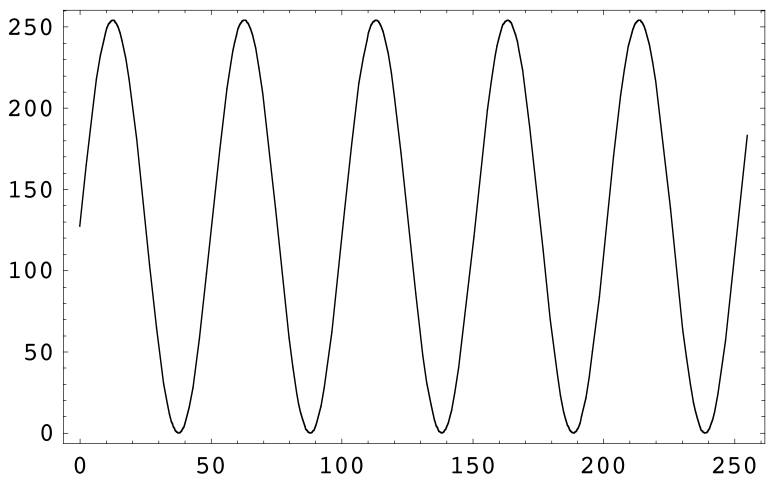
Computer operating systems allow one to adjust for various non-linearities between displays. A typical CRT has a non-linear relationship between measured screen intensity and the voltage supplied. A fairly standard way of summarizing the non-linear relationship is in terms of "gamma": *intensity = a 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:

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




```
output2[input_] := 127.0 * Sin[input / 8] + 127;;

Plot[output2[x], {x, 0, 255}, Frame → True];
```



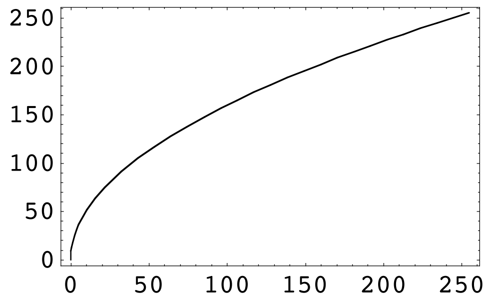
The computer's display card has a look-up-table (LUT) that can be loaded with the inverse gamma function to linearize the display.

```
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. More…

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

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



■ Using gamma to do point operations

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

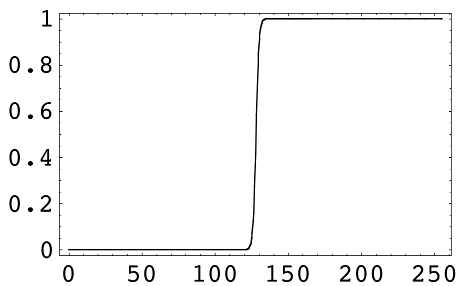
```
ListDensityPlot[face256, Mesh → False, Frame → False, PlotRange → {0, 255}];
```

```
ListDensityPlot[output[face256, 2.5], Mesh → False, Frame → False,
  PlotRange → {0, 255}];
```

■ Sigmoidal contrast manipulation

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

```
squash[x_,μ_,γ_] := N[1/(1 + Exp[-γ*(x-μ)])];
Plot[squash[x,128,1],{x,0,255},Frame→True];
```




```
gain = 0.045; μ = Mean[Flatten[face256]];
ListDensityPlot[squash[(face256 - μ) + μ, 128, 1], Mesh → False,
  Frame → False, PlotRange → {Min[face], Max[face]}];
```

■ Hard-thresholds: "Mooney faces"

Here is 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}];
```

A hard-thresold is used to produce "Mooney faces":

```
ListDensityPlot[Mooney[face256, 200], Mesh → False, Frame → False,
  PlotRange → {Min[face], Max[face]}];
```

Moore and Engel recently used this manipulation to study brain activity related to object perception.

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

Requires add-in package (above) to have been loaded.

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

■ Mean, variance, r.m.s. contrast

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

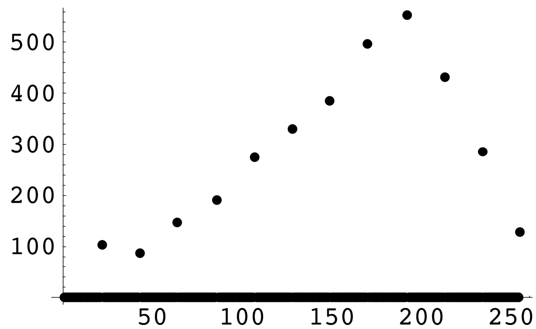
r.m.s. contrast can be calculated as:

```
Sqrt[Variance[Flatten[face]]] / Mean[Flatten[face]]
```

```
0.600059
```

■ Histograms

```
histoface = BinCounts[Flatten[face256], {0, 255}];
ListPlot[histoface, PlotStyle → PointSize[0.02]];
```



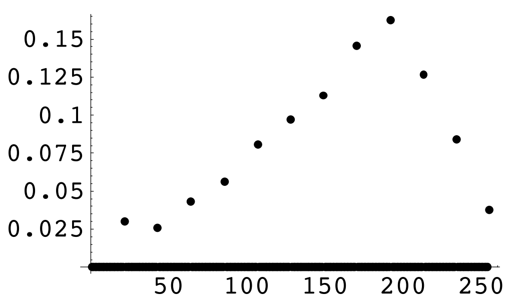
You can tell that the image is quantized at a coarse level (less than 4 bits).

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[]**:

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

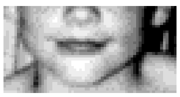
If we normalize the histogram so that the sum is one, then we have a probability:

```
ListPlot[histoface / Apply[Plus, histoface], PlotStyle → PointSize[0.02]];
```



Getting regions of images

```
ListDensityPlot[Take[face256, {1, 32}, {1, 64}], Mesh → False,
  Frame → False, PlotRange → {Min[face256], Max[face256]},
  AspectRatio → Automatic];
```



```
ListDensityPlot[Take[face256, {1, 64}, {1, 64}], Mesh → False,
  Frame → False, PlotRange → {Min[face256], Max[face256]},
  AspectRatio → Automatic];
```

By selecting the image above, and holding down the option key on the Mac (or control key in Windows), you can use the mouse click to select coordinates. 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 the coordinates for diagonal points on the left eye:

```
{{11.0143, 39.8432}, {24.1569, 46.2025}}
```

$$\begin{pmatrix} 11.0143 & 39.8432 \\ 24.1569 & 46.2025 \end{pmatrix}$$

```
Round[{{11.0143, 39.8432}, {24.1569, 46.2025}}]
```

$$\begin{pmatrix} 11 & 40 \\ 24 & 46 \end{pmatrix}$$

Origin is in the lower-left corner (the starting point in matrix is the upper right). So to select the eye, for example, if you first click on the lower left, then the lower right, you can re-arrange the coordinates so that you can copy and past the output of

```
Reverse[Transpose[Round[{{10.5904, 39.4193}, {25.0048, 47.4744}}]]]
```

$$\begin{pmatrix} 39 & 47 \\ 11 & 25 \end{pmatrix}$$

into the Take[] function.

Exercise: Try taking a sub-image using the Range[] command.

Exercise: Use ListDensityPlot and Take[] to plot a sub-picture

---

Geometric image manipulations using function interpolation

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 → 200, Mesh → False, AspectRatio → Automatic, Frame → None];
```

More filtering: Gradient edge detection using function interpolation

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

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

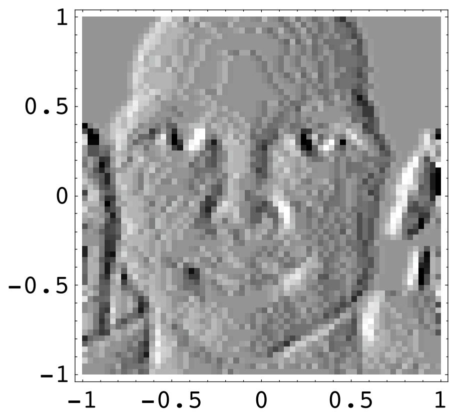
Let face =f.

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

```
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] ];
```

```
DensityPlot[nx[x, y], {x, -1, 1}, {y, -1, 1}, PlotPoints -> 64, Mesh -> False];
```




```
DensityPlot[ImageGradient[x, y], {x, -1, 1}, {y, -1, 1}, PlotPoints → width,
  Mesh → False, Frame → False];
```

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