Class business

- Any questions from last time?
- Second Midterm??
  - Grades 15%, 15%, 30% vs. 20% 40%
  - But would probably have to be Nov. 20 (Tuesday before Thanksgiving)
- Piazza?

Images, Histograms and Matlab Intro

Interpreted Matlab

- Arithmetic, loops, variables…
  - No defined types, garbage collection, …
  - Matrices
    - Dynamic space allocation, simple concatenation with [,;], matrix created as accessed.
      - Can be slow; faster if pre-allocated.
    - Lots of matrix operations
    - Matrix operations are very efficient, loops may not be.
- Help

Some Matlab expressions to try

```
>> 7 + 3
>> x = sqrt(17*pi);
>> x
>> A = randi(4,3,4);
>> B = randi(3,4,3);
>> A*B
>> C = randi(4,3,3)
>> inv(C)
>>C*inv(C)
>>[A,B']
>>[A;B']
>>[A,B]
>> D(3,4) = 7;
>> help
>> help images
>> help inv
```

Functions and M-Files

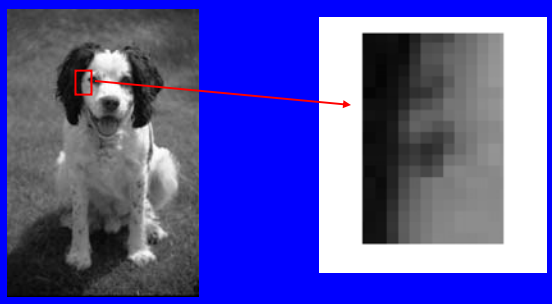
- Function: one visible with same name as file (sub-functions allowed but not visible outside file).
- Function definition
  - Values returned by just assigning them.
  - Call by value
- M-files
  - I never use these, but my ECE colleagues do.
- Debugging
  - Keyboard, debug on error, print w/ no semicolon

```
% Example of a simple function

function o = myNorm(a)
% One argument, a, is passed to function.  Called as: b = myNorm([1 2 3]);
% Value assigned to o will be returned.
o = sqrt(sum(a.^2));
keyboard
% Can insert this keyboard command for debugging.
```

Grayscale Images

- Matrix of scalars
- Usually 8 bit: 0 – 255
  - 0 is black, 255 is white



Images in Matlab

- *imread* and *imwrite* for images.
- Images are uint8, this can cause weird errors or bugs.
- Displaying images: figure, imshow.

```
>> I = imread('swanbw.jpg');  % swan.jpg is a file in the current directory that
contains a grayscale image.
>> figure; imshow(I);
>> size(I)
>> I(1:10, 1:10)
>> figure(1); imshow(I+ 128)
>> figure(1); imshow(I*2)
>> figure(1); imshow(I*.5)
>> figure(1); imshow(I>128)
>> J=I;
>> J(100,:) = 255;
>> figure(1); imshow(J)
```

Java & Grayscale

- Buffered Image
  - Extends image. Contains color model and raster.
  - import java.awt.image.*;
  - import java.io.*;
  - BufferedImage img = **null**;
  - img = ImageIO.*read*(file);
- Copy Image

```
private static BufferedImage copyBufferedImage (BufferedImage source) {
    BufferedImage target = new BufferedImage(source.getWidth(),
            source.getHeight(), source.getType());
    Graphics g = target.getGraphics();
    g.drawImage(source, 0, 0, null);
    return target;
}
```

Java & Pixels

- Pixels are signed bytes (-128 to 127)
- Read a pixel value:

```
byte pix[] = new byte[1];
int pix_int;
img.getRaster().getDataElements(0, 0, pix);
pix_int = ((int) pix[0]) & 0xff;
```

- Write a pixel value

```
if (newval > 127)
    newval = - (256 - newval);
pix[0] = (byte) newval;
img.getRaster().setDataElements(0, 0, pix);
```

More Java

```
private static BufferedImage
        convertToGrayscale(BufferedImage source) {
        BufferedImageOp op = new ColorConvertOp(
        ColorSpace.getInstance(ColorSpace.CS_GRAY),
    null);
     return op.filter(source, null);
}
```

Image Processing

- *Image Processing* means transforming images into new images
- Simplest image processing treats every pixel independently.
- I is input image, J is output image. I(x,y), J(x,y) are corresponding pixels.
- J(x,y) = f(I(x,y)).

Thresholding

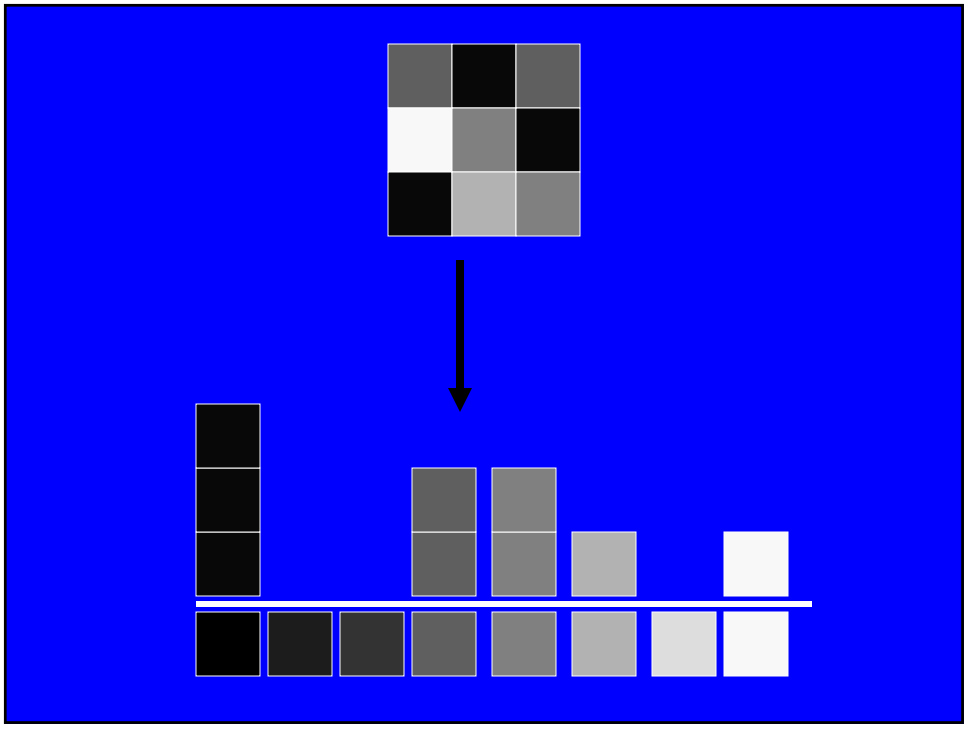
- One of the simplest operations we can perform on an image is *thresholding.* For example, if we take the swan image and threshold it with a threshold of T, we make all pixels >= T into 1, and all pixels < T into 0.

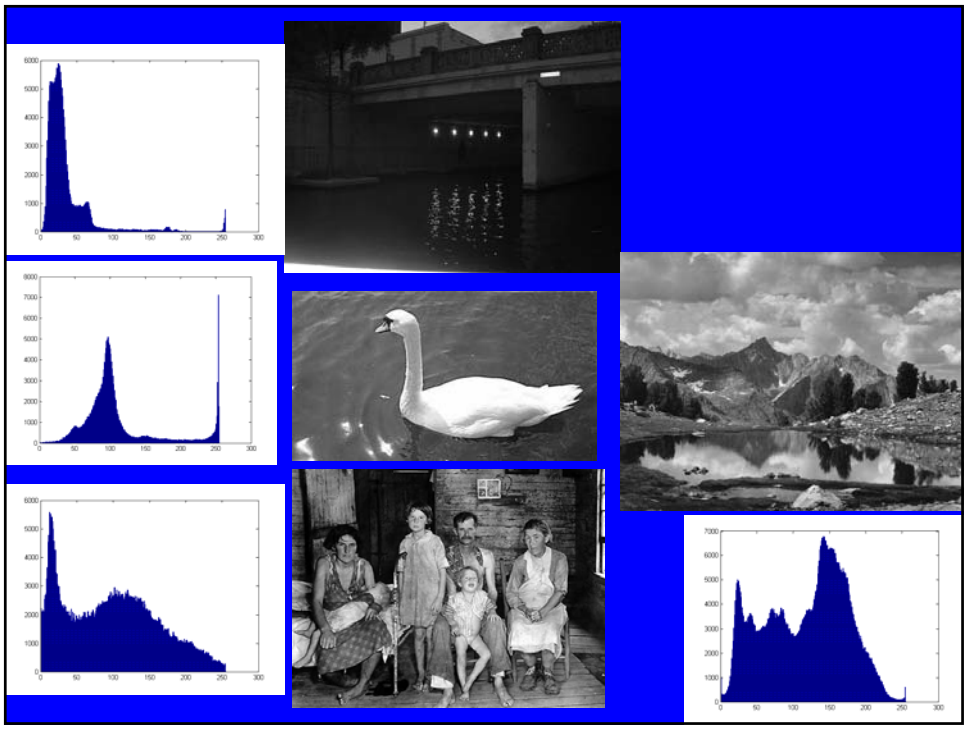
Swan Image

Swan Image
thresholded at T = 128

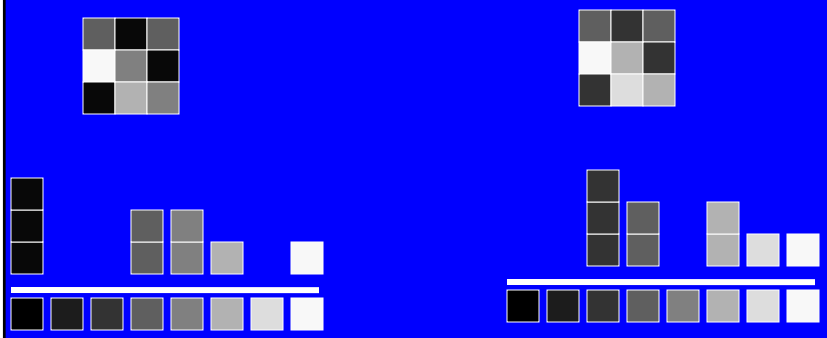
Grayscale Histogram

- Count intensities
- Normalize
- What determines Histogram?
  - Contrast, aperture, lighting levels, scene, …





Histogram Equalization (intuition)



Cumulative Distribution Function

- Let *h(i)* denote histogram
  - That is, *k=h(i)* means that *k/N* pixels in the image have intensity *i* if image has N pixels
  - Define: $$C(i) = \sum_{j=0}^{i} h(j)$$
  - *C* is the CDF (fraction of pixels with intensity <= *i*).

Histogram Equalization

Let's take an example of histogram equalization. We'll just consider 1D images. Suppose we have the image:
I = (4 2 1 2 3 3 8 3)
The histogram of this is:
h = (1 2 3 1 0 0 0 1)
The cumulative distribution function is:
C = (1 3 6 7 7 7 7 8)/8
We want to find a mapping of intensities, f, so that if a pixel in image I has intensity d, the corresponding pixel in image J has the intensity f(d). That is, J(x) = f(I(x)). And we want the cumulative distribution of J to be uniform, so if the cumulative distribution is D, it looks like:
D = (1 2 3 4 5 6 7 8)/8.
We can see by comparing C and D that we want f(1) = 1, f(2) = 3, f(3) = 6, f(4) = 7, f(5) = 7, f(6) = 7, f(8) = 8. This means that f(i) = C(i)*8. Applying this, we get:
J = (7 3 1 3 6 6 8 6)
The histogram of this, j, is:
j = (1 0 2 0 0 3 1 1) and the CDF is:
D = (1 1 3 3 3 6 7 8)/8. This isn't exactly uniform, but it's the best we can do without mapping two intensities that are identical in I to different values in J.

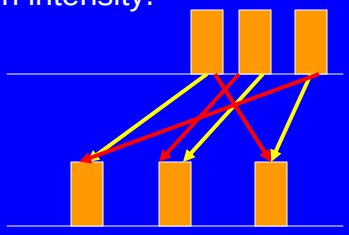
Histogram Equalization

- Make histogram as uniform as possible.
- Make CDF as linear as possible.
  - *h(i)* uniform -> *C(i) = i/K,* where *K* is number intensity values.
- *f(i)* transforms intensities.
  - *f* is a monotonic function
- For H.E.
  - *f(i) = C(i)*K.*
  - New Image J, *J(x,y) = f(I(x,y))*
- Let *D be* CDF of new image
  - *D(f(i)) = C(i).* Ie, #pixels in I <= i is same as #pixels in J <= f(i).
  - *-> D(C(i)*K) = C(i) -> D(j) = j/K,* if we let j = C(i)*K
  - *C(i)/K* may not give all integer values. Must round

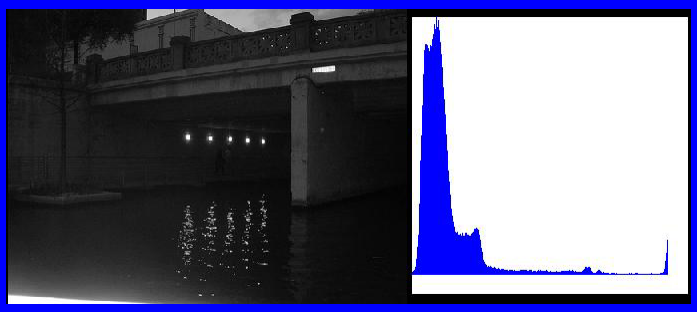
Properties of Histogram Equalization

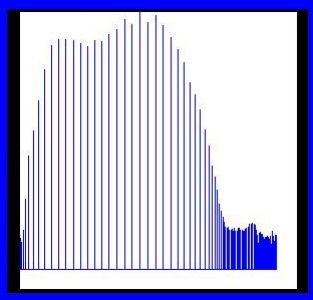
1. *f* generates an approximately uniform histogram.
2. *f* is a monotonic function.
    1. If one pixel is darker than another before equalization, it is afterwards.
    2. If pixels have same intensity before, they still do after
3. Of all functions that generate the new histogram, *f* does it with smallest changes in intensity.

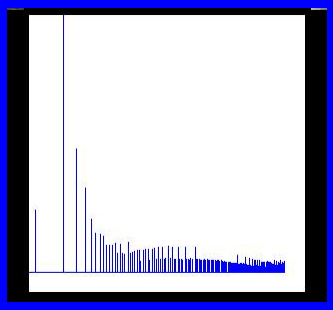
Properties 2 and 3 are equivalent



If the mapping is non-monotonic (red), paths cross, and can be shortened by switching their targets.







Histogram Specification

- Choose *f* so that new histogram matches a specific profile.
- For example, give one image histogram of another.
- Details left as exercise in PS1

Applications

- Make images look better
- Lighting Insensitivity
  - Removes additive and multiplicative changes.
  - In fact, removes all monotonic changes
    - These can be due to camera response.
    - Some evidence human vision is insensitive to these changes.
  - Does not adjust for changes in lighting position.
  - Image comparison, with lighting change

Invariance to monotonicity:

Suppose we have some histogram, h(i), which is not uniform. We can make it uniform with the monotonic transformation f (ie., h(f(i)) is uniform).

Now, suppose a second histogram, k(i), is related to h(i) by a monotonic transformation, g (ie., k(i) = h(g(i)).

Then, if we want to make k uniform, we can do it with the monotonic transformation $t=fg^{-1}$. This is because $k(t(i)) = k(fg^{-1}(i))=h(fg^{-1}g(i))=h(f(i))$ is uniform.

This means that if we transform h and k to have uniform histograms, they will become identical again, as we undo the transformation g.

This allow depends on two things. 1) We can combine two monotonic transformations, and we still get a monotonic transformation. 2) We can invert a monotonic transformation and get a monotonic transformation. (In fact, monotonic transformations form a *group*).

Histogram Comparison

- SSD

Let h and g be two histograms.

$$\|h - g\| = \sum_{i=1}^{N} \left(h(i) - g(i)\right)^2$$

- Cosine

$$\cos(h, g) = \frac{\langle h, g \rangle}{\|h\| \|g\|}$$

Example: Let I = [0 1 2 1 2 1 2 2] let J = [2 1 1 2 1 0 2 1], K = [0 1 1 0 2 2 1 2]
Histograms: h = (1 3 4), j = (1 4 3), k = (2, 3, 3)
SSD: I vs. J (1-2)^2 + (3-3)^2 + (4-3)^2 = 2 (We might normalize histograms before computing SSD).
I vs. K is also 2.
Cosine: I vs. J <(1,3,4), (2,3,3)> / ||(1,3,4)||||(2,3,3)|| = 23/sqrt(26)sqrt(22) = .9615. I vs. K = .9617
Chi-squared: I vs. J = ½ * [0/2 + 1/7 + 1/7] = 1/7.
I vs. K = ½ * [1/3 + 0 + 1/7] = 5/21 > 1/7.

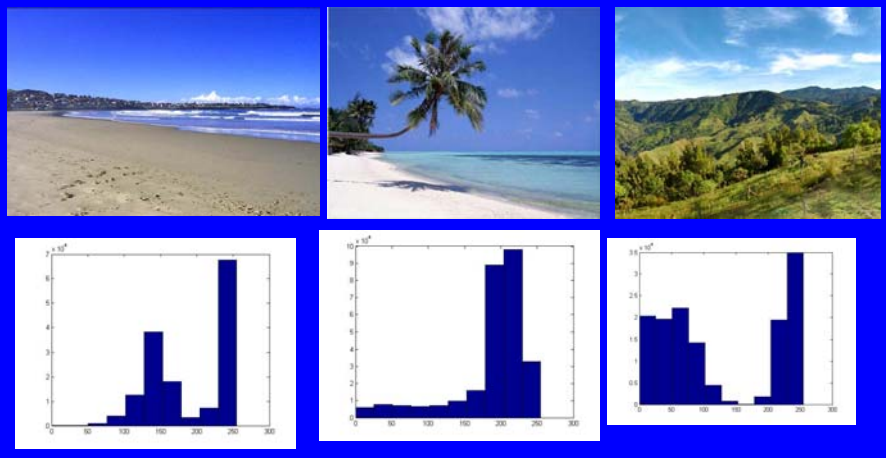
Histogram Comparison: as a Probability Distribution

- Chi-Squared

$$\chi^2(h_i, h_j) = \frac{1}{2}\sum_{m=1}^{K}\frac{[h_i(m) - h_j(m)]^2}{h_i(m) + h_j(m)}$$

- Smoothing probability distributions
  - Use bigger buckets.
  - Add a constant value (eg., 1) to every bucket.
  - Gaussian smoothing

Histogram Comparison: EMD



Histograms of blue color channel. Beaches and forests have distinctive histograms. But left and right images have histograms with minimal L2 norm.

More Java Code

```
JFrame f = new JFrame("Image");
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setPreferredSize(new Dimension (1200, 800));
JPanel mainpane = new JPanel();
mainpane.setBackground(Color.black);
mainpane.add(panel);

f.getContentPane().add(mainpane);
f.pack();
f.setVisible(true);
```

See code for creation of panes with content.

Application: Image Retrieval