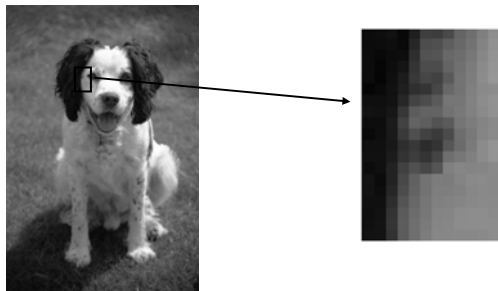


# Histograms

## Grayscale Images

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



# 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);
```

## Color Images

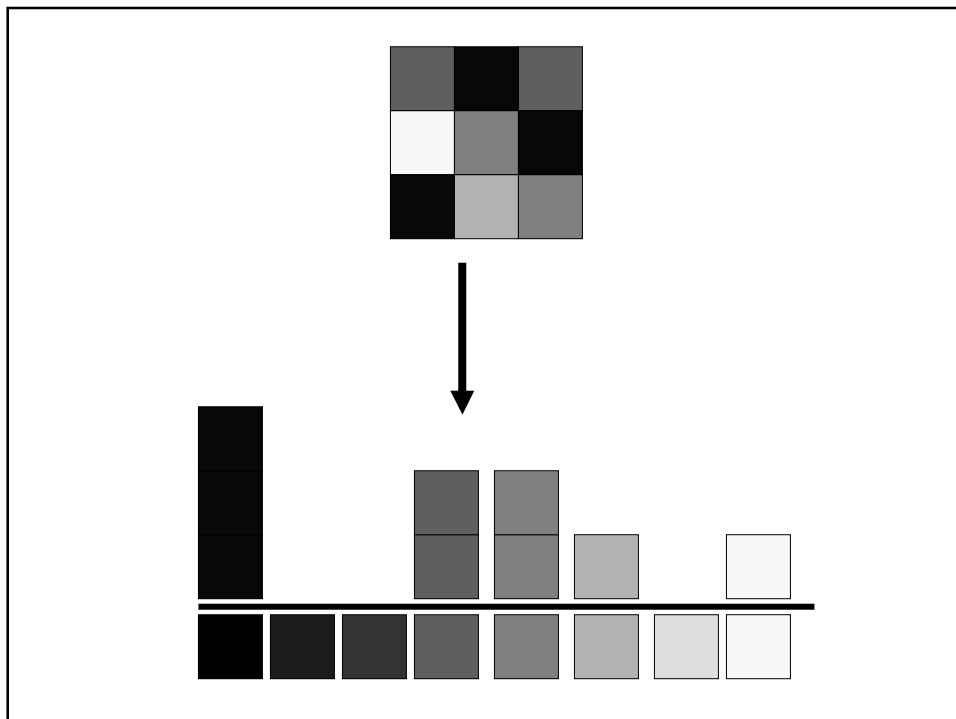
- RGB – 3 8-bit numbers
  - ~All colors can be composed of a mixture of Red, Green and Blue.
- We'll say more later

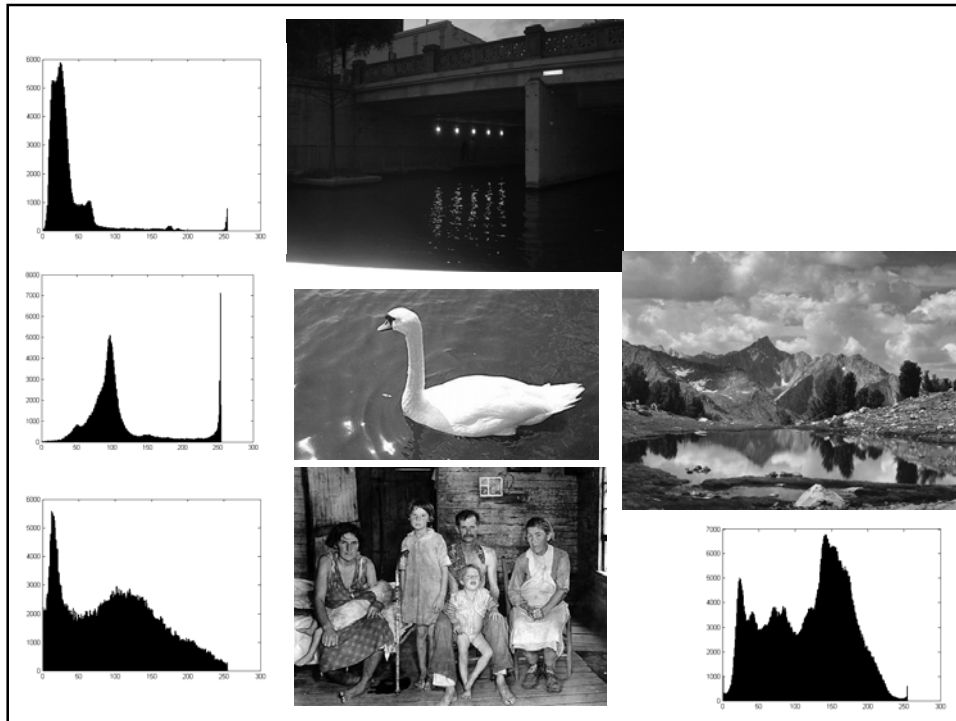
## More Java

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

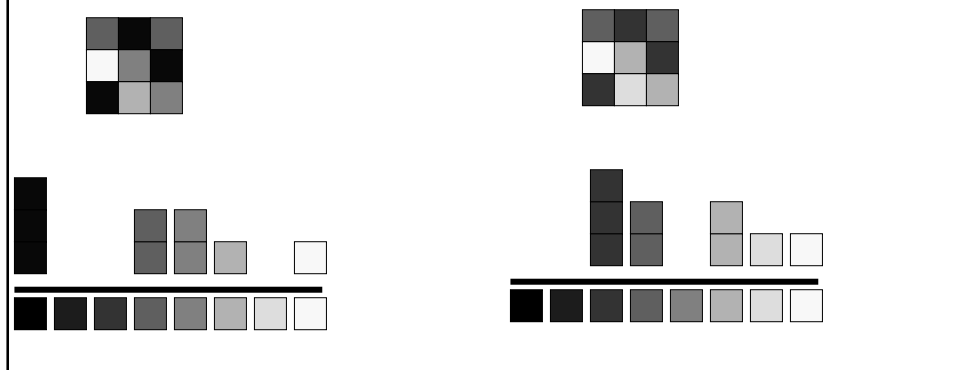
# Grayscale Histogram

- 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  $\leq i$ ).

## Histogram Equalization

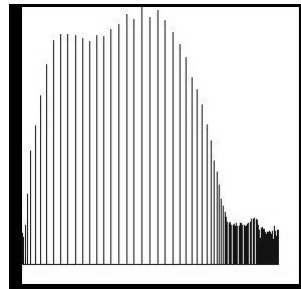
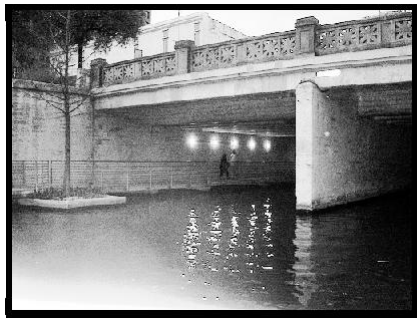
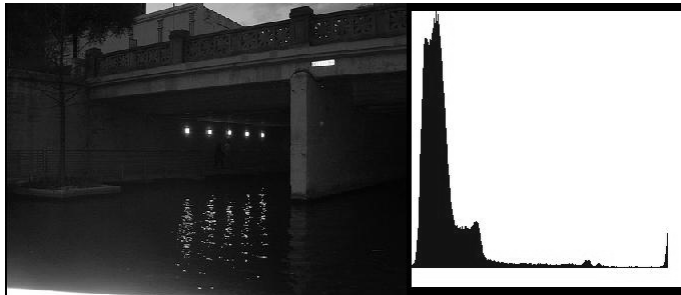
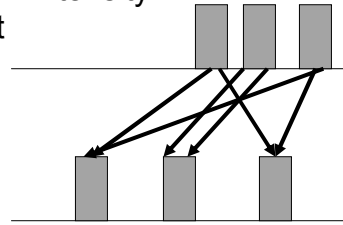
- Make histogram as uniform as possible.
- Make CDF as linear as possible.
  - $h(i)$  uniform  $\rightarrow C(i) = i/K$ , where  $K$  is number intensity values.
- $f(i)$  transforms intensities.
  - $f$  is a monotonic function
- For H.E.
  - $f(i) = C(i)*K$ .
  - Let  $D$  be CDF of new image
    - $D(f(i)) = C(i) \rightarrow D(C(i)*K) = C(i) \rightarrow D(j) = j/K$
  - $C(i)/K$  may not be an integer. Must round.
- New Image  $J$ ,  $J(x,y) = f(I(x,y))$

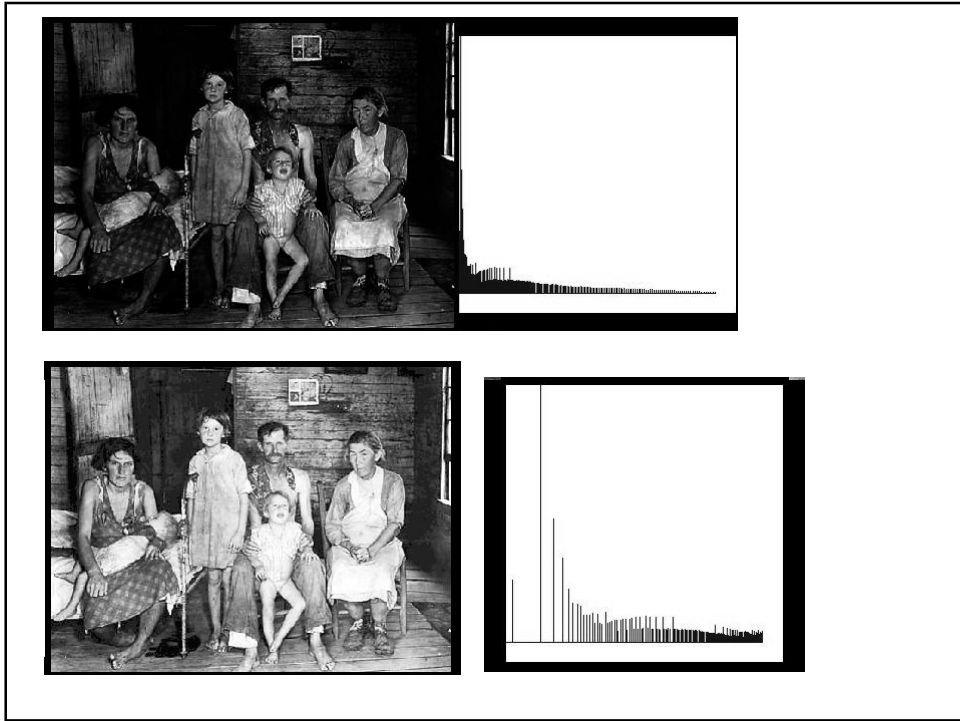
# Properties of Histogram Equalization

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

## A Look at PS 1

- *Demo of results*

## 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:

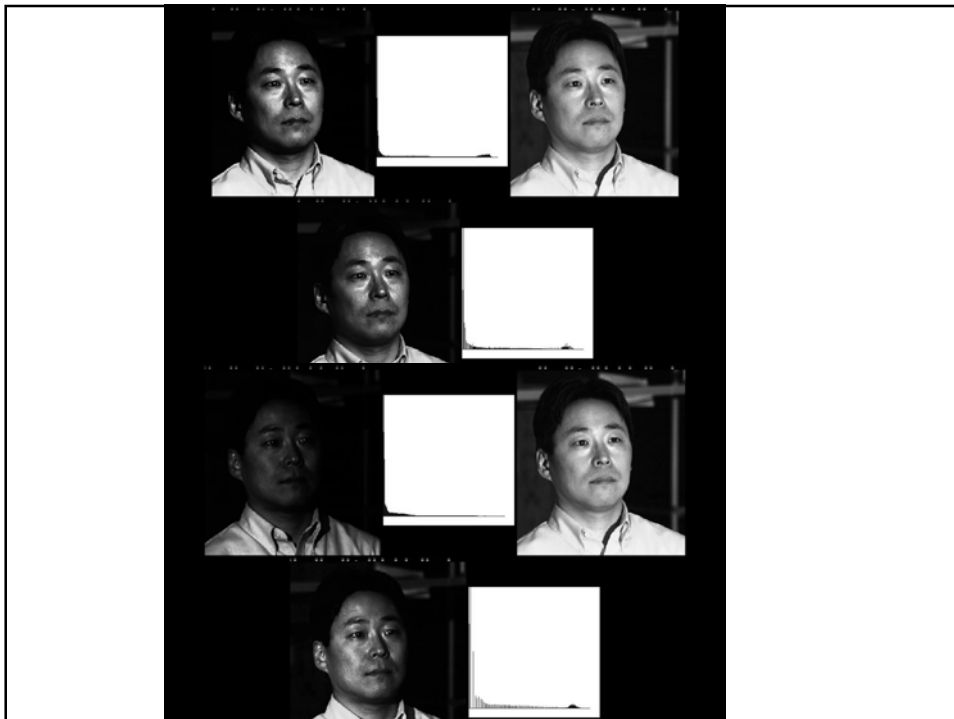
Suppose we have some histogram,  $h(i)$ , which is not uniform. We can make it uniform with the monotonic transformation  $f$  (ie.,  $h(f(i))$  is uniform).

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*).



## Color Histogram

- 3D
- Discretization of space
  - 16 million values too many.
  - Uniform discretization
  - Issues in arbitrariness of discretization
    - Points near boundary of bucket, how do we choose number of buckets (all answers may be wrong in some part of space).
- Note that Histogram Equalization/Specification not trivial
  - How do you decide on mapping?

## Histogram Comparison

- SSD Let  $h$  and  $g$  be two histograms.

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

- Cosine

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

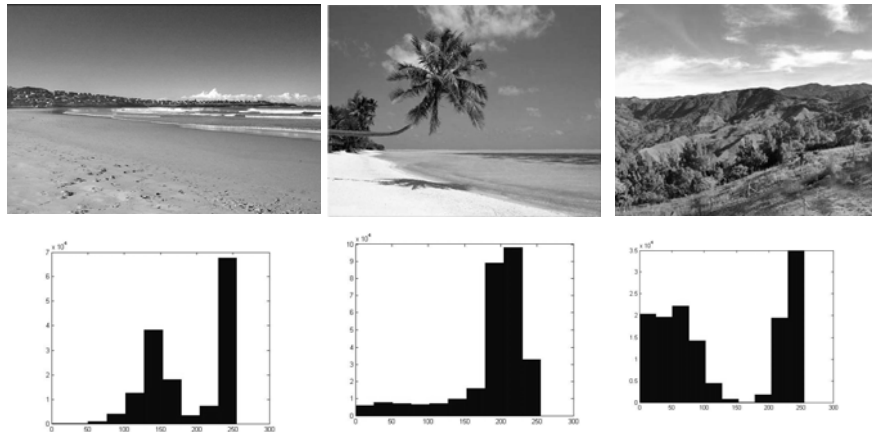
## Histogram Comparison: as a Probability Distribution

- 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

