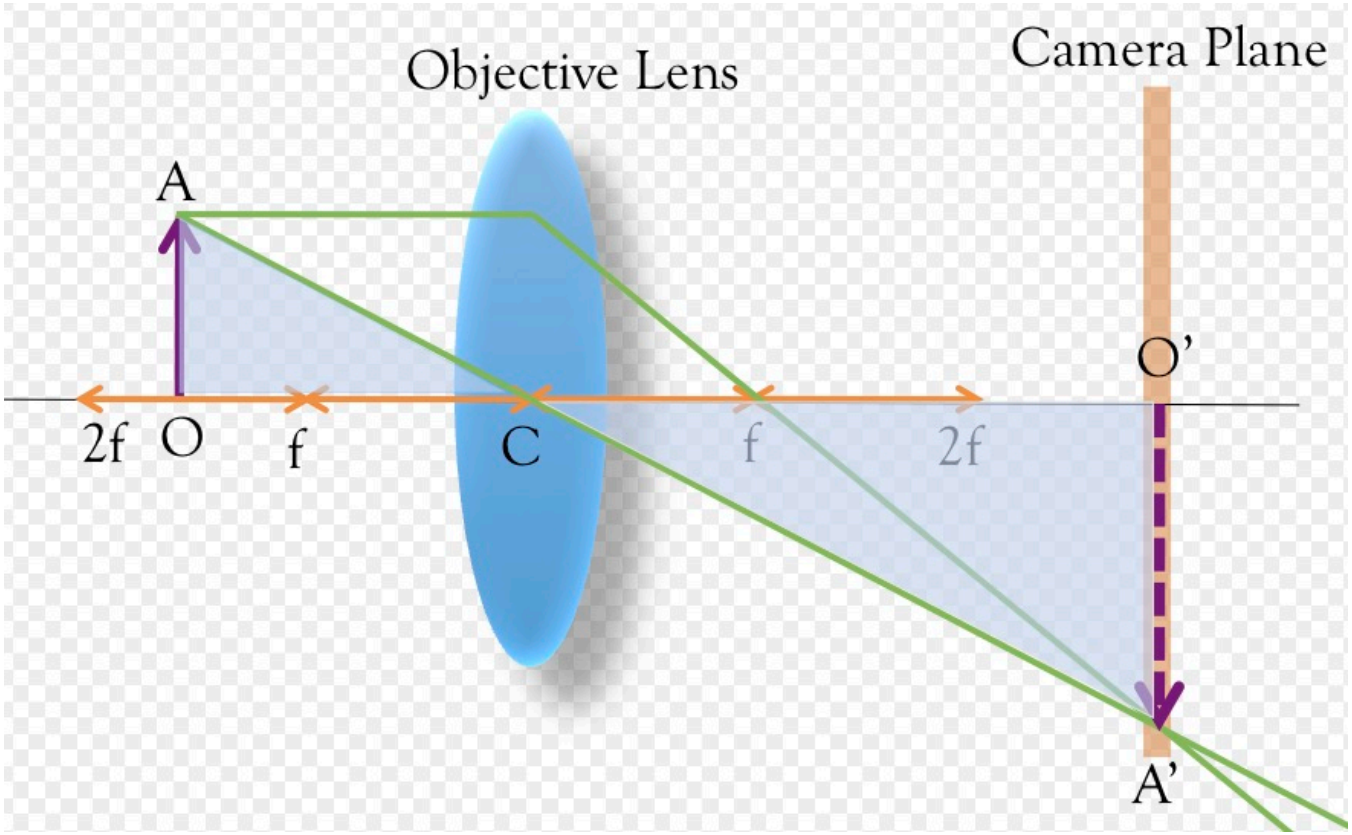
Introduction to Image Processing

Digital Image Acquisition and Representation

Lens and Image Plane



http://www.lenstrotek.com/archives/category/microscope-fundamentals

Image Sensor

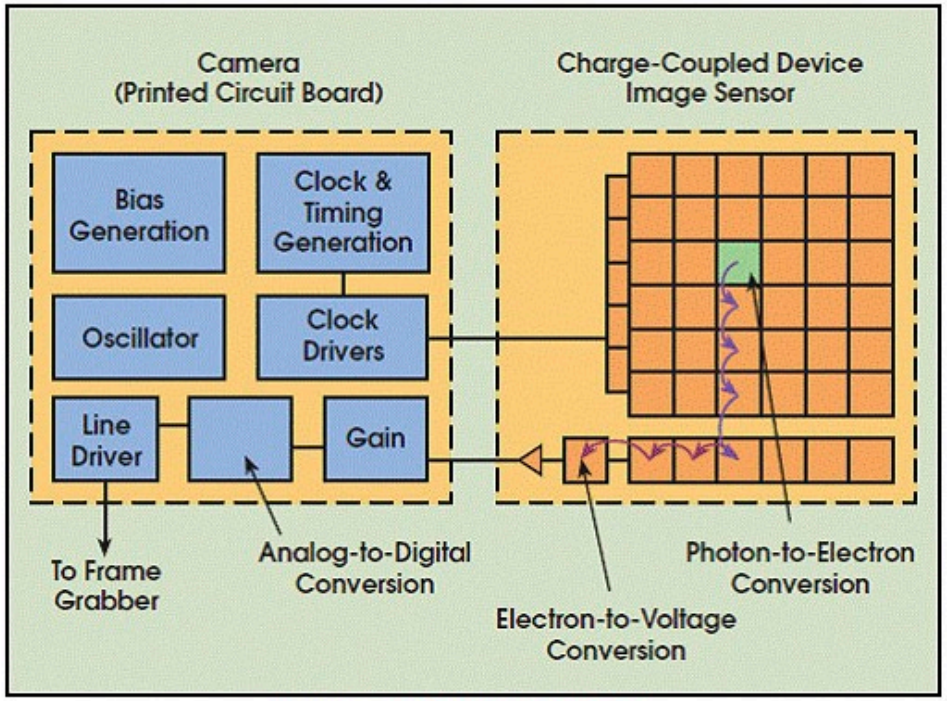


Figure 1: Diagram of a CCD.

https://www.tedpella.com/cameras_html/ccd_cmos.htm

RGB Images (e.g. BMP)

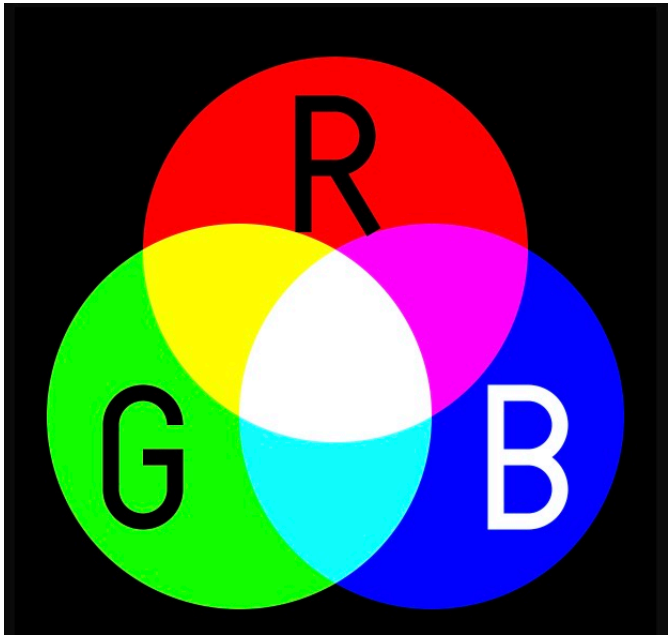
- **Digital images are generated by sampling light intensity in a 2-D plane**
- **Conceptually, three values are measured at each location: (R, G, B)**
  - The sample density determines the resolution

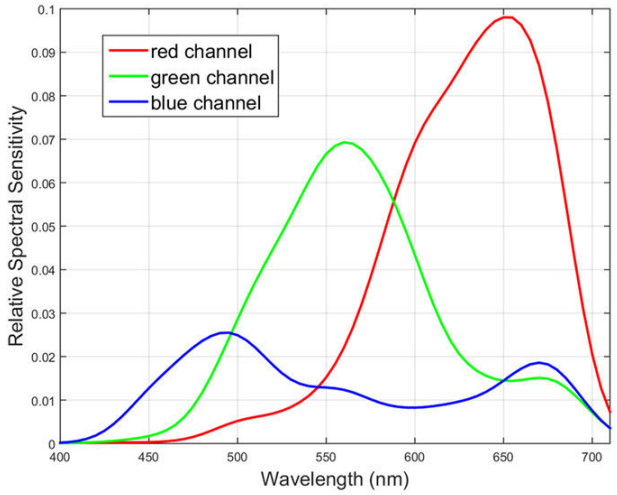
(R,G,B)	(R,G,B)	(R,G,B)	(R,G,B)
(R,G,B)	(R,G,B)	(R,G,B)	(R,G,B)
(R,G,B)	(R,G,B)	(R,G,B)	(R,G,B)
(R,G,B)	(R,G,B)	(R,G,B)	(R,G,B)

Each value R, G, B is
a number from [0, 255]

Primary Colors

- **Different mixtures of (R, G, B) determine which color you perceive**
  - R, G, and B aren't "pure" colors, they have a spectrum

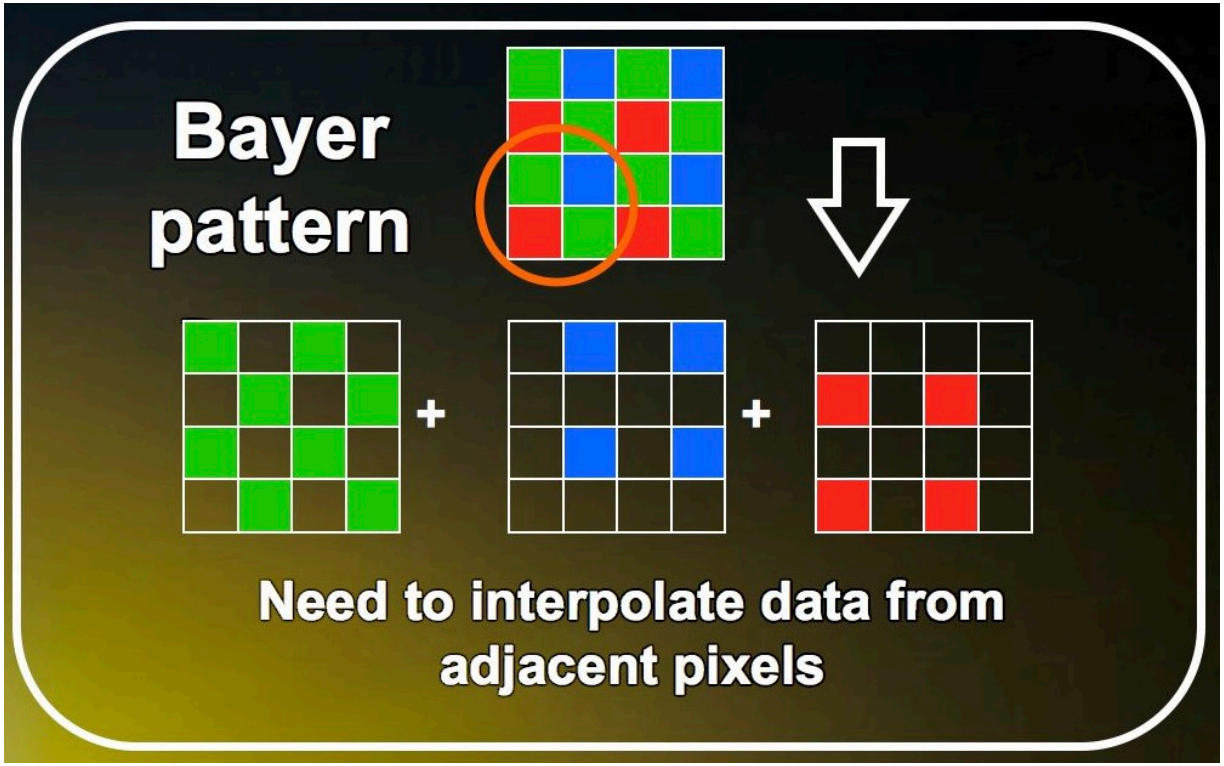




https://upload.wikimedia.org/wikipedia/commons/thumb/c/c2/AdditiveColor.svg/1200px-AdditiveColor.svg.png

Bayer Sampling Pattern

- **The RGB Pixels on an image sensor are close to each other, but not exactly co-located**



http://3.bp.blogspot.com/-wBQ7xIuzF74/TZYJv27zPEI/AAAAAAAABcY/aNgrL3NX6MQ/s1600/Sony+Bayer+sampling+comparison.jpg

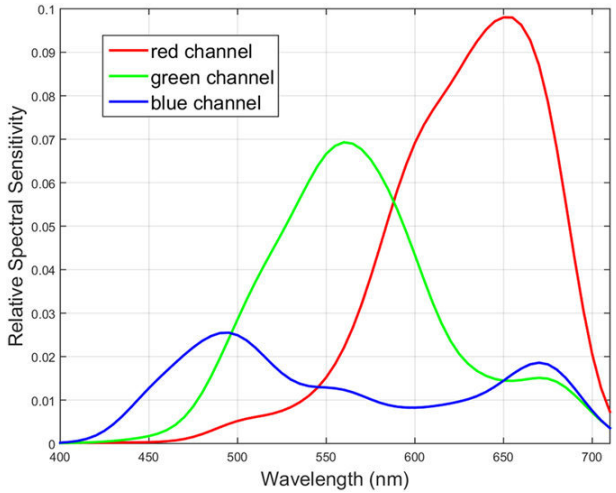
RGB to Grayscale Conversion

RGB to Grayscale conversion

- **Often it is sufficient to process only a grayscale image**
  - Less computation

RGB to Grayscale conversion cont.

- **In general, converting between color spaces is a matrix multiply**
  - Different colorspaces assume different spectral characteristics for their primaries
    - ✓ Different primary spectral characteristics require different weights to generate the same perceived color



Color space conversion can be implemented as a matrix multiply

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} .299 & .587 & .114 \\ -.14713 & -.28886 & .436 \\ .615 & -.51499 & -.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

https://www.researchgate.net/figure/Spectral-response-of-a-Canon-T3i-APS-C-CMOS-image-sensor-with-a-Bayer-pattern-CFA-This_fig5_287249644

RGB to Grayscale conversion cont.

- **(Y, U, V) is the color space used for TV**
- **When converting (R, G, B) to grayscale, it is only necessary to compute Y**
  - Computing Y is just a dot product (first row of matrix "dot" (R, G, B))

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} .299 & .587 & .114 \\ -.14713 & -.28886 & .436 \\ .615 & -.51499 & -.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

https://www.researchgate.net/figure/Spectral-response-of-a-Canon-T3i-APS-C-CMOS-image-sensor-with-a-Bayer-pattern-CFA-This_fig5_287249644

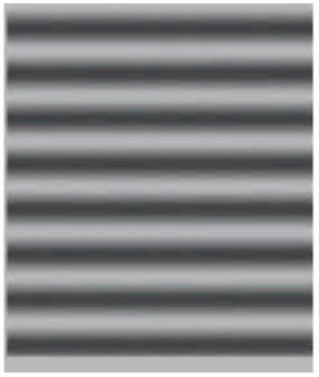
Contrast Enhancement

Contrast

- **Contrast, intuitively, deals with how distinct two things are**

One example of image contrast. Other definitions are also used in other contexts

Low frequency horizontal



High frequency vertical

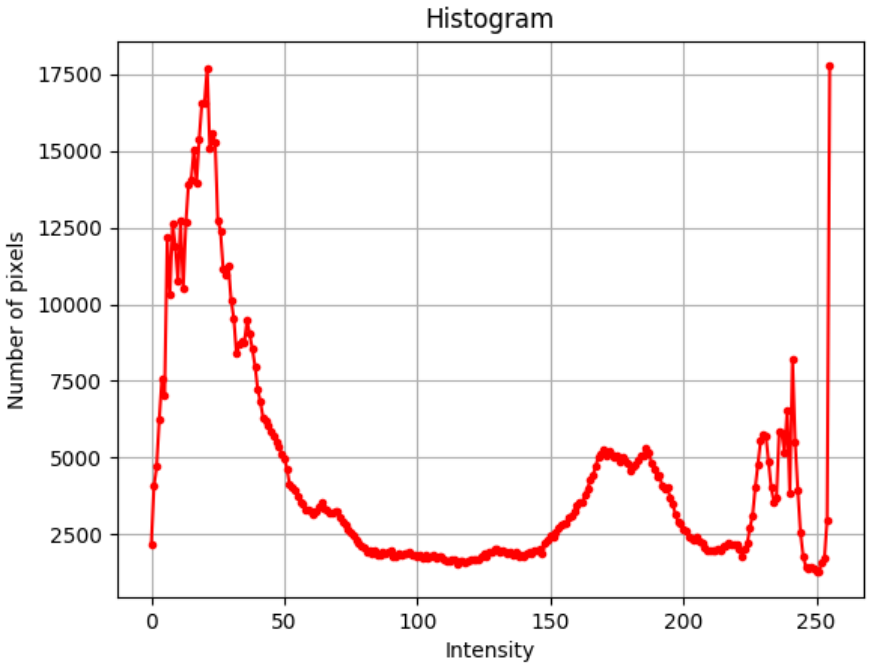
Figure 8: Spatial frequency patterns for measuring contrast sensitivity

The sine wave's frequency, orientation, and intensity (i.e. peak-to-peak magnitude variation) can all varied, and a viewer's ability to detect the sine wave can be measured. The contrast used for this measurement is known as the *Michelson contrast* and is defined as

$$C(I_{max}, I_{min}) = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \quad (13)$$

Image Histogram

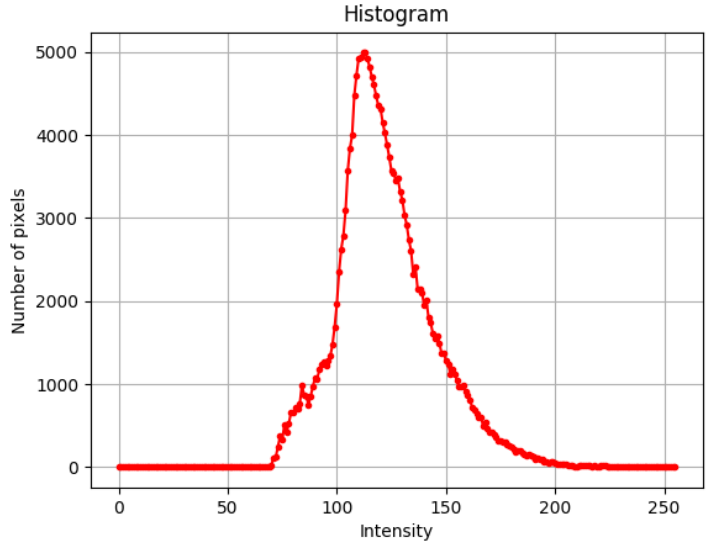
- **Count the number of pixels at each intensity level and graph them**
  - Often normalize to yield relative frequencies



Histogram of previous picture: note the saturation at 255

Low-Contrast Image

- **Low-contrast Images have narrow histograms**



Histogram of a low-contrast image

Contrast Enhancement

- **To enhance the contrast, we need to "spread-out" the histogram**
  - One possibility: apply a linear-stretch to the image, mapping all values less than t_{min} to 0, and all values greater than t_{max} to 255
- **Another approach is to apply a non-linear function to the pixel intensities**
  - A common choice is "equalize" the histogram. Ideally, every intensity would then occur with the same frequency
    - ✓ This is called *histogram equalization* in the literature

Algorithm for Lab

- **Build a table of 256 entries that maps input intensities to output intensities**
  - A numpy array of size 256, indexed by input intensity
- **Do not remap pixels in the original image that have intensities of 0 or 255**
  - Only remap pixels that have intensities between [1, 254]
- **Find the sum, *S*, of how many pixels have an intensity in the range [1, 254] (you will need the histogram)**
- **Compute the ideal number of pixels, *P*, for each intensity level**
  - $P = S / 254$

Algorithm for Lab cont.

- **Define a current target *T*. Initially, *T* = *P***
- **Starting at input intensity 1 (i.e. histogram index 1) and output intensity (*out_val*) 1:**
  - Step through the histogram, incrementing a temporary sum (*curr_sum*), until a number of pixels greater than *T* have been processed
    - ✓ Map all these input levels to 1 by making appropriate entries in your remapping array
  - Determine the next output intensity (we skip forward depending on by how much we exceeded our target)
    - ✓ *out_val = round(curr_sum / P)*
  - Update *T* to *T = out_val*P*
  - Repeat until the remapping array is complete

Edge Detection

Edge Detection

- **In many computer vision applications, it is useful to detect the edges of objects**
- **What characterizes edges?**
  - Edges occur where transitions occur
    - ✓ Intensity
    - ✓ Color
    - ✓ Texture
- **We'll focus on intensity since we're processing grayscale images**
  - Rapid intensity transitions occur where the deriviative is large

Gradient Review

- **The gradient of a real-valued function with an N-D domain is an N-D vector which points in the direction of maximum change**
  - Its magnitude is the rate of change in that direction. In 2-D we have

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$$

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

Edge Detection with the Gradient

- **Image intensity is a 2-D function of (x, y)**
- **Compute the magnitude of the gradient**
  - Apply a threshold. Declare values *greater* than the threshold to be edges
- **We can locally approximate the gradient at each pixel**
- **The Sobel edge detection operator combines two operations**
  - Lowpass image filtering
  - Estimating the gradient of the filtered image

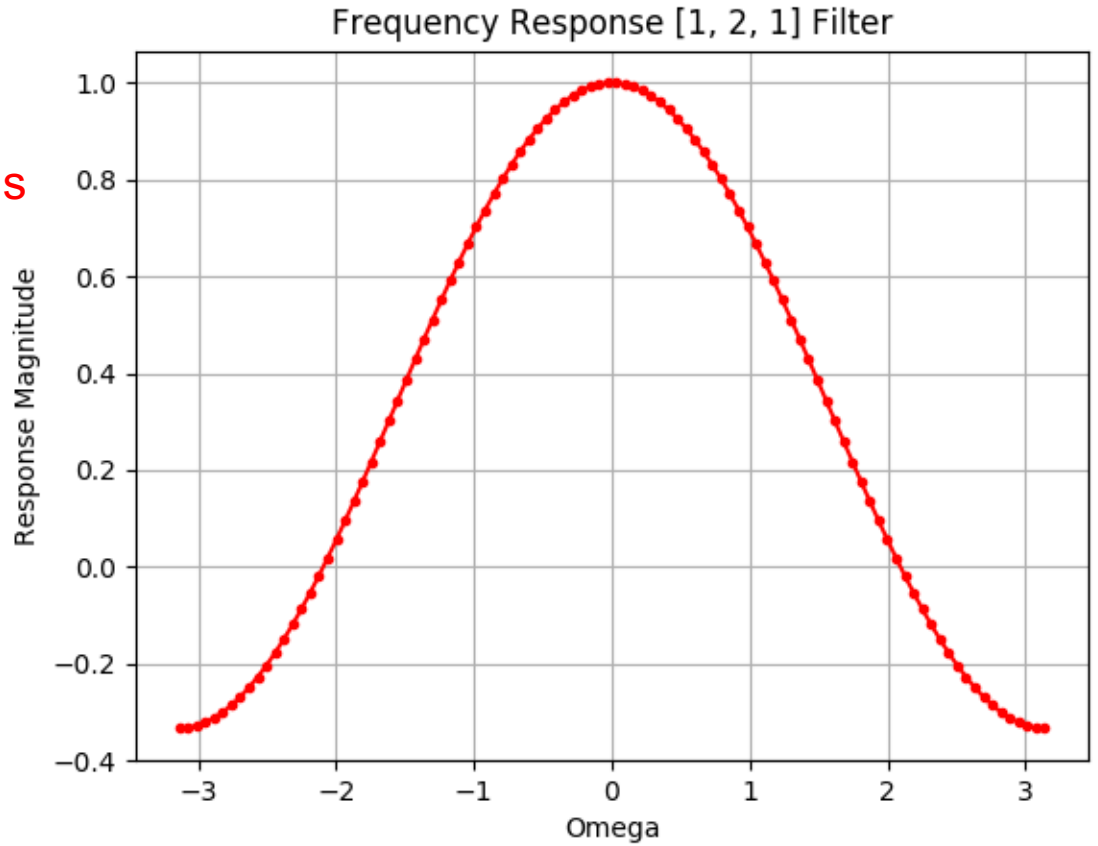
Convolution

- **Convolution in the time domain is multiplication in the frequency domain**
- **Exercise: Convolve {1, 2, 1} with {1, 2, 3, 4, 5, 4, 3, 2, 1}**
  - Convolution by strips of paper!

A Simple lowpass filter: (1, 2, 1)

- **Frequency response of (1, 2, 1) is a gentle rolloff**
  - Convolving with (1, 2, 1) is lowpass filtering

Remember: π corresponds to ½ the sampling rate



Estimating the Horizontal Derivative

- **Estimating** $\partial f / \partial x$
  - First, lowpass filter the image vertically
  - Second, take the difference in the horizontal direction
    - ✓ The 2-D convolution kernel to apply to the image looks like this

$$\Delta h = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

This estimates the horizontal derivative for the center pixel

Estimating the Vertical Derivative

- **Estimating** $\partial f / \partial y$
  - First, lowpass filter the image horizontally
  - Second, take the difference in the vertical direction
    - ✓ The 2-D convolution kernel to apply to the image looks like this

$$\Delta v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

This estimates the horizontal derivative for the center pixel

Estimating the Gradient

- **Use the horizontal and vertical derivate estimates to estimate the magnitude of the gradient for a pixel**

$$\|\nabla f\| = \sqrt{(\Delta h)^2 + (\Delta v)^2}$$

Edge Detection for Previous Picture

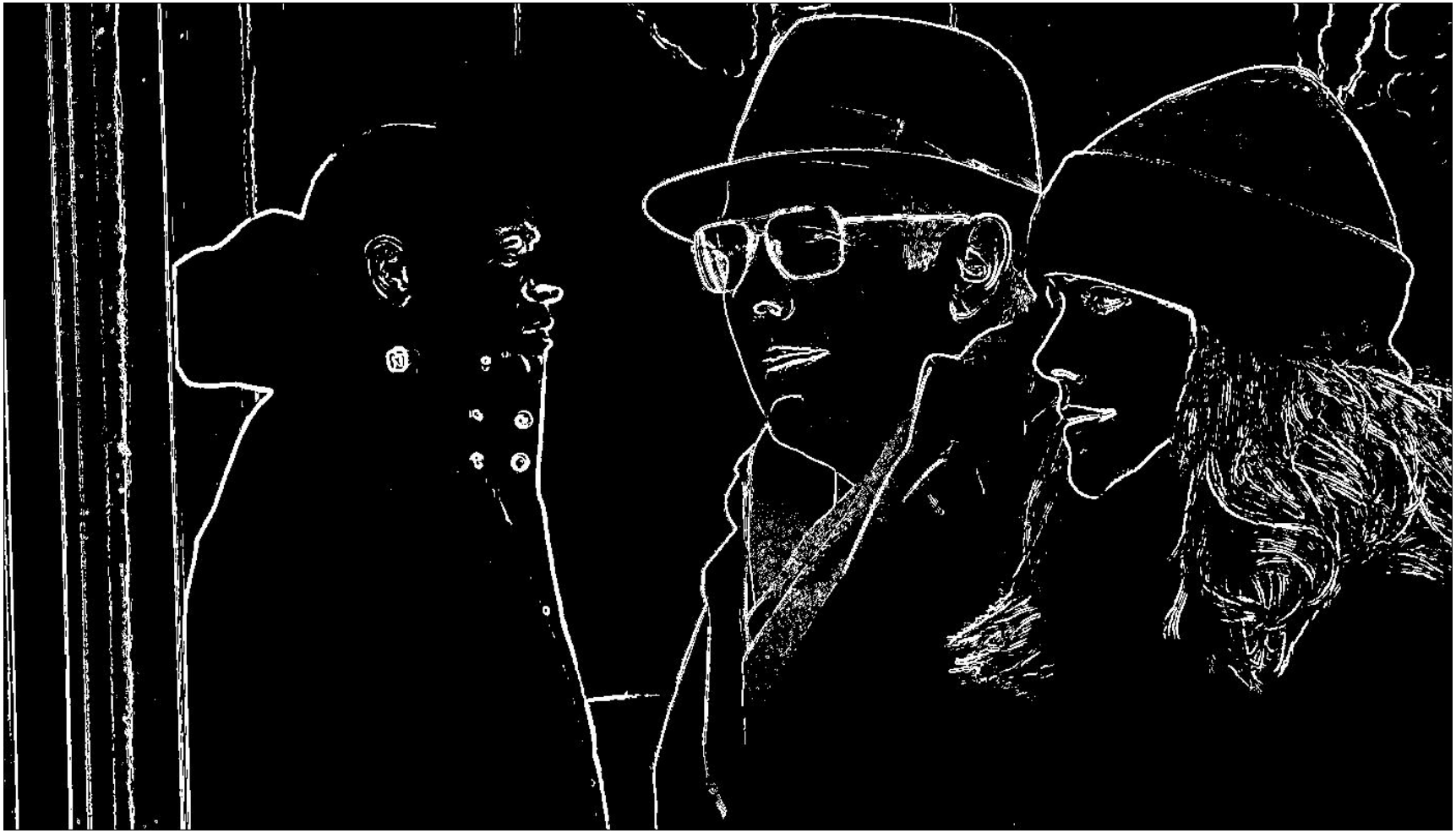
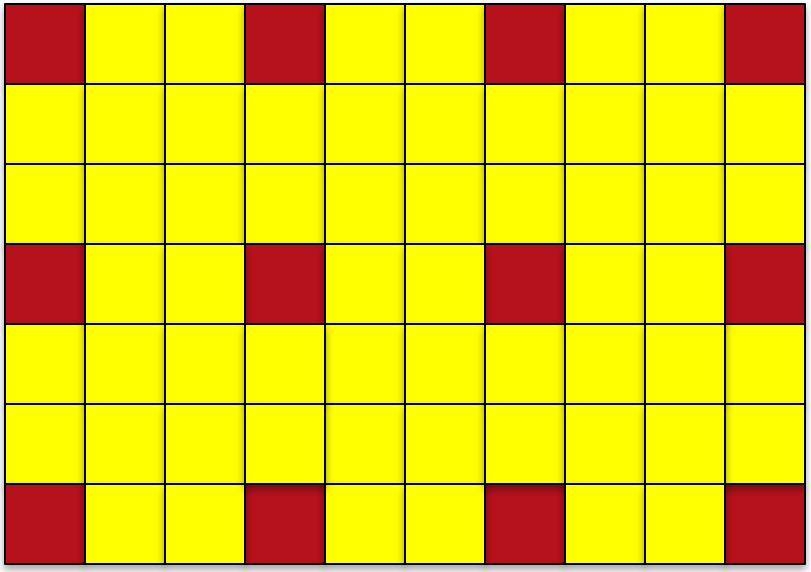


Image Resizing

Image Sampling

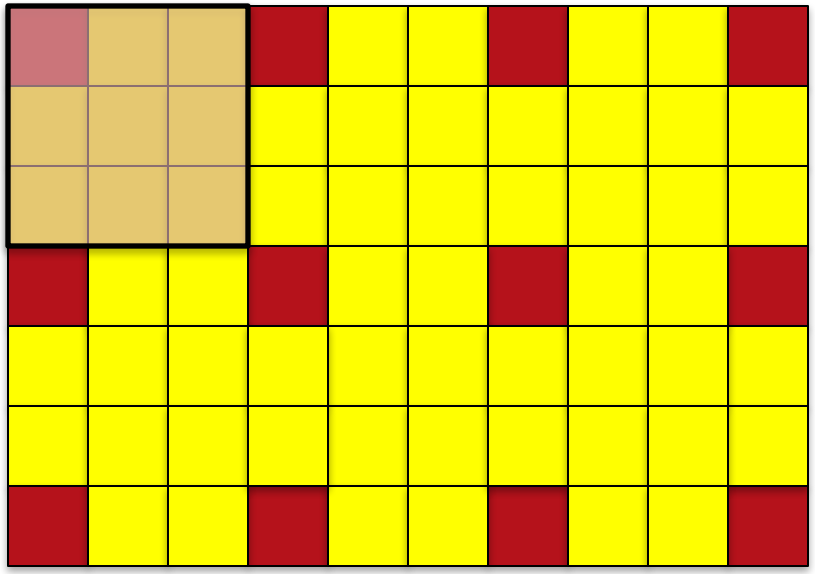
- **Downsampling**
  - This illustrates sub-sampling by 3 in each dimension: keep only the red pixels, and discard the yellow



Filter Before Subsampling

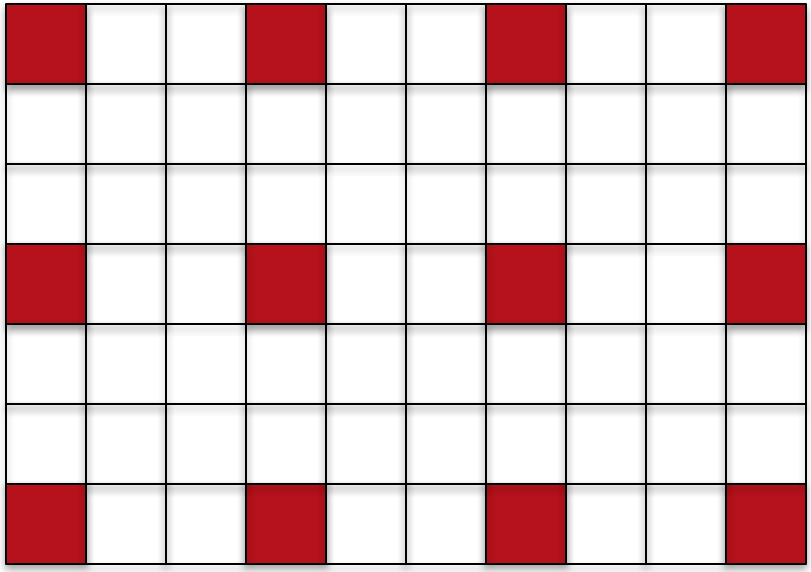
- **Correct process**
  - Lowpass filter before subsampling
    - ✓ This minimizes aliasing
    - ✓ A simple lowpass filter: average 3-by-3 blocks

After filtering, each red pixel is the average of a 3-by-3 block



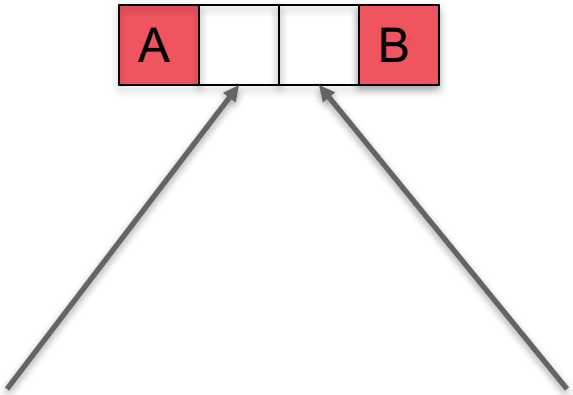
Upsampling

- **We upsample to make a larger image**
  - Spread out the pixels to occupy the red squares
    - ✓ The white squares need to be filled in by filtering or interpolation



1-D Linear interpolation

- **Assume the intensities of the red pixels are as indicated**



This pixel becomes

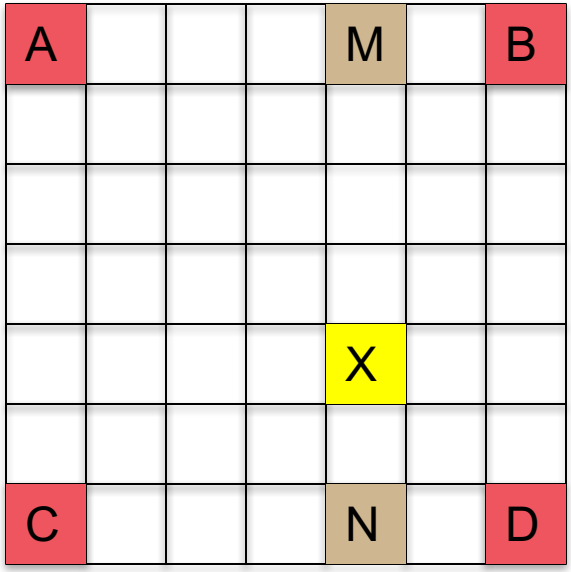
$$A + \frac{1}{3}(B - A)$$

This pixel becomes

$$A + \frac{2}{3}(B - A)$$

Bi-linear Interpolation

- **Assume the intensities of the red pixels are as indicated**



To bi-linearly interpolate X:
--Interpolate M using A and B
--Interpolate N using C and D
--Interpolate X using M and N

Example

- **Downsampling then up-sampling by 4**