

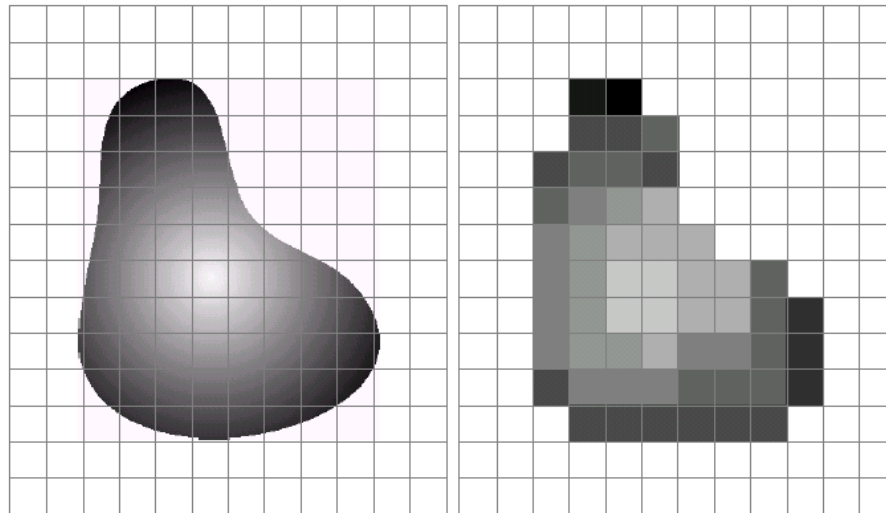
Advanced Multimedia

Image Content Analysis

Tamara Berg

How are images stored?

Reminder: Images

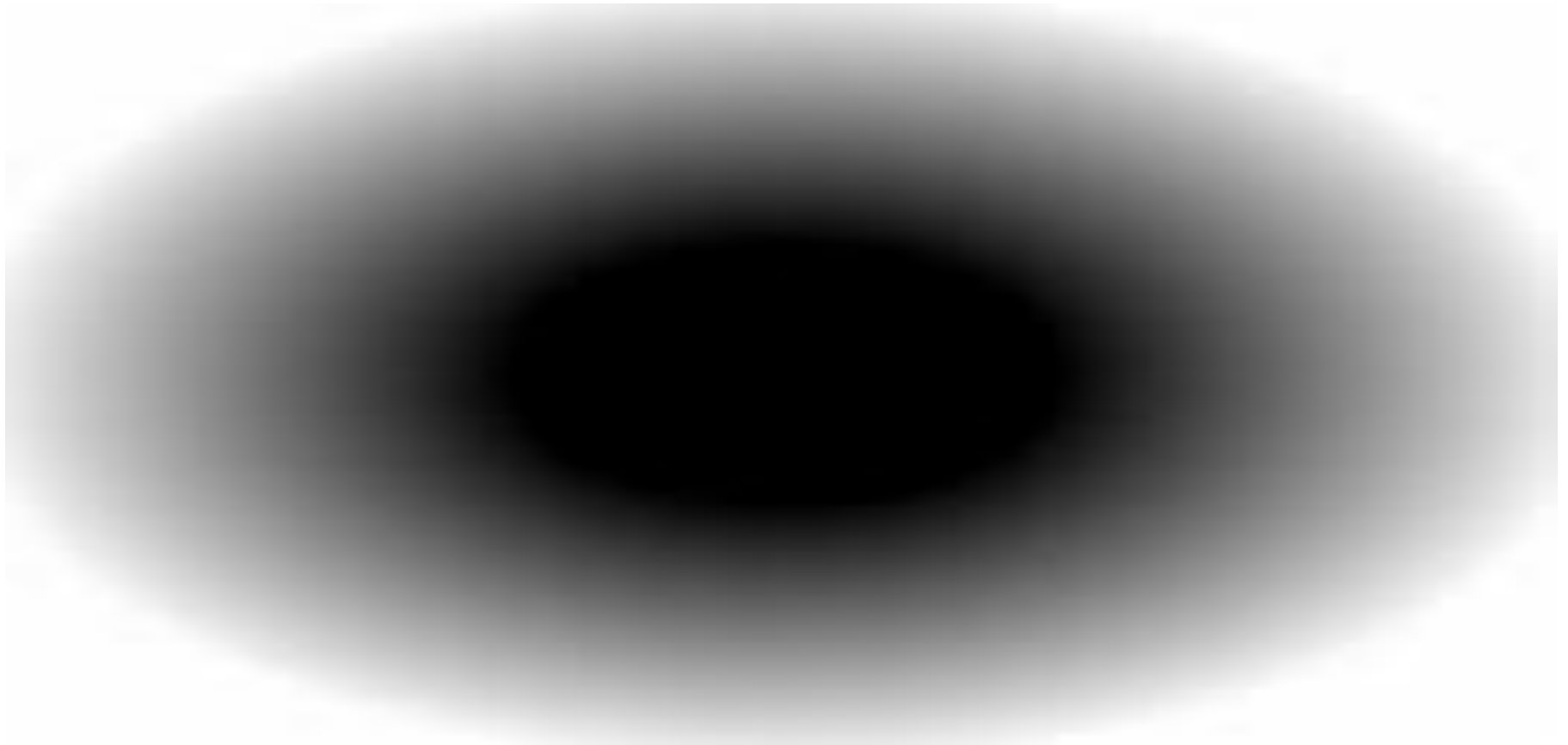


a b

FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Images are sampled and quantized measurements of light hitting a sensor.

Images in the computer



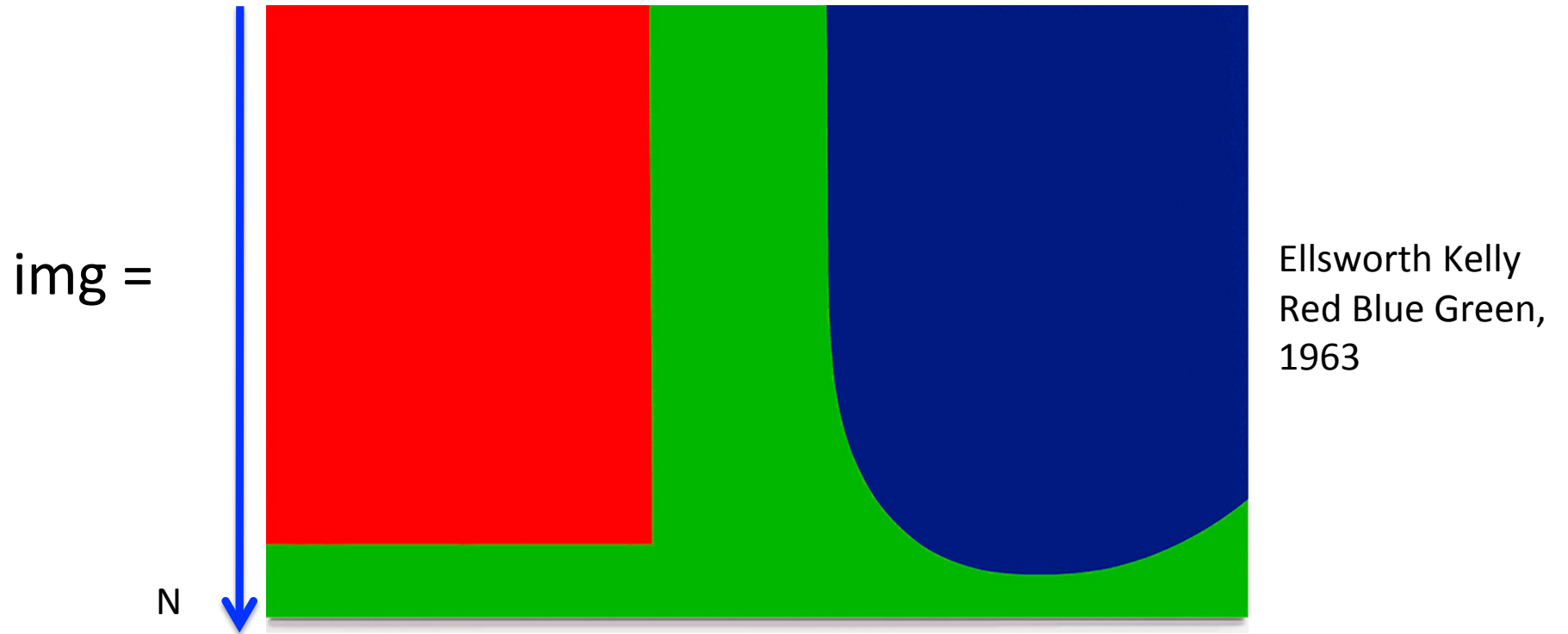
Color Images

img =

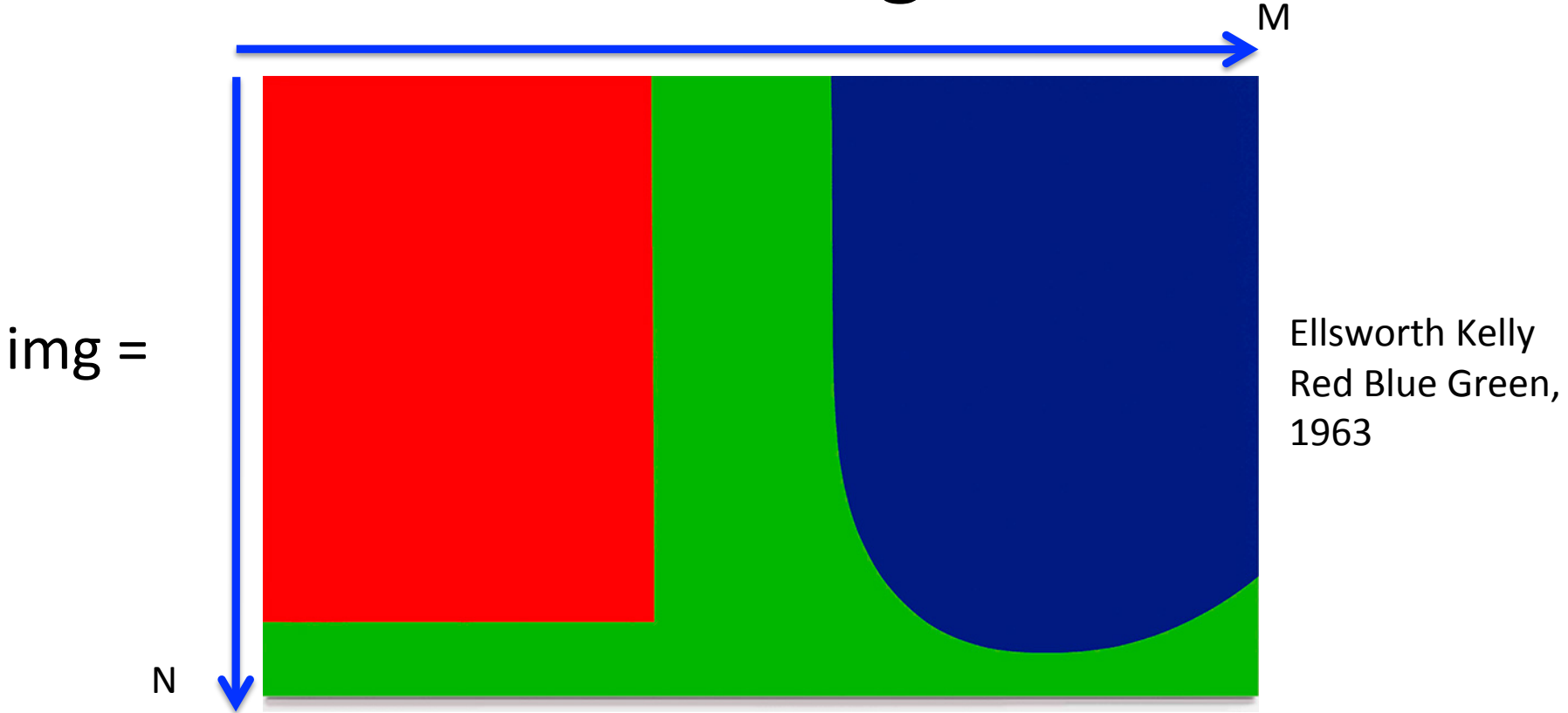


Ellsworth Kelly
Red Blue Green,
1963

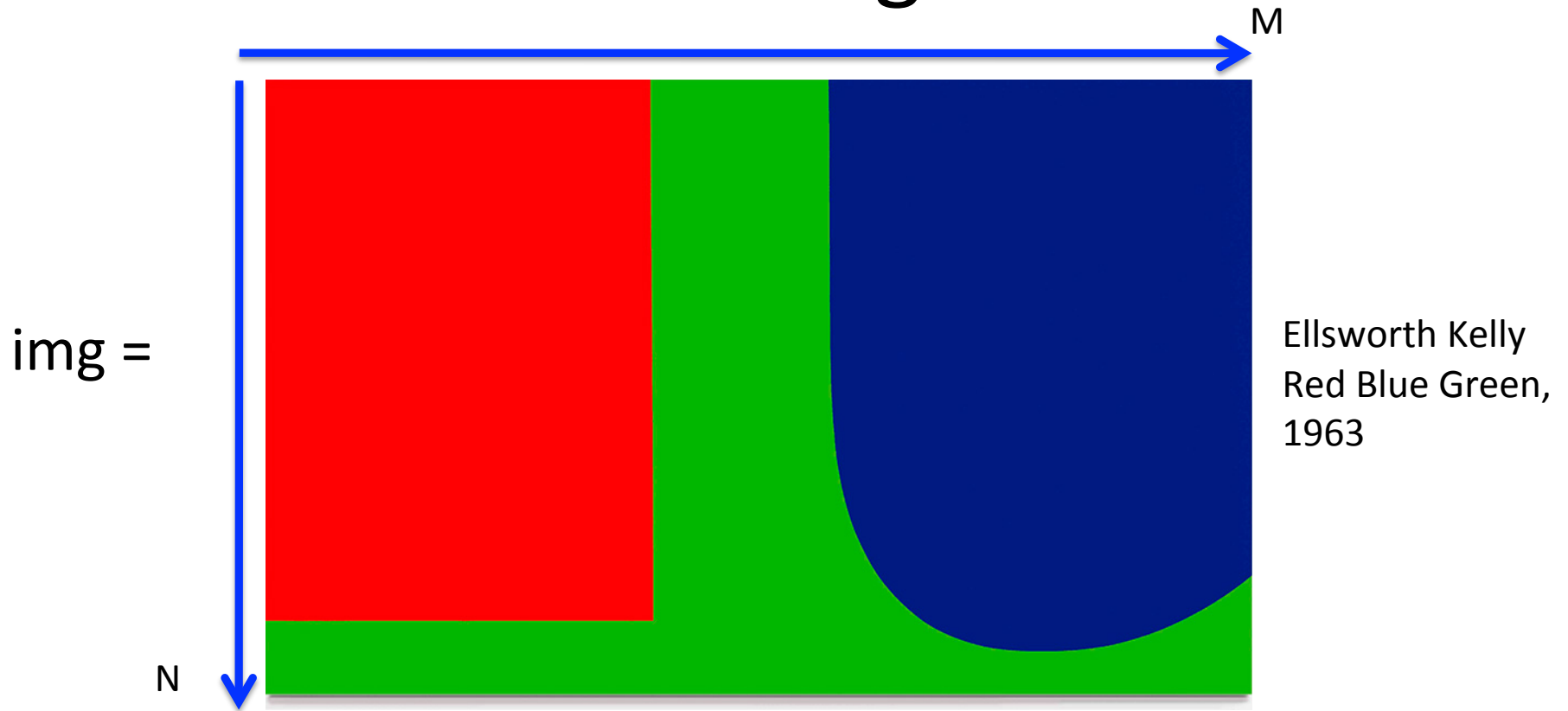
Color Images



Color Images

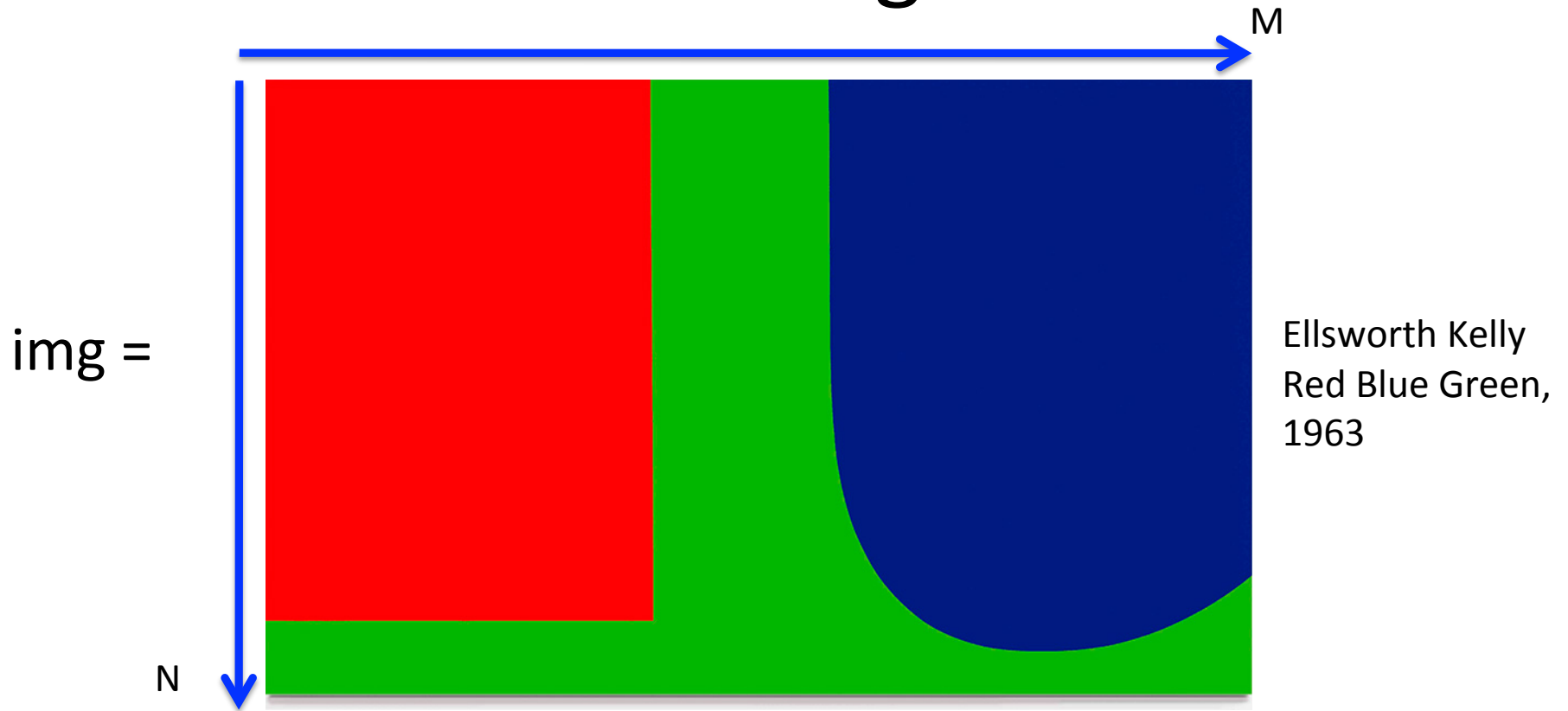


Color Images



- Stored as 3d matrix of r, g, and b values at each pixel
 - Image matrix would be size $N \times M \times 3$
 - $? = \text{img}(:, :, 1)$ $? = \text{img}(:, :, 2)$ $? = \text{img}(:, :, 3)$

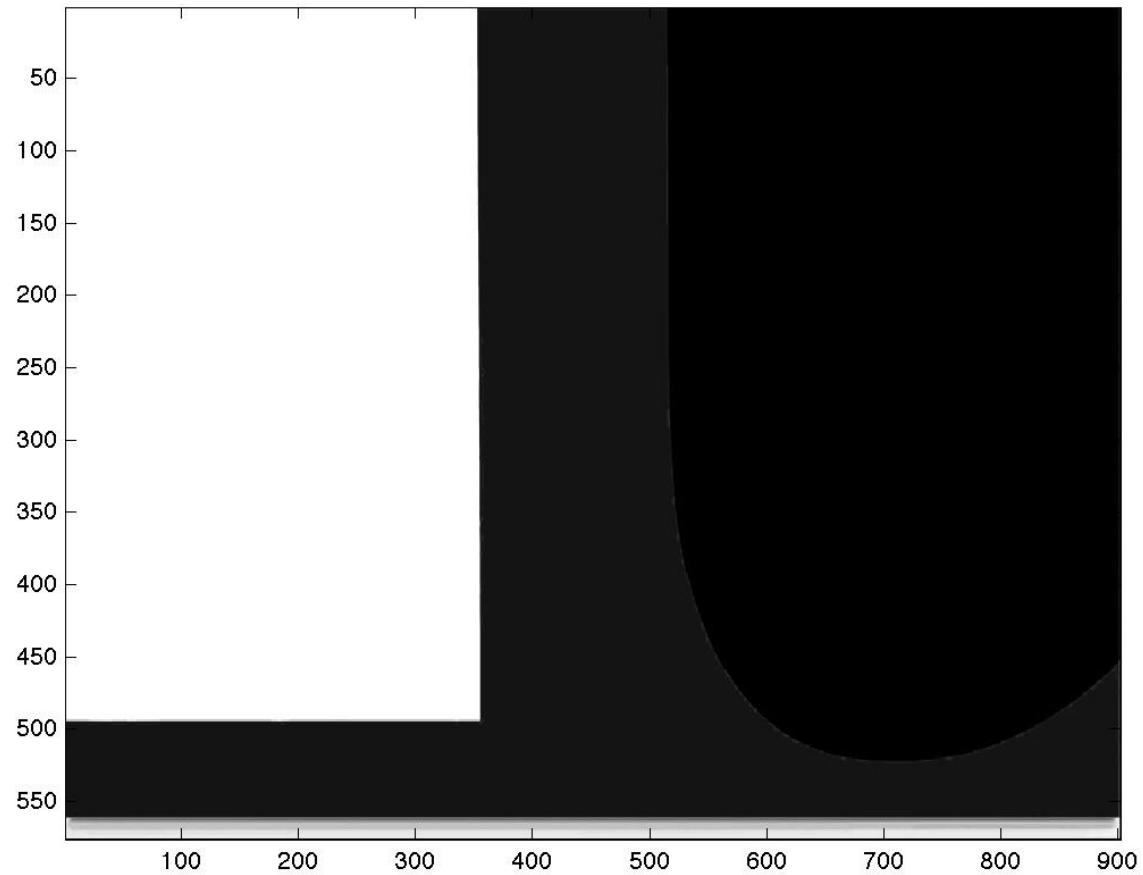
Color Images



- Stored as 3d matrix of r, g, and b values at each pixel
 - Image matrix would be size $N \times M \times 3$
 - $R = \text{img}(:, :, 1)$ $G = \text{img}(:, :, 2)$ $B = \text{img}(:, :, 3)$

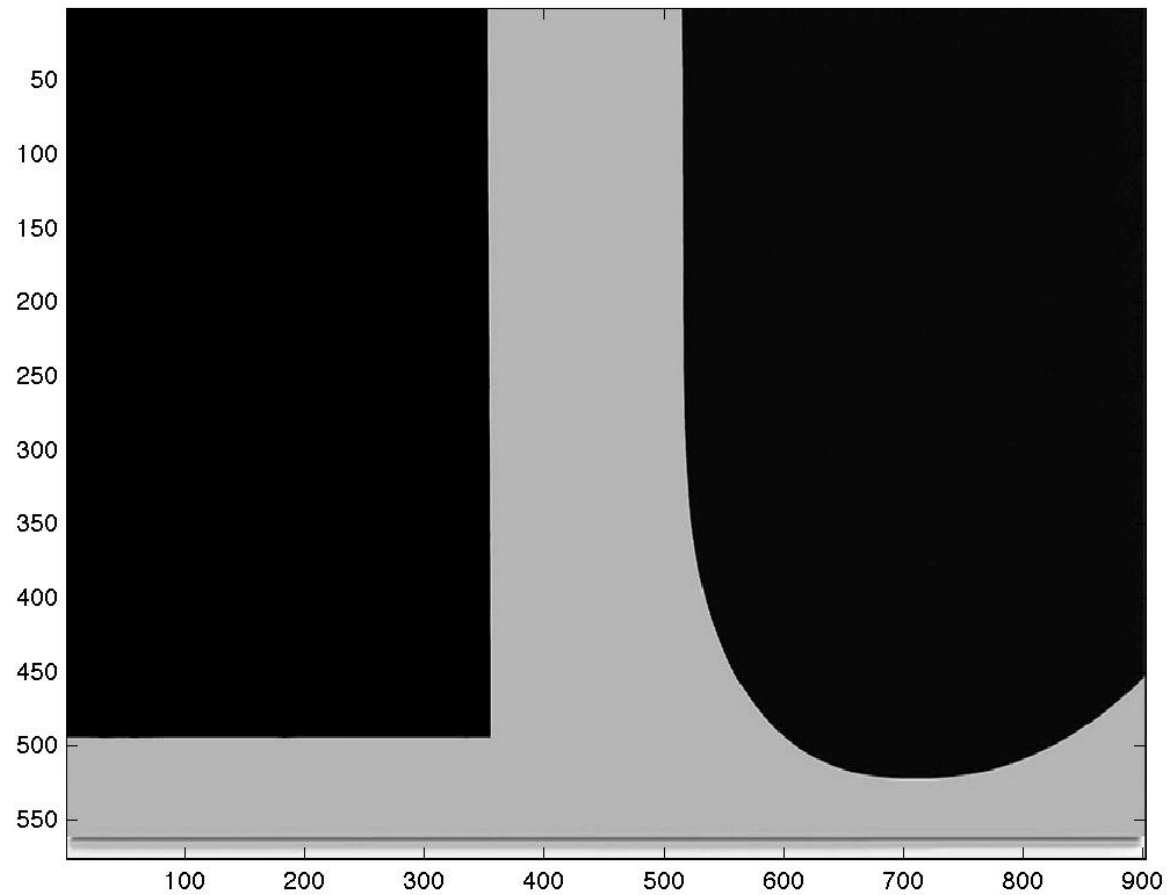
Red component

`img(:,:,1) =`



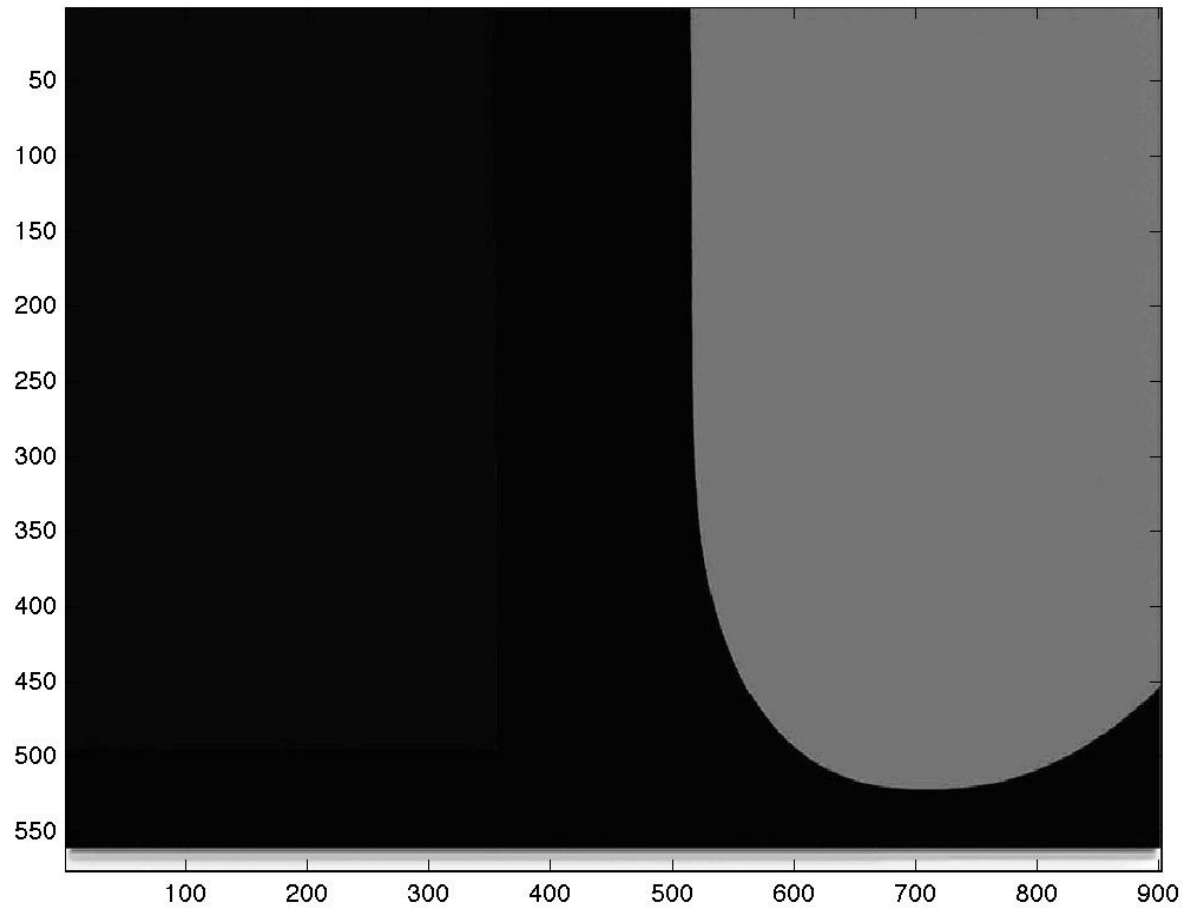
Green component

`img(:,:,2) =`

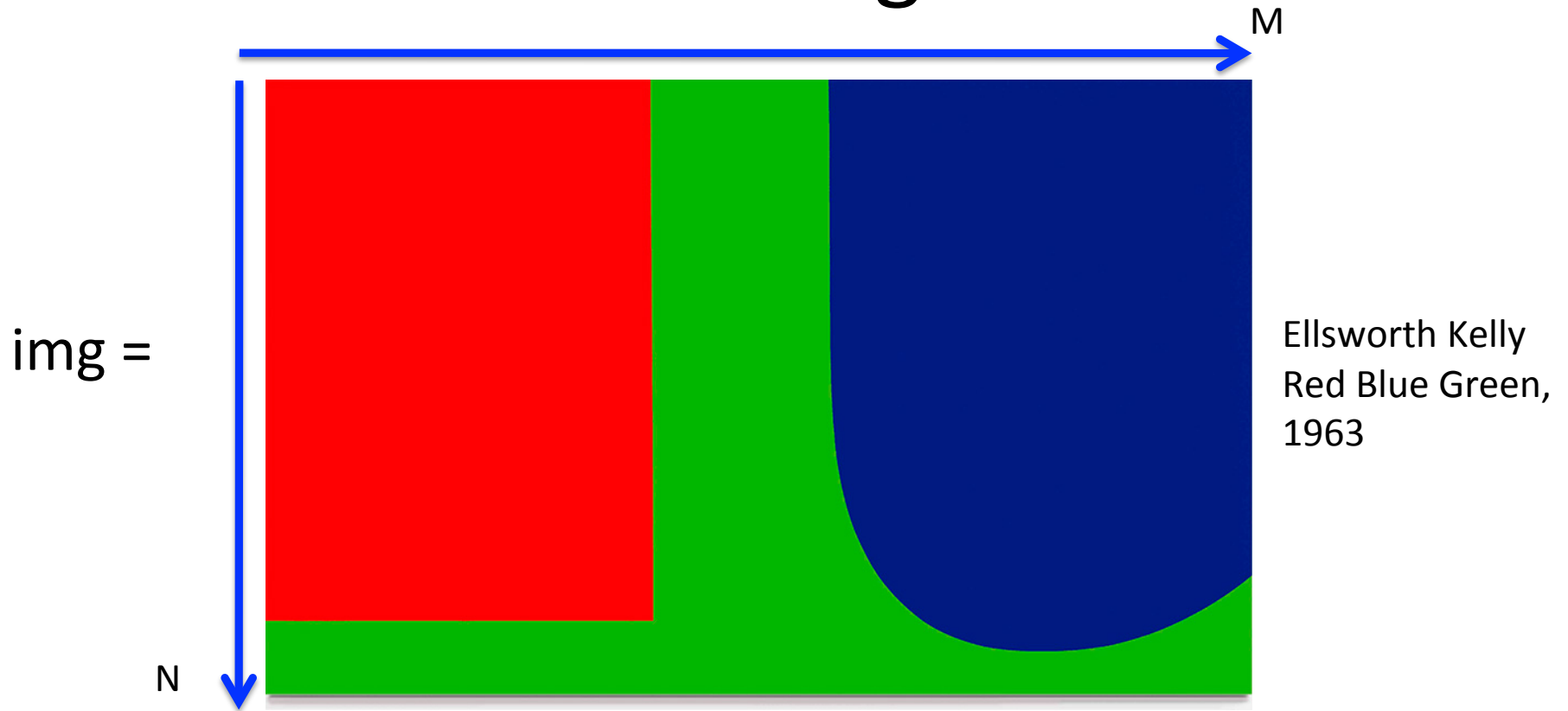


Blue component

`img(:,:,3) =`

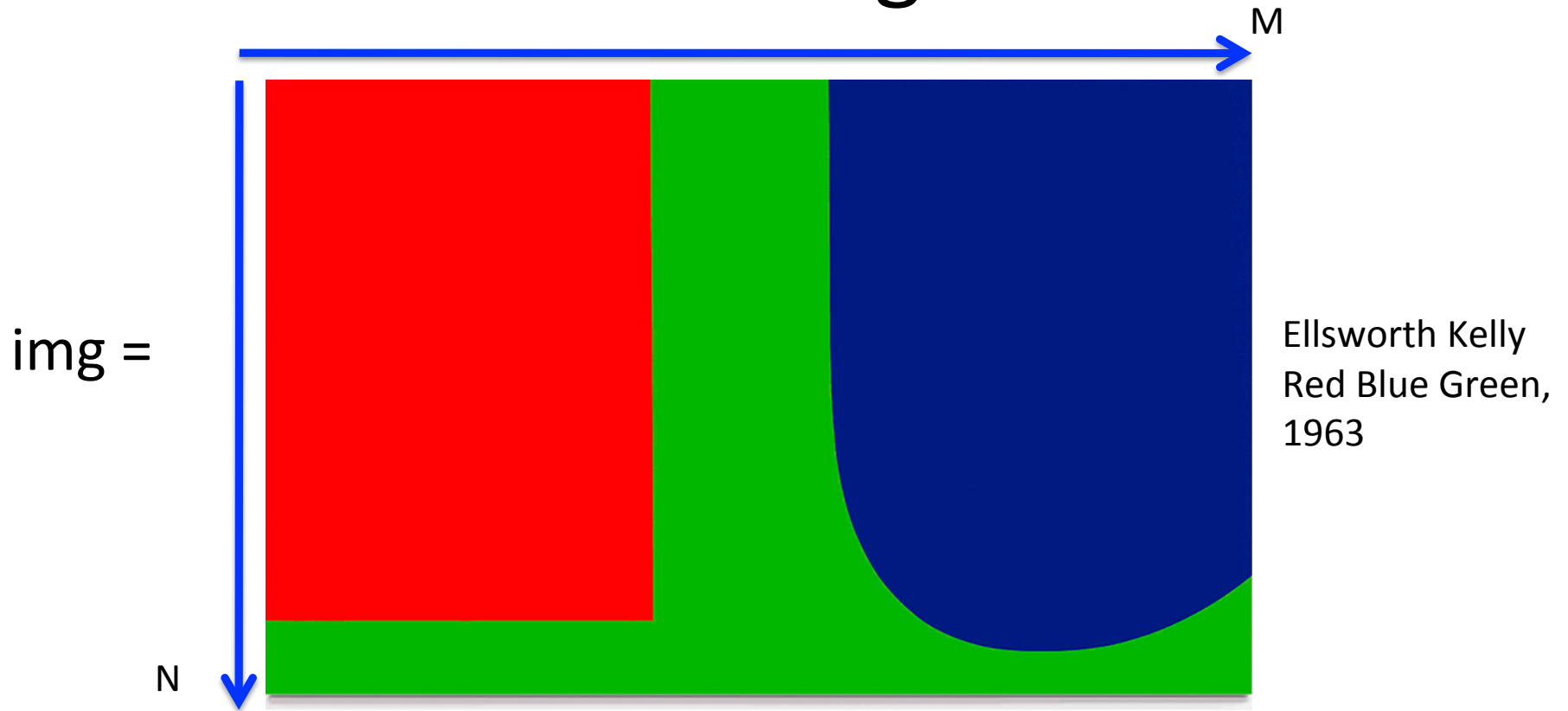


Color Images



- Stored as 3d matrix of r , g , and b values at each pixel
 - How could you get the r, g, b values for pixel (i, j) ?

Color Images



- Stored as 3d matrix of r, g, and b values at each pixel

```
vals = img(i,j,:);
```

```
Then r= vals(1); g = vals(2); b = vals(3);
```

Color Images



- Stored as 3d matrix of r , g , and b values at each pixel
 - So r, g, b values are stored in $\text{img}(i, j, :)$.
 - In the case above these values might be $[1 \ 0 \ 0]$.

How should we represent images?

Motivation

- Image retrieval
 - We have a database of images
 - We have a query image
 - We want to find those images in our database that are most similar to the query

Motivation

- Image retrieval
 - We have a database of images
 - We have a query image
 - We want to find those images in our database that are most similar to the query
- Similarly to text retrieval, & music retrieval we first need a representation for our data.

How should we represent an image?



First try



Just represent the image by all its pixel values

First try

Img1 =



Img2 =



Say we measure similarity as:
 $\text{sim} = \text{sum}(\text{abs}(\text{img1} - \text{img2}))$

First try

Img1 =



Img2 =



Say we measure similarity as:
 $\text{sim} = \text{average diff between values in img1 and img2}$

How similar are these two images? Is this bad?

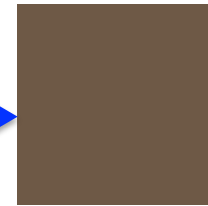
What do we want?

- Features should be robust to small changes in the image such as:
 - Translation
 - Rotation
 - Illumination changes

Second Try



Photo by: [marielito](#)

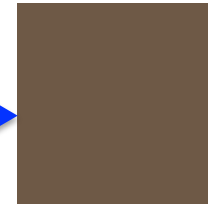


Represent the image as its average pixel color

Second Try



Photo by: [marielito](#)



Represent the image as its average pixel color

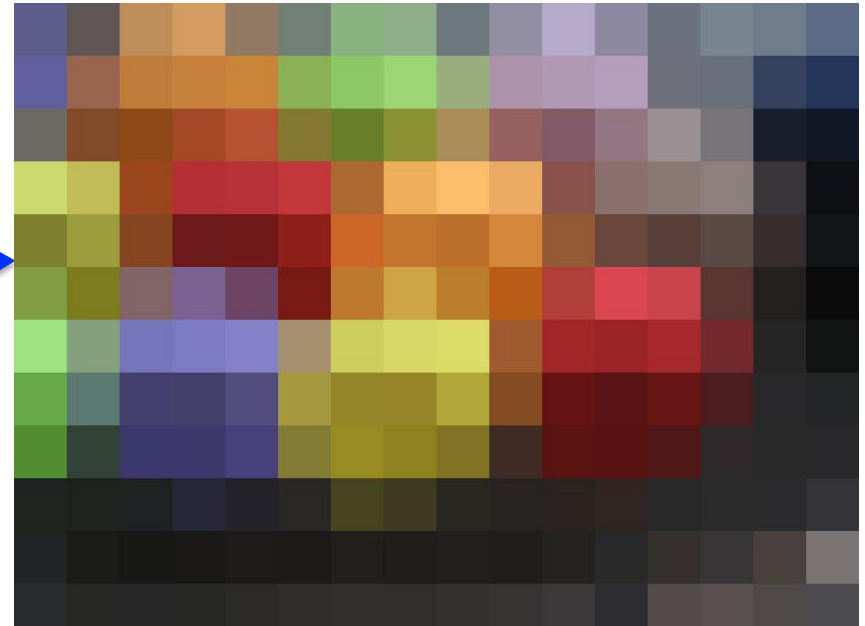
Pros?

Cons?

Third Try



Photo by: [marielito](#)



Represent the image as a spatial grid of average pixel colors

Pros?

Cons?

QBIC system



[QBIC link](#)

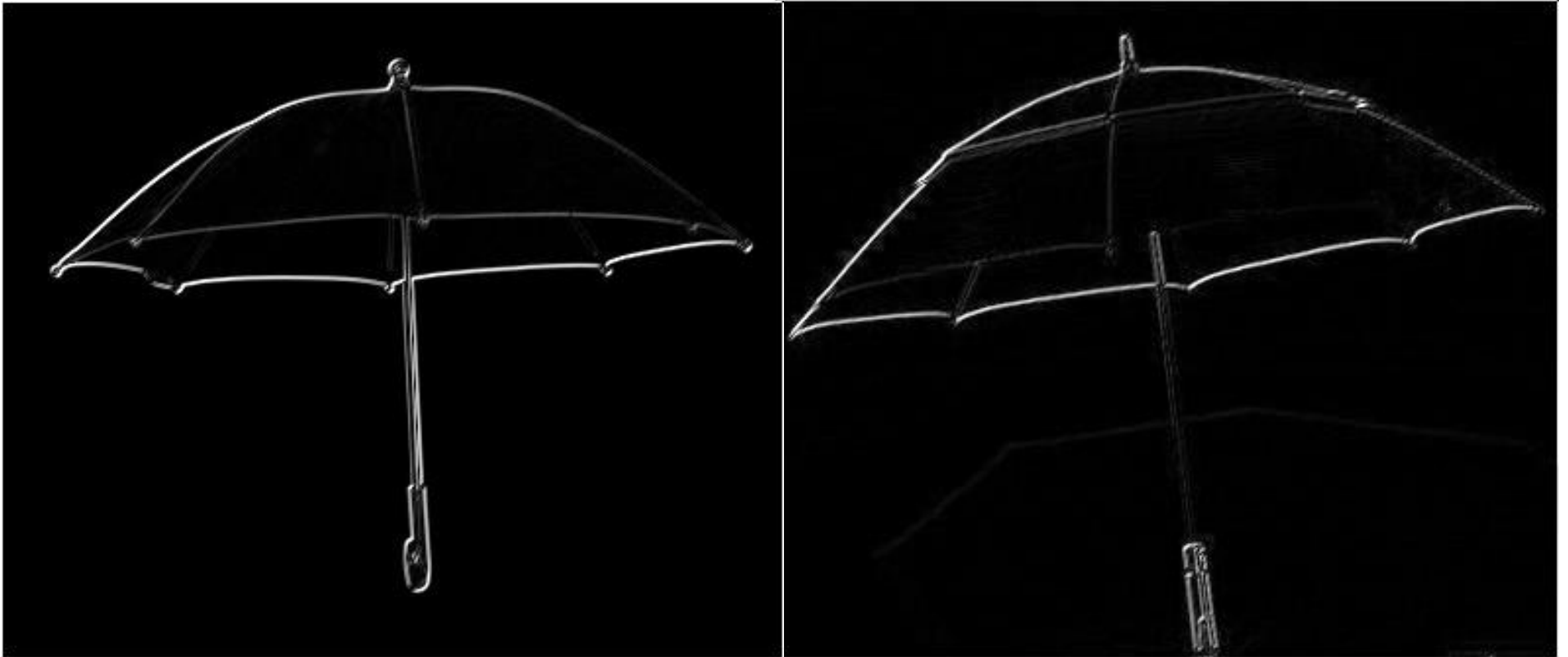
- First content based image retrieval system
 - Query by image content (QBIC)
 - IBM 1995
 - QBIC interprets the virtual canvas as a grid of coloured areas, then matches this grid to other images stored in the database.

Color is not always enough!



The representation of these two umbrellas should be similar....
Under a color based representation they look completely different!

What next?



Edges! But what are they & how do we find them?

Reminder: Convolution

Filtering

Alternatively you can convolve the input signal with a filter to get frequency limited output signal.

Convolution:

$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m] \cdot g[n - m] \quad (\text{convolution } \a href="#">\text{demo})$$

signal

f =

1	3	2	5	3	2	4	5
---	---	---	---	---	---	---	---

filter

g =

1/4	1/2	1/4
-----	-----	-----

Filtering

Alternatively you can convolve the input signal with a filter to get frequency limited output signal.

Convolution:

$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m] \cdot g[n - m] \quad (\text{convolution } \a href="#">\text{demo})$$

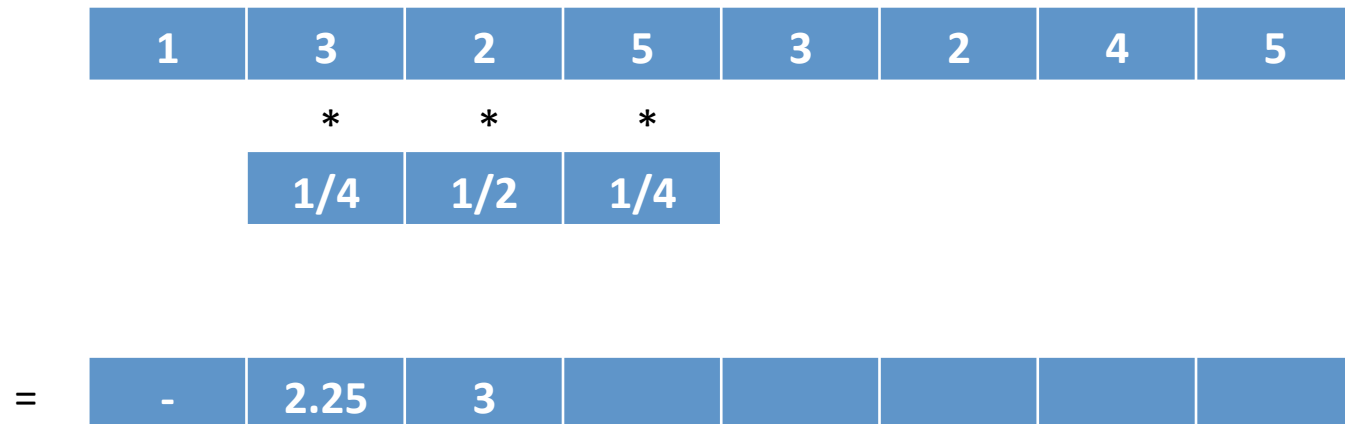
	1	3	2	5	3	2	4	5
	*	*	*					
	1/4	1/2	1/4					
=	-	2.25						

Filtering

Alternatively you can convolve the input signal with a filter to get frequency limited output signal.

Convolution:

$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m] \cdot g[n - m] \quad (\text{convolution } \a href="#">\text{demo})$$



Filtering

Alternatively you can convolve the input signal with a filter to get frequency limited output signal.

Convolution:

$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m] \cdot g[n - m] \quad (\text{convolution } \a href="#">\text{demo})$$

	1	3	2	5	3	2	4	5
			*	*	*			
			1/4	1/2	1/4			
=	-	2.25	3	3.75				

Filtering

Alternatively you can convolve the input signal with a filter to get frequency limited output signal.

Convolution:

$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m] \cdot g[n - m] \quad (\text{convolution } \a href="#">\text{demo})$$

	1	3	2	5	3	2	4	5
				*	*	*		
				1/4	1/2	1/4		
=	-	2.25	3	3.75	3.25			

Filtering

Alternatively you can convolve the input signal with a filter to get frequency limited output signal.

Convolution:

$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m] \cdot g[n - m] \quad (\text{convolution } \a href="#">\text{demo})$$

	1	3	2	5	3	2	4	5
					*	*	*	
					1/4	1/2	1/4	
=	-	2.25	3	3.75	3.25	2.75		

Filtering

Alternatively you can convolve the input signal with a filter to get frequency limited output signal.

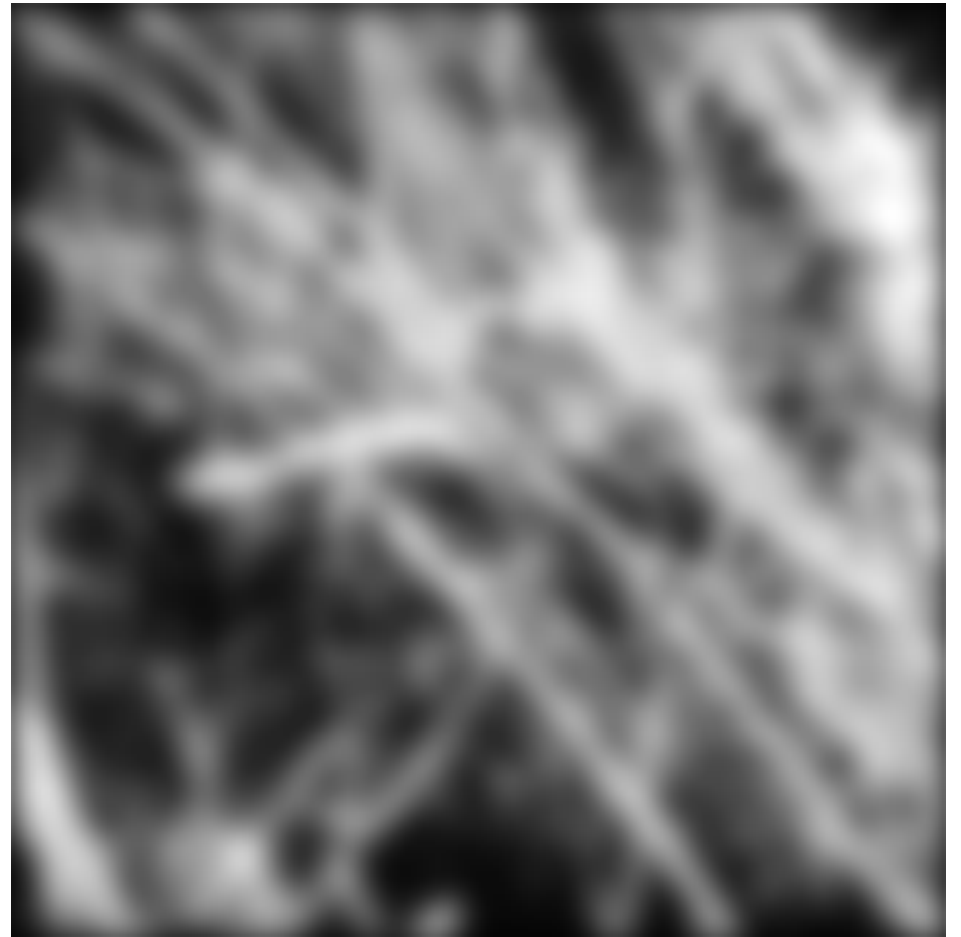
Convolution:

$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m] \cdot g[n - m] \quad (\text{convolution } \a href="#">\text{demo})$$

Convolution computes a weighted average.

	1	3	2	5	3	2	4	5
						*	*	*
						1/4	1/2	1/4
=	-	2.25	3	3.75	3.25	2.75	3.75	-

Images -> 2d filtering



Moving average

- Let's replace each pixel with a *weighted* average of its neighborhood
- The weights are called the *filter kernel*
- What are the weights for a 3x3 unweighted moving average?

Moving average

- Let's replace each pixel with a *weighted* average of its neighborhood
- The weights are called the *filter kernel*
- What are the weights for a 3x3 unweighted moving average?

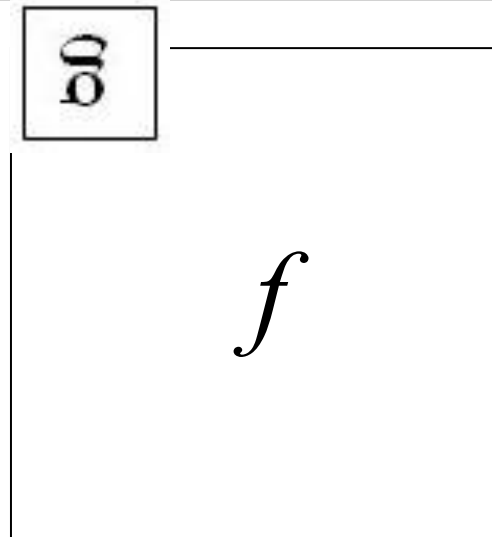
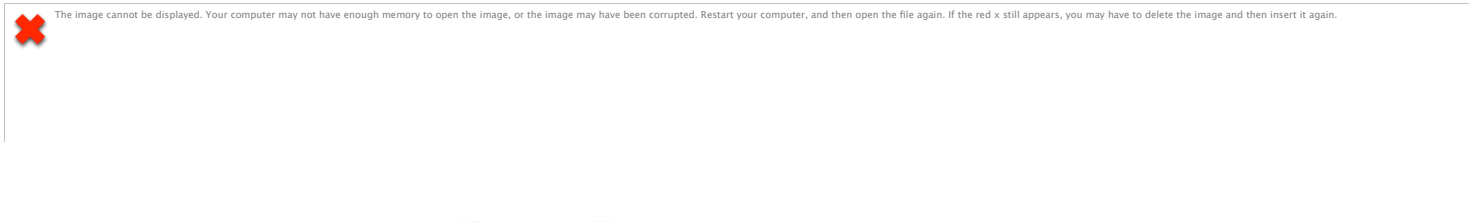
$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

“box filter”

Defining convolution in 2d

- Let f be the image and g be the kernel. The output of convolving f with g is denoted $f * g$.



- Convention: kernel is “flipped”
- MATLAB: conv2 vs. filter2 (also imfilter)

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x,y)$

1	1	1
1	1	1
1	1	1

$\frac{1}{9}$

“box filter”

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0								

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10							

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20						

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30					

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30				

Moving Average In 2D

$$F[x, y]$$

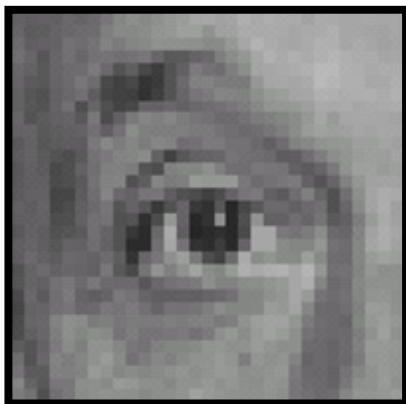
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

What is this filter doing?

Practice with linear filters

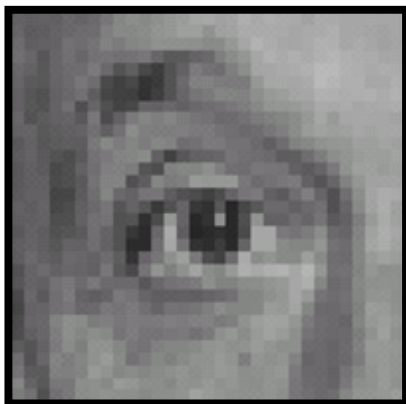


Original

0	0	0
0	1	0
0	0	0

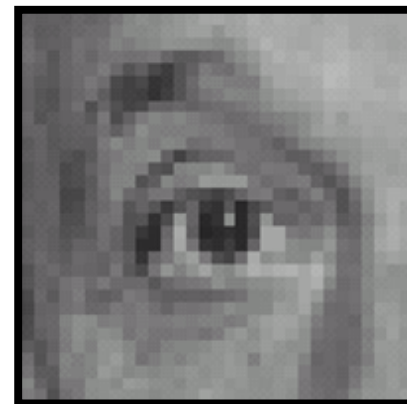
?

Practice with linear filters



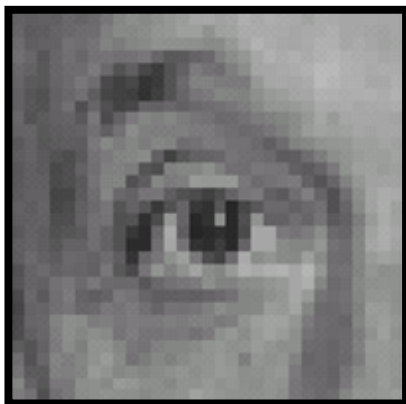
Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Practice with linear filters

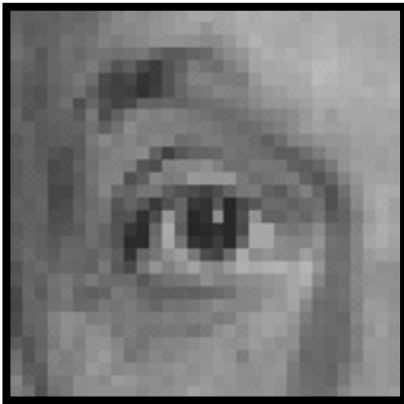


Original

0	0	0
0	0	1
0	0	0

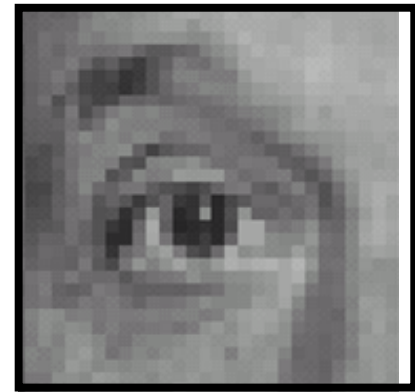
?

Practice with linear filters



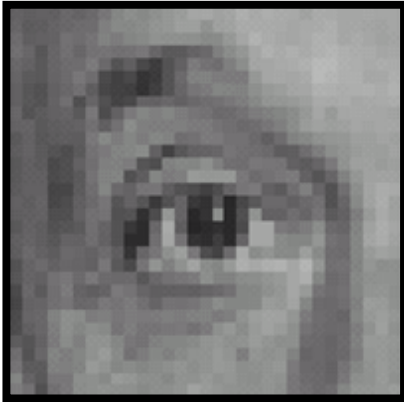
Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Practice with linear filters



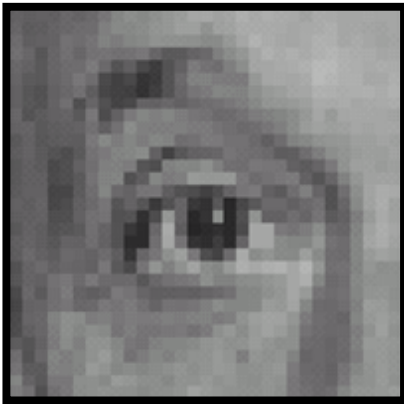
Original

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

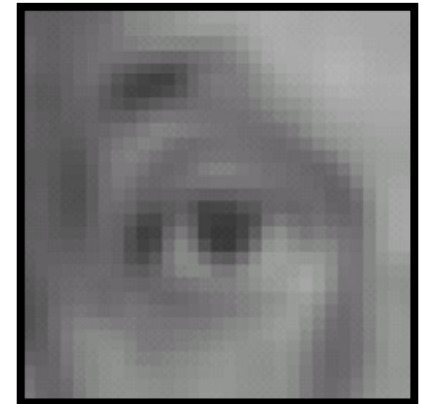
?

Practice with linear filters



Original

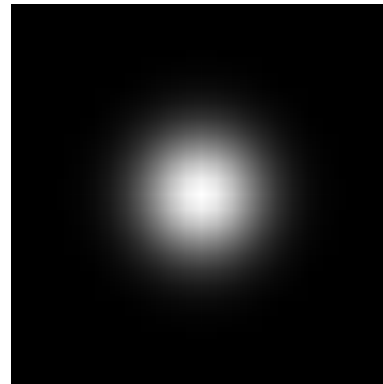
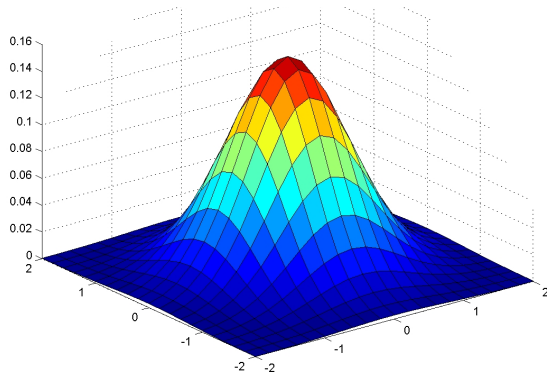
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Blur (with a
box filter)

Gaussian Kernel

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

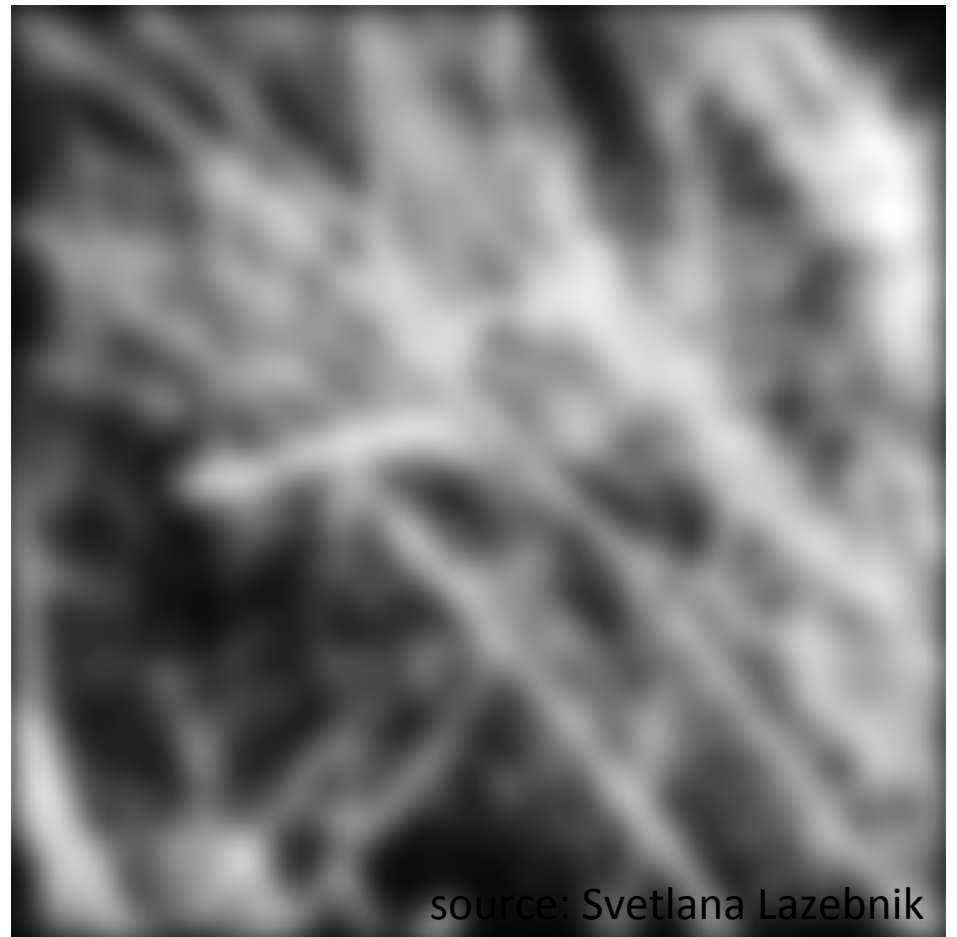
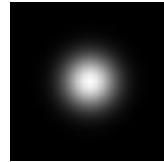


0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

- Constant factor at front makes volume sum to 1 (can be ignored, as we should re-normalize weights to sum to 1 in any case)

Example: Smoothing with a Gaussian



source: Svetlana Lazebnik

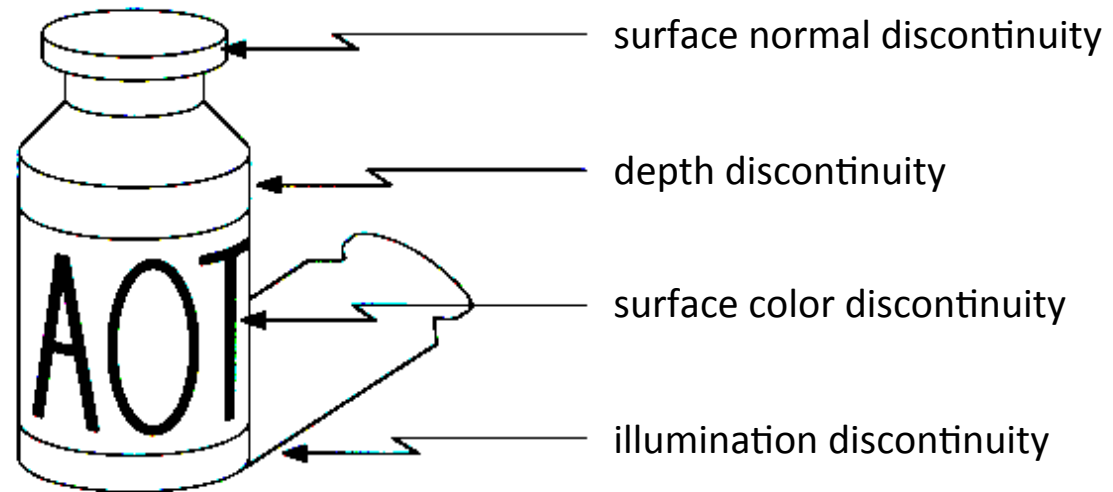
Edges

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
 - Intuitively, most semantic and shape information from the image can be encoded in the edges
 - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)



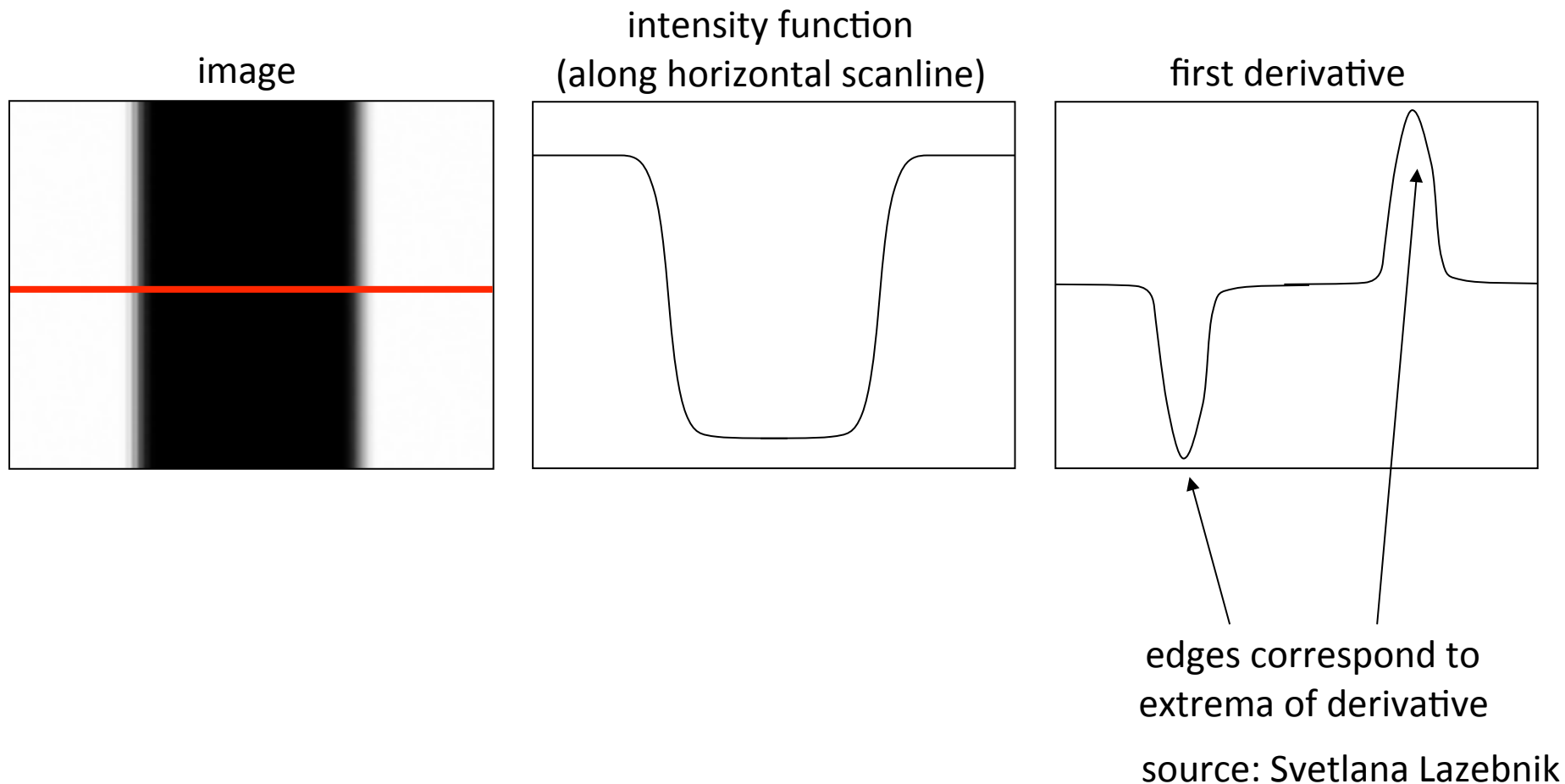
Origin of Edges



Edges are caused by a variety of factors

Characterizing edges

- An edge is a place of rapid change in the image intensity function



Edge filters

Approximations of derivative filters:

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Convolve filter with image to get edge map

Edge filters

Approximations of derivative filters:

Prewitt:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} ; M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Sobel:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} ; M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Roberts:

$$M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} ; M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Respond highly to vertical edges

Edge filters

Approximations of derivative filters:

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Respond highly to horizontal edges

Source: K. Grauman

Edges: example

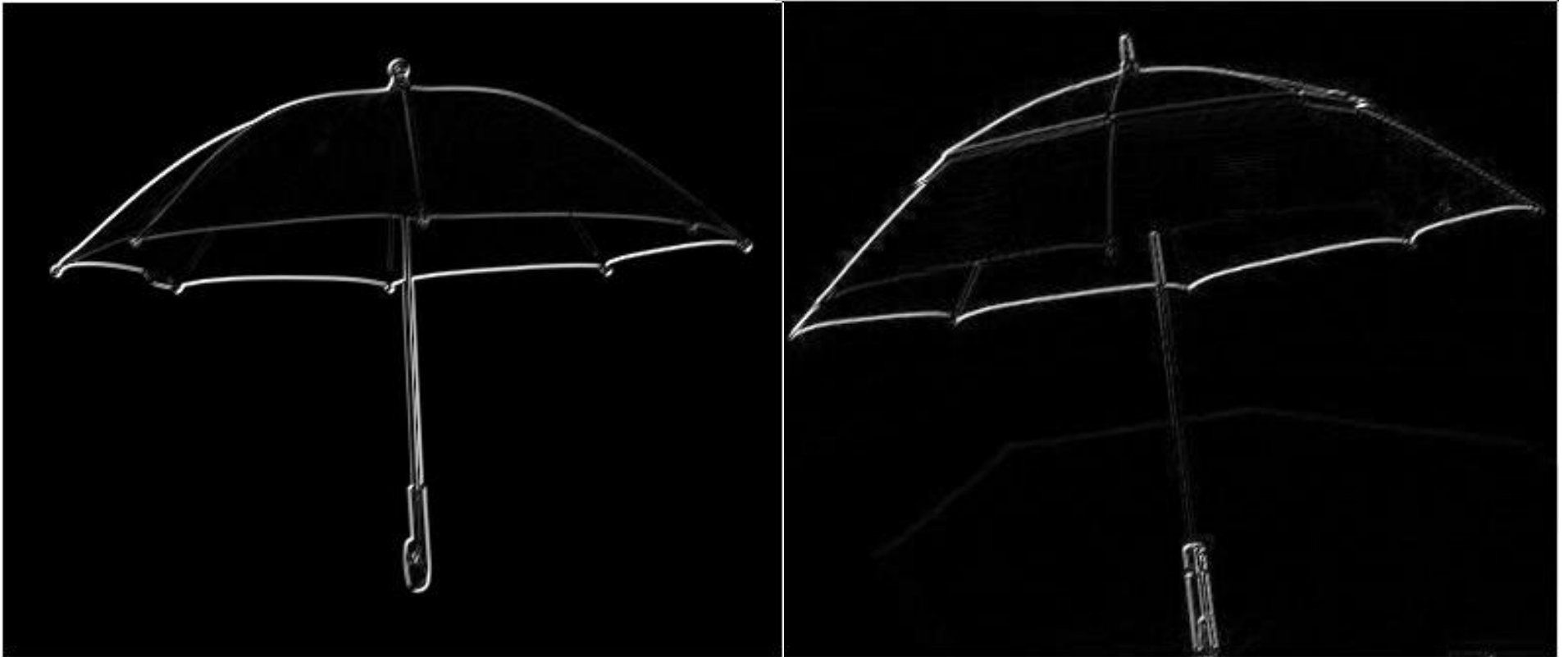
What about our umbrellas?



The representation of these two umbrellas should be similar....
Under a color based representation they look completely different!
How about using edges?

Matlab Demo 4

Edges



Red umbrella

Gray umbrella

Edges extracted using convolution with Prewitt filter

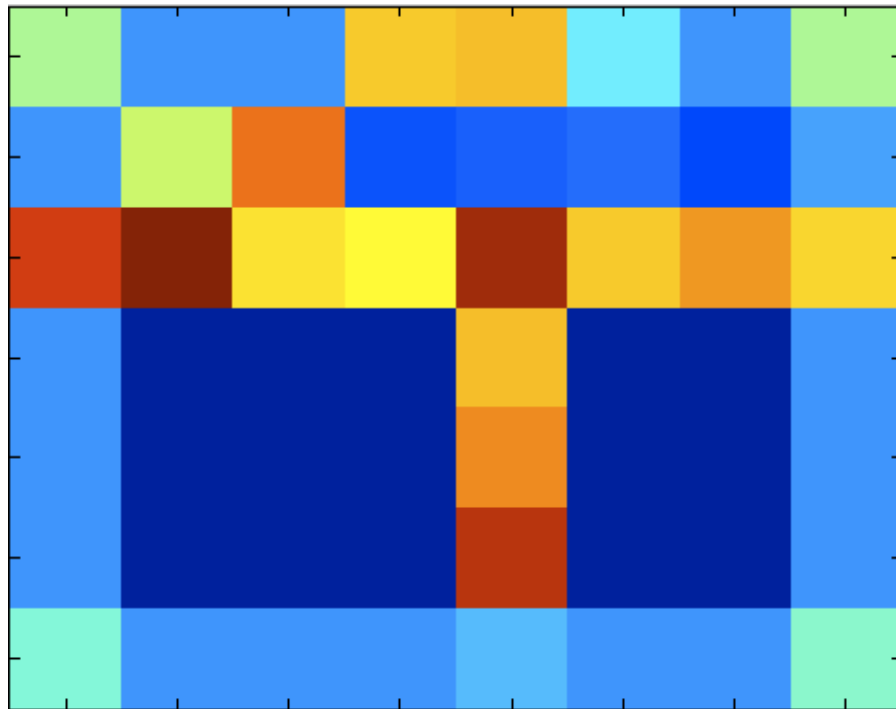
Edges



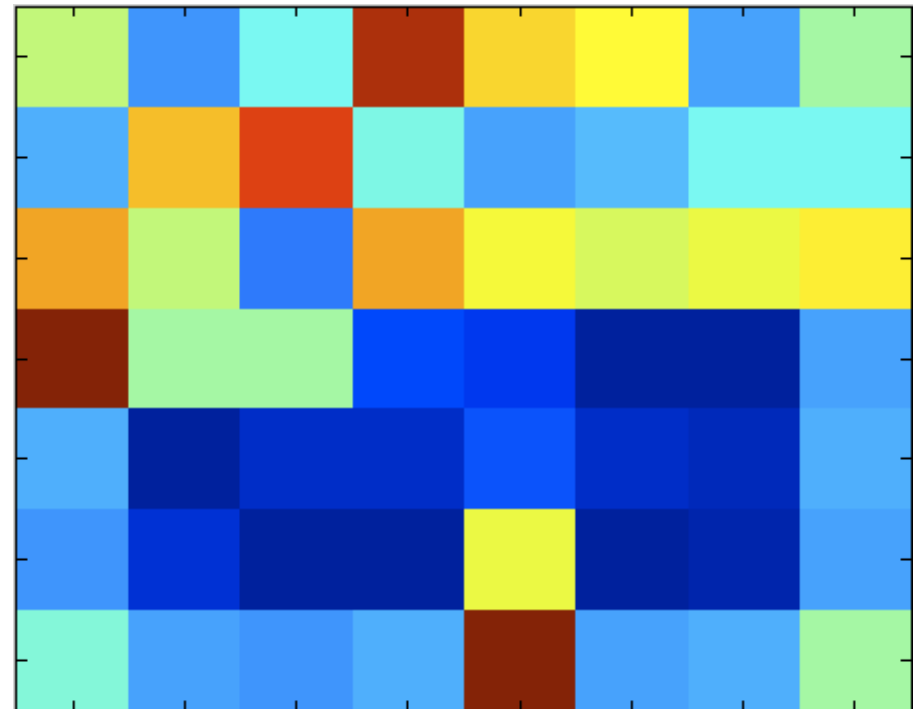
Edges overlaid from red and gray umbrellas.

How is this?

Edge Energy in Spatial Grid



Red Umbrella

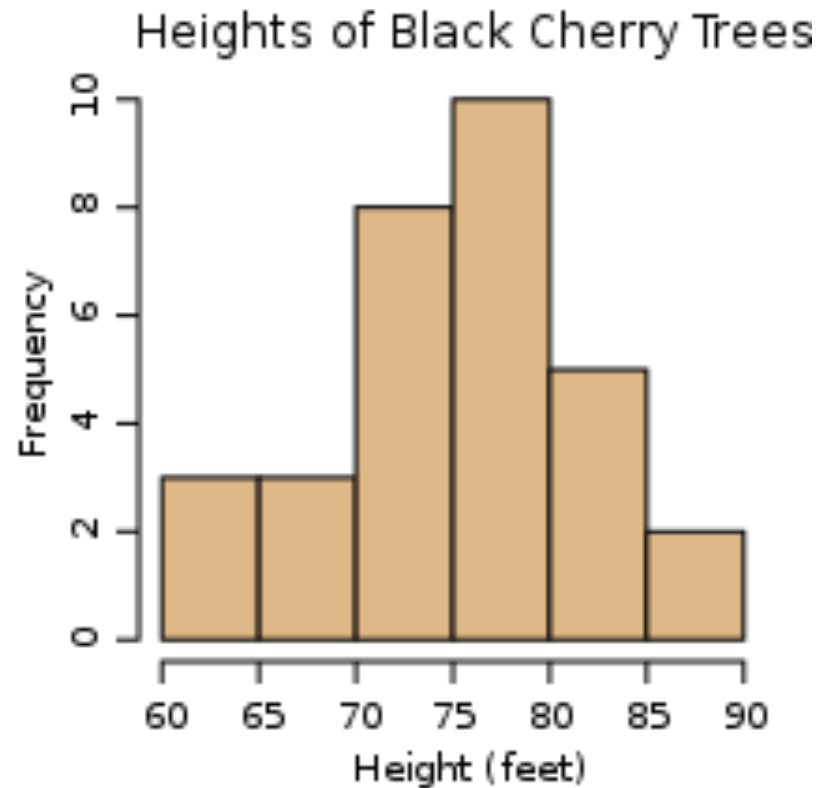


Gray Umbrella

How is this representation?

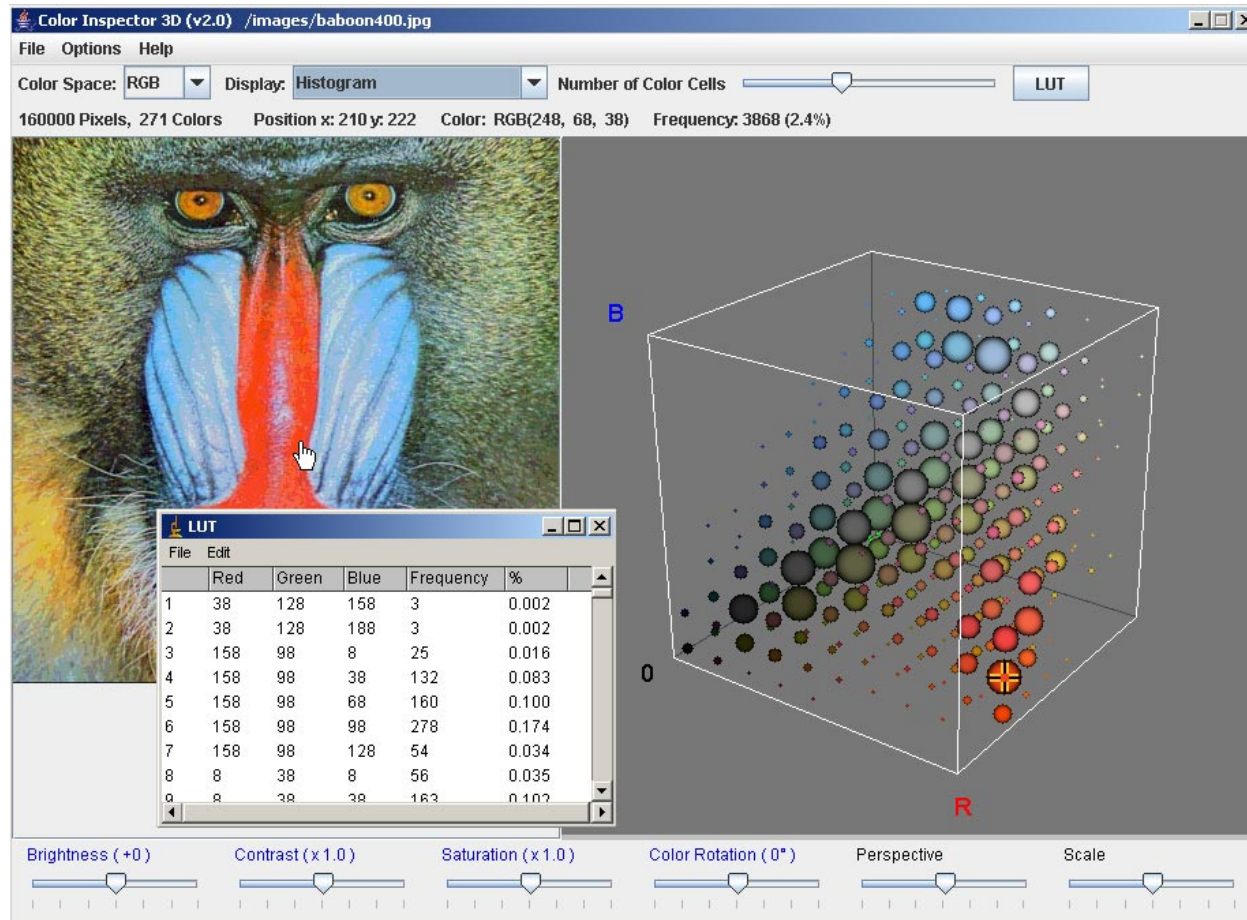
Quick overview of other common kinds of Features

Important concept: Histograms



Graphical display of tabulated frequencies, shown as bars. It shows what proportion of cases fall into each of several categories. The categories are usually specified as non-overlapping intervals of some variable.

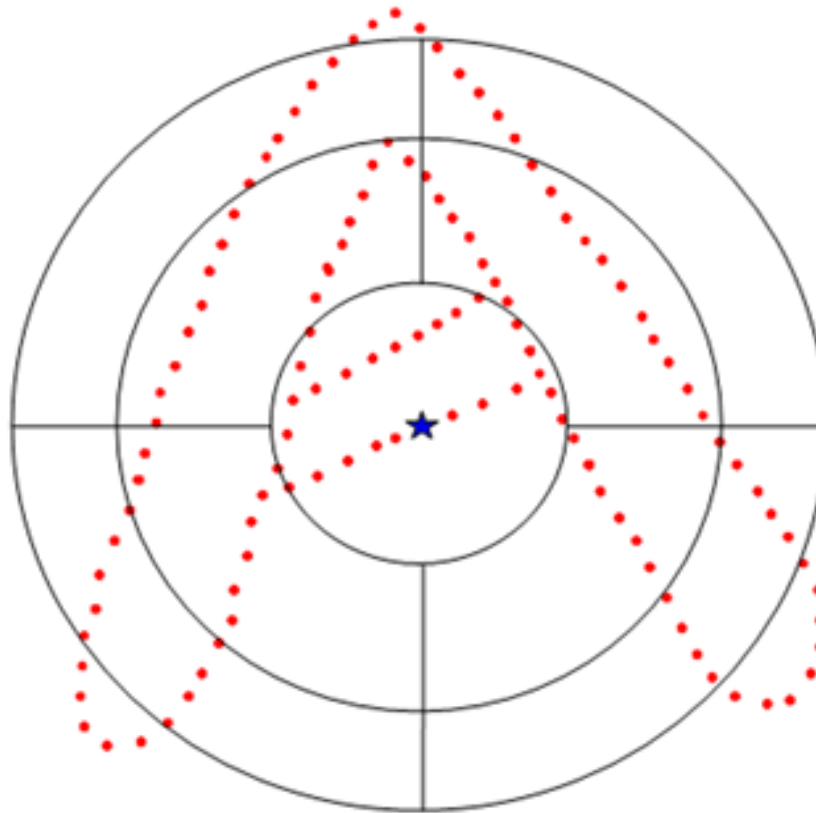
Color Histograms



Representation of the distribution of colors in an image, derived by counting the number of pixels of each of given set of color ranges in a typically (3D) color space (RGB, HSV etc).

What are the bins in this histogram?

Shape Descriptors: shape context

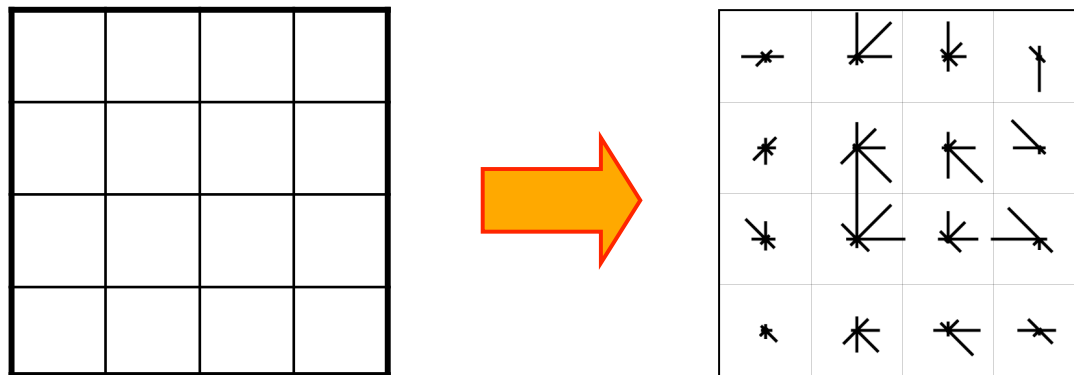


Representation of the local shape around a feature location (star) – as histogram of edge points in an image relative to that location. Computed by counting the edge points in a log polar space.

So what are the bins of this histogram?

Shape descriptors: SIFT

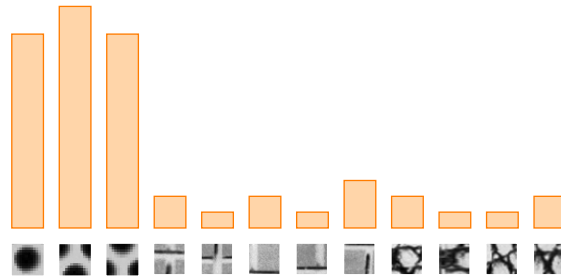
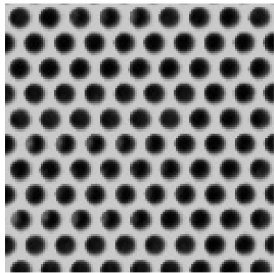
- Descriptor computation:
 - Divide patch into 4x4 sub-patches
 - Compute histogram of gradient orientations (convolve with filters that respond to edges in different directions) inside each subpatch
 - Resulting descriptor: $4 \times 4 \times 8 = 128$ dimensions



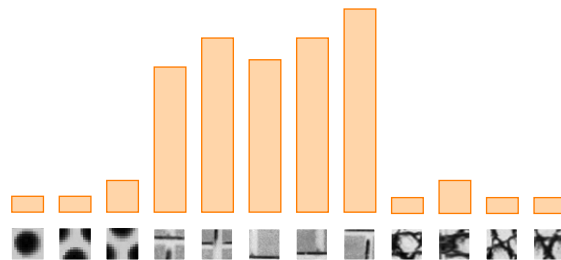
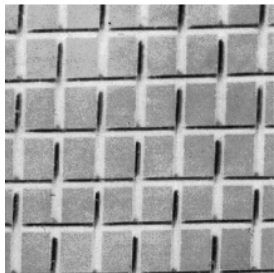
David G. Lowe. ["Distinctive image features from scale-invariant keypoints."](#) *IJCV* 60 (2), pp. 91-110, 2004.

source: Svetlana Lazebnik

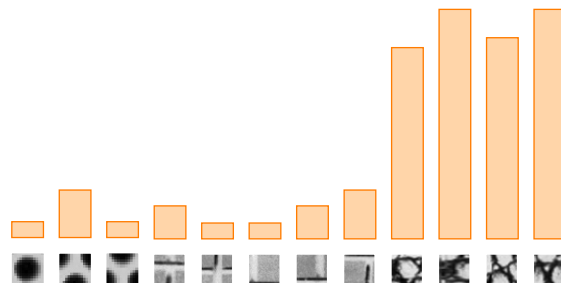
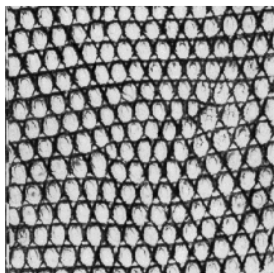
Texture Features



Convolve with various filters – spot, oriented bar.



Compute histogram of responses.



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003