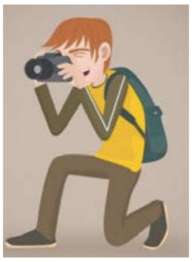
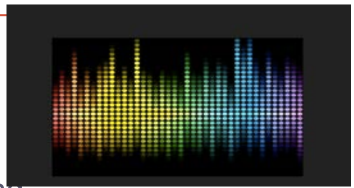


DIGITAL IMAGE PROCESSING



Lecture 3

Filters cont., Scale-space, Colors

Tammy Riklin Raviv

Electrical and Computer Engineering

Ben-Gurion University of the Negev

Last Class: Filtering

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

Linear filtering is a weighted sum/difference of pixel values

- Can smooth, sharpen, translate (among many other uses)
- Filtering in Matlab, e.g. to filter image f with h

Option 1:

```
g = filter2( h, f );
```

h=filter f=image

```
e.g. h = fspecial('gaussian');
```

Filtering in Matlab

Option 2

imfilter - N-D filtering of multidimensional images

I_h = imfilter(I, h)

filters the multidimensional array **I** with the multidimensional filter **h** and returns the result in **I_h**.

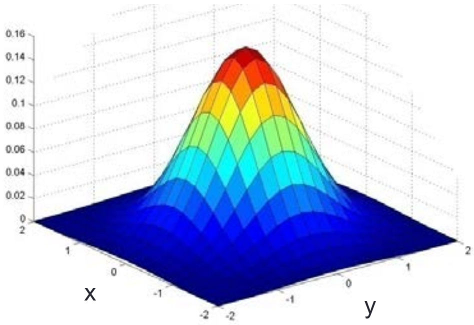
You optionally can filter a multidimensional array with a 2-D filter using a GPU

Filtering in Matlab

- h= fspecial(type)
- h = fspecial('average',hsize)
- h = fspecial('disk',radius)
- h = fspecial('gaussian',hsize,sigma)
- Use imgaussfilt or imgaussfilt3 instead.
- h = fspecial('laplacian',alpha)
- h = fspecial('log',hsize,sigma)
- h = fspecial('motion',len,theta)
- h = fspecial('prewitt') $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
- h = fspecial('sobel') $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

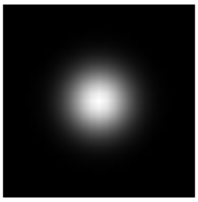
Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness



	x				
	0.003	0.013	0.022	0.013	0.003
	0.013	0.059	0.097	0.059	0.013
y	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, σ = 1

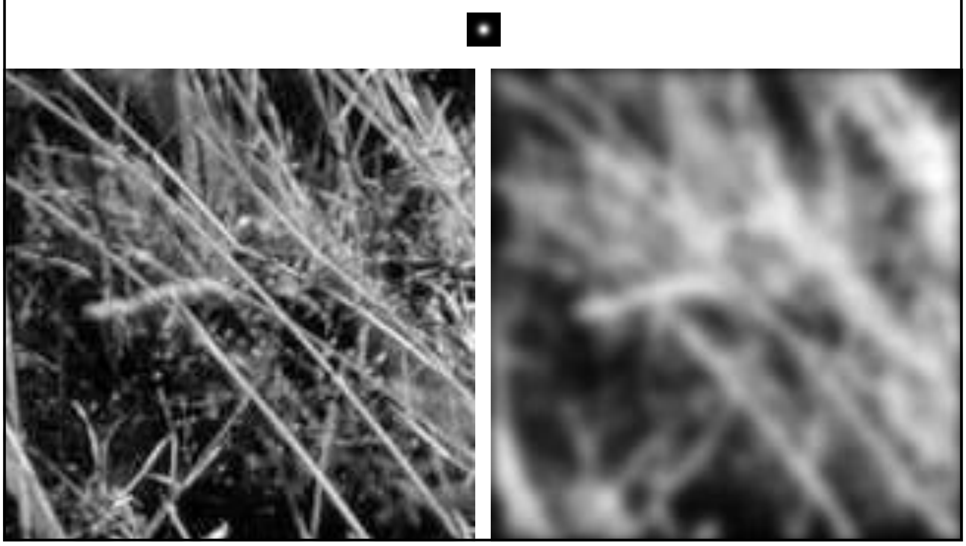


$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

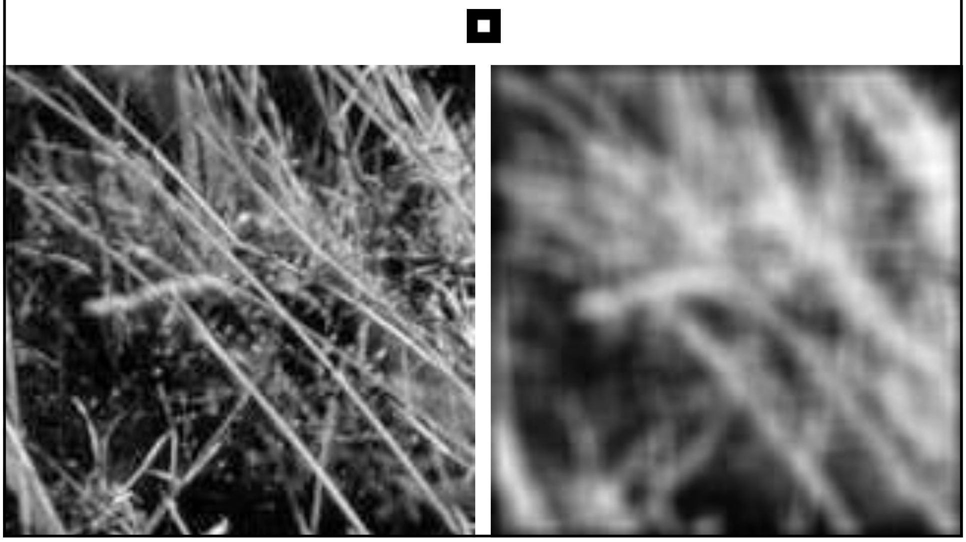
James Hays

Smoothing with Gaussian filter



James Hay

Smoothing with box filter



Gaussian filters

- Remove “high-frequency” components from the image (low-pass filter)
  - Images become more smooth
- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

Separability of the Gaussian filter

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

$$= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}}\right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}}\right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of *x* and the other a function of *y*

In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe

Separability example

2D convolution (center location only)

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 5 & 5 \\ \hline 4 & 4 & 6 \\ \hline \end{array}$$

65

The filter factors into a product of 1D filters:

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} \times \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

Perform convolution along rows:

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 5 & 5 \\ \hline 4 & 4 & 6 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline & 11 & \\ \hline & 18 & \\ \hline & 18 & \\ \hline \end{array}$$

Followed by convolution along the remaining column:

$$\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline & 11 & \\ \hline & 18 & \\ \hline & 18 & \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline & & \\ \hline & 65 & \\ \hline & & \\ \hline \end{array}$$

Source: K. Grauman

Separability

- Why is separability useful in practice?

Separability

- Why is separability useful in practice?
- If K is width of convolution kernel:
  - 2D convolution = K^2 multiply-add operations
  - 2x 1D convolution: 2K multiply-add operations

Partial Derivatives

First order partial derivatives:

$$\frac{\partial I(x,y)}{\partial x} = I(x+1,y) - I(x,y)$$ Left Derivative

$$\frac{\partial I(x,y)}{\partial x} = I(x,y) - I(x-1,y)$$ Right Derivative

$$\frac{\partial I(x,y)}{\partial x} = \frac{I(x+1,y) - I(x-1,y)}{2}$$ Central Derivative

Partial Derivatives

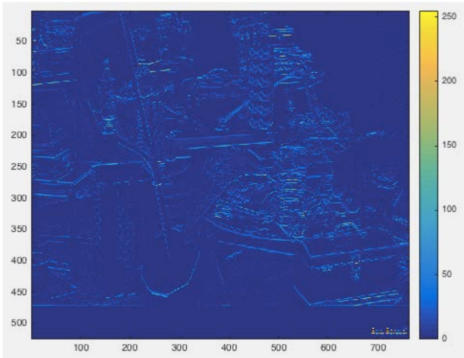
First order partial derivatives:

$$\frac{\partial I(x,y)}{\partial y} = I(x,y+1) - I(x,y)$$

$$\frac{\partial I(x,y)}{\partial y} = I(x,y) - I(x,y-1)$$

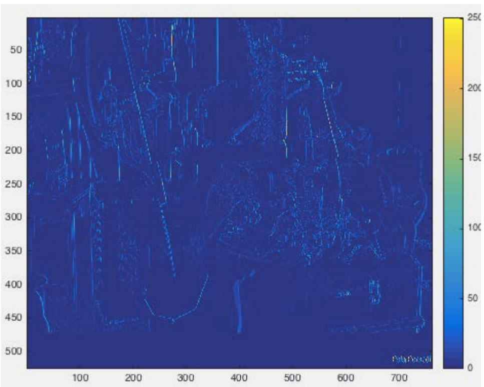
$$\frac{\partial I(x,y)}{\partial y} = \frac{I(x,y+1) - I(x,y-1)}{2}$$

Partial Derivatives



```
>> myFirstImage = imread('someImage.png');
>> I = myFirstImage(:,:,1);
>> Ir = I(2:end,:);
>> Il = I(1:end-1,:);
>> Idx = Il-Ir;
>> figure;imagesc(Idx);colorbar
>> figure;imagesc(abs(Idx));colorbar
```

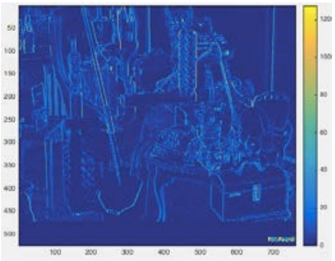
Partial Derivatives

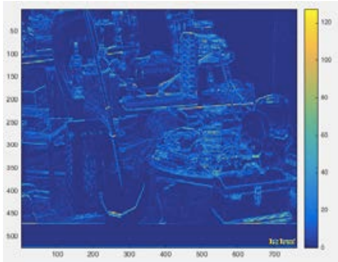



```
>>
>> figure;imagesc(diff(I,1));colorbar
>> figure;imagesc(diff(I,2));colorbar
```

```
>> Iu = I(:,1:end-1);
>> Id = I(:,2:end);
>> Idy = Iu-Id;
>> figure;imagesc(abs(Idy));colorbar
```

Gradients





Can you explain the differences?

```
>> [Igx,Igy] = gradient(double(I));
>> figure;imagesc(abs(Igx));colorbar
>> figure;imagesc(abs(Igy));colorbar
>>
```

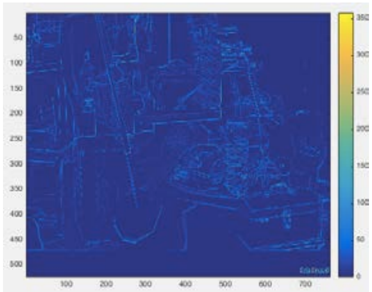
Gradients

Gradients:

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

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

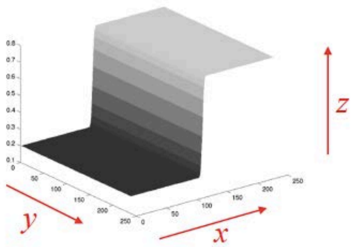
Gradients



```
>> G = sqrt(double(Idx).^2+double(Idy).^2);
>> figure;imagesc(G);colorbar
```

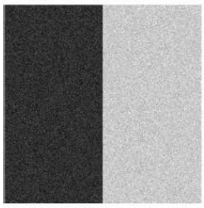
Computing Derivatives using Linear Filters

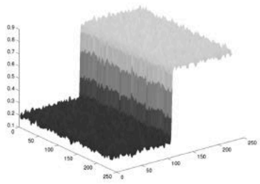
Think of the image as a surface with $z = f(x,y)$



Computing Derivatives using Linear Filters

Objective: compute gradient $\nabla f(x,y) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})$ of the image "surface"





- e.g. as a method to find the edge in the image
- there is the problem of noise ….

Computing Derivatives using Linear Filters

1D

Want $f'(x)$, use central difference

$$f'(i) = \frac{f_{i+1} - f_{i-1}}{2}$$

which is equivalent to the molecule $[-\frac{1}{2}, 0, \frac{1}{2}]$.

For a noisy signal the derivative amplifies the noise.

Solution?

Computing Derivatives using Linear Filters

1D

Want $f'(x)$, use central difference

$$f'(i) = \frac{f_{i+1} - f_{i-1}}{2}$$

which is equivalent to the molecule $[-\frac{1}{2}, 0, \frac{1}{2}]$.

For a noisy signal the derivative amplifies the noise.

Solution

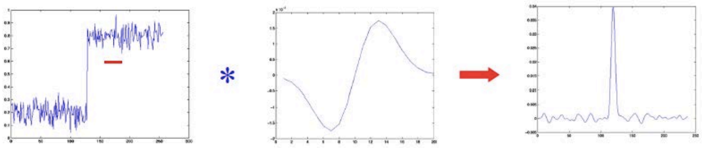
- smooth with a Gaussian filter
- then differentiate

Computing Derivatives using Linear Filters

Differentiate smoothed signal $G(x) * f(x)$

$$\begin{aligned}\frac{d(G(x) * f(x))}{dx} &= G(x) * \frac{df(x)}{dx} \\ &= \frac{dG(x)}{dx} * f(x) \\ &= \left(\frac{-x}{\sqrt{2\pi}\sigma^3} e^{-x^2/2\sigma^2}\right) * f(x)\end{aligned}$$

Convolution with a derivative of Gaussian filter



e.g. for $\sigma = 1$ the molecule is
$[-0.0133, -0.1080, -0.2420, 0.0000, 0.2420, 0.1080, 0.0133]$.

Computing Derivatives using Linear Filters

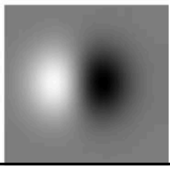
2D

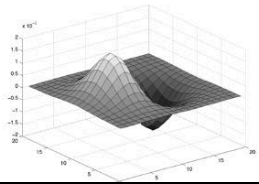
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-r^2/2\sigma^2}$$

$$\frac{\partial}{\partial x} G(x,y) * f(x,y) = \left(\frac{-x}{2\pi\sigma^4} e^{-(x^2+y^2)/2\sigma^2} \right) * f(x,y)$$

$$= \left(\frac{-x}{\sqrt{2\pi\sigma^3}} e^{-x^2/2\sigma^2} \right) \times \left(\frac{1}{\sqrt{2\pi\sigma}} e^{-y^2/2\sigma^2} \right) * f(x,y)$$

Filtering with a 1D derivative of Gaussian filter in x and a 1D Gaussian filter in y – it is a separable filter

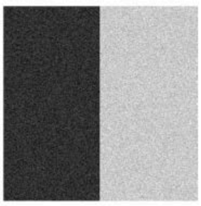


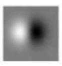


Computing Derivatives using Linear Filters

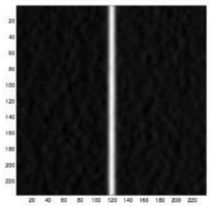
Example

filter with x and y derivatives of Gaussian to obtain directional image derivatives



$*$ 

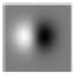
$$I_x = G_x(x,y) * I(x,y)$$



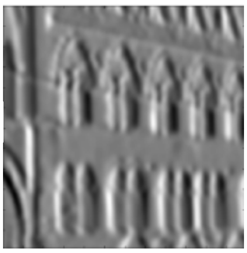
Computing Derivatives using Linear Filters

x-deriv

*

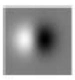


Computing Derivatives using Linear Filters

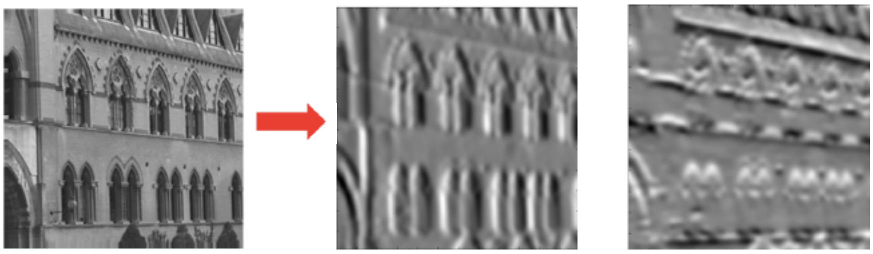


x-deriv

*

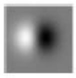


Computing Derivatives using Linear Filters



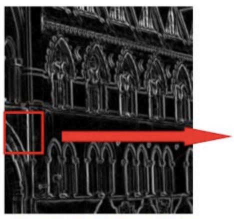
x-deriv

y-deriv

$*$ 

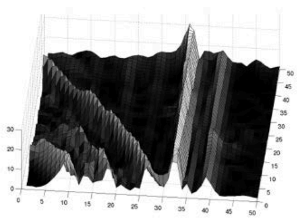
Computing Derivatives using Linear Filters

image



gradient magnitude

$|\nabla(G(x,y) * I(x,y))|$



Derivatives & the Laplacian

- Second order derivatives

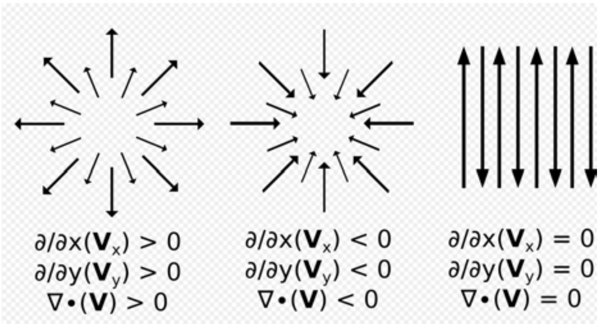
$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

$$\frac{\partial^2 I}{\partial x^2} = I(x+1,y) + I(x-1,y) - 2I(x,y)$$

$$\frac{\partial^2 I}{\partial y^2} = I(x,y+1) + I(x,y-1) - 2I(x,y)$$

$$\nabla^2 I = I(x+1,y) + I(x-1,y) + I(x,y+1) + I(x,y-1) - 4I(x,y)$$

Divergence



divergence is a vector operator that operates on a vector field, producing a scalar field

$$\nabla \cdot (\mathbf{V}(x,y)) = \frac{\partial\, \mathbf{V}_x(x,y)}{\partial x} + \frac{\partial\, \mathbf{V}_y(x,y)}{\partial y}$$

Back to Laplacian

The Laplacian of a scalar function or functional expression is the divergence of the gradient of that function or expression:

$$\Delta I = \nabla \cdot (\nabla I)$$

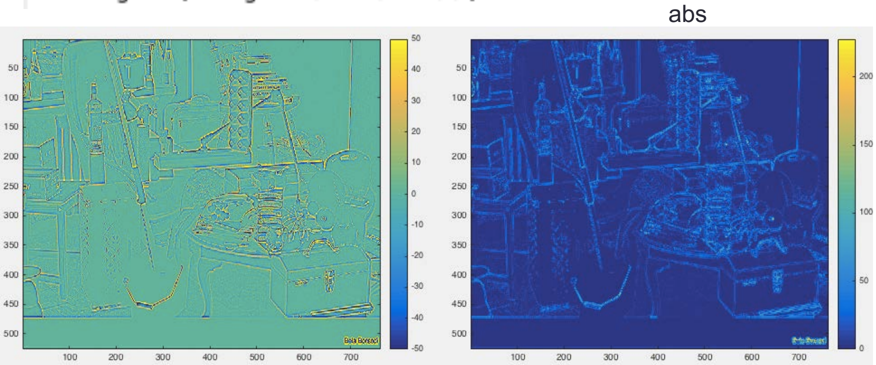
Therefore, you can compute the Laplacian using the divergence and gradient functions:

```
>> [Igx,Igy] = gradient(double(I));
>> div = divergence(Igx,Igy);
>> figure;imagesc(div);colorbar
>> figure;imagesc(abs(div));colorbar
```

Back to Laplacian $\Delta I = \nabla \cdot (\nabla I)$

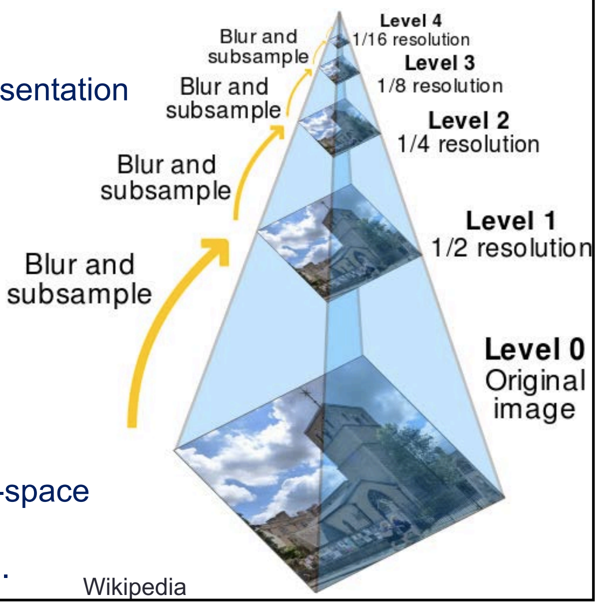
```
>> [Igx,Igy] = gradient(double(I));
>> div = divergence(Igx,Igy);
>> figure;imagesc(div);colorbar
>> figure;imagesc(abs(div));colorbar
```

abs



Pyramids

Multi-scale signal representation

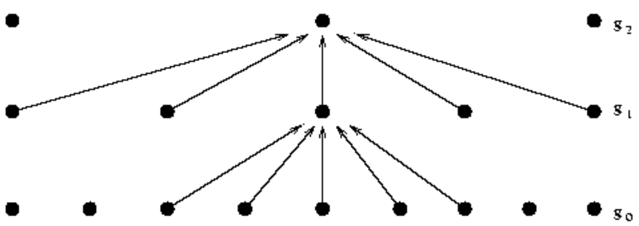


A predecessor to scale-space representation and multiresolution analysis.

Wikipedia

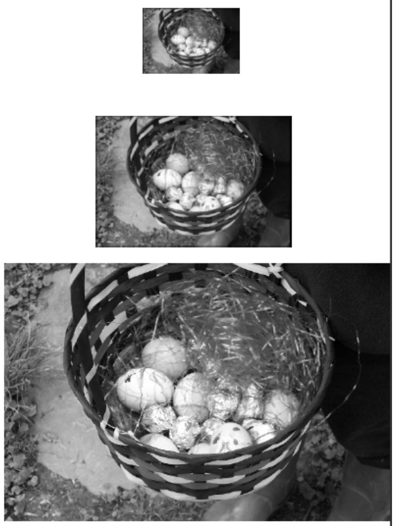
Gaussian Pyramid

The Gaussian Pyramid is a hierarchy of low-pass filtered versions of the original image, such that successive levels correspond to lower frequencies.



Gaussian Pyramids

- Algorithm:
  - 1. Filter with $\mathcal{G}(\sigma = 1)$
  - 2. Resample at every other pixel
  - 3. Repeat



Laplacian Pyramid

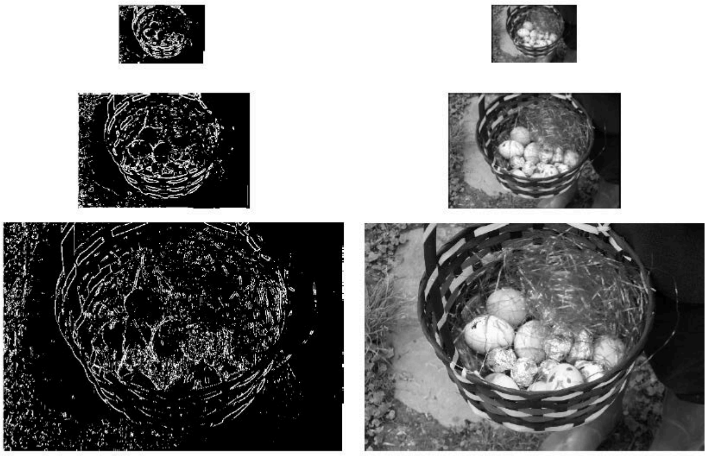
The Laplacian Pyramid is a decomposition of the original image into a hierarchy of images such that each level corresponds to a different band of image frequencies. This is done by taking the difference of levels in the Gaussian pyramid.

For image *I* the Laplacian pyramid *L*(*I*) is:

$$L_i = G_i - \mathrm{expand}(G_{i+1})$$

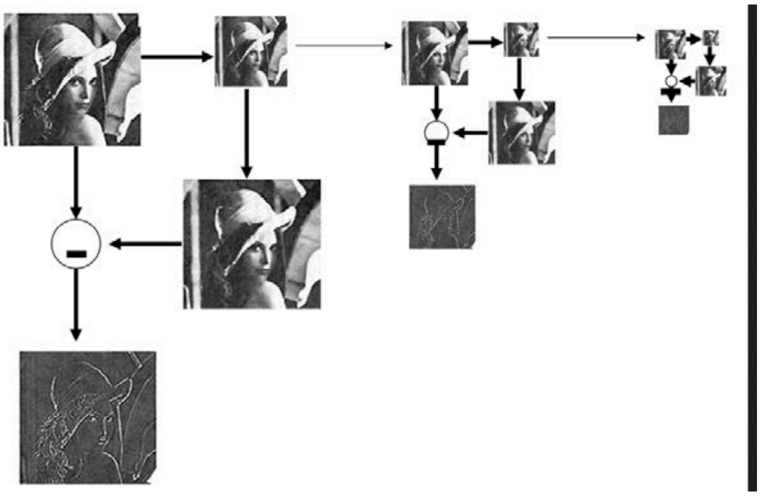
$$L_i = G_i - \mathrm{blur}(G_i)$$

Laplacian Pyramid Algorithm



Source: G Hager Slides

Pyramids Construction



Laplacian Pyramid & Laplacian

The well-known Laplacian derivative operator (isotropic second derivative) is given by

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

For Gaussian kernels, $g(x;\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$,

$$\begin{aligned}
\frac{dg(x;\sigma)}{dx} &= \frac{-x}{\sigma^2} g(x;\sigma) \\
\frac{d^2 g(x;\sigma)}{dx^2} &= \left(\frac{x^2}{\sigma^2} - 1\right) \frac{1}{\sigma^2} g(x;\sigma) \\
\frac{dg(x;\sigma)}{d\sigma} &= \left(\frac{x^2}{\sigma^2} - 1\right) \frac{1}{\sigma} g(x;\sigma)
\end{aligned}$$

http://www.cs.toronto.edu/~jepson/csc320/notes/pyramids.pdf

Therefore

$$\frac{d^2 g(x;\sigma)}{dx^2} = c_0(\sigma) \frac{d\, g(x;\sigma)}{d\sigma} \approx c_1(\sigma)\left(g(x;\sigma) - g(x;\sigma + \Delta\sigma)\right)$$

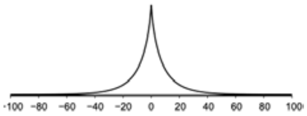
Uses of Laplacian Pyramid: Coding

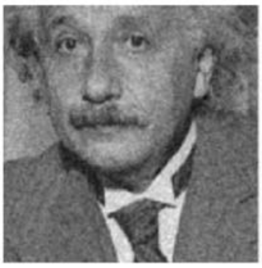
0.03 0.1 0.31 0.81 1.58

bits per pixel

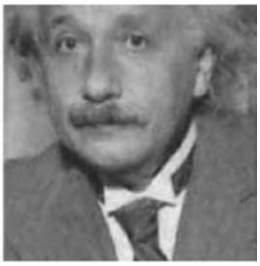
http://www.cs.toronto.edu/~jepson/csc320/notes/pyramids.pdf

Uses of Laplacian Pyramid: Image Restoration (Coring)





Original image + additive noise (SNR = 9dB)



Cored image (SNR = 13.82dB)

http://www.cs.toronto.edu/~jepson/csc320/notes/pyramids.pdf

Uses of Laplacian Pyramid: Image Enhancement

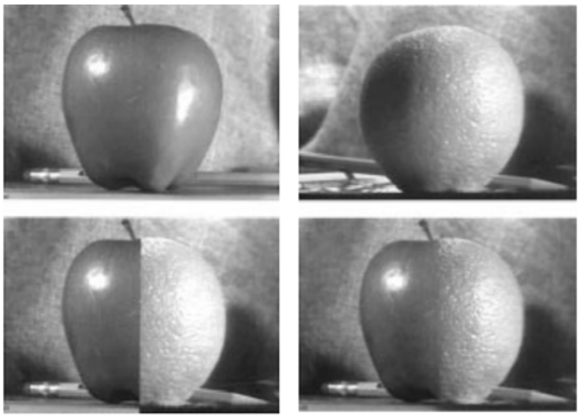
Image 1

Image 2

Composite

http://www.cs.toronto.edu/~jepson/csc320/notes/pyramids.pdf

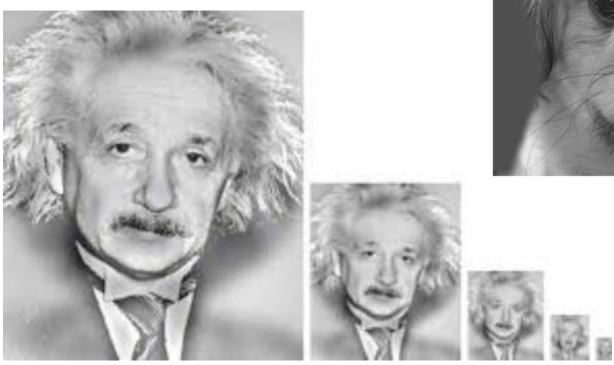
Uses of Laplacian Pyramid: Image Compositing



http://www.cs.toronto.edu/~jepson/csc320/notes/pyramids.pdf

Hybrid Images

Ask about the bonus !



Isotropic Diffusion

The diffusion equation is a general case of the heat equation that describes the density changes in a material undergoing diffusion over time. Isotropic diffusion, in image processing parlance, is an instance of the heat equation as a partial differential equation (PDE), given as:

$$\frac{\partial I}{\partial t} = \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

where, I is the image and *t* is the time of evolution.

Manasi Datar

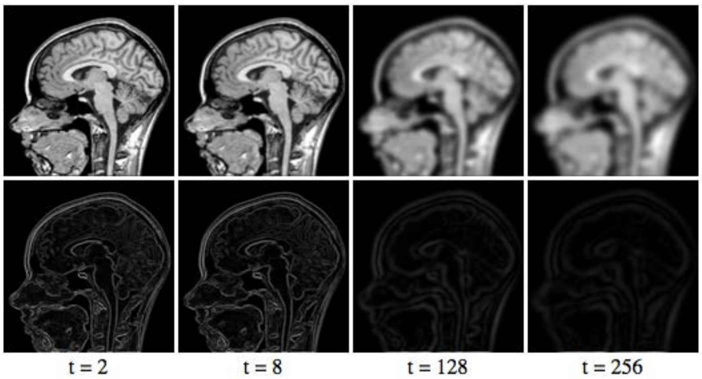
Isotropic Diffusion

Solving this for an image is equivalent to convolution with some Gaussian kernel.

In practice we iterate as follows:

$$I_{i,j}^{t+1} = I_{i,j}^{t} + \lambda \left[I_{i-1,j}^{t} + I_{i+1,j}^{t} + I_{i,j-1}^{t} + I_{i,j+1}^{t} - 4 I_{i,j}^{t} \right]$$

Isotropic Diffusion



We can notice that while the diffusion process blurs the image considerably as the number of iterations increases, the edge information progressively degrades as well.

Anisotropic Diffusion: Perona-Malik

Perona & Malik introduce the flux function as a means to constrain the diffusion process to contiguous homogeneous regions, but not cross region boundaries. The heat equation (after appropriate expansion of terms) is thus modified to:

$$\frac{\partial I}{\partial t} = \operatorname{div}\left(c(x,y,t)\nabla I\right) = \nabla c \cdot \nabla I + c(x,y,t)\Delta I$$

where c is the proposed flux function which controls the rate of diffusion at any point in the image.

Anisotropic Diffusion: Perona-Malik

A choice of *c* such that it follows the gradient magnitude at the point enables us to restrain the diffusion process as we approach region boundaries. As we approach edges in the image, the flux function may trigger inverse diffusion and actually enhance the edges.

Anisotropic Diffusion: Perona-Malik

Perona & Malik suggest the following two flux functions:

$$c\left(||\nabla I||\right) = e^{-(||\nabla I||/K)^2}$$

$$c\left(||\nabla I||\right) = \frac{1}{1 + \left(\frac{||\nabla I||}{K}\right)^2}$$

Anisotropic Diffusion: Perona-Malik

The flux functions offer a trade-off between edge-preservation and blurring (smoothing) homogeneous regions. Both the functions are governed by the free parameter κ which determines the edge-strength to consider as a valid region boundary. Intuitively, a large value of κ will lead back into an isotropic-like solution. We will experiment with both the flux functions.

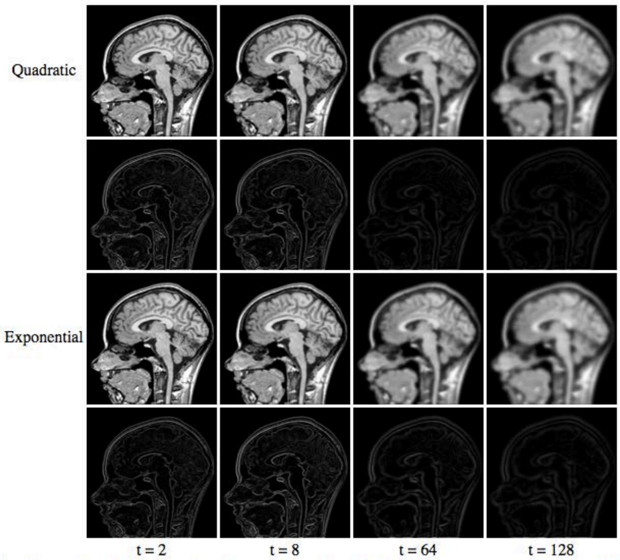
Anisotropic Diffusion: Perona-Malik

A discrete numerical solution can be derived for the anisotropic case as follows:

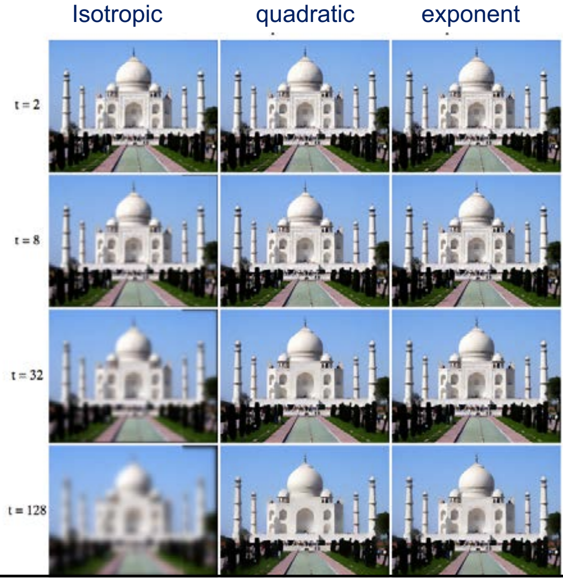
$$I_{i,j}^{t+1} = I_{i,j}^{t} + \lambda \left[c_N \cdot \nabla_N I + c_S \cdot \nabla_S I + c_E \cdot \nabla_E I + c_W \cdot \nabla_W I \right]_{i,j}^{t}$$

where {N,S,W,E} correspond to the pixel above, below, left and right of the pixel under consideration (i,j).

Anisotropic Diffusion: Perona-Malik

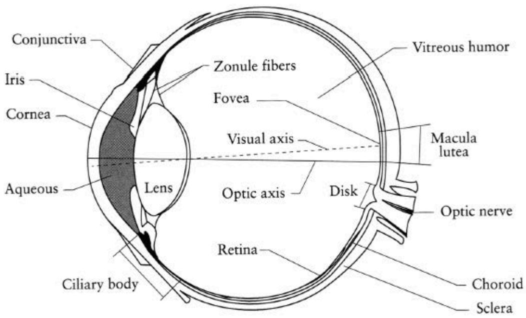


Anisotropic vs. Isotropic Diffusion



Colors

The Eye

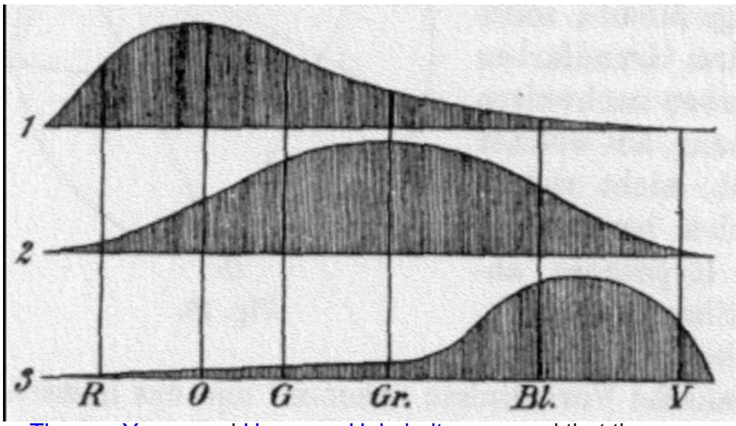


The human eye is a camera

- **Iris** - colored annulus with radial muscles
- **Pupil** - the hole (aperture) whose size is controlled by the iris
- What's the sensor?
  - photoreceptor cells (rods and cones) in the **retina**

Slide by Steve Seitz

Eye receptors



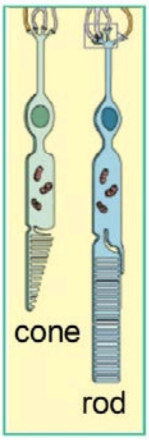
Thomas Young and Hermann Helmholtz assumed that the eye's retina consists of three different kinds of light receptors for red, green and blue. Source: Wikipedia

Eye receptors

Two types of light-sensitive receptors

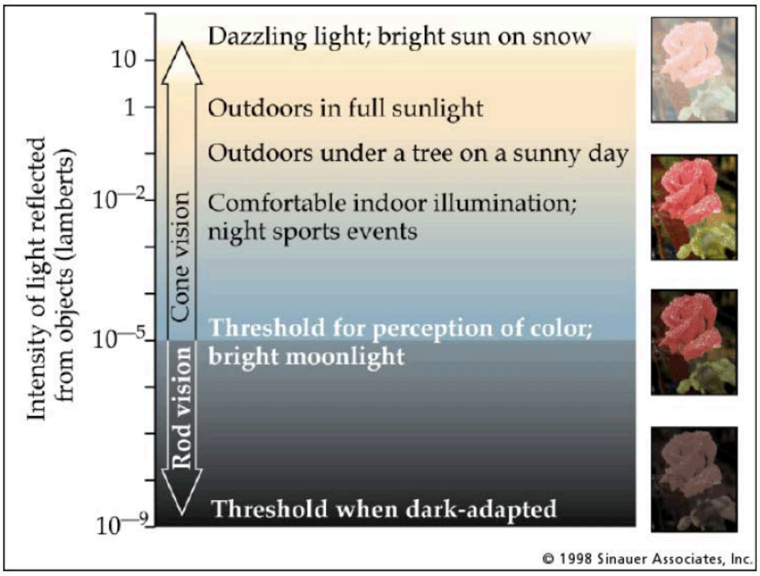
Cones
- cone-shaped
- less sensitive
- operate in high light
- color vision

Rods
- rod-shaped
- highly sensitive
- operate at night
- gray-scale vision

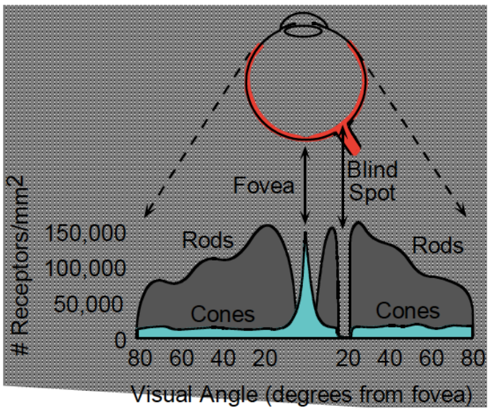


James Hays

Rod & Cone Sensitivity



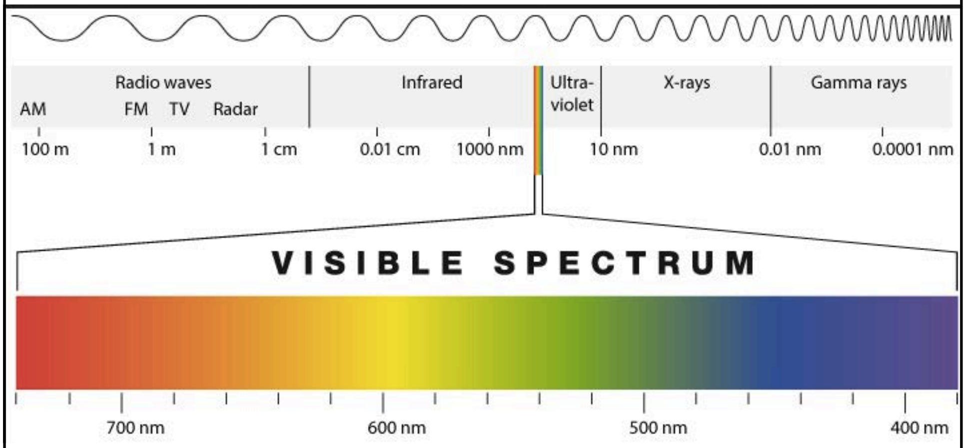
Distribution of Rods & Cones



Night Sky: why are there more stars off-center?
Averted vision: http://en.wikipedia.org/wiki/Averted_vision

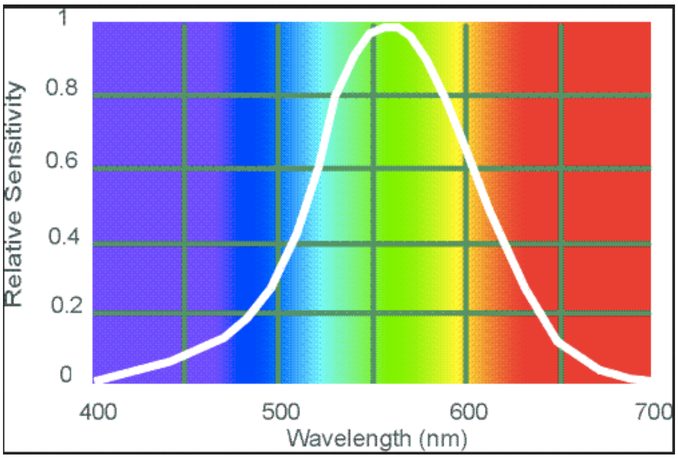
James Hays

Visible Spectrum

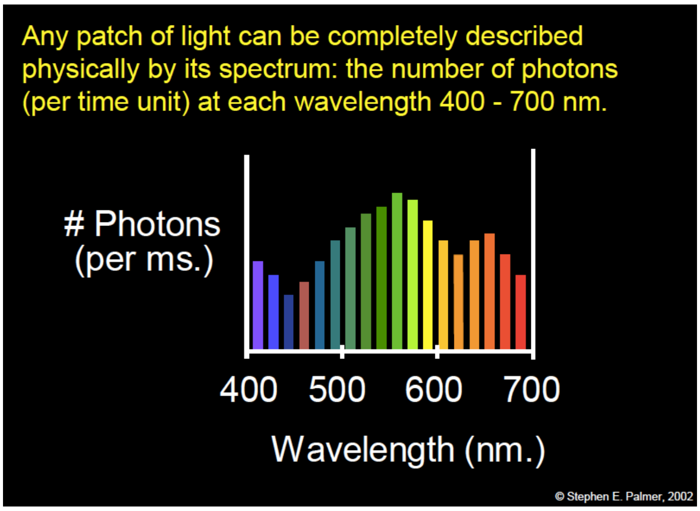


http://www.chromacademy.com/lms/sco736/images/Electromagnetic-spectrum.jpg

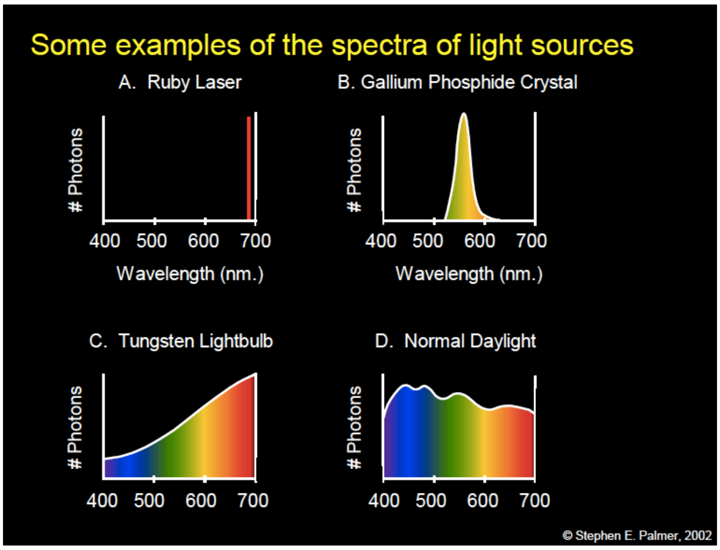
Visible Spectrum



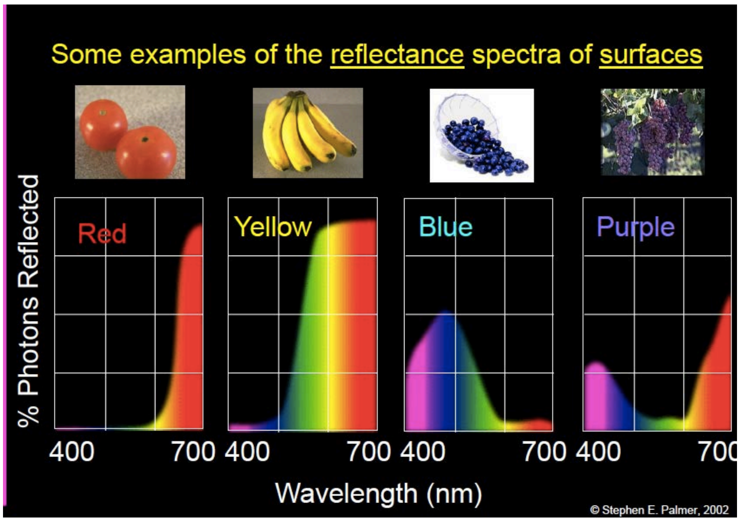
The Physics of Light



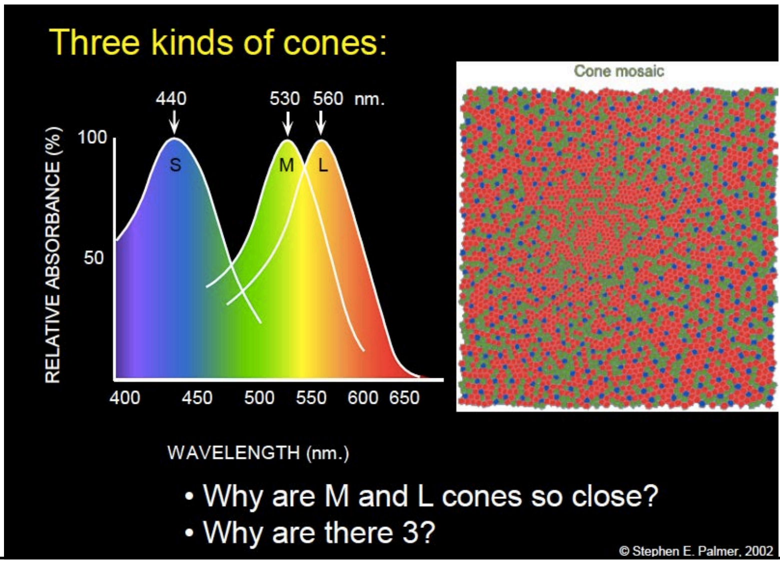
The Physics of Light



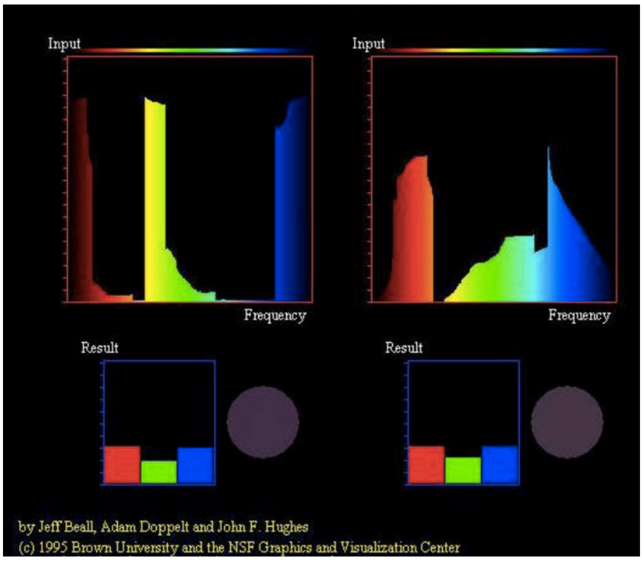
The Physics of Light



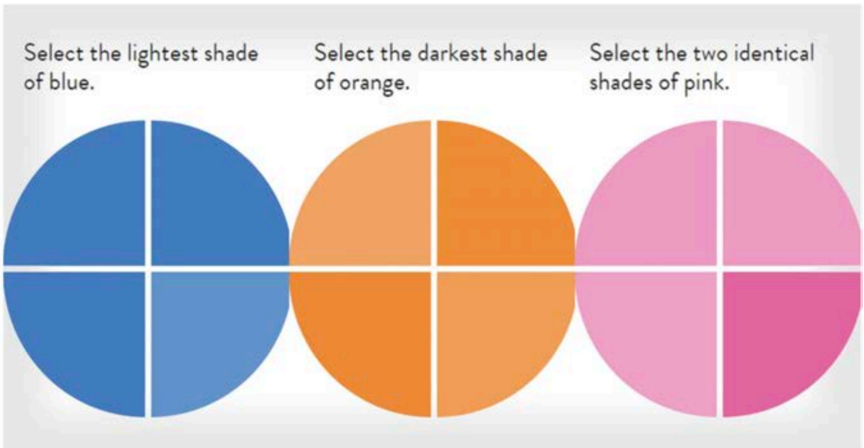
Physiology of Color Vision



Metamers



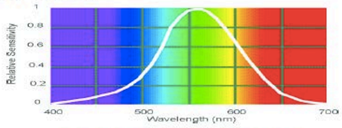
Color Perception

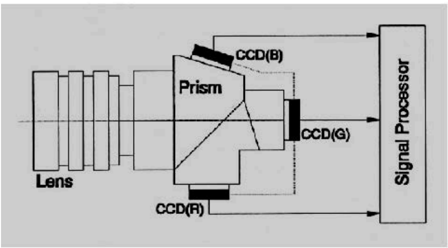


https://petapixel.com/2019/06/05/less-than-1-of-people-can-ace-this-color-perception-test/

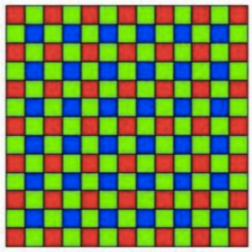
Color Sensing in Camera (RGB)

- 3-chip vs. 1-chip: quality vs. cost
- Why more green?





Why 3 colors?

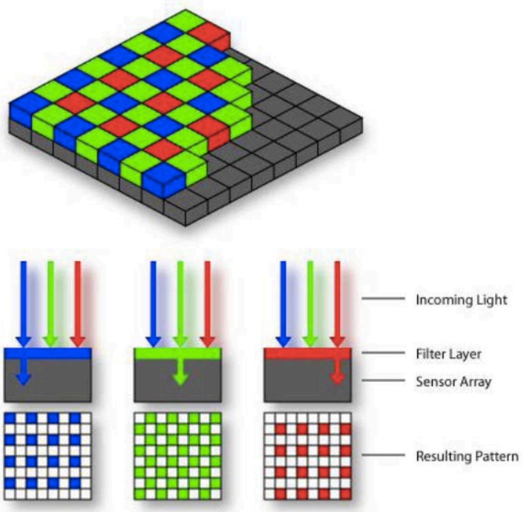


Bayer filter

http://www.cooldictionary.com/words/Bayer-filter.wikipedia

Slide by Steve Seitz

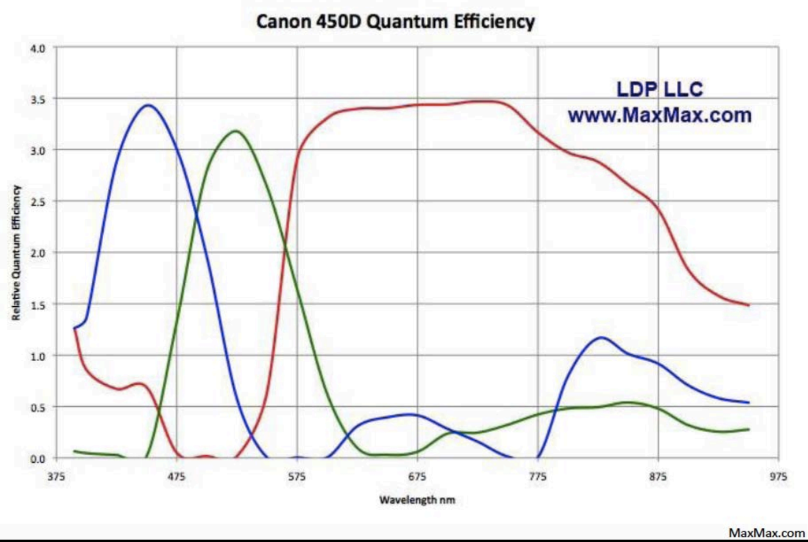
Practical Color Sensing: Bayer Grid



- Estimate RGB at 'G' cells from neighboring values

Slide by Steve Seitz

Camera Color Response



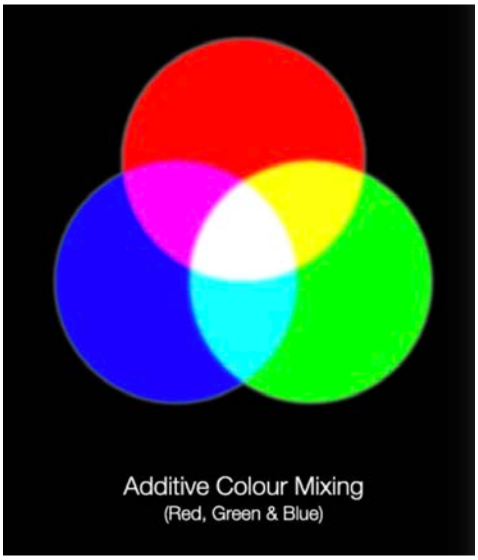
Color Space: How can we represent colors

https://www.liveabout.com/top-color-mixing-tips-2578046

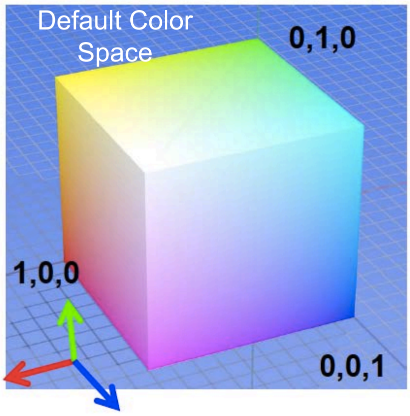
Color Space: How can we represent colors

http://en.wikipedia.org/wiki/File:RGB_illumination.jpg

Additive and Subtractive Colors



Color Spaces: RGB



R = 1
(G=0,B=0)

G = 1
(R=0,B=0)

B = 1
(R=0,G=0)

Any color = r*R + g*G + b*B
- Strongly correlated channels
- Non-perceptual

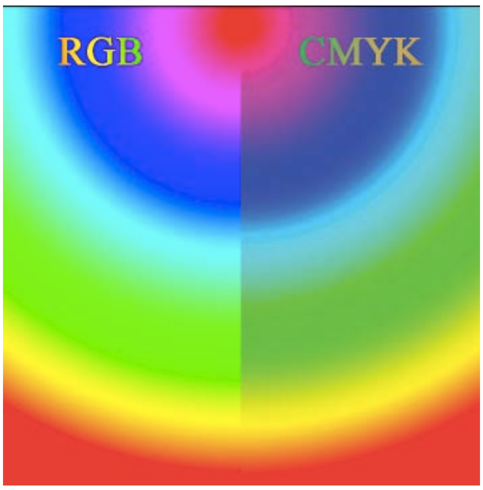
Image from: http://en.wikipedia.org/wiki/File:RGB_color_solid_cube.png

Color Space: CMYK

C – Cyan
M – Magenta
Y – Yellow
K -Black

Subtractive primary colors –
Used in printing

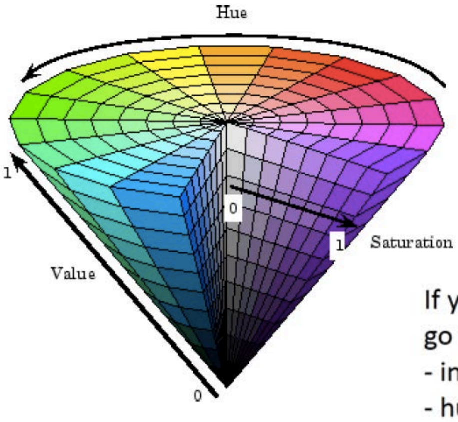
In contrast:
RGB
Additive Primary colors
Computer monitors



Color Spaces: HSV

hue, *saturation*, and *value*

Intuitive color space



If you had to choose, would you rather go without:
- intensity ('value'), or
- hue + saturation ('chroma')?

Color Spaces: HSV

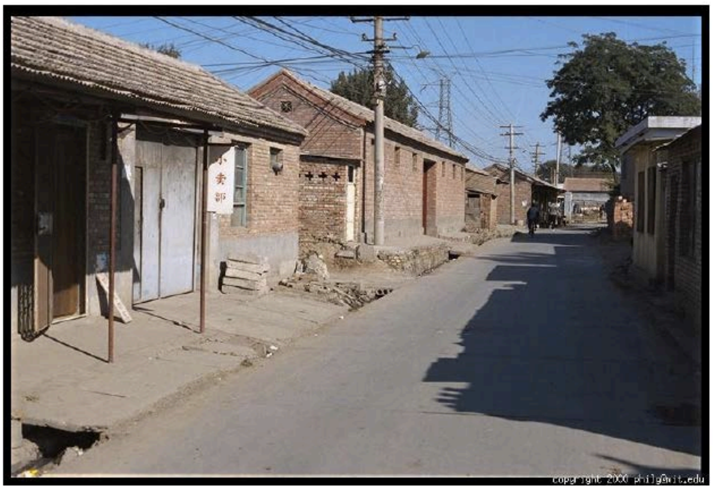
James Hays

Color Spaces: HSV

Constant Color; Only Intensity

James Hays

Color Spaces: HSV

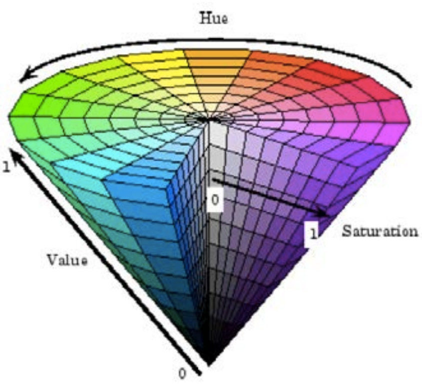


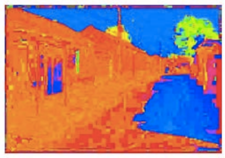
Original Image

James Hays

Color Spaces: HSV

Intuitive color space





H
(S=1,V=1)

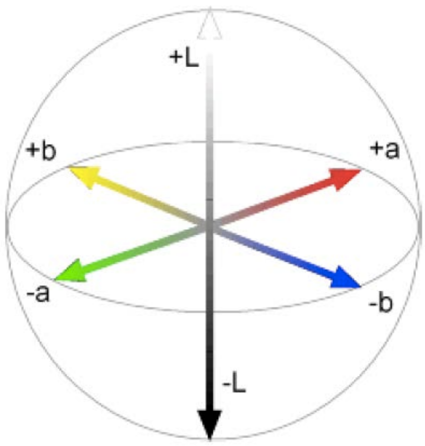
S
(H=1,V=1)

V
(H=1,S=0)

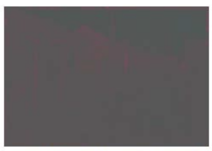
James Hays

Color Spaces: L*a*b*

"Perceptually uniform"* color space



L
(a=0,b=0)



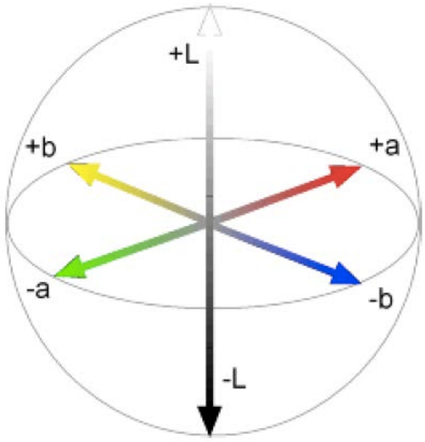
a
(L=65,b=0)

b
(L=65,a=0)

James Hays

Color Spaces: L*a*b*

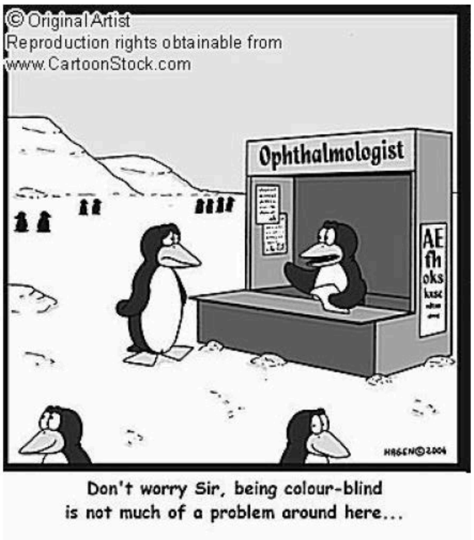
"Perceptually uniform"* color space



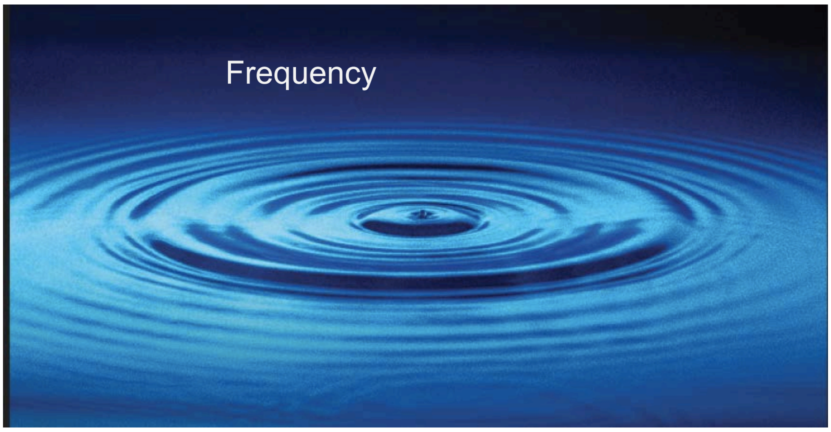
L – Lightness

a,b color opponents

Color



Next class



Links to Some Previous Years Projects

- Supermarket
- AirDrums
- BallBounce
- PizzaPlanner
- VirtualShooting