Image Processing for Pedestrians

Jason M. Kinser

Sept 2008

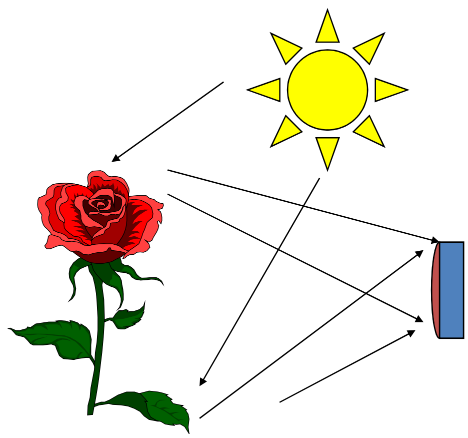
Importance

In the last decade the ability to store biological data as images has increased dramatically.

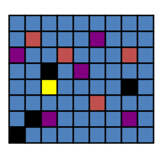
Image databases are being constructed (Allen Brain, Flies, cells, micro-arrays, and MUCH MORE)

A word is worth 10^{-3} images.

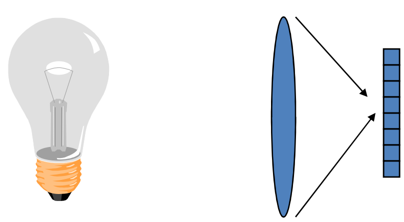
Collecting Information



Digital Camera

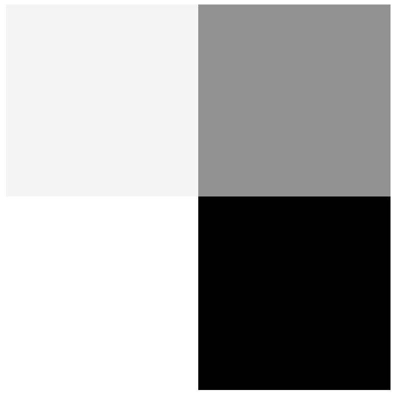


Each pixel receives illumination and produces a single value for that area.

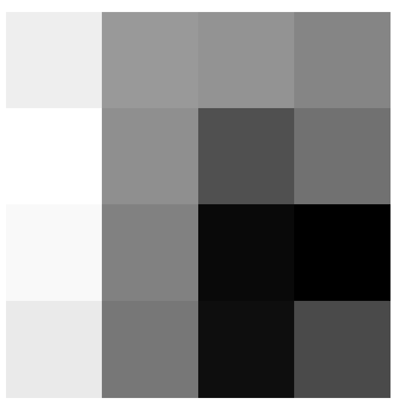


All the illumination is going to a single pixel. Can't tell what the object is.

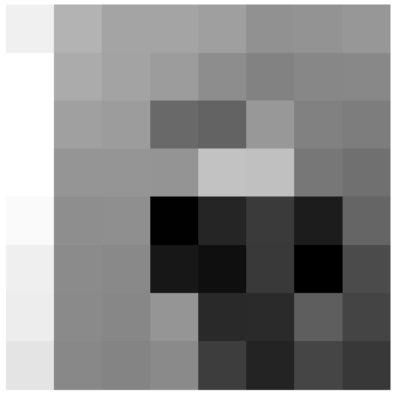
Spatial Scale.



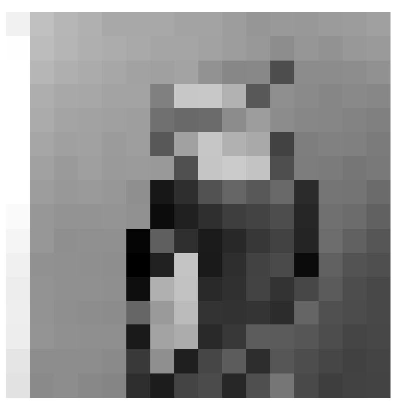
Spatial Scale



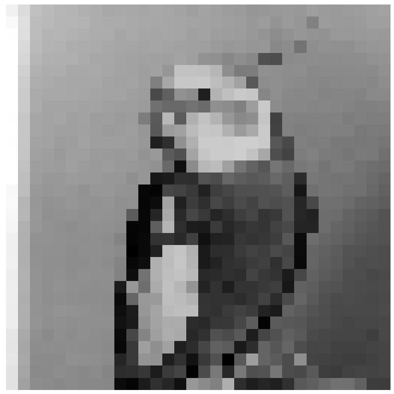
Spatial Scale



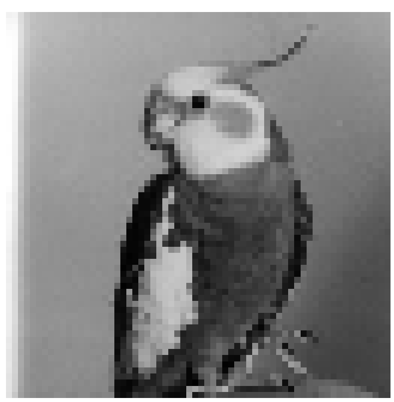
Spatial Scale



Spatial Scale



Spatial Scale



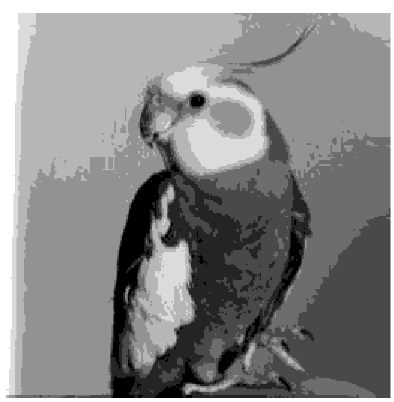
Spatial Scale

What is the pixel resolution of your image?

Number of Bits - 1

Number of Bits - 2

Number of Bits - 3



Number of Bits - 4

Number of bits - 5

What is the gray scale resolution of your image?
Also called *bit depth*.

Number of bits - 16

Some detectors give us 16 bits of intensity information. That's 65536 levels of gray scale.

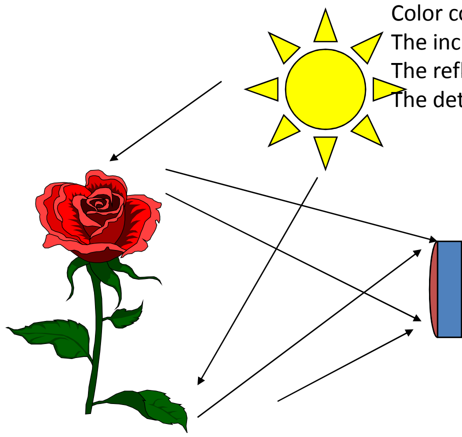
Can we see that?

Can our computers display it?

COLOR

RGB

Collecting Information

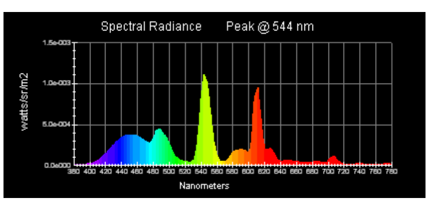


Color comes from:
The incident light,
The reflectance of the object,
The detectors response.

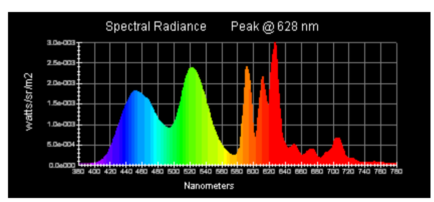
SMPTE 240M

Standard for Digital TV.

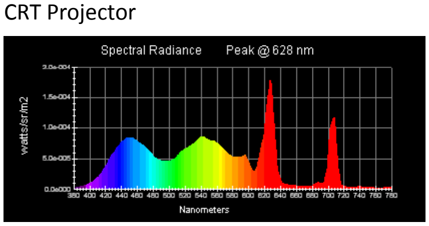
See http://www.videoessentials.com/jkp_facts.htm



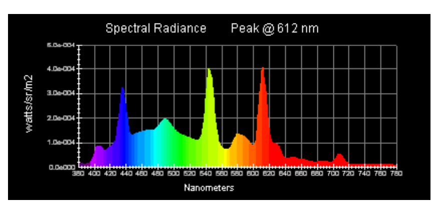
CRT Projector



LCD Projector

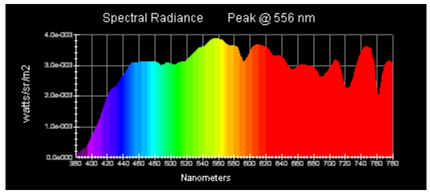


Direct TV



Fluorescent Lamp

Nature



Sunlight

RGB

There are 3 channels - 3 different gray scale images. R, G, B.

Each pixel has 3 numbers each between [0,255].

Bits

How many different colors?

1 bit = 2^1 = 2 colors

4 bits = 16 colors

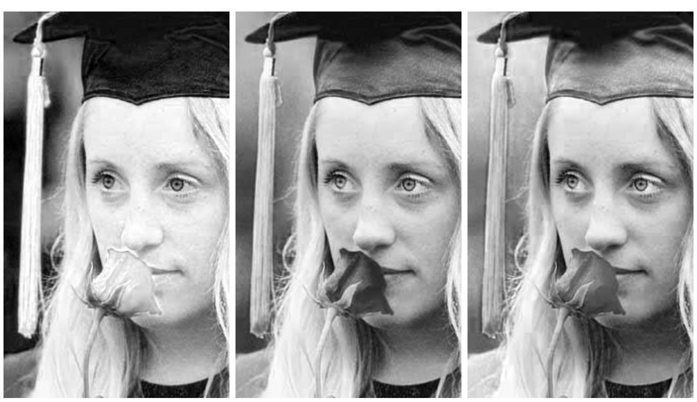
8 bit = 256 colors (just fine for gray)

16 bits = 65,536 colors

24 bits = 16.7 million different colors. (We must name them all.)

I'm so blue...

Notice that the blue is grain-ier.



GBR

Dynamic Range

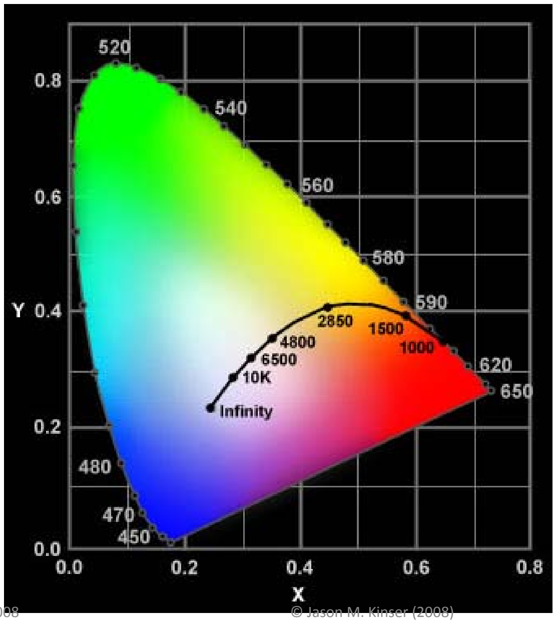
Difference between the lowest and highest values. Obviously, a low dynamic range means you have a duller picture.

RGB?

What are the three primary colors?

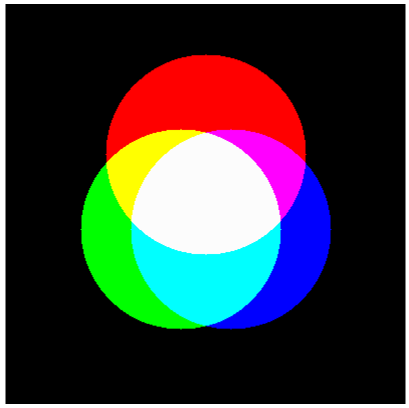
It's not Red, Green, and Blue. Why then do we use RGB instead of RYB?

CIR Color Chart



Pick ANY 3 colors. It makes a triangle. You can get anything in that triangle. RGB is the obvious choice?

3 Colors



Problems with RGB

A single color could have several numerical representations.

Not all colors can be realized.

Color
CMY

C = Cyan = green + blue

M = Magenta = blue + red

Y = Yellow = red + green

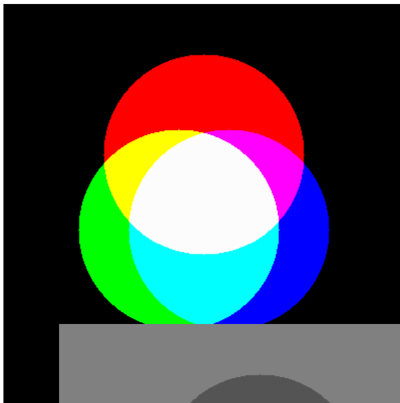
Color _ YUV

YUV

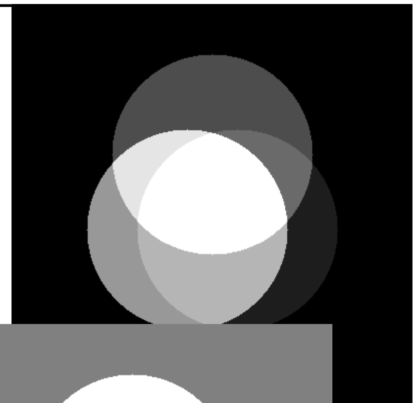
YIQ

YCbCr

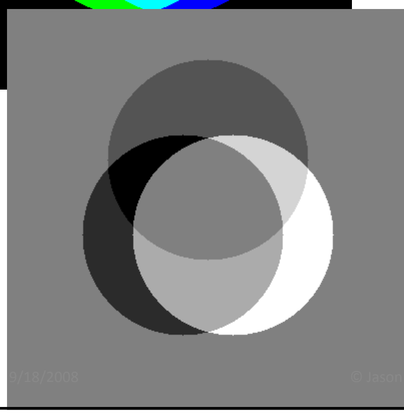
Try #1



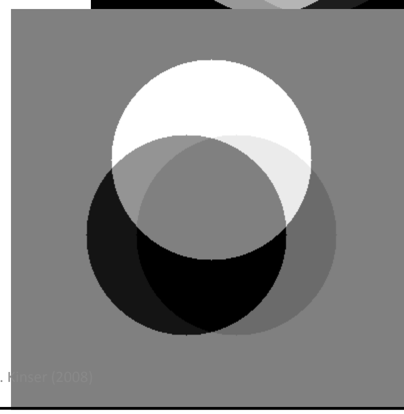
Y



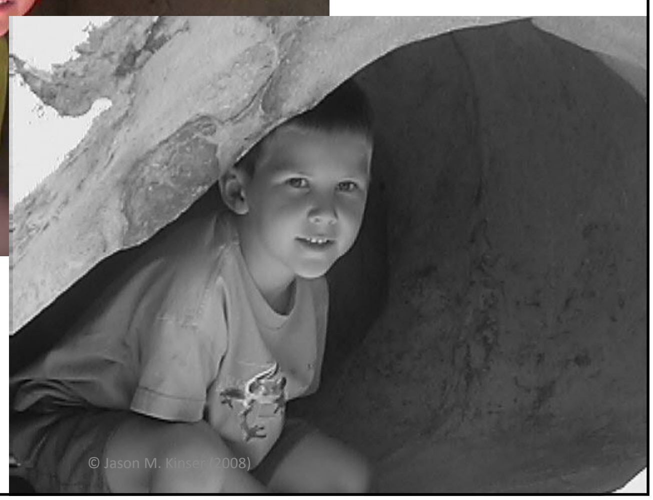
U



V



Try #2



Jacob's U

Why the rectangular grid? This image was originally a JPG.

Jacob's V

HSV

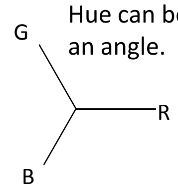
Hue Saturation and Value

Hue = Color

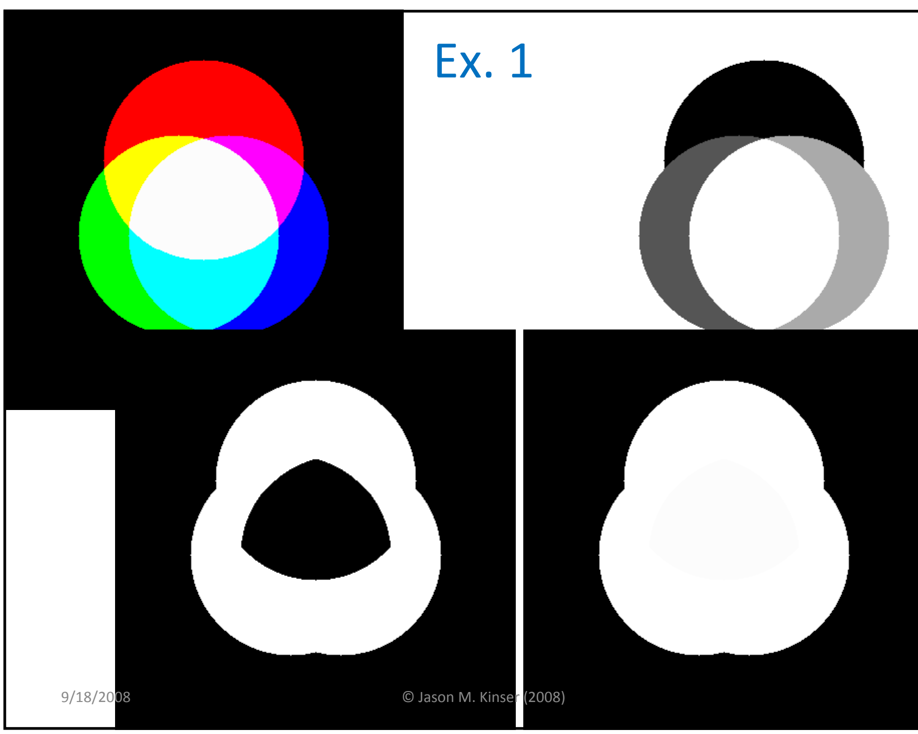
Saturation = Brilliance (gray = 0)

Value = Intensity [0,100]

Hue can be considered as an angle.



Ex. 1



Ex.2

H

S

V

Reproduction

Original

RGB > HSV > RGB

Change H

H=H+30

H=H+180

Change S or V

s = s + 0.25

v = v+0.05

How Many Image Formats??

BMP. ".bmp", ".dib"
CUR. ".cur"
DCX. ".dcx"
EPS. ".eps", ".ps"
FLI. ".fli", ".flc"
FPX. ".fpx"
GBR. ".gbr"
GD. ".gd"
GIF. ".gif"
ICO. ".ico"
IM. ".im"
JPEG. ".jpg", ".jpe", ".jpeg"
MIC. ".mic"
MSP. ".msp"
PCD. ".pcd"
PCX. ".pcx"
PDF. ".pdf"
PNG. ".png"
PPM. ".pbm", ".pgm", ".ppm"
PSD. ".psd"
SGI. ".bw", ".rgb", ".cmyk"
SUN. ".ras"
TGA. ".tga"
TIFF. ".tif", ".tiff"
XBM. ".xbm"
XPM. ".xpm"
Keep in mind that not all of these formats can actually be saved by the library.

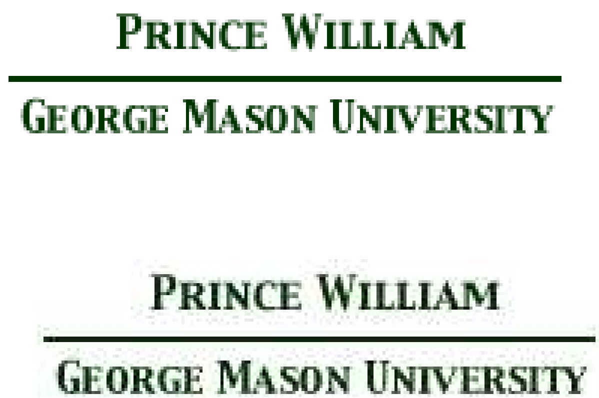
Just to name a few…

Most popular: TIF, JPG, GIF, PNG

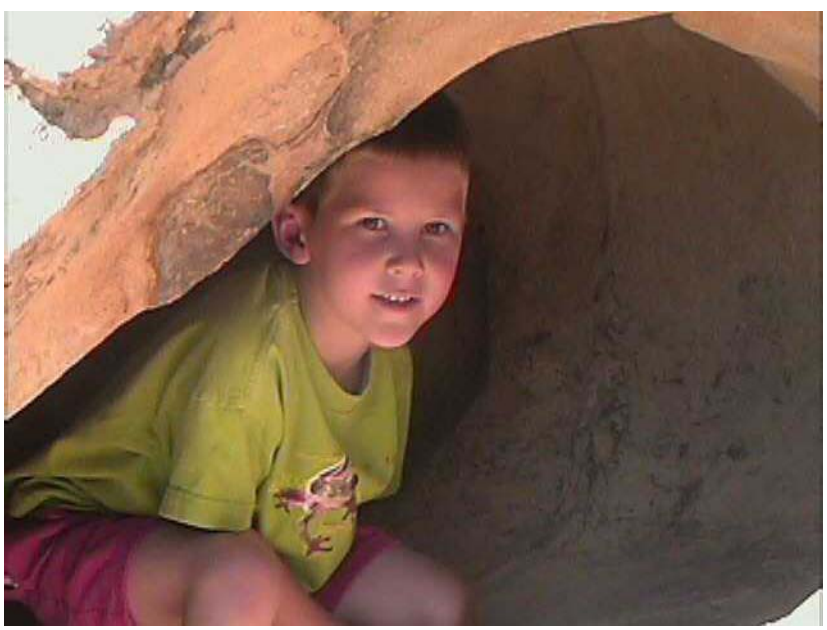
GIF Bad News

The limit of only 256 colors can destroy your image.

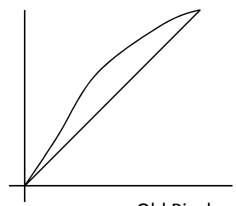
Ringing or Ghosts



JPEG bad news



Brightness



No Change

Instead, the brightness is adjusted by bending the line but the endpoints are kept constant. There are 2 advantages: 1. We don't have many pixel values becoming a single value. 2. We still have a range of 255.

B = A + 50 * exp(-(A-128)^2/(2*20^2))

Threshold

There are several different types of threshold operations. Basically, these operations activate an event if the pixel value is above (below) a specified value.

- Gray level
- Variable or Adaptive
- Band Threshold
- Multi-Threshold
- Semi-Threshold or Passive Threshold

Threshold

$$a_{i,j} = \begin{cases} 1 & if\ b_{i,j} > \gamma \\ 0 & Otherwise \end{cases}$$

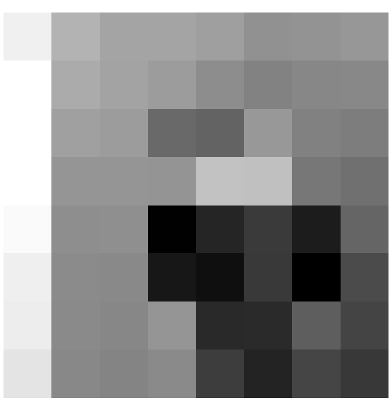
γ = 125

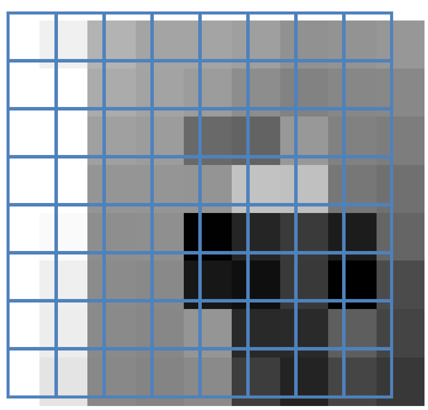
γ = 175

Re-sampling

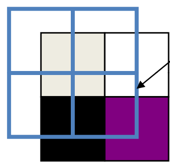
In this section we are concerned with moving information from one location in the image to another location. We have already seen a **shift**.

But we must be concerned with the fact that our image is pixelated.





Pixel Averaging

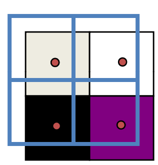


In this square there are four pixel values. Each has a percentage of coverage.

$$a_{i,j} = \alpha b_{i,j} + \beta b_{i,j} + \gamma b_{i,j} + \delta b_{i,j}$$

$$\alpha + \beta + \gamma + \delta = 1$$

Fast Re-sampling



The blue dots represent the center of the original pixels. The new pixels (green) assume the color of whichever original pixel has its blue dot inside.

For large images, pixel averaging provides very little improvement and costs a lot. So, we can use this method to keep things fast. HOWEVER, repeated operations may multiply the error.

Smooth k=7

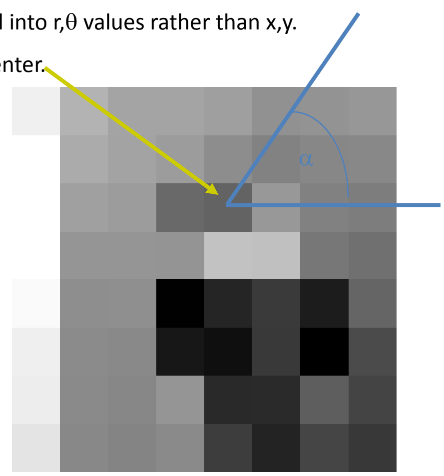
Rotation

The logic is much the same as scaling. We compute from which input pixels the output pixels receive their information.

First we convert the location of each pixel into r,θ values rather than x,y.

In order to do this, we must define the center.

Then we define the angle of rotation, α.



Rotate Compute

Each pixel's location is computed by,

$$\theta_{i,j} = \arctan((j - j_c)/(i - i_c))$$
$$r_{i,j} = \sqrt{(j - j_c)^2 + (i - i_c)^2}$$

We rotate by adding to the angle.

$$\theta_{i,j}^{'} = \theta_{i,j} + \alpha$$

Then revert back to i,j coordinates.

$$i' = r_{i,j} \cos\theta_{i,j}^{'}$$
$$j' = r_{i,j} \sin\theta_{i,j}^{'}$$

Rotate 3

Now, we have the coordinates in the first image (i',j') that correspond to the coordinates in the output image (i,j).

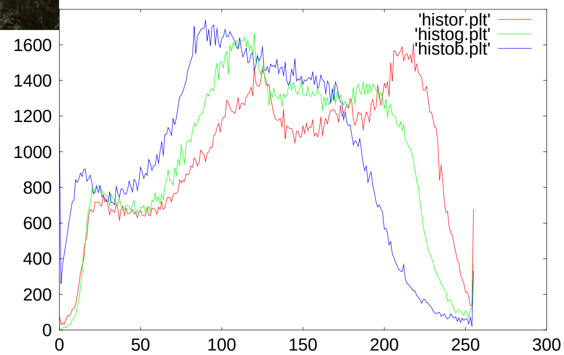
Rotate: center = (100,100) α = 10 degrees.

HISTOGRAMS

Gray

Count the number of pixels at each gray level.

Color



Is it true that blue has more dark pixels than red?

White Noise

Randomly add or subtract values from each pixel.
We must select the level.
L[i,j] = r[i,j] * (1 + (r-0.5)*level) + ...

Gives it a grainy look.

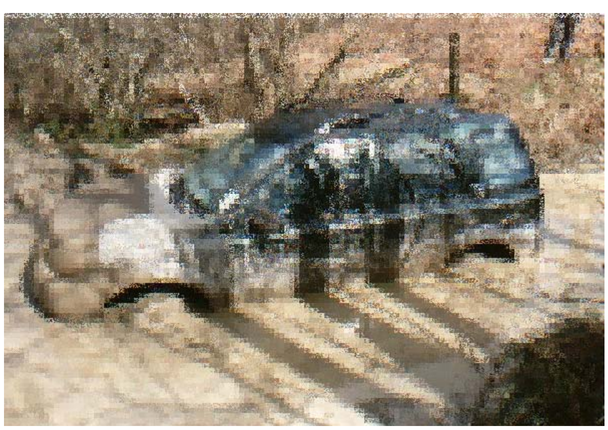
Colored Noise

Alter the frequencies. More on this when we do Fourier transforms.

Splotchy look.

Spread Noise

Select a 5x5 area. Randomly move the pixels around inside this 5x5.

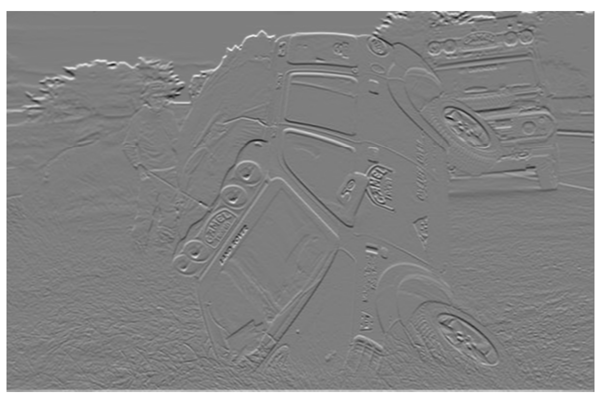


Block look.
We could get rid of the squares by allowing the pixels to move a random distance *d* but the probability of *d* is inversely proportional to *d*.

Picture for the Day

Horizontal Sobel

```
1  2  1
0  0  0
-1 2 -1
```



Note that there are dark and bright lines near the edges. Sharper edges create higher intensities. Dark and bright depicts which way the edge is.

Note, these are only horizontal edges.

Barrel Example

Parallel lines do not remain parallel.

Gives us more resolution near the center - similar to human eye foveation.

Pin Cushion

Same as Barrel except that the bending coefficient is less than 1.

R-Polar

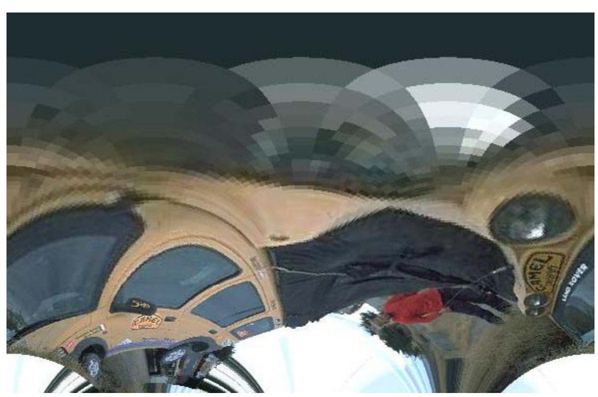
Converts xy coordinates to r,theta coordinates. Plot the new picture in r-theta space.

θ

R

Log_polar

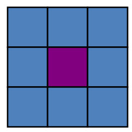
Same only plot log(r)



This puts more resolution near the center.

Erosion/Dilation

Erosion and Dilation operators remove single pixel noise.



Given a radius, *r*

For each pixel:

find the min (or max) of all pixels within *r*

replace the pixel with this value.

Example

Erosion 2

Dilate 2

Noise

I've subtracted 10% noise. Note the grainy nature

Dilate/Erode 1

Grainy nature is gone, but so is some of the resolution. The word 'CAMEL' is gone. It is darker and since we did the dilation first, small dark items were destroyed. Small bright items were maintained.

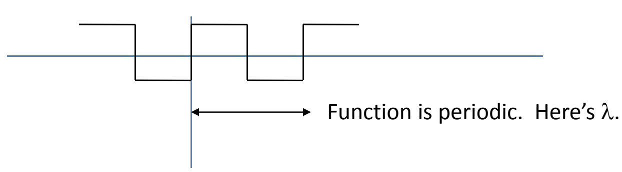
Color Dilate

Fourier Transforms

Any signal (including an image) can be decomposed into a set of frequencies.

This is accomplished by computing the inner products of the data with a set of ortho-normal functions. For Fourier transforms these functions are sine/cosine.

Saw tooth



$$f(x) = \begin{cases} 1 & 0 < x < \lambda/2 \\ -1 & \lambda/2 < x < \lambda \end{cases}$$

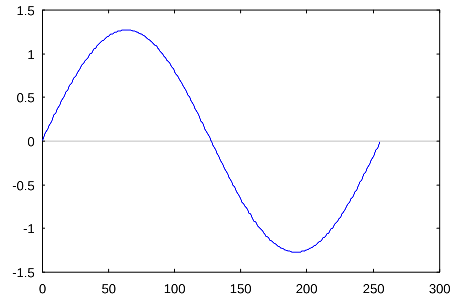
Do the math

$$A_0 = A_m = 0 \qquad B_m = \frac{2}{m\pi}(1 - \cos m\pi)$$

The A's and B's are the transform coefficients.

Reconstruct

```
>>> aft = zeros( 256, Complex )
>>> aft[1] = af[-1]
>>> aft[1] = af[1]
>>> aft[-1] = af[-1]
>>> b = FFT.inverse_fft( aft )
```

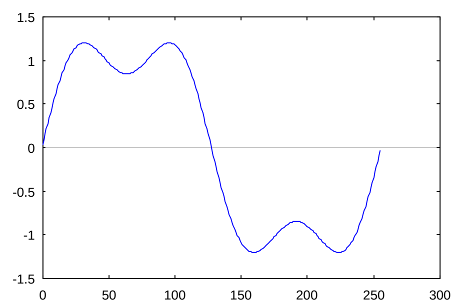


Plot of REAL component of *b*.

```
>>> gnu.Save('dud.txt', b.real )
```

Again

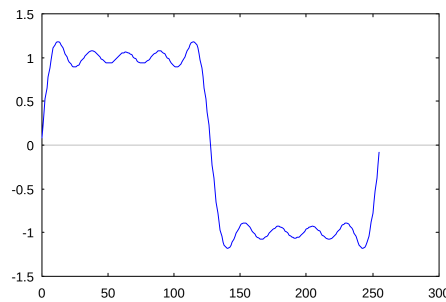
```
>>> af[2] # No value.  Move to 3
0j
>>> aft[3] = af[3]
>>> aft[-3] = af[-3]
>>> b = FFT.inverse_fft( aft )
>>> gnu.Save('dud.txt', b.real )
```



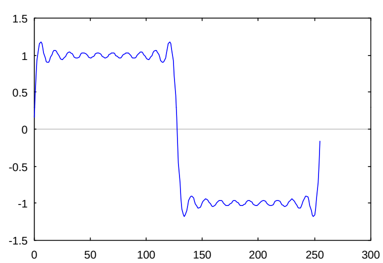
MORE

```
>>> for i in range( 1, 11):
            aft[i] = af[i]
            aft[-i] = af[-i]


>>> b = FFT.inverse_fft( aft )
>>> gnu.Save('dud.txt', b.real )
```

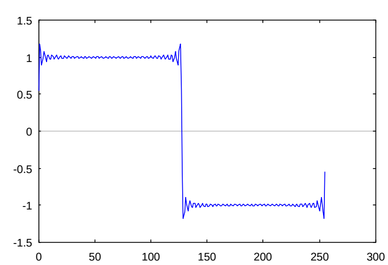


Etc.

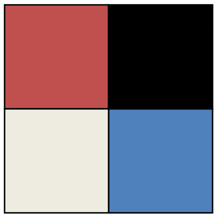


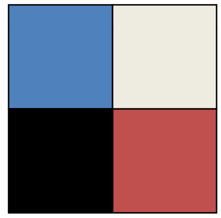
21 components

71 Components



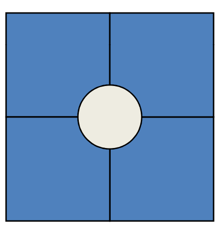
Swap





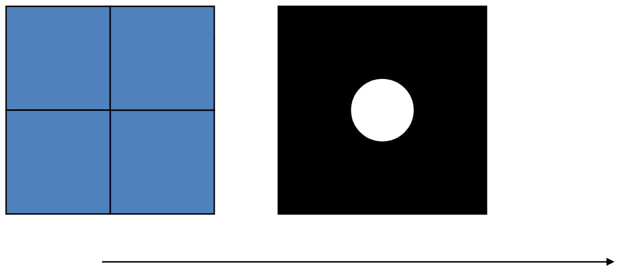
Puts the DC in the center of the frame.

Filtering



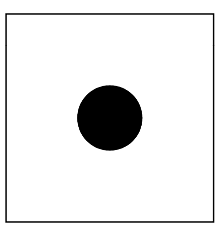
After a swap, the low frequencies are in the center.

LoPass

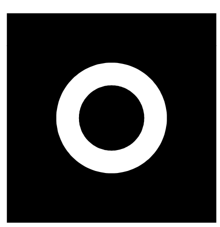


Pass only those low frequencies.
Try it.

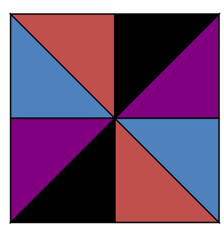
Pass Filters



Hi Pass



Band Pass



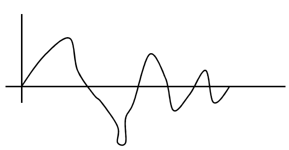
Wedge

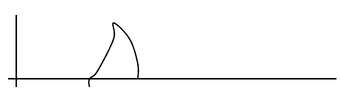
Try them out

Colored Noise

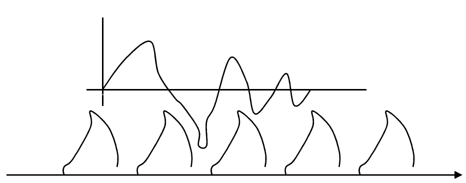
Randomly change the frequencies.

Correlation





Does this function reside in previous signal?



All possible shifted inner products.

Formally

$$c(p) = \int a(x-p)b(x)dx$$

If A is length *m* and B is length *n* then there are *m*x*n* multadds.

What does it mean?

$$\Im_p\{c(p)\} = F(p)G^*(p)$$

The Fourier Transform of the Correlation is:
The elemental-multiply of the Fourier Transforms of the individual functions
- with one being conjugated.

No, really, what does it mean?

$$\Im_p\{c(p)\} = F(p)G^*(p)$$

If f(x) and g(x) are N dimensional vectors then the number of multadds needed to perform the correlation is N^2.

For a FFT we need N $\log_2$(N) operations. To perform the correlation with FFTs we would need 3N $\log_2$(N) + N operations.

N	N*N	FFT
1	1	1
2	4	8
4	16	28
8	64	80
16	256	208
32	1024	512
64	4096	1216
128	16384	2816
256	65536	6400
512	262144	14336
1024	1048576	31744
2048	4194304	69632
4096	16777216	151552
8192	67108864	327680

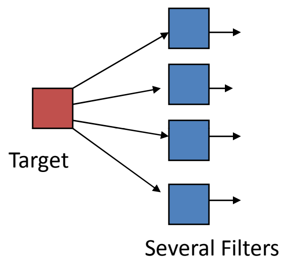
ATR

Problems with ATR.

Illumination spectrum, illumination angle, diffusivity, viewing angle, background, noise.

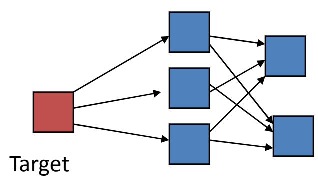
Occlusion, distortion, camouflage

Several Filters



Pros: Easy to train

Cons: Too many filters, still doesn't deal with unexpected presentations



Pros: Higher order decisions, Kolomogorov is happy

Cons: Which filters to we build?

Kolomogorov: Linear > Nonlinear > Linear

Composite Filters

Desire:

$x_i * h = y_i$ A single filter can recognize several targets.

Rewrite

$$\mathbf{X} \cdot \vec{h} = \vec{c}$$

All we need to do is solve for **h**.

This could be done with

$$\mathbf{X}^{-1}\mathbf{X} \cdot \vec{h} = \mathbf{X}^{-1}\vec{c}$$

If **X** was a square matrix.
It's not.

FPF

Fractional Power Filter:

What's the difference between the SDF

$$\vec{h} = \mathbf{X}^T \left(\mathbf{X}^T \mathbf{X} \right)^{-1} \vec{c}$$

And the MACE

$$\vec{h} = D^{-1/2} \mathbf{X}^T \left(\mathbf{X}^T D^{-1} \mathbf{X} \right)^{-1} \vec{c}$$

$$D = \frac{\delta_{ij}}{N} \sum_n \left| x_{n,ij} \right|^2$$

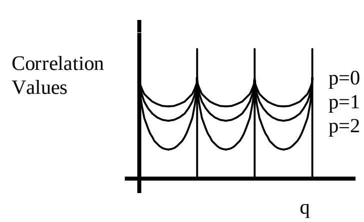
It's the matrix **D**. What if **D** were diagonals of 1's (the Identity)?
Then the MACE would become the SDF

Parameter Estimation

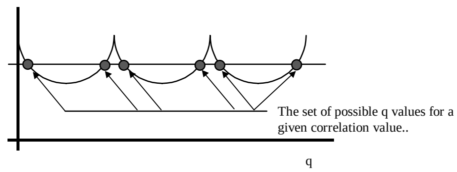
It's one thing to build an invariant filter.

It's quite another to measure the parameter.

What does the correlation peak do when we change the parameter *q*?

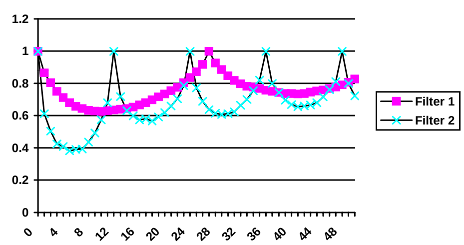


Estimation 2



Given an input with an unknown *q* we can perform the correlation. We get a correlation peak value. There are several *q*'s that can produce that value.

Estimation 3



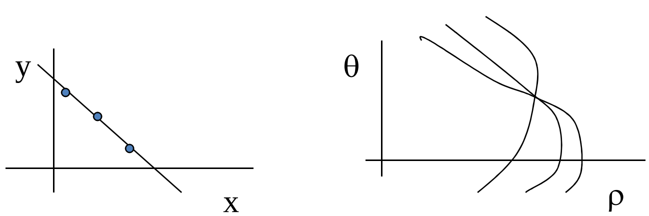
Build 2 filters with different training *q*'s. For each an input with an unknown *q* will produce a set of possible *q* values. However, there should only be one member of the intersection of these two sets.

Fractional Fourier Transforms

$$F\{f(x,y)\} = \int\int f(x,y)\, e^{\iota \frac{k \cos\phi}{2 \sin^2\phi}\left[(x^2+\alpha^2)+(y^2+\beta^2)\right]} e^{-\iota \frac{k}{f \sin^2\phi}(x\alpha+y\beta)}\, dxdy$$

This lies halfway between the image space and the F space. It has some shift invariance but also sensitive to location. Perhaps you know that the target is going to be somewhere in the middle.

Linear Hough

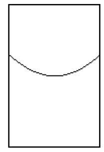


$\rho = x_i \cos\theta + y_i \sin\theta$

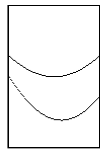
Points become lines in the new space. Intersections indicate that a straight line exists. We can use this to look for circles, parabolas, etc.

Run

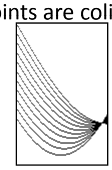
```
>>> a = zeros( (100,100) )
>>> a[50,50] = 1
>>> ho = LineHough( a, 0.5 )
```



```
>>> a[10,70] = 1
>>> ho = LineHough( a, 0.5 )
```



When multiple points are colinear



```
>>> a = zeros( (100,100) )
>>> a[20] = ones( 100 )
>>> ho = LineHough( a, 0.5 )
```

```
>>> a = zeros( (100,100) )
>>> for i in range( 100 ):
            y = 0.4 * i + 2
            a[i,y] = 1
>>> ho = LineHough( a, 0.5 )
```

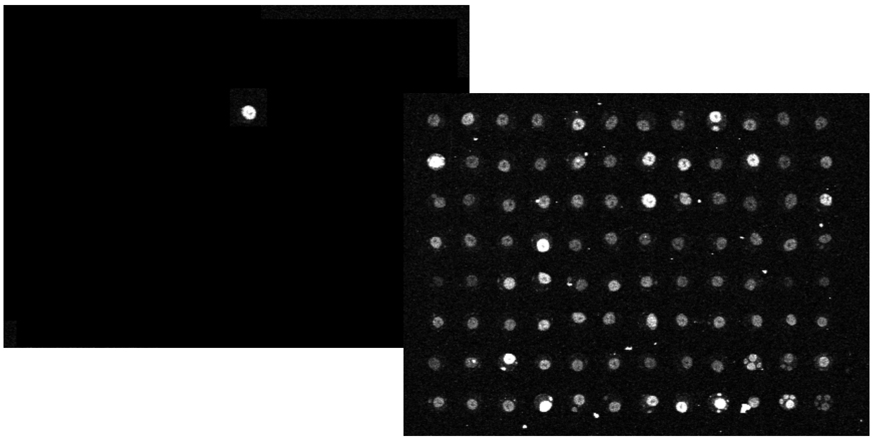
Circular Hough

We have 3 dimensions: x,y,radius.

So, Hough space will be 3D and a each point in the input space creates a cone in Hough space.

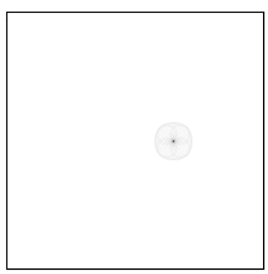
However,

There are cases in which you will know the radius of the object. ‘*r*’ is fixed and therefore Hough space is 2D.

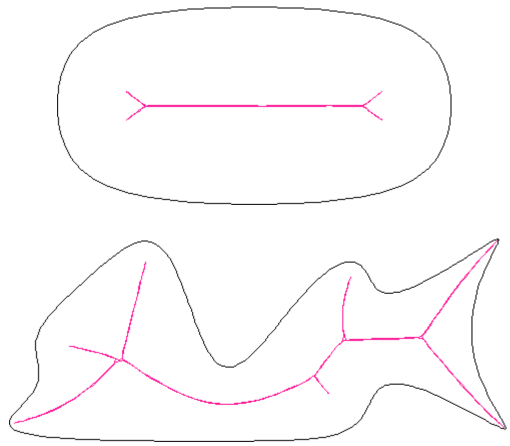


Example

```
>>> a = akando.Circle( (256,256), (90,128), 10 )
>>> a -= akando.Circle( (256,256), (90,128), 9 )
>>> ho = CircleHough( a, 9, 0.5 )
>>> akando.a2i( ho.max()-ho ).save('dud.gif')
```

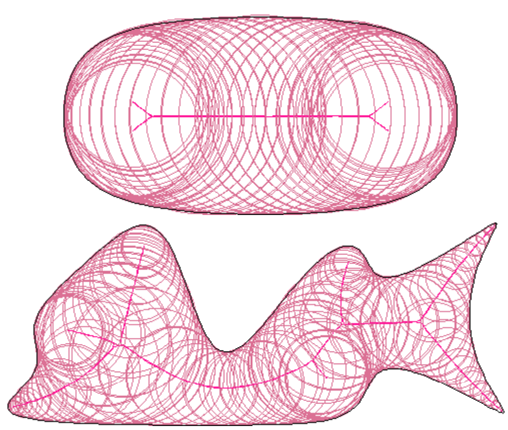


Medial Axes



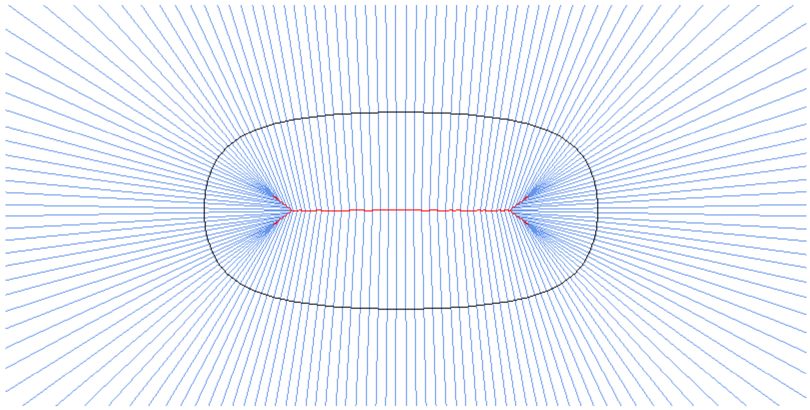
http://www.mpi-sb.mpg.de/~shin/Research/CCurve/node19.html

Bitangent circles

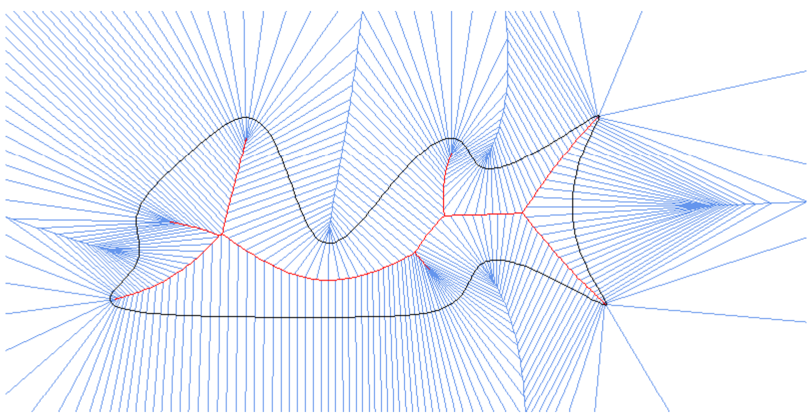


Find the center of circles that have two tangents along the perimeter.

Voronoi Diagrams

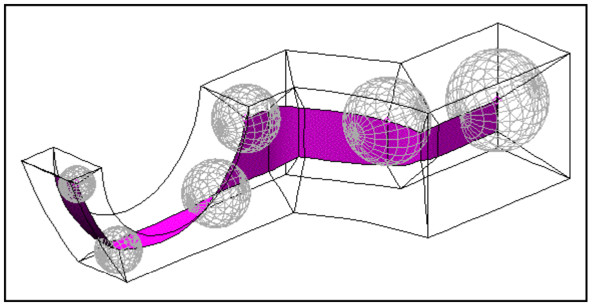


Voronoi 2



Normal vectors equidistant from perimeter to spine.

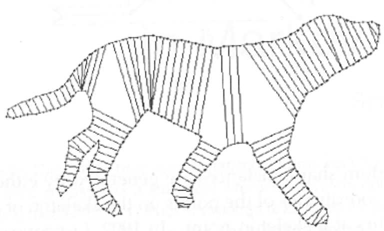
3d

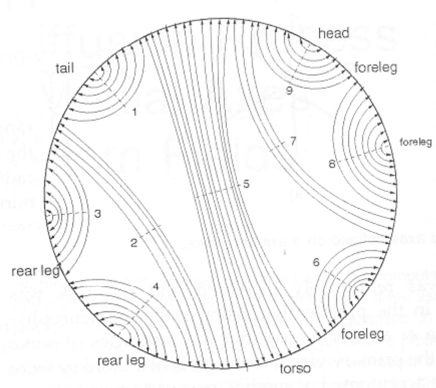


http://www.fegs.co.uk/medial.html

What can you do with them?

Match Graphing

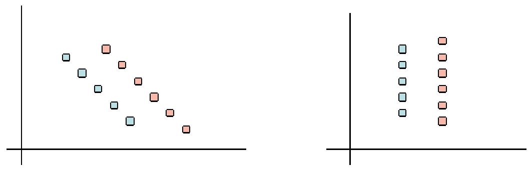




IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 21, NO. 11, NOVEMBER 1999

Eigenimages

Start with PCA.



PCA

Morph data

Center Data

Compute Covariance matrix.

Compute eigenvectors and values of covariance matrix.

$$\mathbf{A}\vec{v} = \mu\vec{v}$$

Select the best vectors (according to eigenvalues).

Convert data to this new space (inner products).

Bad Average

```
>>> mg = Image.open(dr+'mugshot.ppm').convert('L')
>>> m1 = i2a( mg )
>>> mg = Image.open(dr+'jen.ppm').convert('L')
>>> m2 = i2a( mg )
>>> mg = Image.open(dr+'jaker.ppm').convert('L')
>>> m3 = i2a( mg )
>>> mg = Image.open(dr+'kate1.ppm').convert('L')
>>> m4 = i2a( mg )
>>> m = (m1+m2+m3+m4)/4
>>> q.Paste( a2i( m ))
>>> q.mg[0].save( dr+'badavg.gif')
```

Badavg.gif

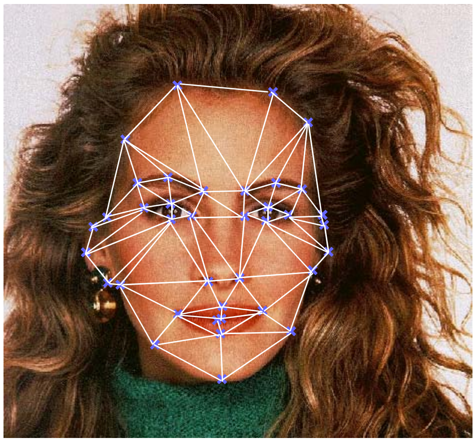
We cannot average a face by averaging the pixels because features are not co-registered.

Morph

Lattice structure.

Data movement.

Triangles

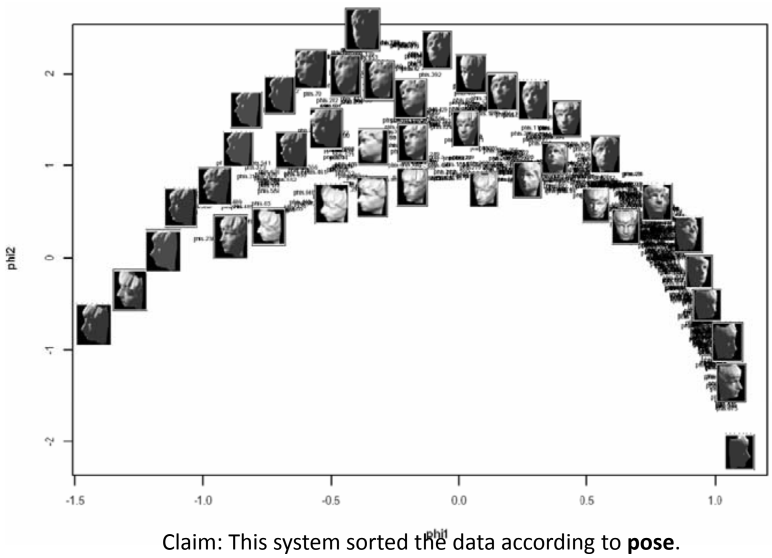


Triangle vertices are at fiducial points. You want the corners to be at junctions in the image.

See touch.tcl

Changing One into Another

Brandon Higgs



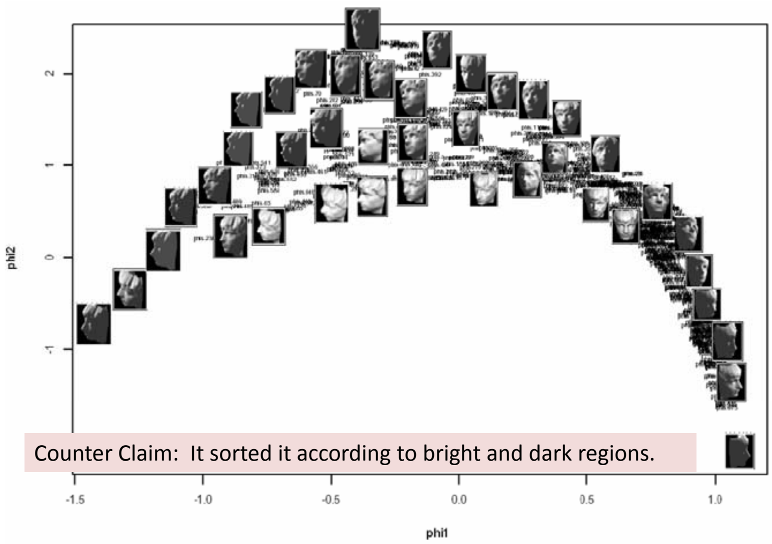
Claim: This system sorted the data according to **pose**.

Not a Panacea

PCA is usually a *first order* system.

Uses pixel-by-pixel comparisons

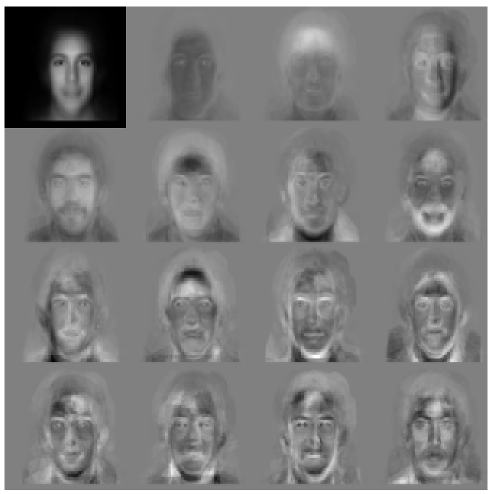
First Order



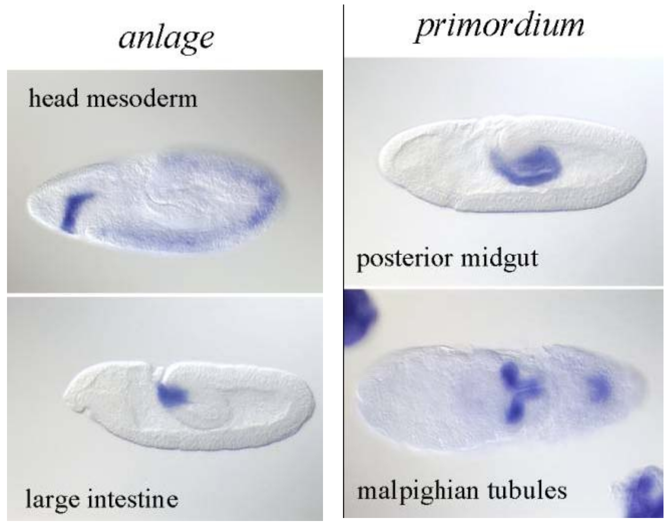
Counter Claim: It sorted it according to bright and dark regions.

EigenFaces

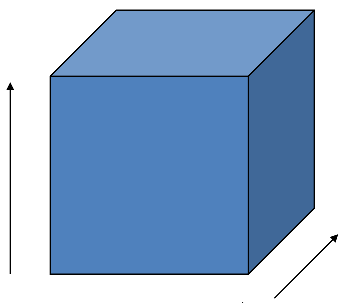
1. Morph all faces to the same grid. Make them 0 sum.
2. Convert images to vectors (raster or ravel).
3. Compute the eigenvectors. Convert the eigenvectors to images.
4. Rank them according to their eigenvalues.



Flies



Cube



A datacube has three axes.

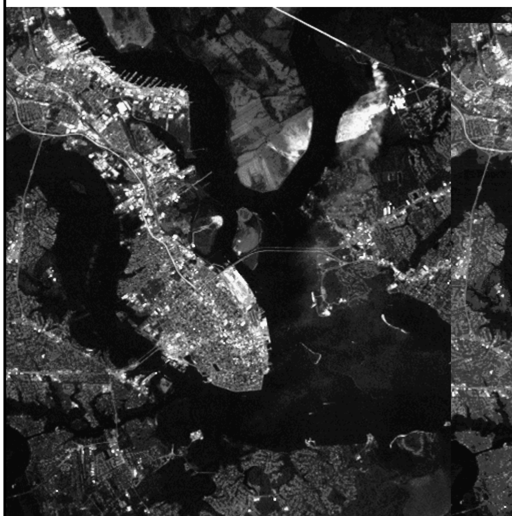
Usually, **x** and **y** are spatial coordinates much like a normal gray image.

The **z** axis is space, time, wavelength, or other.

So many bytes:

256 x 256 x 256 = 16 Meg!

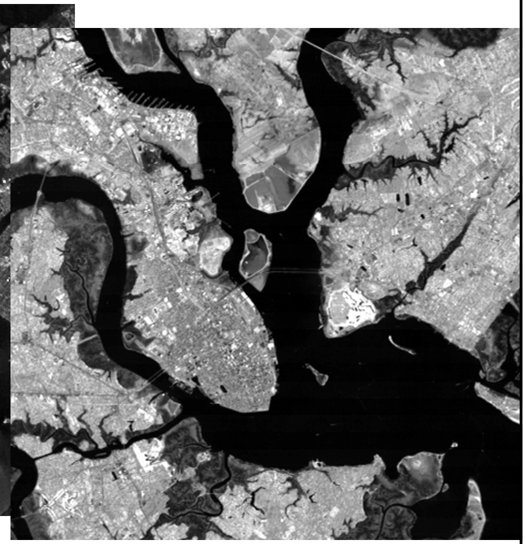
Sample from the Web


Band 0

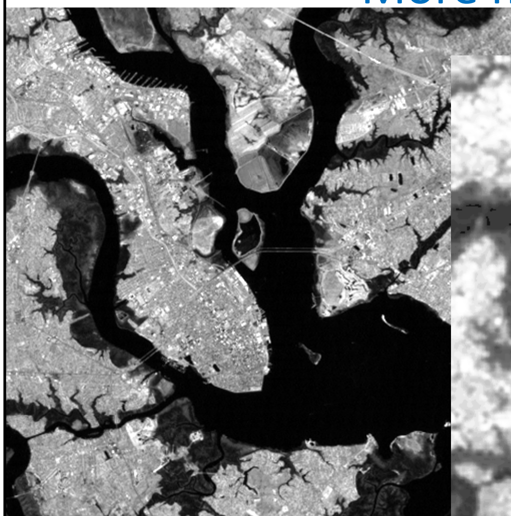

Band 1

More

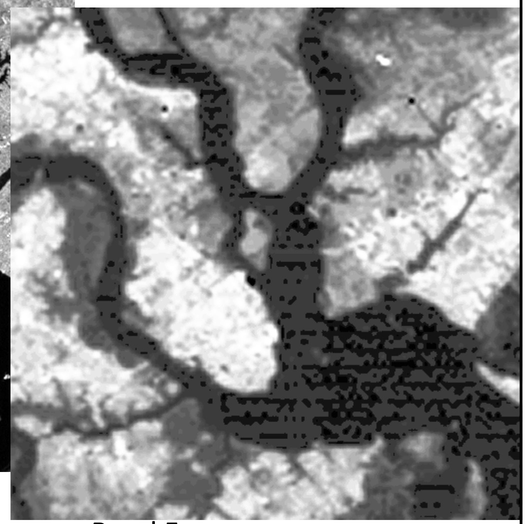

Band 2


Band 3

More more



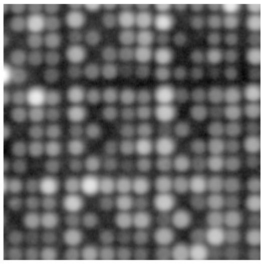
Band 4



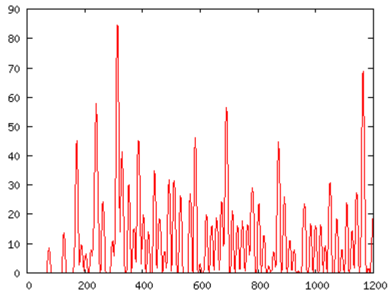
Band 5

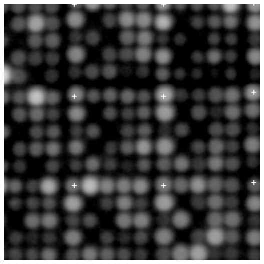
Cubes in Biology

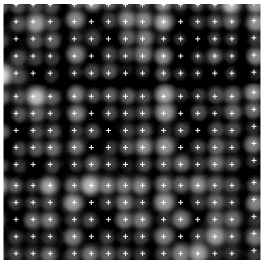
Marrays

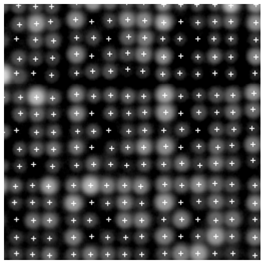


Spot Finding

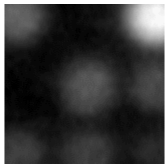








Quality



$$q_{com} = (q_{size} \times q_{snr} \times q_{bkg1} \times q_{bkg2})^{1/4} \times q_{sat}$$

$$q_{size} = \exp\left(-\frac{|A - A_0|}{A_0}\right)$$

$$q_{snr} = 1 - \left[\frac{bkg_I}{sig + bkg_I}\right] = \frac{sig}{sig + bkg_I}$$

$$q_{bkg1} = \frac{f_1}{CV_{bkg}}$$

$$q_{bkg2} = f_2\left\{1 - \frac{bkg_1}{bkg_1 + bkg_0}\right\}$$

Fini

Upcoming courses using Image processing:

BINF 734 – Advanced Programming

??? CSI - Image Processing

??? BINF 739 – Image Processing for Bioinformatics

jkinser@gmu.edu

3-3785