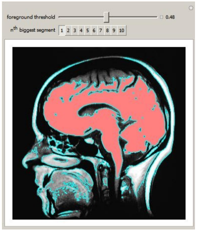
Image Processing

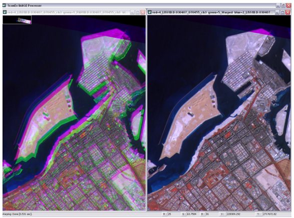
September 6, 2013

Chloe Eghtebas, Luke Metz, Brendan Ritter

Basics of Image Processing



Using computational software to manipulate an image using signal processing techniques.

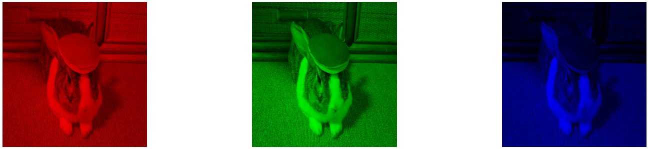


What You'll Learn

1. "Signal Processing"
   - Shear
   - brightness and contrast
   - grayscale
   - invert
   - Rotating
   - Flipping
2. Edge Detection
   - Gaussian blur
   - Sobel/convolution

Image Representation and Software

Images have 3 channels. Red, Blue, Green. To manipulate these arrays we chose to not use matlab. We chose to use Python and Numpy mainly for its re usability.



Flipping

To flip our images, we took our 3 large matrices, rgb, and multiplied them by a matrix like the one bellow except the right size. $A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$ To flip horizontally, one can multiply AC. To flip vertically, one can CA. Where C is the image matrix.

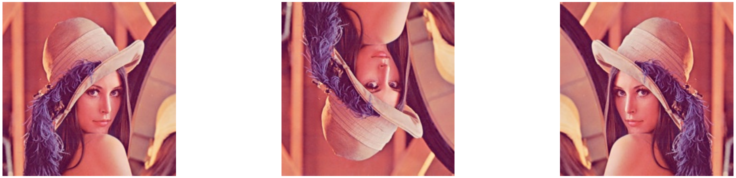


Figure: Original Image, flipped vertically, flipped horizontally.

Transformations

To apply more complicated transformations to the image one first has to re arrange the image. Currently x and y position are stored at the index of the matrix. This gives us no easy way to manipulate them.

We can rearrange The image to look like:

$$A = \begin{pmatrix} x & 0 & 1 & \dots \\ y & 0 & 0 & \dots \\ 1 & 1 & 1 & \dots \\ r & 255 & 255 & \dots \\ g & 255 & 255 & \dots \\ b & 255 & 255 & \dots \end{pmatrix}$$

This will let us translate, rotate, scale, and shear the image in any way.

Rotation

Example Rotation: The Default rotation is always around the origin. To make that the center of the image, one must first translate, rotate, then translate back.

$$A_{out} = \begin{pmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{pmatrix} A$$

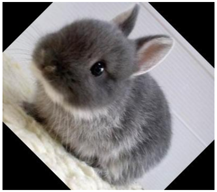


Figure: Rotated around the center at $\theta = 45^\circ$

Shear

Example Shear:

$$A_{out} = \begin{pmatrix} 1 & \lambda_y & 0 \\ \lambda_x & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} A$$

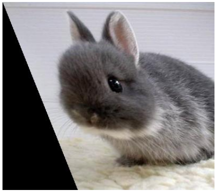


Figure: Sheared with $\lambda_X = .4$ and $\lambda_Y = 0$

Color

Our next feature that we implemented was color controls per channel. This would let us control how much of one color there is as well as increase or decreases brightness, changing all channels. This can be done by simply adding values to the matrix at each channel.

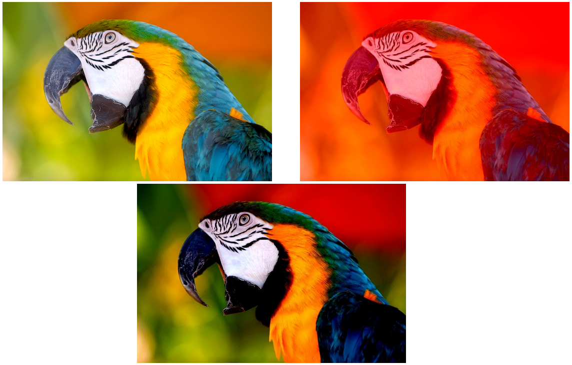


Figure: In order from left to right, original image, red increased with green and blue decreased, all channels decreased.

Convolution

Convolution is applying one function to another to construct an output of the system.

Same priciple for discreate convolution. It's like overlapping one signal with another. One signal has original data and is convolved with a kernel which represents the other function.

Convolution in 1d

In 1D, one can think of superimposing the kernel over each point of data.

$$A = (0, 2, 3, 1, 2, 1, 0)$$

$$K = (1, 1, 1)$$

$$S = AK = (2, 5, 6, 6, 4, 3, 1)$$

Or Mathematically for a kernel of size 3,
$S_n = A_{n-1}K_0 + A_nK_1 + A_{n+1}K_2$
Because our vectors are not infinite, we have to decide what happens on the edges. We will assume that the $A_{-1} = A_{n-1}$ where n is the size of the vector.

Multiplying Matrices

Convolution around a kernel is a linear operator, thus one can find a matrix, C, that transforms A to S. $CA = S = AK$.

$$C = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

This matrix is in the form of a Circulant matrix. Matrices in this form are easy to analyze.

Eigenthings

Because this matrix is Circulant, its eigenvectors and values are easy to find. Assuming the following circulant matrix,

$$\begin{array}{ccccc} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{array}$$

The eigenvaluess and vectors are:

$$\lambda_j = c_0 + c_{n-1} w_j + c_{n-2} w_j^2 + ... + c_1 w_j n - 12$$

$$v_j = (1, w_j, w_j^2, w_j^3, ..., w_j^{n-1})$$

Where w are the roots of unity, $z^n = 1$ or

$$w_j = e \frac{2\pi ij}{n}$$

Eigenthings Cont.

For our demo kernel,

$$\lambda_j = 1 + e^{\frac{2\pi ij}{n}} + e^{\frac{(n-1)\pi ij}{n}}$$

we can say $(n-1) = -1$ to simplify because we multiplying by 2π. Thus

$$\lambda_j = 1 + e^{\frac{2\pi ij}{n}} + e^{\frac{-\pi ij}{n}}$$

Using eulers formula

$$\lambda_j = 1 + 2\cos(\frac{2\pi j}{n})$$

$$\lambda_0 = 3$$

is the maximum eigenvalues. This value corresponds to $v_0 = (1, 1, 1, 1, 1...)$. Take for example $\lambda_{n/2} = -1$. $v_{n/2} = (1, -1, 1, -1, 1...)$.

What this Means?

One can look at this filter and see that it is a form of blur. It takes the two values next to a point and adds them. This also multiplies the magnitude of the list by 3, hence an eigenvalues of 3. This eigenvalues coresponds to a very low frequency. The smallest eigenvaluess, $\lambda_{n/2} = -1$ corresponds to the highest frequency. The blur is passing through low frequencies and reducing high frequencies.

Convolution in 2d

One can extend the above by thinking about an image as a flat vector. This makes it quite hard to think about kernels that act in both x and y. This is why 2d convolution makes a lot of sense.

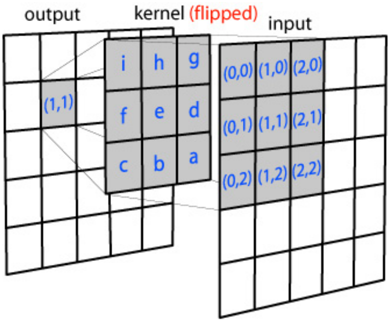


Figure: Convolution in 2D.

Doing this allows for a much more concise way to write down convolution kernels.

Gaussian Blur

Gaussian blur is an application of convolution. It 'blends' a Gaussian, a normal curve onto the image. One has control over blurriness by the standard deviation, σ.

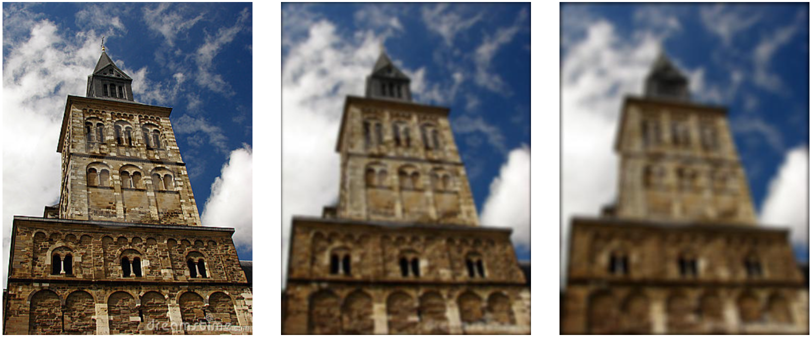


Figure: Original image on left. $\sigma = 2$ in middle. $\sigma = 4$ on left.

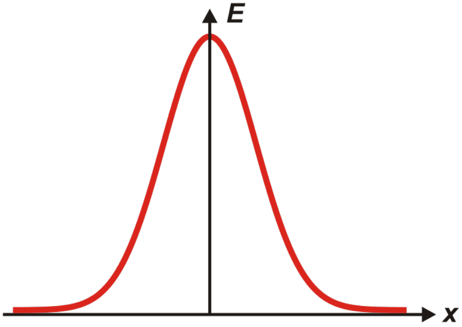
Gaussian Function

The Gaussian function in 2d is:

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

In 1d it is:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}}$$

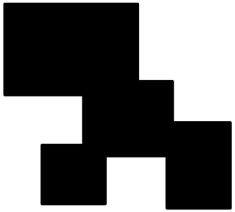


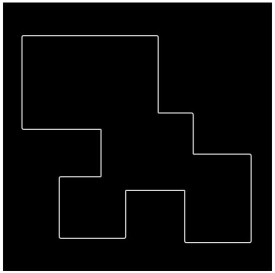
Eigenvalues

The eigenvalues of this Gaussian kernel preforms a form of bluring operation. Because of this, we know that its dominant eigenvalue is λ_0 which corresponds to a eigenvector which is low frequency. And it less dominant eigenvalues, $\lambda_{n/2}$ corresponds to high frequency.

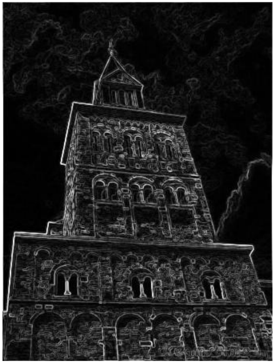
Sobel Edge Detection

Another application of convolution is edge detection. The Sobel operator can be thought of as a discrete differentiation operator. It gets how fast one color goes to another.





Example



Sobel Kernel

The kernel used to create this edge detection is in two parts:

$$G_X = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \; G_Y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Where G_x is how quickly the colors change in the x direction and G_y is in the y direction.

Where the output of the convolutions combined (by taking the magnitude)is the edge detected image.

Example of G_x and G_y Output

After G_x and G_y are convolved with every element in the image matrix the output looks like:

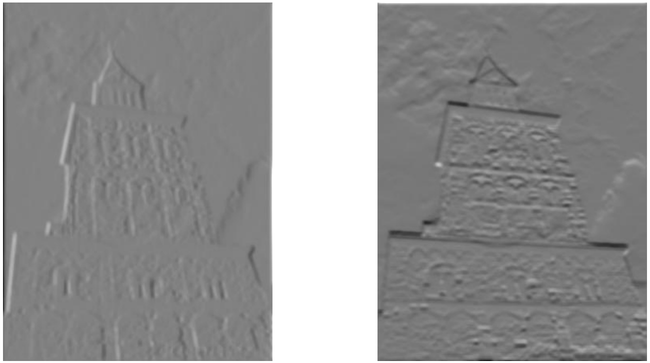
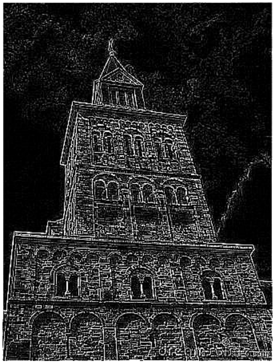


Figure: G_x and G_y respectively

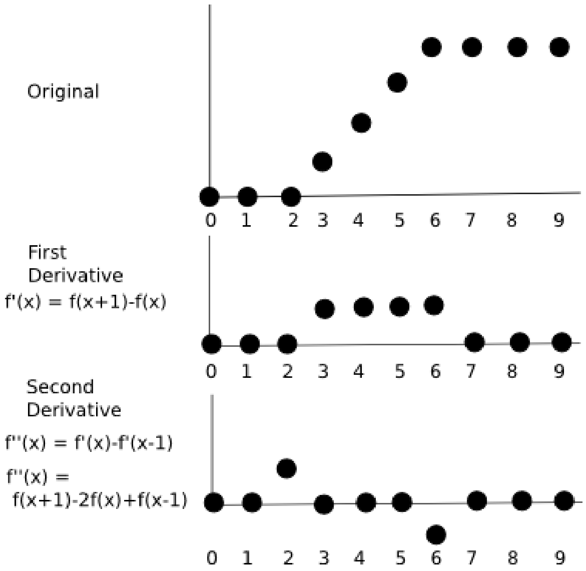
Laplacian kernel

A discrete Laplacian kernel can also be used to do edge detection.

$$K_L = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$



Laplacian in 1d



The kernel for this is simply [1, -2, 1].

The eigenvalues for this kernel is

$$\lambda_j = -2 + 2\cos(\frac{2\pi j}{n})$$

. As one can see here, the dominant eigenvalue is when $j = n/2$. This will yeild an eigenvector of

$$v_{n/2} = (1, -1, 1, -1, ...)$$

This makes a lot of sense when thinking about what edge detection does. It looks to places with very large, fast, high frequency changes in color and keeps those. It lowers all parts that are at low frequency, j=0, $\lambda_j = 0$, $v_0 = (1, 1, 1, 1, ...)$
Another interesting note here, is the nullspace. The nullspace of this operator is a image with no edges, or just a flat image.

Questions

Hope you enjoyed our presentation.
Any questions?