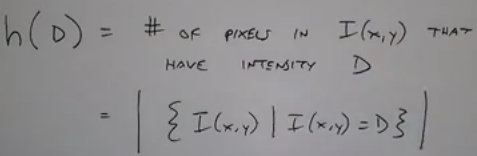
**RICH RADKE D.I.P**

**HISTOGRAMS AND POINT OPERATIONS**

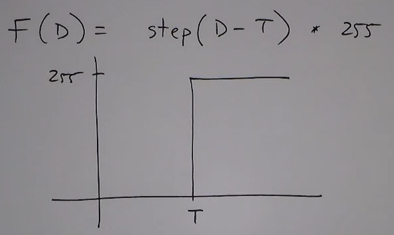
Used to improve contrast of washed-out image

Histograms are not one-to-one mapping of an image, because different images can have same histograms. It is not unique to an image.

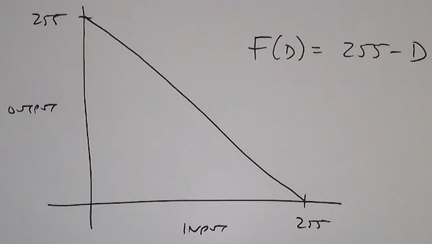


Mathematical representation of histogram

**Threshold** is a point operation



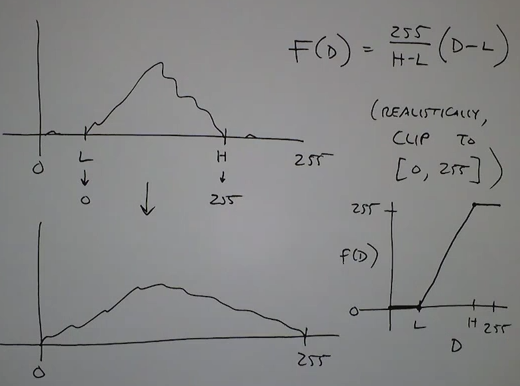
**Digital negative (inverting)**



**Contrast stretching:**

Uses the entire histogram. Sometimes if a portion of the image (like a frame/photo) is washed out you can also squeeze the histogram to narrow down on those details.

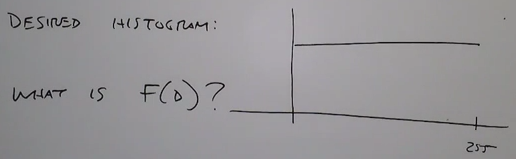
The first portion 255/(H-L) is the slope of the line



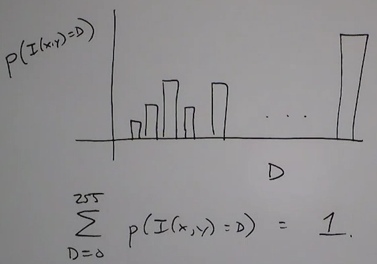
**Histogram equalization**:

In some images contrast stretching alone would not work. Because if you do so either dark portions become darker or bright portions become washed out. Hence a linear mapping of pixels will not help,

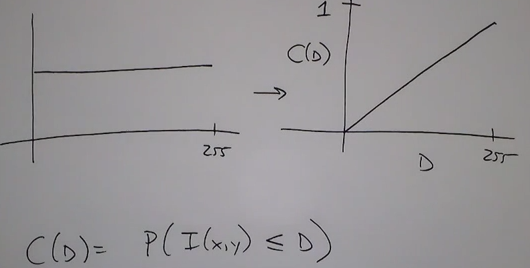
Non-linear mapping helps in such cases.



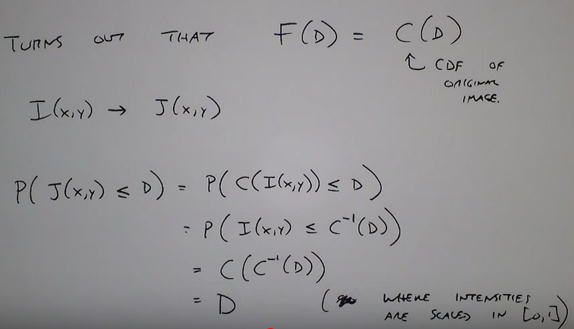
Basic idea is to think of image histogram as a probability mass function.

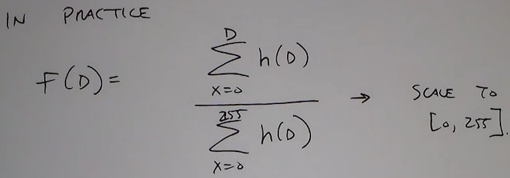


We should be using the cdf:

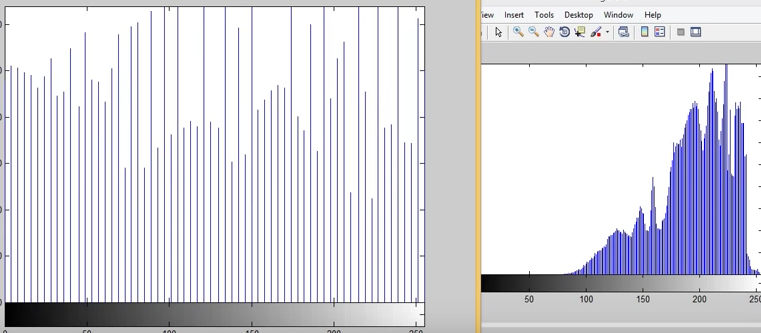


We want our image histogram to be mapped in the above manner.



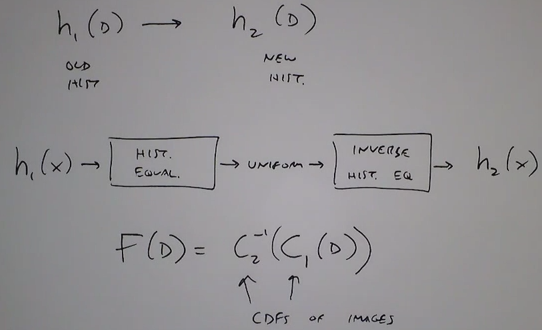


Comparison of original histogram and equalized histogram:



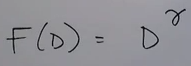
Histogram equalization is applied equally to all pixels having the same intensity. Hence pixels in biger bars cannot be combined (on right) but pixels in smaller bars can be combined (on left). Hence darker regions of image will not be clear while brighter portion of image would have been enhanced.

**Histogram Specification:**

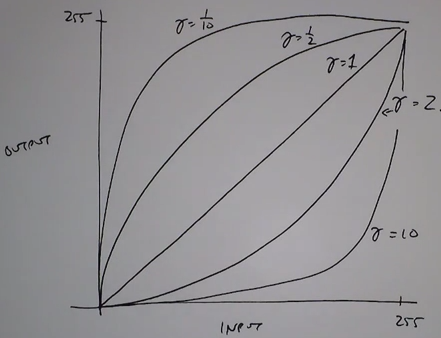


**Gamma Correction**

Every display device has a different non-linear relationship between pixel input intensity and display output luminance. (relation between current and voltage not linear in the cathode tube)

The relationship for real devices is often modeled as a power function: 

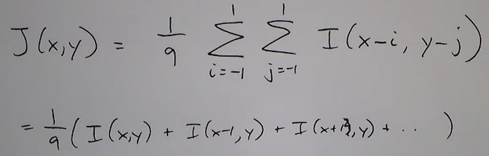
(Gamma correction is an act of compensating the input so that the output looks in the manner desired.)



If we know the ‘gamma’ value of the display device then we can precompensate the intensities.

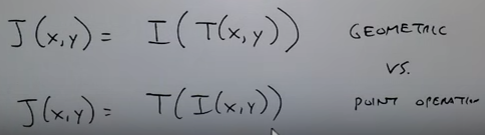
All above mentioned point operations affect every pixel with the same intensity in the same way, these were global operations. So we go for operations that are more local, eg: spatial filters.

Eg of spatial filter replace each pixel in image by the average of all its neighbors:



**GEOMETRIC OPERATIONS**

Different from point operations. Here the position of pixels change. In point operations the intensities of pixels change:



1. ***Translation:***



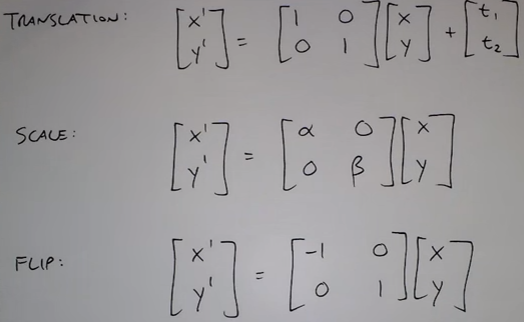
1. ***Scaling:***



1. ***Flipping:***

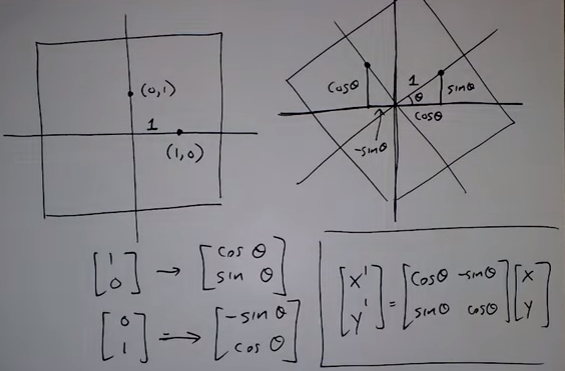


Representing in 2D



Flipping is a special case of scaling.

1. ***Rotation:***



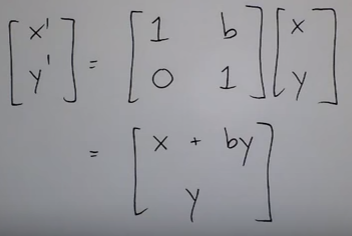
Any combination (scale, shift, rotate) of the above is called a similarity transformation.

It preserves parallel lines

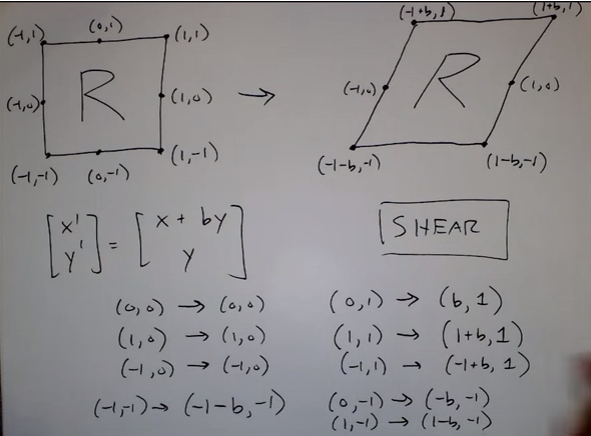
If alpha and beta are +/- 1 it is called isometric transformation

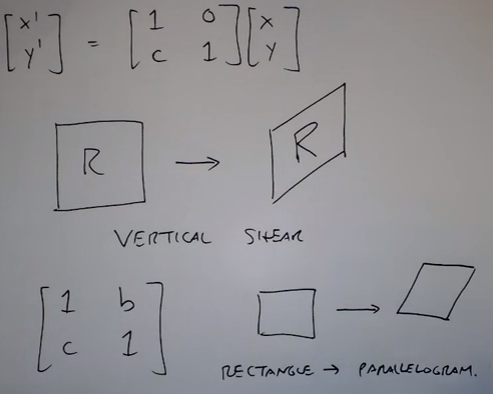
It preserves shapes and angles

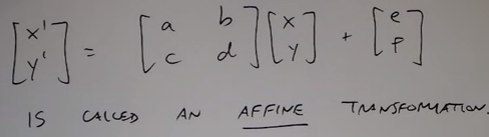
1. ***Bending:***



Shear:

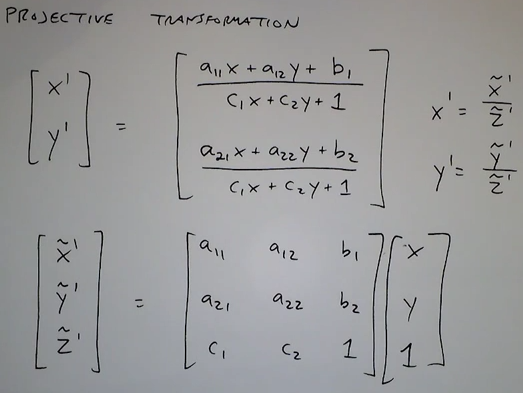








Projective Transformation:



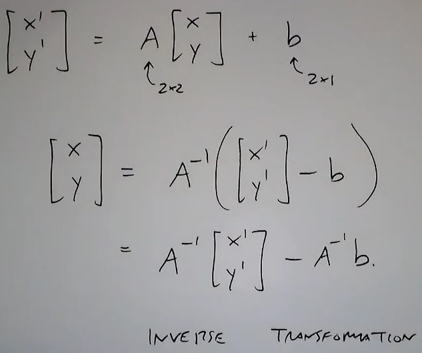
‘c1’ and ‘c2’ control how skewd an image is.

Sometimes you might not get the output image in integer, but some non-integer positions (below):



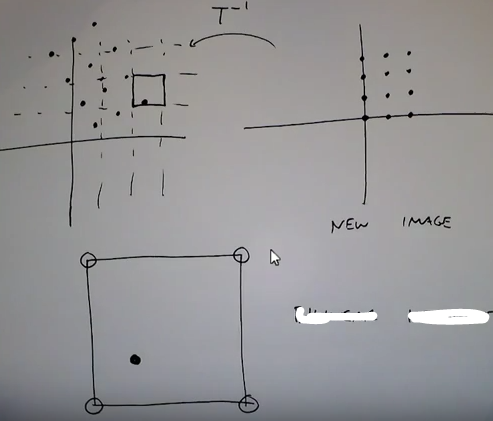
Question is how to get image color/intensities on the new grid?

It is conventional to use backward mapping.



Pixel intensities in the new image can be filled using either of the following operations:

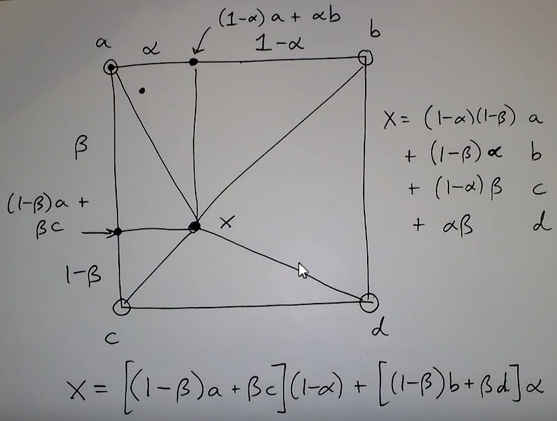
* Nearest neighbor interpolation
* Bilinear
* Bicubic



*Bilinear interpolation:*

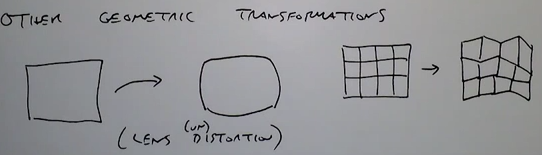
Now there are three positions where a pixel may fall:

* On the point exactly
* Between two points along a grid
* Somewhere in the square



*Bicubic interpolation:*

Consider 16 points as neighbors to compute intensity values. Result is a much more smoother image than bilinear but more computational.

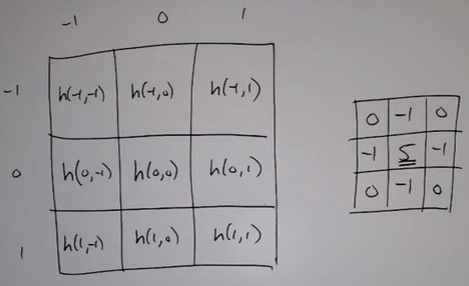


**SPATIAL FILTERING**

In 1-D it is called the ‘Time Domain’, with the central element of the filter called the ‘zero element’.

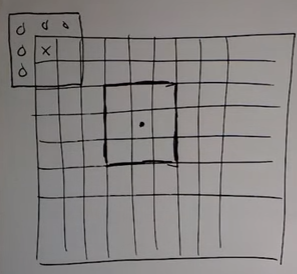
* [ -1, 2, -1]

In 2-D it is called the ‘Spatial Domain’



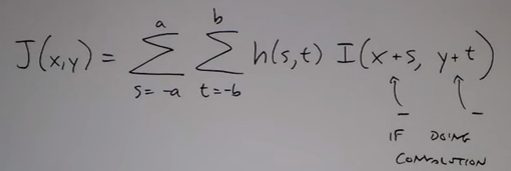
Result is focused on the center pixel.

Sometimes the filter will not cover the entire image, so you can do one of these two things:



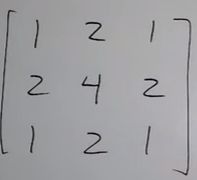
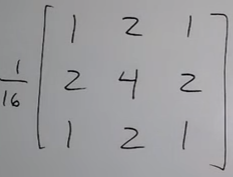
* Since there are not enough neighbors there won’t be any response
* Imagine there are zeros around the boundaries of the image, to get some value in the center

Convolution of two functions in spatial domain is equivalent to multiplication of Fourier transforms in the frequency domain.



***Smoothing filter (low pass filter)***:

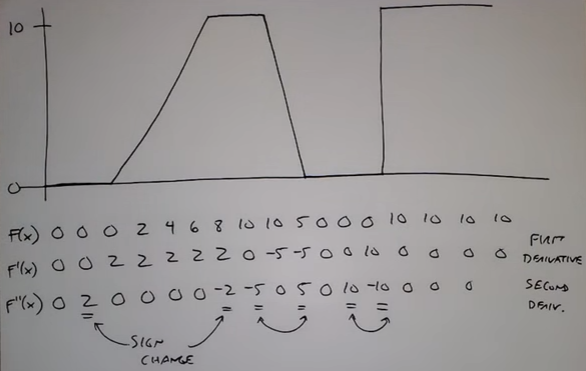
* Replace pixels by weighted average of its neighbors
* Reduces/removes noise (pro)
* Removes details/blurs image (con)
* Different values in the filter will give different results (enlightening, darkening, image, etc.)
* Divide by the sum of all the elements in the filter so that the intensity in the image corresponds to what it originally was.

=> 

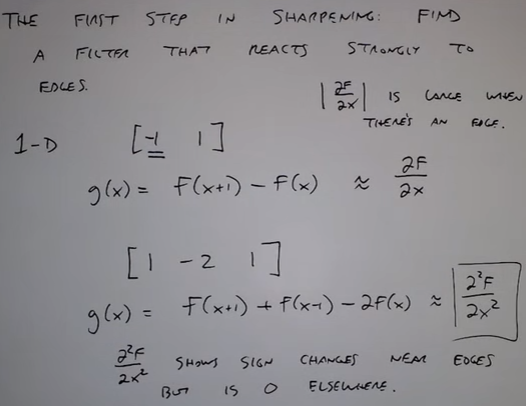
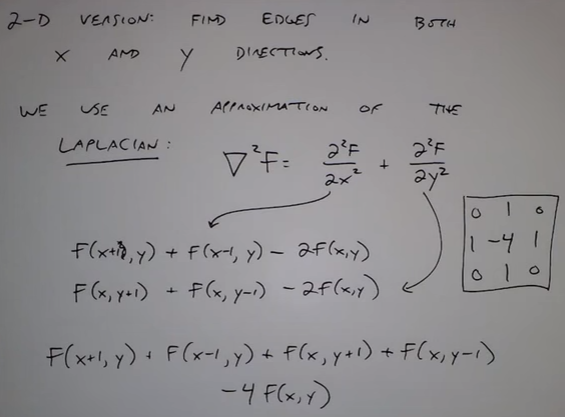
* Averaging in an image is basically an operation of integration

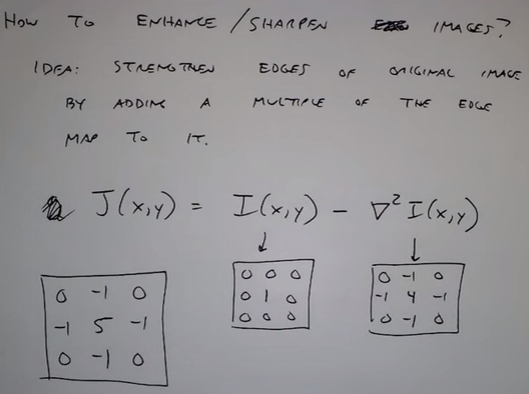
***Sharpening filter***:

* Sharpening is basically taking the difference of the pixels in image, (taking the derivative of the image).
* Consider the following distribution of pixels in an image:



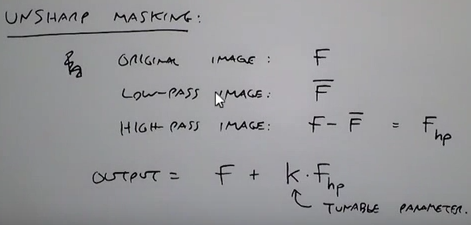
* Places where the difference is large means there is quite a lot of difference among the pixels.
* First derivative:
  + gives the change in pixels (edges).
* Second derivative:
  + gives where this edge starts/stops
  + there are also sign changes
* Sharpening is an act of enhancing edges

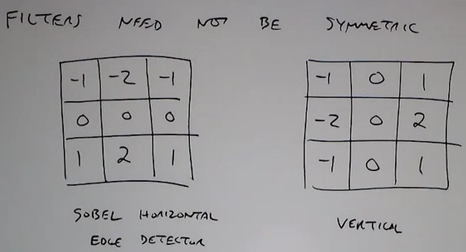


Enhancing the edges also enhances the noise(too much of color intensity is added).

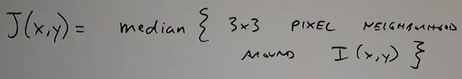
To avoid the noise being sharpened we use ‘unsharp mask’ technique.



We add a fraction of the edges back in for a more subtle effect.

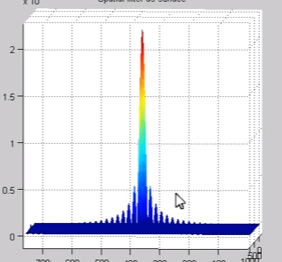


Not all filters are linear, eg: median filter: to eliminate ‘salt and pepper’ noise.

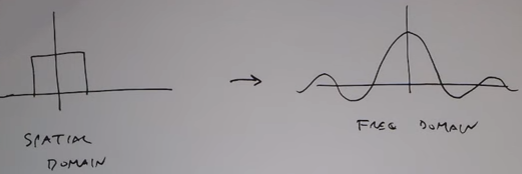


Median filtering is an example of non-linear filtering because there is no averaging done based on entire collection of pixels in filter but it is just a pixel-dependent choice making it non-linear filter.

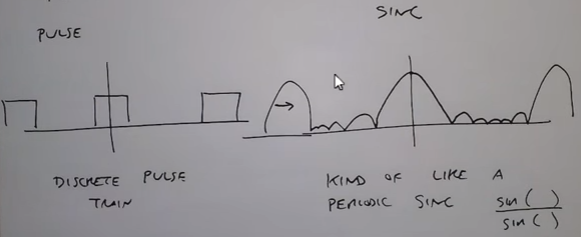
There is no smooth roll off but bumps in values.

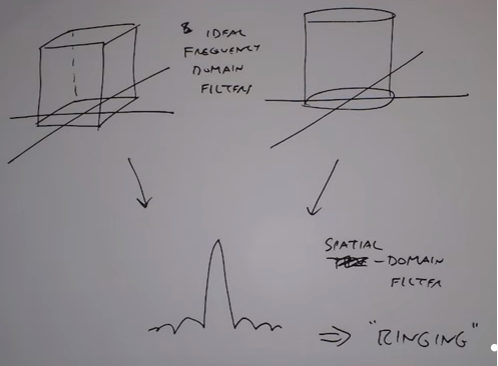


In general a pulse in spatial domain is a sinc function in frequency domain.



But with digital signals they periodically repeat themselves.



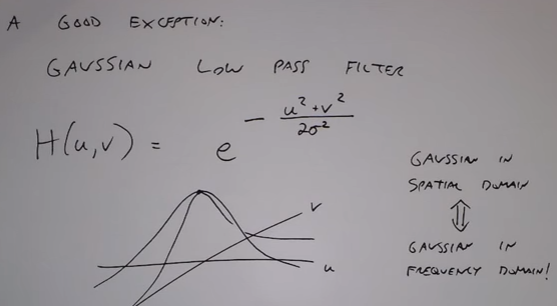


Basic Idea: smaller filter cut-offs in frequency domain (i.e. lesser frequency filters) end up having:

* More blurriness
* More ringing effect on images

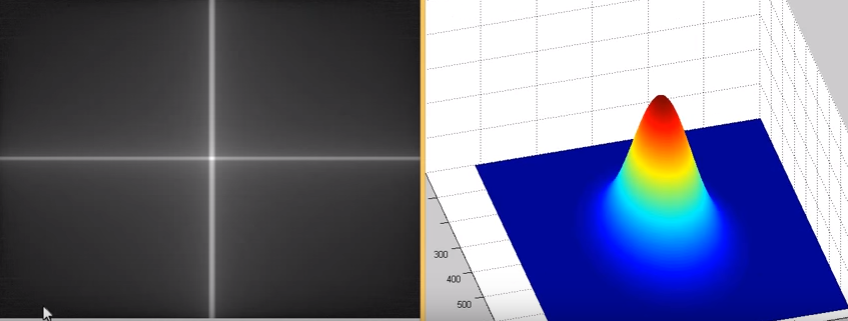
To avoid ringing effect, we can perform filtering in the spatial domain.

Something that works perfectly in both worlds (frequency and spatial domain).

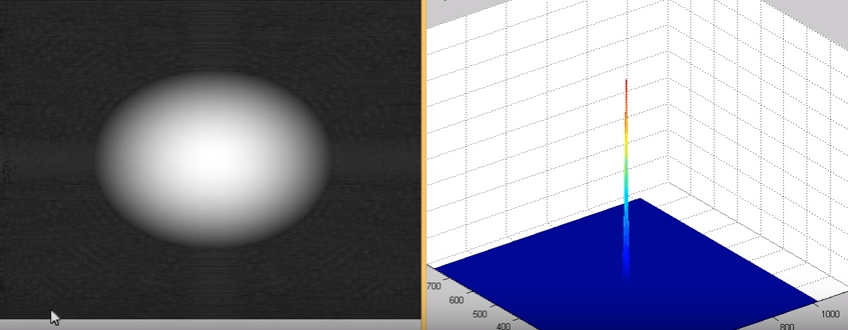


There is no ‘ringing’ effect with Gaussian filter in frequency domain.

1. For bigger radius/variance of the Gaussian in spatial domain => blurrier image

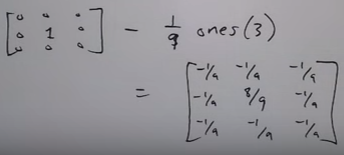


1. For smaller radius/variance of the Gaussian in spatial domain => less blurry image



***Highpass filtering***:

It is basically image minus lowpass filtering.



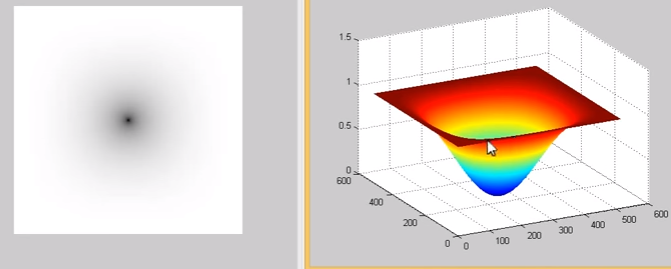
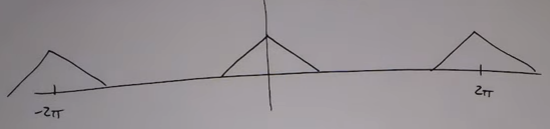
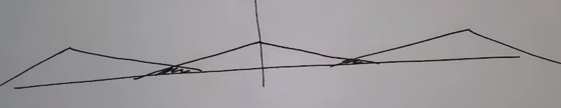




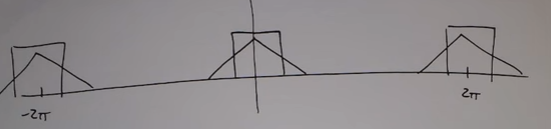
Image in the frequency domain:



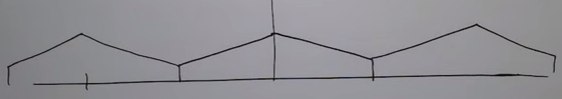
After being down-sampled:



So before down sampling the image filter with a low pass filter:



We avoid aliasing:



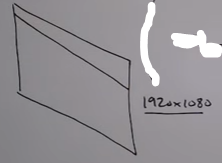
Effect of down-sampling:



To avoid it use low pass filter.

Note: in order to get the perfect trade-off between aliasing and losing information, come to the right cut-off frequency to be used in the low pass filter.

Anti-aliasing applications:

1. In text to convert cursive  to digital .
2. In video games imagine a line  which on down sampling becomes 

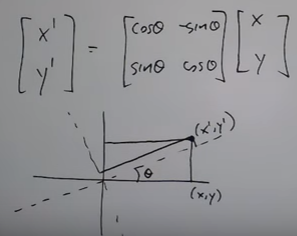
Solution:

* For down sampling the image having size (1920 x 1080) first increase the resolution of the image to ((1920 \* 4) x (1080 \* 4)).
* Filter each 4 x 4 square of the higher resolution image (averaging) or down sample each of the 4 x 4 pixels in the higher resolution image to avoid aliasing.

***Moire Patterns***

**UNITARY IMAGE TRANSFORMS (GENERAL IMAGE TRANSFORMS)**

Consider a point in 2D rotated by angle of ‘theta’.



The point can be represented by:

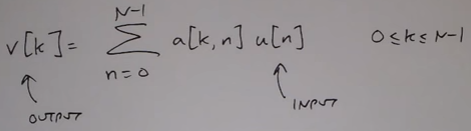
1. The present coordinates (x,y)
2. Or by the new coordinates (x’, y’)

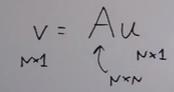
They ae both equivalent ways of representing the same 2D point because no information is lost.

So rotation is like a change of basis of the coordinate system.

Expanding this from 2D to a higher dimension we can use unitary transform.

UNITARY TRANSFORM:

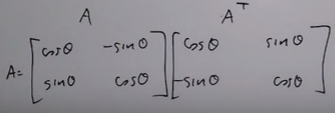




Here is ‘A’ is representing a change of basis from old ‘n’ numbers that represented ‘u’ to a new set pf ‘n’ numbers represent ‘v’.

If you see the rotation matrix above containing cos and sin ’theta’ it is represented by ‘A’.

Eg: ‘A’ is the rotation matrix ‘A(transpose)’ is the transpose of ‘A’. Multiplying the two we get a unit matrix.

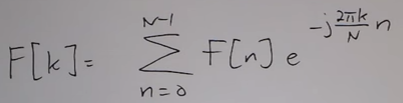


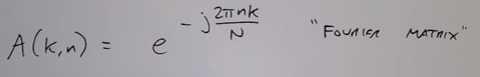
Columns of ‘A’ are in unit length and are perpendicular to each other.

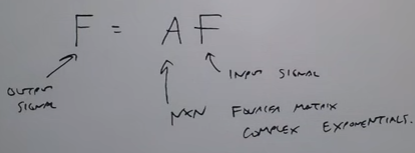
To establish unity: 

THIS is analogous to DFT. DFT is change of basis from time domain to frequency domain. Information is not changed. Instead of what the signal is doing at every point of time, we get content present at every frequency.

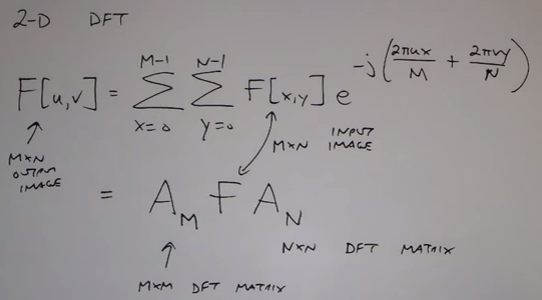
So in 1D we also have the DFT basis:



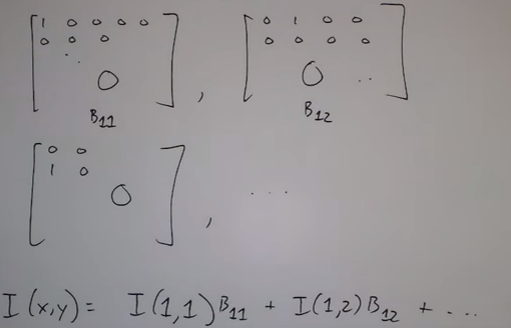
Here : 

Hence: 

We can do the same thing in 2D DFT:

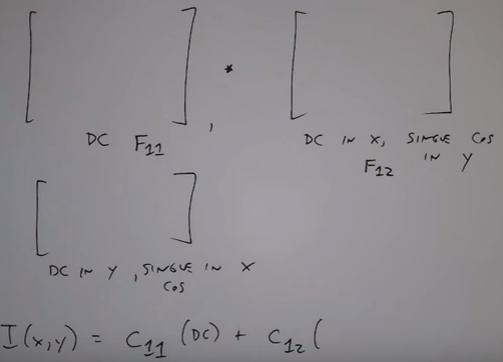


Spatial basis in 2D look like the following:



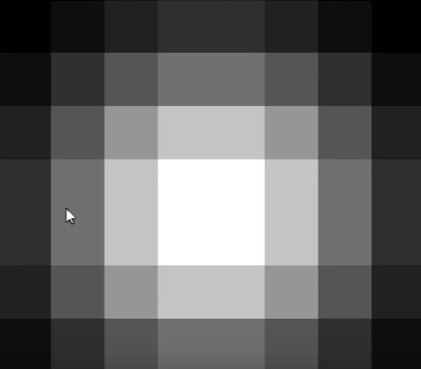
Here the value of the image is based on the scalar values I(1,1), I(1,2),…. and the basis functions B11, B12, … So since the basis functions in spatial domain do not overlap the value of the image can be obtained easily.

But things are more complex in the Fourier basis:

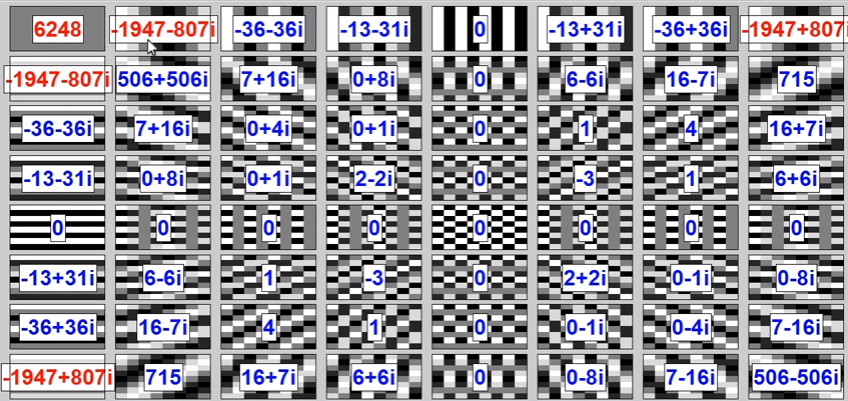


Here the value of the image is the composition of each of the individual matrices in a different way.

Fourier basis helps represent an image in the least amount possible. Therefore helping ni image compression.

An 8 x 8 image : .

The Fourier basis representation is:



So as we move from top left to top right and from left top to left bottom we see bigger values. But as we move inside the matrix, the values decrease. Hence these bigger values are alone enough to represent the image. But in spatial basis the contribution of each basis is EQUALLY important so no image compression is done in spatial basis.

One property satisfied by unitary transform is:” The sum of the squares of the original pixel values is the same as the sum of the squares of the transformed pixel values”. Hence, signal energy is preserved.

