                         

**Computer Vision**

Images are functions. So as we can add two functions, we can add noise as a function to the image.

Most common noise is Gaussian noise. So the more the sigma more noise we have.

So if we have a noise ridden image and we want to filter it what can be done? We take an average of nearby filter. To do this, we can take a odd shaped kernel (so that we have a centre pixel always) 1/9(3\*3 all 1s) this will give us blurred image as output. (This is known as correlation filtering) But it won't look good. A blurred blob is circular and hence we prefer a Gaussian kernel. And the process is called Gaussian filtering which gives better blur. The more the sigma, more the blur.

Impulse Response:

For images, let the impulse be a single pixel. Whats happens if we filter (correlation) with some random kernel( check the image) the kernel gets flipped upside down left right.

To get rid of this flipping, we can flip the kernel or the image before doing filtering. And that is called convolution (in dsp convolution is...) . So if we have a symmetric kernel like Gaussian, correlation and convolution gives same results.

Convolution is linear operation.

For everything to work operator should behave same way for pixels everywhere. The output should depend on pattern of pixel in neighbourhood, not position of neighborhood. To apply w\*w kernel on N\*N image, we will need N\*N\*w\*w multiplication.

We can utilize the linearly seperable property. (Ref Image) so our w\*w kernel we split into w\*1 and 1\*w. This computation goes down to 2w\*N\*N.

Types of convolution: full, valid, same

What to put in the boundary for valid convolution?

We can put 0s, but that will give black border.

We can wrap around, but that brings tinge of bottom of image on top.

We can copy edge, but the variability of image is affected.

So we can reflect the image and this gives the best result.

Using impulse filter, we get the same image as output.

If we have a filter with 1 shifted to right, we get a left shifted image (correlation)

If we have two filters ( ref Image) we get sharpening!!

Normalised correlation

You standardize the filter. Also scale (?confirm) the result after correlating filter and image.

So if we take a chunk of image as filter. And correlate it with the whole image. We get a maxima from the patch the filter was taken from. That's because filter and image are centered at zero. To get maximum, positive parts of image should be multiplied with positive of filter. This can be used for template matching.

Finding important features in image. Edge is enough for us to understand what's in image

To find edges, we can gradient of image to see where there is sudden jump in intensity. We need to define the % jump and the neighborhood window for the same.

Gradient become finite difference as image is discrete. Epsilon ->1. Now it matters which way we take the derivative. Also, we take derivatives in x,y direction. If we take RMS of these we get edge detection

 The common one is sobel operator. There are many more. We have discussed things change slowly in images hence we have particular Matrix structure for our filter. In real world, we have booze hence we need to smoothen the image before applying gradient.

 To find the gradient of an image, we just need to take gradient of the filter( by linear property)

 So smoothening + gradient can be equivalent to convolving image with derivative of smoothening function. This gives us maxima where there is an edge. But to detect exact edge, we can take second derivative and that will give us a zero crossing where there is an edge. This operator is called inverted mexican hat ( second derivative of smoothening)

In 2d, we need to look for the direction of the derivative as well.

So I can take derivative of my Gaussian smoothening filter and apply that to image and get derivative of smoothened image. More the smoothening, less edges we get. But how do we get edge from the gradients.

Note: Gradients won’t be {0, 255} but a range of values ranging [0,255] hence there is a need to threshold.

Enters canny edge. It calculates gradient, thresholds it. If we have high threshold we lose many edge, low threshold brings to much of details. So can't had two thresholds. The high threshold gives the initial edges. But that will miss some edges. So we find edges with a lower threshold and take the edges backed by the edges we got with higher threshold. (Linking operation) thinking the edges: non max Suppression. Didn't get this one ...

Another method is laplacian of Gaussian, Mexican hat filter. It's just the double derivative of Gaussian. But mostly can't performs better.