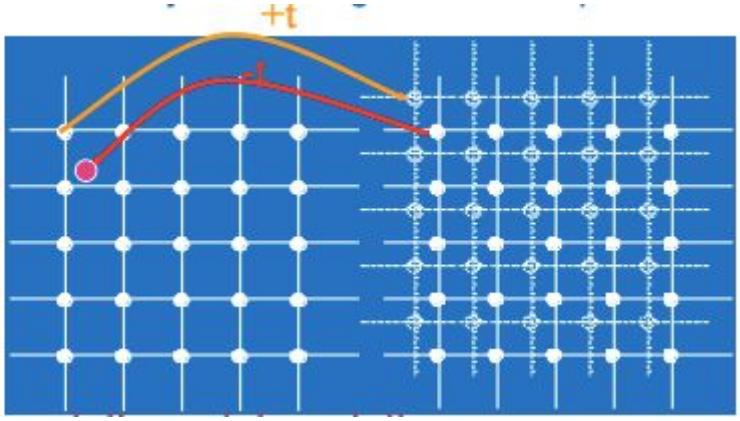
1 \ How to perform geometrical transformation of a digital image ?

Digital Image consists of a grid of pixels and can be represented as a 2D array. Therefore, performing a geometrical transformation of an image means mapping discrete pixel coordinates to a corresponding coordinate on the transferred image, but the corresponding coordinates may not correspond to discrete values.

In order to solve this issue. For each pixel (in the transferred image) , we apply inverse transformation to get original coordinate mapping, but the coordinate might be non-integer. Therefore, to get true value out of this non-integer we apply interpolation between the neighboring pixels.
There are many types of interpolation like (translation, rotation ..etc) we choose the one that fits well.

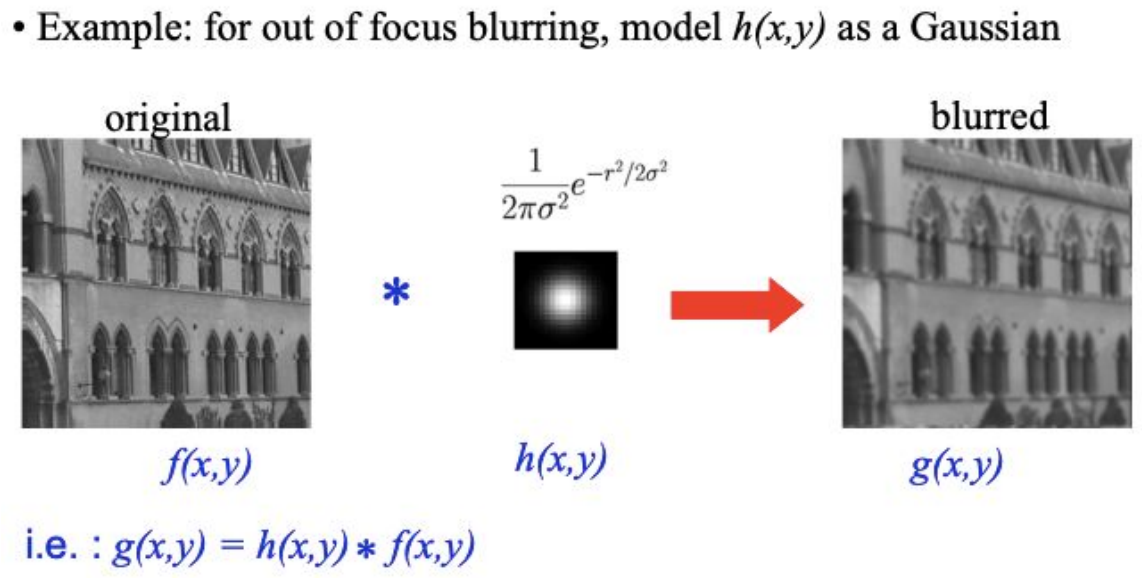
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$



2 \ Image Restoration:

The objective is to restore a degraded image to its original form.



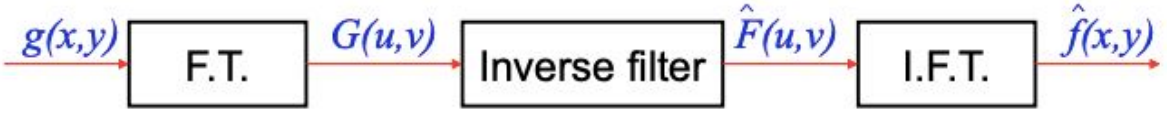
* Blurring acts as a low pass filter and attenuates higher spatial frequencies.

Inverse filtering assumes that degradation was caused by a linear function h(i,j). After applying the Fourier transform, we get

G(u,v) = F(u,v) . H(u,v)

The degradation can be eliminated using the restoration filter with a transfer function that is inverse to the degradation h. We derive the original image F (its Fourier transform to be exact) from its degraded version G as

F(u,v) = G(u,v) . $H^{-1}(u,v)$ == > $R(u,v) = H^{-1}(u,v)$

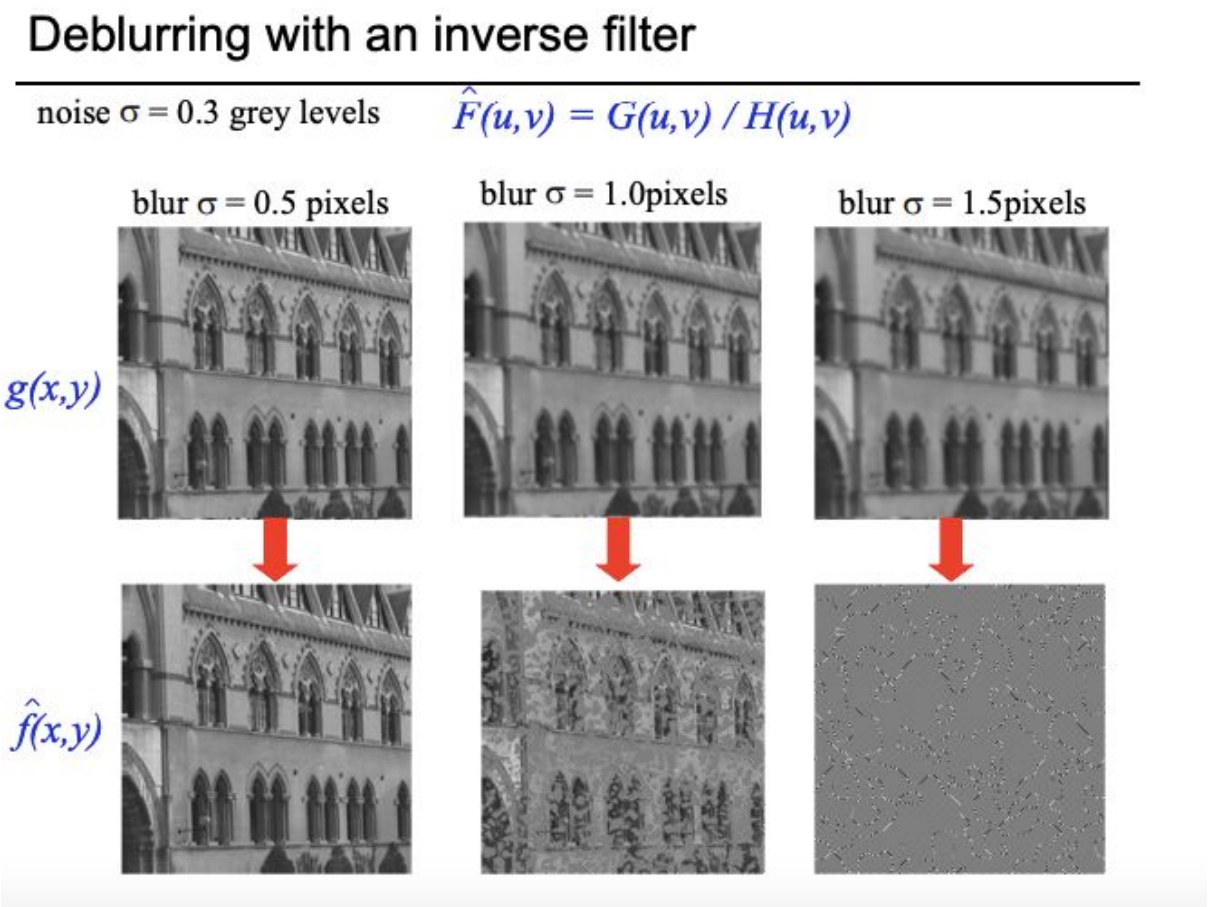


Therefore, we can do restoration using an inverse filter of low-filter which is a high filter to get the initial image.

- However, if noise is present, problems arise. First, the noise in uence may become significant frequencies where(u,v) has small magnitude.

- We usually do not have enough information about the noise to determine N(u,v) sufficiently.

- In the ideal case, we would just invert all the elements of **H** to get a high pass filter. However, notice that a lot of the elements in **H** have values either at zero or very close to it. Inverting these elements would give us either infinity or some extremely high values. In order to avoid these values, we will need to set some sort of a threshold on the inverted element



Wiener Filtering : Wiener filtering executes an optimal tradeoff between inverse filtering and noise smoothing. It removes the additive noise and inverts the blurring simultaneously

it minimizes the overall mean square error in the process of inverse filtering and noise smoothing

$$\varepsilon = E\left\{\left[f_i(x,y) - \hat{f}_i(x,y)\right]^2\right\}$$

- Calculating the 1st derivative, the error is minimal when

$$E\left\{\left[f_i(x,y) - \hat{f}_i(x,y)\right] f_o(x',y')\right\} = 0$$

- By replacing $\hat{f}_i$ by its value, we get

$$E\{f_i(x,y) f_o(x',y')\} = \int_{-\infty}^{\infty}\int E\{f_o(i,j) f_o(x',y')\}\ h_R(x-i, y-j)didj$$

- The expectations of this products are the intercorrelation and the autocorrelation of the variables:

$$K_{f_i f_o}(x-x', y-y') = \int_{-\infty}^{\infty}\int K_{f_o}(i-x', j-y')\ h_R(x-i, y-j)didj$$

$$K_{f_i f_o}(x-x', y-y') = \int_{-\infty}^{\infty}\int K_{f_o}(i-x', j-y')\, h_R(x-i, y-j)didj$$

- By FT, we obtain

$$H_R(\omega_x, \omega_y) = \frac{P_{f_i f_o}(\omega_x, \omega_y)}{P_{f_o}(\omega_x, \omega_y)}$$

$P_{f_i f_o}(\omega_x, \omega_y)$ is the power interspectrum
$P_{f_o}(\omega_x, \omega_y)$ is the power spectrum of f_o

And we finally obtain the Wiener filter, with frequency response:

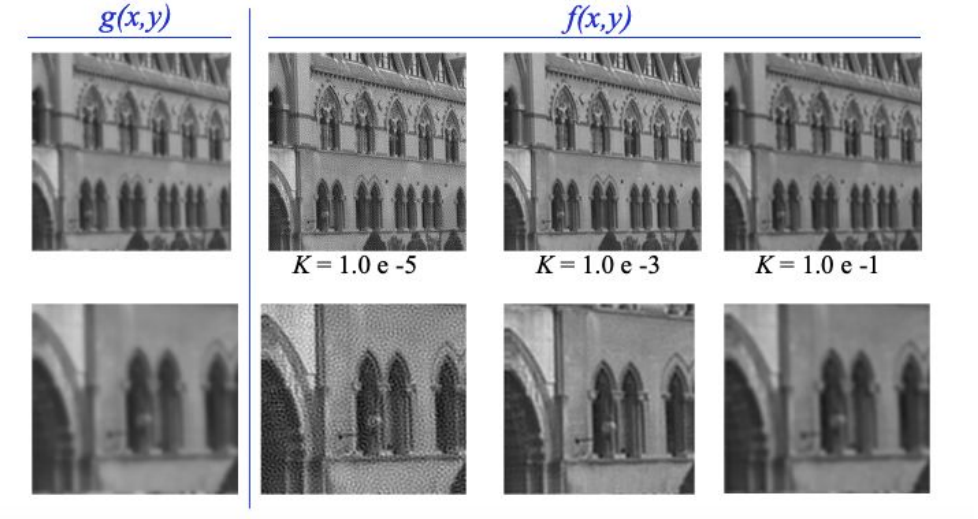
$$H_R(\omega_x, \omega_y) = \frac{H_D^*(\omega_x, \omega_y)}{\left|H_D^*(\omega_x, \omega_y)\right|^2 + \frac{P_N(\omega_x, \omega_y)}{P_{f_i}(\omega_x, \omega_y)}}$$

- If Wiener filtering is used, the nature of degradation H and statistical parameters of the noise need to be known.
- It behaves like the inverse filter at low frequencies and like a low-pass filter for high frequencies
    - If Pn / Pf >> |H(u,v)| for large u,v, then high frequencies are attenuated
    - If Pn / Pf = 0 then W(u,v) = 1 / H(u,v), i.e. an inverse filter

Example 1: Focus deblurring with a Wiener filter

blur σ = 1.5 pixels
noise σ = 0.3 grey levels

$\hat{F}(u,v) = W(u,v)\, G(u,v)$ $\quad W(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + K(u,v)}$



3\ Object Labeling and the algorithm to implement it ?

Support we are given a binary image, in order to do object labeling we have to give an input seed to our function, which goes through the images starting from [x, y] = [0, 0] position, when we encounter a certain pixel below the threshold given we perform a *region growing* and we label each object (group of pixels).

Region growing: is an iterative method used to extract similar parts of an image. One or several points are chosen as a start. The region then grows until it is finally blocked by the stop criteria.

The chosen criteria:

- it may be a difference between the outside pixel's intensity value and the region's mean. The pixel with minimum intensity in the region neighbourhood is chosen to be included. The growing stops as soon as the difference is larger than a threshold.
- The local variance is lower than a threshold (homogeneous texture)

Principe of region growing:

– Let us fix a starting point (seed) in the desired region

– Let us also define the homogeneity criterion used to define the region § e.g. intensity > threshold

– By a recursive procedure, (i.e. neighbor to neighbor), let us include in the region all the pixels that are neighbors of the current pixel and that satisfy the homogeneity criterion § By this the region will grow until it contains all the points connected to the seed point § We obtain a connex region

4\ What are the main principles of edge detection ?

An edge is a sharp transition of intensity in an image, also could be defined as :

- Where the intensity profile is like a step function.
- Where the 1st derivative has a maximum.
- Where the second derivative crosses zero (zero-crossing)

In order to do edge detection, the goal is to measure the edge sheerness, and the methods used can be divided into 3 main categories:

- 1\ Operators approximating derivatives of the image function using differences. Some are rotationally invariant (eg. Laplacian) and thus are computed from convolution masks. Others approximate the first derivative (they try to maximize the first derivative) like:

Robert Operator: is very easy to compute as it uses only 2X2 neighborhood of the current pixel. It masks are :

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad |g(i,j) - g(i+1,j+1)| + |g(i,j+1) - g(i+1,j)|$$

Its disadvantage is that it is very sensitive to noise because a very few pixels are used to calculate the gradient.

Prewitt Operator: is 3X3 convolution mask, the result of the convolution result the greatest magnitude indicates the gradient direction. Also, the gradient is estimated by 8 masks possible direction. The direction of gradient is given the mask which gives maximum response.

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}, \quad h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad \dots$$

- 2\ Operators based zero-crossing of the image function second derivative (eg canny edge, or Laplacian of Gaussian) :

  Marr-Hildreth " Laplacian of Gaussian ": is a very popular operator approximating the second derivative. Idea behind: We know that the first derivative of the image function should have an extreme at the position corresponding to the edge in the image, therefore the second derivative should be zero at that position. So it is much easier and precise to calculate that rather than calculating the maximum.

  It works as follows:

- Noise removing by filtering the image with Gaussian filter for example :

$$G(x, y) = exp((x^2 + y^2)/ 2 * theta^2)$$

Standard deviation is the only parameter Gaussian smoothing filter, which is proportional to the size of the neighborhood pixels on which the filter operates. It does not affect very far pixels which corresponds well to the smoothing filter criteria set by Marr-Hildreth.

- Calculate the 2nd derivative of the filtered image : our goal is to calculate now the second derivative of the image, we know that the Laplacian operator can do that, which is non-directional (isotropic).

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

So considering calculating the Laplacian of a smoothed image (by Gaussian Filter), we get what's called **Laplacian of Gaussian**.

Note: in order to calculate that, we can use the property of convolution which is

$$\nabla^2\big(G(x,y) ** f(x,y)\big) = \big(\nabla^2 G(x,y)\big) ** f(x,y)$$

Thus, it means that we just have to convolve the image by the second derivative of a Gaussian « Laplacian of Gaussian », which can be computed analytically.Therefore, the edge will be at the zeros.