Experiment 3

Spatial Filtering

Spatial filtering is a technique for modifying or enhancing an image. For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and edge enhancement. Filtering is a *neighbourhood operation*, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighbourhood of the corresponding input pixel. A pixel's neighbourhood is some set of pixels, defined by their locations relative to that pixel. A filter is defined by a kernel, which is a small array applied to each pixel and its neighbours within an image. In most applications, the centre of the kernel is aligned with the current pixel, and is a square with an odd number (3, 5, 7, etc.) of elements in each dimension. The process used to apply filters to an image is known as *convolution*, and may be applied in either the spatial or frequency domain.

Problem Objective:

Q 1. Write C/C++ modular functions/subroutines to design spatial filters - **Mean, Median, Prewitt, Laplacian, Sobel kernels (horizontal, vertical, diagonal), Gaussian Blur, Laplacian of Gaussian** on a stack of grayscale images (say, 15 images per stack).

> Use OpenCV for image reading, writing and showing only. Use the OpenCV tracker (slider) functionality to show outputs for varying sizes of neighborhoods. You may have different sliders to select (i) Image (ii) Filter (iii) Neighborhood size (3x3, 5x5, 7x7 and 9x9)

Input: Path to the stack of images. Input stack **should** contain the (provided) noisy images, and may also contain the normal test images, e.g. jetplane.jpg, lake.jpg, livingroom.jpg, mandril_gray.jpg, pirate.jpg, walkbridge.jpg

Output: Filtered stack of images should be shown beside input stack in the same pane of GUI with a slider to vary filter/kernel size/change image.

Note: For reporting the output choose your input image wisely, your choosed input image for to the show the output of each filter should reflect your understanding about the primary application of that particular filter.

Q 2. Create a filter called ***Gaussian_Unblur*** to undo the effects of blurring. It can be implemented by executing the following iterative steps:

Let I_0 be the blurry input image, I_k be the corrected image at iteration k, and G_σ a Gaussian filter. Iterate the following steps over $k = 0, 1, 2, \ldots$

1. Compute $A_k = I_k * G_\sigma$ (convolution)
2. Set, $B_k = I_0 / A_k$ (pixel by pixel division)

3. Compute $C_k = B_k * G_\sigma$ (convolution)
4. Set, $I_{k+1} = I_k * C_k$ (pixel by pixel multiplication)

You should run these steps until the image I_k converges, that is, the change from one iteration to the next is very small (choose a small value). Set a maximum iteration count to bail out just in case it doesn't converge.

Input: Choose any of the given image, and apply your previously implemented Gaussian blur filter with $\sigma = 1$. Then use the blurred image as input for this question. For unblurring also use $\sigma = 1$

Output: The corrected image.

Note

1. Do not hardcode the filenames and/or image size into the code.
2. Use proper code commenting and documentation.
3. Use self-explanatory identifiers for variables/functions etc.

References

1. Gonzalez, Woods "Digital image processing" 3/e, Chapter 3, Prentice Hall.
2. NPTEL Lectures on Digital Image Processing by Prof. P.K.Biswas.