CSL461: Digital Image Analysis

Semester I, 2017 – 2018

Programming Assignment 2: Spatial Filtering

(***Due Date/Time: Monday, 25th Sep 2017, Midnight***)

- **Aim**

  To give students hands-on experience with implementing Spatial Filtering methods

- **Introduction**

  - Bilateral Filter: is a non-linear, edge-preserving, and noise-reducing smoothing filter for images. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. This weight can be based on a Gaussian distribution. Crucially, the weights depend not only on Euclidean distance of pixels, but also on the radiometric differences (e.g., range differences, such as color intensity, etc.). This preserves sharp edges.

    The bilateral filter is defined as[1][2]

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|),$$

    where the normalization term

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

    ensures that the filter preserves image energy and

    I^{filtered} is the filtered image;

    I is the original input image to be filtered;

    x are the coordinates of the current pixel to be filtered;

    Ω is the window centered in x;

    f_r is the range kernel for smoothing differences in intensities (this function can be a Gaussian function);

    g_s is the spatial kernel for smoothing differences in coordinates (this function can be a Gaussian function).

    As mentioned above, the weight W_p is assigned using the spatial closeness and the intensity difference.[2] Consider a pixel located at (i, j) that needs to be denoised in image using its neighbouring pixels and one of its neighbouring pixels is located at (k, l). Then, the weight assigned for pixel (k, l) to denoise the pixel (i, j) is given by

$$w(i, j, k, l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|I(i, j) - I(k, l)\|^2}{2\sigma_r^2}\right),$$

    where σ_d and σ_r are smoothing parameters, and $I(i, j)$ and $I(k, l)$ are the intensity of pixels (i, j) and (k, l) respectively.

    After calculating the weights, normalize them:

$$I_D(i, j) = \frac{\sum_{k,l} I(k, l) w(i, j, k, l)}{\sum_{k,l} w(i, j, k, l)},$$

    where I_D is the denoised intensity of pixel (i, j).

- As the range parameter σ_r increases, the bilateral filter gradually approaches Gaussian convolution more closely because the range Gaussian widens and flattens, which means that it becomes nearly constant over the intensity interval of the image. As the spatial parameter σ_d increases, the larger features get smoothened.

- Olympic Filter: is closely related to mean filter, and therefore, can be used to emphasize the longer-range variability in an image, effectively acting to smooth the image. This can be useful for reducing the noise in an image. The algorithm operates by calculating the average value in a moving window centered on each grid cell. In estimating the average, the highest and lowest values in the neighborhood are not used. Because averages are highly affected by extremely high and low values, the Olympic filter is far less sensitive to shot noise in an image than the mean filter.

- Adaptive Filter: can be used to reduce the level of random noise (shot noise) in an image. The algorithm operates by calculating the average value in a moving window centered on each grid cell. If the absolute difference between the window mean value and the center grid cell value is beyond a user-defined threshold, the grid cell in the output image is assigned the mean value, otherwise it is equivalent to the original value. Therefore, the algorithm only modifies the image where grid cell values are substantially different than their neighboring values.

- **Implementation**

  - You will be implementing each of the above described spatial filters
    - `[outImg] = myBilateral( inImg, wSize, ANY OTHER REQUIRED PARATMERS!!);`
    - `[outImg] = myOlympic( inImg, wSize, ANY OTHER REQUIRED PARATMERS!!);`
    - `[outImg] = myAdaptive( inImg, wSize, ANY OTHER REQUIRED PARATMERS!! );`

  Where:

  `inImg` – Input Image
  `outImg` – Output Image
  `wSize` – Window Size

1. Show the performance of these filters on sample input images (`TestPA2.m`)
   a. For each sample image, display the results in 3 x 5 subplot with each row display results from different filters and each column displaying results with varying filter parameters (filter size, variance, etc.)
   b. There should be a pause for user to evaluate the results
   c. When resumed, the figure should be closed and new figure should be displayed with next sample image.
   d. This process then should be repeated for all sample images
2. Comment on the trends you observe for different values of input parameters of the filters in README file.

- **Submitting your work:**
  - All source files and class files as one tar-gzipped archive.
    - When unzipped, it should create a directory with your ID. Example: **P2008CS1001–PA1** (NO OTHER FORMAT IS ACCEPTABLE!!! Case sensitive!!!)

        - ***Negative marks if the TA has to manually change this to run his/her scripts!!***
    - Source / class files should include the following: (Case-Sensitive file names!!)
        - `TestPA2.m, myBilateral.m, myOlympic.m and myAdaptive.m`
        - **README** (Should provide all the necessary details to the TA for ease of grading – what works, what doesn't, any assumptions made, etc.)
        - *Any other utility functions you might end up creating*
    - ***Negative marks for any problems/errors in running your programs***
    - Submit/Upload it to Moodle

- **Grading Scheme : 50 Points (Total)**
    - Bilateral Filter (10 Points)
    - Bilateral Filter – Color Images (10 Points)
        - Must include your own conversion to CIE-Lab color space!
    - Olympic Filter (5 Points)
    - Adaptive Filter (5 Points)
    - Test Script (10 Points)
    - README (5 Points)
    - Code + Comments (5 Points)