Homework 2: Edge-Aware Filtering and Image Denoising

# Bilateral filtering

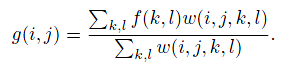
## Background and matlab code

The basic idea underlying bilateral filtering is to do in the range of an image what traditional do in its domain. Two pixels can be close to one another, that is, occupy nearby special location, or they can be similar to one another, that is, have nearby values.

Combining intensities from the entire image makes little sense, since the distribution of image values far away from p ought not to affect the final value at p. In addition, one can show that range filtering without domain filtering merely changes the color map of an image, and is therefore of little use. The appropriate solution is to combine domain and range filtering, thereby enforcing both geometric and photometric locality.

Combined domain and range filtering will be denoted as bilateral filtering. It replaces the pixel value at p with an average of similar and nearby pixel values. In smooth regions, pixel values in a small neighbourhood are similar to each other, and the bilateral filter acts essentially as a standard domain filter, averaging away the small, weakly correlated differences between pixel values caused by noise.

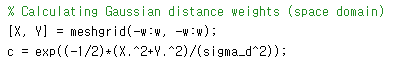
In the bilateral filter, the output pixel value depends on a weighted combination of neighboring pixel values:



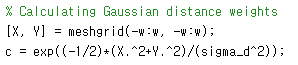
The weighting coefficient w(I,j,k,l) depends on the product of a domain kernel:



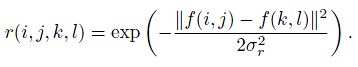
For greyscale image:



For color image:



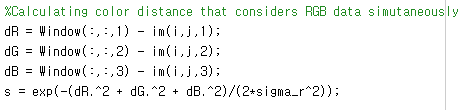
And data-dependent range kernel:



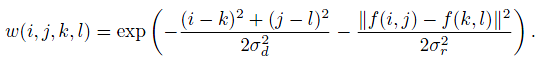
For greyscale image:



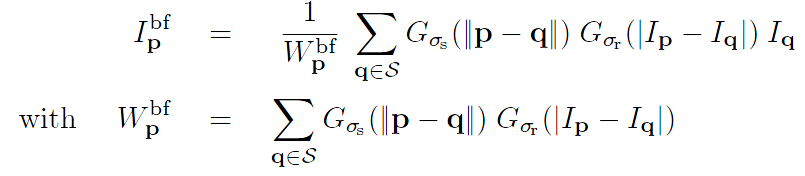
For color image:



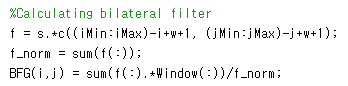
When multiplied together, these yield the data-dependent bilateral weight function:



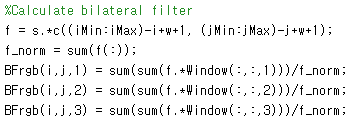
Or in compact way:



For greyscale image:



For color image:



The parameter σs defines the size of the spatial neighbourhood used to filter a pixel, and σr controls how much an adjacent pixel is downweighted because of the intensity differences.

So:

As the range parameter σr increases, the bilateral filter becomes closer to Gaussian filter because the range Gaussian is flatter, i.e. almost a constant over the intensity interval covered by the image.

Increasing the spatial parameter σs smooths larger features.

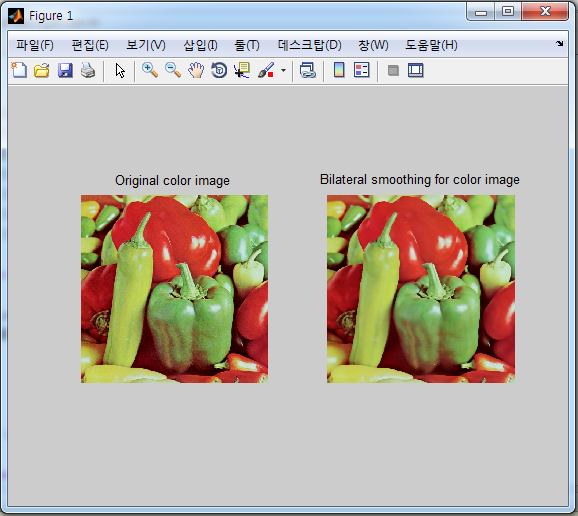
And base on the formulas above show that when the window size increases, the execution time also increases, since the number pixel in the window getting bigger and we need to calculate the difference in space domain and range domain for each pixel in the window, which makes the number of calculation also increases.

## Result

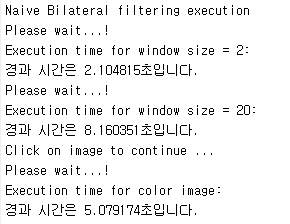
### Greyscale result:

|  |  |
| --- | --- |
|  |  |

### Color smoothing result:



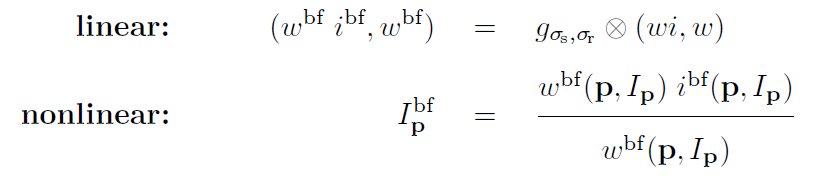
### Timing result:



# O(1) time Bilateral Filtering

## Background and matlab code

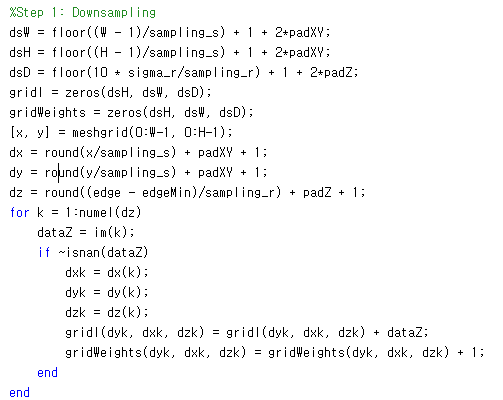
The bilateral filter is expressed as a convolution followed by nonlinear operation:



In practice,

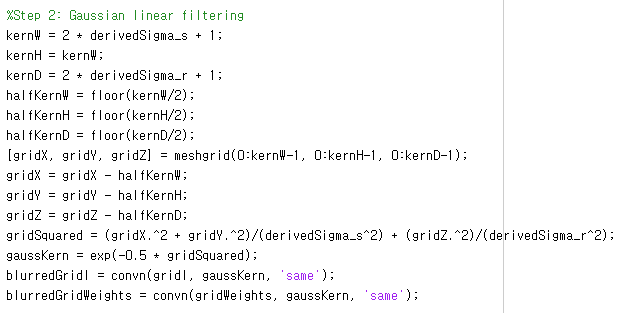
They downsample (wi, w)



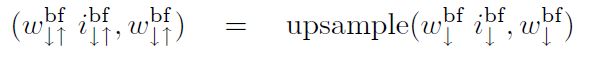


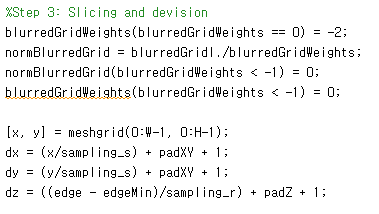
Perform the convolution





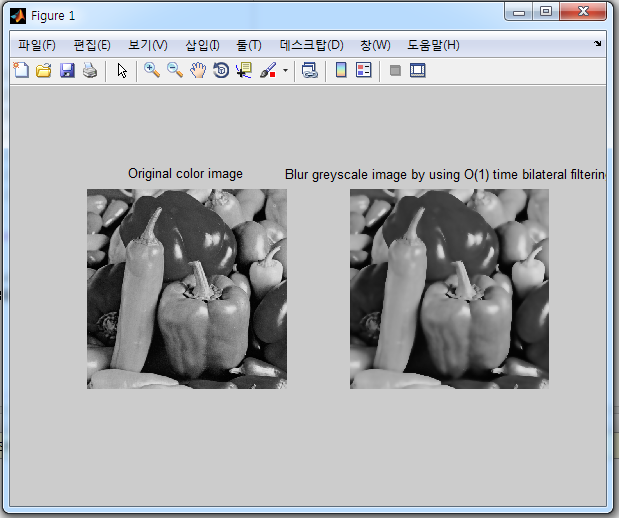
And upsample the result



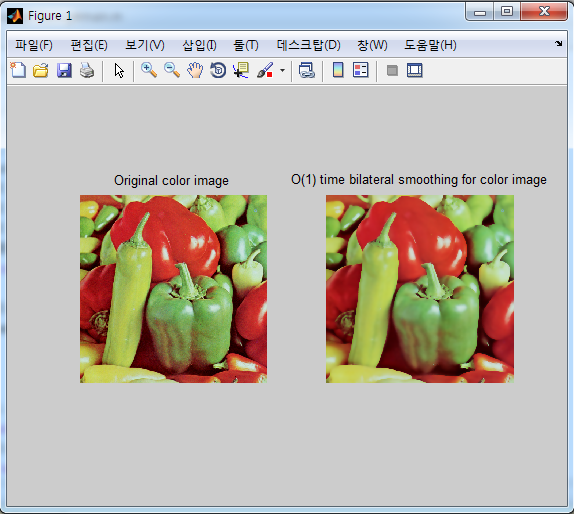


## Results

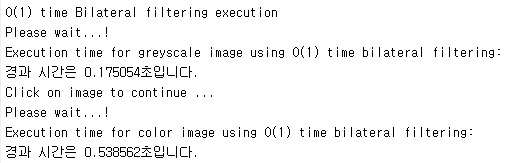
### O(1) time bilateral filtering on grayscale image



### O(1) time bilateral filtering on color image:



### Timing results



# Comparing

As the result, we can see that O(1) time bilateral filtering run really faster than naïve bilateral filtering.

To compare an objective quality, we need to measure the PSNR between two filtering results of the original bilateral filtering and O(1) time bilateral filtering:



The bigger the PSNRs are, the more similar the two images are.

According to paper, sampling step specifies the amount of downsampling used for the approximation. Higher values use less memory but are also less accuracy. The default values equal to sigma\_s and sigma\_r produce reults without visual difference with the exact computation.