Bilateral Filtering

A
Mini Project Report

by

Ragja Palakkadavath(SC18M003)

Department of Mathematics
Indian Institute of Space Science and Technology, Trivandrum

Semester-I

Acknowledgement

I would like to thank Dr. Deepak Mishra Sir for all his guidance through the completion of the project and also for giving me an opportunity to present a mini project in my M. Tech course of Digital Image Processing.

I would like to express my gratitude to the TA's of this course, Sowmya and Minha for guiding me to the right path to a good choice of topic for this mini project.

Lastly I would like to thank my family, peers and friends for all their support, encouragement and help in the completion of the seminar.

Contents

List of Figures

List of Tables

1 Introduction

Blurring or Smoothing filters which are used for image de noising tend to blur the image completely. Due to this, along with the noise, the intensity of the pixels present in the image also gets blurred hence leading to loss of important features from the image. Edge adaptive filter that overcomes this issue however does not consider the relationship between intensity of pixel in the given neighborhood. This is where Bilateral Filtering comes into use. It is a method of filtering the image while still preserving the edges and important characteristics of the image.

The basic idea underlying bilateral filtering is the fact that two pixels can be close to one another in two ways. One is, that they occupy nearby spatial locations, and the second is that they can be similar to one another, that is, have close enough intensity values, possibly in a perceptually meaningful fashion. An example can be an image of a face. The intensities that correspond to a feature, say forehead will be having similar intensities, and also, those pixels are close by spatially as well.
Bilateral filter replaces the pixel value at x with an average of similar and nearby pixel values. In smooth regions, pixel values in a small neighborhood 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.

2 Bilateral Filtering

2.1 What is Bilateral Filtering?

Bilateral filtering is a pixel-based approach. For a given pixel, its de noised counterpart is obtained by the weighted average of its neighbours, where the weights are given by some function that depends on their colour/intensity similarity and distance from the given pixel and the neighbour pixel considered. Intensity of the restored image is calculated as [2]:

$$g(i,j) = \frac{\sum_{k,l} f(k,l) * w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

where the weights by pixels (choosen for the particular neighborhood) on the pixel to be denoised is given by:

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

These weights are obtained from the range and domain filtering based on the σ_r and σ_d values - which are to be chosen by the user based on the blur/noise reduction required.

$$d(i,j,k,l) = exp(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2})$$

$$r(i,j,k,l) = exp(-\frac{||(f(i,j) - f(k,l))||^2}{2\sigma_r^2})$$

2.2 Extension to color images

Colour image smoothed by a standard lpf filter isnt just blurred, it also exhibits odd, coloured auras around objects as average between any two colours maybe a rather different colour. [3]Edge-preserving smoothing could be applied to the RGB bands of the image separately. However, the intensity profiles across the edge in the three colour bands are in general different and will still cause some distortion in the image. A bilateral filter allows combination of the three colour bands appropriately, and measuring photometric distances between pixels in the combined space. Moreover, this combined distance can be made to correspond closely to perceived dissimilarity by using Euclidean distance in the CIE-Lab color space. In this space, there are already many trained images that perceive to human eyes. Therefore, the system can use this information to correlate colors that can be meaningful to a human and calculate the range filter based on the trained data from CIE-Lab color space.

2.3 Advantages and Disadvantages

2.3.1 Advantages

1. Gaussian filtering in uniform areas of the image, no filtering across object borders.
2. [1]It depends only on two parameters that indicate the size and contrast of the features to preserve.
3. Simple, intuitive and non iterative

2.3.2 Disadvantages

1. Bilateral Filtering is quite slow compared to regular separable filtering. Complex, spatially varying kernels - difficult to compute
2. Staircase effect intensity plateaus that lead to images appearing like cartoons
3. May not always lead to the best result

3 Algorithm

1. For every pixel (i,j) in the image:
2. Decide the neighbourhood values (k and l) around the given pixel
3. For every pixel (k,l) in the neighbourhood of (i,j):
4. Compute the range kernel and spatial kernel using the formula
5. Form the weight coefficient using both the kernels obtained in the above step
6. Apply the weight coefficients to every (k,l) neighbourhood pixel
7. The weighted sum of all the neighbourhood value is returned as the new value of the (i,j)th pixel.

4 Code

4.1 Bilateral Filter

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 28 18:43:44 2018

@author: IIST
"""
import matplotlib.pyplot as plt
import numpy as np
import math
from PIL import Image
def distance(i, j):
    return np.absolute(i-j)

def bilateral_filter(i,j,d,I,sigma_d,sigma_r):
    arr=[]
    sum_num=0
    sum_den=0
    for k in range(i-math.floor(d/2),i+math.ceil(d/2)):
        for l in range(j-math.floor(d/2),j+math.ceil(d/2)):
            term1=(((i-k)**2)+(j-l)**2)/(sigma_d**2*2)
            term2=(distance(I[i,j],I[k,l]))/(sigma_r**2*2)
            term=term1+term2
            w=math.exp(-term)
            arr.append(w)
            sum_num=sum_num+(I[k,l]*w)
            sum_den=sum_den+w
    return sum_num/sum_den

```

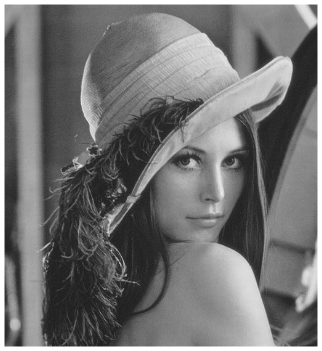
```
def main():
    img = Image.open('gaussian_noise_lenabw.png').convert('L')
    img.load()
    I = np.asarray( img, dtype="float" )
    data=I
    radius=7
    #print(I)
    I=np.lib.pad(I, 1, 'mean')
    I_new=np.copy(data)
    for i in range(1,data.shape[0]):
        for j in range(1,data.shape[1]):
            I_new[i-1,j-1]=bilateral_filter(i-1,j-1,radius,I,7,6.5)
            #print(I_new[i-1,j-1])
    plt.imsave('gn_lo2_bilateral_filtering_lena_new.png',I_new,cmap='gray')
    #isnr_calc(img,Image.fromarray(I_new))



main()
```

4.2 Calculating ISNR

```
# -*- coding: utf-8 -*-
"""
Created on Mon Dec  3 12:06:08 2018

@author: IIST
"""

# -*- coding: utf-8 -*-
"""
Created on Sat Dec  1 18:41:26 2018

@author: IIST
"""

import numpy as np
import math
import cv2
from PIL import Image
from numpy import linalg as la

original = Image.open("lenabw.jpg").convert('L')
noised = Image.open("gaussian_noise_lenabw.png").convert('L')
restored_bl = Image.open("gn_lo_bilateral_filtering_lena_new.png").convert('L'
    )
restored_edge=Image.open("after_edge_smoothening.png").convert('L')
restored_lpf=Image.open("gaussian_noise_filtered_lena_lpf2.png").convert('L')
#original = cv2.imread("gaussian_noise_lenabw.png")

def findSum(arr):

    # inner map function applies inbuilt function
    # sum on each row of matrix arr and returns
    # list of sum of elements of each row
    return sum(map(sum,arr))

def lsnr(img1,img2,img3):
    diff=np.subtract(img1,img2)
    temp=np.power(diff,2)
    final=findSum(temp)

    diff=np.subtract(img1,img3)
    temp=np.power(diff,2)
    final2=findSum(temp)

    return 10 * math.log10(final/final2)

original=np.asarray(original, dtype="float" )
noised=np.asarray(noised, dtype="float" )
restored_bl=np.asarray(restored_bl, dtype="float" )
restored_lpf=np.asarray(restored_lpf, dtype="float" )
restored_edge=np.asarray(restored_edge, dtype="float" )

lsnrrr=lsnr(original,noised,restored_bl)
print(lsnrrr)

lsnrrr=lsnr(original,noised,restored_lpf)
print(lsnrrr)

lsnrrr=lsnr(original,noised,restored_edge)
print(lsnrrr)
```

5 Results

5.1 Images after De-noising


(a) Original Image

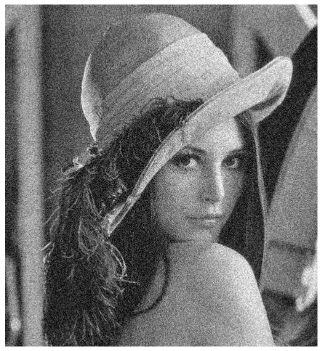

(b) Noised Image

Figure 1: Lena - Original and Noised Images

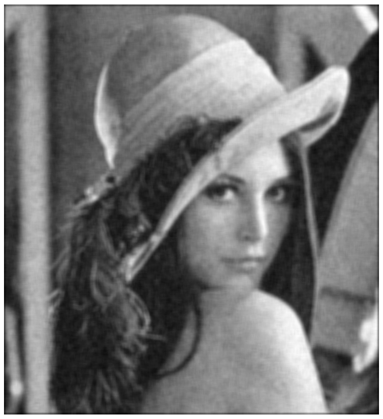


Figure 2: Low Pass Filtering

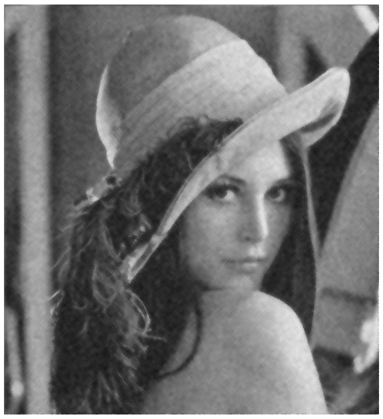


Figure 3: Bilateral Filtering

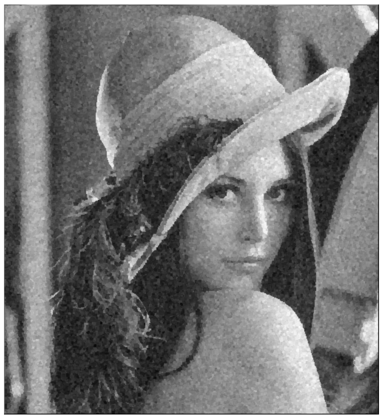
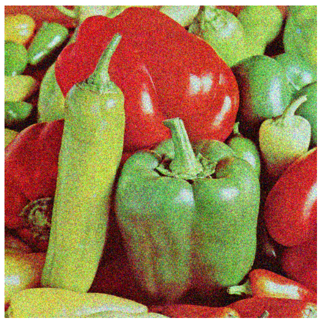


Figure 4: Edge Adaptive Filtering


(a) Pepper Noised

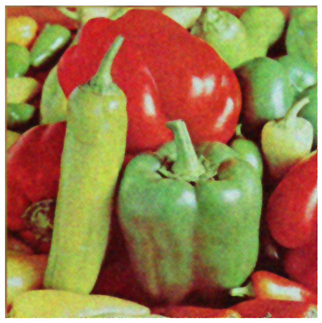

(b) Bilateral Filtered

Figure 5: Peppers - Color Image Filtering

5.2 Improvement in Signal to Noise Ratio

ISNR metric was used for assessing the quality of the results of restoration techniques used in the images by the various filtering methods.
Expression of ISNR is given by [**dip**]:

$$ISNR = 10log_{10}\left\{\frac{\sum[f[m,n]-y[m,n]]^2}{\sum[f[m,n]-\hat{f}[m,n]]^2}\right\}$$

Here $f[m,n]$ is the original image, $y[m,n]$ represents the degraded image. $\hat{f}[m,n]$ is the restored image.

Filter	ISNR Values
Bilateral Filtering	6.145030272320282
Low Pass Filtering	4.4108232206772335
Edge Adaptive Filtering	2.51157371372074

Table 1: ISNR values obtained after Filtering

6 Conclusion

The aim of this project was to implement the Bilateral Filtering for denoising images. Bilateral Filtering works on the principle that two pixels are said to be close when they share the same intensity or if they are occupy close enough spatial co-ordinates. Bilateral Filter assigns weight to a pixel based on their closeness and intensity values measures. It combines both domain and range filtering. Bilateral Filtering was implemented using the algorithm given in [2]. It was used to denoise black and white images like, cameraman, lena and also colored image, peppers. Its results were compared with that of other filters like Low Pass Filter and Edge Adaptive Filter. The results were checked against ISNR values. Bilateral Filter was found to perform better than Low Pass and Edge Adaptive Filters based on the ISNR values.

References

[1] Sylvain Paris et al. “A Gentle Introduction to Bilateral Filtering and Its Applications”. In: *ACM SIGGRAPH 2007 Courses*. SIGGRAPH ’07. San Diego, California: ACM, 2007. ISBN: 978-1-4503-1823-5. DOI: `10.1145/1281500.1281602`. URL: `http://doi.acm.org/10.1145/1281500.1281602`.

[2] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 1st. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN: 1848829345, 9781848829343.

[3] C. Tomasi and R. Manduchi. “Bilateral Filtering for Gray and Color Images”. In: *Proceedings of the Sixth International Conference on Computer Vision*. ICCV ’98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 839–. ISBN: 81-7319-221-9. URL: `http://dl.acm.org/citation.cfm?id=938978.939190`.