Project Report

Image Filtering Using Machine Learning

Submitted by

GOPAL SINGH

PG DIPLOMA IN DATA SCIENCE AND TECHNOLOGY,
CCSD,
JC BOSE UNIVERSITY OF SCIENCE AND TECHNOLOGY,
FARIDABAD.

Mentor

MR. RANJAN PRASAD

CCSD,
JC BOSE UNIVERSITY OF SCIENCE AND TECHNOLOGY,
FARIDABAD.

A ML Major project report is submitted in CCSD of JC BOSE University Of Science And Technology in partial fulfilment of the requirements for Post Graduation Diploma in Data Science And Analytics.

Date:

Place:

Approved by

Professor Name

Image Filtering Using ML

By Gopal Singh

Accepted in partial fulfillment of the requirements for the Degree of Post Graduation Diploma in Data Science and Analytics

_____	_____
Professor name	External Examiner
Date:	Date:

Contents

1. Abstract

Image filtering is changing the appearance of an image by altering the colors of the pixels. Increasing the contrast as well as adding a variety of special effects to images are some of the results of applying filters. In order to obtain a high success rate of OCR (optical character recognition) performed on text images, the main target of filters is, however, to reduce the noise around characters in the image. Thereby, I created two new nonlinear efficient image filters (for RGB and ARGB images) and elaborated the optimization technique suitable for each of the filters in order to make them perform faster. The first filter, namely, Smart Contrast, increases the contrast of the image in a way depending on the value of each component (Red, Green, and Blue) of each pixel in the image. The second image filter, called Highlight, produces a remarkable increase in OCR success rate on the filtered text images. As Highlight filter is carried out, the implementation differs from all other known filters, while the visual effect on the images can be described as a combination between increased contrast as said before with other two visual effect: sharpening and highlighting the edges (of the characters). Precisely, combining these specific visual effects on the resulting image makes Highlight filter so powerful in improving OCR on text images.

Recent advances in 3D technology have made the need for reliable quality evaluation of synthesized virtual views. Using RGB image and the corresponding depth map, a synthesized view can be constructed. Depth map upsampling has gained much interest following the release of time-of-flight cameras. As upsampling can yield artifacts on sharp edges like ringing artifacts and jagged edges near the depth map boundaries, it will degrade the final reconstructed stereoscopic image quality. In this article, several upsampling methods are applied to depth maps. Then, diverse full-reference and no-reference quality assessment tools are utilized to measure the quality of depth maps obtained from selected upsampling approaches. Furthermore, subjective test is performed to determine the quality of stereoscopic images reconstructed from upsampled depth maps. Finally, the relation between subjective assessment and each

objective quality assessment is investigated using correlation coefficients. The evaluation results introduce the objective quality assessment tools that can correctly render human judgement.

This work describes perceptual generalized bitplane-by-bitplane shift (ρGBbBShift) a perceptual method for coding of region of interest (ROI) and it is based on Moreno et al. (2013) [1]. Then, this article introduces perceptual criteria to the GBbBShift method when bit planes of ROI and background areas are shifted. This additional feature is intended for balancing perceptual importance of some coefficients regardless of their numerical importance. Perceptual criteria are applied using a contrast band-pass filtering, which is a low-level computational model that reproduces color perception in the human visual system. Results show that there is no perceptual difference at ROI between the MaxShift method and ρGBbBShift and, at the same time, perceptual quality of the entire image is improved when using ρGBbBShift. Furthermore, when ρGBbBShift method is applied to H*i*-SET coder and it is compared against MaxShift method applied to both the JPEG2000 standard and the H*i*-SET, the images coded by the combination ρGBbBShift-H*i*-SET get the best results when the overall perceptual image quality is estimated. The ρpGBbBShift method is a generalized algorithm that can be applied to other Wavelet-based image compression algorithms such as JPEG2000, SPIHT, or SPECK.

2. Acknowledgements

This work would not have been possible without the support of many people. I wish to express my sincere gratitude to all those who gave me the possibility to complete this thesis.

First of all, I would like to thank my supervisor, Mr Ranjan Prasad YMCA UST,Faridabad. for helping me realize this project and for his guidance during this work and giving me the opportunity to work in one of well-equipped lab of the institution. His valuable ideas and suggestions were immensely helpful which make this thesis work possible.

I would also like to thank Mr. Rajinder Chitoria (Data Scientist) and Co-Founder of Froyo ,Faridabad for granting me opportunity for Training at Froyo technology by Antrix Academy at Faridabad.

And Most importantly, I want to thank my family for providing unlimited support and helping me realize my dreams.

3. INTRODUCTION

Most of the images are influenced by some kind of unwanted noises causing disturbance in image quality and resolution. , Analyzing the images are usually processed right after removing these noises from the images. Filtering has various meanings and applications in most of the engineering and scientific problems. It provides researchers with fine results comparing with other available methods. Researchers in image/signal processing, electrical and electronic, mechanical engineering, chemistry and physics are utilizing filtering in their problems. One of the most common applications of filtering is in image processing to give a better resolution or to emphasize the edges properly.

Filtering :

Box filter, Gaussian filter and bilateral filters are kind of well-known filters used in image processing. As we know all these filters are used for de-blurring and smoothing. In addition, with the help of these filters some facts which are clear in the original image will be blurred and the final image will be enhanced. With considering the best sigma value for bilateral filter, after comparing all these four filters we will find out that the bilateral filter is the best. As conserving the edge is needed, we want our image to be blurred and smoothed, so bilateral filtering is very useful in such a case [1][2]. This filter has various applications in engineering and science [3-9]. In this technique, colors and gray levels are joined and close values are more preferred than the far or distant ones. The main aim of the cited method is removing phantom colors in the original image which may be appeared in the edges.

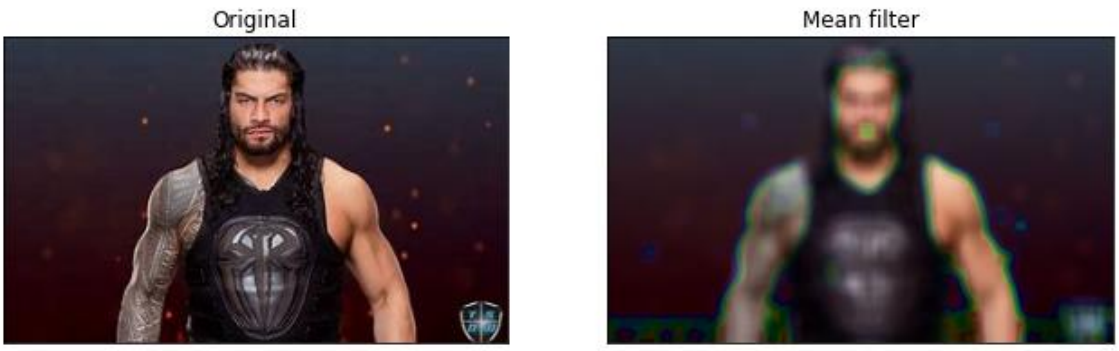
Bilateral filtering is kind of de-noising methods which can preserve sharp edges and remove some colors across the edges. This filter because of weighted mean of neighborhood pixels are easy to understand. In addition, there is a distance between pixel values, so it can be adopted properly. Last but not least, this filter is non-iterative which is very simple

4. Methods For Image Filtering

a. Mean Filter

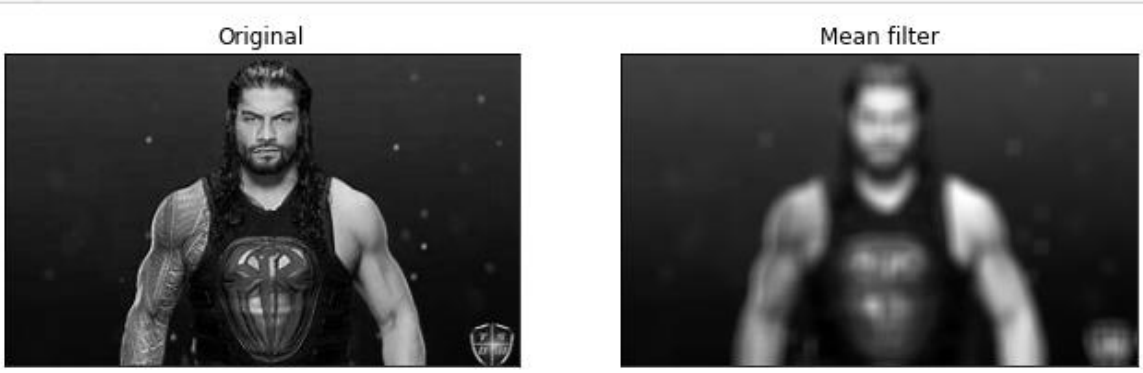
The mean filter is used to give a blur effect to an image to remove the existing noisiness. It determines the mean of the pixels within the n×n method. Then it replaces the intensity of pixels by the mean. This reduces some of the noisiness present in the image and also improves the edges of an image.

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from PIL import Image, ImageFilter
%matplotlib inline
image = cv2.imread('OIP.jpg') # reads the image
image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV) # convert to HSV
figure_size = 9 # the dimension of the x and y axis of the kernal.
new_image = cv2.blur(image,(figure_size, figure_size))
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_HSV2RGB)),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_HSV2RGB)),plt.title('Mean filter')
plt.xticks([]), plt.yticks([])
plt.show()
```



The above figure shows that some of the noisiness has been reduced, but there are some marks now which were not present in the image before. Let's see if those marks still appear if we filter the image into grayscale:

```
# The image will first be converted to grayscale
image2 = cv2.cvtColor(image, cv2.COLOR_HSV2BGR)
image2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
figure_size = 9
new_image = cv2.blur(image2,(figure_size, figure_size))
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(new_image, cmap='gray'),plt.title('Mean filter')
plt.xticks([]), plt.yticks([])
plt.show()
```

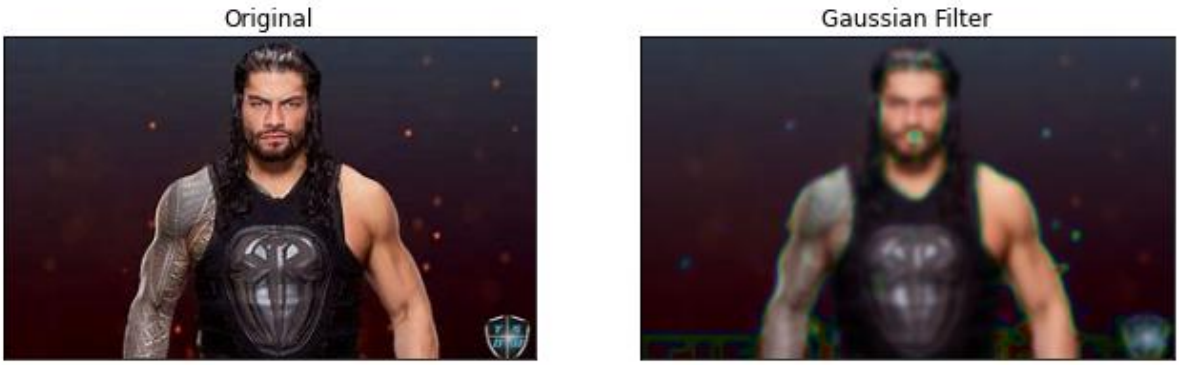


b. Gaussian Filter

The Gaussian filter is very similar to the mean filter, but it involves a weighted mean of the pixels with a parameter as sigma. Let’s go through this method of Image Filtering:

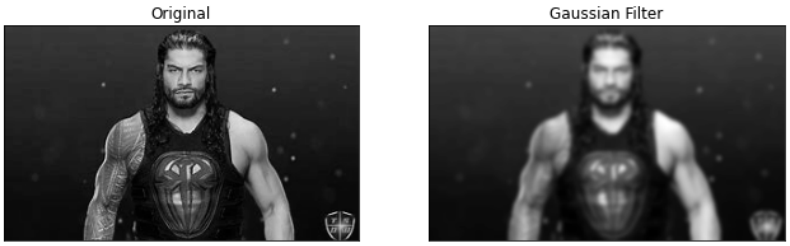
```
new_image = cv2.GaussianBlur(image, (figure_size, figure_size),0)

plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_HSV2RGB)),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_HSV2RGB)),plt.title('Gaussian Filter')
plt.xticks([]), plt.yticks([])
plt.show()
```



The above figure is representing a good job comparing to the mean filter, although it also leaves some marks we saw in the mean filter. Now let's have a look whether the Gaussian Filter removes the marks when converted into a grayscale:

```
new_image_gauss = cv2.GaussianBlur(image2, (figure_size, figure_size),0)
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(new_image_gauss, cmap='gray'),plt.title('Gaussian Filter')
plt.xticks([]), plt.yticks([])
plt.show()
```

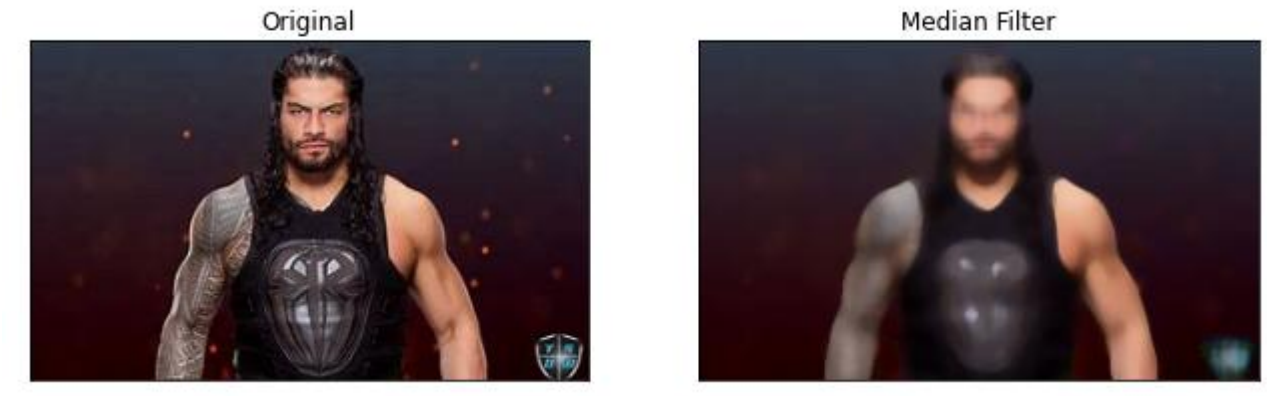


As you can see, the marks are removed in the grayscale effect. Now let's move to the next method of Image Filtering.

c. Median Filter

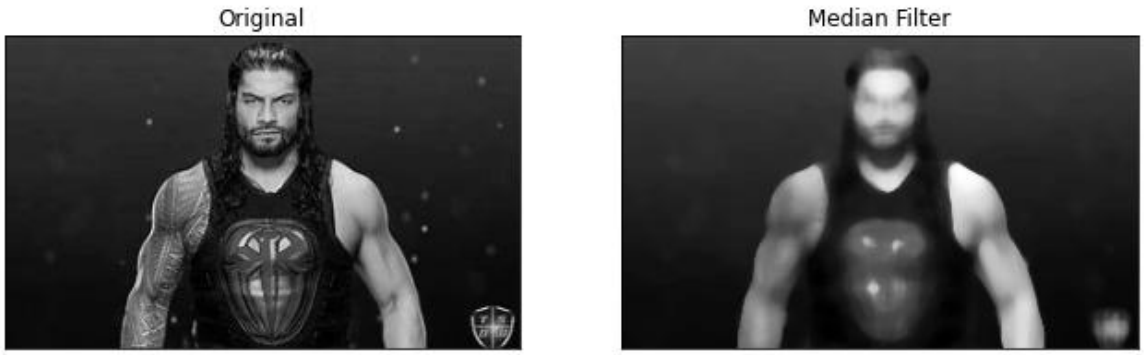
The median filter computes the median of the intensity of pixels. It then replaces the norm with the pixel intensity of mean pixels. Now let's go through this technique:

```
new_image = cv2.medianBlur(image, figure_size)
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_HSV2RGB)),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_HSV2RGB)),plt.title('Median Filter')
plt.xticks([]), plt.yticks([])
plt.show()
```



The median effect leaves a better output by removing the noisiness of the image. It also did not leave any marks in the image that we saw in the above methods. Now let's convert it into a grayscale image:

```
new_image = cv2.medianBlur(image2, figure_size)
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(new_image, cmap='gray'),plt.title('Median Filter')
plt.xticks([]), plt.yticks([])
plt.show()
```



5. Creating Our Own Filter

Now I will create our own filter for Filtering Images than can improve the quality of the image and improves the smoothness of the picture. I will first create a function to design my own Image filter:

```
def aman_effect(data, filter_size):
  temp = []
  indexer = filter_size // 2
  new_image = data.copy()
  nrow, ncol = data.shape
  for i in range(nrow):
    for j in range(ncol):
      for k in range(i-indexer, i+indexer+1):
        for m in range(j-indexer, j+indexer+1):
          if (k > -1) and (k < nrow):
            if (m > -1) and (m < ncol):
              temp.append(data[k,m])
      temp.remove(data[i,j])
      max_value = max(temp)
      min_value = min(temp)
      if data[i,j] > max_value:
        new_image[i,j] = max_value
      elif data[i,j] < min_value:
        new_image[i,j] = min_value
        temp =[]
    return new_image.copy()
```

```
# Now let's use our filter to improve the quality of the grayscale image:

new_image = aman_effect(image2,5)
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(new_image, cmap='gray'),plt.title('Image Filteration')
plt.xticks([]), plt.yticks([])
plt.show()
```

6. Conclusion:

In this Project various filtering approaches are utilized to address the issue of image recovery from its noisy counterpart. The image filtering algorithm provides us with smoothness and better resolution from the noisy version. Experiments are conducted that bilatereal filtering performs better than other filtering techniques which are available in the literature. For the future work, we will deall with applying some optimization algorithms and machine learning approaches [10- 25] to check and compare the performance analysis of different methods.