Delving in Principles and Implementation of Gaussian Smoothing

Araz Sharma
Department of Computer Science and Engineering
PES University
Bangalore, India
arazsharma1103@gmail.com

Shreya Shukla
Department of Computer Science and Engineering
PES University
Bangalore, India
12345shreya12345@gmail.com

Rudra Prasad Baksi
Department of Computer Science and Engineering
PES University
Bangalore, India
rudranger@gmail.com

***Abstract*—Gaussian Smoothing is an important technique in image processing, often used in the pre-processing stage for computer vision algorithms. It is an excellent application of Linear Algebra concepts and demonstrates their power in real life models. It is often used to reduce image noise and detail. The Gaussian filter is inspired from human visual perception system as the neurons create a similar filter when processing visual images. For the smoothing application, we use this as a 2D Convolution operator, where this distribution function is used as a 'point – spread' function. Gaussian Smoothing is often a pre-step for segmentation edge detection algorithms like Sobel Filter and Canny Edge Detector, to remove the noise, to make them more efficient. They are also used for Small Animal PET Systems, Pose Measurement based on Vision Perception and Computer-Aided Diagnoses System for Detecting Multiple Ocular Diseases using Colour Retinal Fundus Images. This paper will focus on exploring the Gaussian Smoothing technique, understanding the concepts of mathematics behind it, and implementing it with real life images using OpenCV Library.**

***Index Terms*—Linear Algebra, Gaussian Smoothing, Convolution, OpenCV, Image Processing**

I. Introduction

Humans have always geared towards evolution, whether it is in the field of art, science or commerce. As curious beings, we have always questioned the unreasonable, always pursued the unthinkable and always changed the definition of impossible. Keeping this nature of ours in mind, we endeavoured to do the same in this field of Image Processing and specifically Gaussian Smoothing.

The very word Gaussian brings the good old distribution to mind, and provokes the question of how did Carl Friedrich Gauss get his hands dirty in this field of Image Processing, a technique germane to the 21st Century. It can be blamed on his knack for rising beyond his grave, or being claimed clairvoyant, for his work still pertains to the simple actions we do in our daily lives. The idea behind usage of a simple 2-D distribution, in the field pertinent to this paper, lies in using it as a 'point-spread' function, which is achieved with convolution, a very popular term in this document.

To dive in deep in this wonderful field of using mathematical functions to help take better selfies from your smartphone, perhaps the most popular daily usage of Image Processing, it calls upon to throw light on the term 'Convolution'. [1]

Convolution is nothing but a simple mathematical operation, which aids in 'multiplying together' two arrays of numbers, different in size, however belonging to the same dimensionality, to conjure a third array of numbers of the given dimensionality. The usage of convolution is most popular to implement operators whose output pixel values are simple 'linear combinations' of given input pixel values.

Since the term linear combination isn't self explanatory, we shall also shine some light there, to elucidate its meaning and relevance to the matter at hand.

Linear Combination, in the field of Linear Algebra, is the combined sum of 2 types of quantities, for example vectors scalars, where if V1, V2...Vn are vectors and S1, S2...Sn are scalars, is denoted by: S1*V1 + S2*V2 + ... Sn*Vn

With regards to the role of Convolution and Linear Combination in Gaussian Smoothing, it can be briefly introduced by the figure given below:

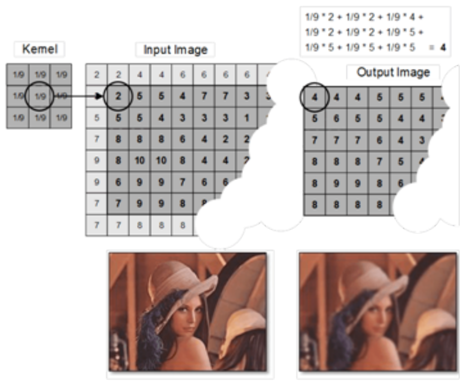


Fig. 1. Example of Convolution.

Now that we have covered some of the basic pre-requisites, it is apt to take a deeper plunge in Gaussian Smoothing, which contrary to its name, looks as a blurred effect.

The detailed explanation for the process of how it blurs is covered in the methodology. Before jumping to that, it is important to know some of its key features and compare it with other techniques used in Image Processing.

Gaussian Smoothing is an essential feature in graphics software [12], typically to reduce image noise and lower detail.

The visual effect from this is a smooth blur, thus giving the term ‘smoothing’ in the name. This distinctly differs from the Bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination. Comparing the two effects mathematically, we find that applying a Gaussian blur to an image is the same as convolving the image with a Gaussian function, also known as a two-dimensional Weierstrass transform. Whereas, convolving by a circle or a circular box blur results in the Bokeh effect.

Another interesting concept linked to this, is the involvement of the famous Fourier Transforms. The Fourier Transform of a Gaussian function results in the same function [20], and thus we see that the Gaussian filter does not have a sharp cut-off at some pass band frequency beyond which all higher frequencies are removed. To the contrary, it encompasses a graceful and natural tail that becomes ever lower as the frequency increases. This establishes that it will act as a low pass filter, but also allow in higher frequency components in proportion with how quickly its tail decays. Thus applying a Gaussian blur reduces the image’s high-frequency components, making it a low pass filter.

Thus Gaussian filters are used in image processing because they have a property that their support in the time domain is equal to their support in the frequency domain. On the other hand, [9] other low pass filters will have a higher time bandwidth product, because its support in the F-domain is not nearly as big as that of Gaussian. This gives the Gaussian filter a fair advantage over other low pass filters in blurring images in image processing.

To conclude the introduction for this paper, one can review some characteristic properties of Gaussian Filtering or Smoothing:

•Gaussian smoothing is very effective for removing Gaussian noise

•The weights give higher significance to pixels near the edge (reduces edge blurring)

•They are linear low pass filters

•It is computationally efficient (large filters are implemented using small 1D filters)

•It is rotationally symmetric (perform the same in all directions)

•The degree of smoothing is controlled by (larger for more intensive smoothing)

II. Literature Survey

A. *Refer to [1]*

This research paper begins with the need and different arenas of image filtering. The image signal data is converted into a digital form by sampling and quantization, resulting in a two-dimensional array of matrix of numbers representing the pixel values. The process of image filtering is used for modifying or enhancing an image, by smoothing, sharpening, removing noise and edge detection. Filtering can be done in the spatial domain, or frequency domain, but applying filters in the frequency domain is computationally faster than that in the image domain. Some basic terms of image filtering were then discussed and explained ,namely Noise, Gaussian noise, Salt and Pepper Noise, Blurring and Sharpening.

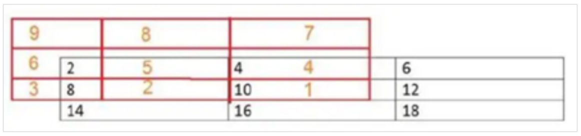
The theory and implementation of Discrete Fourier transforms (DFT), to convert the image from the spatial domain to the frequency domain, and the Inverse transform to convert from frequency domain to spatial domain is then discussed. If the input signal is an image, then the number of frequencies in the frequency domain is equal to the number of pixels in the image or spatial domain. Since, using DFT and its inverse is an expensive operation; we use Fast Fourier Transform (FFT) and its inverse. The 2D transform can be done as two 1D transform, one in horizontal direction followed by other in vertical direction on the result of horizontal transform. The resulting value is equivalent to performing the 2D transform in the frequency space. FFT and its inverse are obtained as:

$$F(x,y) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(m,n)e^{-j2\pi\left(\frac{sm}{M}+\frac{yn}{N}\right)}$$

$$f(m,n) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(x,y)e^{j2\pi\left(\frac{sm}{M}+\frac{yn}{N}\right)}$$

Further on in this study, [12] we learn about the process of convolution, more in depth for the spatial domain, and get an idea for the frequency domain.

In the spatial domain:



The coloured box is the mask obtained after flipping the original mask horizontally once and vertically once, while the second box represents the original image pixels

In the frequency domain:

$$y(k) = \sum_{n=0}^{N} x(k-n)h(n)$$
$$\text{where n=0,1,2...N}$$

We then learn in detail about the Gaussian filters, which is a smoothing filter, highly effective for removing noise drawn from a normal distribution. The Gaussian filter in 1 – D and 2- D is given as in ”Fig. 2” and ”Fig. 3”.

Gaussian kernel coefficients are sampled from the 2D Gaussian function. The Gaussian filter is a non-uniform low pass filter. The kernel coefficients diminish with increasing distance from the kernel’s centre. Central pixels have a higher weighting than those on the periphery

$$G(x) = \frac{1}{\sqrt{(2\pi\sigma^2)}}\, e^{-\frac{x^2}{2\sigma^2}}$$

Fig. 2. Gaussian Filter in 1D.

$$G(x,y) = \frac{1}{2\pi\sigma^2}\, e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Fig. 3. Gaussian Filter in 2D.

B. *Refer to [2]*

This research work focused on inspecting the qualitative and quantitative effects of the convolution of a Gaussian function with an image. Besides focusing on Gaussian blur, the methodology of segmentation using the same was also highlighted. As an establishment for the research work, the inevitability of the presence of noise in images was spoken about. Noise, is inherent to the process of acquiring images and therefore images needs to be filtered before using it, using a process which reduces as much noise as possible while losing a minimum amount of information. This work investigates the effects of a Gaussian blur on an image and evaluates the results of the filtering using a qualitative index, then it tackles an application to segmentation, using the Gaussian blur to generate a mask.

The topic of interest for this research paper, the Gaussian Blur involves the convolution of a kernel, described by a Gaussian function with the pixels of the image. The convolution in the discrete case is given by:

$$f(x,y) = A.e^{-\left(\frac{(x-x_0)^2}{2\sigma x^2}+\frac{(x-y_0)^2}{2\sigma y^2}\right)}$$

The function used to generate the kernel is the two dimensional Gaussian function. In the following function, A is the amplitude, (xo, yo) the centre, x , y the standard deviations in the x and y directions:

$$f * g[n] \overset{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m].g[n-m]$$

In image processing, the Gaussian distribution needs to be approximated by a convolution kernel. Therefore, values from this distribution are used to build a convolution matrix then applied to the original image. Each pixel's new value is a weighted average of that pixel's neighbourhood. Thus, the original pixel's value receives the heaviest weight (having the highest Gaussian value) and neighbouring pixels receive smaller weights as their distance to the original pixel increases. The indicator used to evaluate the noise level in the images is the Signal-to-Noise Ratio (SNR). The SNR basically compares the intensity of the signal with the intensity of the noise. A higher SNR means better image quality. It was finally concluded that the Gaussian blur technique is particularly useful to filter images with a lot of noise, since the results of the filtering showed a relative independence on the noise characteristics, and strong dependence on the variance value of the Gaussian kernel.

C. *Refer to [3]*

This research work talks about 2D Gaussian filter and its vitality in the image processing domain. It begins with an introduction of the need for different architectures to be able to implement high computation processing and multiprocessing at the real time. It then moves on to a comparison between different general purpose parallel computing architectures, namely, CPU, GPU and FPGA. Further on, our topic of interest for this research paper, the process of image filtering was spoken about. We get to know about how the image is sent to a particular architecture and how the values are coded. The method of convolution is then described wherein carrying out this process for an image with a [3 x 3] kernel requires extracting each [3 x 3] sub-image from the original image, from left to right, and from top to bottom. After that, each of those elements is multiplied with the kernel image and the resulting products are finally added together (through a multiply-accumulate -MAC- operation), returning a single value for the output pixel. Figure2 illustrates the MAC process. This study about 2D-Gaussian filters, used to filter the images to eliminate the noise from the images is then concluded with the establishment that the FPGA implementation has proved to be the best amongst CPU or GPU.

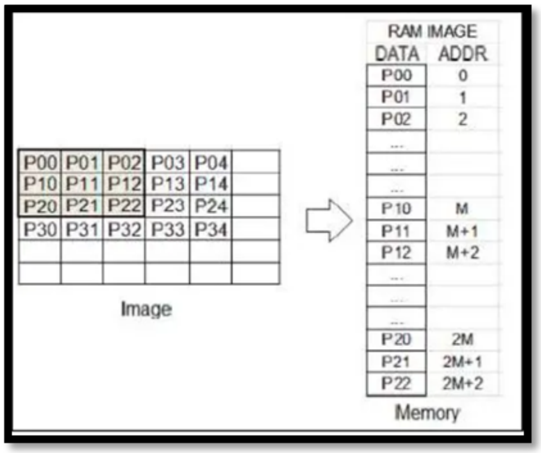


Fig. 4. Conversion of an Image Matrix to Image Vector.

D. *Refer to [4]*

This research paper talks about how the Gaussian filter can be used for signal smoothing and noise filter/denoising and how we can come up with an optimal Gaussian filter, with specific properties which proves to be the most effective noise filter. This has been specifically shown for the additive white Gaussian noise (AWGN). As mentioned in this literature, there

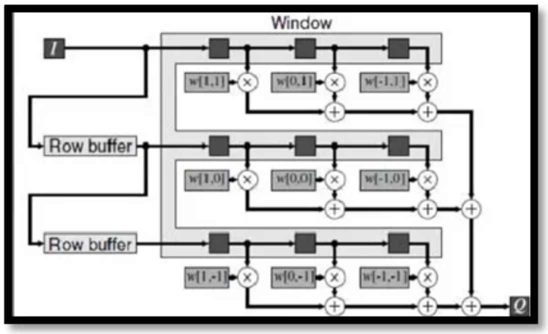


Fig. 5. Extraction of a [3x3] sub-image and MAC operations.

are essentially two ways of taking care of noise in a signal, namely pre-processing of the signal to enable noise removal or use of a set of robust algorithms that can compensate for the inherent noise. In this study, pre-processing of the signal is the adopted approach. Various tests and experiments were performed to identify the ideal characteristics of the Gaussian filter to be used. The major contribution of this paper is identification of a method to obtain the optimal Gaussian filter that best filters a signal contaminated with AWGN. It was shown experimentally that the identified method works well for signals whose bandwidth and the input signal to noise ratio is known.

E. Refer to [5]

In this research work, we see an elaboration on the ideas of digital image processing when applied to images, and get an idea about the concepts of convolution as a means to apply DSP techniques and simplify calculations. We focus on the MATLAB implementation of some image processing techniques, including the highlight of our study in this research paper, the Gaussian smoothing function. We also get to know that pre-processing images before applying other filters is shown to produce improved results when extracting edges from images with noise. We start with getting a background of analog and digital signals, and the difference in how image processing works for both of them. We then get a brief about digital images and kernel matrices. We discuss about convolution, which according to the text can be intuitively described as function that is the integral or summation of two component functions, and that measures the amount of overlap as one function is shifted over the other. If f and g are functions in t, then the convolution of f and g over an infinite interval, and over a finite interval [0,t] are integrals given by:

$$f * g \equiv \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau$$

$$[f * g](t) \equiv \int_{0}^{t} f(\tau) g(t-\tau) d\tau$$

Moving on, we learn about image filters for image enhancement and edge extraction. More importantly, for the topic of our concern in this research work, we learn about Noise Reduction and Gaussian and low-pass filters, and their implementations using MATLAB, which motivated us to do the same with OpenCV. The Gaussian filter is known as a 'smoothing' operator, as its convolution with an image averages the pixels in the image, effectively decreasing the difference in value between neighbouring pixels. The effect of Gaussian filters on images with several types of noise incorporated into the test image, and with different values of was illustrated, and the results noted. The study was concluded with a discussion on the future scopes of these techniques.

This literature survey gave us detailed knowledge about various aspects of the image processing domain in general, and the technique of Gaussian smoothing, in particular. In the first paper we surveyed, we saw the differences in filtering images in the frequency and spatial domain, and deciding their suitability according to the scenario. From the study of the second paper, we inferred that the results of Gaussian filtering showed a relative independence on the noise characteristics and on the other hand a dependence on the variance values of the Gaussian kernel we are using. An exploration of the fourth report saw us looking into the image filtering process and the conversion of an image matrix to an image vector. The analysis of the fourth paper gave us an insight on how it is possible to identify an optimal Gaussian filter that best filters noise, under the constraint of certain assumptions. Our fifth survey paper, presented a MATLAB implementation of Gaussian smoothing, and acted as the base and motivation for our OpenCV based implementation of the same.

III. Methodology

After going through a brief analysis of our literature survey the thing that becomes most important is the methodology of the process of smoothing. Basically it throws light on the mathematical implementation of smoothing using the Gaussian function in image processing. Not one, but many concepts synchronize and work hand in hand, to make the process of smoothing of an image work.

Now it comes to how we can use these concepts for Gaussian filters in blurring images. The first thing that comes here is the Gaussian 1-D distribution as it is the backbone of the convolution process being done in 2-D Gaussian distribution [19] The Gaussian distribution in 1-D has the form as in "Fig. 2" In 2-D, an isotropic (i.e. circularly symmetric) Gaussian has the form as in "Fig. 3".
Here sigma is the standard deviation of the distribution.

The idea of Gaussian smoothing revolves around the use of this 2-D distribution as a point-spread function, and this objective can be achieved by convolution. The concept of convolution which has been the very backbone of Gaussian smoothing has been very fundamental to many image processing operations as it provides a way of multiplying together two arrays of numbers generally of different sizes but having the dimension thus producing a third array of numbers having same dimensionality. [13]

Since the image is stored as a collection of discrete pixels, it becomes the need of the hour for us to produce a discrete approximation to the Gaussian function before we can perform the convolution. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but practically it is effectively zero more than about three standard deviations from the mean, and so we are able to truncate the kernel at this point. [14] [15]

After drawing the 2D Gaussian distribution with mean (0, 0) and sigma=1 a suitable integer-valued convolution kernel that approximates a Gaussian with a sigma of 1.0 is obtained. There is no compulsion of selecting some particular values of the mean and the standard deviation, sigma. [14] [15] Moreover different values of standard deviation, sigma of the Gaussian will give different types of blurred images on the basis of degree of smoothing of the image [18]

Gaussian theoretically has infinite support, but practically, we need a filter of finite size. From the figure of 2D Gaussian distribution it has been found out that +/-2.5 sigma covers 98.76% of the area and +/-3 sigma covers 99% percent of the area of the Gaussian. A clear-cut method of picking the values of the mask to approximate a Gaussian isn't available. One could use the value of the Gaussian at the centre of a pixel in the mask, but this is never accurate because the value of the Gaussian varies non-linearly across the pixel. We have to integrate the value of the Gaussian over the whole pixel (by summing the Gaussian at 0.001 increments). The integrals are not integers: we have to re-scale the array so that the corners will have the value 1. [13] [16] [17] [18]

After this, the sum of all values in the mask has to be calculated. Once a suitable kernel has been calculated, then the Gaussian smoothing can be performed using standard convolution methods. [13] [16] [17] [18] The convolution can be performed fairly quickly since the equation for the 2-D isotropic Gaussian is easily separable into x and y components. Thus the 2-D convolution can be performed by first convolving with a 1-D Gaussian in the x direction, and then convolving with another 1-D Gaussian in the y direction. [13] [16] [17] [18] While convolving in 1-D Gaussian in x direction, we first have to convolve each row with a 1-D filter, and then we have to convolve each column with a 1-D filter. For convolving with 1-D Gaussian in y direction the calculation also goes by the same way as it is done for in x direction. [13] [16] [17] [18] Another way to compute a Gaussian smoothing with a large standard deviation is to convolve an image several times with a smaller Gaussian to stimulate the effects of a larger one. This is known as Cascaded Gaussians [18] This is possible as the Gaussian follows the following associativity property:

$$G*(G*f) = (G*G)*f$$

[13] [16] [17] [18] While this is computationally complex and time consuming, it can be used if the processing is carried out using the concept of hardware pipeline.

IV. Implementation

A. Resizing the Image to Display Optimally:

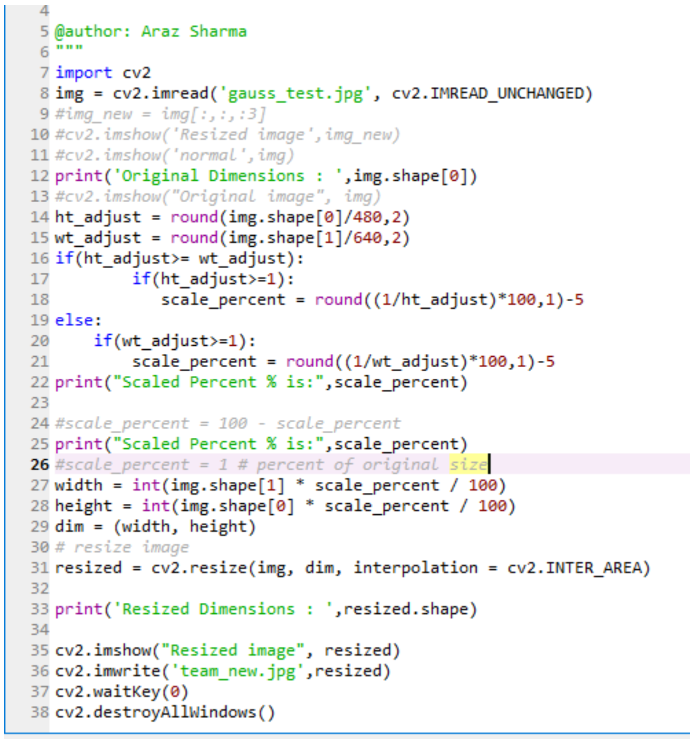
a. The first step to Implement Gaussian Smoothing with OpenCV is to find an image to implement it on.
b. The brilliant part about this concept, and it's execution with OpenCV is that you aren't restricted to images of a particular type, format, style or size. OpenCV offers many functions which can tailor images to view them in an optimal manner.
c. OpenCV displays your input image with a pop up display window. This display window for any device, like a webcam or an external output has the dimensions of 480 X 640, in height width respectively, in pixels. This is a standard display size, optimal to Computer Vision and Image Processing concepts, which must be adhered to, for applying Gaussian Smoothing at Industry Level Projects.
d. Any image exceeding these dimensions isn't optimal for us. So to ensure that the images we choose can fit well under these dimensions, we have created a program to resize them, keeping in mind their current dimensions.
e. However, for the sake of argument, we researched and experimented to conclude that Gaussian Smoothing is not restricted to the size of the display window.

B. Python Implementation for the above problem statement:

a. This code is taking in the dimensions of our input images, and comparing them with the required dimensions for the standard display output.
b. The height of input image is factored by 480, and the width is factored by 640.
c. The image is then resized, by keeping a margin of 5% with required dimensions, while maintaining its aspect ratio, so as to keep it same as the original requirements of the client.
d. The new resized image is then saved on to the node terminal, and is further used to implement Gaussian Smoothing on that.

C. Implementing the Gaussian Smoothing on the Resized Image:

a. OpenCV has a pre-compiled function known as 'GuassianBlur'. This is a very powerful tool, which will be used by us to implement the Gaussian Smoothing technique talked about in this paper.
b. This function is represented as follows:
• GaussianBlur (src, dst, ksize, sigmaX)
• Src: This is a Matrix Object represented by the Input Image
• Dst: This is a Matrix Object representing the destination Image



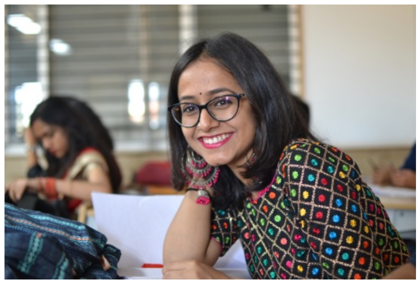
```
@author: Araz Sharma
"""
import cv2
img = cv2.imread('gauss_test.jpg', cv2.IMREAD_UNCHANGED)
#img_new = img[:,:,:3]
#cv2.imshow('Resized image',img_new)
#cv2.imshow('normal',img)
print('Original Dimensions : ',img.shape[0])
#cv2.imshow("Original image", img)
ht_adjust = round(img.shape[0]/480,2)
wt_adjust = round(img.shape[1]/640,2)
if(ht_adjust>= wt_adjust):
        if(ht_adjust>=1):
            scale_percent = round((1/ht_adjust)*100,1)-5
else:
    if(wt_adjust>=1):
        scale_percent = round((1/wt_adjust)*100,1)-5
print("Scaled Percent % is:",scale_percent)

#scale_percent = 100 - scale_percent
print("Scaled Percent % is:",scale_percent)
#scale_percent = 1 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
# resize image
resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

print('Resized Dimensions : ',resized.shape)

cv2.imshow("Resized image", resized)
cv2.imwrite('team_new.jpg',resized)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fig. 6. Code for Resizing Image with OpenCV

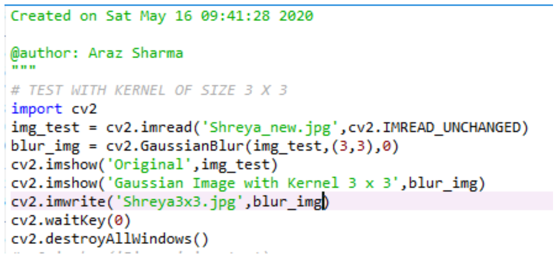
• Ksize: The Gaussian Kernel Size. The height width can differ, but they both must be positive odd, to ensure a centre pixel can be chosen
• SigmaX: Gaussian Kernel Standard Deviation in X – Direction
c. In order to investigate the effect of sigma on Gaussian smoothing we can look into the fact how it actually affects the 2D Gaussian distribution. If we take partial derivative of Gaussian with respect to sigma(standard deviation) we will come to see as the sigma value increases the area under the curve increases and the bandwidth of the curve increases and this results in flattening of the curve and as a result the height of the curve decreases. So as the height of the curve decreases many high frequencies are missed out resulting in loss of details of the image thus giving rise to a more blurry image. Thus varying the value of sigma will affect the degree of smoothing of the curve.
d. The Kernel Size is the parameter which determines the size of the convolution matrix, and is responsible for implementing the Gaussian Smoothing Process, by applying the average of all pixels in the convolution matrix to the centre pixel, on the input of Image. Therefore a larger kernel size will result in harshening of the blur effect. We have written a program in python, using the OpenCV Library to show this effect in real time for a particular input Image.
e. We are taking 3 Kernel Sizes:
• 3x3
• 7x7
• 11x11

D. Python Implementation using OpenCV to demonstrate the Gaussian Smoothing Effect:

a. Input Image:



b. Code and Output for Gaussian Smoothing with a 3x3 Kernel:



```
Created on Sat May 16 09:41:28 2020

@author: Araz Sharma
"""
# TEST WITH KERNEL OF SIZE 3 X 3
import cv2
img_test = cv2.imread('Shreya_new.jpg',cv2.IMREAD_UNCHANGED)
blur_img = cv2.GaussianBlur(img_test,(3,3),0)
cv2.imshow('Original',img_test)
cv2.imshow('Gaussian Image with Kernel 3 x 3',blur_img)
cv2.imwrite('Shreya3x3.jpg',blur_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fig. 7. Code with 3x3 Kernel

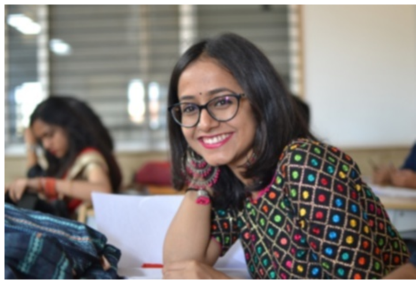
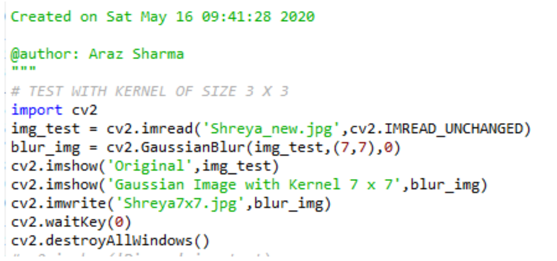


Fig. 8. Gaussian Smoothing with 3x3 Kernel

c. Code and Output for Gaussian Smoothing with a 7x7 Kernel:



```
Created on Sat May 16 09:41:28 2020

@author: Araz Sharma
"""
# TEST WITH KERNEL OF SIZE 3 X 3
import cv2
img_test = cv2.imread('Shreya_new.jpg',cv2.IMREAD_UNCHANGED)
blur_img = cv2.GaussianBlur(img_test,(7,7),0)
cv2.imshow('Original',img_test)
cv2.imshow('Gaussian Image with Kernel 7 x 7',blur_img)
cv2.imwrite('Shreya7x7.jpg',blur_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fig. 9. Code with 7x7 Kernel

d. Code and Output for Gaussian Smoothing with a 11x11 Kernel:

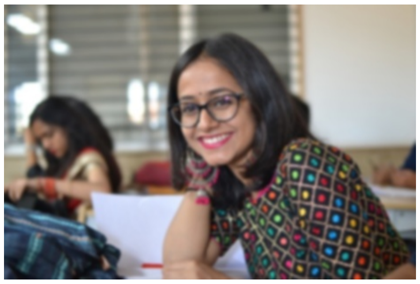


Fig. 10. Gaussian Smoothing with 7x7 Kernel

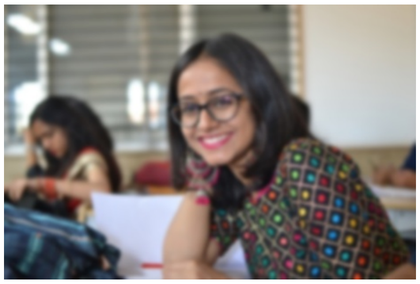
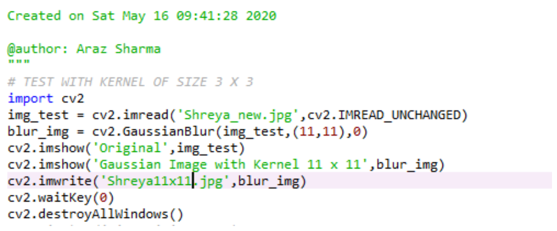


Fig. 12. Gaussian Smoothing with 11x11 Kernel



```
Created on Sat May 16 09:41:28 2020

@author: Araz Sharma
"""
# TEST WITH KERNEL OF SIZE 3 X 3
import cv2
img_test = cv2.imread('Shreya_new.jpg',cv2.IMREAD_UNCHANGED)
blur_img = cv2.GaussianBlur(img_test,(11,11),0)
cv2.imshow('Original',img_test)
cv2.imshow('Gaussian Image with Kernel 11 x 11',blur_img)
cv2.imwrite('Shreya11x11.jpg',blur_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fig. 11. Code with 11x11 Kernel

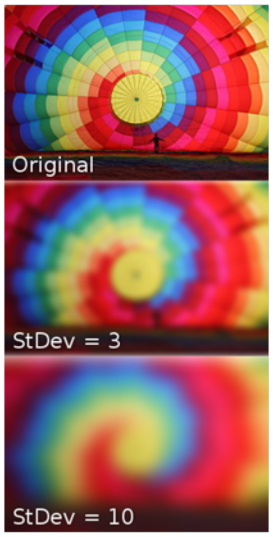


Fig. 13. Image from Wikipedia : Relation between Sigma and Gauss Blur

V. Results

Our mission objective for this paper was to get a fundamental understanding of the concept of Gaussian Smoothing and the techniques involved to implement it on actual images. We have fulfilled that objective, by diving deep into the relevant concepts, both preliminary subsequent to this topic.

We started with understanding concepts like Convolution, Linear Combination mathematics of the Gaussian distribution. We moved on to exploring their applicability and relation to the techniques for Gaussian Smoothing, and how simple mathematical operators can enable complex Image Processing Techniques.

We further understood the methodology involved in combining these mathematical operations, to achieve the Gaussian Blurring or Smoothing effects on a picture. The importance and pertinence of Convolution Kernels was understood in this application, and we were able to map this to the concepts involved with Gaussian Smoothing.

Furthermore, to get a feel of this model, and to experience it first hand, we executed it on sample images. We approached this with a solution using Python programming language, and the OpenCV Library it supports.

We explored the concepts of resizing images, with OpenCV functions, and were able to get the optimum output. Then we fed this output, as an input for Gaussian Smoothing, using the GaussianBlur method provided by the OpenCV library. We explored the parameters attached in the method, and understood their importance, by experimenting with them.

Firstly, we explored the relation between the Standard Deviation (Sigma) Parameter and the Gaussian Blur Effect, in brief, and explored the mathematical reasons for this. A sample for this relation is shown by "Fig. 13"

Then we were able to derive a relation between the Kernel Size and the Gaussian Blur effect. It is found that the blurring is more with increase in kernel size, results for which have been attached below:

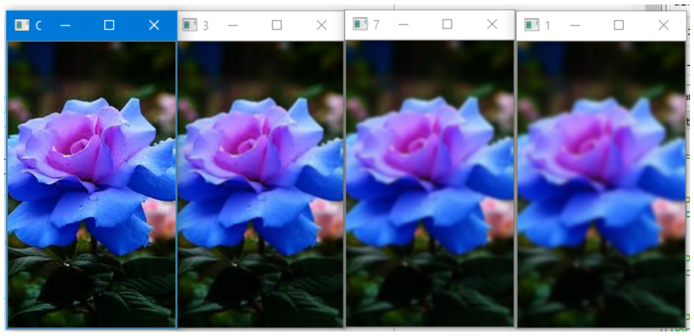
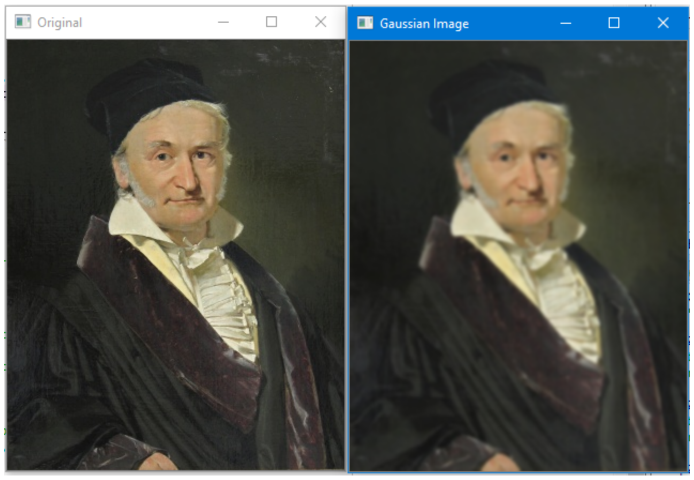


Fig. 14. Images tested by us for Kernel Relation

In the end, we were able to research some real life applications for Gaussian Smoothing. These include as a preprocess for Edge Detection algorithms [10], usage in Small Animal PET Systems, improving Captcha's Security System and Computer Vision applications.We also found out that an evolved form of the Linear Gaussian Blur can be used in detection of blurry images. [11]

VI. Conclusion

In the recent times, where every sector of the world is technologically drenched, there is not one area of the world devoid of digital images and signals. So, here in this paper, we took up one section which can be used for processing this digital information. We look into the concept of convolution, and more importantly into how this, compounded with the linear algebraic segment of the Gaussian function, amongst various others, can be implemented in the signal and image processing domain. Gaussian function finds its primary use in the reduction of noise, which is immanent in almost any kind of digital signal. We also saw the reasonableness and ease of implementation of this technique, be it using our approach of execution involving python and OpenCV, or other diverse platforms which it can easily adapt with. A supplementary asset that the Gaussian filter comes with is that of rotational symmetry, and direction independence. Despite of having a wide application-set and an assorted range of benefits, Gaussian blur does come with its set of limitations. Gaussian blurred images tend to lose fine image details and contrast. So, if there is an application requiring refined details, for instance, for the analysis of a tumour in the medical domain, or a robot pointing at a particular point, we might want to look for a different alternative. It has also proven to be incompatible with working on salt-and-pepper noise, which are sparsely occurring white and black pixels in an image. Median filters prove to be defter as compared to Gaussian filters here. Further works could focus on using this technique at its maximum efficiency in medical, robotics or other domains, [8] working on algorithms and techniques for an efficient approximation of Gaussian Function for signal and image processing, and further developing finer filters, taking this as the base, which overcome the shortcomings of this model.



References

[1] S. Maharjan, N. Manandhar, S. Lal Shrestha and R. Tandukar, ”Significance of Convolution in Image Filtering”. [Online]. Available: https://www.academia.edu/42221207/Significance_of_Convolution_in_Image_Filtering.

[2] Geraint, Estevao Hadad, M... (2011). Investigation on the effect of a Gaussian Blur in image filtering and segmentation. 393-396.

[3] T. Tyagi and V. Mishra, ”2D Gaussian Filter for Image Processing: A Study”, 2016. [Online]. Available: https://www.academia.edu/30908066/2D_Gaussian_Filter_for_Image_Processing_A_Study.

[4] S. Kopparapu and M. Satish, ”Identifying Optimal Gaussian Filter for Gaussian Noise Removal”, 2011. [Online]. Available: https://www.academia.edu/12882730/Identifying_Optimal_Gaussian_Filter_for_Gaussian_Noise_Removal.

[5] S. KIM, ”Applications of Convolution in Image Processing with MATLAB”, Academia.edu, 2013. [Online]. Available:https://www.academia.edu/13962682/Applications_of_Convolution_in_Image_Processing_with_MATLAB.

[6] M. Deisenroth, R. Turner, M. F. Huber, U. D. Hanebeck and C. Edward Rasmussen, ”Robust filtering and smoothing with Gaussian processes”, 2012. [Online]. Available: https://www.academia.edu/2835423/Robust_filtering_and_smoothing_with_Gaussian_processes.

[7] O. Zoeter, A. Ypma and T. Heskes, ”Deterministic and Stochastic Gaussian Particle Smoothing”, Academia, 2006. [Online]. Available: https://www.academia.edu/22633709/Deterministic_and_Stochastic_Gaussian_Particle_Smoothing.

[8] E. Ait Mansour and S. Barth, ”Efficient Approximation of Gaussian Function for Signal and Image Processing Applications”, 2019. [Online]. Available: https://www.academia.edu/42103047/Efficient_Approximation_of_Gaussian_Function_for_Signal_and_Image_Processing_Applications.

[9] A. DOGRA and P. BHALLA, ”Image Sharpening By Gaussian and Butterworth High Pass Filter”, 2014. [Online]. Available: https://www.academia.edu/38878515/Image_Sharpening_By_Gaussian_And_Butterworth_High_Pass_Filter.

[10] ”Gaussian Blur - an overview — ScienceDirect Topics”, Sciencedirect.com, 2020. [Online]. Available: https://www.sciencedirect.com/topics/engineering/gaussian-blur.

[11] E. Tsmoko, H. Kim and E. Izquierdo, ”Linear Gaussian Blur evolution for detection of blurry images”, 2010. [Online]. Available: https://www.academia.edu/15477382/Linear_Gaussian_blur_evolution_for_detection_of_blurry_images.

[12] M. Kumar Kaushik and R. Kasyap, ”A Review Paper on Denoting Filter using 2D Gaussian Smooth Filter for Multimedia Application”, Irjcs.com, 2016. [Online]. Available: http://irjcs.com/volumes/Vol3/iss5/05.MYCS10085.pdf.

[13] E. Davies Machine Vision: Theory, Algorithms and Practicalities, Academic Press, 1990, pp 42 - 44.

[14] R. Gonzalez and R. Woods Digital Image Processing, Addison-Wesley Publishing Company, 1992, p 191.

[15] R. Haralick and L. Shapiro Computer and Robot Vision, Addison-Wesley Publishing Company, 1992, Vol. 1, Chap. 7.

[16] . Horn Robot Vision, MIT Press, 1986, Chap. 8.

[17] D. Vernon Machine Vision, Prentice-Hall, 1991, pp 59 - 61, 214.

[18] Robert Collins, Lecture 4: Smoothing (pdf available online).

[19] Gaussian Filtering (pdf available online) by University of Auckland.

[20] Notes on Fourier transforms (pdf available online).