Introduction

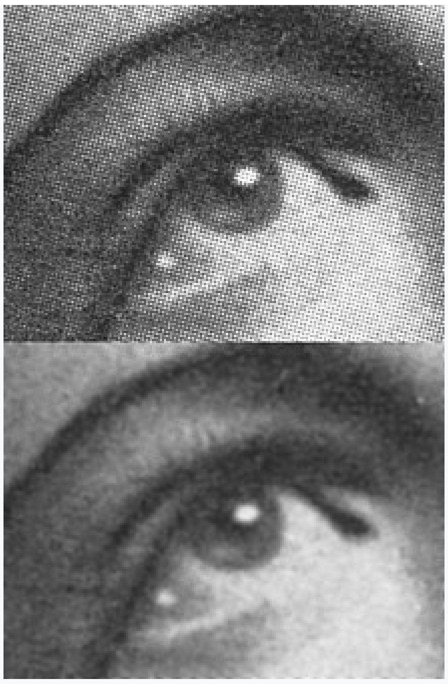
Gaussian blur also known as Gaussian smoothing is a type of image processing the results in the blurring of an image by a Gaussian function (named after mathematician and scientist Carl Friedrich Gauss).

It is a widely used effect in graphics software, typically to **reduce image noise and reduce detail.** The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination.

Gaussian smoothing is also used as a pre-processing stage in computer vision algorithms in order to enhance image structures at different scales

Mathematically, applying a **Gaussian blur** to an image is the same as **convolving the image with a Gaussian function.** This is also known as a two-dimensional Weierstrass transform. By contrast, convolving by a circle (i.e., a circular box blur) would more accurately reproduce the bokeh effect.

Since the Fourier transform of a Gaussian is another Gaussian, applying a Gaussian blur has the effect of **reducing the image's high-frequency components**; a **Gaussian blur is thus a low pass filter.**



A halftone print rendered smooth through Gaussian blur

The Gaussian blur is a **type of image-blurring filter** that **uses a Gaussian function** (which also expresses the normal distribution in statistics) for **calculating the transformation to apply to each pixel in the image.** The formula of a Gaussian function in one dimension is

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

In two dimensions, it is the product of two such Gaussian functions, one in each dimension:

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

where *x* is the distance from the origin in the horizontal axis,

y is the distance from the origin in the vertical axis, and

σ is the standard deviation of the Gaussian distribution.

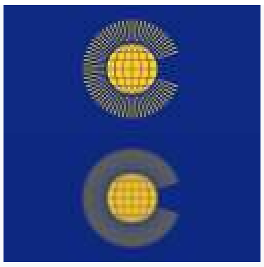
It is important to note that the origin on these axises are at the center (0, 0). When applied in two dimensions, this **formula produces a surface** whose **contours are concentric circles** with a **Gaussian distribution from the center point.**

Values from this distribution are used to **build a convolution matrix** which is **applied to the original image.** This convolution process is illustrated visually in the figure above. Each **pixel's new value** is set to a **weighted average of that pixel's neighborhood.** The **original pixel's value receives the heaviest weight** (having the highest Gaussian value) and **neighboring pixels receive smaller weights as their distance to the original pixel increases**. This results in a blur that preserves boundaries and edges better than other, more uniform blurring filters; see also scale space implementation.

In theory, the Gaussian function at every point on the image will be non-zero, meaning that the entire image would need to be included in the calculations for each pixel. In practice, when computing a discrete approximation of the Gaussian function, **pixels at a distance of more than 3*σ* have a small enough influence to be considered effectively zero**. Thus contributions from pixels outside that range can be ignored. Typically, an image processing program need only calculate a **matrix with dimensions 6*σ* ×6*σ*** to ensure a result sufficiently close to that obtained by the entire Gaussian distribution.

In addition to being circularly symmetric, the Gaussian blur can be applied to a two-dimensional image as two independent one-dimensional calculations, and so is termed a **separable filter**. That is, the effect of applying the two-dimensional matrix can also be achieved by applying a series of single-dimensional Gaussian matrices in the horizontal direction, then repeating the process in the vertical direction. In computational terms, this is a useful property, since the calculation can be performed faster in a better time complexity .

Applying successive Gaussian blurs to an image has the same effect as applying a single, larger Gaussian blur, whose radius is the square root of the sum of the squares of the blur radii that were actually applied. For example, applying successive Gaussian blurs with radii of 6 and 8 gives the same results as applying a single Gaussian blur of radius 10 Because of this relationship, processing time cannot be saved by simulating a Gaussian blur with successive, smaller blurs — the time required will be at least as great as performing the single large blur.



Two downscaled images of the Flag of the Commonwealth of Nations. Before downscaling, a Gaussian blur was applied to the bottom image but not to the top image. The blur makes the image less sharp, but prevents the formation of moiré pattern aliasing artifacts.

Gaussian blurring is commonly used when reducing the size of an image.
When downsampling an image, it is common to apply a low-pass filter to the image prior to resampling. This is to **ensure that spurious high-frequency information does not appear in the downsampled image** (aliasing). Gaussian blurs have nice properties, such as **having no sharp edges, and thus do not introduce ringing into the filtered image.**

Gaussian blur is a **low-pass filter**, attenuating high frequency signals.

Its amplitude Bode plot (the log scale in the frequency domain) is a **parabola.**

Gaussian filter with standard deviation σ_f we can determine how much does the Gaussian filter reduce the standard deviation of pixel values in the picture? Assume the grayscale pixel values have a standard deviation σ_X , then after applying the filter the reduced standard deviation σ_r can be approximated as $\sigma_r \approx \frac{\sigma_X}{\sigma_f 2\sqrt{\pi}}.$

Sample Gaussian matrix

This sample matrix is produced by sampling the Gaussian filter kernel (with σ = 0.84089642) at the midpoints of each pixel and then normalizing. The center element (at [0, 0]) has the largest value, decreasing symmetrically as distance from the center increases. Since the filter kernel's origin is at the center

$$\begin{bmatrix} 0.00000067 & 0.00002292 & \mathbf{0.00019117} & 0.00038771 & \mathbf{0.00019117} & 0.00002292 & 0.00000067 \\ 0.00002292 & 0.00078633 & 0.00655965 & 0.01330373 & 0.00655965 & 0.00078633 & 0.00002292 \\ \mathbf{0.00019117} & 0.00655965 & 0.05472157 & 0.11098164 & 0.05472157 & 0.00655965 & \mathbf{0.00019117} \\ 0.00038771 & 0.01330373 & 0.11098164 & \mathbf{0.22508352} & 0.11098164 & 0.01330373 & 0.00038771 \\ \mathbf{0.00019117} & 0.00655965 & 0.05472157 & 0.11098164 & 0.05472157 & 0.00655965 & \mathbf{0.00019117} \\ 0.00002292 & 0.00078633 & 0.00655965 & 0.01330373 & 0.00655965 & 0.00078633 & 0.00002292 \\ 0.00000067 & 0.00002292 & \mathbf{0.00019117} & 0.00038771 & \mathbf{0.00019117} & 0.00002292 & 0.00000067 \end{bmatrix}$$

The element 0.22508352 (the central one) is 1177 times larger than 0.00019117 which is just outside 3σ.

Edge detection

Gaussian smoothing is commonly used with edge detection. **Most edge-detection algorithms are sensitive to noise**; the 2-D Laplacian filter, built from a discretization of the Laplace operator, is highly sensitive to noisy environments.

Using a Gaussian Blur filter before edge detection aims to **reduce the level of noise in the image**, which improves the result of the following edge-detection algorithm. This approach is commonly referred to as Laplacian of Gaussian, or LoG filtering

Photography

Lower-end digital cameras, including many mobile phone cameras, commonly **use gaussian blurring to cover up image noise** caused by higher ISO light sensitivities.

Gaussian blur is **automatically applied as part of the image post-processing of the photo by the camera software**, leading to an irreversible **loss of detail.**

Review of Literature

In digital image processing, the blur can be originating by many factors such as defocus, unbalance, motion, noise and others. As we know our vision is one of the important senses in our body. So we can say that image processing plays an important role in our life. An image is a 2-D digital function or we can say that it can be considered as a matrix in which the row and column represent the position of image and the elements represents its pixel value. An image is made up of digital data by inputting it and the output of this digital data is an image. An important problem in image processing is its blurring problem which degrades its performance and quality. Gaussian function is used firstly to degrade the image quality and blurring an image because it is a low pass filter.In digital world there are mainly three type of blur we study mainly. These are of three types i.e.- 1. Average Blur 2. Gaussian Blur 3. Motion Blur Gaussian blur: Atmospheric condition causes image degradation this effect is called Gaussian effect. It is a type of filter that is used for normal distribution to calculating the transformation to apply to each and every pixel in an image. This type of filter shows a bell shaped curve for the image. This shows that blurring is maximum on the center and less on edges. Gaussian blur is mainly caused by Gaussian function. It blends some pixel in-clemently. In this paper we consider the Gaussian blur for improving the image which is blurred and noisy and the blur is removed by using some method. Image quality is the main task of image processing world. Image deblurring basically is to get the sharp image by removing its noise and blur from an image Blurring may be due to many factors such as noise, dust, camera shake, object shake etc. Deblurring may be done by different methods such as sharpening the edges, filling the pixels which are blank and removing the noise. So the main thing in this review is to obtain the deblurred image by using a degraded and blurry image.

Report on the present investigation:

Implementation

A Gaussian blur effect is typically generated by convolving an image with an FIR kernel of Gaussian values.

In practice, it is best to take advantage of the Gaussian blur’s separable property by dividing the process into two passes. In the first pass, a one-dimensional kernel is used to blur the image in only the horizontal or vertical direction. In the second pass, the same one-dimensional kernel is used to blur in the remaining direction. The resulting effect is the same as convolving with a two-dimensional kernel in a single pass, but requires fewer calculations.

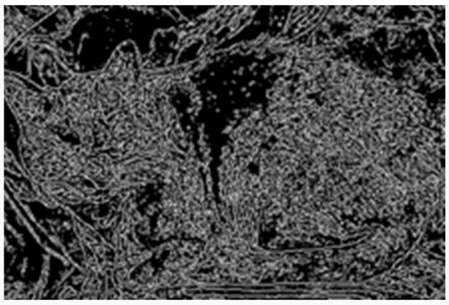
Discretization is typically achieved by sampling the Gaussian filter kernel at discrete points, normally at positions corresponding to the midpoints of each pixel. This reduces the computational cost but, for very small filter kernels, point sampling the Gaussian function with very few samples leads to a large error.

In these cases, accuracy is maintained (at a slight computational cost) by integration of the Gaussian function over each pixel's area.

When converting the Gaussian’s continuous values into the discrete values needed for a kernel, the sum of the values will be different from 1. This will cause a darkening or brightening of the image. To remedy this, the values can be normalized by dividing each term in the kernel by the sum of all terms in the kernel.

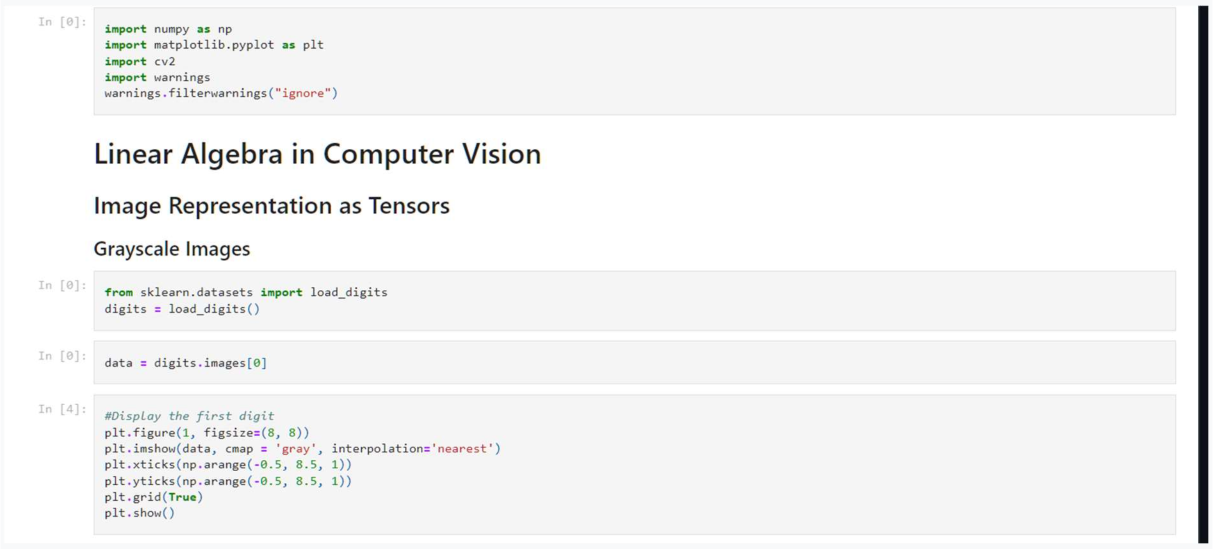
The efficiency of FIR breaks down for high sigmas. Alternatives to the FIR filter exist. These include the very fast multiple box blurs, the fast and accurate IIR Deriche edge detector, a "stack blur" based on the box blur, and more.

Common uses



This shows how smoothing affects edge detection. With more smoothing, fewer edges are detected

Code explanation:



In [0]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import warnings
warnings.filterwarnings("ignore")
```

Linear Algebra in Computer Vision

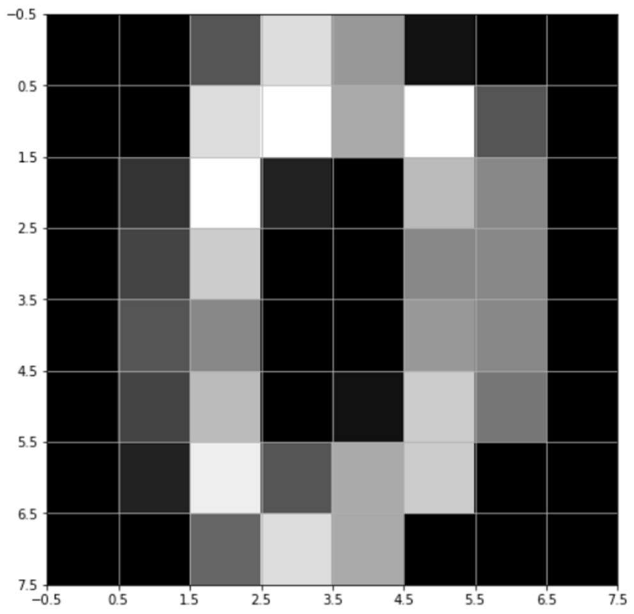
Image Representation as Tensors

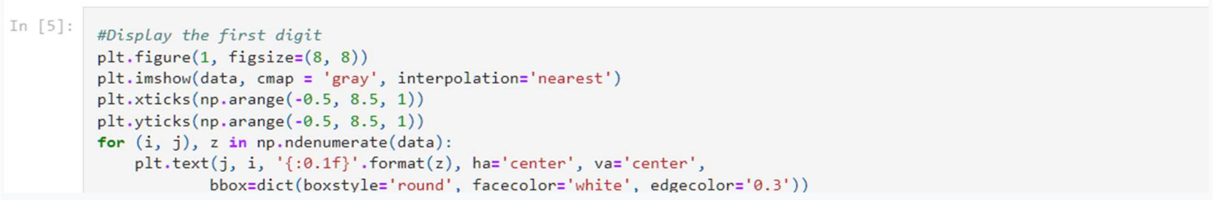
Grayscale Images

In [0]:
```
from sklearn.datasets import load_digits
digits = load_digits()
```

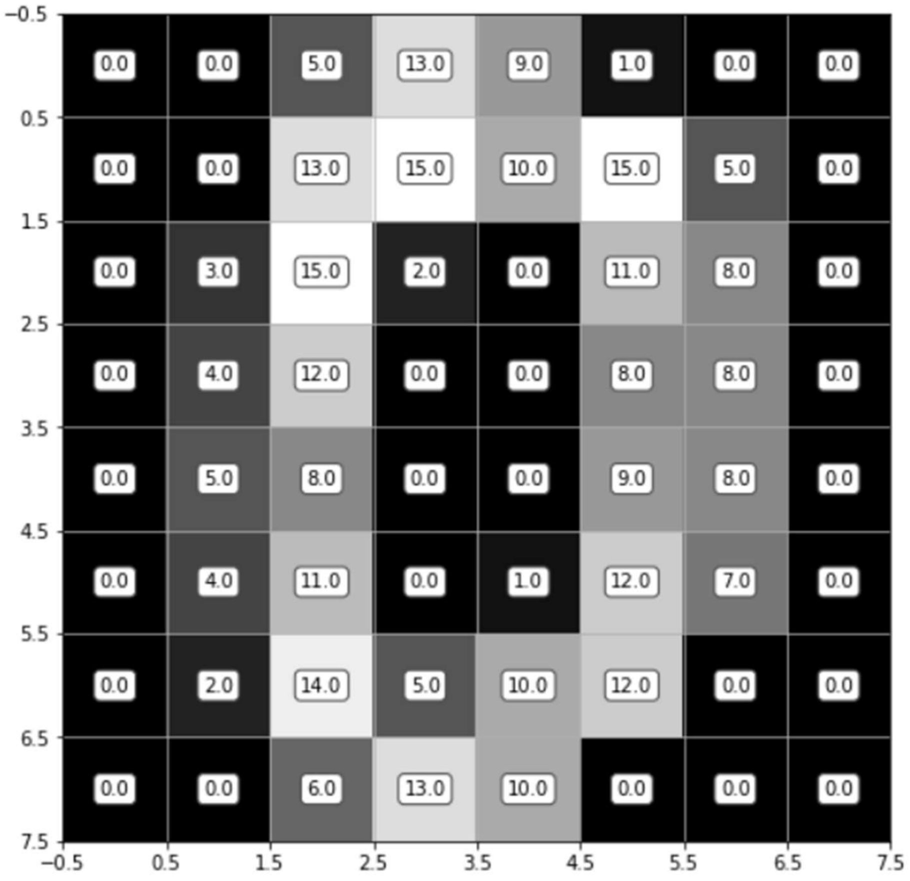
In [0]:
```
data = digits.images[0]
```

In [4]:
```
#Display the first digit
plt.figure(1, figsize=(8, 8))
plt.imshow(data, cmap = 'gray', interpolation='nearest')
plt.xticks(np.arange(-0.5, 8.5, 1))
plt.yticks(np.arange(-0.5, 8.5, 1))
plt.grid(True)
plt.show()
```



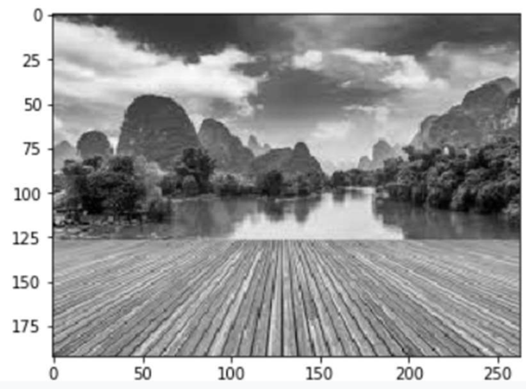


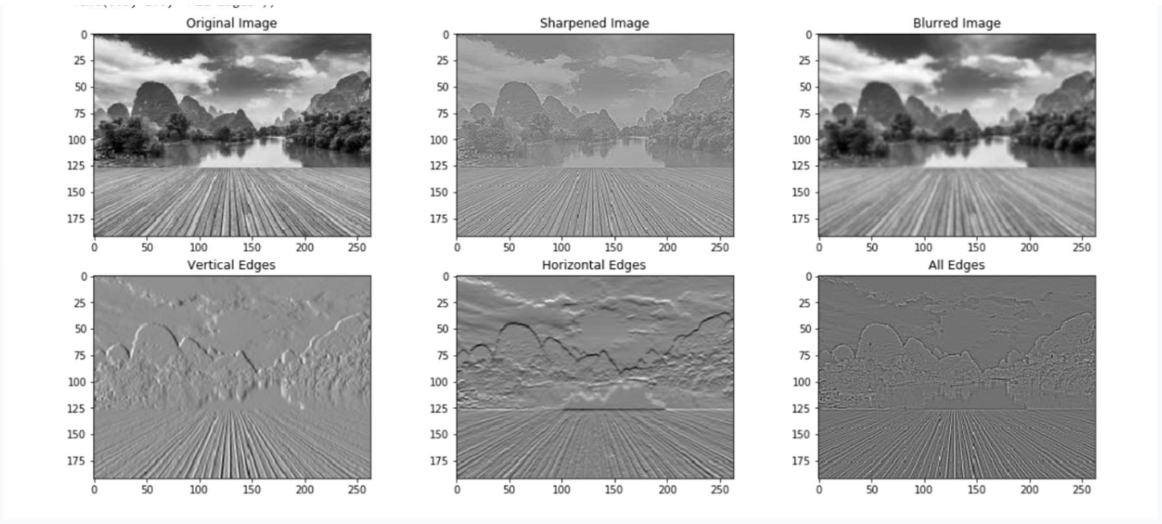
In [5]:
```
#Display the first digit
plt.figure(1, figsize=(8, 8))
plt.imshow(data, cmap = 'gray', interpolation='nearest')
plt.xticks(np.arange(-0.5, 8.5, 1))
plt.yticks(np.arange(-0.5, 8.5, 1))
for (i, j), z in np.ndenumerate(data):
    plt.text(j, i, '{:0.1f}'.format(z), ha='center', va='center',
             bbox=dict(boxstyle='round', facecolor='white', edgecolor='0.3'))
```



In [15]:
```python
# image in grayscale
from skimage.color import rgb2gray
gray = rgb2gray(img)
plt.imshow(gray, cmap = 'gray')
```

Out[15]:
```
<matplotlib.image.AxesImage at 0x7f8469cc17f0>
```

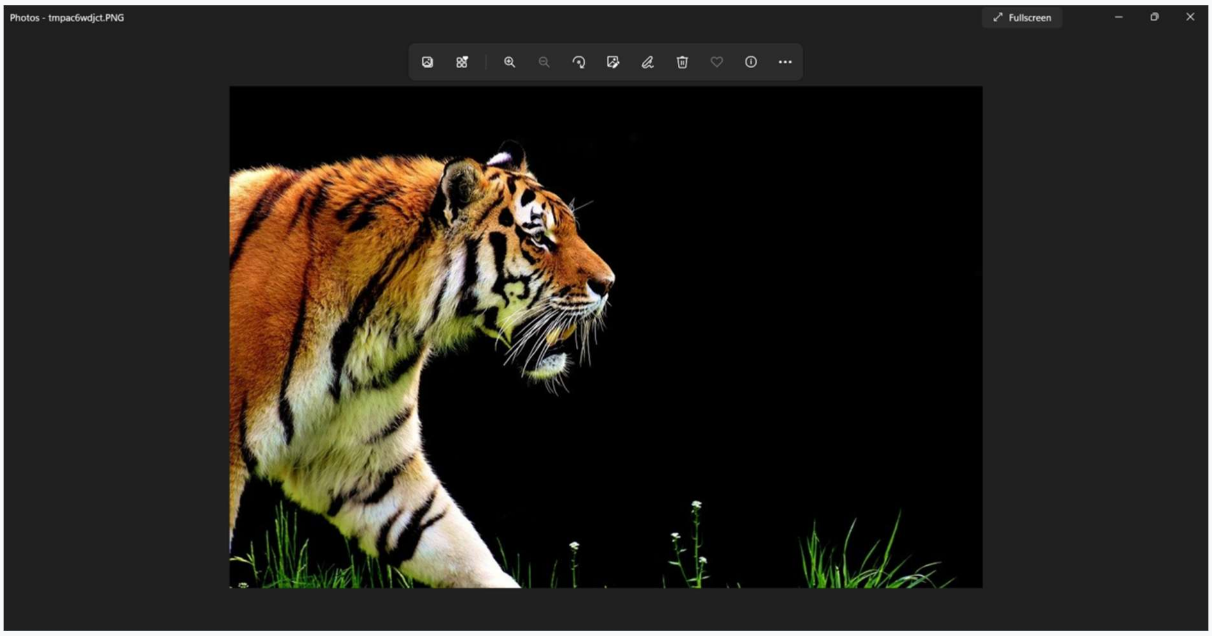




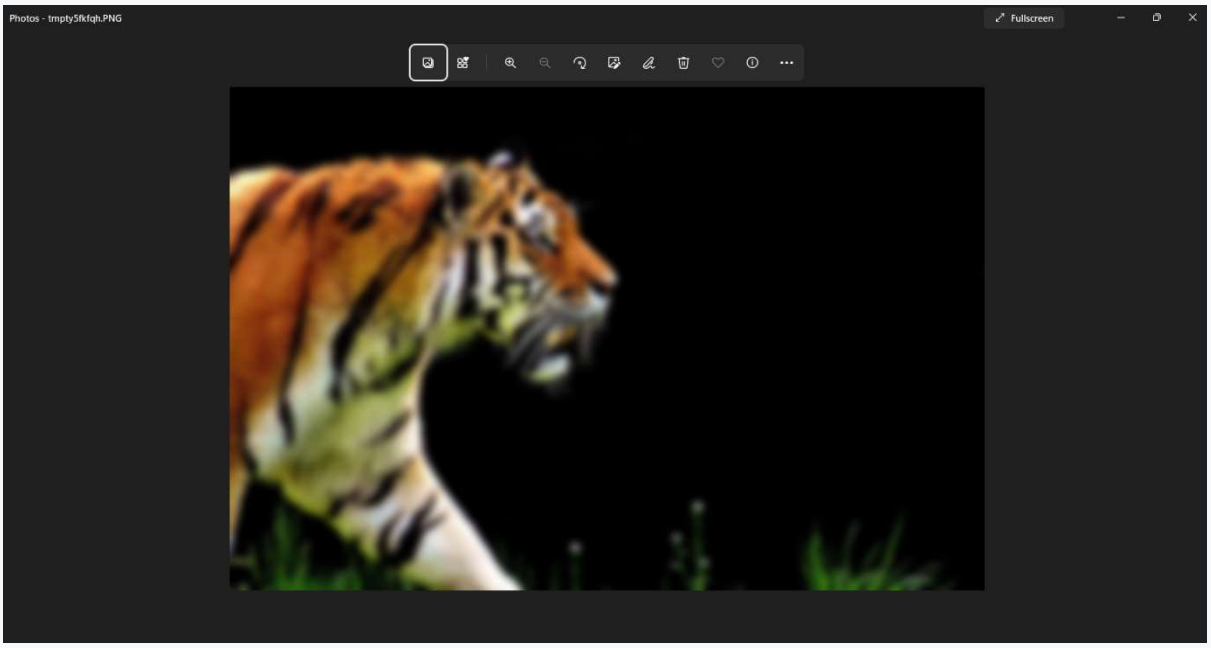
Python code using Pillow module:

```
#Import required Image library
from PIL import Image, ImageFilter

#Open existing image
OriImage = Image.open('image.jpeg')
OriImage.show()

#Applying GaussianBlur filter
gaussImage = OriImage.filter(ImageFilter.GaussianBlur(5))
gaussImage.show()

#Save Gaussian Blur Image
gaussImage.save('images/gaussian_blur.jpg')
```

output:Oringinal image



Blurred image



The medical diagnosis is improving: today it has become possible to visualize internal organs without any surgical intervention, and we can even track metabolic process with imaging technology, such as Positron Emission Tomography. However, the presence of noise is unavoidable: sometimes noise is inherent to the process of acquiring images and can affect the diagnosis of a specialist, if, for instance a small structure, like a small tumor, disappears in the image due to excessive noise. Thus, we need to filter data before using it. The filtering process involves getting rid of as much noise as possible while losing a minimum of information. Indeed, noise can have a negative impact on the diagnosis: small tumors that would disappear in the filtered image would seriously endanger it. Therefore, it is critical to understand the effects of the filtering techniques we use, and situations where they should be used. First, the proposed work investigates the effects of a Gaussian blur on a chosen image and evaluates the results of the filtering using a qualitative index, proposed in [1], then it tackles an application to segmentation, using the Gaussian blur to generate a mask. II. MATERIALS AND METHODS The original image was taken from the example of ImageJ ("Bridge") [2]. Using the —Add noise‖ built-in function, it was added noise and then filtered, using the —Gaussian Blur‖, also a built-in function from ImageJ. 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 [3]:

$$f * g[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m].g[n-m] \qquad (1)$$

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

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

The kernel size will be detailed in further section. One example of Gaussian distribution is the

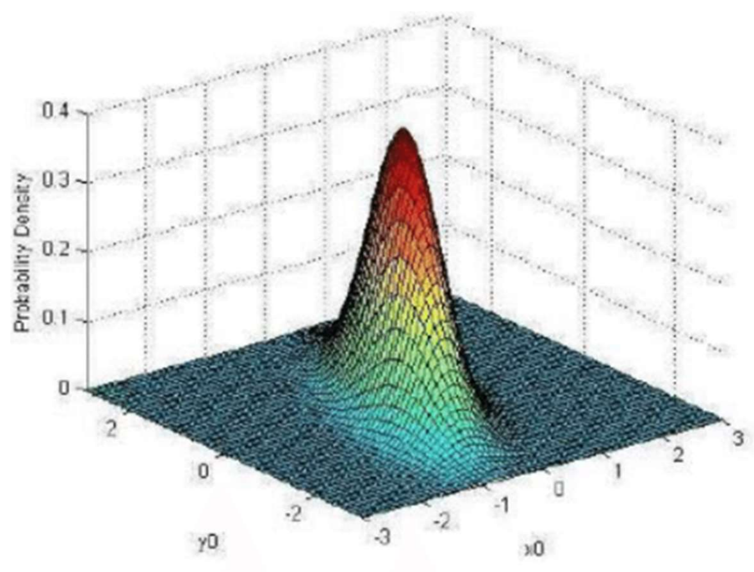


Figure 1. Gaussian distribution

Fig. 1: Figure 1. Gaussian distribution 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 neighborhood. Thus, the original pixel's value receives the heaviest weight (having the highest Gaussian value) and neighboring pixels receive smaller weights as their distance to the original pixel increases

A. Tools for quantization of the results The convolution with the Gaussian kernel is a low-pass filter [2], that smoothes the borders of the image. The parameter that we use to define the filter is the variance of the Gaussian distribution, since it affects dramatically the results of the filtering [2]. As the present work focuses on the effects of the Gaussian blur, as proposed in [2] and [3], we will base our results on the comparison of the filtered image with the original image through the quality factor given by [1]:

$$Q(f,g) = \frac{\sigma_{f,g}}{\sigma_g{}^2.\sigma_f{}^2}.2.\frac{\overline{f.g}}{\overline{f}^2+\overline{g}^2}.2.\frac{\sigma_g{}^2.\sigma_f{}^2}{\sigma_f^2+\sigma_g^2} \qquad (3)$$

Where g is the gold standard (original image, without noise) and f is the filtered image. is the covariance between the two images, is the variance of the image f, is the variance of the image g and̄and̄ are the means of the image f and g, respectively. This quality factor compares the covariance between the two images, the distortion in luminance (the mean values) and the distortion in contrast (the variance values). This method is closer to human perception than other methods proposed in literature [5], [6]. The indicator we used to evaluate the noise level in the images is the Signal-to-Noise Ratio (SNR), given by:

$$SNR = 20.\log\left(\frac{\sigma_{signal}}{\sigma_{noise}}\right) \qquad (4)$$

The SNR basically compares the intensity of the signal with the intensity of the noise. A higher SNR means better image quality. B. Methodology for generation of noisy images We used ImageJ [2] and

Eclipse [6] for the availability of the documentation and the possibility to generate our codes in the form of Plugins. The image we chose to work on is the "Bridges" image, available in ImageJ examples [2]. This image was chosen to enable visualizing changes after the filtering. As mentioned above, in order to evaluate the filter performance, different values of SNR were taken: 5, 10 and 15, with a Gaussian Noise model. This represents high, medium and low noise in the image, respectively. We also tested an impulsive noise on the original image, commonly called "Salt and Pepper noise". To add noise, we used the function provided by ImageJ (—Noise‖ function). The first step was to measure the mean and standard deviation of the original image with the resulting images. We then used (4), to calculate the noise variance and we added the selected noise to image. Table 1 shows the calculated variance for specific SNR.

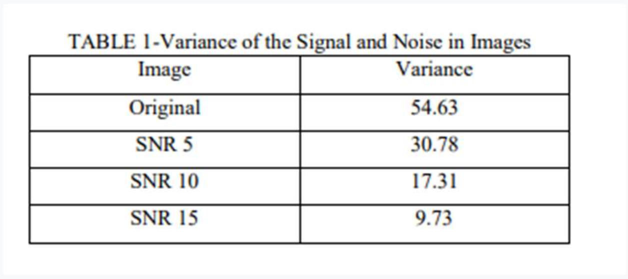


TABLE 1-Variance of the Signal and Noise in Images

Image	Variance
Original	54.63
SNR 5	30.78
SNR 10	17.31
SNR 15	9.73

form a variance for the noise. Fig. 2 shows the original image along with the noisy images. III. EVALUATION OF THE FILTER As exposed before, the main characteristic to be evaluated is the variance of the Gaussian function, since the kernel window size is calculated using the variance values in the ImageJ implementation [2]. We used three values for this evaluation: variance of 1, 3 and 10. The Table 2 shows the quality indicators for the resulting filtering:

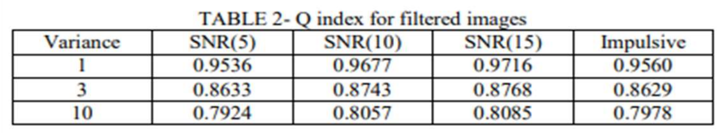
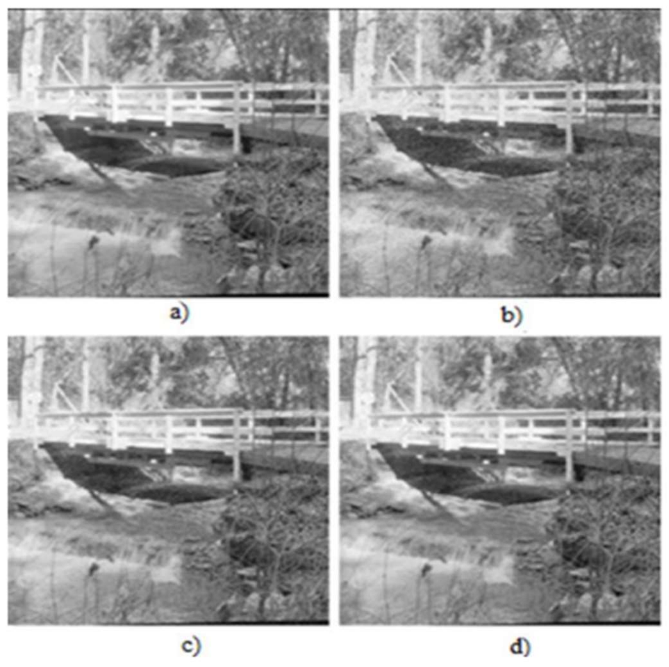


TABLE 2- Q index for filtered images

Variance	SNR(5)	SNR(10)	SNR(15)	Impulsive
1	0.9536	0.9677	0.9716	0.9560
3	0.8633	0.8743	0.8768	0.8629
10	0.7924	0.8057	0.8085	0.7978



In the first row, the Gaussian blurring method shows similar results in all cases of SNR. The second row shows that when we increase the variance of the Gaussian function, the quality values are closer to each other and lower than the quality values from the first case. By comparing this data to the smallest variance data, we conclude that the variance of the Gaussian function has a greater influence than the SNR values. The last row confirms that the Gaussian variance is more relevant than the noise variance to the final quality index, especially when we use a large variance in the function. With the data from the Table 2, we can conclude that the Gaussian filter response is closely related to the variance of the Gaussian function, and not to the variance of the noise

present in the image (Signal to Noise Ratio). Therefore, the Gaussian filtering should rather be used in images that have a low SNR (i.e. contain a lot of noise), since the results are close in all the SNR investigated. For images containing less noise, other filters could be more efficient. Fig. 3 shows how the filtering affects the image quality. Because the variance of the Gaussian function is small, the image is not so blurry after the filtering.

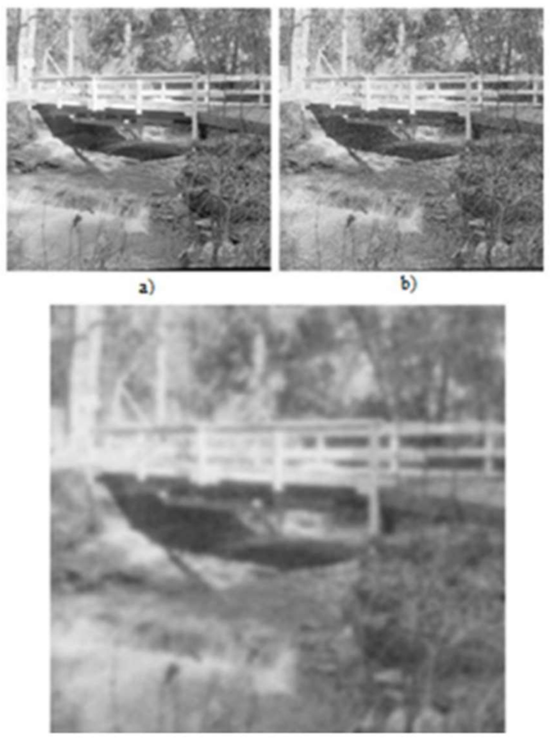


Figure 3. a) Original Image b) Noisy image c) Filtered Image

IV. APPLICATION TO SEGMENTATION OF MEDICAL IMAGES Segmentation is the process of extracting parts of the image [4]. This is important to perform accurate measurements. It is often difficult to the computer and expensive in human labor if performed manually. Consequently, developing an automatic and trustworthy method of segmentation is of the greatest importance. The Gaussian blur tool used with larger variance values is not very effective as a filter, due to the distortion that occurs in the borders, especially in the case of medical images. Nevertheless, there is one application, proposed in [7] that shows to what extent the large variance values could be used to construct masks applied in segmentation in the medical imaging area Since each tissue from the human body responds in a different way to tomography, they all have different values of intensity in the image. A tissue that absorbs more radiation glows more than a tissue that absorbs less radiation. This could be seen as if each tissue had a region that represents itself in the histogram of intensity. Thus, we could isolate a region from histogram and use it as a mask to the original image, after some processing and blurring. Eventually, as proposed in [7], we could perform a segmentation of the liver in an automatic way. The method starts with selecting from the histogram the peaks correspondent to the liver. These values are obtained from clinical experimentation [7]. The algorithm then chooses the upper part of the image, since the method focuses on liver, to avoid the presence of other organs with similar structure [7]. The next step is to use a Gaussian blur to generate a mask. In this case, it has a variance of 7. After the mask is ready, we use a binary operation: when the mask pixel is white, the image original pixel is kept, otherwise it is discarded. The whole process is illustrated in Fig. 4, from the original image to the segmented image. It is possible to see the effects of a large variance in image.

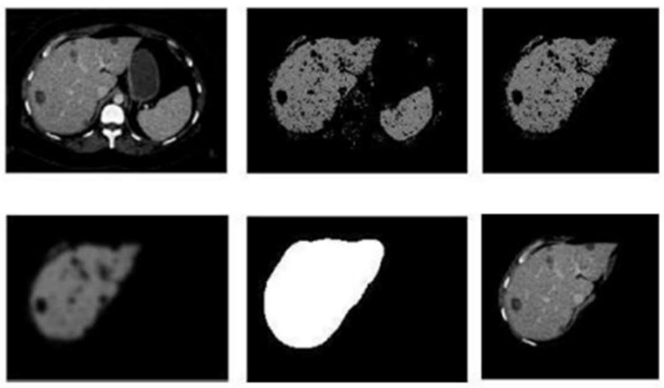


Figure 4. Segmentation process of a liver tomography

II. LITERATURE REVIEW

Mane and Panwar [1] represented its approach in edge detection field using the blind deconvolution methodology with canny edge detection method. Using this method ringing effect of a blurring image is reduced. It is used for mainly the motion blurred image in which there is no information of blur kernel is given. Firstly, the image is degraded using degraded model and then recovered the original image.

Saini and Himral [2] focuses on mainly image restoration concept for getting true image from the noisy and uncorrupted image. In it there is use of **Blind Deconvolution strategy** for recover a sharp image using the image restoration technique. A true image provides the valuable information.

S. Suryanarayana, Deekshatulu, Kishore and Rakesh Kumar [3] In this paper a novel algorithm for Gaussian noise estimation and removal is proposed by using 3x3 subwindows in which the test pels appears. The standard deviation (STD) for all sub-windows are used to defined reference STD (σref) and minimum (σmin) and maximum (σmax) standard deviations. The average STD (σavg) is then deliberated as the average of those STDs of all sub-windows whose STD falls within the range of [σmin, σmax]. This σavg is used for identifying and removing additive Gaussian noise. The performance is differentiate with that of the standard mean filter. The present scheme is outperforming than the standard mean filter.

De and Masilamani [4] presented a new method concerned the NR-IQA (No-reference Image Quality assessment). In this paper the standard deviation of Gaussian filter kernel is used for different images. This concept is used for deblurring the images. **When there is blur increases in image the frequency component is decreases**. So it is an **image quality measure for the image**. Image Quality measure is obtained after center Fourier transform for detecting sharpness in an image.

Gavilan, R. Arahal and Ierardi [5] presented their work on International Journal For Technological Research In Engineering Volume 5, Issue 10, June-2018 ISSN (Online): 2347 - 4718 www.ijtre.com Copyright 2018.All rights reserved. 3943 roll angle estimation. The **estimation of plane angles is used to remove the blur in an image**. Gradient algorithm is also used in this technique. These are basically vision enhanced methods for the aerial images.

It is related to the automatic landing methods. Saleh Al-amri,Kalyankar and Khamitkar S.D. [6] studied the method of Restored Gaussian Blurred Images when there is no information of PSF is given. In this paper different type of deblurring methods are compared and different experiments are done on **different type of techniques, such as Wiener Filter, Lucy-Richardson Algorithm Method, Blind Deconvolution Algorithm Method, Regularized Filter Deblurring Method etc**.

Singh and Sahu [7] purposed a method for deblurring images using **transformation spread functions** (TSFs). Quality measurement parameters are also calculated in this. HDR images are derived by PSFs. **Deblurring is said to be a method used to sharp and clear the image.** In this paper, the PSF is estimated for blurry image.

Madghe and Kasturiwala [8] discussed the methodology of image enhancement for improving the quality of an image by using GLAS algorithm. The image enhancement is basically a process of obtaining the original image which is free from blur and noise. The two methods of **image enhancement are: Spatial domain and Transform domain.** These methodology increases the quality of an image. Image enhancement is the demand of today's world for improving the quality of image.

Henawy, Amin, Ahmed, Adel [9] concerned about the blur kernel and the deblurred image both. They also introduced the blur type, noise model and different deblurring techniques. **Image blur may be occur due to many reasons** such as camera shake, object movements etc. After that he obtained image degrades and we cannot see it clearly. According to this paper, all captured images are less or more blurry. And there are a lot of factors for degrading the quality of an image.

Kamboj and Moudgil [10] worked on Hybrid Median Filter to remove the noise or blur from an image. They considered different filtering techniques for sharpening an image but the suitable technique purposed by them is hybrid median filter by which we can get the better result of a blurry image. By using **different filter the value of PSNR is calculated and the hybrid filter gets the better result than median filter.**

Tyagi and Singh [11] have discussed about the detection of regionto be inpainting in an image and then fill the hole and scratches by reconstructing them to get fine image. There are two approaches used for image deblurring one is texture synthesis and inpainting to restore the image. For filling the pixel value two types of algorithm are used i.e. Boundary fill and Flood fill algorithm.

Bhawre and Ingle [12] presented an approach which is based on Group based sparse representation (GSR). The technique used in this approach is self-adaptive dictionary learning. In this papers, the three image restoration problem such as inpainting, deblurring and compressive sensing recovery. In this method, the discontinuities of a blurred image are filled with inpainting method

Results and Discussions:

Parallel computer systems popularity is highly increasing for their **ability to solve complex problems** and to **deal with the significant increase in the data sizes and huge data sets** (Bozkurt et al, 2015). **Image processing** has moved from the sequential approach to the **parallel programming approach** as the image processing applications such as **image filtering and convolution consume time**, **resources** and the **complexity increases with the image size** (Reddy et al, 2017). Convolution is known to be a complex mathematical operation that is highly used in image processing (Novák et al, 2012). The **parallel programming approach for the image processing** is done through a **multi-core central processing unit CPU and graphics processing unit GPU**. **GPU** with the presence of the **multithreaded parallel programming capabilities** provided by CUDA gained more popularity with image processing applications as it **increased the speedup factor by hundreds to thousands compared to the CPU as the CPU speedup factor is limited to the number of available cores** (Reddy et al, 2017). We observe the performance of Image convolution using Gaussian filter technique on parallel systems using Google Colaboratory platform and studies the effect of using Google Colaboratory platform. We uses **two different parallel systems**: **multi-core** central processing unit **CPU** and graphics processing unit **GPU**. Section 1 presents the introduction on the parallel systems: multi-core central processing unit CPU and graphics processing unit GPU, Gaussian Blur filter and related work. Section 2 presents the experiment, the architecture of the platform used, the results and the final section is the conclusion.

2.1 CPU and GPU

A **Multicore CPU** is a **single computing component** with **more than one independent core**. OpenMP (**Open Multi-Processing**) and TBB (**Threading Building Blocks**) are **widely used** application programming interfaces (**APIs**) to **make use of multicore CPU** efficiently (Polesel et al, 2000). In this study, a **general purpose platform using Python pymp tool is used to parallelize the algorithm**. On the other hand, **GPU** is a single instruction and multiple data (**SIMD) stream architecture** which is suitable for applications where the **same instruction is running in parallel on different data elements.** In image convolution, **image pixels are treated as separate data elements** which makes GPU architecture more suitable for parallelizing the application In this study, we are using CUDA platform with GPU since **CUDA is the most popular platform used to increase the GPU utilization**. The **main difference** between CPU and GPU is the **number of processing units**. In **CPU it has less processing units** with cache and control units while in **GPU it has more processing units** with its own cache and control units. **GPUs contain hundreds of cores which causes higher parallelism compared to CPUs.**

2.2 NVIDIA, CUDA Architecture and Threads

The **GPU follows the SIMD** programming model. The GPU **contains hundreds of processing cores**, called the **Scalar Processors (SPs)**. **Streaming multiprocessor (SM) is a group of eight SPs** forming the graphic card. **Group of SPs in the same SM execute the same instruction at the same time hence they execute in Single Instruction Multiple Thread (SIMT) fashion** (Lad et al, 2012). Compute Unified Device Architecture **(CUDA)**, developed by NVIDIA, is a **parallel processing architecture**, which with the help of the GPU produced a significant performance improvement. **CUDA enabled GPU is widely used in many applications as image and video processing in chemistry and biology,** fluid dynamic simulations, computerized tomography (CT), etc (Bozkurt et al, 2015). CUDA is an extension of C language for executing on the GPU, that **automatically creates parallelism with no need to change program architecture for making them multithreaded.** It also supports memory scatter bringing more flexibilities to GPU (Reddy et al, 2017). The CUDA API allows the **execution of the code** using a **large number of threads,** where **threads are grouped into blocks** and **blocks make up a grid**. **Blocks are serially assigned for execution on each SM** (Lad et al, 2012).

2.3 Gaussian Blur Filter The Gaussian blur,

is a **convolution technique** used as a pre-processing stage of many computer vision algorithms used for **smoothing**, **blurring** and **eliminating noise in an image** (Chauhan, 2018). Gaussian blur is a linear **low-pass filter**, where the **pixel value is calculated using the Gaussian function** (Novák et al, 2012). The 2 Dimensional (2D) Gaussian function is the product of two 1 Dimensional (1D) Gaussian functions, defined as shown in equation (1)

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

(1) where (x,y) are coordinates and ‘σ’ is the standard deviation of the Gaussian distribution. The linear spatial filter mechanism is in the movement of the center of a filter mask from one point to another and the value of each pixel (x, y) is the result of the filter at that point is the sum of the multiplication of the filter coefficients and the corresponding neighbor pixels in the filter mask range (Putra et al, 2017). The outcome of the Gaussian blur function is a bell shaped curve as shown in Figure 1 as the pixel weight depends on the distance metric of the neighboring pixels (Chauhan, 2018). The filter kernel size is a factor that affects the performance and processing time of the convolution process. In our study, we used odd numbers for the kernel width: 7x7, 13x13, 15x15 and 17x17.

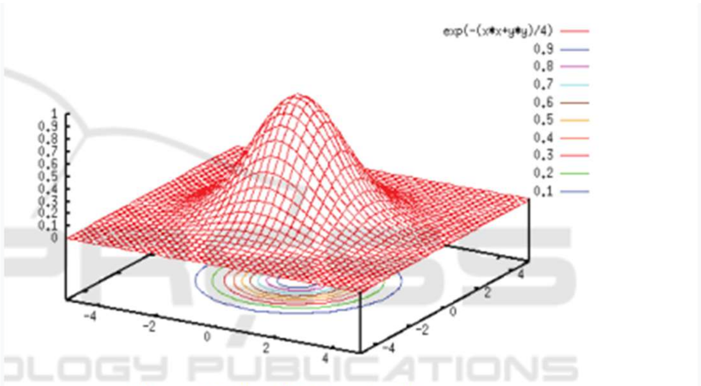


Figure 1: The 2D Gaussian Function.

Figure 1: The 2D Gaussian Function.

2.4 Related Work

Optimizing image convolution is one of the important topics in image processing that is being widely explored and developed. The **effect of optimizing Gaussian blur** by running the filter on **CPU multicore systems** and its improvement from single CPU had been explored by Novák et al. (2012). Also, exploring the effect of **running Gaussian blur filter using CUDA** has been explored by Chauhan (2018). In the previous studies, the focus was getting the **best performance from multicore CPUs or from GPU compared to the sequential code**. Samet et al. (2015) presented a comparison between the speed up of real time applications on CPU and GPU using C++ language and Open Multi-Processing OpenMP. Also, Reddy et al. (2017) presented a comparison between the performance of CPU and GPU was studied for image edge detection algorithm using C language and CUDA on NVIDIA GeForce 970 GTX. In our study we are exploring the performance improvement between two different parallel systems, CPU multicore and GPU using python parallel libraries pymp and CUDA respectively. We are using Intel Xeon CPU and NVIDIA Tesla P100 GPU.

3 EXPERIMENTAL SETUP

In our experiment, we are **using Google Colaboratory or "colab"** to run our code. Google Colaboratory is a **free online cloud-based Jupyter notebook environment** that allows us to **train PyCUDA** and which gives you easy, pythonic access NVIDIA's CUDA parallel computation API. To be able to run our code on Google Colabs, we used the Python programming language. **Python pymp library is used for the multiprocessor code.** This **package brings OpenMP-like functionality to Python.** It takes the good qualities of OpenMP such as minimal code changes and high efficiency and combines them with the Python Zen of code clarity and ease-of-use. **Python PyCUDA library is used to be able to call CUDA function in our python code and PyCUDA gives you easy, pythonic access to NVIDIA's CUDA parallel computation API.**

3.1 Architecture of the Used **GPU NVIDIA Tesla P100 GPU accelerators** are one of the **advanced data center accelerators**, powered by the breakthrough NVIDIA Pascal™ architecture and designed to **boost throughput and save money for HPC and hyperscale data centers.** The newest addition to this family, Tesla P100 for PCIe enables a single node to replace half a rack of commodity CPU nodes by delivering lightning-fast performance in a broad range of HPC applications (Nvidia Corporation, 2016). CUDA toolkit 10.1 is used.

3.2 Architecture of the Used CPU Intel Xeon is a high-performance version of Intel desktop processors intended for use in servers and high-end workstations. Xeon family spans multiple generations of microprocessor cores. Two CPUs available with two threads per core and one core per socket

3.3 Experiment Results

Three images are used in our experiment, their particular sizes are of 256 x 256, 1920 x 1200 and 3840 × 2160. We ran our Gaussian filter algorithm on the images using different kernel sizes and calculated the average processing time of 10 runs. First, the color channels of the images are being extracted to red channel, green channel and blue channel. In the second step the convolution of each channel respectively with the Gaussian filter. Using the pycuda library in python code, a CUDA C function had been used to be able to run the code on the GPU. The data is transferred from host to device and after the convolution operation is being brought back to the host. In the end we merge all the channels together to get the output of the blurred image. The processing time calculated is the time taken by the convolution function of the filter on the CPU using two threads and on the GPU as shown in Table 1 and Table 2 respectively. The processing time and speedup are shown in Figure 2 and 3 respectively. Figures 4 and 5 respectively show the original image and blurred image of resolution 256 x 256 using 7 x 7 filter kernel. **From the results** of Bozkurt et al. (2015), it is observed that our **performance speedup on GPU was higher** than the one proposed by Bozkurt et

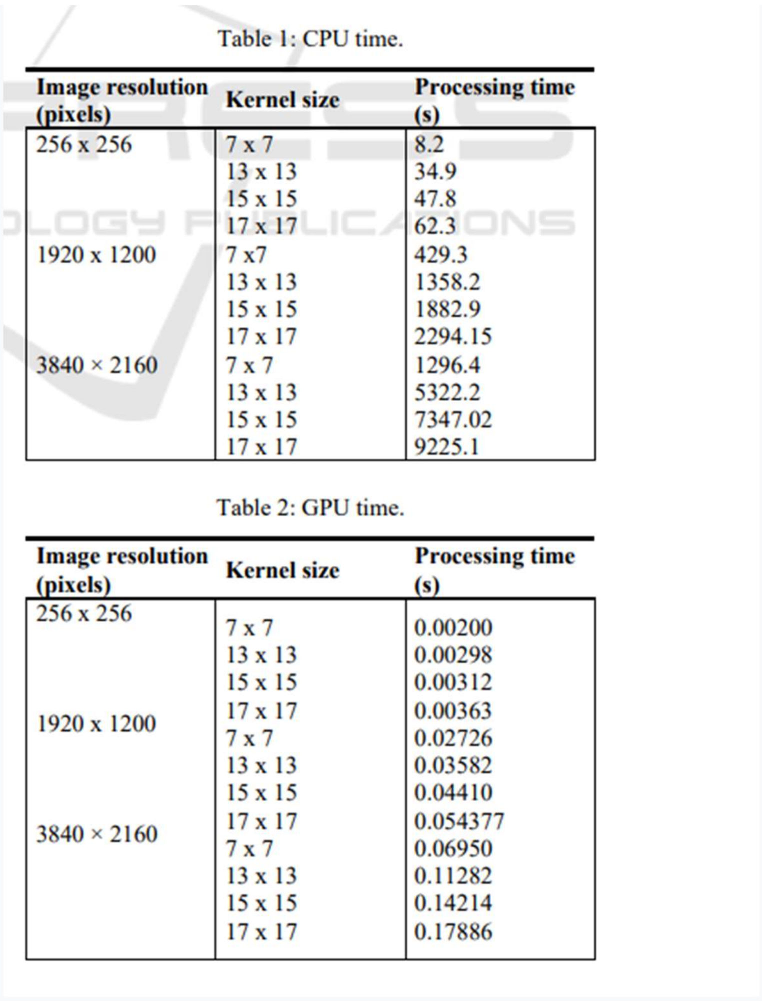


Table 1: CPU time.

Image resolution (pixels)	Kernel size	Processing time (s)
256 x 256	7 x 7	8.2
	13 x 13	34.9
	15 x 15	47.8
	17 x 17	62.3
1920 x 1200	7 x7	429.3
	13 x 13	1358.2
	15 x 15	1882.9
	17 x 17	2294.15
3840 × 2160	7 x 7	1296.4
	13 x 13	5322.2
	15 x 15	7347.02
	17 x 17	9225.1

Table 2: GPU time.

Image resolution (pixels)	Kernel size	Processing time (s)
256 x 256	7 x 7	0.00200
	13 x 13	0.00298
	15 x 15	0.00312
	17 x 17	0.00363
1920 x 1200	7 x 7	0.02726
	13 x 13	0.03582
	15 x 15	0.04410
	17 x 17	0.054377
3840 × 2160	7 x 7	0.06950
	13 x 13	0.11282
	15 x 15	0.14214
	17 x 17	0.17886

al. (2015) by more than 3 times.

IMPROVE 2021 - International Conference on Image Processing and

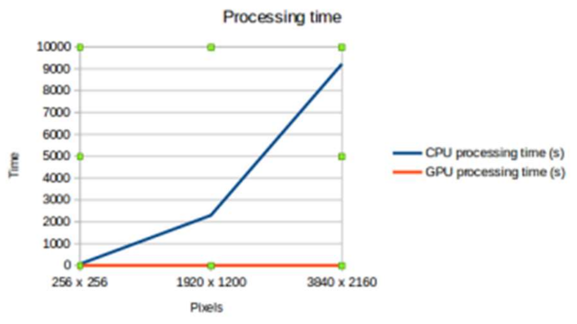


Figure 2: Processing Time.

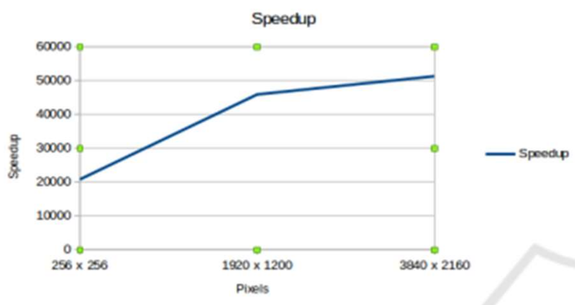


Figure 3: Speedup.

Figure 3: Speedup.

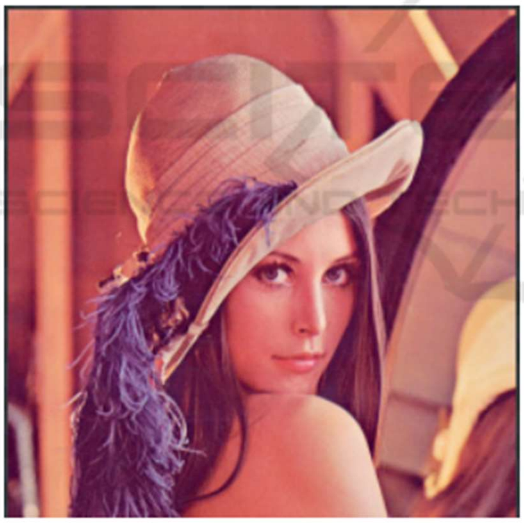


Figure 4: Original Image.

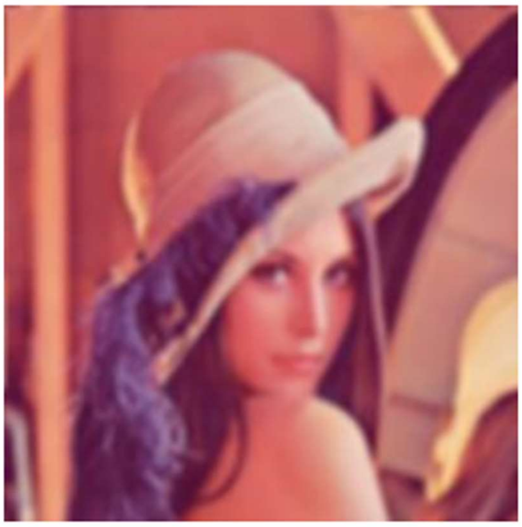


Figure 5: Blurred image using 7 x 7 filter kernel.

Google Colaboratory platform showed great results for using their free access to run our code on the GPUs. **Better performance and speedup were gained using CUDA GPU architecture when compared to multicore CPU**. The graphs in Figures 2 and 3 show that the processing time for the GPU is almost zero compared to the CPU processing time. **Increasing the number of threads for the CPU could give better results but still it will not be able to fill the gap clearly shown between CPU and GPU. GPU is a better candidate** for applications having **complex operations** where the **same operation is applied to a big set of data as convolution.** Image convolution with Gaussian blur filter is better handled with GPU and gives much better performance. As future work, to run the same algorithm on different multicore CPU and GPU. Also to investigate the image transfer time from the CPU to GPU and optimize the transfer time of the image for better performance and speedup.

Summary and Conclusions:

i)Blurring:

CONCLUSIONS The Gaussian blur technique is particularly useful to **filter images with a lot of noise,** results of the filtering showed a **relative independence on the noise characteristics**, and **strong dependence on the variance value of the Gaussian kernel.** In fact, if the **image has a high SNR**, the use of the **technique investigated here could worsen the image**. The **Gaussian blur** is **better used when the original image has a low SNR.** Besides, although filtering images with a large variance in the Gaussian function blurs and worsens the image, it could also be used for generating a mask in order to segment it. The most important feature of this segmentation is the possibility of an automated method for segmentation. Future work could focus on developing the segmentation technique, applying this tool to other medical imaging techniques.

ii)Deblurring

In this review of different research papers has given the different parameters for various techniques for deblurring the image for Gaussian blur. The overall complete review is about the image quality. Many of parameters are used to improve the quality of an image. So the proposed algorithm is about the image quality. Deblurring uses different parameters such as degraded model, restoration techniques, different algorithms and other techniques.

Bibliography

1. Shapiro, L. G. & Stockman, G. C: "Computer Vision", page 137, 150. Prentice Hall, 2001
2. Mark S. Nixon and Alberto S. Aguado. Feature Extraction and Image Processing. Academic Press, 2008, p. 88.
3. Jump up to:a b R.A. Haddad and A.N. Akansu, "A Class of Fast Gaussian Binomial Filters for Speech and Image Processing," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 39, pp 723-727, March 1991.
4. Erik Reinhard. High dynamic range imaging: Acquisition, Display, and Image-Based Lighting. Morgan Kaufmann, 2006, pp. 233–234.
5. Getreuer, Pascal (17 December 2013). "ASurvey of Gaussian Convolution Algorithms". Image Processing on Line. 3: 286–310. doi:10.5201/ipol.2013.87. (code doc)
6. Ritter, Frank (24 October 2013). "Smartphone-Kameras: Warum gute Fotos zu schießen nicht mehr ausreicht [Kommentar]". GIGA (in German). GIGA Television. Retrieved 20 September 2020. Bei Fotos, die in der Nacht entstanden sind, dominiert Pixelmatsch.

[1] Z. Wang and A.C. Bovik, —A universal image quality index‖ IEEE Signal Process Lett 9:81–84, 2002. [2] T. Ferreira and W. Rasband, The ImageJ User Guide — Version 1.44, Feb 20
[3] A.Oppenheim, A.S. Willsky and S.H. Nawab, Signals and Systems,2nd ed., Prentice Hall, 1997 [4] R. C Gonzales and R. E. Woods. Processamento de imagens digitais.São Paulo, Editora Blücher, 2000 Jan Novák et al, GPU Computing: Image Convolution. Karlsruhe Institute of Technology. Shrenik Lad et al, Hybrid Multi-Core Algorithms for Regular Image Filtering Applications. International Institute of Information Technology. Hyderabad, India Ferhat Bozkurt et al, 2015, Effective Gaussian Blurring Process on Graphics Processing Unit with CUDA. International Journal of Machine Learning and Computing, Vol. 5, No. 1. Singapore, Asia. Munesh Singh Chauhan, 2018, Optimizing Gaussian Blur Filter using CUDA Parallel Framework. Information Technology Department, College of Applied Sciences. Ibri, Sulatanate of Oman. B. N. Manjunatha Reddy et al, 2017, Performance Analysis of GPU V/S CPU for Image Processing Applications. International Journal for Research in Applied Science & Engineering Technology (IJRASET). India