Journal of Sensor Research and Technologies
Volume 4 Issue 3

Python OpenCV Image Filtering with Edge Detection

Dr. B.Surekha Reddy
Assistant Professor of ECE, IARE-Hyderabad, Telangana, INDIA,

Corresponding Author
E-Mail Id: b.surekhareddy@iare.ac.in

ABSTRACT

The processing of photographs entails a number of different phases, each of which influences a different component of the image. The processing of photos, often known as the modification of images, can alter the way a photograph looks. The stages involved in digitising a picture and then modifying it in various ways, such as adding effects or gleaning information, are referred to collectively as "image processing," and the phrase "image processing" refers to those steps. When applying a filter to an image, you have the ability to modify its quality in a number of different ways, such as changing its size, shape, colour, depth, or smoothness. technology that allows for the creation of designs as well as the modification of media. Filters are utilised in order to complete the task of reducing noise, which is an essential aspect of the image enhancement process. To achieve the intended outcome, it primarily makes use of a wide range of graphic editing techniques to make adjustments to the individual pixels that compose an image. This article covers a collection of filters that, when used on an image, will result in an improvement in the image's overall quality. By applying a variety of filters, it is possible to obtain photographs that are distinct and free of any background noise.

Keywords: *Image Filtration, Filters, Noise, Python OpenCV Software*

Introduction:

Visuals and animations In many cases, impulse noises are modified at both the recording and transmission stages. Errors in either the transmission of bits or the recording of signals are responsible for the appearance of this impulsive noise in a photograph. Detection errors spike when noise is present, making it difficult to pinpoint the precise location of features like edges and contours.

Therefore, noise cancellation and adjusting pixel brightness are both necessary. When noise and other undesired features are eliminated from an image via image filtering, the resulting version is better. An integral part of the image-preparation process is the "image filtering" step. With no interference, isolating individual signals from their associated channel is a difficult task. It has been suggested that various denoising technologies could be used to remove noise from digital data. Noise in digital signals can be reduced using wavelet denoising with edge computation. Here, wavelets and edge calculations are brought together to form a single effective method.

Noise in images can be amplified or muted, and the saturation or brightness of an image can be adjusted with the use of appropriate filters.

We are attempting to resolve an issue with picture filters (s). We provide a number of filters that, depending on the user's preferences, can either increase or decrease the overall level of image noise, as there are many different kinds of noise that can be introduced into an image. That begs the question, why do we utilise OpenCV? OpenCV is a library made to help with various computer vision issues. OpenCV is compatible with a wide variety of OO languages, including Python, C++, and Java.

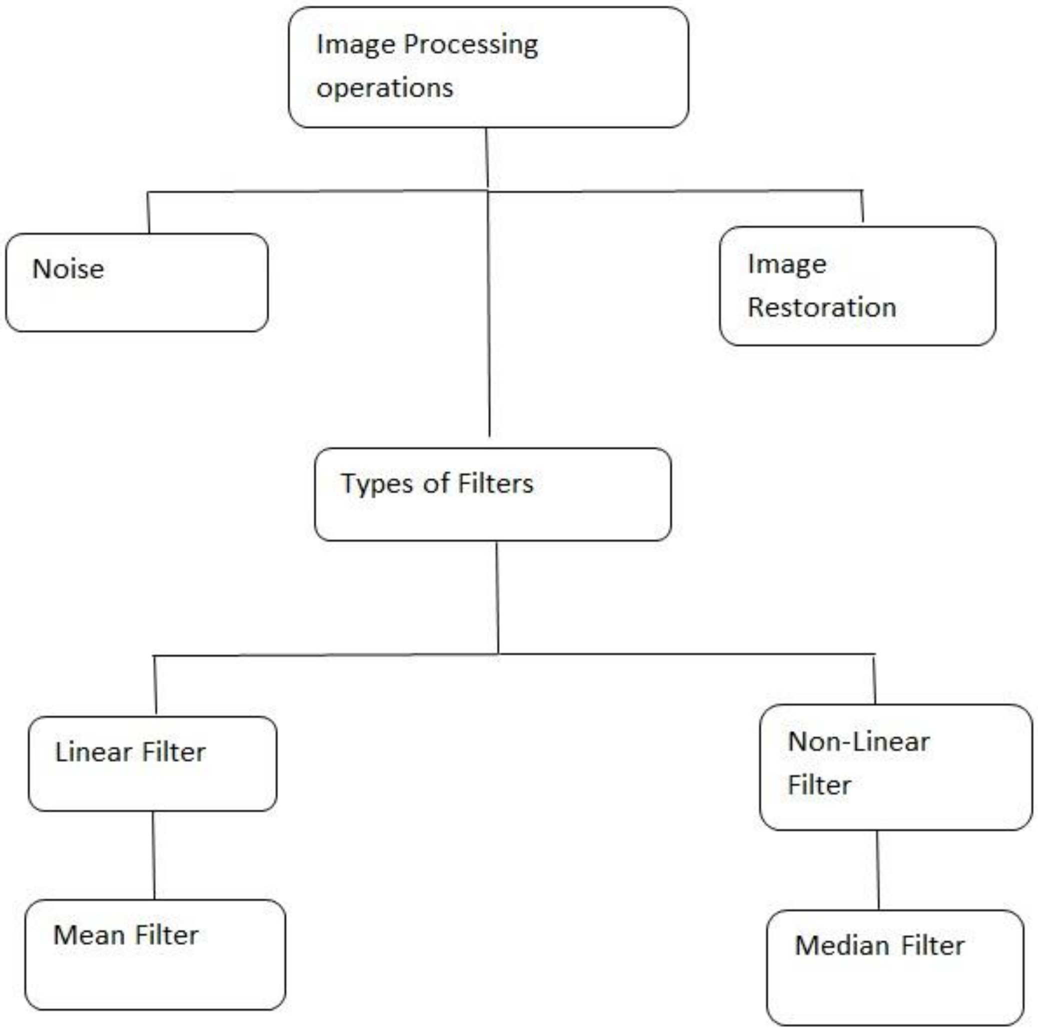
OpenCV is also available on many different operating systems, including iOS, Android, OS X, Linux, and Windows. Quick Graphics Processing Unit function interfaces are also being developed continuously. OpenCL and CUDA form the basis of these interfaces. The OpenCV Python API is commonly referred to by its name. It bridges the gap between the OpenCV C++ API and the Python programming language (API). Plus, with the help of NumPy, the task can be done swiftly and easily.

For advanced mathematical tasks, go no further than NumPy, the state-of-the-art, de facto standard library.

Literature Survey:

SNo	Algorithms	Proposed Method	Advantages	Disadvantages
1)	Linear Algorithm	Python – OpenCV	This section discusses linear filtering in MATLAB and the Image Processing Toolbox	the value of output pixel is linear combinations of the neighboring input pixels.
2)	Non – Linear Algorithm		If the filter outputs signals R and S for two input signals r and s separately, but does not always output αR + βS when the input is a linear combination αr + βs.	A filter whose output is not a linear function of its input.
3)	Python - OpenCV		It is a Library inside the Python. Which is used to execute different type of filters in a simple way	The output of the image may not be much accurate and effective.

Existing Block Diagram:



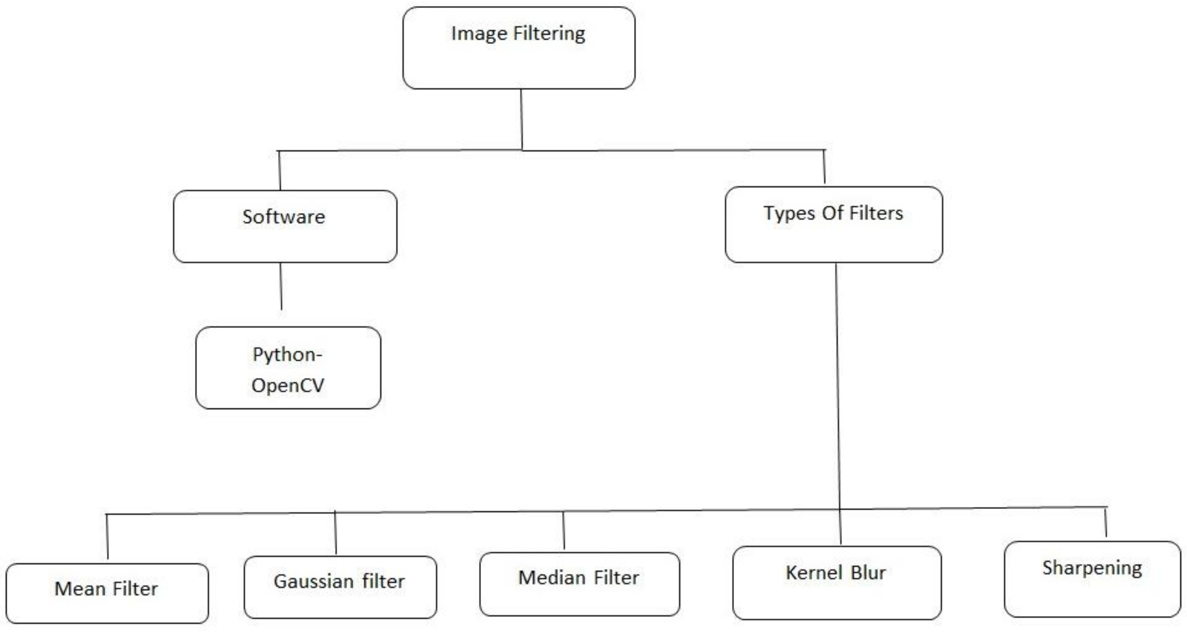
Existing Algorithm:
The Existing algorithm Used is Linear and Non-Linear Algorithm for Filtering Of images.

Problem Identification:
It's likely that the image we get from the camera won't be totally free of noise all the way through. Noise, which might more accurately be described as oscillations that are not audible but cannot be avoided, is an inevitable result of the process of capturing an image.

Images captured by a digital camera are susceptible to being affected by noise if the lenses are not properly aligned with the sensors inside of the camera. Even if it is not immediately apparent, there is always going to be some level of noise present in an image.

Any electronic system will take in the surrounding noise and then transmit that noise to whatever it is that it makes. Images undergo transformation as they travel through channels due to the presence of impulse noise. Before the processing can begin, filters must be applied in order to remove the sounds. We have access to a wide variety of filters that can cut down on noise.

Proposed Block Diagram:

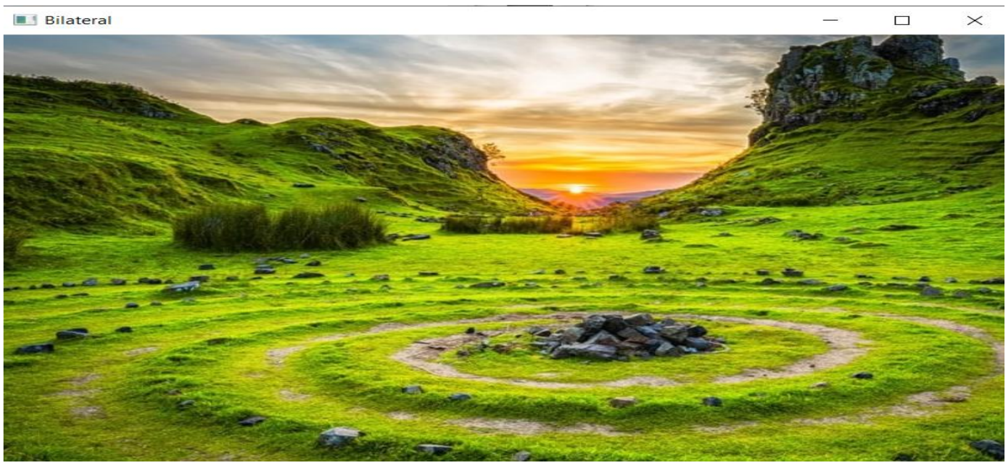


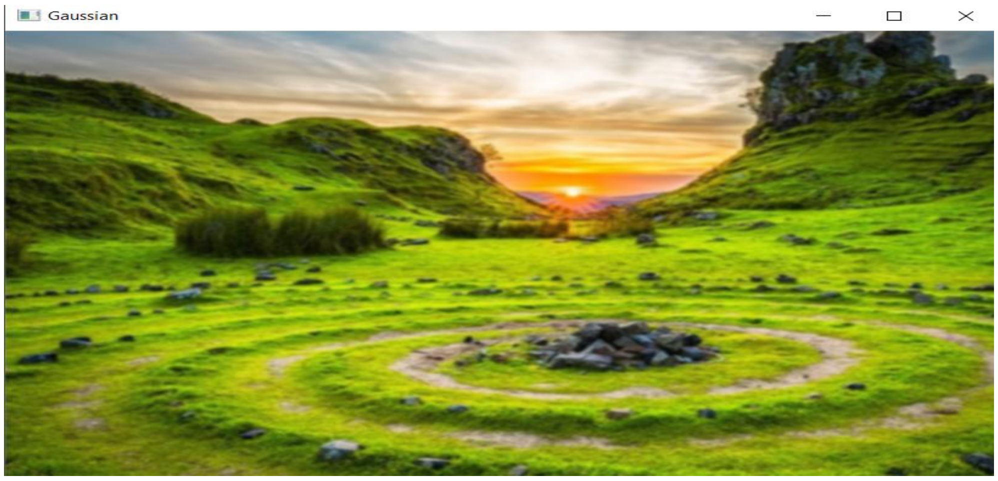
Source Code:

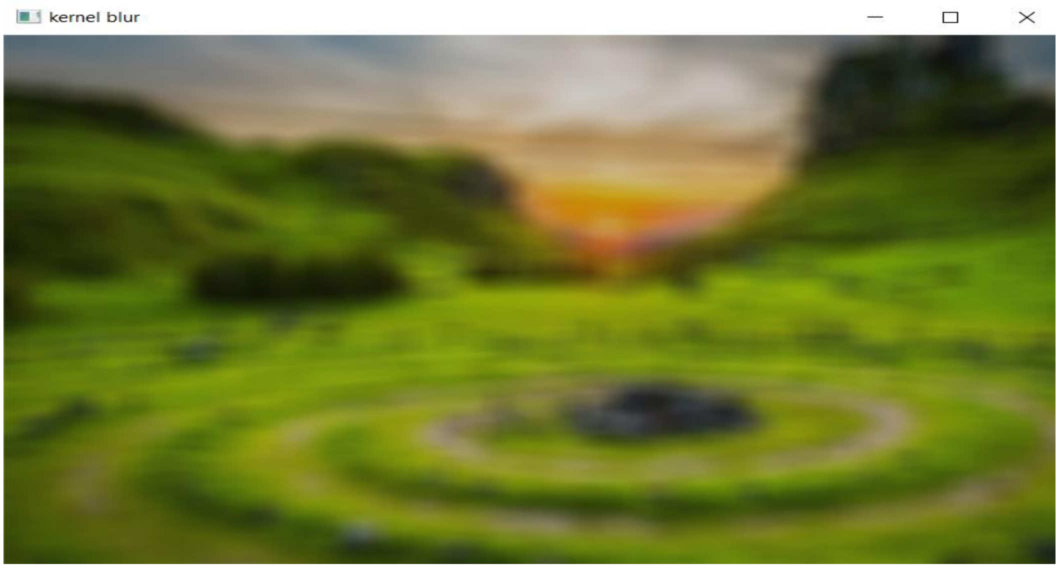
```
#Code For Gaussian Filter
import cv2
import NumPy as np

img = cv2.imread(r"C:\Users\Bhavishya.G\Desktop\nature1.jfif")
rowa, cols = img. shape [:2]


output_gaus = cv2.GaussianBlur(img, (5,5),0)

cv2.imshow('original', img)
cv2.imshow('Gaussian', output_gaus)

cv2.waitkey(0)

#Code For Median Blur
import cv2
import NumPy as np

img = cv2.imread(r"C:\Users\Bhavishya.G\Desktop\nature1.jfif")
rowa, cols = img.shape[:2]
output_med = cv2.medianBlur(img,5)

cv2.imshow('original',img)
cv2.imshow('MedianBlur',output_med)
```

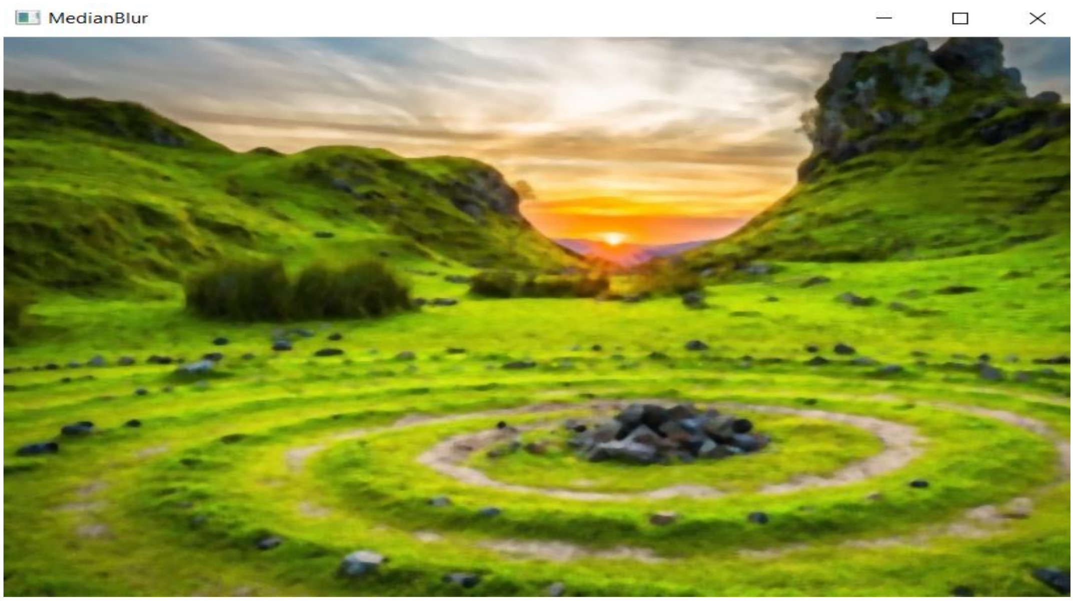
```
cv2.waitkey(0)
#Code For Kernel Blur
import cv2
import NumPy as np

img = cv2.imread(r"C:\Users\Bhavishya.G\Desktop\nature1.jfif")
rowa, cols = img.shape[:2]

kernel_25 = np.ones((20,20), np.float32) / 500.0
output_kernel = cv2.filter2D(img, -1, kernel_25)

cv2.imshow('kernel blur',output_kernel)
cv2.imshow('original',img)
cv2.waitkey(0)
#Code For Bilateral Filter
import cv2
import numpy as np

img = cv2.imread(r"C:\Users\Bhavishya.G\Desktop\nature1.jfif")
rowa, cols = img.shape[:2]


output_bil = cv2.bilateralFilter(img,5,6,6)

cv2.imshow('original',img)
cv2.imshow('Bilateral',output_bil)

cv2.waitkey(0)

#Code for Sharpening
import cv2
import numpy as np

img = cv2.imread(r"C:\Users\Bhavishya.G\Desktop\nature1.jfif")
gaussian_blur = cv2.GaussianBlur(img,(7,7),2)

sharpened1 = cv2.addWeighted(img,1.5, gaussian_blur, -0.5, 0)
sharpened2 = cv2.addWeighted(img,3.5, gaussian_blur, -2.5, 0)
sharpened3 = cv2.addWeighted(img,7.5, gaussian_blur, -6.5, 0)

cv2.imshow('original',img)
cv2.imshow('sharpened3',sharpened3)
cv2.imshow('sharpened2',sharpened2)
cv2.imshow('sharpened1',sharpened1)

cv2.waitkey(0)
```

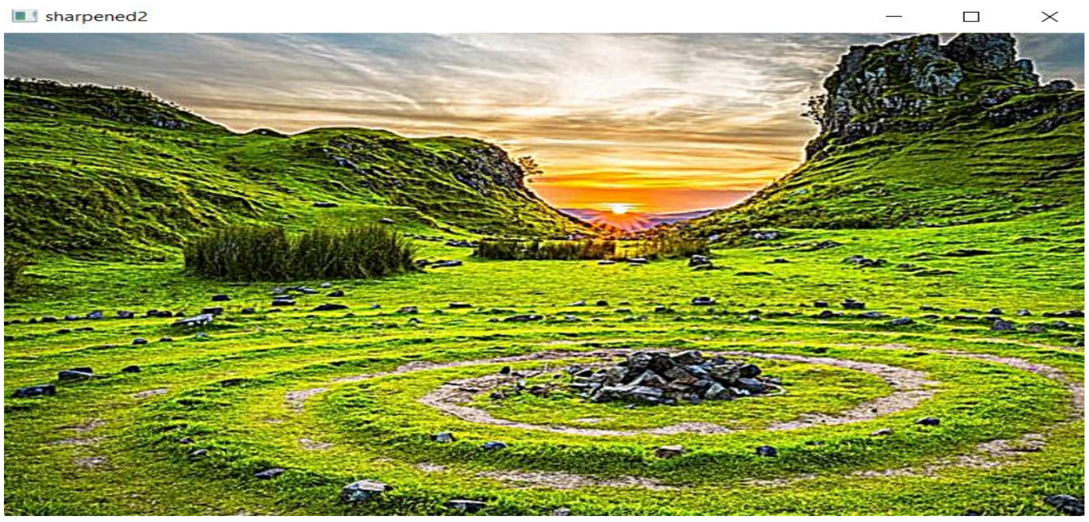
Results and Discussions:











Conclusion and Future Scope:

We provided a wide variety of filtering strategies and noises. There are defining features of each filter that set it apart. The appropriate filter is chosen based on the nature of the noise and the degree of filtering necessary. Because they preserve original edges, filtering processes have widespread acceptance as a processing method. There are a wide variety of filters available, each with its own set of pros and cons. For this reason, it is crucial to have access to many different kinds of filters for fixing image processing problems. For some applications, nonlinear filters might provide better results than their linear counterparts. The nonlinear filter outperforms the linear filter when it comes to restoring images.

References:

1. Face Detection and Recognition using OpenCV, Article.
2. Image filters in python https://towardsdatascience.com/image - filters-in-python26ee938e57d2
3. Facial Recognition using OpenCV
4. Image Enhancement on OpenCV based on the Tools: Python 2.7 February 2
5. Sweety Desal, Shai lender Gupta and Bharat Bhushan”. A Survey of

Various Bilateral Filtering Techniques" International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.8, No.3 (2015).

6. Image Filtering Techniques-A Review Ranjeet Kaur 1, Er. Ram Singh 21 Student, Department of Computer Engineering, Punjabi University Patiala, Punjab, (India) 2 Assistant Professor, Department of Computer Engineering, Punjabi University Patiala, Punjab, (India)
7. **7.**MelikaMostaghim, Elnaz Ghouls and Farshad "Image Smoothing Using Non- Linear Filters A Comparative Study" IEEE 2014
8. D. Z. P. F. L. I. Abdalla Mohamed Hamble, "Image Noise Reduction and Filtering Techniques, "International Journal of Science and Research, 2015
9. Image Enhancement on OpenCV based on the Tools: Python 2.7 V. Ravi1, Ch. Rajendra Prasad2, Sanjay Kumar3, Ramchandra Rao4 Dept. of ECE, S R Engineering College, Warangal, Telangana, India.