Image Processing in Python:

Images define the world, each image has its own story, it contains a lot of crucial information that can be useful in many ways. This information can be obtained with the help of the technique known as **Image Processing**.

It is the core part of computer vision which plays a crucial role in many real-world examples like robotics, self-driving cars, and object detection. Image processing allows us to transform and manipulate thousands of images at a time and extract useful insights from them. It has a wide range of applications in almost every field.

Python is one of the widely used programming languages for this purpose. Its amazing libraries and tools help in achieving the task of image processing very efficiently.

Through this article, you will learn about classical algorithms, techniques, and tools to process the image and get the desired output.

Let’s get into it!

What is image processing?

As the name says, image processing means processing the image and this may include many different techniques until we reach our goal.

The final output can be either in the form of an image or a corresponding feature of that image. This can be used for further analysis and decision making.

But what is an image?

## An image can be represented as a 2D function F(x,y)

## Image processing tools

### **1. OpenCV**

It stands for Open Source Computer Vision Library. This library consists of around 2000+ optimised algorithms that are useful for computer vision and machine learning. There are several ways you can use opencv in image processing, a few are listed below:

* Converting images from one color space to another i.e. like between BGR and HSV, BGR and gray etc.
* Performing thresholding on images, like, simple thresholding, adaptive thresholding etc.
* Smoothing of images, like, applying custom filters to images and blurring of images.
* Performing morphological operations on images.
* Building image pyramids.
* Extracting foreground from images using GrabCut algorithm.
* Image segmentation using watershed algorithm.

where x and y are spatial coordinates. The amplitude of F at a particular value of x,y is known as the intensity of an image at that point. If x,y, and the amplitude value is finite then we call it a digital image. It is an array of pixels arranged in columns and rows. Pixels are the elements of an image that contain information about intensity and colour. An image can also be represented in 3D where x,y, and z become spatial coordinates. Pixels are arranged in the form of a matrix. This is known as an **RGB image**.

### **2. NumPy**

With this library you can also perform simple image techniques, such as flipping images, extracting features, and analyzing them.

Images can be represented by numpy multi-dimensional arrays and so their type is **NdArrays**. A color image is a numpy array with 3 dimensions. By slicing the multi-dimensional array the RGB channels can be separated.

Below are some of the operations that can be performed using NumPy on the image (image is loaded in a variable named **test\_img** using imread).

* To flip the image in a vertical direction, use **np.flipud(test\_img).**
* To flip the image in a horizontal direction, use **np.fliplr(test\_img).**
* To reverse the image, use **test\_img[::-1]**  (the image after storing it as the numpy array is named as <img\_name>).
* To add filter to the image you can do this:

Example: **np.where(test\_img > 150, 255, 0)**, this says that in this picture if you find anything with 150, then replace it with 255, else 0.

* You can also display the RGB channels separately. It can be done using this code snippet:

To obtain a red channel, do **test\_img[:,:,0]**, to obtain a green channel, do **test\_img[:,:,1]** and to obtain a blue channel, do **test\_img[:,:,2].**

### **Gaussian Image Processing**

Gaussian blur which is also known as gaussian smoothing, is the result of blurring an **image** by a **Gaussian** function.

It is **used to reduce image noise and reduce details**. The visual effect of this blurring technique is similar to looking at an image through the translucent screen. It is sometimes used in computer vision for image enhancement at different scales or as a data augmentation technique in deep learning.

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. Let’s see an example to understand what gaussian filters do to an image.

If we have a filter which is normally distributed, and when its applied to an image, the results look like this:

### **Edge Detection in image processing**

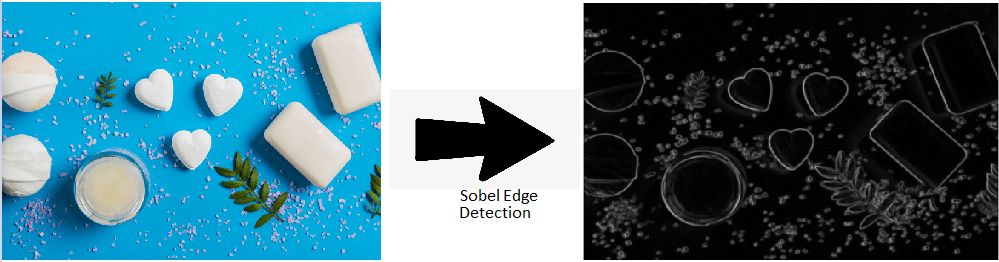
Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness.

This could be very beneficial in extracting useful information from the image because most of the shape information is enclosed in the edges. Classic edge detection methods work by detecting discontinuities in the brightness.

It can rapidly react if some noise is detected in the image while detecting the variations of grey levels. Edges are defined as the local maxima of the gradient.

The most common edge detection algorithm is **sobel edge detection algorithm**. Sobel detection operator is made up of 3\*3 convolutional kernels. A simple kernel Gx and a 90 degree rotated kernel Gy. Separate measurements are made by applying both the kernel separately to the image.

 \* denotes the 2D signal processing convolution operation.

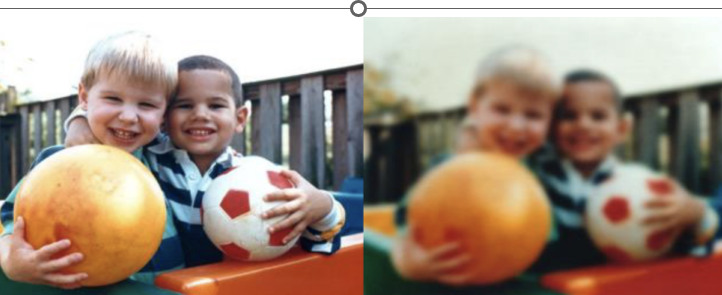


**BLURRING AN IMAGE:**

The first two parameters to skimage.filters.gaussian() are the image to blur, image, and a tuple defining the sigma to use in ry- and cx-direction, (sigma, sigma). The third parameter truncate gives the radius of the kernel in terms of sigmas. A Gaussian function is defined from -infinity to +infinity, but our kernel (which must have a finite, smaller size) can only approximate the real function. Therefore, we must choose a certain distance from the centre of the function where we stop this approximation, and set the final size of our kernel. In the above example, we set truncate to 3.5, which means the kernel size will be 2 \* sigma \* 3.5. For example, for a sigma of 1.0 the resulting kernel size would be 7, while for a sigma of 2.0 the kernel size would be 14. The default value for truncate in scikit-image is 4.0.

The last parameter to skimage.filters.gaussian() tells skimage to interpret our image, that has three dimensions, as a multichannel colour image.

Finally, we display the blurred image:



Sharpening enhances the definition of edges in an image. Whether your images come from a digital camera or a scanner, most images can benefit from sharpening. When sharpening images, keep the following in mind:

* Sharpening cannot correct a severely blurred image.
* Sharpen your image on a separate layer so that you can resharpen it later if you need to change the adjustment. Set the layer’s blending mode to Luminosity to avoid color shifts along edges. If you find that highlights or shadows are lessened after you sharpen, use the layer blending controls to prevent sharpening in highlights and shadows.
* If you need to reduce image noise, do so before sharpening so that you don’t intensify the noise.
* Sharpen your image multiple times in small amounts. Sharpen the first time to correct blur captured by a scanner or digital camera. After you’ve color corrected and resized your image, sharpen it again.
* If possible, judge your sharpening by outputting your image. The amount of sharpening needed varies depending on whether the image is printed or displayed on a web page.



You have seen a few of the features of a good introductory image processing program. There are many more complex modifications you can make to the images. For example, you can apply a variety of filters to the image. The filters use mathematical algorithms to modify the image. Some filters are easy to use, while others require a great deal of technical knowledge. The software also will calculate the ra, dec, and magnitude of all objects in the field if you have a star catalog such as the Hubble Guide Star Catalog (although this feature requires the purchase of an additional CD-ROM). The standard tricolor images produced by the SDSS are very good images. If you are looking for something specific, you can frequently make a picture that brings out other details. The "best" picture is a very relative term. A picture that is processed to show faint asteroids may be useless to study the bright core of a galaxy in the same field. Research Challenges