

TP5 This lesson focus on filters and convolution on images using OpenCV.

1. Filter Images and Videos

The image filtering is a neighbourhood operation in which the value of any given pixel in the output image is determined by applying a certain algorithm to the pixel values in the vicinity of the corresponding input pixel. This technique is commonly used to smooth, sharpen and detect edges of images and videos.

Let's start learning the meaning of some terms used when discussing image filtering techniques: *kernel* and *convolution*.

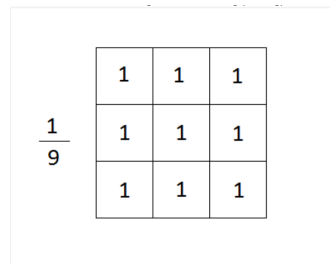
Kernel

Kernel is a **matrix** with an **odd height** and an **odd width**. It is also known as convolution matrix, mask or filter. Kernels are named based on their value distribution. Kernel is used to define a neighbourhood of a pixel in a image filtering operation.

Some of the popular kernels are Normalised box filter, Gaussian kernel, Laplacian kernel, edge detecting kernels etc. Kernels can be defined with different sizes. Larger kernels result in bigger processing time.

In next picture (1) there is a 3 x 3 *Normalised Box filter*. This kernel is used in homogeneous smoothing (blurring).

Figure 1: 3 x 3 *Normalised Box filter*.



$\frac{1}{9}$	1	1	1
	1	1	1
	1	1	1

In next picture a 5 x 5 Normalized Box filter. This kernel is also used in the homogeneous smoothing. In the same way, you can create a Normalized Box filter with any size to be used in the homogeneous smoothing.

Figure 2: 5 x 5 *Normalised Box filter*.

$$\frac{1}{25}$$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

In here a 3 x 3 Gaussian kernel used in Gaussian smoothing (blurring):

Figure 3: 3 x 3 *Gaussian kernel*.

$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

In the same way, you may create a Gaussian kernel with any size. The value distribution of the kernel should be calculated using the 2-D Gaussian function:

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

2-D Gaussian Function

in which x and y are the coordinates of the kernel where the origin is placed at the center (i.e. $x = 0$ and $y = 0$ at the center element of the kernel.); σ is the standard deviation of the Gaussian distribution. For larger standard deviations, larger kernels are required in order to accurately perform the Gaussian smoothing. For example, next picture represent a 5 x 5 Gaussian kernel in which the standard deviation is 1.

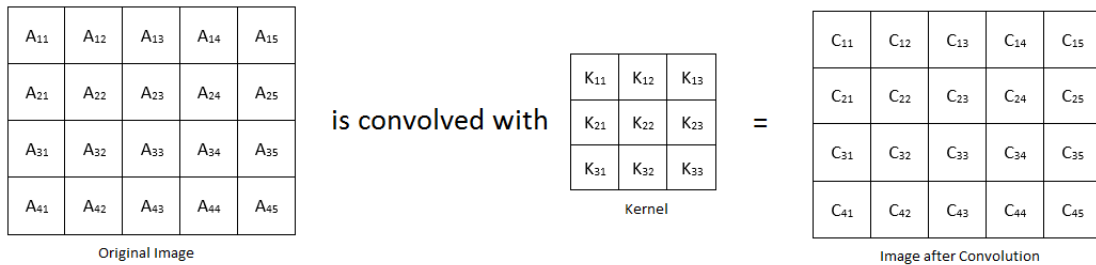
Figure 4: 5 x 5 *Gaussian kernel*.

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Convolution

Convolution is a mathematical operation performed on images by sliding a kernel across the whole image and calculating new values for each pixels based on the value of the kernel and the value of overlapping pixels of original image.



$$\begin{aligned}
 C_{22} &= (K_{11} \times A_{11} + K_{12} \times A_{12} + K_{13} \times A_{13}) + (K_{21} \times A_{21} + K_{22} \times A_{22} + K_{23} \times A_{23}) + (K_{31} \times A_{31} + K_{32} \times A_{32} + K_{33} \times A_{33}) \\
 C_{23} &= (K_{11} \times A_{12} + K_{12} \times A_{13} + K_{13} \times A_{14}) + (K_{21} \times A_{22} + K_{22} \times A_{23} + K_{23} \times A_{24}) + (K_{31} \times A_{32} + K_{32} \times A_{33} + K_{33} \times A_{34}) \\
 C_{24} &= (K_{11} \times A_{13} + K_{12} \times A_{14} + K_{13} \times A_{15}) + (K_{21} \times A_{23} + K_{22} \times A_{24} + K_{23} \times A_{25}) + (K_{31} \times A_{33} + K_{32} \times A_{34} + K_{33} \times A_{35}) \\
 \\
 C_{32} &= (K_{11} \times A_{21} + K_{12} \times A_{22} + K_{13} \times A_{23}) + (K_{21} \times A_{31} + K_{22} \times A_{32} + K_{23} \times A_{33}) + (K_{31} \times A_{41} + K_{32} \times A_{42} + K_{33} \times A_{43}) \\
 C_{33} &= (K_{11} \times A_{22} + K_{12} \times A_{23} + K_{13} \times A_{24}) + (K_{21} \times A_{32} + K_{22} \times A_{33} + K_{23} \times A_{34}) + (K_{31} \times A_{42} + K_{32} \times A_{43} + K_{33} \times A_{44})
 \end{aligned}$$

In the same way C_{34} , can be calculated in the convolved image. But other values in the boundary of the convolved image (e.g C_{11} , C_{12} , etc) cannot be calculated in the same way because the kernel is partially overlapped with the original image at the boundary. Therefore non-overlapped non-existing pixel values should be calculated in order to determine the pixel values at the boundaries of the convolved image. There are various techniques to handle this boundary values but they will not be discussed in here.

Smooth / Blur Images

When an image is acquired by a camera or other method, the image may be corrupted by random dots and noises. Smoothing/blurring is one of image processing techniques to eliminate such imperfections

and improve the image. Image smoothing techniques can be applied even for each frames of a video to eliminate imperfections and improve the video. We will focus on **Homogeneous Blur** and **Gaussian Blur**.

Homogeneous Blur

Homogeneous Blur is the most simplest method of smoothing an image. It is also called as Homogeneous Smoothing, Homogeneous Filtering and **Box Blurring**. In this technique, **each pixel value is calculated as the average value of the neighbourhood of the pixel defined by the kernel**.

As we have seen before, kernels used in the homogeneous blur is called normalized box filter. You may define any size for this kernel according to your requirement. But it is preferable to define square kernels with a size of odd width and height. In 1 and 2 images, we have seen 3 x 3 and 5 x 5 normalized box filters.

You have to choose a right size of the kernel to define the neighborhood of each pixel. If it is too large, small features of the image may be disappeared and the image will look blurred. If it is too small, you cannot eliminate noises in the image.

The simple blur operation is provided in OpenCV by `void cv::blur (InputArray src, OutputArray dst, Size ksize, Point anchor = Point(-1,-1), int borderType = BORDER_DEFAULT)`. Each pixel in the output is the simple mean of all of the pixels in the kernel, around the corresponding pixel in the input. The size of the kernel is specified by the argument `ksize`. The argument `anchor` can be used to specify how the kernel is aligned with the pixel being computed. By default, the value of `anchor` is `cv::Point(-1,-1)`, which indicates that the kernel should be centered relative to the filter. In the case of multichannel images, each channel will be computed separately. Here are the blur function parameters in details:

- `src` is the input image; it can have any number of channels, which are processed independently, but the depth should be `CV_8U`, `CV_16U`, `CV_16S`, `CV_32F` or `CV_64F`;
- `dst` is the output image of the same size and type as `src`;
- `ksize` is the blurring kernel size;
- `anchor` anchor point by default the value is `Point(-1,-1)` means that the anchor is at the kernel center;
- `borderType` is the border mode used to extrapolate pixels outside of the image.

Complete the code 'students_blur.cpp' using both 3 x 3 and 5 x 5 normalized box filter on the 'van_gogh.jpg' image.

Homogeneous Blur on Videos

Blur/smooth a video is pretty much similar to blur an image as we have seen in the previous program.

Complete the code 'students_video_blur.cpp' using both 3 x 3 and 5 x 5 normalized box filter on the 'chaplin.mp4' video.

Gaussian Blur

Gaussian blur/smoothing is the most commonly used smoothing technique to eliminate noises in images and videos. In this technique, an image should be convolved with a Gaussian kernel to produce the smoothed image.

You may define the size of the kernel according to your requirement. But the standard deviation of the Gaussian distribution in X and Y direction should be chosen carefully considering the size of the kernel such that the edges of the kernel is close to zero. We will use the 3 x 3 and 5 x 5 Gaussian kernels we presented previously (namely figure 3 and 4). It's important to choose a right size of the kernel to define the neighborhood of each pixel. If it is too large, small features of the image may be disappeared and the image will look blurred. If it is too small, you cannot eliminate noises in the image.

The Gaussian blur operation is provided in OpenCV by `void cv::GaussianBlur (InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY = 0, int borderType = BORDER_DEFAULT)` in which :

- src is the input image; the image can have any number of channels, which are processed independently, but the depth should be CV_8U, CV_16U, CV_16S, CV_32F or CV_64F;
- dst is the output image of the same size and type as src;
- ksize is the Gaussian kernel size. ksize.width and ksize.height can differ but they both must be positive and odd. Or, they can be zero's and then they are computed from sigma.
- sigmaX Gaussian kernel standard deviation in X direction.
- sigmaY Gaussian kernel standard deviation in Y direction; if sigmaY is zero, it is set to be equal to sigmaX, if both sigmas are zeros, they are computed from ksize.width and ksize.height, respectively; to fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of ksize, sigmaX, and sigmaY;
- borderType is the border mode used to extrapolate pixels outside of the image.

3 x 3 and 5 x 5, and 7 x 7 kernels which have the “standard” sigma (i.e., sigmaX = 0.0) give really better performance than other kernels (and they are the ones we will use in our program). Gaussian blur supports single- or three-channel images in either 8-bit or 32-bit floating-point formats, and it can be done in place.

Complete the code 'students_gaussian.cpp' using both 3 x 3 and 5 x 5 normalized box filter on the 'van_gogh.jpg' image.

Gaussian Blur on Videos

Complete the code 'students_gaussian_video.cpp' using both 3 x 3 and 5 x 5 normalized box filter on the 'chaplin.mp4' video.

To sum up

There are many reasons for smoothing, but it is often done to reduce noise or camera artefacts. Smoothing is also important when we wish to reduce the resolution of an image in a principled way.

We have studied 2 of the 5 smoothing operations that OpenCV offers. Each of them has its own associated library function, which each accomplish slightly different kinds of smoothing. The src and dst arguments in all of these functions are the usual source and destination arrays. After that, each smoothing operation has parameters that are specific to the associated operation. Of these, the only common parameter is the last, borderType. This argument tells the smoothing operation how to handle pixels at the edge of the image.

There are other kinds of operation on image that concern filters and convolution (Derivatives and Gradients, Morphology). You will discover them during the project implementation.