# Image Filtering

Image filtering allows the user to apply various dynamic effects on a digital image. For image filtering two dimensional filters are used. Two dimensional filters are defined as matrices of a certain size. The process of image filtering is that for every pixel of an image, take the sum of the products. Each product is obtained by multiplying the color value of the current pixel or a neighbor of it, with the corresponding value of the filter matrix. The center of the filter matrix has to be multiplied with the current pixel and the other elements of the filter matrix with corresponding neighbor pixels. This operation is called two dimensional convolution.

In MatDeck, there is a function called image filter() which performs two dimensional filtering as explained above. The function has two arguments, image object to be filtered and two dimensional filter. Two dimensional filtering operation is computationally very demanding. These filters are quadratic small matrices of an uneven size, in such a manner that they have a center. The function image filter() accepts two dimensional filters of a size 3*3, which is the most common and in most cases of satisfactory performance.

Besides that the used two dimensional filter should have the sum of all the elements equal to one. If this is not the case the resulting image will not have the same brightness. Furthermore, if some of the elements are larger than one, the resulting image will be brighter, otherwise it will be darker. Here are several examples of image filtering. We start with the original image of Tulips.

```
1 img := image read("tulips.png")
2 pic := image widget(0, img)
3 set size(pic, 300, 200)
```

### Blur

Blurring is done by taking the average of the current pixel and its neighbors. We can perform blurring by filtering with the next filter:

```
4 blur := [0, 1, 0; 1, 1, 1; 0, 1, 0]
```

Gaussian blur filter is defined as:

```
5 gblur := [1, 2, 1; 2, 4, 2; 1, 2, 1]
```

Motion blur can be achieved using the following box filter:

```
6 mblur := [0, 0, 1; 0, 0, 0; 1, 0, 0]
```

We can compare the results of these three filters.

```
7
```

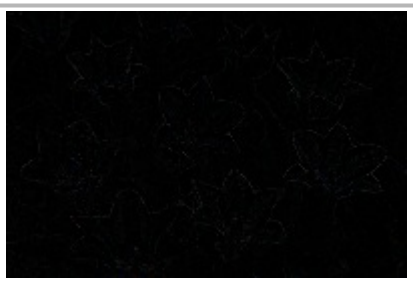
```
imgb := image filter(img, blur)
picb := image widget(0, imgb)
set size(picb, 200, 133)
imgg := image filter(img, gblur)
picg := image widget(0, imgg)
set size(picg, 200, 133)
imgm := image filter(img, mblur)
picm := image widget(0, imgm)
set size(picm, 200, 133)
```

$$\text{blur} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \text{gblur} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \text{mblur} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

## Difference

From the images above, it is not possible to conclude the effect regarding the blur until careful inspection. Instead, we can use the function image absdiff() to find the absolute difference between a processed image and to visualize it. We can calculate the mean square error (MSE) and peak signal to noise ratio (PSNR) using image mse psnr() to obtain objective measure.

```
imdiff := image absdiff(img, imgb)
picdiff := image widget(0, imdiff)
set size(picdiff, 200, 133)
imdiff1 := image absdiff(img, imgg)
picdiff1 := image widget(0, imdiff1)
set size(picdiff1, 200, 133)
imdiff2 := image absdiff(img, imgm)
picdiff2 := image widget(0, imdiff2)
set size(picdiff2, 200, 133)
MSE := image mse psnr(img, imgb)
MSE1 := image mse psnr(img, imgg)
MSE2 := image mse psnr(img, imgm)
```



MSE[0] = 72.032

MSE[1] = 29.556 dB

MSE1[0] = 75.614

MSE1[1] = 29.345 dB

MSE2[0] = 149.940

MSE2[1] = 26.372 dB

## Other filtering options

Using image filter(), we can define filters for other effects. A filter to find the edges can look like this:

```
29 edge := [ -1, -1, -1; -1, 8, -1; -1, -1, -1]
```

Sharpening the image is very similar to finding edges, and the result will be a new image where the edges are enhanced, making it to look sharper. A sharpening filter can be defined as:

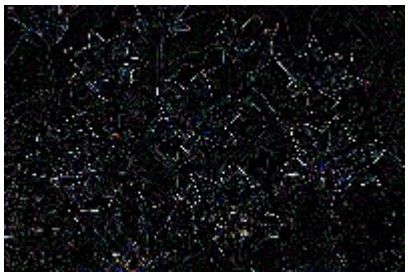
```
30 sharp := [ 1, 1, 1; 1, -7, 1; 1, 1, 1]
```

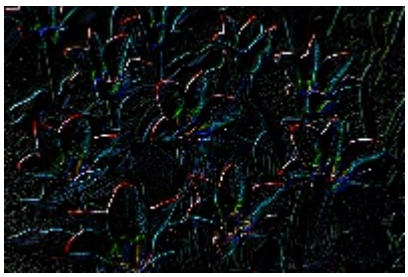
An emboss filter gives a 3D shadow effect to the image. It can be achieved by taking a pixel on one side of the center, and subtracting one of the other side from it:

```
31 emboss := [ -1, -1, 0; -1, 0, 1; 0, 1, 1]
```

We illustrate the obtained images as so.

```
32 im1 := image filter(img, edge)
33 pc1 := image widget(0, im1)
34 set size(pc1, 200, 133)
35 im2 := image filter(img, sharp)
36 pc2 := image widget(0, im2)
37 set size(pc2, 200, 133)
38 im3 := image filter(img, emboss)
39 pc3 := image widget(0, im3)
40 set size(pc3, 200, 133)
```





## Conclusions

This example contains codes which can be used to apply to convolution filters on images, and showed several different filters and their results. These are only the very rudimentary examples of image filtering, in MatDeck there are several built in filters for further image processing to enhance your resulting image.