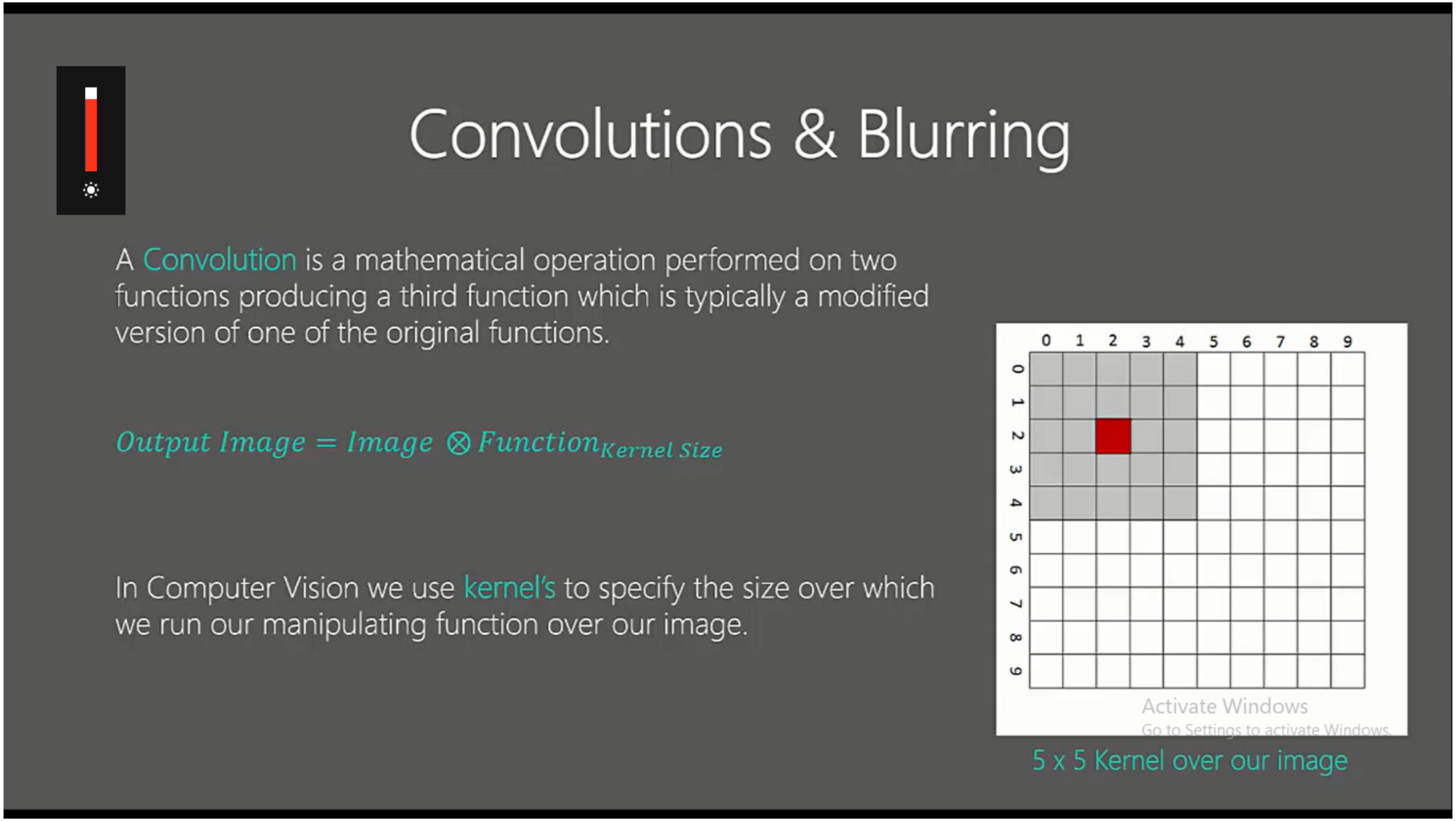
In [5]:
```
from IPython.display import Image
import numpy as np, cv2
```

In [3]:
```
Image('tutorial images/convolution and blurring.png')
```

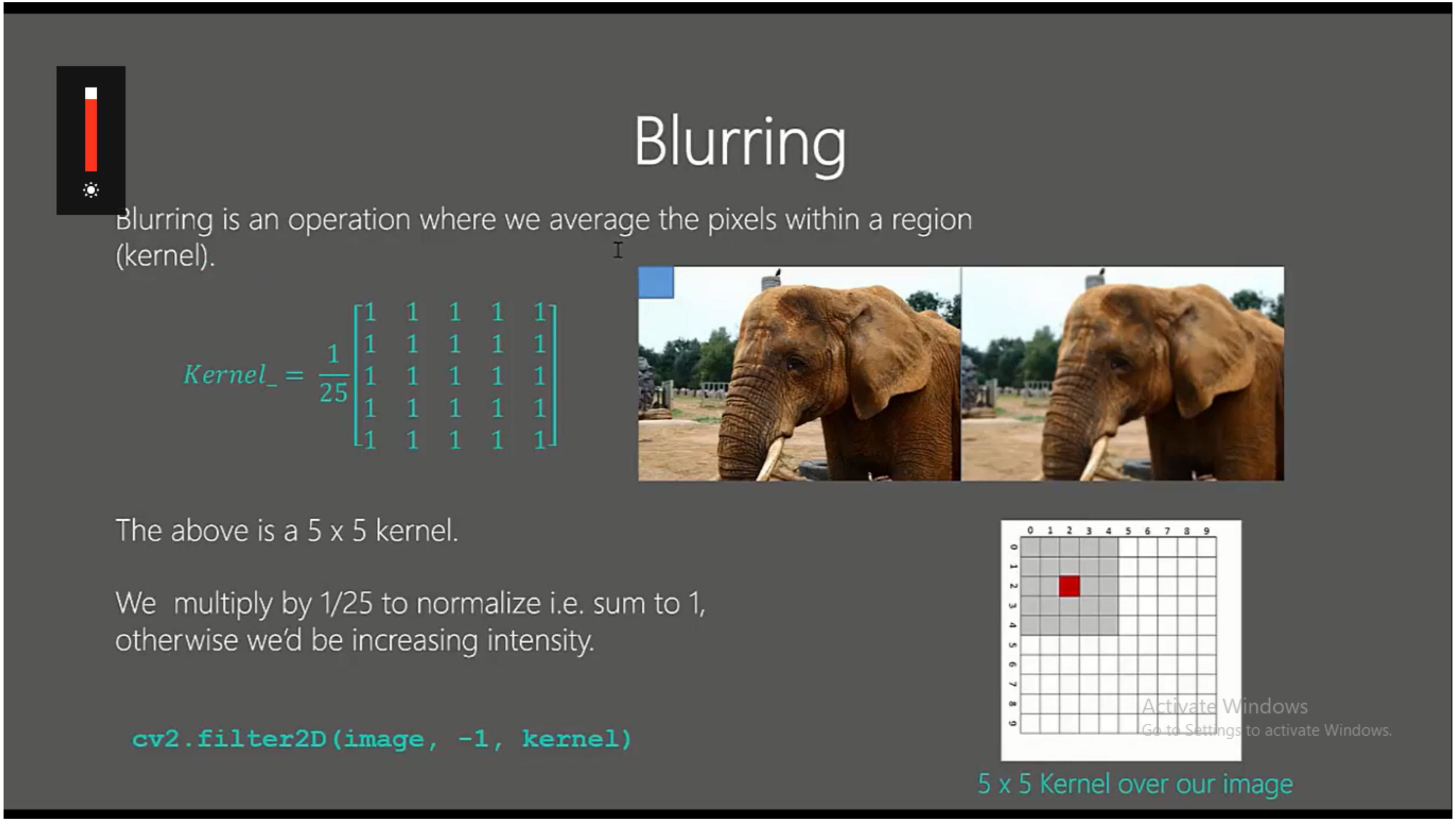
Out[3]:



convolution operates one pixel at a time

In [4]:
```
Image('tutorial images/Blurring.png')
```

Out[4]:



Convolution and Blurring

In [1]:
```python
try:
    import numpy as np, cv2

    image = cv2.imread('my.JPG')
    cv2.imshow('Original Image', image)
    cv2.waitKey()

    #Creating our 3 x 3 kernel
    kernel_3x3 = np.ones((3, 3), np.float32) / 9

    #We use the cv2.filter2D to convolve the kenel with an image
    blurred = cv2.filter2D(image, -1, kernel_3x3)

    cv2.imshow('3x3 Kenel Blurred', blurred)

    cv2.waitKey()

    #creating our 7 x 7 kernel
    kernel_7x7 = np.ones((7, 7), np.float32) / 49

    blurred2 = cv2.filter2D(image, -1, kernel_7x7)
    cv2.imshow('7 x 7 kernel Blurring', blurred2)
    cv2.waitKey()
    cv2.destroyAllWindows()
except Exception as e:
    print(f'{type(e).__name__}:-- {e}')
```

Other commonly used blurring methods in OpenCV

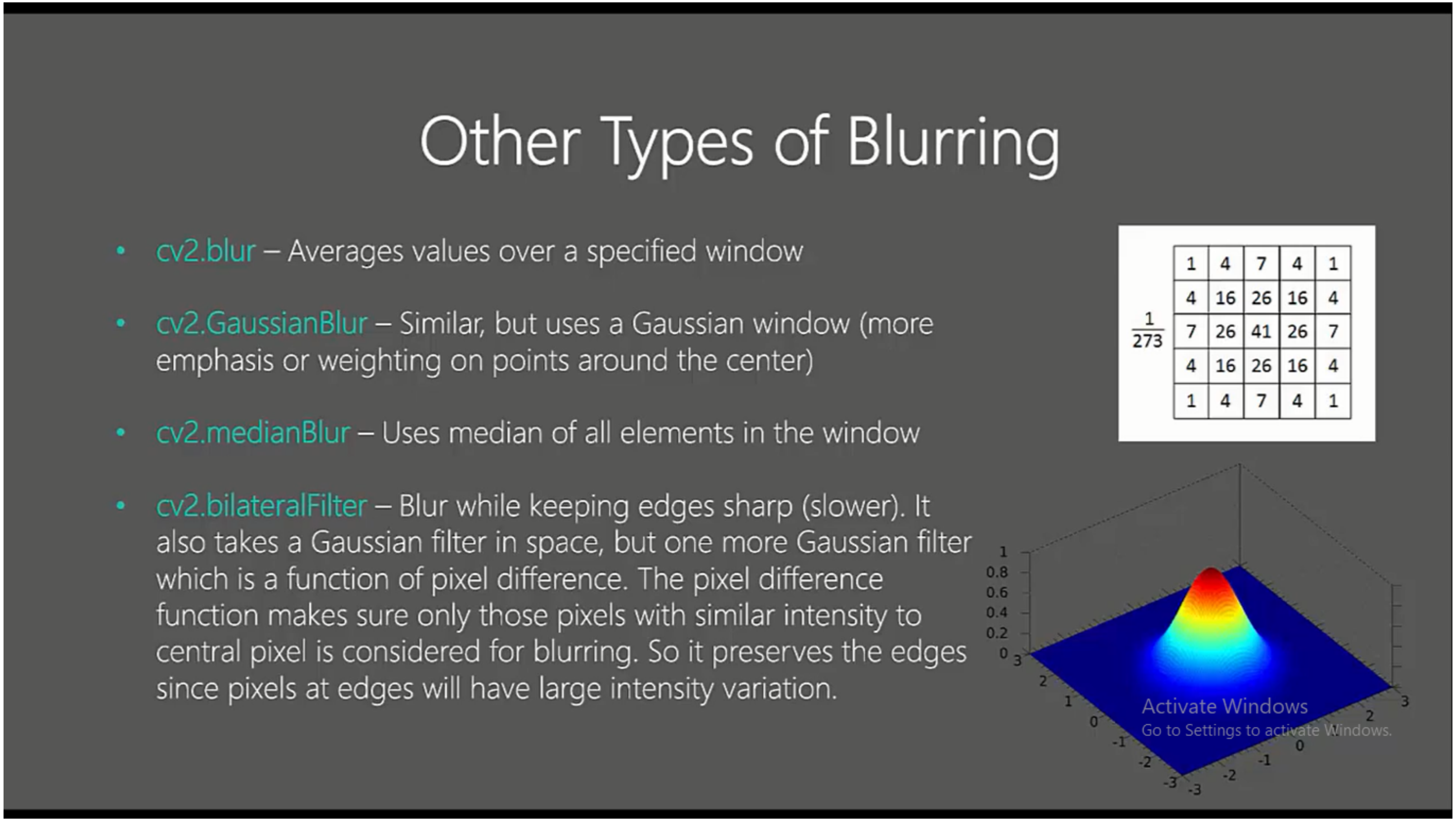
In [3]:

```
import cv2, numpy as np
image = cv2.imread('my.JPG')

#Averaging done by convolving the image with a normalized box filter.
#This takes the pixels under the box and replaces the central element
#Box size needs to odd and positive

blur = cv2.blur(image, (3, 3))
cv2.imshow('Averaging', blur)
cv2.waitKey()

#Instead of box filter, gaussian kernel
Gaussian = cv2.GaussianBlur(image, (7, 7), 0)
cv2.imshow('Gaussian Blurring', Gaussian)
cv2.waitKey()

#Takes median of all pixels under kernel area and central
#element is replaced with this median value
median = cv2.medianBlur(image, 5)
cv2.imshow('Median Blur', median)
cv2.waitKey()

#Bilateral is very effective in noise removal while keeping edges sharp
bilateral = cv2.bilateralFilter(image, 9, 75, 75)
cv2.imshow('Bilateral Blurring', bilateral)
cv2.waitKey()
cv2.destroyAllWindows()
```

Other commonly used blurring methods used in opencv

In [6]:
```
Image('tutorial images/other types of blurring.png')
```

Out[6]:



^^^^^^^This matrix and graph is for GaussianBlur^^^^^^^^

this is how gaussianBlur works it mainly focuses at center more

Image De-nosing - Non-Local Means Denoising

In [1]:

```
import numpy as np, cv2

image = cv2.imread('my.JPG')

cv2.imshow('Original image', image)
cv2.waitKey()

#parameters, after None are - the filter strength 'h' (5-10 is a good range)
#Next is h for ColorComponents, set as same value as h again
#
dst = cv2.fastNlMeansDenoisingColored(image, None, 6, 6, 7, 21)

cv2.imshow('Fast Means Denoising', dst)

cv2.waitKey()
cv2.destroyAllWindows()
```