CVIT Workshop Session 2

•••

Image Processing Continued

- Vansh Garg and Akshat Sanghvi

Dilation and Erosion

- Dilation: At each pixel, take the maximum value among neighbouring pixels. This results in adding a layer of 8 white pixels around each white pixel. This is as white pixels have value 255, and black have a value of 0.
- Erosion: Take the minimum of neighbouring pixels, so this results in removing one outer layer of white pixels, essentially

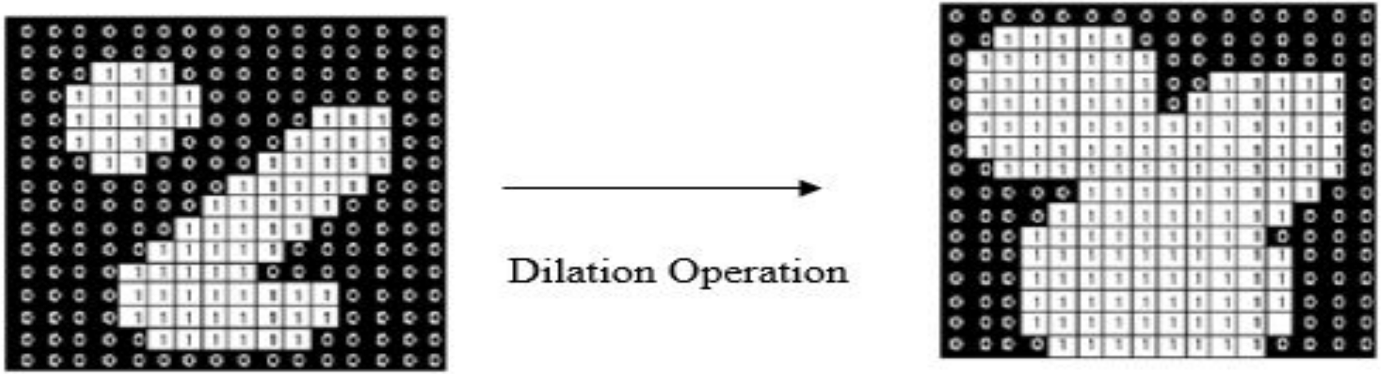
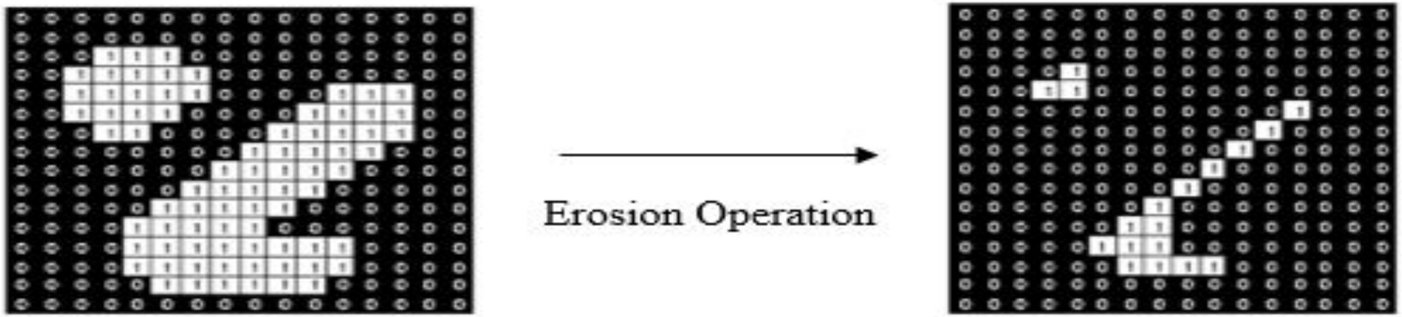
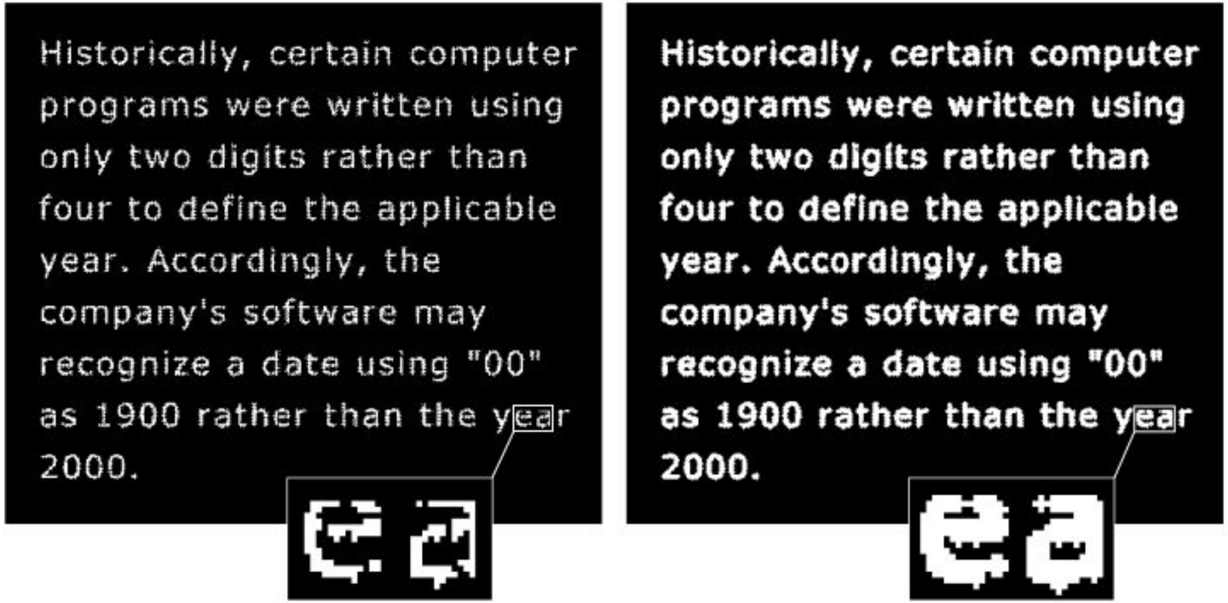


Image Pre-processing techniques

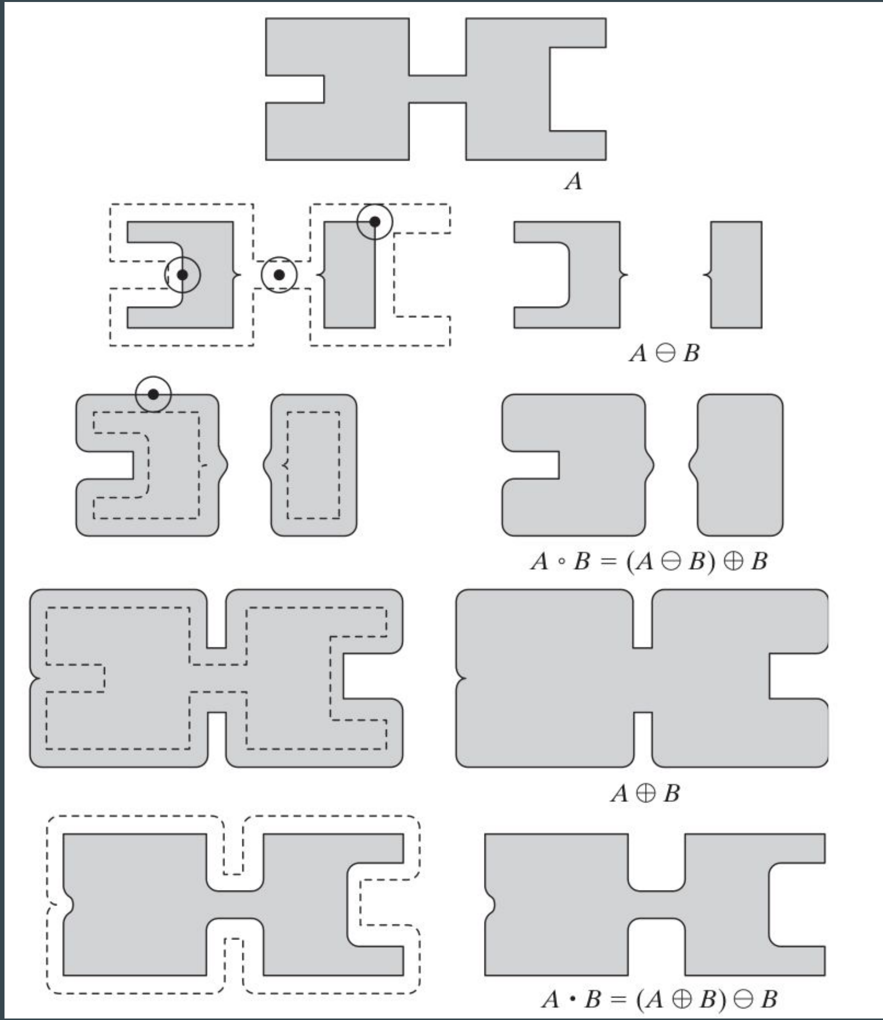


Fixing Broken Characters



0	1	0
1	1	1
0	1	0

Morphological Opening and Closing



Note

The dilation / erosion kernel size can be more than 1 also. This will involve looking at the “kernel size” amount of neighbourhood, basically more the kernel size, it will involve taking the max / min among more number of pixels, therefore increasing the effect both in the case of dilation and erosion

Closing and Opening Operations

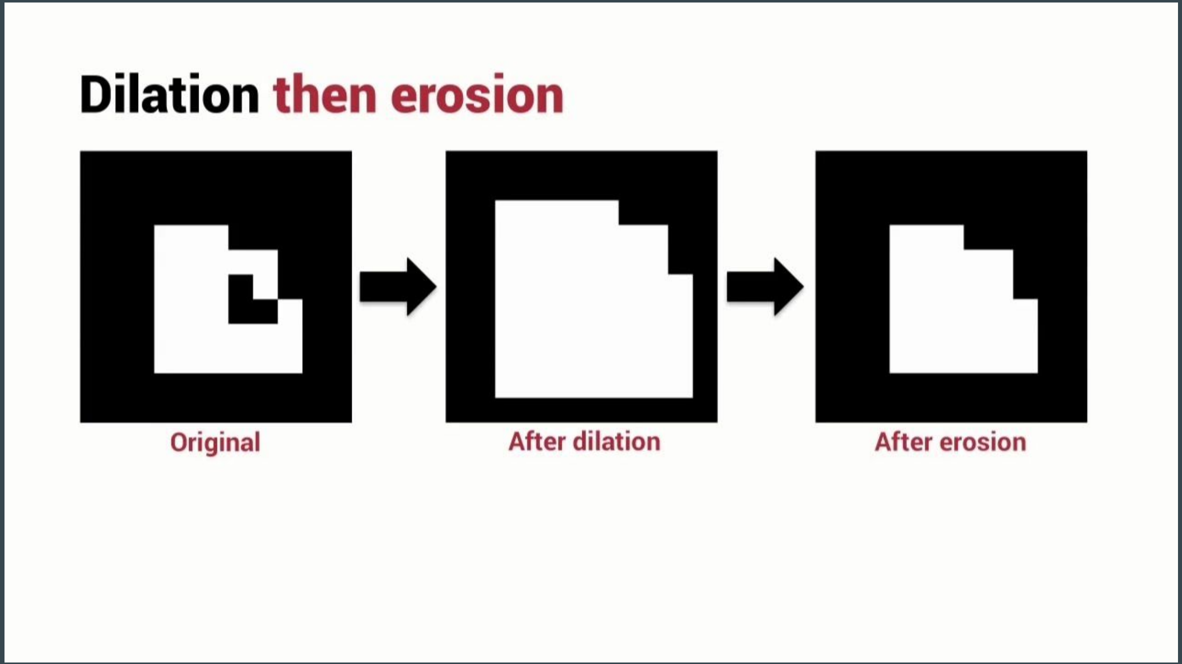
Closing - This operation involves dilation followed by erosion

Opening - This operation involves erosion followed by a dilation

Closing can connect white structures, whereas Opening mainly disconnects them

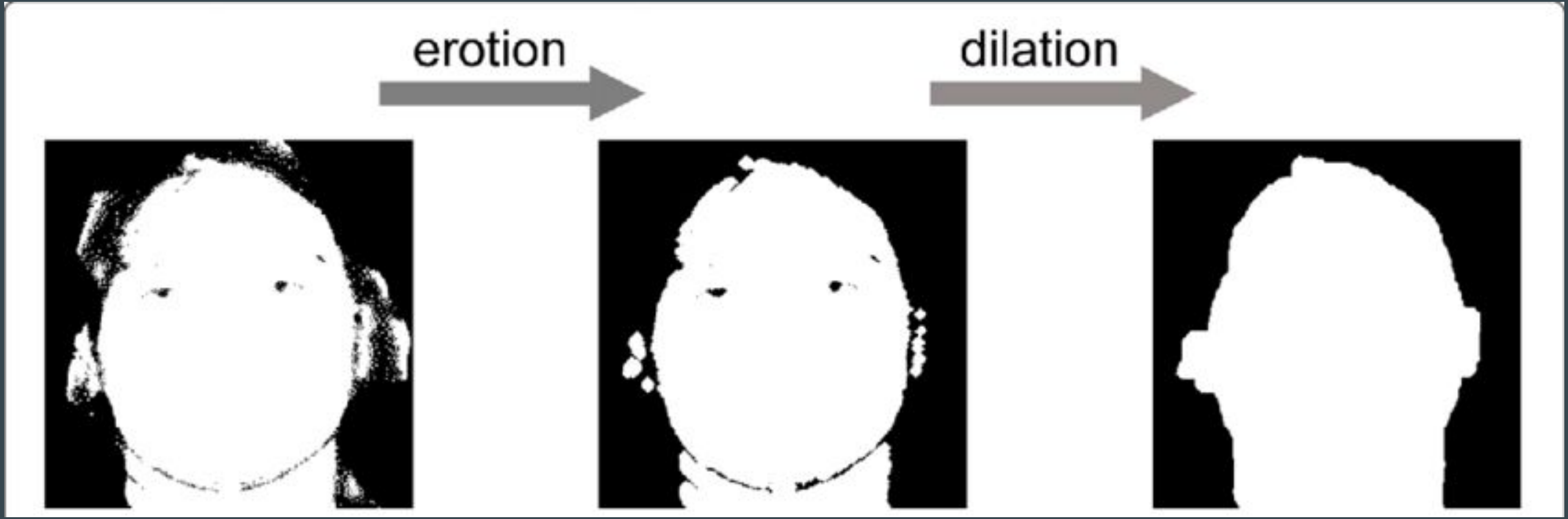
Closing operation

Closes the black hole in white components in the image.

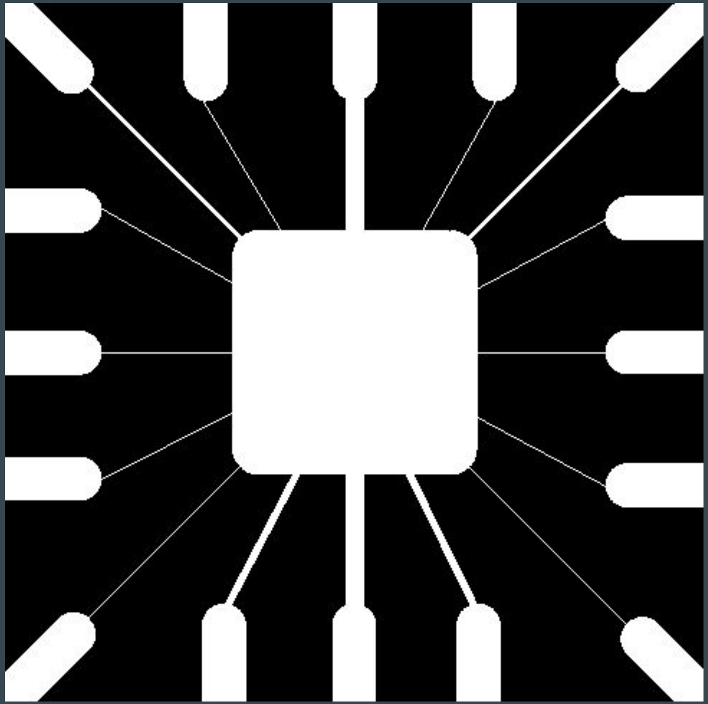


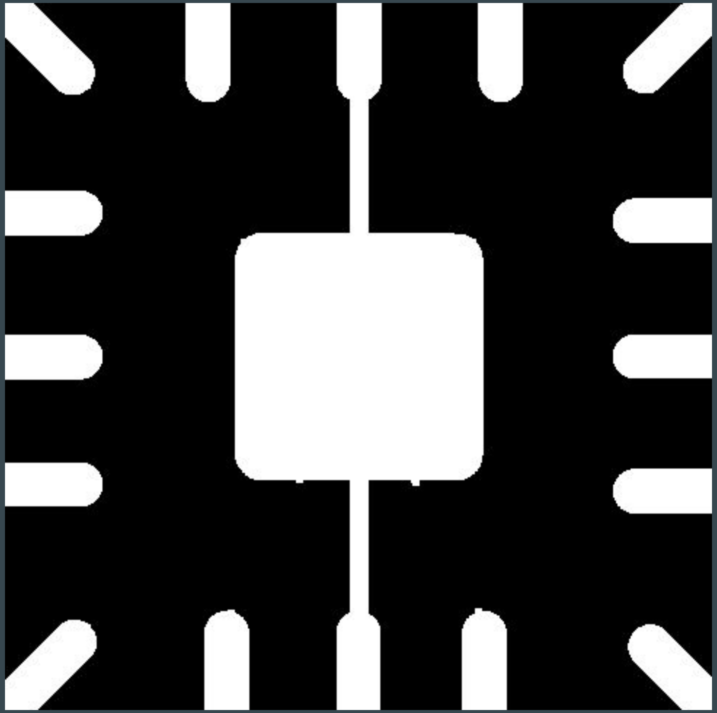
Opening operation

Helps in removing the non-major / very small white components



Opening Example





Normalizing the effect

Opening followed by closing can help in first disconnecting the components, then restore each of the individual eroded components again, essentially overall just removing the boundary.

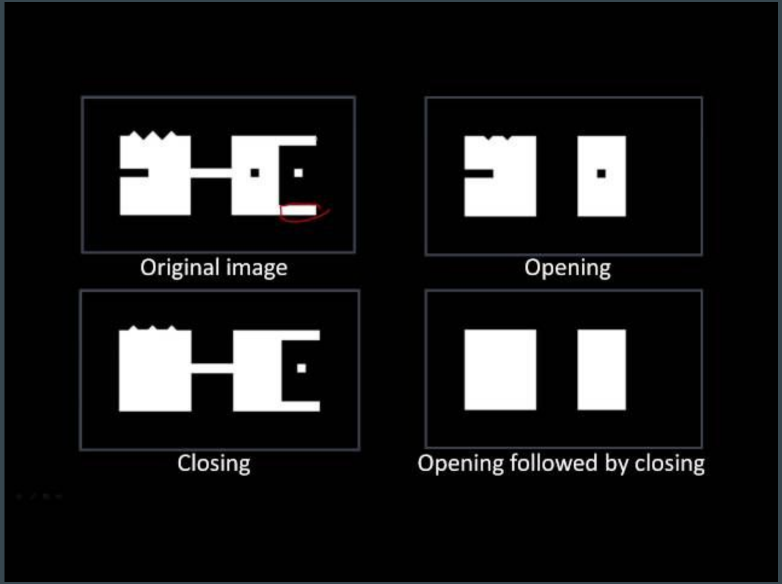
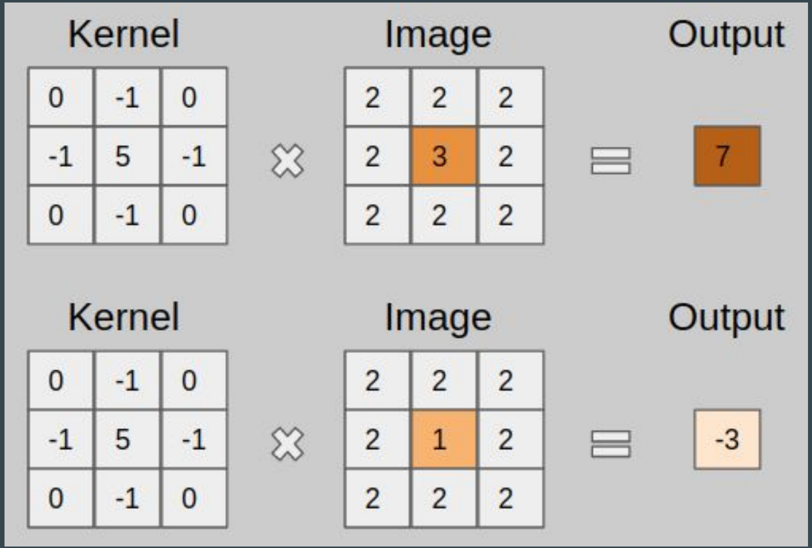


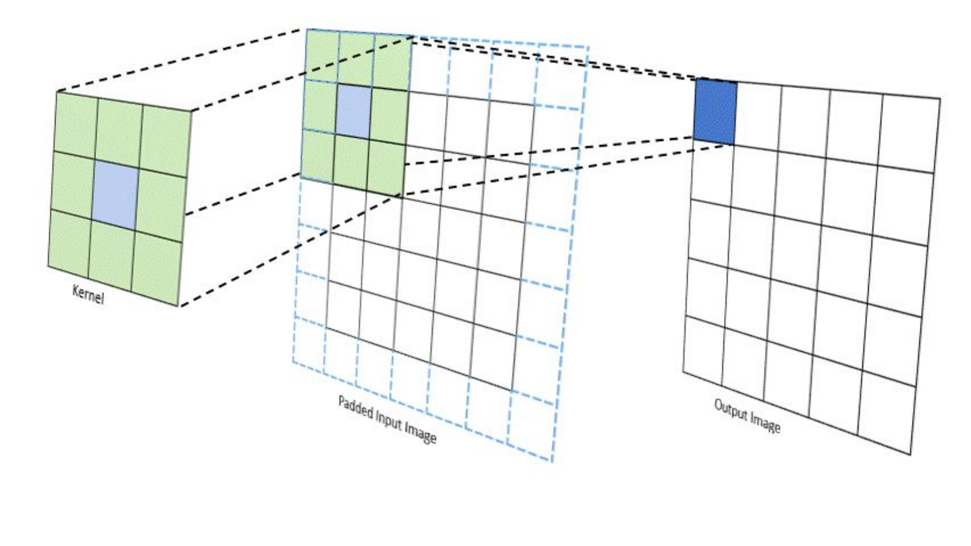
Image Filters (aka Kernels)

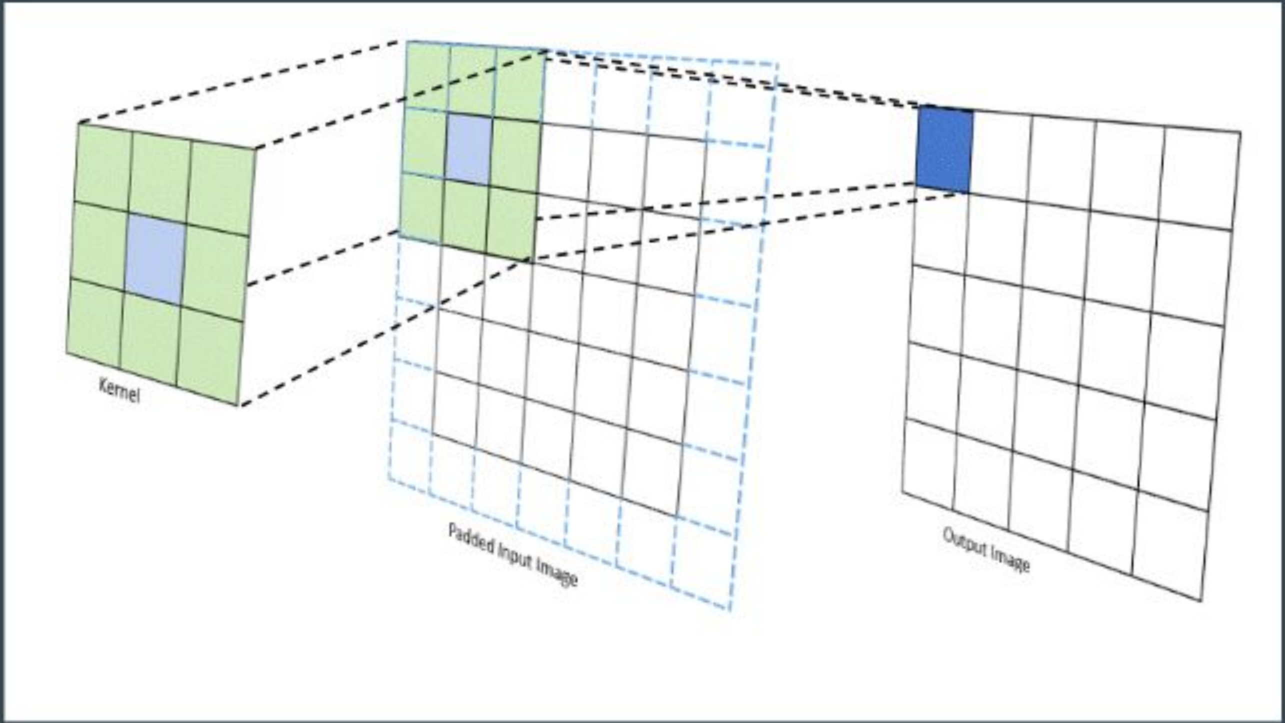
- Kernels are a key essence for processing images.
- They are used everywhere, for almost all image processing tasks, and also in the extremely popular, convolutional networks (CNNs).
- They are just a matrix, applied on an image, to get a new image



Kernels

- Check this website, has a great demo: https://setosa.io/ev/image-kernels/
- You place the matrix on top of the image and slide it...throughout the image. On each slide, you find the weighted sum of the values of pixels values in the overlapped region.
- The kernel values are just the weights and the same kernel is used for the entire image.




Kernel
Padded Input Image
Output Image

Kernel output shape

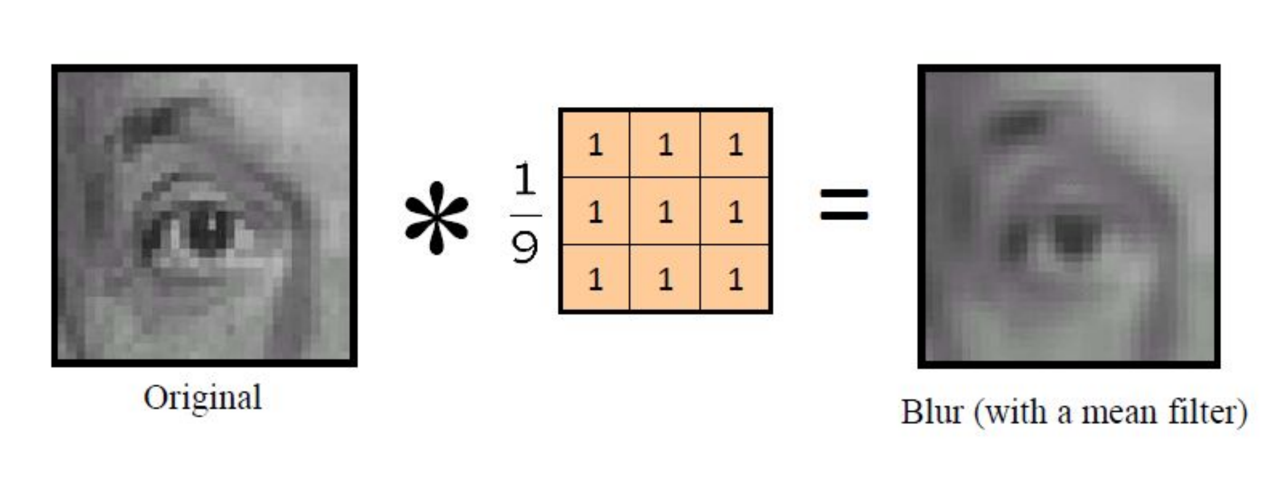
Note that ,you can have different strides horizontally and vertically. You can use the following equations to calculate the exact size of the convolution output for an input with the size of (width = W, height = H) and a Filter with the size of (width = F_w, height = F_h):

$$\text{output width} = \frac{W - F_w + 2P}{S_w} + 1$$

$$\text{output height} = \frac{H - F_h + 2P}{S_h} + 1$$

Blurring an Image

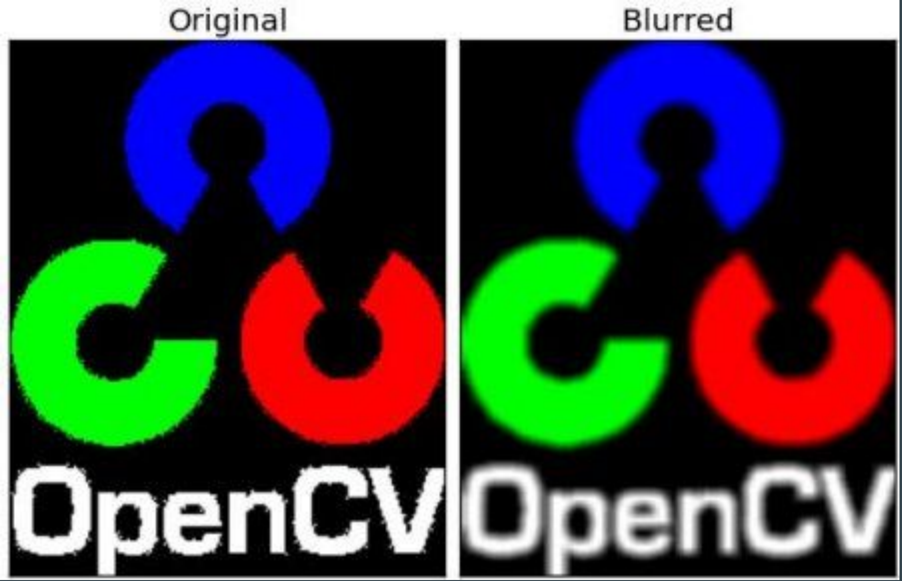
- The kernel has all ones, so the effect of applying the kernel on the image is just averaging the pixel values. This is called the mean kernel



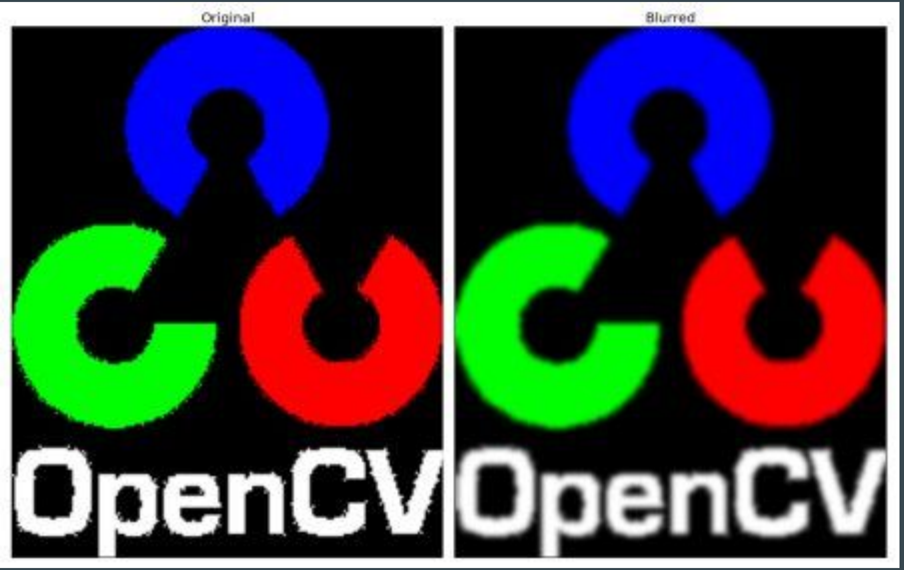
Gaussian Blur

- Instead of just taking the mean of the neighbouring pixels to find the average, here we take the weighted mean of them, with the middle pixel having the most weight / contribution to the final value.
- This is done using the Gaussian Kernel

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} \qquad \frac{1}{1003}\begin{bmatrix} 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 3 & 13 & 22 & 13 & 3 & 0 \\ 1 & 13 & 59 & 97 & 59 & 13 & 1 \\ 2 & 22 & 97 & 159 & 97 & 22 & 2 \\ 1 & 13 & 59 & 97 & 59 & 13 & 1 \\ 0 & 3 & 13 & 22 & 13 & 3 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \end{bmatrix}$$



Mean Blurring



Gaussian Blurring

Comparison

Test it out in the colab notebook and observe the differences

Edge Detection

This can be done using a kernel, essentially that finds the gradient of the image. X / Y - Gradient means how much pixel intensity value (in grayscale or colored image) change in just moving a little bit in the x / y - direction.

The kernel used is thus finding the difference between the left and the right part of the pixel, basically like the limit definition of the gradient. This can be done separately in X and Y. Below is the **Sobel Filter.**

X – Direction Kernel

-1	0	1
-2	0	2
-1	0	1

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

Edge Detection

We can also have the same kernel for finding both the X and Y direction gradients.

Also, for colored images, we can either first convert it to grayscale and then find the gradients, or we can find the gradients in the R,G,B channel individually and process them separately (maybe take their max for the final gradient output)

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Image Sharpening - An Important Application

- Unsharp Masking: This is a sharpening process to enhance the edges.
- The steps to do unsharp masking are:
  - Blur the image
  - Subtract the blurred image from the original image. (like removing the blurry part of the image).
  - Add these sharp edges left from the subtraction to the original image so that the edges become more sharp and thus the image looks more crisp.