IMAGE PROCESSING ESSENTIALS

# A Comprehensive Guide to Image Processing: Part 3

Morphological operators and their usage with OpenCV Python

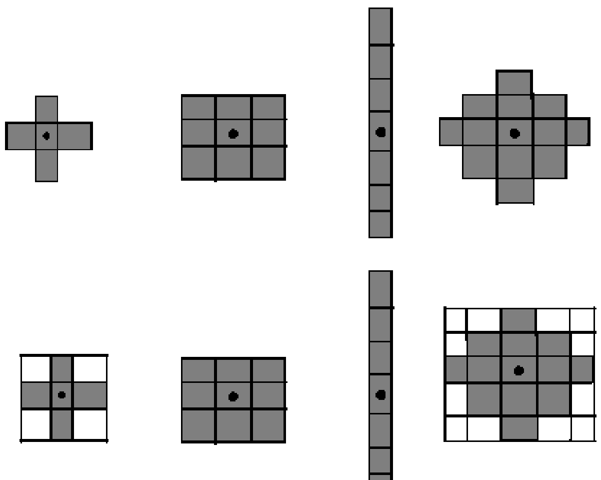
Yağmur Çiğdem Aktaş Aug 18 · 6 min read ★

In the penultimate part of this Image Processing series we will examine **Morphological Image Processing.**

Morphological image processing (or *morphology*) describes a range of image processing techniques that **deal with the shape (or morphology) of features in an image**. Morphological operations are typically applied to remove imperfections introduced during segmentation, and they are typically operate on binary (where the pixels of the image can only be 0 or 1) images.

**!!!** To read more about what is a feature, what is image segmentation etc., keep in touch with my upcoming publications.

***Structuring Element*** is the base structure we will use to apply a morphological operation. The shape of a Structuring Element can change depends of what type of a shape are we looking for in an image. Here are some examples of SE and their rectangular array forms. (Surely we need to convert a SE in a 2D matrix form to be able to represent it in programming)



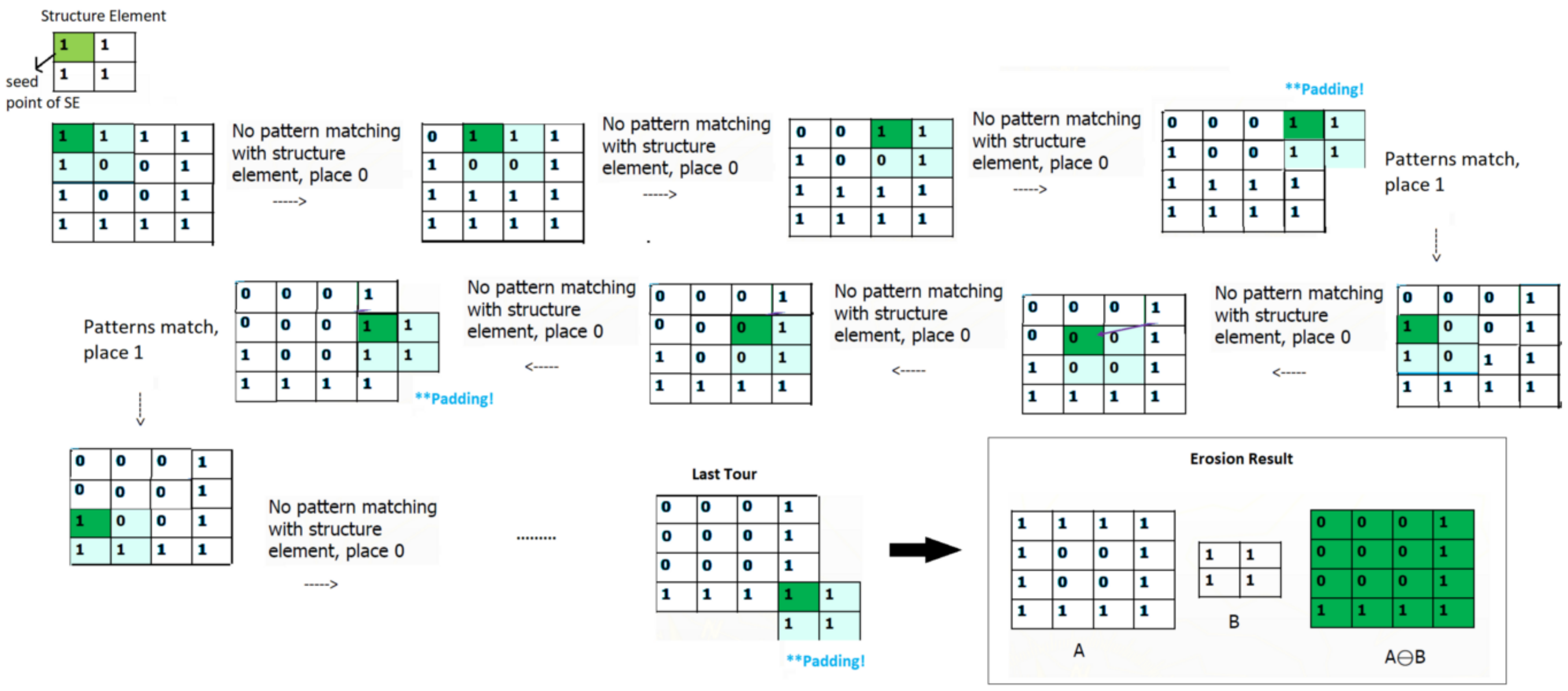
4 different Structural Element and their 2D matrix form “Image by Author”

We have 4 main morphological operations:

- Erosion
- Dilation
- Opening
- Closing

**Erosion**

In Erosion, the Structure Element travels trough the image and where the image pattern and Structure Element pattern **match exactly**, the pixel at the **seed point of Structure Element** in image is changed to 1, and to 0 if they don’t. Note that Erosion operation usually represented by ⊖



“Image by Author”

Border pixels are a common problem with window based operations as we saw in Image Processing 2 for convolution and correlation operations.
In Erosion, when **padding is needed** to the **structure element seed to be applied for border pixels**, we apply **1-padding**. So we add additional 1 values at the border. It is automatically done in the built-in functions of MATLAB, OpenCV or Scipy.

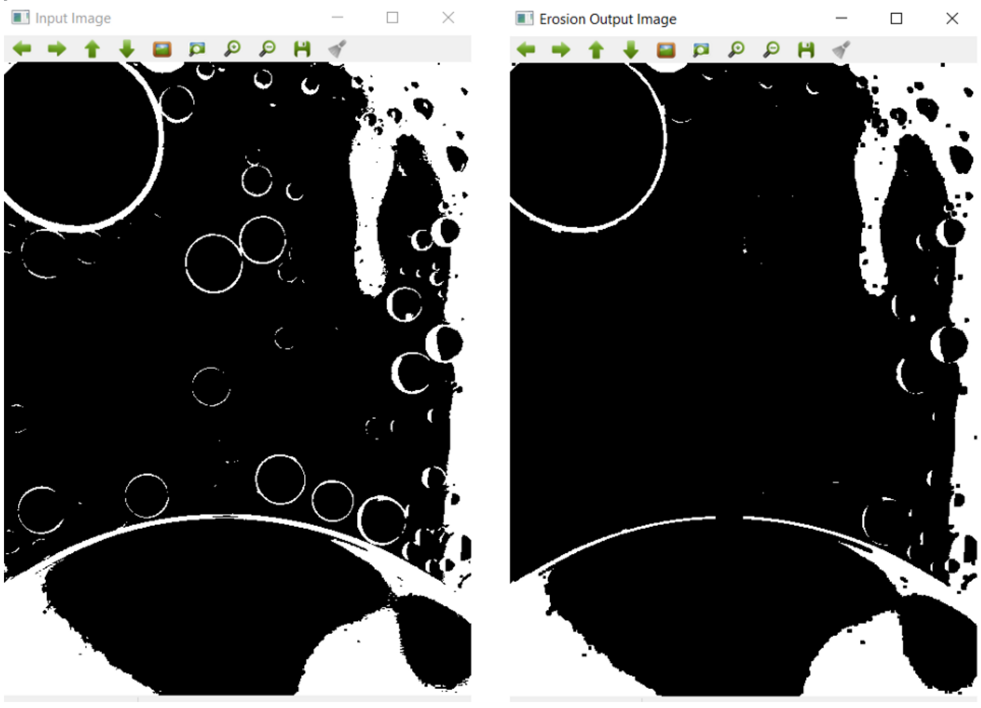
*Properties Of Erosion*

- Erosion removes pixels on object boundaries. In other words, it **shrinks** the **foreground objects**.

- **Enlarge** foreground **holes**.
- Like in Image Processing Kernels, a larger size of the Structure Element, the effect of Erosion increase.
- Of course, a different Structure Element gives different outputs on the same input image.

Having these properties, erosion can **split apart the joined objects or augment the distance between them** or **remove some redundant pixels.**

Let's look at some output images after applying erosion:



We see how the foreground holes get bigger and foreground objects disappear "Image by Author"

It's a very easy process to apply using OpenCV as shown in the **"Erosion with OpenCV " part of the code** which is attached at the end of this post.

If you would like to work with 2D arrays to examine some results, compare with what you calculated, you can use the Scipy library as shown in **"Erosion for 2D arrays" part of the code** which is attached at the end of this post.

### Dilation

In Dilation, the Structure Element travels trough the image and where the image pattern and Structure Element pattern has **1 matched pixel**, 1 is written as the output pixel value, 0 if they don't have any. Note that Dilation operation is usually represented by ⊕

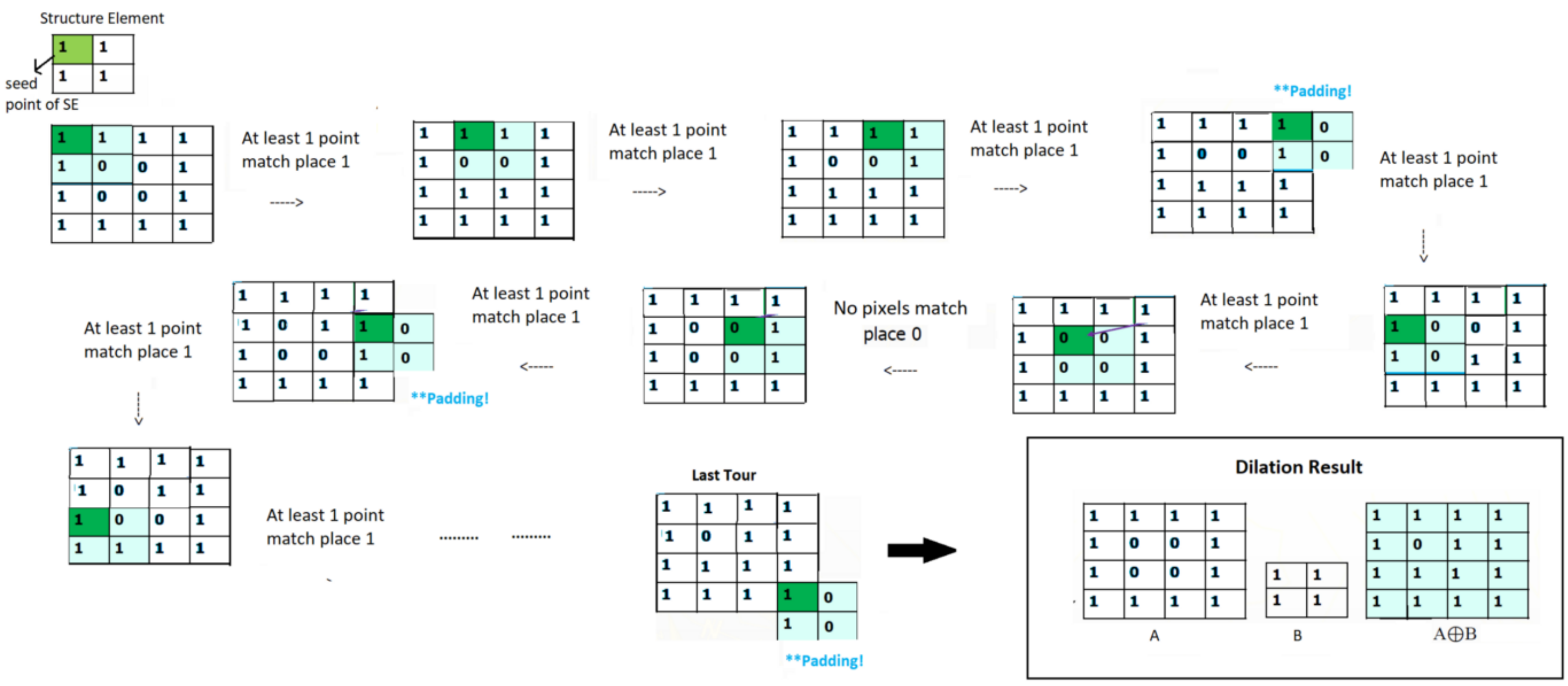


Figure 2 “Image by Author”

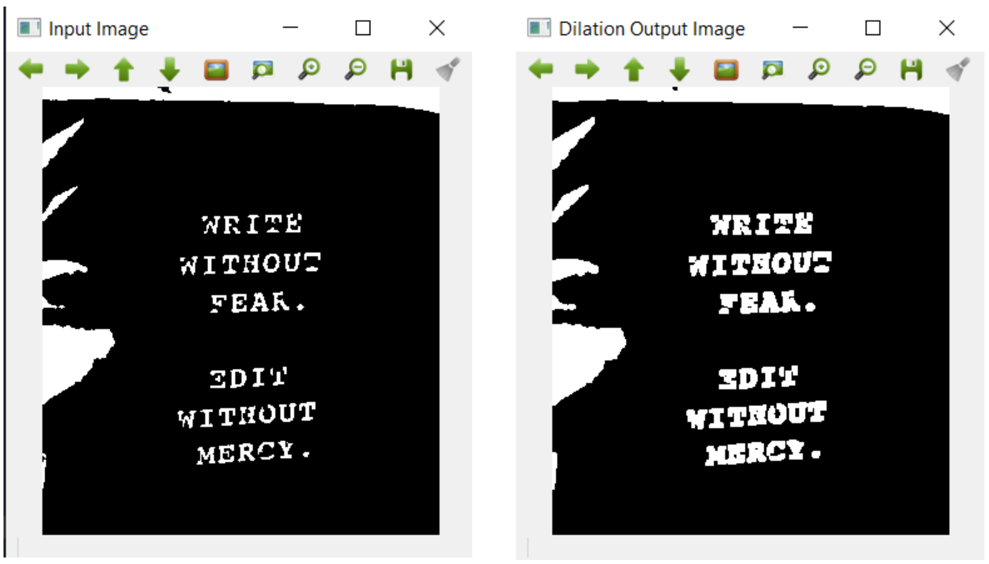
Note that in dilation, **0-padding is applied** unlike in Erosion!

*Properties Of Dilation*

The effects of Dilation are the opposite of Erosion.

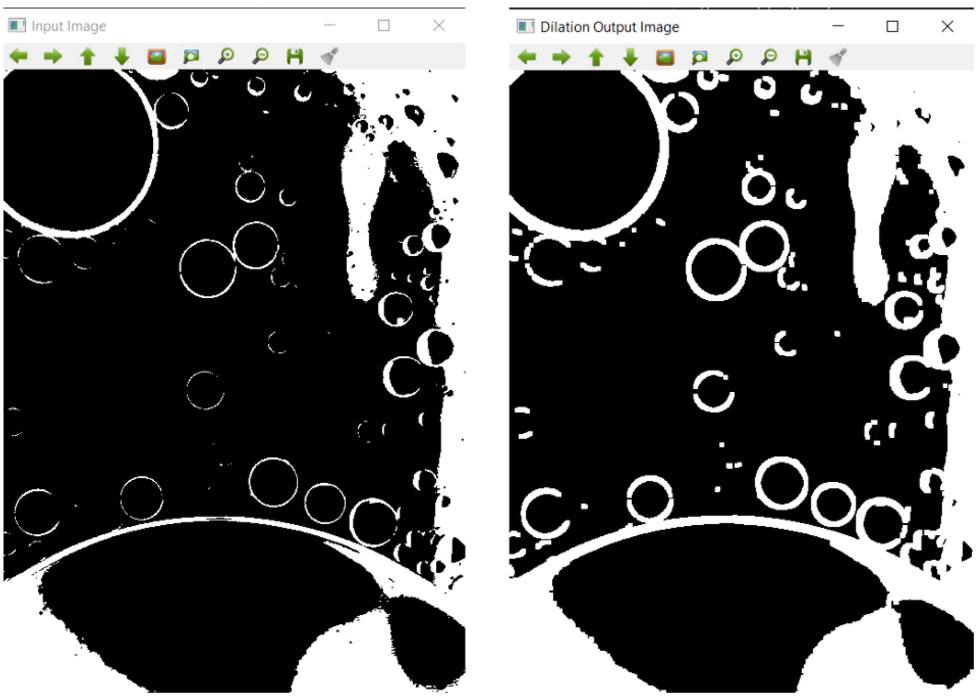
- Dilation adds pixels on object boundaries.
- Fill the holes in the foreground and **enlarge** foreground **objects.**
- But in the same way with Erosion, a larger structuring element gives larger dilation effect and the result is dependent on the structuring element.

Having these properties, Dilation can **repair breaks and missing pixels in foreground objects.**



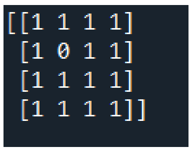
“Image by Author”

It’s a very easy process to apply using OpenCV as shown in the **“Dilation with OpenCV “ part of the code** which is attached at the end of this post.



“Image by Author”

If you would like to work with 2D arrays to examine some results and compare with what you calculated, you can use Scipy library again. Please look at the **“Dilation for 2D arrays” part of the code** which is attached at the end of this post. Here is the example output of that code part.



```
[[1 1 1 1]
 [1 0 1 1]
 [1 1 1 1]
 [1 1 1 1]]
```

“Image by Author”

## Combining Erosion & Dilation

We learned the base operations of Morphology. Now it’s time to use them to obtain more complex Morphological Operators. Complex ones are used to perform many different tasks.

- *Noise Removal with Morphology*

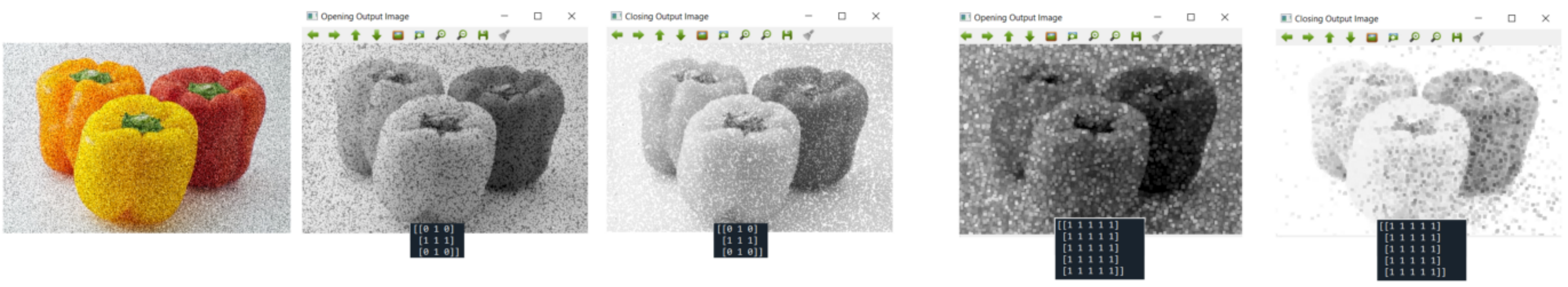
### Opening (A⊖B) ⊕B

Opening is just another name of **erosion followed by dilation.** It’s effective to **remove salt noise.**

### Closing (A ⊕ B) ⊖ B

Closing is just another name of **dilation followed by erosion.** It’s effective to **remove pepper noise.**

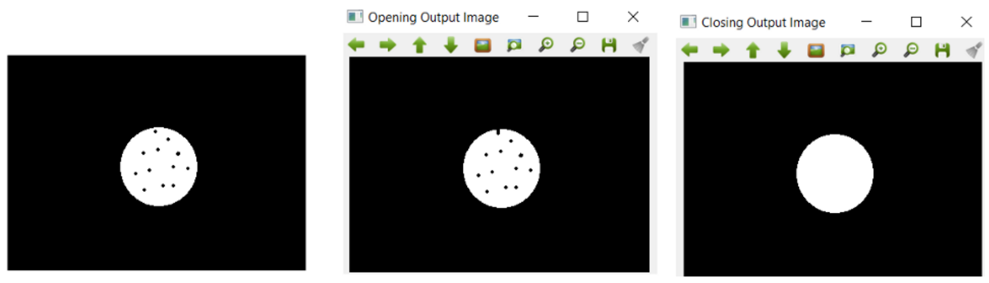
Let's examine some Opening and Closing outputs:

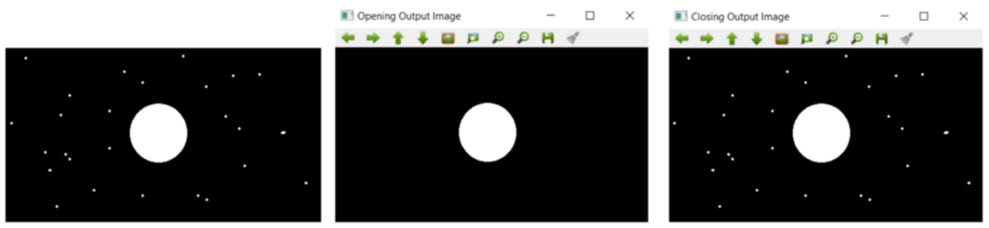


```
[[0 1 0]
 [1 1 1]
 [0 1 0]]
```

```
[[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]]
```

Input image with salt(white) and pepper(black) noise. 2 different Structure Element is used. We see Opening removes only salt noise while closing removes only pepper noise





The most left side images are the input images and we see the results of Opening or Closing operations with 2 different Structural Elements given in the middle. Please refer to the notes under the figures to better understand these 3 inferences:

- We see the effect of Opening on salt noise (white points) and Closing on pepper noise (black points).
- The result is very dependent on the Structural Element. If you don't choose a good one for your image, you may not obtain the expected effect of the morphological operators.
- Using Opening operator where Closing is needed or vice versa may corrupt the image.

Sure, if you have an image having salt and pepper noise at the same time, you can apply first opening than closing to have a better removal of noise.

You can easily select an image and perform Opening — Closing operations using OpenCV as shown in **"Opening & Closing Opencv" part of the code** which is attached at the end of this post

- ***Edge Detection with Morphology***

Another operation we learned how to do with Spatial Linear Filters in Image Processing Part 2 can also be performed with combined Morphological Operators

**External Boundary Extraction (A ⊕ B)-A**

We take the Dilation of the image and substract the original input image to obtain external edges.

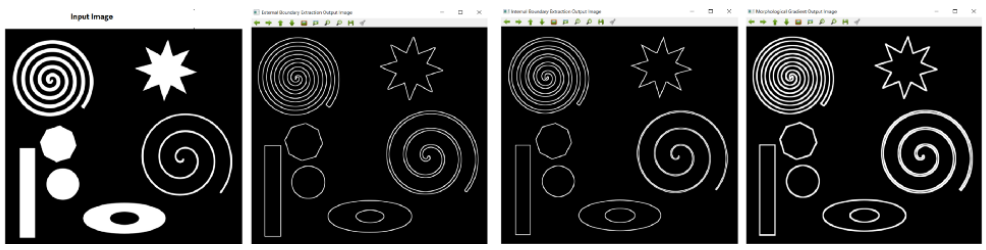
**Internal Boundary Extraction A- (A⊖B)**

We take the Erosion of the image and substract it from the original input image to obtain internal edges.

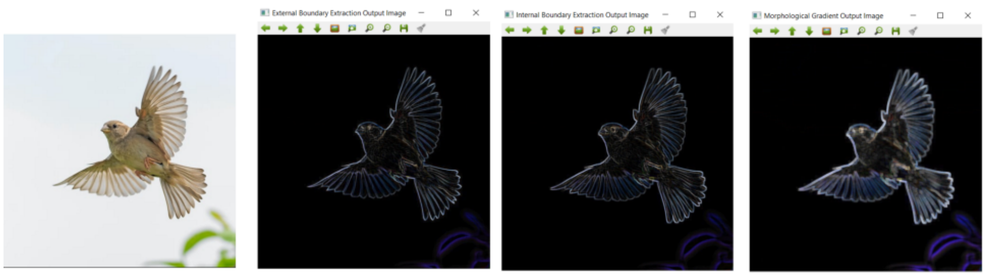
**Morphological Gradient (A ⊕ B)-(A⊖B)**

We subtract the Erosion output image from Dilation output image. While external boundary extraction is effective to obtain external side pixels of an edge and internal boundary extraction is effective to obtain internal side pixels of an edge, morphological gradient provides us the pixels "on" the edge. That gives **thicker edges** than internal or external boundary extraction.

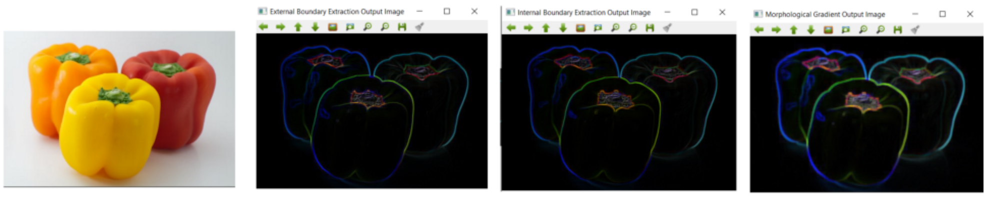
Look at the similarity of External and Internal Boundary Detection where Morphological Gradient gives thicker edges.


Input Image
External Boundary Extraction Output Image
Internal Boundary Extraction Output Image
Morphological Gradient Output Image

"Image by Author"


External Boundary Extraction Output Image
Internal Boundary Extraction Output Image
Morphological Gradient Output Image

"Image by Author"


External Boundary Extraction Output Image
Internal Boundary Extraction Output Image
Morphological Gradient Output Image

"Image by Author"

Using Dilation and Erosion operations you can implement your own edge detection codes with OpenCV as shown in **"Morphological Edge Detection" part of the code** which is attached just below:

That is all what we will discuss in this topic but if you want to move further with what you can do using Morphological Operators, you may search about **Region (Hole) Filling**, **Connected Components Extraction**, **Skeletonisation**, **Hit or Miss Transform,** etc.

Click that **github** link to reach the code and the images I used for this post and try it yourself!

You can move on to the final part on using an Image Processing Tool by clicking **here**!

Morphology Morphological Operations Dilation And Erosion Morphological Gradient Opening And Closing