Mathematical Morphology

Introduction

Mathematical morphology gathers several techniques that apply to binary images, the latter result for example from thresholding. The pixels in binary images take only two values: 0 or 1, true or false, white or black, etc. In this chapter, we consider that the images contain objects (represented by white pixels) on a background (black pixels), as in Fig 65.

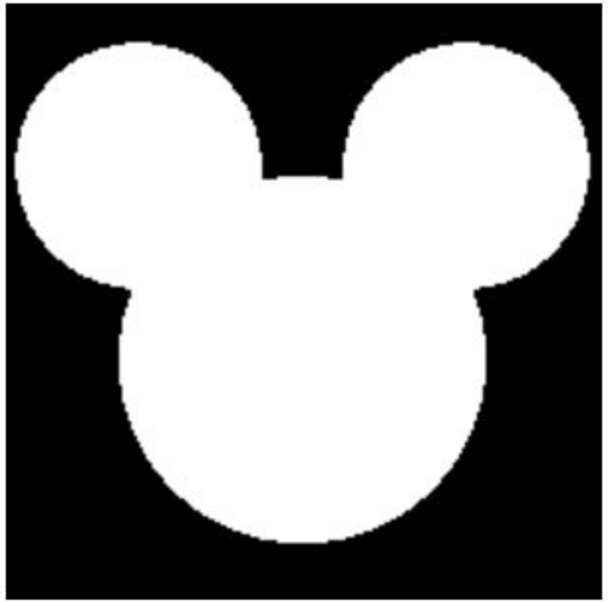


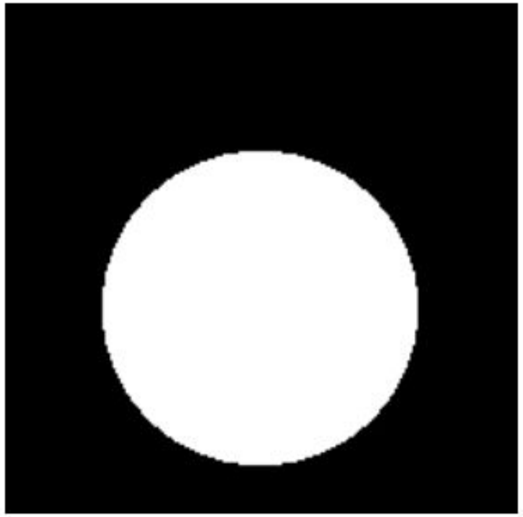
Fig 65. Example of a binary image composed of an object (a famous mouse head) on a background.

The basic operators of mathematical morphology, namely dilation and erosion are first presented. The combination of these two operators allows the definition of new operators, namely opening, closing and the hit-or-miss transform. A comparison of the operators is then presented. At last, we deal with measuring the geometric characteristics of binary objects.

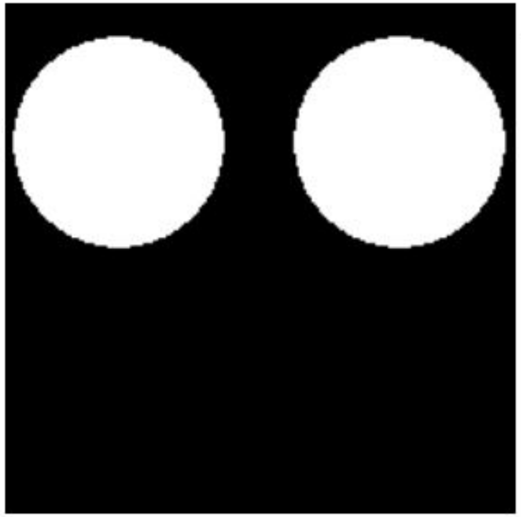
Set of Operations

We also consider that the objects in the image define a set of pixels, so mathematical morphology can use the usual operations on sets, listed below. Consider *A* and *B* as two sets.

Set A:

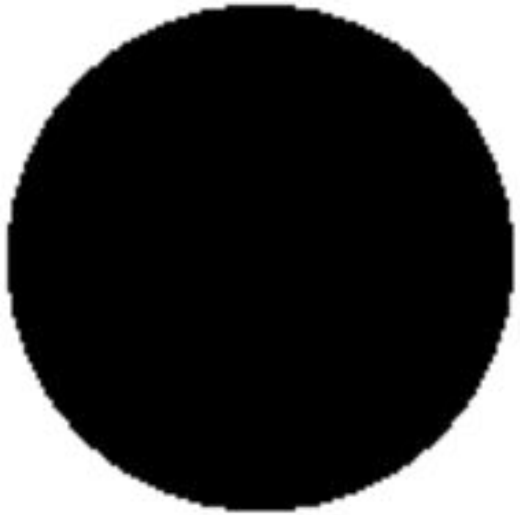


Set B:



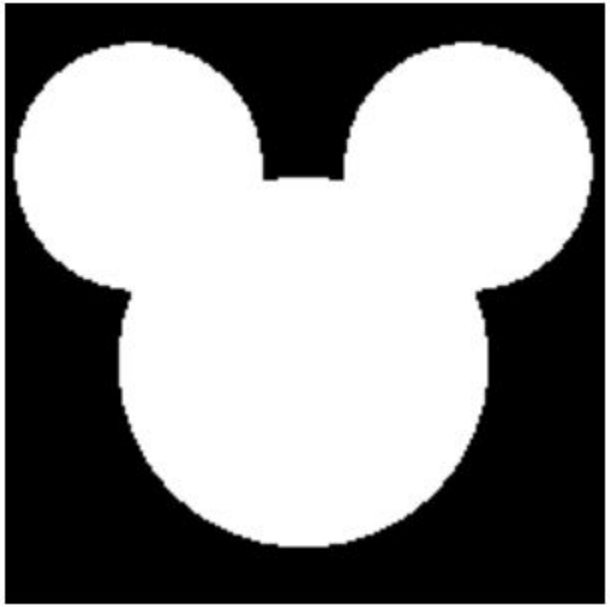
The complement of ***A*** is denoted ***A^c*** and is the set of pixels that are not in ***A.***

$$A^c = \{p \notin A\}$$



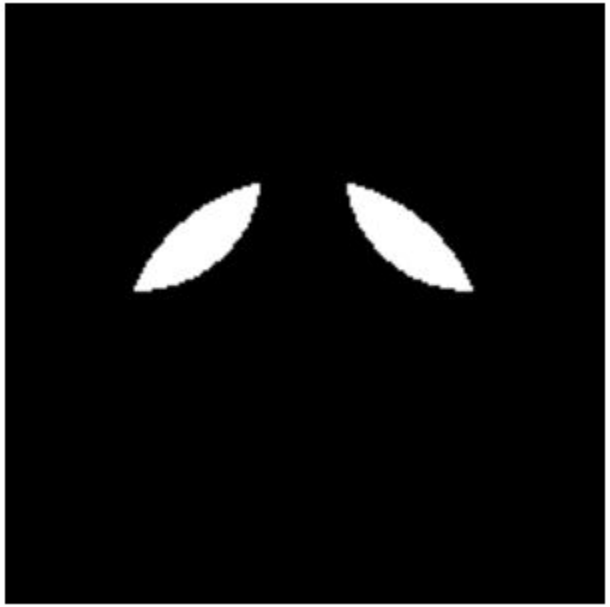
The union ***A* U *B*** is the set of pixels present in ***A*** or ***B*** or both:

$$A \cup B = \{p \in A \text{ or } p \in B\}$$



The intersection ***A*∩*B*** is the set of pixels present simultaneously in ***A*** and ***B***.

$$A \cap B = \{p \in A \,\text{and}\, p \in B\}$$



In a usual image, a pixel at coordinates (**m,n**) has four horizontal and vertical neighbors whose coordinates are given by

$$(m+1,n),\ (m-1,n),\ (m,n+1),\ (m,n-1).$$

Considering these 4 neighbors, we speak of 4-connectivity (french: *4-connexité*).

Besides, there exists also four diagonal pixels with coordinates

$$(m+1,n+1),\ (m+1,n-1),\ (m-1,n+1),\ (m-1,n-1).$$

These pixels, together with the 4-neighbors, are the 8 neighbors in 8-connectivity (french: *8-connexité*).

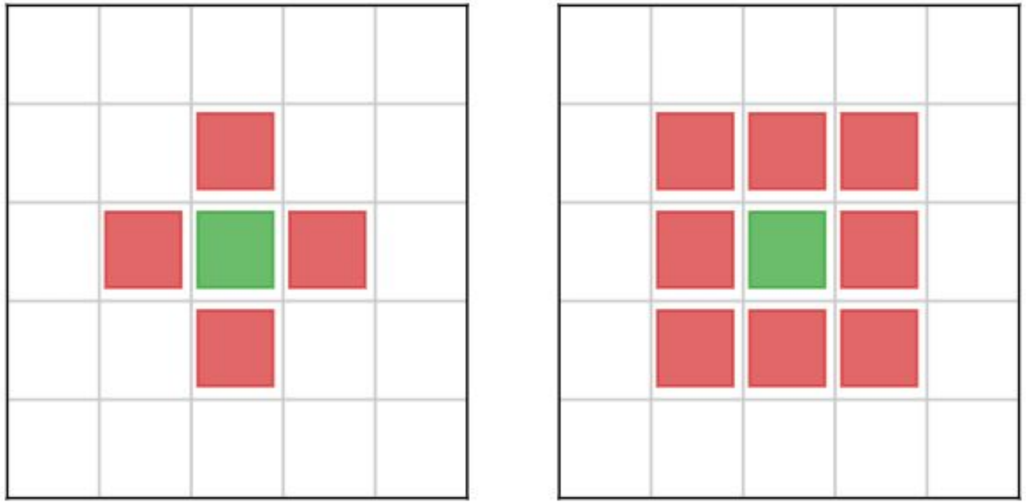


Fig. 66 The neighbors of the green pixel are represented in red, with 4-connectivity (left) and 8-connectivity (right). #

A path between two pixels with coordinates (**m1,n1**) and (**mN,nN**) is a sequence of pixels such that two consecutive pixels are neighbors in the considered connectivity.

Let ***S*** represent a set of pixels in an image. Two pixels are said to be connected if there exists a path between them consisting entirely of pixels in ***S***.

The set of pixels that are connected is called a connected component (french: *composante connexe*).

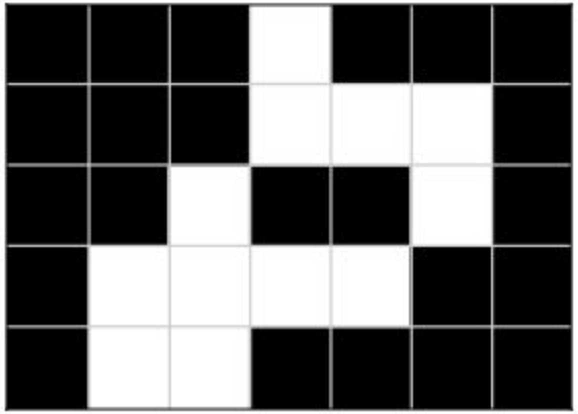


Fig. 67 In this image, there are 2 connected components with 4-connectivity, but only 1 connected component with 8-connectivity. #

Structuring element

In addition to this, the operators of mathematical morphology need a so-called structuring element (french: *élément structurant*). A structuring element $\boldsymbol{E}$ is a set of pixels (equivalent to a binary image) associated with an origin. Generally, the origin is located at the centre of the structuring element; but it may be elsewhere, even outside the pixels of the structuring element. In the sequel, we denote by $\boldsymbol{E_c}$ the structuring element centred on the pixel $\boldsymbol{c}$.

Basic operators

Dilation

Having an image $\boldsymbol{I}$ and a structuring element $\boldsymbol{E_c}$, the dilation (french: *dilatation*) of $\boldsymbol{I}$ by $\boldsymbol{E_c}$ is noted $\boldsymbol{I} \oplus \boldsymbol{E_c}$

The result of the dilation is obtained by moving the origin of the structuring element onto the white pixels of the image and keeping the set of pixels of each displaced structuring element.

Mathematically speaking:

$$I \oplus E_c = \{E_c \mid c \in I\}$$

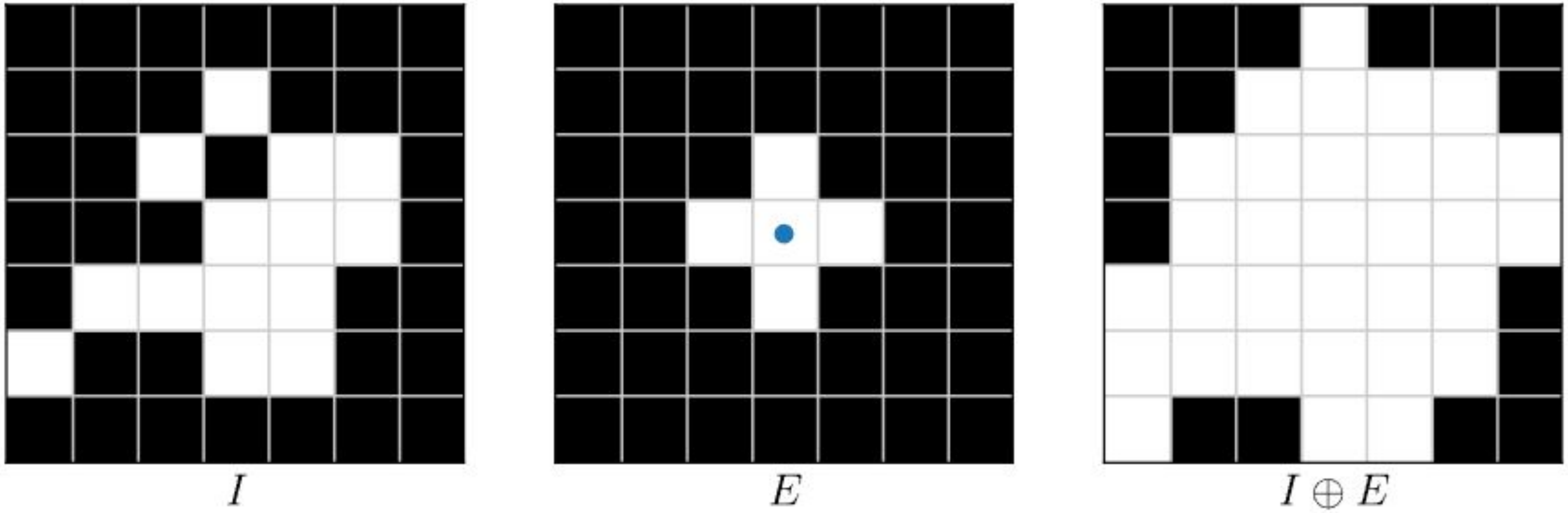


Fig. 68 Example of dilation on a small image I by the structuring element E_c (with the origin c is at the centre and represented by the blue dot). #

The structuring element is often described with a matrix. So, the structuring element in Fig. 68 is written as:

$$E = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Note that the matrix does not consider the zero pixels surrounding the main part of ***E***.

Properties

- Dilation is a binary operation and is not linear. Therefore, it cannot be expressed as convolution which is a linear mathematical operator.
- Dilation is associative, *i.e.* the application of two consecutive dilations can be done in any order:
- $(I \oplus E_1) \oplus E_2 = (I \oplus E_2) \oplus E_1 = I \oplus (E_1 \oplus E_2)$
- (here, the subscripts 1 and 2 mean two different structuring elements.)
- Dilation is a monotonous operation since the relations of inclusions are conserved:
- $\mathbf{I_1} \subseteq \mathbf{I_2} \quad \Rightarrow \quad \mathbf{I_1} \oplus \mathbf{E_c} \subseteq \mathbf{I_2} \oplus E_c$

Erosion

The erosion (french: *érosion*) of ***I*** by $\boldsymbol{E_c}$ is noted $\boldsymbol{I \ominus E_c}$.

The result of the erosion is obtained by moving the structuring element into the white pixels of the image and keeping only the origin of each displaced structuring element.

$$I \ominus E_c = \{c \mid E_c \subseteq I\}$$

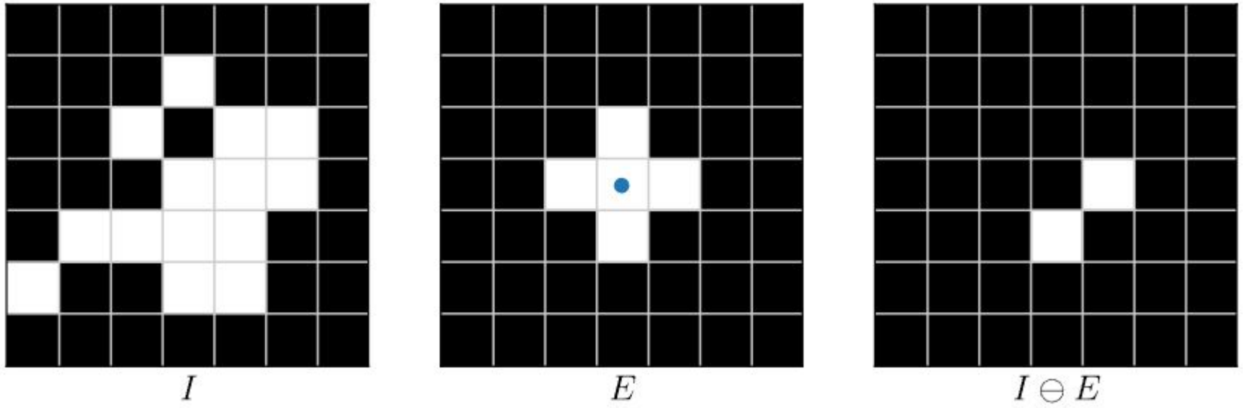


Fig. 69 Example of erosion on a small image I by the structuring element E_c (with the origin c is at the centre and represented by the blue dot). #

Properties

Erosion has similar properties as dilation.

- Erosion cannot be expressed as convolution.
- Erosion is associative:
- $\boldsymbol{(I \ominus E_1) \ominus E_2 = (I \ominus E_2) \ominus \mathrm{E}_1 = I \ominus (E_1 \oplus E_2)}$
- Note that the result of two successive erosions is equivalent to an erosion whose structuring element is the *dilation* of the two first structuring elements.
- Erosion is a monotonous operation:
- $\boldsymbol{I_1 \subseteq I_2 \quad \Rightarrow I_1 \ominus E \subseteq I_2 \ominus E}$

Duality

Dilation and erosion are dual operators. Considering the background as the object and the object as background (*i.e.* by working with the complement of the image), the dilation is converted to erosion and vice versa:

$$I^{c} \ominus E = (I \oplus E)^{c}$$
$$I^{c} \oplus E = (I \ominus E)^{c}$$

Composition of basic operators

Opening

Opening (french: *ouverture*) consists of an erosion followed by a dilation. The erosion removes small objects but also decreases the size of bigger objects. To avoid this, the result is dilated with the same structuring element.

$$I \circ E = (I \ominus E_c) \oplus E_c$$

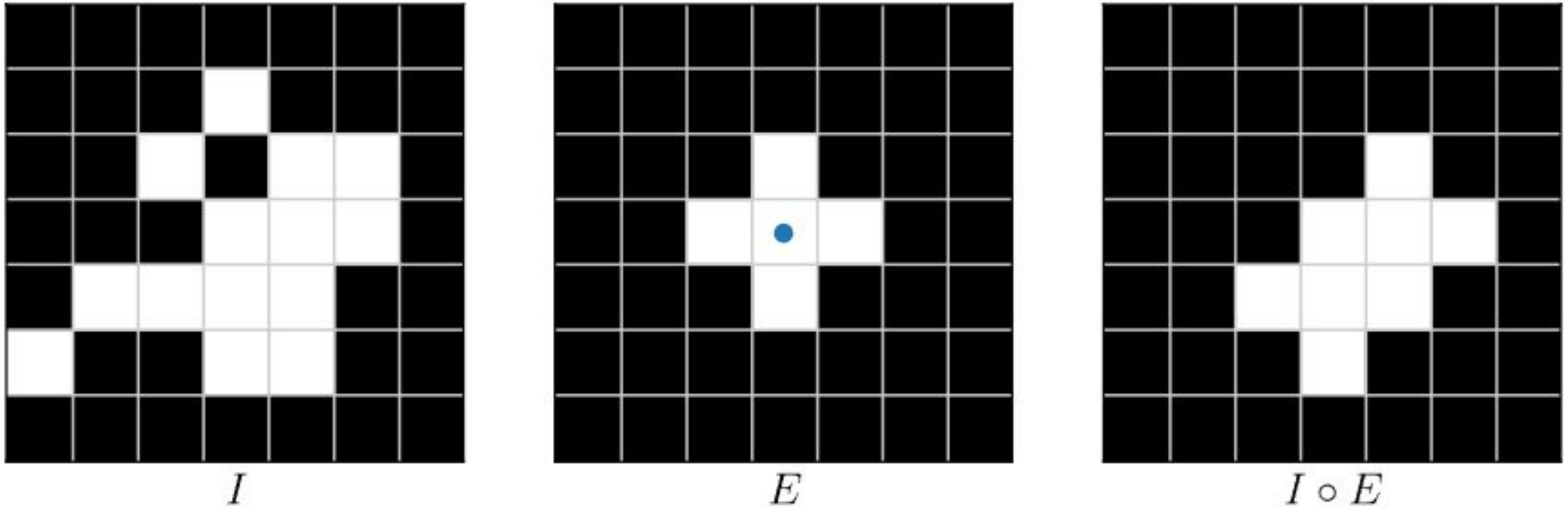


Fig. 70 Example of an opening on a small image I by the structuring element E_c (with the origin c is at the centre and represented by the blue dot). #

Property

- Opening is an idempotent operation, that is to say, applying twice the same opening gives the same result as only one opening:
- $(\boldsymbol{I} \circ \boldsymbol{E}) \circ \boldsymbol{E} = \boldsymbol{I} \circ \boldsymbol{E}$

Closing

Contrary to opening, closing (french: *fermeture*) is firstly a dilation, then an erosion. Indeed, expansion closes holes but enlarges objects. To avoid the widening of the objects, an erosion can be applied with the same structuring element.

$$I \cdot E = (I \oplus E) \ominus E$$

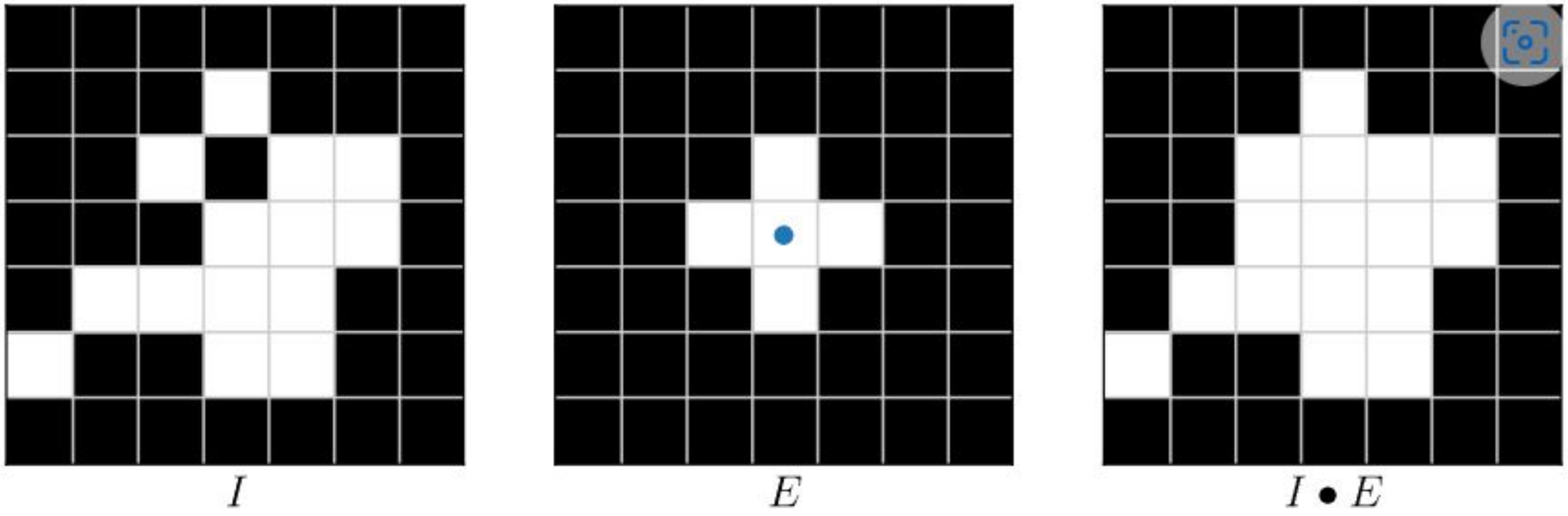


Fig. 71 Example of closing on a small image I by the structuring element E_c (with the origin c is at the centre and represented by the blue dot). #

Property

- Similarly to opening, closing is an idempotent operation:
- $(I \cdot E) \cdot E = I \cdot E$

Hit-or-miss transform

The hit-or-miss transform (french: *transformée tout-ou-rien*) is used to detect objects of a particular shape. It is the intersection of the two sets given by:

- the erosion by a first structuring element $\boldsymbol{E_1}$: $\boldsymbol{I \ominus E_1}$,
- and the erosion of the background by a second structuring element $\boldsymbol{E_2}$: $\boldsymbol{I_c \ominus E_2}$ with $\mathbf{E_1 \cap E_2 = \varnothing}$ (the structuring elements must be disjointed).

The hit-or-miss transform by the two structuring elements $\mathbf{E_1}$ and $\mathbf{E_2}$ is noted $\boldsymbol{I \otimes (E_1, E_2)}$:

$$\begin{aligned} I \otimes (E_1, E_2) &= (I \ominus E_1) \cap (I^{\mathrm{c}} \ominus E_2) \\ &= (I \ominus E_1) \cap (I \oplus E_2)^{\mathrm{c}} \end{aligned}$$

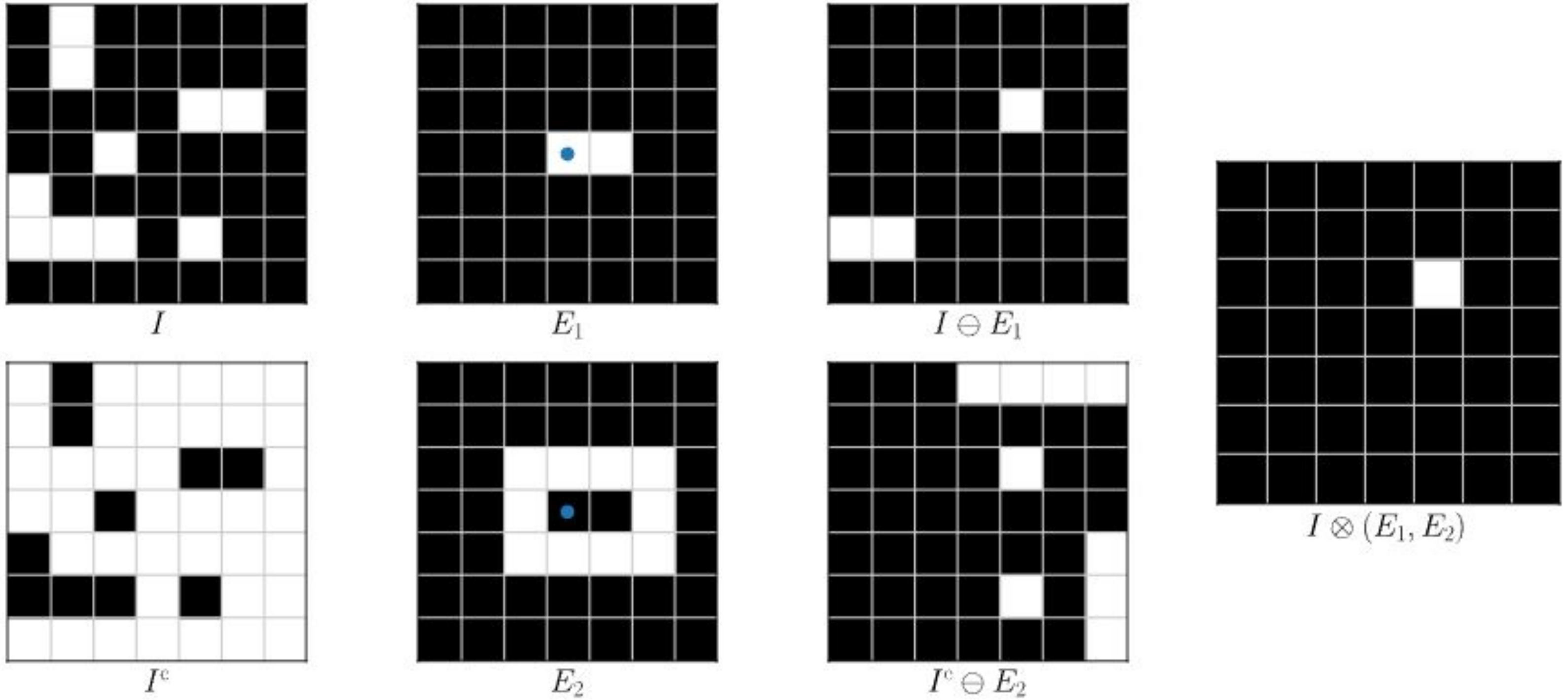


Fig. 72 Example of a hit-or-miss transform applied on the image I by the structuring elements E_1 and E_2. The origins of the structuring elements are marked by the blue and green dots.

Sometimes, the two structuring elements are combined into a single structuring element whose pixels have the following values:

- 1: pixels that belong to the object to detect,
- −1: pixels that do not belong to the object to detect (*i.e* pixels of the background),
- 0: unused pixels (also called “don’t care pixels”).

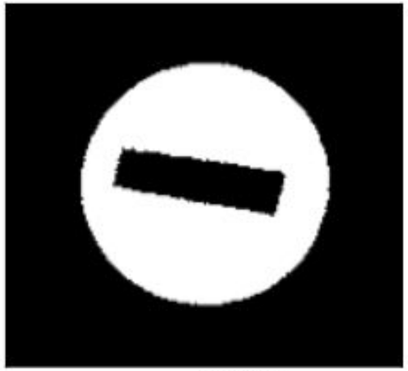
With this notation, the structuring element of the hit-or-miss transform in Fig. 72 writes:

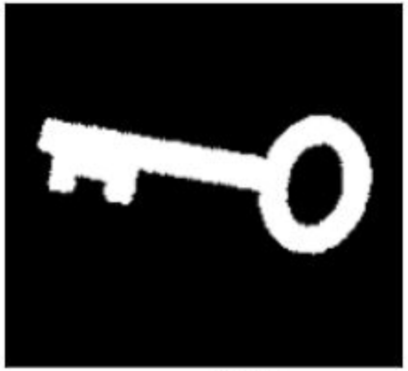
$$(E_1, E_2) = \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & -1 \\ -1 & -1 & -1 & -1 \end{pmatrix}$$

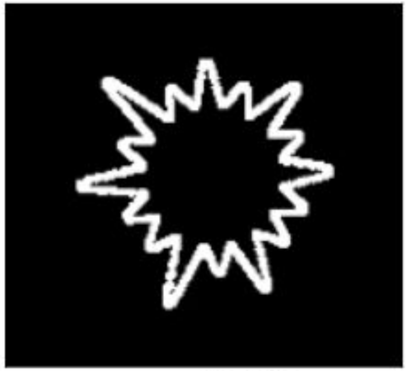
Measure

It is possible to measure the geometric characteristics of the objects in a binary image, such as the area or position. In scikit-image, the most common properties are implemented in `skimage.measure`, especially in the function `regionprops`. This section lists some of them, as an example of possibilities.

Area

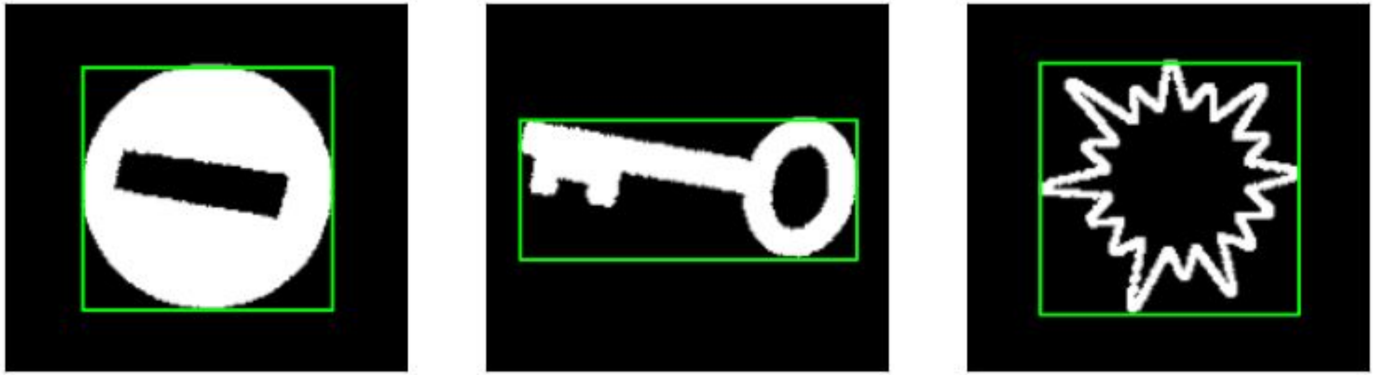

area = 7730 px


area = 3601 px


area = 2432 px

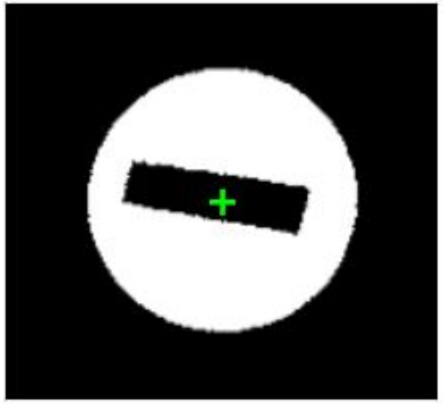
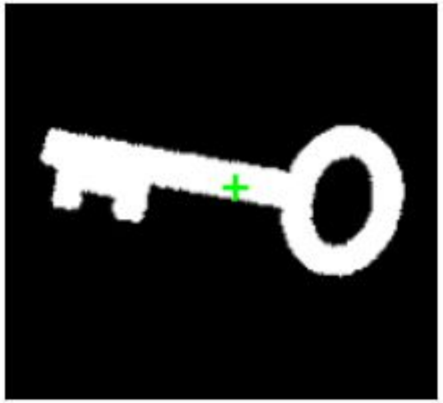
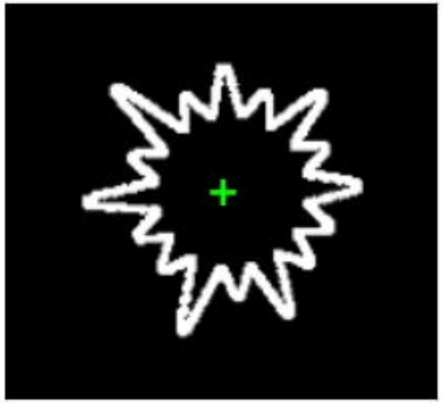
Area of an object is the number of pixels of this object. The larger the object and the fewer holes it contains, the larger the area.

Bounding box



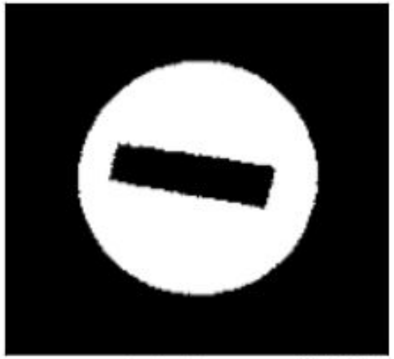
The bounding box (french: *boîte englobante*) is the smallest rectangle that contains the object.

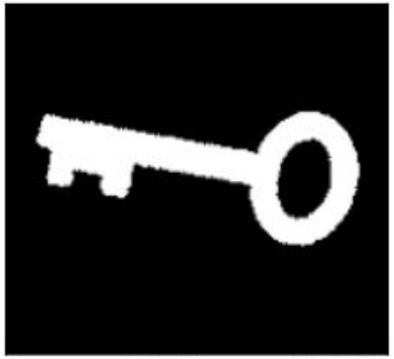
Centroid

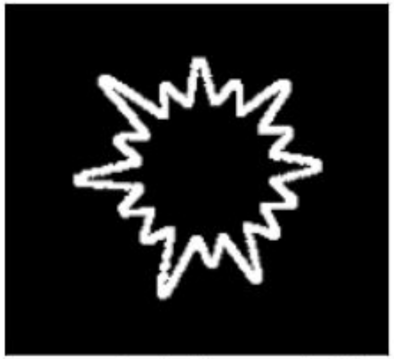




The centroid is the centre of gravity of the object. It can be outside the object.

Eccentricity

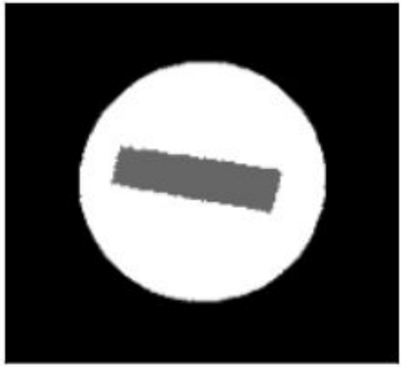

eccentricity = 0.23

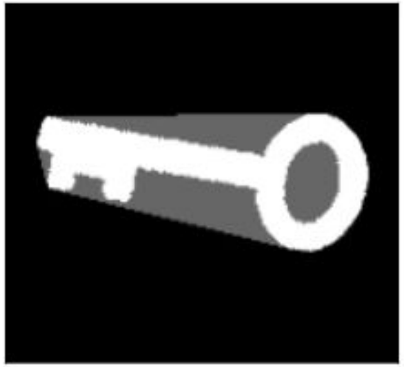

eccentricity = 0.96

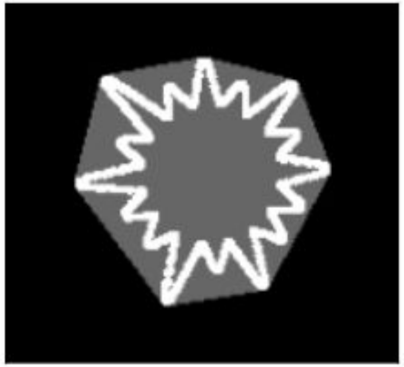

eccentricity = 0.31

Eccentricity measures the elongation, or stretching, of the object, as it would be an ellipse. It is a positive value, where 0 corresponds ot a round object.

Solidity


$\text{solidity} = 0.84$


$\text{solidity} = 0.53$

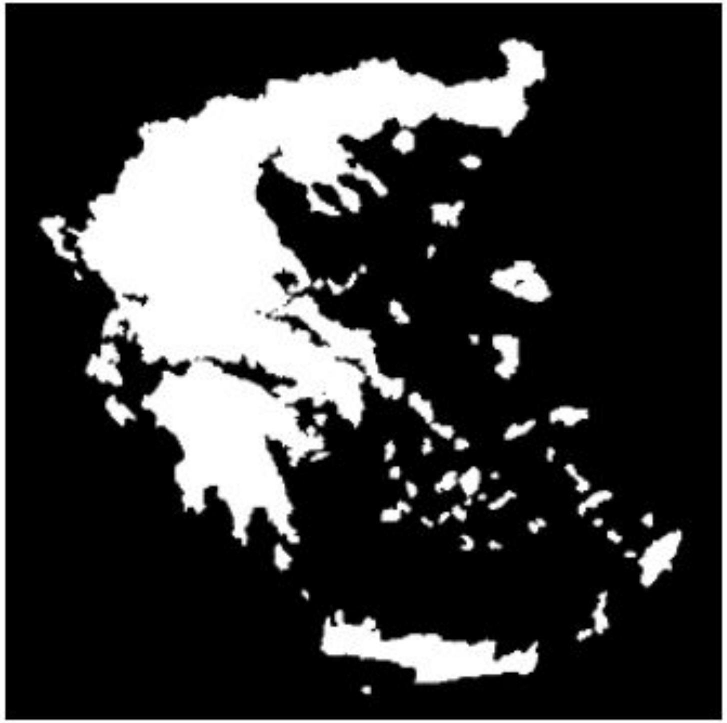

$\text{solidity} = 0.26$

Solidity is the ratio of pixels in the objects to pixels of the convex hull image. For clarity, the convex hull of the objects is represented in dark gray into the images. It gives a measure of the compactness of the object.

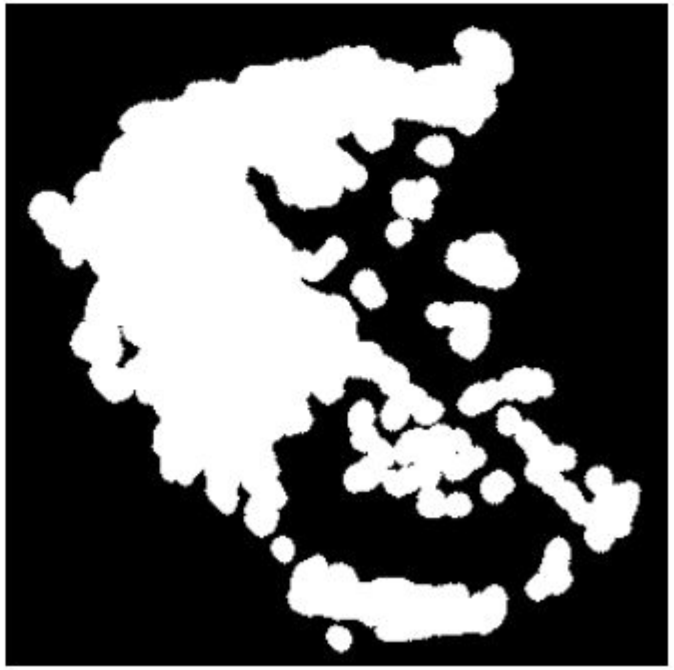
Conclusion

Mathematical morphology gathers operators that work on the form of objects. We have studied some operators applied to binary images, but there are extensions of these techniques to grayscale images.

The results given by the four main operators on binary images are listed



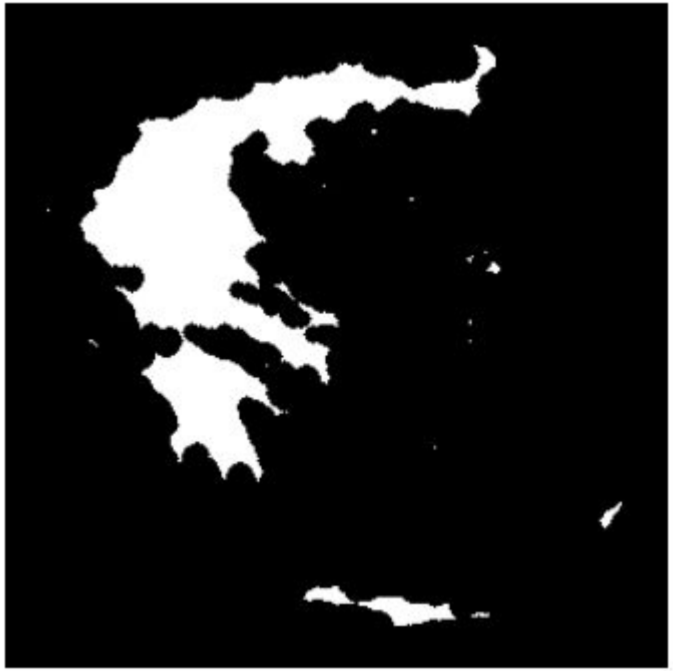
Original image



Dilation

$\oplus$

- increases the size of objects (*e.g.* the islands are bigger)
- fills the small holes (*e.g.* the Gulf of Corinth does not more exist after dilation)
- welds close objects (*e.g.* some islands are grouped into a larger island)



Erosion

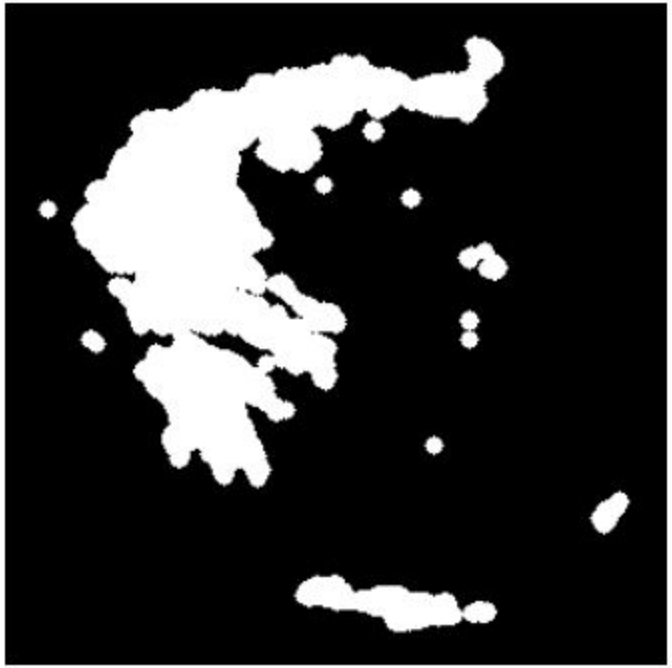
$\ominus$

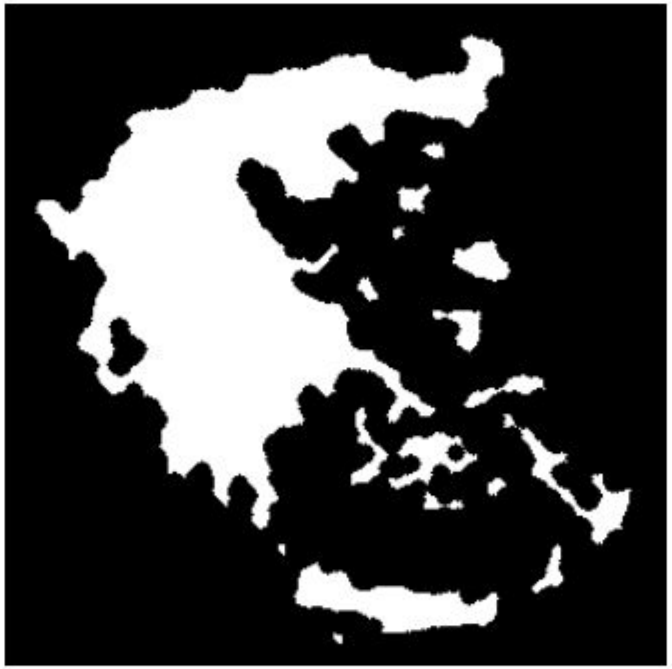
- decreases the size of objects (*e.g.* Crete is smaller)
- widens the holes (*e.g.* the Gulf of Corinth is bigger)
- separates the connected objects by a small bridge (*e.g.* the Peloponnese and mainland Greece are now disconnected)
- removes small items (*e.g.* small islands have disappeared)

Opening

◦

- conserve the size of the objects (while they can be slightly distorted)
- removes small objects (*e.g.* small islands have disappeared)
- smoothes the contours (*e.g.* the coasts are no more jagged)



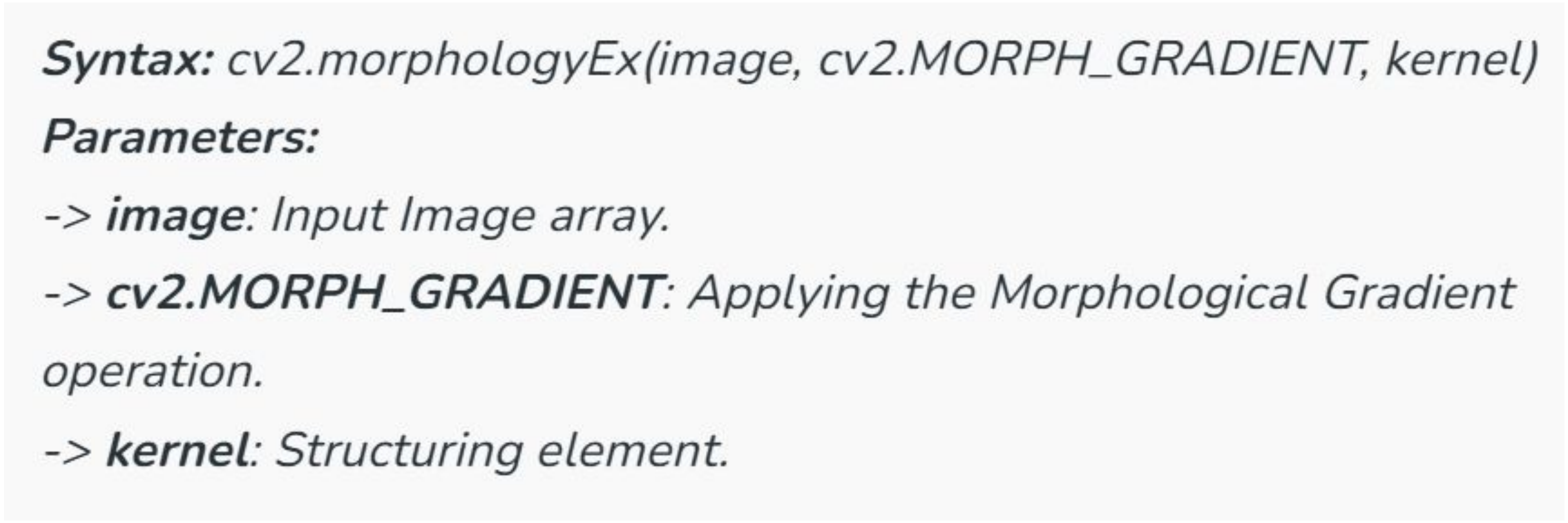


Closing

.

- conserve the size of the objects (while they can be slightly distorted)
- fills small holes (*e.g.* gulfs and lakes are removed)
- welds close shapes (*e.g.* some separated islands now form only one)

Python | Morphological Operations in Image Processing (Gradient) | Set-3



Syntax: *cv2.morphologyEx(image, cv2.MORPH_GRADIENT, kernel)*

Parameters:

*-> **image**: Input Image array.*

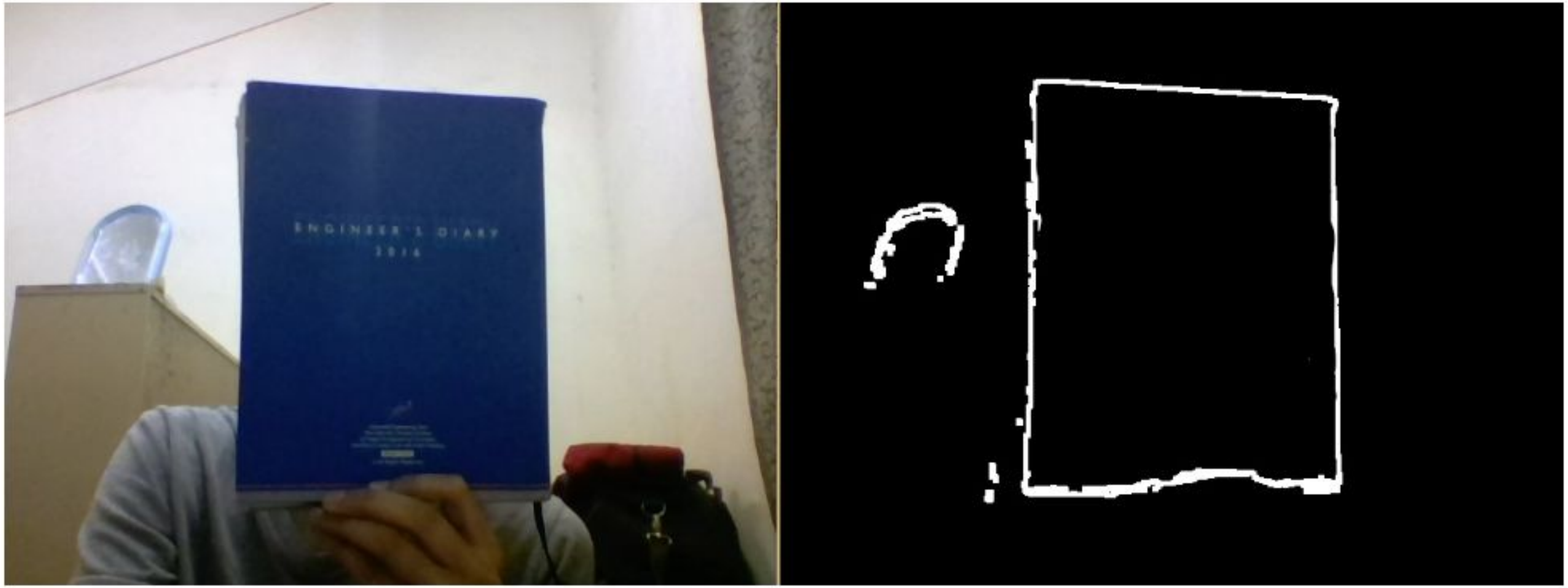
*-> **cv2.MORPH_GRADIENT**: Applying the Morphological Gradient operation.*

*-> **kernel**: Structuring element.*

```
# Python program to illustrate
# Gradient morphological operation
# on input frames

# organizing imports
import cv2
import numpy as np

# return video from the first webcam on your computer.
screenRead = cv2.VideoCapture(0)

# loop runs if capturing has been initialized.
while(1):
	# reads frames from a camera
	_, image = screenRead.read()

	# Converts to HSV color space, OCV reads colors as BGR
	# frame is converted to hsv
	hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

```
# initializing the mask to be
# convoluted over input image
mask = cv2.inRange(hsv, blue1, blue2)

# passing the bitwise_and over
# each pixel convoluted
res = cv2.bitwise_and(image, image, mask = mask)

# defining the kernel i.e. Structuring element
kernel = np.ones((5, 5), np.uint8)

# defining the gradient function
# over the image and structuring element
gradient = cv2.morphologyEx(mask, cv2.MORPH_GRADIENT, kernel)

# The mask and closing operation
# is shown in the window
cv2.imshow('Gradient', gradient)
```

```
        # Wait for 'a' key to stop the program
        if cv2.waitKey(1) & 0xFF == ord('a'):
                break

# De-allocate any associated memory usage
cv2.destroyAllWindows()

# Close the window / Release webcam
screenRead.release()
```



The output image frame shows the outline generated over the blue book and the blue object in top left.