Morphological Feature Extraction of Image

Kirti Mandloi (0801CS171035)

gs0801cs171035@sgsitsindore.in

December 20, 2020

Contents

Chapter 1

1 Introduction

1.1 Morphological Image Processing

The field of mathematical morphology contributes a wide range of operators to image processing, all based around a few simple mathematical concepts from set theory. It is a collection of non-linear operations related to the shape or morphology of features in an image, such as boundaries, skeletons, etc. Morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images, but if needed color images can be converted to binary for further processing. For a binary image, white pixels are normally taken to represent foreground regions, while black pixels denote background.

1.2 Applications

Mathematical Morphological operations are used to extract image components that are useful in the representation and description of region shape, such as

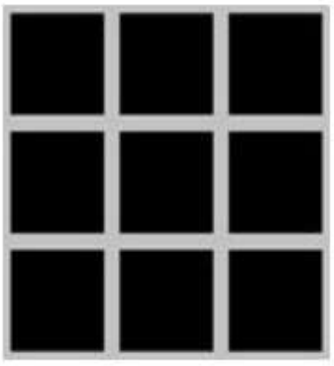
- boundaries extraction
- skeletons
- convex hull
- morphological filtering
- thinning
- region filling
- noise removal
- Medical images processing

Chapter 2

Mathematical Formulation

2.1 Structuring Element

An essential part of the morphological operations is the structuring element used to probe the input image. A structuring element is a matrix that identifies the pixel in the image being processed and defines the neighborhood used in the processing of each pixel. We typically choose a structuring element based on size and shape of the objects we want to process .In many implementations of morphological operators, the structuring element is assumed to be a particular shape and so is hardwired into the algorithm. Example: 3 X 3 square (SE)
In practical , it is matrix of 1's and 0's.

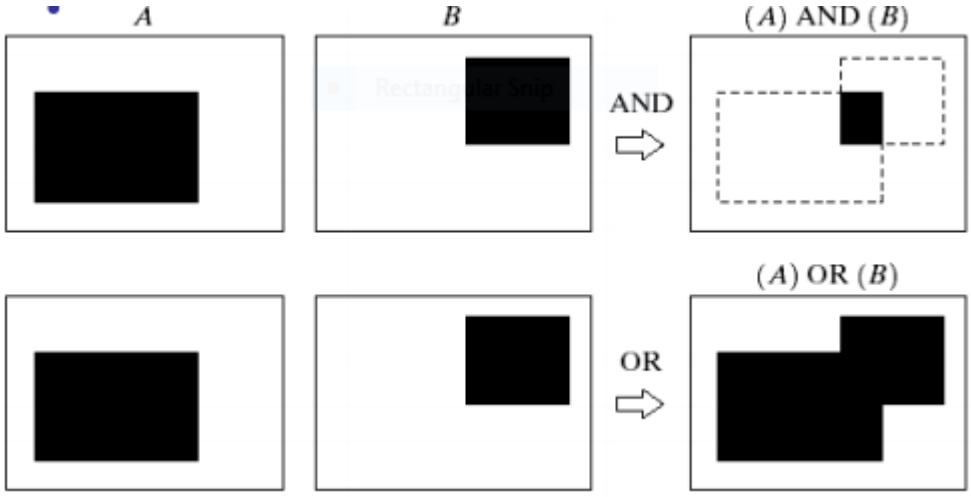


Structure Element(S)

2.2 Formulation

Morphological operators often take a binary image and a structuring element as input and combine them using a set operator (intersection, union, inclusion, complement).

Example:



1. **Dilation**

   Dilation adds pixels to the boundaries of objects in an image. The basic effect of the operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels (*i.e.* white pixels, typically). Thus areas of foreground pixels grow in size while holes within those regions become smaller.

   The dilation of an image *A* by a structuring element *s* (denoted $A \oplus B$) produces a new binary image $g = A \oplus B$ with ones in all locations (x,y) of a structuring element's origin at which that structuring element *B* hits the the input image *A*, i.e. $g(x,y) = 1$ if *B* hits *A* and 0 otherwise, repeating for all pixel coordinates (x,y).

$$A \oplus B = \{z | (B)_z \cap A \neq \Phi\}$$

.

2. Erosion

The basic effect of the operator on a binary image is to erode away the boundaries of regions of foreground pixels (*i.e.* white pixels, typically). Thus areas of foreground pixels shrink in size, and holes within those areas become larger. The value of the output pixel is the minimum value of all pixels in the neighborhood. In a binary image, a pixel is set to 0 if any of the neighboring pixels have the value 0.

The erosion of a binary image A by a structuring element B (denoted $A \ominus B$) produces a new binary image $g = A \ominus B$ with ones in all locations (x,y) of a structuring element's origin at which that structuring element B fits the input image A, i.e. $g(x,y) = 1$ is B fits A and 0 otherwise, repeating for all pixel coordinates (x,y).

$$A \ominus B = \{z | (B)_z \subseteq A\}$$

3. **Opening** :

The opening operation erodes an image and then dilates the eroded image, using the same structuring element for both operations. Morphological opening is useful for removing small objects from an image while preserving the shape and size of larger objects in the image.
Any regions that have survived the erosion are restored to their original size by the dilation

$$A \circ B = (A \ominus B) \oplus B$$

4. Closing

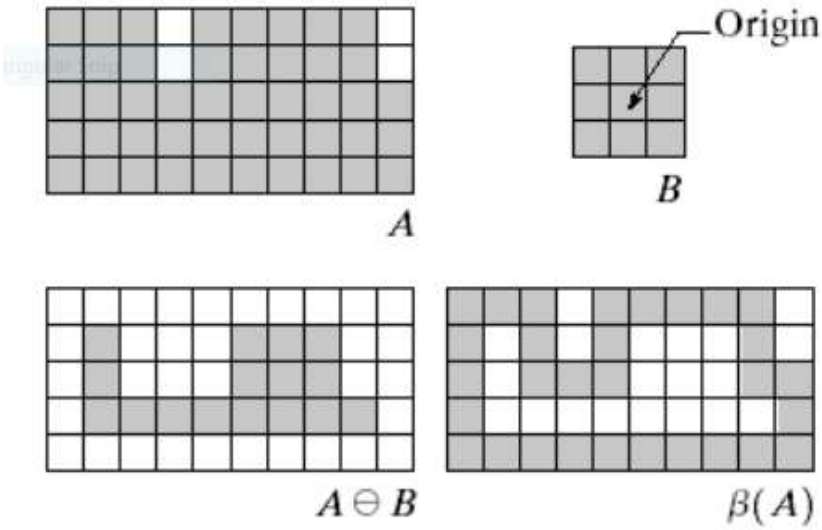
The closing operation dilates an image and then erodes the dilated image, using the same structuring element for both operations. Morphological closing is useful for filling small holes from an image while preserving the shape and size of the objects in the image. It fuses narrow breaks, thin gulfs and smoothen contours.

$$A \bullet B = (A \oplus B) \ominus B$$

5. Perimeter

This performs boundary extraction of image. In this first the image is eroded and then difference is taken between original image and eroded image.

$$\beta(A) = A - (A \ominus B)$$



Chapter 3

3.1 Algorithm

The structuring element with its origin at the center pixel is shifted over the image and at each pixel of the image its elements are compared with the set of the underlying pixels. If the two sets of elements match the condition defined by the set operator (*e.g.* if the set of pixels in the structuring element is a subset of the underlying image pixels), the pixel underneath the origin of the structuring element is set to a pre-defined value (0 or 1 for binary images). A morphological operator is therefore defined by its structuring element and the applied set operator.

1. **Dilation**

   The value of the output pixel is the maximum value of all pixels in the neighborhood. In a binary image, a pixel is set to 1 if any of the neighboring pixels have the value 1.

   In dilation:

   1. First we take image, structuring element and its dimension as input
   2. Convert input image into binary image.
   3. We define empty(Zero array) 2D array of dimensions

      New_row =image_row+struct_row-1

      New_col =image_col+struct_col-1

      We have done this as to keep the image quality as it is otherwise no of pixels will reduce, i.e. image is padded with zeroes.
   4. We copy whole image to New_arr .
   5. We run a loop over image over image and probe it with structuring element

      If any value of structuring element hits the original image, we set values of the Neighboring pixel to 1 ,0 otherwise.
   6. To do this first we take portion of new_arr equal to structuring element's dimensions and store it in variable 'k' (defined in code).
   7. 'result ' contains Boolean values of hit and miss in array of dimension equal to that of structuring element.

      result= (k==struct)
   8. If *any* of the value results in 'True' i.e. hit so we replace 0's to 1's in the neighborhood.
   9. Thus image boundary enlarges.

2. **Erosion:**.

In erosion

1. First we take image, structuring element and its dimension as input.
2. Convert input image into binary image.
3. We define empty(Zero array) 2D array of dimensions
    i. New_row =image_row+struct_row-1
    ii. New_col =image_col+struct_col-1

We have done this as to keep the image quality as it is otherwise no of pixels will reduce, i.e. image is padded with zeroes.

4. We copy whole image to New_arr .
5. We run a loop over image over image and probe it with structuring element.If any value of structuring element hits the original image, we set values of the Neighboring pixel to 1 ,0 otherwise.
6. To do this first we take portion of new_arr equal to structuring element's dimensions and store it in variable 'k' (defined in code).
7. 'result ' contains Boolean values of hit and miss in array of dimension equal to that of structuring element.
    i. result= (k==struct)
8. If *all* of the value results in 'True' i.e. fit then we leave it as it is otherwise replace with 0's in the neighborhood.
9. Thus image boundary shrinks.

3. Opening:

1. Here we first perform erosion on input image.
2. Output image from first operation (i.e. eroded) undergoes dilation.

4. **Closing:**

1. Input image is first dilated .
2. Output from the first operation is then eroded.

Chapter 4

Documentation of API

All methods are defined inside class ‘Morphology’ of ‘Morphology.py’.

Methods:

1. **square(dim, dtype=np.uint8)**
   Parameter
   dim- int, dimension of structure array
   dtype- user do not need to give this as input (*in any of the method*).

   Return
   ndarray- structuring array of the shape dim X dim

2. **rectangle(nrows, ncols)**
   Parameter
   nrows- int, numbers of rows or length of the structuring element
   ncols- int, width of structuring element

   Return
   ndarray- 2D -structuring array of shape nrows X ncols

3. **dilation(img ,shape,*args)**
   Parameter
   Img- input RGB image
   Shape- type of structuring element(square or rectangle)
   *args- dimension of shape(int)
   (as no of dimension values can differ according to shape of structuring element).

   Return
   img- dilated binary image

4. **erosion(img, shape, *args)**
   Parameter
   img- input RGB image

   shape- type of structuring element(square or rectangle)
   *args- required dimensions of shape

   Return
   Img- eroded binary image

5. **opening(img, shape,*args)**
   Parameter
   img- input RGB image
   shape- type of structuring element(square or rectangle)
   *args- required dimensions of shape

   Return
   Img- binary image after performing opening operation

6. **closing**(**img, shape,*args**)
   Parameter
   img- input RGB image
   shape- type of structuring element(square or rectangle)
   *args- required dimensions of shape

   Return
   Img- binary image after performing closing operation

7. **skeleton(img)**
   Parameter
   img- input RGB image

   Return
   Img- binary image(extracted boundaries of input image)

   Here, type of structuring element is hardwired in the program along with dimensions.

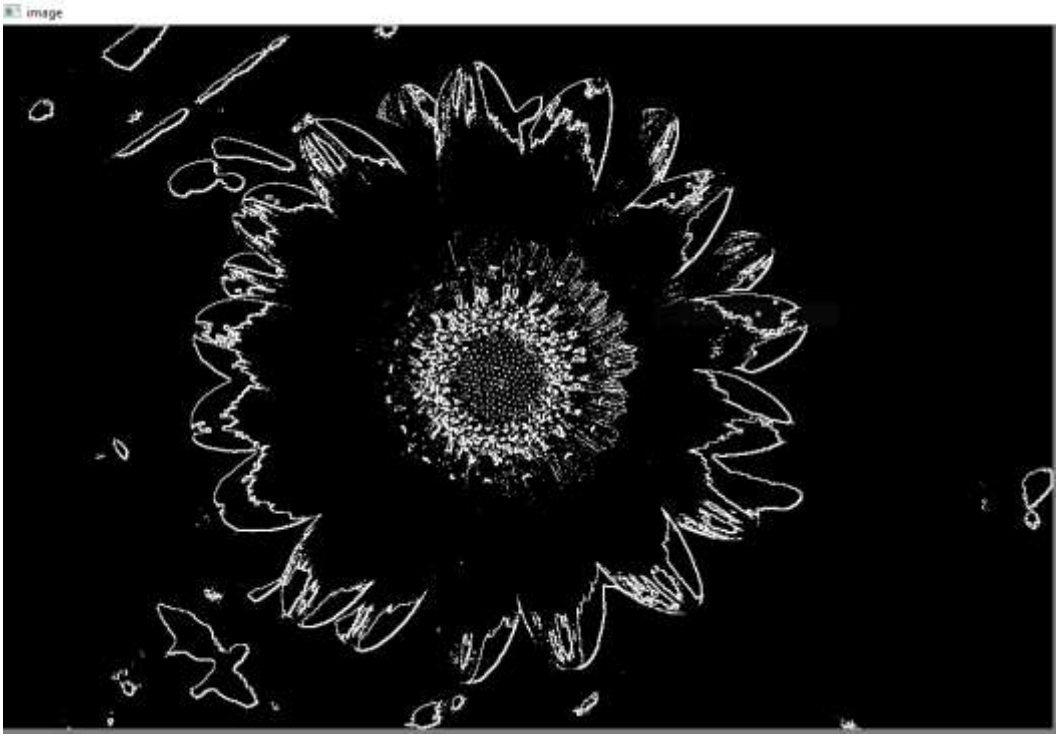
Chapter 5

Example

5.1 Example 1

```
#Making an object of class Morphology and importing necessary libraries.
Obj=Morphology()
#Reading image
 Img= cv.imread('E:\7th sem all material\Morphological\flower.jpg')
# displaying boundaries of object
  final_Img=obj.skeleton(img)
 cv.imshow('Final Image', final_img)
```

Output Image:



5.2 Example 2

Reading the flower image and applying dilation operation.

```
final_img=obj.dilation(img ,'square',3)    # write dimension type in small letters
cv.imshow('Final Image', final_img)
```

Output Image:

Input Image

Chapter 6

Learning Outcome

- This project gave me opportunity to go deep into image processing in python.
- Different formats of image representation and methods of conversion from one to other.
- Applying algorithm on images and how they affect structure of the image.
- Applying different morphological operation according to our need.
- Proper documentation and arrangement of code structure.

References

- http://www.researchmanuscripts.com/isociety2012/40.pdf
- http://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm
- https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm
- https://ieeexplore.ieee.org/document/536902