Morphology Fundamentals: Dilation and Erosion

|  |
| --- |
| **On this page…** |
| [Understanding Dilation and Erosion](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-24720)  [Understanding Structuring Elements](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-22581)  [Dilating an Image](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-20972)  [Eroding an Image](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-22016)  [Combining Dilation and Erosion](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-13322)  [Dilation- and Erosion-Based Functions](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-14379) |

To see how morphological processing can solve an image processing problem, view the Image Processing Toolbox™ watershed segmentation example.

**Understanding Dilation and Erosion**

*Morphology* is a broad set of image processing operations that process images based on shapes. Morphological operations apply a structuring element to an input image, creating an output image of the same size. In a morphological operation, the value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors. By choosing the size and shape of the neighborhood, you can construct a morphological operation that is sensitive to specific shapes in the input image.

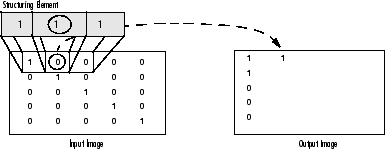
The most basic morphological operations are dilation and erosion. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the *structuring element* used to process the image. In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbors in the input image. The rule used to process the pixels defines the operation as a dilation or an erosion. This table lists the rules for both dilation and erosion.

**Rules for Dilation and Erosion**

| **Operation** | **Rule** |
| --- | --- |
| Dilation | The value of the output pixel is the *maximum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1. |
| Erosion | The value of the output pixel is the *minimum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0. |

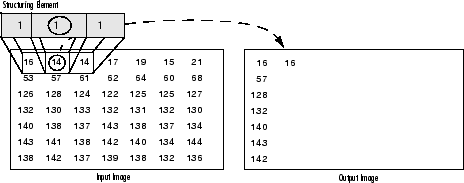
The following figure illustrates the dilation of a binary image. Note how the structuring element defines the neighborhood of the pixel of interest, which is circled. (See [Understanding Structuring Elements](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-22581) for more information.) The dilation function applies the appropriate rule to the pixels in the neighborhood and assigns a value to the corresponding pixel in the output image. In the figure, the morphological dilation function sets the value of the output pixel to 1 because one of the elements in the neighborhood defined by the structuring element is on.

**Morphological Dilation of a Binary Image**



The following figure illustrates this processing for a grayscale image. The figure shows the processing of a particular pixel in the input image. Note how the function applies the rule to the input pixel's neighborhood and uses the highest value of all the pixels in the neighborhood as the value of the corresponding pixel in the output image.

**Morphological Dilation of a Grayscale Image**



**Processing Pixels at Image Borders (Padding Behavior)**

Morphological functions position the origin of the structuring element, its center element, over the pixel of interest in the input image. For pixels at the edge of an image, parts of the neighborhood defined by the structuring element can extend past the border of the image.

To process border pixels, the morphological functions assign a value to these undefined pixels, as if the functions had padded the image with additional rows and columns. The value of these padding pixels varies for dilation and erosion operations. The following table describes the padding rules for dilation and erosion for both binary and grayscale images.

**Rules for Padding Images**

| **Operation** | **Rule** |
| --- | --- |
| Dilation | Pixels beyond the image border are assigned the minimum value afforded by the data type.  For binary images, these pixels are assumed to be set to 0. For grayscale images, the minimum value foruint8 images is 0. |
| Erosion | Pixels beyond the image border are assigned the *maximum* value afforded by the data type.  For binary images, these pixels are assumed to be set to 1. For grayscale images, the maximum value foruint8 images is 255. |

|  |
| --- |
| **Note**   By using the minimum value for dilation operations and the maximum value for erosion operations, the toolbox avoids *border effects*, where regions near the borders of the output image do not appear to be homogeneous with the rest of the image. For example, if erosion padded with a minimum value, eroding an image would result in a black border around the edge of the output image. |

**Understanding Structuring Elements**

An essential part of the dilation and erosion operations is the structuring element used to probe the input image. A structuring element is a matrix consisting of only 0's and 1's that can have any arbitrary shape and size. The pixels with values of 1 define the neighborhood.

Two-dimensional, or *flat*, structuring elements are typically much smaller than the image being processed. The center pixel of the structuring element, called the *origin*, identifies the pixel of interest -- the pixel being processed. The pixels in the structuring element containing 1's define the *neighborhood* of the structuring element. These pixels are also considered in dilation or erosion processing.

Three-dimensional, or *nonflat*, structuring elements use 0's and 1's to define the extent of the structuring element in the *x*- and *y*-planes and add height values to define the third dimension.

**The Origin of a Structuring Element**

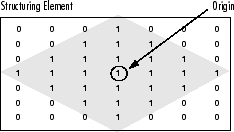
The morphological functions use this code to get the coordinates of the origin of structuring elements of any size and dimension.

origin = floor((size(nhood)+1)/2)

(In this code nhood is the neighborhood defining the structuring element. Because structuring elements are MATLAB® objects, you cannot use the size of the STREL object itself in this calculation. You must use the STREL getnhood method to retrieve the neighborhood of the structuring element from the STREL object. For information about other STREL object methods, see the [strel](http://www.mathworks.in/help/images/ref/strel.html) function reference page.)

For example, the following illustrates a diamond-shaped structuring element.

**Origin of a Diamond-Shaped Structuring Element**



**Creating a Structuring Element**

The toolbox dilation and erosion functions accept structuring element objects, called STRELs. You use the strel function to create STRELsof any arbitrary size and shape. The strel function also includes built-in support for many common shapes, such as lines, diamonds, disks, periodic lines, and balls.

|  |
| --- |
| **Note**   You typically choose a structuring element the same size and shape as the objects you want to process in the input image. For example, to find lines in an image, create a linear structuring element. |

For example, this code creates a flat, diamond-shaped structuring element.

se = strel('diamond',3)

se =

Flat STREL object containing 25 neighbors.

Decomposition: 3 STREL objects containing a total of 13 neighbors

Neighborhood:

0 0 0 1 0 0 0

0 0 1 1 1 0 0

0 1 1 1 1 1 0

1 1 1 1 1 1 1

0 1 1 1 1 1 0

0 0 1 1 1 0 0

0 0 0 1 0 0 0

**Structuring Element Decomposition**

To enhance performance, the strel function might break structuring elements into smaller pieces, a technique known as *structuring element decomposition*.

For example, dilation by an 11-by-11 square structuring element can be accomplished by dilating first with a 1-by-11 structuring element, and then with an 11-by-1 structuring element. This results in a theoretical speed improvement of a factor of 5.5, although in practice the actual speed improvement is somewhat less.

Structuring element decompositions used for the 'disk' and 'ball' shapes are approximations; all other decompositions are exact. Decomposition is not used with an arbitrary structuring element unless it is a flat structuring element whose neighborhood matrix is all 1's.

To view the sequence of structuring elements used in a decomposition, use the STREL getsequence method. The getsequence function returns an array of the structuring elements that form the decomposition. For example, here are the structuring elements created in the decomposition of a diamond-shaped structuring element.

sel = strel('diamond',4)

sel =

Flat STREL object containing 41 neighbors.

Decomposition: 3 STREL objects containing a total of 13 neighbors

Neighborhood:

0 0 0 0 1 0 0 0 0

0 0 0 1 1 1 0 0 0

0 0 1 1 1 1 1 0 0

0 1 1 1 1 1 1 1 0

1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 1 1 0

0 0 1 1 1 1 1 0 0

0 0 0 1 1 1 0 0 0

0 0 0 0 1 0 0 0 0

seq = getsequence(sel)

seq =

3x1 array of STREL objects

seq(1)

ans =

Flat STREL object containing 5 neighbors.

Neighborhood:

0 1 0

1 1 1

0 1 0

seq(2)

ans =

Flat STREL object containing 4 neighbors.

Neighborhood:

0 1 0

1 0 1

0 1 0

seq(3)

ans =

Flat STREL object containing 4 neighbors.

Neighborhood:

0 0 1 0 0

0 0 0 0 0

1 0 0 0 1

0 0 0 0 0

0 0 1 0 0

**Dilating an Image**

To dilate an image, use the imdilate function. The imdilate function accepts two primary arguments:

* The input image to be processed (grayscale, binary, or packed binary image)
* A structuring element object, returned by the strel function, or a binary matrix defining the neighborhood of a structuring element

imdilate also accepts two optional arguments: SHAPE and PACKOPT. The SHAPE argument affects the size of the output image. ThePACKOPT argument identifies the input image as packed binary. (Packing is a method of compressing binary images that can speed up the processing of the image. See the [bwpack](http://www.mathworks.in/help/images/ref/bwpack.html) reference page for information.)

This example dilates a simple binary image containing one rectangular object.

BW = zeros(9,10);

BW(4:6,4:7) = 1

BW =

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 1 0 0 0

0 0 0 1 1 1 1 0 0 0

0 0 0 1 1 1 1 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

To expand all sides of the foreground component, the example uses a 3-by-3 square structuring element object. (For more information about using the strel function, see [Understanding Structuring Elements](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-22581).)

SE = strel('square',3)

SE =

Flat STREL object containing 3 neighbors.

Neighborhood:

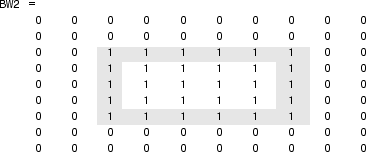
1 1 1

1 1 1

1 1 1

To dilate the image, pass the image BW and the structuring element SE to the imdilate function. Note how dilation adds a rank of 1's to all sides of the foreground object.

BW2 = imdilate(BW,SE)



**Eroding an Image**

To erode an image, use the imerode function. The imerode function accepts two primary arguments:

* The input image to be processed (grayscale, binary, or packed binary image)
* A structuring element object, returned by the strel function, or a binary matrix defining the neighborhood of a structuring element

imerode also accepts three optional arguments: SHAPE, PACKOPT, and M.

The SHAPE argument affects the size of the output image. The PACKOPT argument identifies the input image as packed binary. If the image is packed binary, M identifies the number of rows in the original image. (Packing is a method of compressing binary images that can speed up the processing of the image. See the [bwpack](http://www.mathworks.in/help/images/ref/bwpack.html) reference page for more information.)

The following example erodes the binary image circbw.tif:

1. Read the image into the MATLAB workspace.

BW1 = imread('circbw.tif');

1. Create a structuring element. The following code creates a diagonal structuring element object. (For more information about using thestrel function, see [Understanding Structuring Elements](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-22581).)
2. SE = strel('arbitrary',eye(5));
3. SE=
5. Flat STREL object containing 5 neighbors.
6. Neighborhood:
7. 1 0 0 0 0
8. 0 1 0 0 0
9. 0 0 1 0 0
10. 0 0 0 1 0

0 0 0 0 1

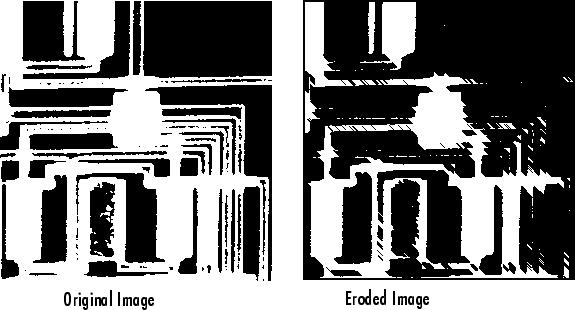
1. Call the imerode function, passing the image BW and the structuring element SE as arguments.

BW2 = imerode(BW1,SE);

Notice the diagonal streaks on the right side of the output image. These are due to the shape of the structuring element.

imshow(BW1)

figure, imshow(BW2)



**Combining Dilation and Erosion**

Dilation and erosion are often used in combination to implement image processing operations. For example, the definition of a morphological *opening*of an image is an erosion followed by a dilation, using the same structuring element for both operations. The related operation, morphological *closing* of an image, is the reverse: it consists of dilation followed by an erosion with the same structuring element.

The following section uses imdilate and imerode to illustrate how to implement a morphological opening. Note, however, that the toolbox already includes the imopen function, which performs this processing. The toolbox includes functions that perform many common morphological operations. See [Dilation- and Erosion-Based Functions](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-14379) for a complete list.

**Morphological Opening**

You can use morphological opening to remove small objects from an image while preserving the shape and size of larger objects in the image. For example, you can use the imopen function to remove all the circuit lines from the original circuit image, circbw.tif, creating an output image that contains only the rectangular shapes of the microchips.

To morphologically open the image, perform these steps:

1. Read the image into the MATLAB workspace.

BW1 = imread('circbw.tif');

1. Create a structuring element.

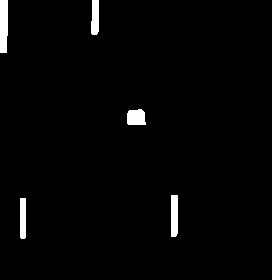
SE = strel('rectangle',[40 30]);

The structuring element should be large enough to remove the lines when you erode the image, but not large enough to remove the rectangles. It should consist of all 1's, so it removes everything but large contiguous patches of foreground pixels.

1. Erode the image with the structuring element.
2. BW2 = imerode(BW1,SE);

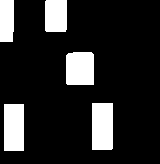
imshow(BW2)

This removes all the lines, but also shrinks the rectangles.



1. To restore the rectangles to their original sizes, dilate the eroded image using the same structuring element, SE.
2. BW3 = imdilate(BW2,SE);

imshow(BW3)



**Dilation- and Erosion-Based Functions**

This section describes two common image processing operations that are based on dilation and erosion:

* [Skeletonization](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-22448)
* [Perimeter determination](http://www.mathworks.in/help/images/morphology-fundamentals-dilation-and-erosion.html#f18-34991)

This table lists other functions in the toolbox that perform common morphological operations that are based on dilation and erosion. For more information about these functions, see their reference pages.

**Dilation- and Erosion-Based Functions**

| **Function** | **Morphological Definition** |
| --- | --- |
| [bwhitmiss](http://www.mathworks.in/help/images/ref/bwhitmiss.html) | Logical AND of an image, eroded with one structuring element, and the image's complement, eroded with a second structuring element. |
| [imbothat](http://www.mathworks.in/help/images/ref/imbothat.html) | Subtracts the original image from a morphologically closed version of the image. Can be used to find intensity troughs in an image. |
| [imclose](http://www.mathworks.in/help/images/ref/imclose.html) | Dilates an image and then erodes the dilated image using the same structuring element for both operations. |
| [imopen](http://www.mathworks.in/help/images/ref/imopen.html) | Erodes an image and then dilates the eroded image using the same structuring element for both operations. |
| [imtophat](http://www.mathworks.in/help/images/ref/imtophat.html) | Subtracts a morphologically opened image from the original image. Can be used to enhance contrast in an image. |

**Skeletonization**

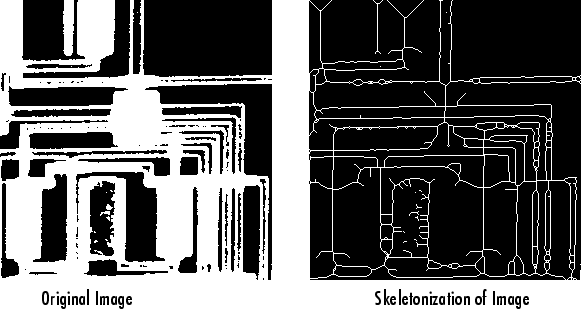
To reduce all objects in an image to lines, without changing the essential structure of the image, use the bwmorph function. This process is known as *skeletonization*.

BW1 = imread('circbw.tif');

BW2 = bwmorph(BW1,'skel',Inf);

imshow(BW1)

figure, imshow(BW2)



**Perimeter Determination**

The bwperim function determines the perimeter pixels of the objects in a binary image. A pixel is considered a perimeter pixel if it satisfies both of these criteria:

* The pixel is on.
* One (or more) of the pixels in its neighborhood is off.

For example, this code finds the perimeter pixels in a binary image of a circuit board.

BW1 = imread('circbw.tif');

BW2 = bwperim(BW1);

imshow(BW1)

figure, imshow(BW2)

