

Morphological Operations

Overview

- Morphology is a technique of image processing based on shapes. 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.
- Morphologic operations are especially suited to the processing of binary images and greyscale images.

Dilation and Erosion

- Dilation and erosion are two fundamental morphological operations.
- Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries.

Understanding Dilation and Erosion

- 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.

Rules for Dilation and Erosion

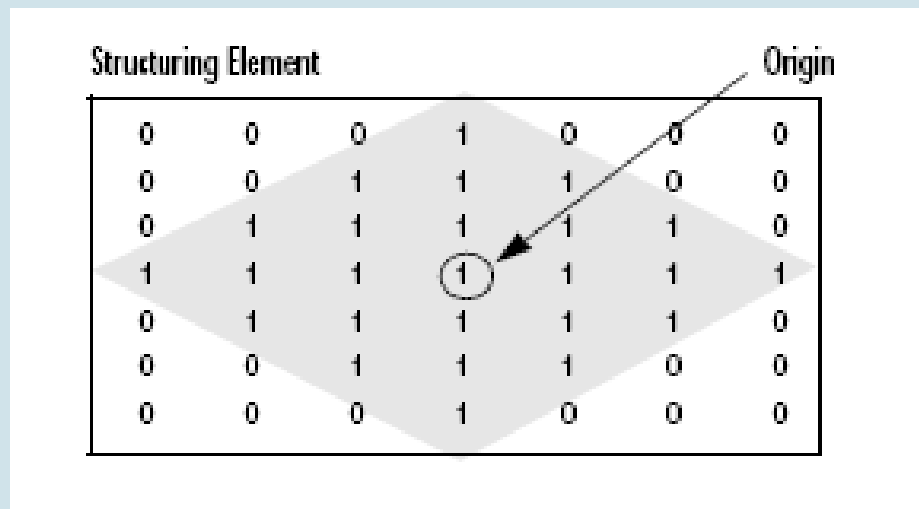
Operation	Rule
Dilation	The value of the output pixel is the maximum value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.
Erosion	The value of the output pixel is the minimum value of all the pixels in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

Structuring Elements

- An essential part of the dilation and erosion operations is the structuring element used to probe the input image.
- Two-dimensional, or flat, structuring elements consist of a matrix of 0's and 1's, typically much smaller than the image being processed.

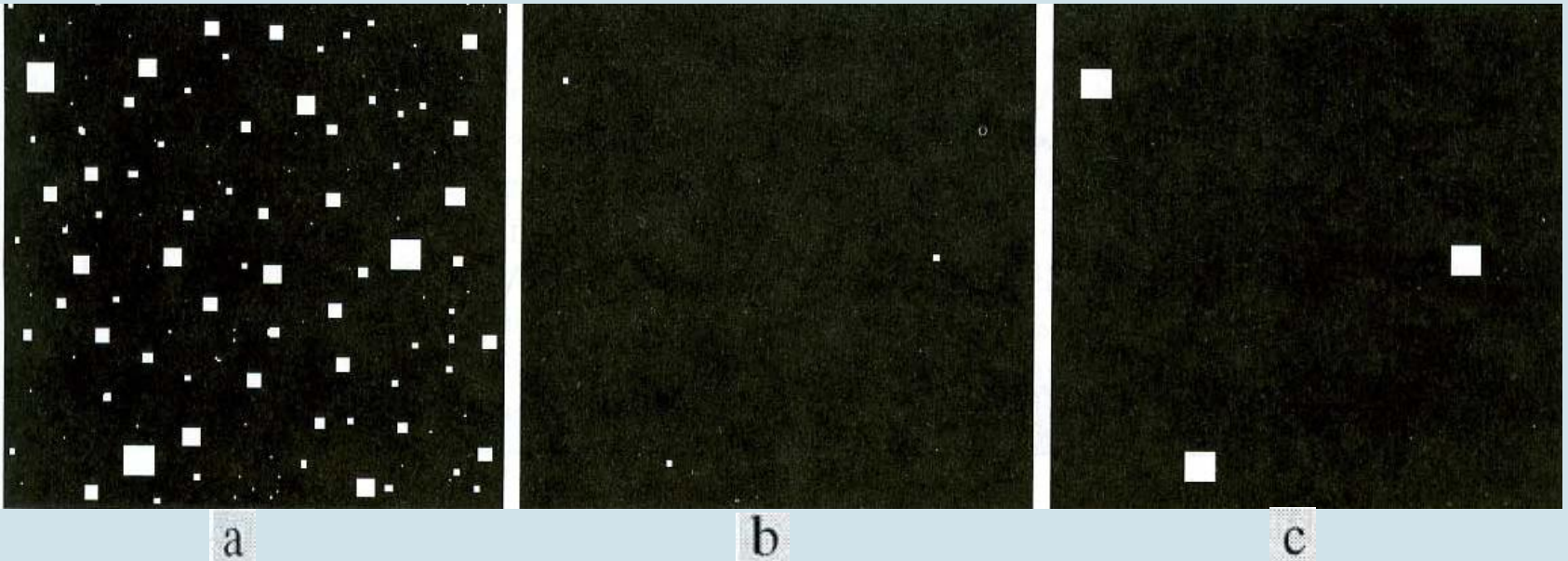
Structuring Elements

- 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.

Erosion and Dilation Example



- (a) images of squares of size 1,3,5,7,9,15 pixels on the side.
- (b) erosion of (a) with a structuring element of 1's,13 pixels on the side.
- (c) Dilation of (b) with the same structuring element.

Dilating an Image (Mathlab)

- 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: `PADOPT` and `PACKOPT`. The `PADOPT` argument affects the size of the output image. The `PACKOPT` argument identifies the input image as packed binary.

Dilating an Image (Mathlab)

- This example dilates a simple binary image containing one rectangular object.
- ```
BW = zeros(9,10);
BW(4:6,4:7) = 1
SE = strel('square',3) % the example uses a 3-by-3 square structuring element object.
BW2 = imdilate(BW,SE)
```

# Eroding an Image (Mathlab)

- 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: `PADOPT`, `PACKOPT`, and `M`.
  - ✓ The `PADOPT` 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.

# Eroding an Image (Mathlab)

- The following example erodes the binary image, circbw.tif:

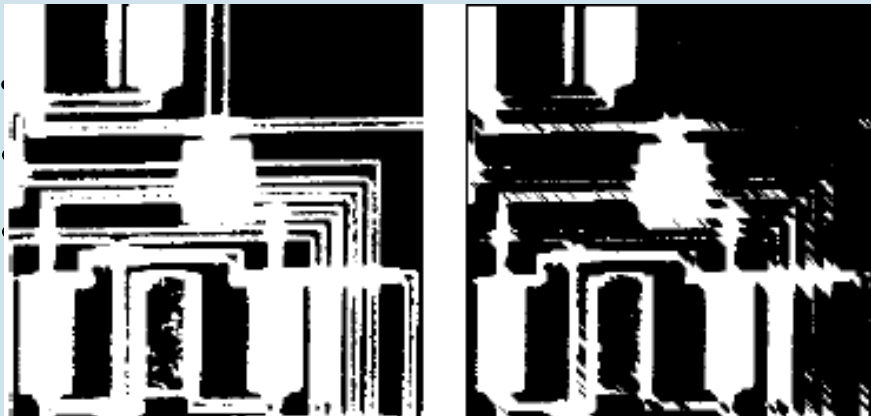
```
BW1 = imread('circbw.tif');
```

```
SE = strel('arbitrary',eye(5)); %creates a diagonal structuring element object
```

```
BW2 = imerode(BW1,SE);
```

```
imshow(BW1)
```

```
figure, imshow(BW2)
```



Circbw.tif Before and After Erosion  
with a Diagonal Structuring

# Combining Dilation and Erosion

- Morphological Opening

Opening is defined as an erosion, followed by a dilation.

- Morphological Closing

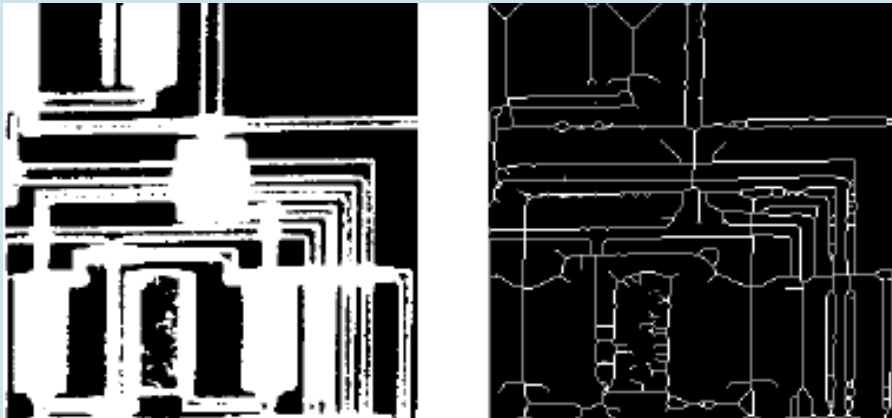
Closing is defined as a dilation, followed by an erosion.

# Dilation- and Erosion-Based Functions

- 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)
```



Circbw.tif Before and After
Skeletonization

Dilation- and Erosion-Based Functions

- 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)
```



Dilation- and Erosion-Based Functions

Function	Morphological Definition
bwhitmiss	Logical AND of an image, eroded with one structuring element, and the image's complement, eroded with a second structuring element
imbothat	Subtracts a morphologically closed image from the original image—can be used to find intensity troughs in a image
imclose	Dilates an image, and then erodes the dilated image using the same structuring element for both operations
imopen	Erodes an image and then dilates the eroded image using the same structuring element for both operations.
imtophat	Subtracts a morphologically opened image from the original image. Can be used to enhance contrast in a image.

Morphological reconstruction

- Morphological reconstruction is another major part of morphological image processing. Based on dilation, morphological reconstruction has these unique properties:
- Processing is based on two images, a marker and a mask, rather than one image and a structuring element
- Processing repeats until stability; i.e., the image no longer changes
- Processing is based on the concept of connectivity, rather than a structuring element

Morphological reconstruction

- Marker and Mask

Morphological reconstruction processes one image, called the marker, based on the characteristics of another image, called the mask. The high-points, or peaks, in the marker image specify where processing begins. The processing continues until the image values stop changing.

- Pixel Connectivity

Morphological processing starts at the peaks in the marker image and spreads throughout the rest of the image based on the connectivity of the pixels. Connectivity defines which pixels are connected to other pixels.

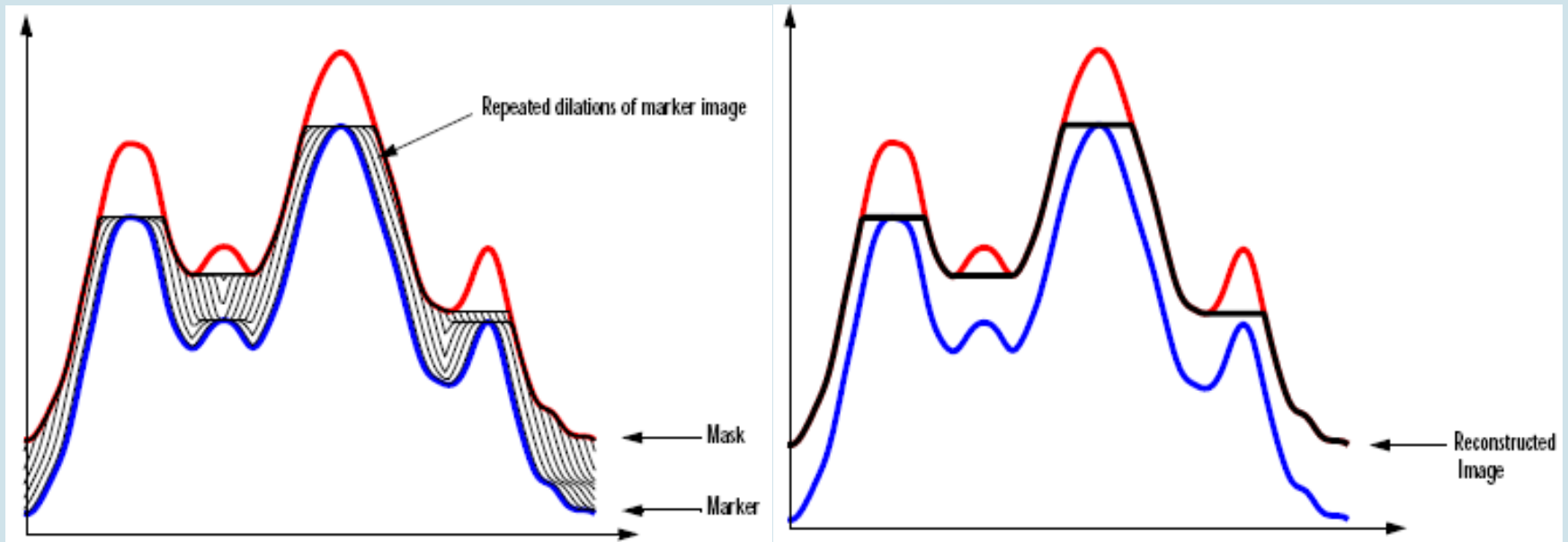
Morphological reconstruction

- “Flood-Fill Operations” – The imfill function performs a flood-fill operation on binary and grayscale images. For binary images, imfill changes connected background pixels (0s) to foreground pixels (1s), stopping when it reaches object boundaries. For grayscale images, imfill brings the intensity values of dark areas that are surrounded by lighter areas up to the same intensity level as surrounding pixels.
- “Finding Peaks and Valleys” – Grayscale images can be thought of in three-dimensions: the x- and y-axes represent pixel positions and the z-axis represents the intensity of each pixel. In this interpretation, the intensity values represent elevations as in a topographical map. The areas of high-intensity and low-intensity in an image, peaks and valleys in topographical terms, can be important morphological features because they often mark relevant image objects.

Understanding Morphological Reconstruction

- Morphological reconstruction can be thought of conceptually as repeated dilations of the marker image until the contour of the marker image fits under the mask image. In this way, the peaks in the marker image “spread out”, or dilate.
- The figure illustrates this processing in 1-D. Each successive dilation is constrained to lie underneath the mask. When further dilation ceases to change the image, processing stops. The final dilation is the reconstructed image. The figure shows the successive dilations of the marker.

Understanding Morphological Reconstruction



Repeated Dilations of Marker Image, Constrained by Mask

Distance Transform

- The distance transform provides a metric or measure of the separation of points in the image. The Image Processing Toolbox provides a function, `bwdist`, that calculates the distance between each pixel that is set to off (0) and the nearest nonzero pixel for binary images.
- The `bwdist` function supports several distance metrics

Distance Metric	Description	Illustration																																																		
Euclidean	The Euclidean distance is the straight-line distance between two pixels.	<div><div><table><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table><p>Image</p></div><div><table><tr><td>1.41</td><td>1.0</td><td>1.41</td></tr><tr><td>1.0</td><td>0.0</td><td>1.0</td></tr><tr><td>1.41</td><td>1.0</td><td>1.41</td></tr></table><p>Distance transform</p></div></div>	0	0	0	0	1	0	0	0	0	1.41	1.0	1.41	1.0	0.0	1.0	1.41	1.0	1.41																																
0	0	0																																																		
0	1	0																																																		
0	0	0																																																		
1.41	1.0	1.41																																																		
1.0	0.0	1.0																																																		
1.41	1.0	1.41																																																		
City Block	The City Block distance metric measures the path between the pixels based on a 4-connected neighborhood.	<div><div><table><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table><p>Image</p></div><div><table><tr><td>2</td><td>1</td><td>2</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>2</td><td>1</td><td>2</td></tr></table><p>Distance transform</p></div></div>	0	0	0	0	1	0	0	0	0	2	1	2	1	0	1	2	1	2																																
0	0	0																																																		
0	1	0																																																		
0	0	0																																																		
2	1	2																																																		
1	0	1																																																		
2	1	2																																																		
Chessboard	The Chessboard distance metric measures the path between the pixels based on an 8-connected neighborhood.	<div><div><table><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table><p>Image</p></div><div><table><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table><p>Distance transform</p></div></div>	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1																																
0	0	0																																																		
0	1	0																																																		
0	0	0																																																		
1	1	1																																																		
1	1	1																																																		
1	1	1																																																		
Quasi-Euclidean	The Quasi-Euclidean metric measures the total Euclidean Distance.	<div><div><table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table><p>Image</p></div><div><table><tr><td>2.8</td><td>2.2</td><td>2.0</td><td>2.2</td><td>2.8</td></tr><tr><td>2.2</td><td>1.4</td><td>1.0</td><td>1.4</td><td>2.2</td></tr><tr><td>2.0</td><td>1.0</td><td>0</td><td>1.0</td><td>2.0</td></tr><tr><td>2.2</td><td>1.4</td><td>1.0</td><td>1.4</td><td>2.2</td></tr><tr><td>2.8</td><td>2.2</td><td>2.0</td><td>2.2</td><td>2.8</td></tr></table><p>Distance transform</p></div></div>	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2.8	2.2	2.0	2.2	2.8	2.2	1.4	1.0	1.4	2.2	2.0	1.0	0	1.0	2.0	2.2	1.4	1.0	1.4	2.2	2.8	2.2	2.0	2.2	2.8
0	0	0	0	0																																																
0	0	0	0	0																																																
0	0	1	0	0																																																
0	0	0	0	0																																																
0	0	0	0	0																																																
2.8	2.2	2.0	2.2	2.8																																																
2.2	1.4	1.0	1.4	2.2																																																
2.0	1.0	0	1.0	2.0																																																
2.2	1.4	1.0	1.4	2.2																																																
2.8	2.2	2.0	2.2	2.8																																																

Distance Transform

- This example creates a binary image containing two overlapping circular objects.

- ```
center1 = -10;
center2 = -center1;
dist = sqrt(2*(2*center1)^2);
radius = dist/2 * 1.4;
lims = [floor(center1-1.2*radius) ceil(center2+1.2*radius)];
[x,y] = meshgrid(lims(1):lims(2));
bw1 = sqrt((x-center1).^2 + (y-center1).^2) <= radius;
bw2 = sqrt((x-center2).^2 + (y-center2).^2) <= radius;
bw = bw1 | bw2;
figure, imshow(bw), title('bw')----->
```



```
D = bwdist(~bw);
```

```
figure, imshow(D,[]), title('Distance transform of ~bw')----->
```





# Objects, Regions, and Feature Measurement

- The toolbox includes several functions that return information about the features in a binary image, including:
  - ✓ Connected-component labeling, and using the label matrix to get statistics about an image (bwlabel)
  - ✓ Selecting objects in an image (bwselect)
  - ✓ Finding the area of a binary image (bwarea)
  - ✓ Finding the Euler number of a binary image (bweuler)

# Lookup Table Operations

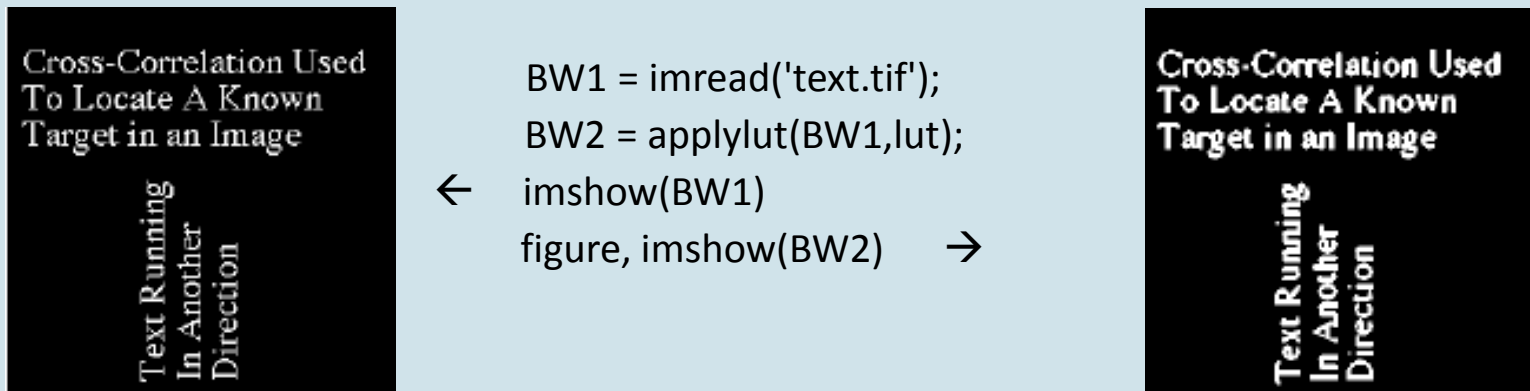
- Certain binary image operations can be implemented most easily through lookup tables. A lookup table is a column vector in which each element represents the value to return for one possible combination of pixels in a neighborhood.
- You can use the `makelut` function to create lookup tables for various operations. `makelut` creates lookup tables for 2-by-2 and 3-by-3 neighborhoods.

# Lookup Table Operations

- The example below illustrates using lookup-table operations to modify an image containing text. You begin by writing a function that returns 1 if three or more pixels in the 3-by-3 neighborhood are 1; otherwise, it returns 0. You then call `makelut`, passing in this function as the first argument, and using the second argument to specify a 3-by-3 lookup table.

```
f = inline('sum(x(:)) >= 3');
lut = makelut(f,3);
```

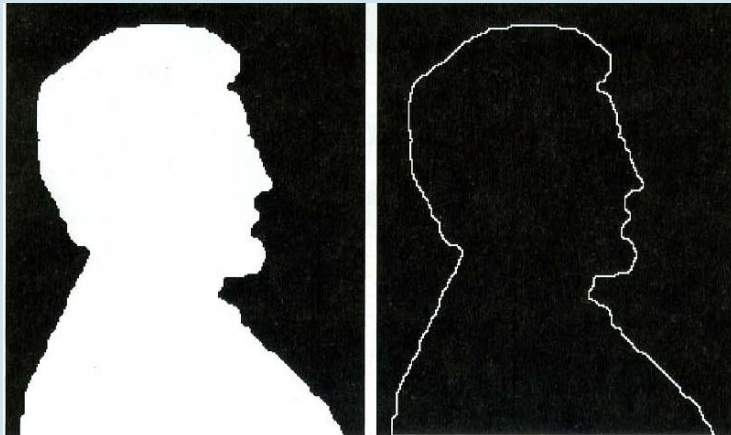
`lut` is returned as a 512-element vector of 1's and 0's. Each value is the output from the function for one of the 512 possible permutations.



# Some basic Morphological Algorithms

- Boundary Extraction

The boundary of set  $A$  can be obtained by first eroding  $A$  by  $B$  and then performing the set difference between  $A$  and its erosion.



# Some basic Morphological Algorithms

- Region Filling
- Extraction of Connected Components
- Convex Hull
- Thinning
- Thickening
- Pruning
- The Hit-or-Miss Transformation