✓Morphological Filtering

- *Morphology* relates to form and structure of objects
- Morphological filtering simplifies a segmented image to facilitate the search for objects of interest
- This is done by smoothing out object outlines, filling small holes, eliminating small projections, and with other similar techniques

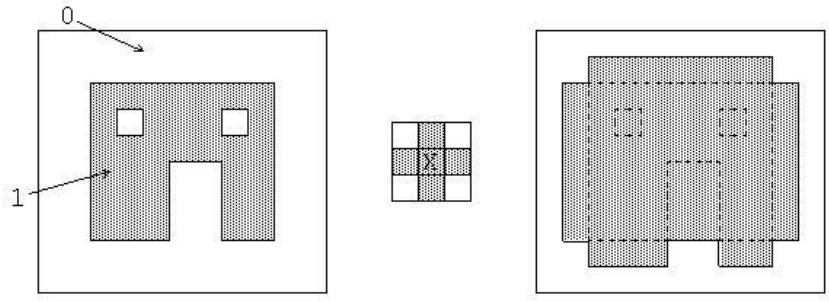
- The two principal morphological operations are **dilation** and **erosion**
- *Dilation* allows objects to expand, thus potent-ially filling in small holes and connecting disjoint objects
- *Erosion* shrinks objects by etching away (eroding) their boundaries

- These operations can be customized for an application by the proper selection of the *structuring element*, which determines exactly how the objects will be dilated or eroded
- Basically, the structuring element is used to probe the image to find how it will fit, or not fit, into the image object(s)

- *Dilation:* It is performed by laying the structuring element on the image and sliding it across the image in a manner similar to convolution

1. If the origin of the structuring element coincides with a '0' in the image, there is no change; move to the next pixel
2. If the origin of the structuring element coincides with a '1' in the image, perform the OR logic operation on all pixels within the structuring element

Figure 4.3-14: DILATION



a) Original image

b) Structuring element; x = origin

c) Image after dilation; original in dashes

- With a dilation operation, all the '1' pixels in the original image will be retained, any boundaries will be expanded, and small holes will be filled

EXAMPLE 4.3.1:

Given the following image and structuring element, perform a dilation operation. We assume the origin of the structuring element is in the center and ignore cases where the structuring element extends beyond the image. Note that since the holes are all smaller than the structuring element, they are all filled.

STRUCTURING ELEMENT

$$\begin{bmatrix} 1 & 0 & \mathbf{0} \\ 1 & 1 & 1 \\ 1 & 0 & \mathbf{0} \end{bmatrix}$$

- *Note: In 1st printing of book, the structuring element is incorrect*

IMAGE

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

RESULT

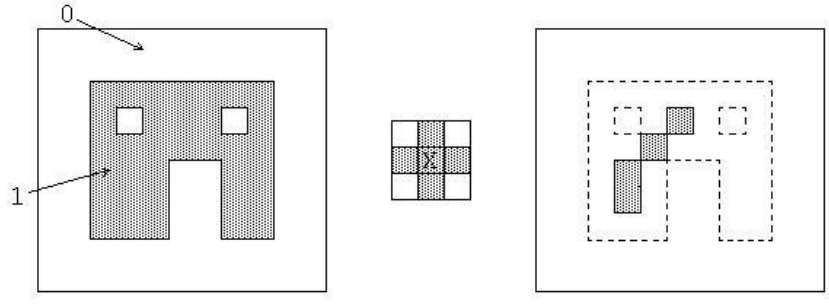
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \mathbf{1} & \mathbf{1} & 1 & \mathbf{1} & \mathbf{1} & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \mathbf{1} & \mathbf{1} & 1 & 1 & 1 \end{bmatrix}$$

- *Erosion:* It is similar to dilation, but we turn pixels to '0', not '1'

1. If the origin of the structuring element coincides with a '0' in the image, there is no change; move to the next pixel

2. If the origin of the structuring element coincides with a '1' in the image, and any of the '1' pixels in the structuring element extend beyond the object ('1' pixels) in the image, then change the '1' pixel in the image, whose location corresponds to the origin of the structuring element, to a '0'

Figure 4.3-15: EROSION



a) Original image

b) Structuring element; x = origin

c) Image after erosion; original in dashes

➢ With an erosion operation, the only remaining pixels are those that coincide to the origin of the structuring element where it is all contained in the object

EXAMPLE 4.3.2:

Given the following image and structuring element, perform an erosion operation. We assume the origin of the structuring element is in the center and ignore cases where the structuring element extends beyond the image. Note the only 1's left inside the image mark places where the shape of the structuring element exists in the image.

STRUCTURING ELEMENT

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

IMAGE

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & \mathbf{0} & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

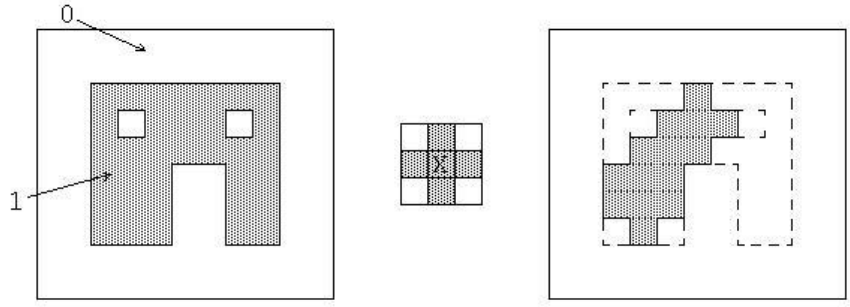
Note: In first printing of book, the 4th row, 4th col '0', should be a '1', in the **IMAGE**

RESULT

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & \mathbf{0} & 0 & 0 & 1 \\ 1 & \mathbf{1} & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & 1 \\ 1 & 1 & 1 & \mathbf{0} & \mathbf{0} & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

- *Opening*: It consists of an erosion followed by a dilation
  - It can be used to eliminate all pixels in regions that are too small to contain the structuring element
  - In this case the structuring element is often called a *probe*, as it is probing the image looking for small objects to filter out of the image

Figure 4.3-16: OPENING



a) Original image

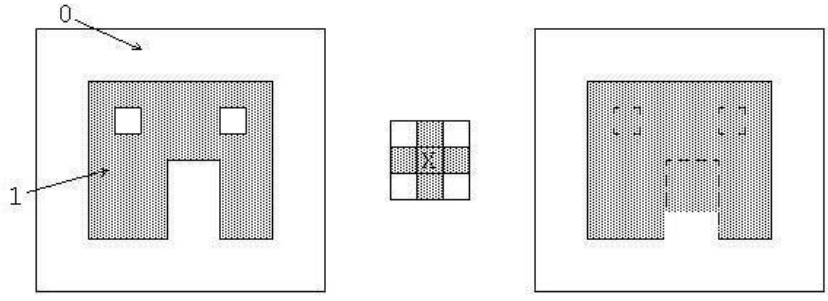
b) Structuring element; x = origin

c) Image after opening = erosion followed by dilation

- The output image tends to take a shape similar to the structuring element itself

- *Closing*: It consists of a dilation followed by erosion
  - It can be used to fill in holes and small gaps
  - It will connect small, adjacent objects
  - Closing tends to “close up” or “fill in” objects

Figure 4.3-17: CLOSING



a) Original image

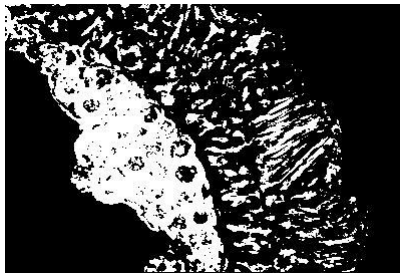
b) Structuring element; x = origin

c) Image after closing = dilation followed by erosion; original in dashes.

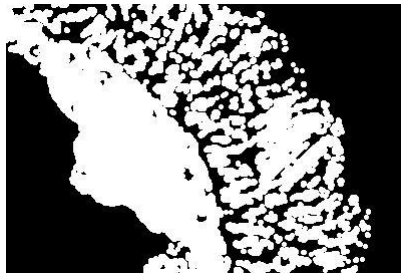
- Note that holes and gaps are filled, but, unlike dilation, more of the original boundary is retained

- Closing and opening will have different results even though both consist of an erosion and a dilation
- Therefore, order of operation is important for morphological operations
- Different structuring elements will also provide different results. As noted before, objects in the output image will tend to take the shape of the structuring element

Figure 4.3-18: Binary Dilation with Various Shape Structuring Elements

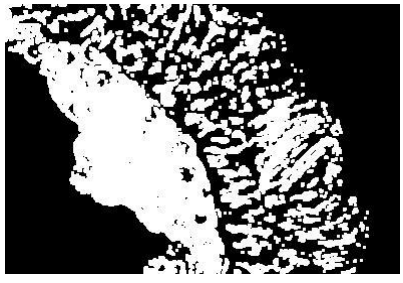


a) Original image, a microscope cell image that has undergone a threshold operation (original image courtesy of Sara Sawyer, SIUE)

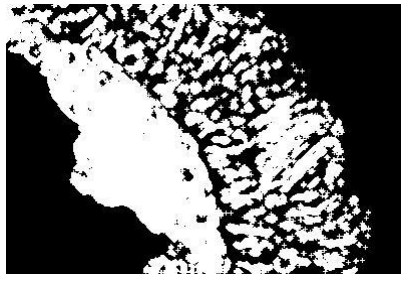


b) Dilation with a circular structuring element

Figure 4.3-18: Binary Dilation with Various Shape Structuring Elements (contd)

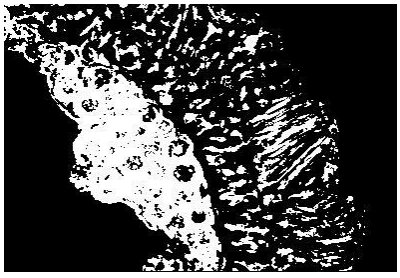


c) Dilation with a square structuring element

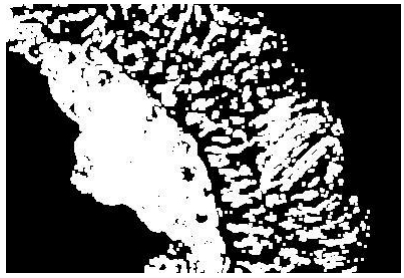


d) Dilation with a cross shape structuring element

Figure 4.3-19: Dilation with Different Size Structuring Elements

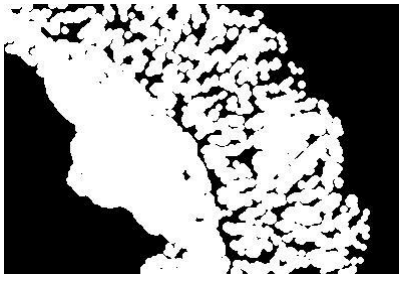


a) Original image

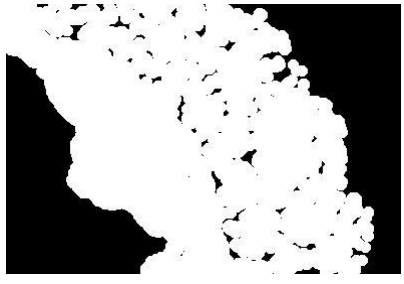


b) Dilation with a circular structuring element of size 3

Figure 4.3-19: Dilation with Different Size Structuring Elements (contd)

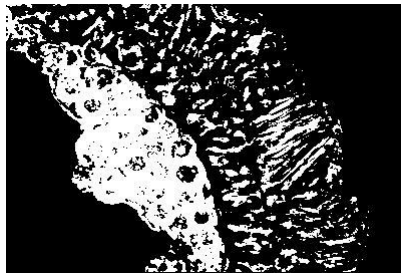


c) Dilation with a circular structuring element of size 7

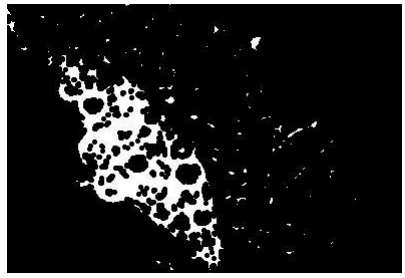


d) Dilation with a circular structuring element of size 11

Figure 4.3-20: Binary Erosion with Various Shape Structuring Elements

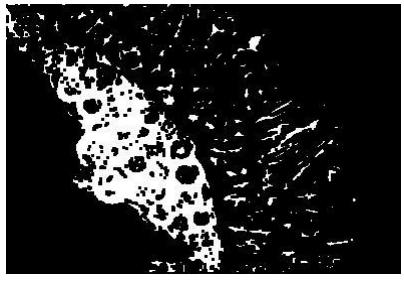


a) Original image, a microscope cell image that has undergone a threshold operation (original image courtesy of Sara Sawyer, SIUE),

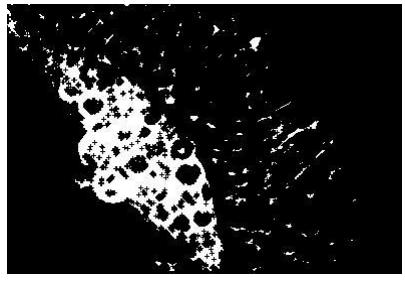


b) Erosion with a circular structuring element

Figure 4.3-20: Binary Erosion with Various Shape Structuring Elements (contd)

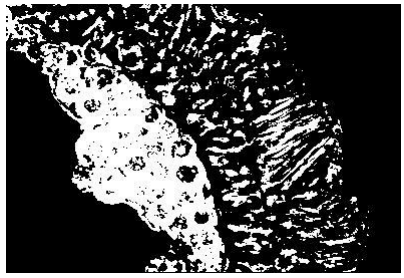


c) Erosion with a square structuring element

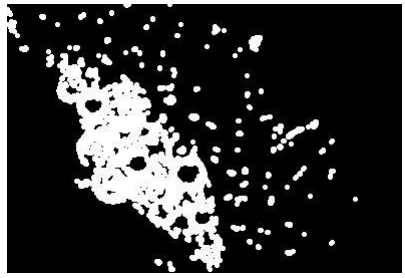


d) Erosion with a cross shape structuring element

Figure 4.3-21: Binary Opening with Various Shape Structuring Elements

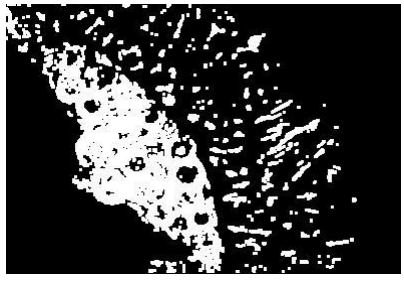


a) Original image, a microscope cell image that has undergone a threshold operation (original image courtesy of Sara Sawyer, SIUE),

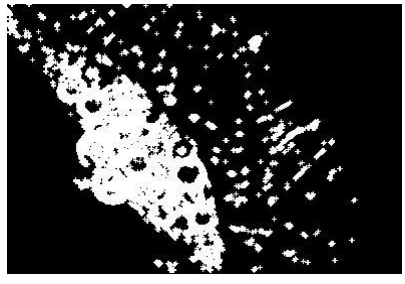


b) Opening with a circular structuring element

Figure 4.3-21: Binary Opening with Various Shape Structuring Elements (contd)

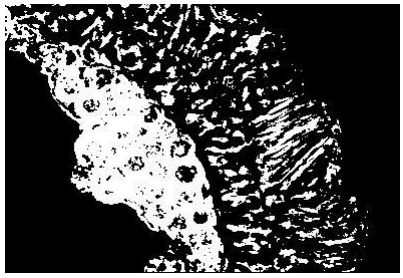


c) Opening with a square structuring element

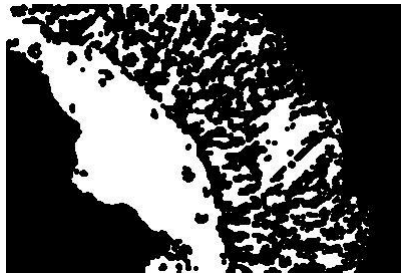


d) Opening with a cross shape structuring element

Figure 4.3-22: Binary Closing with Various Shape Structuring Elements

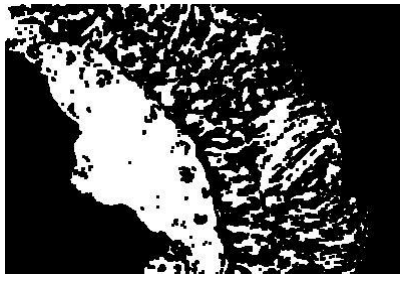


a) Original image, a microscope cell image that has undergone a threshold operation (original image courtesy of Sara Sawyer, SIUE),

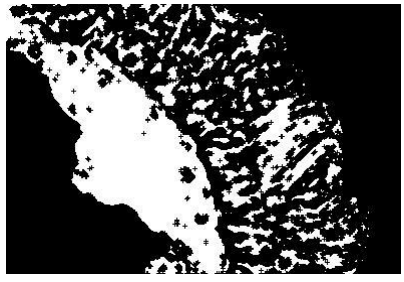


b) Closing with a circular structuring element

Figure 4.3-22: Binary Closing with Various Shape Structuring Elements (contd)

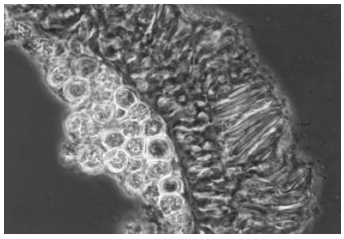


c) Closing with a square structuring element

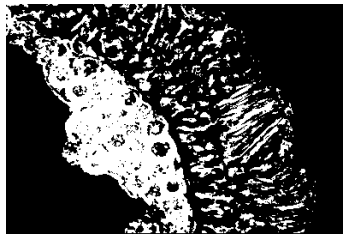


d) Closing with a cross shape structuring element

Figure 4.3-23: Opening and Closing with Different Size Structuring Elements

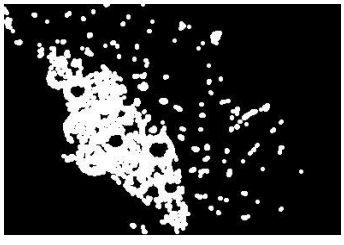


a) Original microscopic cell image (courtesy of Sara Sawyer, SIUE)

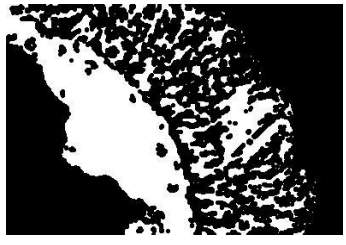


b) Image after undergoing a threshold operation

Figure 4.3-23: Opening and Closing with Different Size Structuring Elements (contd)

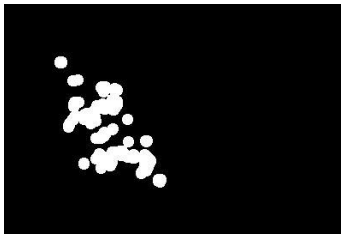


c) Opening with a circular structuring element of size 5

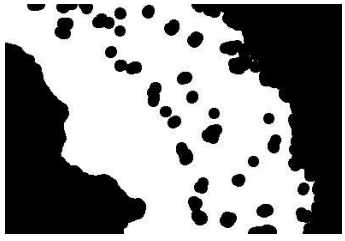


d) Closing with a circular structuring element of size 5

Figure 4.3-23: Opening and Closing with Different Size Structuring Elements (contd)



c) Opening with a circular structuring element of size 13



d) Closing with a circular structuring element of size 13

Hit-or-Miss Transform

- Fundamental method for detection of simple shapes
- Uses a structuring element to find patterns or shapes
- Structuring element may contain “don’t cares”, “x”
- Requires an exact match for a “hit” to occur

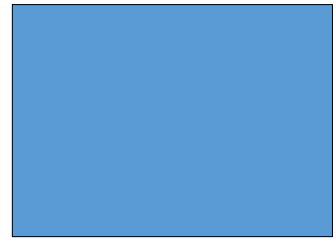
EXAMPLE: Finding upper-right corner points with the hit-or-miss transform

The structuring element $\begin{bmatrix} x & 0 & 0 \\ 1 & 1 & 0 \\ x & 1 & x \end{bmatrix}$

The Image

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Hit-or-miss result



- Note that there is only one upper right corner to this object.

EXAMPLE: Finding all corner points with the hit-or-miss transform

The structuring elements

$$\begin{bmatrix} x & 0 & 0 \\ 1 & 1 & 0 \\ x & 1 & x \end{bmatrix} \begin{bmatrix} 0 & 0 & x \\ 0 & 1 & 1 \\ x & 1 & x \end{bmatrix}, \begin{bmatrix} x & 1 & x \\ 1 & 1 & 0 \\ x & 0 & 0 \end{bmatrix}, \begin{bmatrix} x & 1 & x \\ 0 & 1 & 1 \\ 0 & 0 & x \end{bmatrix}$$

The Image

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

OR of hit-or-miss output from each structuring element

- Note that all corner points have been detected.

Skeletonization

- ***Skeleton****:* what is left of an object when it has been eroded until it is one pixel wide
- Used in optical character recognition (OCR) and in many other applications
- First define the ***thinning*** operation, with a given structuring element, *SE*:

Thin[I(r,c),SE] = I(r,c) – hit-or-miss[I(r,c),SE]

- Subtraction is the logical subtraction defined by:

$$A - B = (A)AND\ (NOT\ B), where\ the\ AND\ and\ NOT\ are\ logical\ operators$$

EXAMPLE: Thinning

The structuring element for top horizontal line: $SE_1 = \begin{bmatrix} 0 & 0 & 0 \\ x & 1 & x \\ 1 & 1 & 1 \end{bmatrix}$

The Image = I(r,c)

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Thin[I(r,c),SE_1] =
I(r,c) – hit-or-miss[I(r,c),SE_1]

hit-or-miss[I(r,c), SE_1]

0	0	0	0	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	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	0	0	0	0	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Skeletonization (cont')

- Skeletonization: thinning with each of the line structuring elements, and then logical AND of the thinned results

The other three line structuring elements

$$SE_2 = \begin{bmatrix} 1 & x & 0 \\ 1 & 1 & 0 \\ 1 & x & 0 \end{bmatrix}, SE_3 = \begin{bmatrix} 1 & 1 & 1 \\ x & 1 & x \\ 0 & 0 & 0 \end{bmatrix}, SE_4 = \begin{bmatrix} 0 & x & 1 \\ 0 & 1 & 1 \\ 0 & x & 1 \end{bmatrix}$$

After thinning with each of the four structuring elements and the logical AND of the results, we obtain:

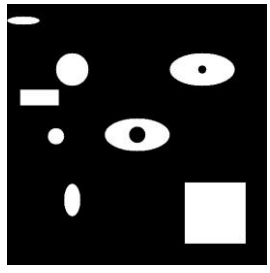
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

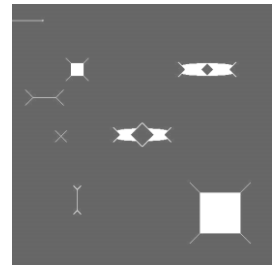
After one more iteration for the skeletonization process we obtain:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Skeletonization with simple shapes

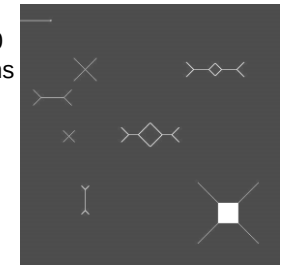
Original image

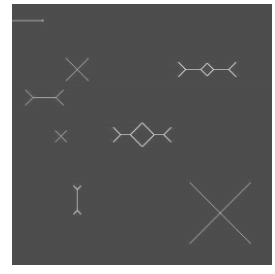




after 20 iterations

after 40 iterations





after 60 iterations

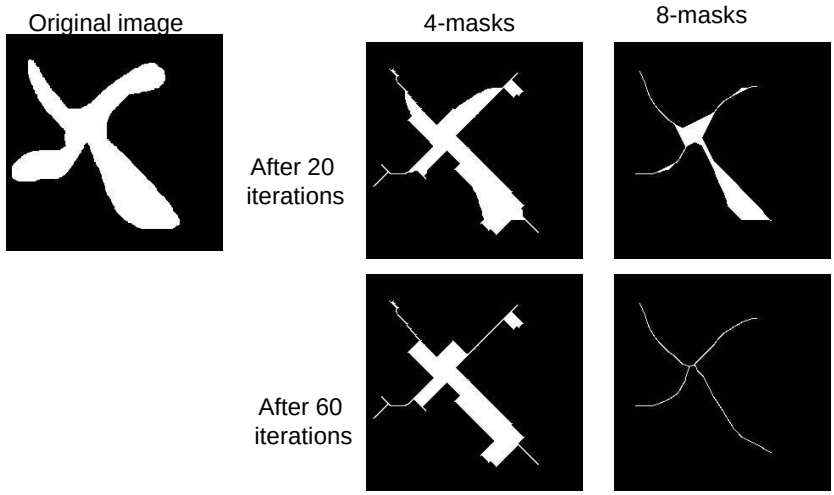
Skeletonization (cont')

- Irregularly-shaped objects, the process is more complex
- Four more structuring elements required:

$$SE_5 = \begin{bmatrix} x & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & x \end{bmatrix}, SE_6 = \begin{bmatrix} 1 & 1 & x \\ 1 & 1 & 0 \\ x & 0 & 0 \end{bmatrix}, SE_7 = \begin{bmatrix} x & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & x \end{bmatrix}, SE_8 = \begin{bmatrix} x & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & x \end{bmatrix}$$

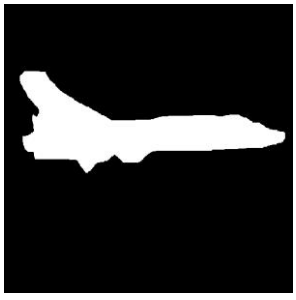
- The previous four, the horizontal and vertical (called the *4-masks*), add the four diagonal masks above to create the *8-masks*.

Skeletonization with 4 or 8 masks

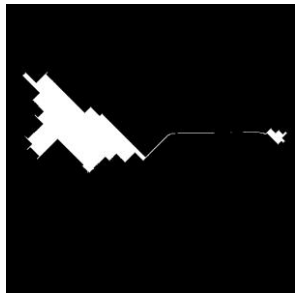


Skeletonization (cont')

- ANDing the images from all the masks after thinning, can result in loss of connectivity



Original image

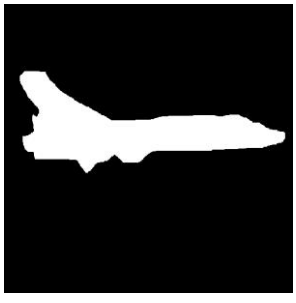


4-masks and AND method

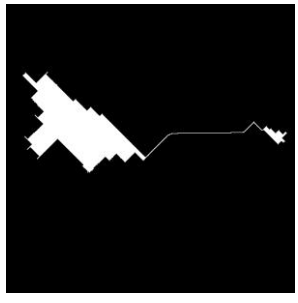
8-masks and AND method

Skeletonization (cont')

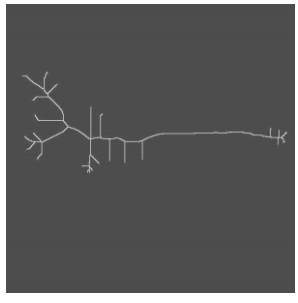
- To avoid connectivity loss use output image after thinning with one mask as input to the thinning with the next – sequential method



Original image



4-masks and Seq. method

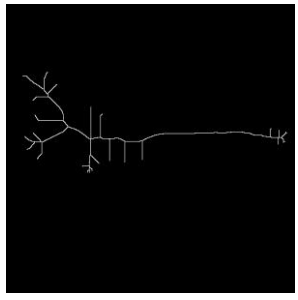


8-masks and Seq. method

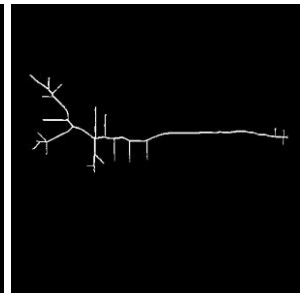
- No connectivity loss, but spurious lines appear.

Pruning: removal of spurs

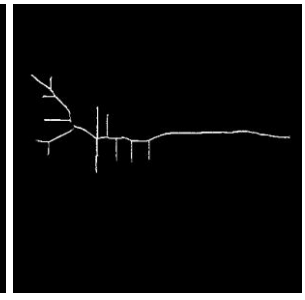
- Hough transform can be used to remove spurious lines



8-masks and Seq. method result



Hough transform to remove spurs min no. of segment pixels = 12



Hough transform to remove spurs min no. of segment pixels = 20

Pruning: removal of spurs

- Standard pruning operation is a form of thinning with a single pixel as the structuring element.

- Structuring element is rotated throughout the 8 possible compass directions to prune lines in all directions

- Typically pruning is only performed for a small number of iterations to remove small spurs, or all lines except for closed loops will be removed

EXAMPLE: Standard Pruning

The structuring elements

$$SE_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & x & x \end{bmatrix} SE_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ x & x & 0 \end{bmatrix}, SE_3 = \begin{bmatrix} 0 & 0 & 0 \\ x & 1 & 0 \\ x & 0 & 0 \end{bmatrix}, SE_4 = \begin{bmatrix} x & 0 & 0 \\ x & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$SE_5 = \begin{bmatrix} x & x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} SE_6 = \begin{bmatrix} 0 & x & x \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, SE_7 = \begin{bmatrix} 0 & 0 & x \\ 0 & 1 & x \\ 0 & 0 & 0 \end{bmatrix}, SE_8 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & x \\ 0 & 0 & x \end{bmatrix}$$

Image from previous skeletonization

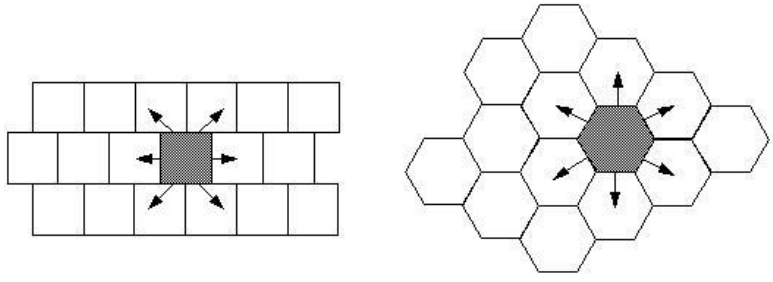
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

After one pruning iteration

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Another approach to binary morphological filtering is based on an iterative approach
- The usefulness of this approach lies in its flexibility
- It is based on a definition of six-connectivity
- This is equivalent to assuming the pixels are laid out on a hexagonal grid

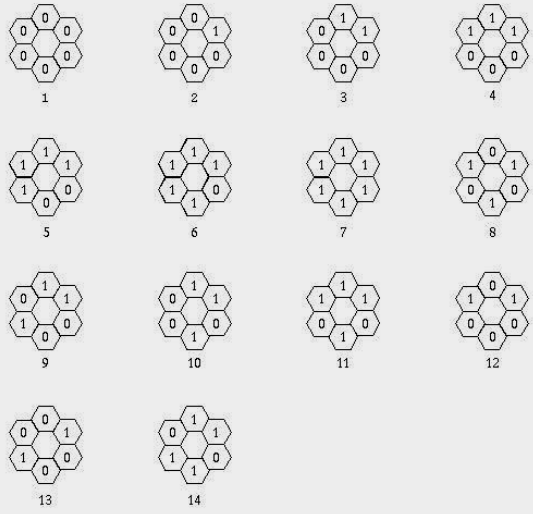
HEXAGONAL GRID



a) Rectangular image grid with every other row shifted by one-half pixel

b) Hexagonal grid

SURROUNDS FOR ITERATIVE MORPHOLOGICAL FILTERING



- **With six–connectivity, a pixel can be surrounded by 14 possible** combinations of 1's and 0's

- For this approach to morphological filtering, we define:

1. The set of surrounds *S*, where *a* = 1
2. A logic function, *L(a,b)*, where *b* is the current pixel value, and the function specifies the output of the morphological operation
3. The number of iterations, *n*

- The function *L(a,b)*, and the values of *a* and *b* are all functions of the row and column, *(r,c)*
- Set *S* can contain any or all of the 14 surrounds
- *L(a,b)* can be any logic function, but it turns out that the most useful are the AND function, which tends to etch away object boundaries (erosion), and the OR function, which tends to grow objects (dilation)

EXAMPLE

Let $L(a,b) = ab$ (logical AND operation).

IMAGE
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$S = \{5\} = \begin{bmatrix} 1 & 1 & x \\ 1 & x & 1 \\ x & 0 & 0 \end{bmatrix}$; assume the origin is in the center

Notes: this means the set S contains surround number 5 from Figure 4.3-25 and the x's are not neighbors, since we are using six-connectivity

The window S (a 3 X 3 window) is scanned across the image. If a match is found, then $a = 1$ and the output is computed by performing the specified $L(a,b)$ function, in this case by ANDing a with b (b is the center pixel of the subimage under the window). This gives the value of our new image, which will equal $ab = (1)b = b$. If the window S does not match the underlying subimage, then $a = 0$ (false) and $L(a,b) = ab = (0)b = 0$. In either case, the resulting value is written to the new image at the location corresponding to the center of the window

EXAMPLE (contd):

The window S is scanned across the entire image in this manner and the resultant image is as follows:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Here we see that the AND operation erodes the object. Also note that the set S can contain more than one surround; if it does, then $a = 1$ when the underlying neighborhood matches *any* of the surrounds in the set S. Another parameter which can be considered is the rotation of the surrounds in S. For example, rotating surround $S=\{5\}$ counterclockwise we have the following 5 possibilities:

$$\begin{bmatrix} 1 & 1 & x \\ 1 & x & 1 \\ x & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & x \\ 1 & x & 0 \\ x & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & x \\ 1 & x & 0 \\ x & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & x \\ 1 & x & 1 \\ x & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & x \\ 0 & x & 1 \\ x & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & x \\ 0 & x & 1 \\ x & 0 & 1 \end{bmatrix}$$

With iterative morphological filtering, normally it is implied that the surrounds in S can be rotated when looking for a match. Additionally, since this is an iterative approach, n is used to define the number of iterations

EXAMPLE

$S = \{\}$, $L(a,b) = 0, n = 1$

The set of surrounds (neighbors) is a null set. This implies $a = 0$; since a surround is not specified. The boolean function $L(a,b) = 0$. For this combination, all the cells of the image are set to zero, i.e., we have a black image as output

EXAMPLE

$S = \{\}$, $L(a,b) = (!b)$, $n = 1$

In this case $a = 0$, but this is irrelevant since $L(a,b) = !b$, which implies that the center pixel is negated (complimented).

If b = 1, L(a,b))((!1) = 0 ;

Elseif b = 0, L(a,b) = (!0) = 1

EXAMPLE

$S = \{7\}$, $L(a,b) = ab$, $n = 1$

Consider the following image with the surround S as follows:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Let $S = \{7\} = \begin{bmatrix} 1 & 1 & x \\ 1 & x & 1 \\ x & 1 & 1 \end{bmatrix}$

EXAMPLE (contd):

In this case, $a = 1$ for the surround shown above. If the surround does not match then $L(a,b) = 0(b) = 0$. If there is a match then $L(a,b) = 1(b) = b$. The resultant image is as follows:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Since the logic function is a logical AND operation, if the edge pixels are not "1's", the edges are removed. This operation retains a cluster of "1's" with the edge pixels removed. So, the appendages (thin lines) are removed from the original image – this is an erosion operation

EXAMPLE

$S = \{1,7\}$, $L(a,b) = (!a)b$, $n = 1$

Consider the following image with the surrounds {S} as follows:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Let S = {1,7}, that is $S = \left\{ \begin{bmatrix} 0 & 0 & d \\ 0 & d & 0 \\ d & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & d \\ 1 & d & 1 \\ d & 1 & 1 \end{bmatrix} \right\}$

EXAMPLE (contd):

If b = 1, L(a,b) = (!a)1 = !a ;

Elseif b = 0, L(a,b) = (!a)0 = 0 ;

The new image after the above operation is:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

We can see that this operation removes interior pixels and keeps the edges only. Hence, this is an edge detection operation.

EXAMPLE

Let $S = \{2,3,4,5,6,7\}$ and $L = a + b$. (+ = OR)

$$\text{IMAGE} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$S = \left\{ \begin{bmatrix} 0 & 1 & x \\ 0 & x & 0 \\ x & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & x \\ 0 & x & 0 \\ x & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & x \\ 1 & x & 0 \\ x & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & x \\ 1 & x & 0 \\ x & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & x \\ 1 & x & 1 \\ x & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & x \\ 1 & x & 1 \\ x & 1 & 1 \end{bmatrix}, \right\}$$

Because $L(a,b)$ is an OR operation, all pixels that are 1 in the original will remain 1. That is:

$L = a + b = a + 1 = 1$

EXAMPLE (contd):

The only pixels that will change are those that are 0 in the original image and have a surround that is S (this means that $a = 1$). That is:

$L = a + b = a + 0 = a$

If we examine the set S we see that this set contains all pixels that are surrounded by a connected set of 1's. This operation will expand the object, and illustrates that the OR operation results in a dilation. The resultant image is:

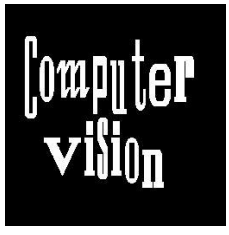
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

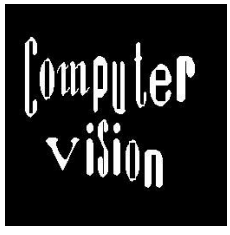
- The iterative morphological approach is quite versatile, and can be iterated, or repeated, to any degree desired
- We can use this technique to define methods for dilation, erosion, opening, closing, marking corners, finding edges, and other binary morphological operations
- With this technique the selection of the set *S* is comparable to defining the structuring element in the previously described approaches, and the operation *L(a,b)* defines the type of filtering that occurs

- A controlled erosion process is called *skeletonization*
- It is often used in optical character recognition and in many other applications
- A *skeleton* is what is left of an object when it has been eroded to the point of being only one pixel wide

Skeletonization Via Iterative Modification Morphological Filtering

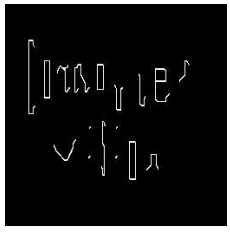
a) Original image





b) After 10 iterations

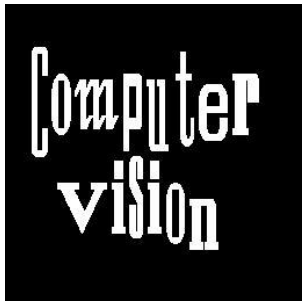
c) After 20 iterations



d) Iterating until no more changes occur

- In this example: L(a,b) = (!a)b, and S = (3,4)

Edge Detection Via Iterative Modification Morphological Filtering



a) Original image

b) Resultant image after one iteration

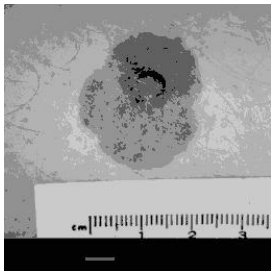
- In this example: L(a,b) = (!a)b, and S = (1,7)

- The morphological operations can be used with gray level images:

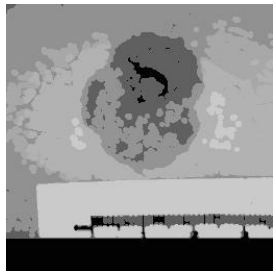
1. Threshold the gray level image to create a binary image, and then apply the existing operators

2. Operate on each gray level as if it were the '1' value and assuming everything else to be '0‘. Combine them by overlaying and "promoting" each pixel to the highest gray level value coincident with that location

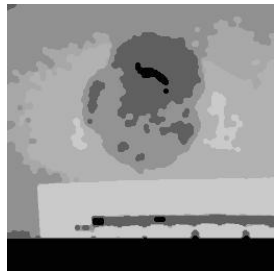
Gray Level Morphological Filtering



a) Original segmented skin tumor image, contains 1,708 objects



b) Image (a) after morphological opening using a 5x5 circular structuring element, contains 443 objects



c) Image (b) after closing using a 5x5 circular element, contains 136 objects

- With two adjacent gray levels, the brightest one is considered to be the object, '1', and the darker is the background, '0'