Imperial College London

6 – Morphological Operations Part 1

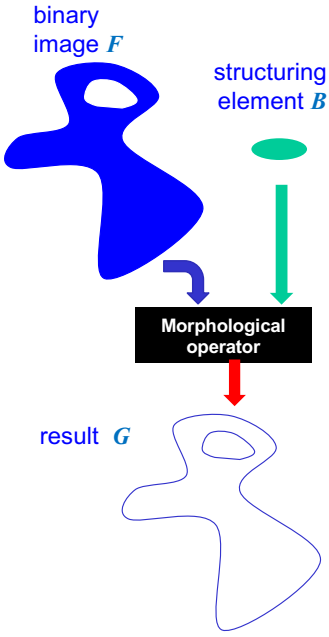
Prof Peter YK Cheung

Dyson School of Design Engineering

URL: www.ee.ic.ac.uk/pcheung/teaching/DE4_DVS/
E-mail: p.cheung@imperial.ac.uk

This lecture is based on materials found in Chapter 9 of the textbook, “Digital Image Processing”, 4th Edition, by RC Gonzalez and RE Woods.

Morphological Operation



- Take binary image ***F***
- Take structuring element ***B*** (similar to filter kernel)
- Perform **morphological operation** (similar to, but different from, convolution in spatial filter)
- Produce a new image ***G***
- Useful for:
  - Image **pre-processing** (noise filtering, shape simplification)
  - **Enhancing object structure** (skeletonizing, thinning, thickening, convex hull,)
  - **Segmenting objects** from the background
  - **Quantitative description** of objects (area, perimeter, projections,)

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

2. The **structuring element** is shifted over the image and at each pixel of the image its elements are compared with the set of the underlying pixels.

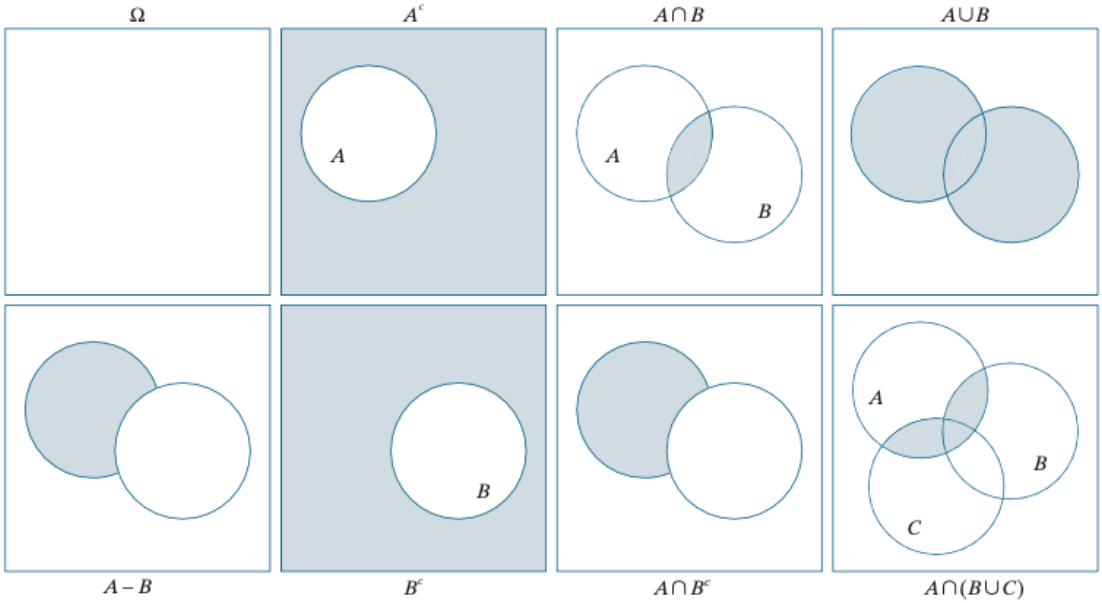
3. If the two sets of elements **match the condition** defined by the set operator (e.g. if set of pixels in the structuring element is a subset of the underlying image pixels), the **pixel underneath the origin of the structuring element** is set to a pre-defined value (0 or 1 for binary images).

4. A morphological operator is therefore defined by its **structuring element** and the applied **set operator**.

5. Morphological operators are used for:
   - Image pre-processing (noise filtering, shape simplification)
   - Enhancing object structures (skeletonization, thinning, convex hull, object marking)
   - Segmentation of the object from background
   - Quantitative descriptors of objects (area, perimeter, projection etc.).

a b c d
e f g h

Logic Operators



Venn diagrams corresponding to some of the set operations.

Ω is the null (or empty) set.

The results of the operations, such as A^c , are shown shaded. Figures (e) and (g) are the same, proving via Venn diagrams that $A - B = A \cap B^c$.

Notations for Set and Logical Operations

- If *a* is an *element* of set *A*, then $a \in A$
- If *a* is NOT an *element* of set *A*, then $a \notin A$
- A set is denoted by the contents of two braces: { .}.
  - For example:

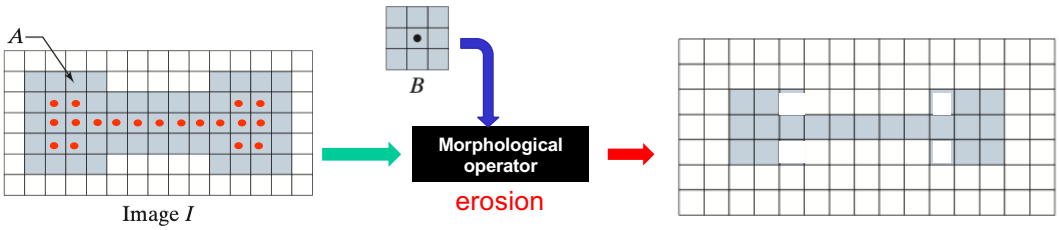
$$C = \{c \mid c = -d, d \in D\}$$

  means that *C* is the set of elements, *c*, such that *c* is formed by multiplying each of the elements of set *D* by −1.
- If every element of a set A is also an element of a set B, then A is said to be a **subset** of B, denoted as: $A \subseteq B$
- **Union**: $C = A \cup B$ **Intersection**: $D = A \cap B$ **disjoint** $A \cap B = \varnothing$
- Complement: $A^c = \{w \mid w \notin A\}$
- Difference of two sets A and B: $A - B = \{w \mid w \in A, w \notin B\} = A \cap B^c$

Here are some notations used throughout this lecture and in Chapter 9 of the textbook.

Example of a Morphological Operation

- Binary image, I, consisting of an object (set) A shown shaded.
- 3 × 3 **structuring element** (SE) whose elements are all 1's (foreground pixels). The background pixels are (0's).
- Morphological operations:
  1. Form an image containing only the object A as image I
  2. Move B over image I, pixel-by-pixel
  3. At each pixel, if B is **completely contained** in A, mark the location of the origin of B as a **foreground** pixel (i.e. 1) in the new image, else leave as **background**



The slides above shows the result after the origin of *B* has visited every element of *I*.

We see that, when the origin of *B* is on a border element of *A*, part of *B* ceases to be contained in *A*, thus eliminating that location of the origin of *B* as a possible foreground point of the new image. The net result is that the boundary of set *A* is *eroded*, as shown.

Because of the way in which we defined the operation, the maximum excursion needed for *B* in *I* is when the origin of *B* (which is at its centre) is contained in *A*. With *B* being of size 3 × 3, the narrowest background padding we needed was one pixel wide, as shown above. By using the smallest border needed for an operation, we keep the drawings smaller. In practice, we specify the width of padding based on the maximum dimensions of the structuring elements used, regardless of the operations being performed.

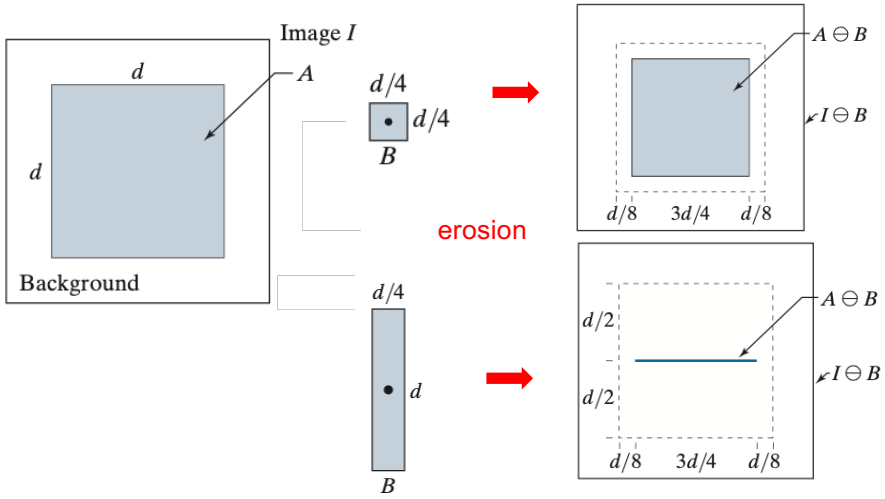
When we use terminology such as "the structuring element *B* is contained in set *A*," we mean *specifically* that the *foreground* elements of *B* overlap *only* elements of *A*. This becomes an important issue when *B* also contains background and, possibly, don't care elements.

Erosion Operator $\ominus$

- Formal definition of **erosion**: $A \ominus B = \{z|(B)_z \subseteq A\}$

  such that — contained in

- In plain words, out pixel is '1' if **ALL** the pixels in image under B are '1'.



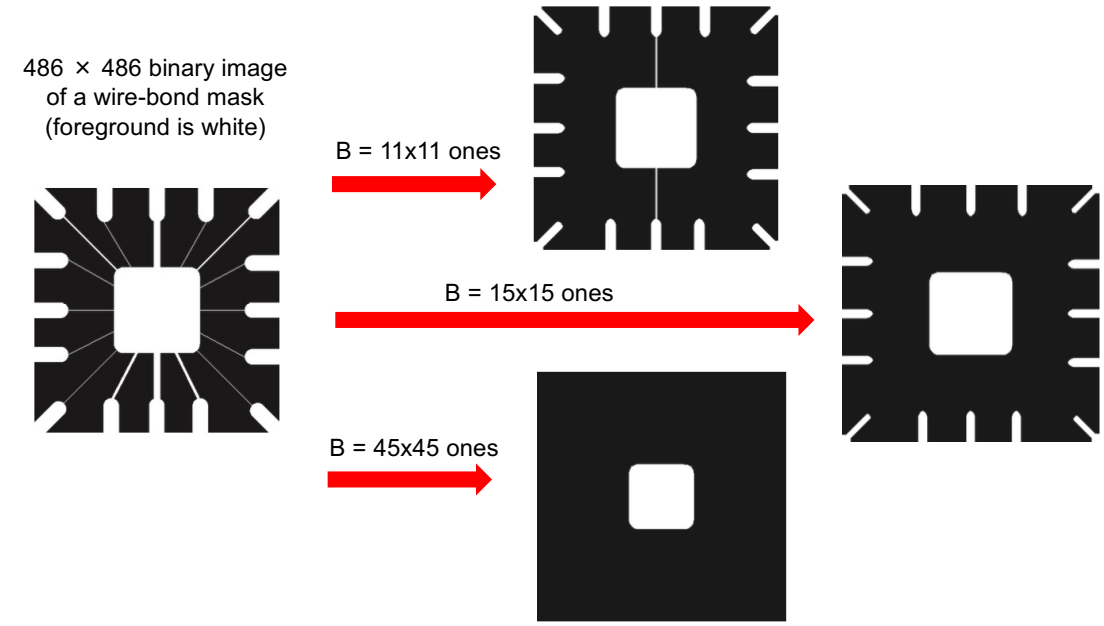
Although we do not emphasize on the formal mathematical proves and definitions in this module, it is worth considering the following definition of *erosion:*

With *A* and *B* as sets in Z^2 (i.e. 2D space), the *erosion* of *A* by *B*, denoted $A \ominus B$, is defined as

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

where *A* is a set of **foreground** pixels, *B* is a **structuring element**, and the *z*'s are **foreground values** (1's). In words, this equation indicates that the *erosion* of *A* by *B* is the **set of** ({ }) of all points *z* **such that** (|) *B*, translated by *z,* is **contained in** ($\subseteq$) *A.*

Application of Erosion Operator ⊖

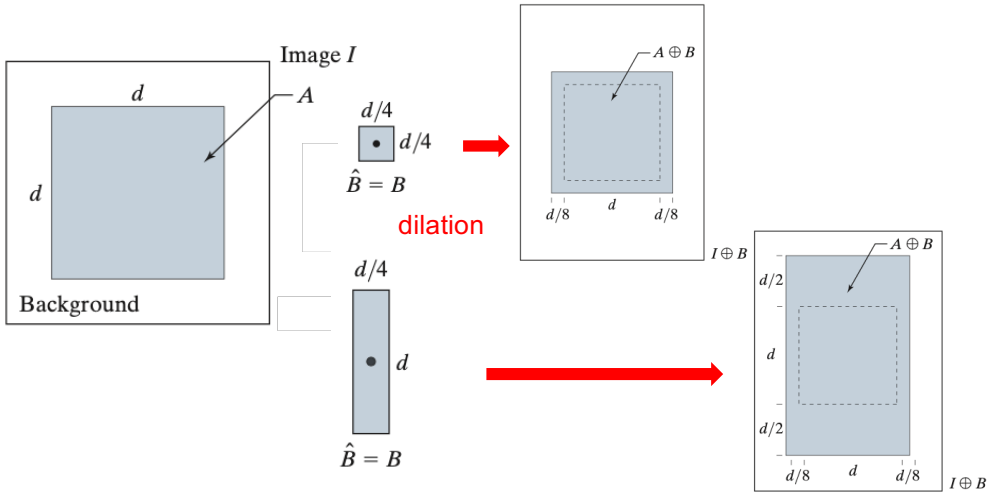


Using erosion to remove image components.

The input image is a 486 × 486 binary image of a wire-bond mask in which foreground pixels are shown in white. The other Images are eroded using square structuring elements of sizes 11 × 11, 15 × 15, and 45 × 45 elements, respectively, all valued 1.

Dilation Operator ⊕

- Formal definition of **dilation** is: $A \oplus B = \left\{ z | \left(\hat{B}\right)_z \cap A \neq \emptyset \right\}$
- In plain words, pixel is '1' if **ANY** of the pixels in image under B is '1'.



This equation is based on **reflecting** *B* about its origin and translating the reflection by *z*, as in erosion. Therefore, the structuring element is $\hat{B}$, which is the reflection of *B*. The dilation of *A* by *B* then is the set of all displacements, *z*, such that the foreground elements of $\hat{B}$ overlap **at least one element of *A***. (Remember, *z* is the displacement of the origin of *B*.) Based on this interpretation, the definition of **dilation** is:

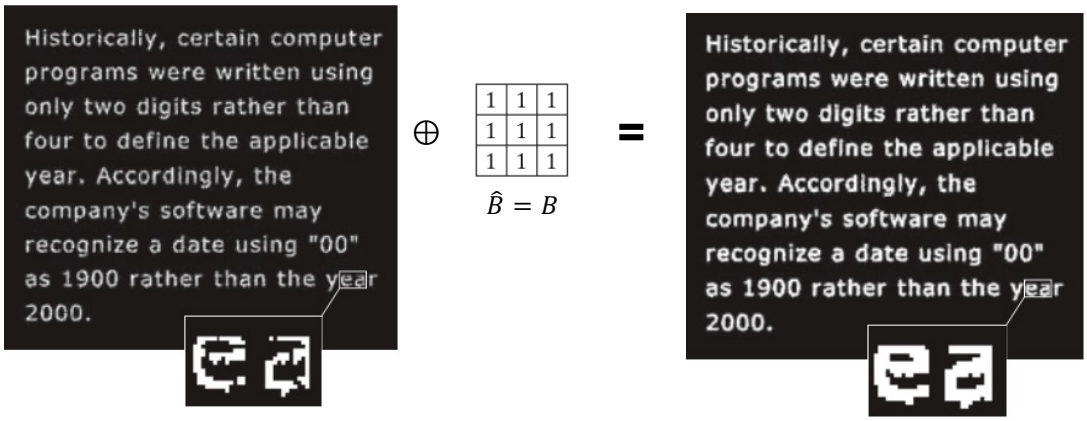
$$A \oplus B = \left\{ z | \left(\hat{B}\right)_z \cap A \neq \emptyset \right\}$$

The basic process of flipping (rotating) B about its origin and then successively displacing it so that it slides over set A is analogous to spatial convolution. However, keep in mind that **dilation** is based on **set operations** and therefore is a **nonlinear operation**, whereas **convolution** is a **sum of products**, which is a **linear operation**.

Unlike erosion, which is a **shrinking** or **thinning** operation, dilation "**grows**" or "**thickens**" objects in a binary image. The manner and extent of this thickening is controlled by the shape and size of the structuring element used.

The figure above shows the same object used in the last slide and the structuring element SE. In this case $\hat{B}$ = B because the SE is symmetric about its origin. The dashed shows the boundary of the original object for reference, and the solid line shows the limit beyond which any further displacements of the origin of $\hat{B}$ by z would cause the intersection of $\hat{B}$ and A to be empty. Therefore, all points on and inside this boundary constitute the dilation of A by B. The lower pair shows a structuring element designed to achieve more dilation vertically than horizontally, and the dilation achieved with this element.

Application of Dilation Operator



One of the simplest applications of dilation is for bridging gaps. The left image shows has broken characters.

The maximum length of the breaks is known to be two pixels. The dilated image on the right shows a structuring element that can be used for repairing the gaps. Here we use white (1) to denote the foreground and black (0) for the background when working with images.

The right image shows the result of dilating the original image with the structuring element. The gaps were bridged. One important advantage of the morphological approach over the lowpass filtering method we used to bridge the gaps is that the morphological method resulted directly in a binary image. Lowpass filtering, on the other hand, started with a binary image and produced a grayscale image that would require thresholding to convert it back to binary form.

Observe that set *A* in this application consists of numerous disjointed objects of foreground pixels.

Duality of Erosion and Dilation

- Erosion of A by B is the complement of the dilation of A^c by $\hat{B}$

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

- If B is symmetrical with respect to its origin, then $\hat{B} = B$. Then erosion of A is the same as dilating its background and complementing the result.

- Dilation of A by B is the complement of the erosion of A^c by $\hat{B}$

$$(A \oplus B)^c = A^c \ominus \hat{B}$$

Formal proof of duality (you don't need to know this, but here it is for completeness):

Starting with the definition of erosion

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

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

If set $(B)_z$ is contained in A, then it follows that $(B)_z \cap A \neq \emptyset$, in which case the preceding expression becomes

$$(A \ominus B)^c = \{z|(B)_z \cap A^c = \emptyset\}^c$$

But the *complement* of the set of z's that satisfy $(B)_z \cap A = \emptyset$ is the set of z's such that $(B)_z \cap A^c \neq \emptyset$. Therefore,

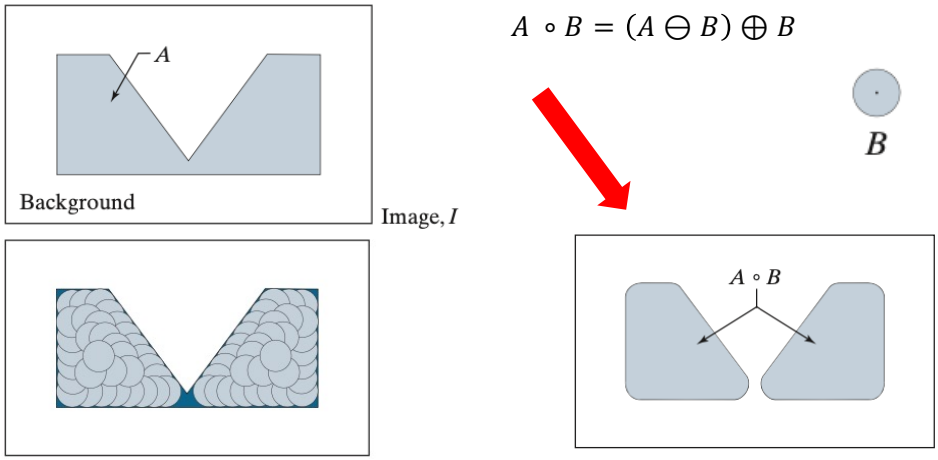
$(A \ominus B)^c = \{z|(B)_z \cap A^c \neq \emptyset\}$, which is the definition of dilation.

Therefore,

$$(A \ominus B)^c = A^c \oplus B$$

Opening Operator - ∘

- Combine erosion and dilation operators results in other operators.
- **Opening operator**: smoothes contour, breaks narrow passages, and eliminates thin protrusions.
- Opening A by B is **erosion** of A by B, **followed by dilation** of the result by B:



The opening equation: $A \circ B = (A \ominus B) \oplus B$ has a simple geometrical interpretation.

The opening of *A* by *B* is the union of all the translations of *B* so that *B* fits entirely in *A*.

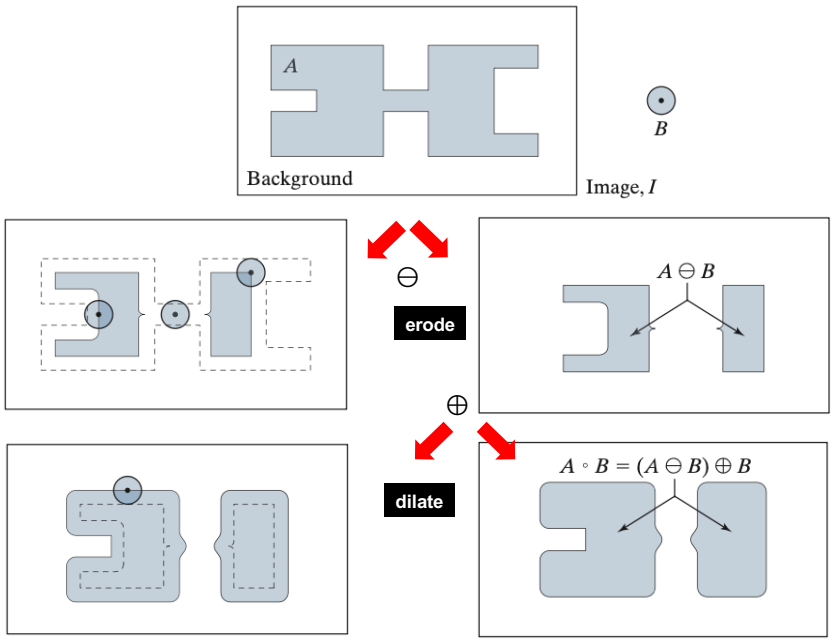
The original shows an image containing a set (object) *A* and the SE is a solid, circular structuring element, *B*.

Bottom left shows some of the translations of *B* such that it is contained within *A*, and the set shown shaded in the final image (bottom right) is the union of all such possible translations.

Observe that, in this case, the opening is a set composed of two disjoint subsets, resulting from the fact that *B* could not fit in the narrow segment in the centre of *A*.

The ability to eliminate regions narrower than the structuring element is one of the key features of morphological opening.

Opening in Action



Morphological opening has the following properties:

(a) $A \circ B$ is a subset of *A*.

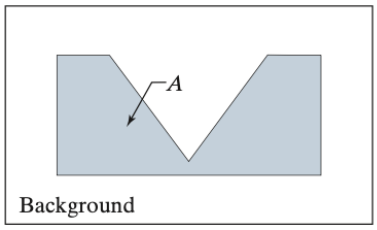
(b) If *C* is a subset of *D*, then $C \circ B$ is a subset of $D \circ B$.

(c) $(A \circ B) \circ B = A \circ B$

The last property means that multiple openings of a set is the same as opening once. Subsequent opening operations have no effect.

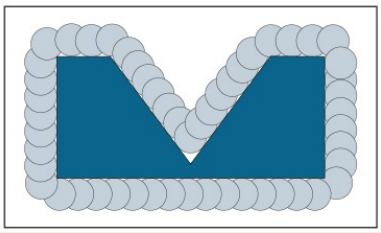
Closing Operator - •

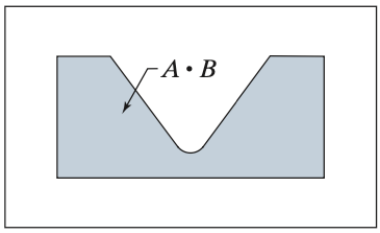
- **Closing operator**: smooth sections of contours, fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour.
- Closing of A by B is dilation of A by B, followed by erosion of result by B
- Closing of set A by structuring element B, denoted as $A \cdot B$, is defined as:



Image, I

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



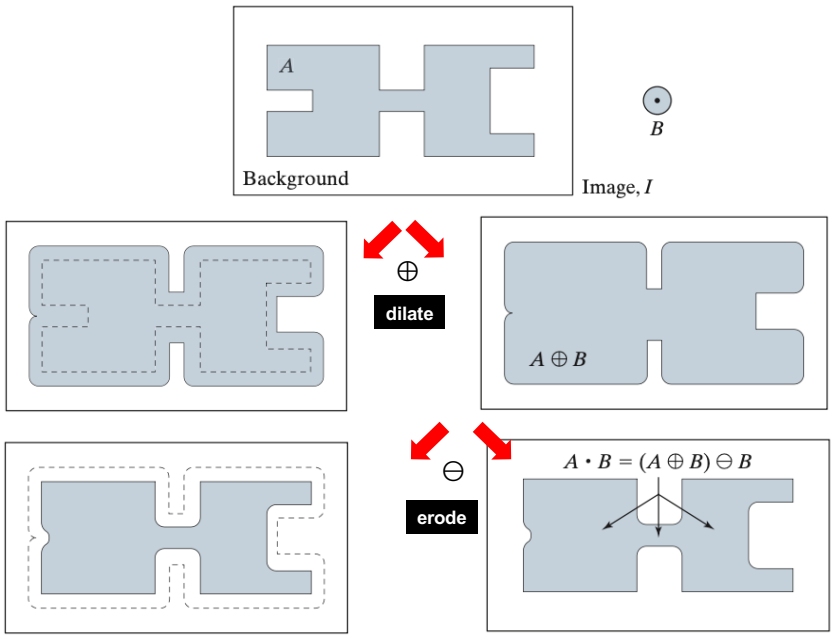


Closing has a similar geometric interpretation, except that now we translate *B outside A*.

The closing is then the *complement* of the union of all translations of *B* that *do not* overlap *A*.

Note that the boundary of the closing is determined by the furthest points *B* could reach without going inside any part of *A*.

Closing in Action



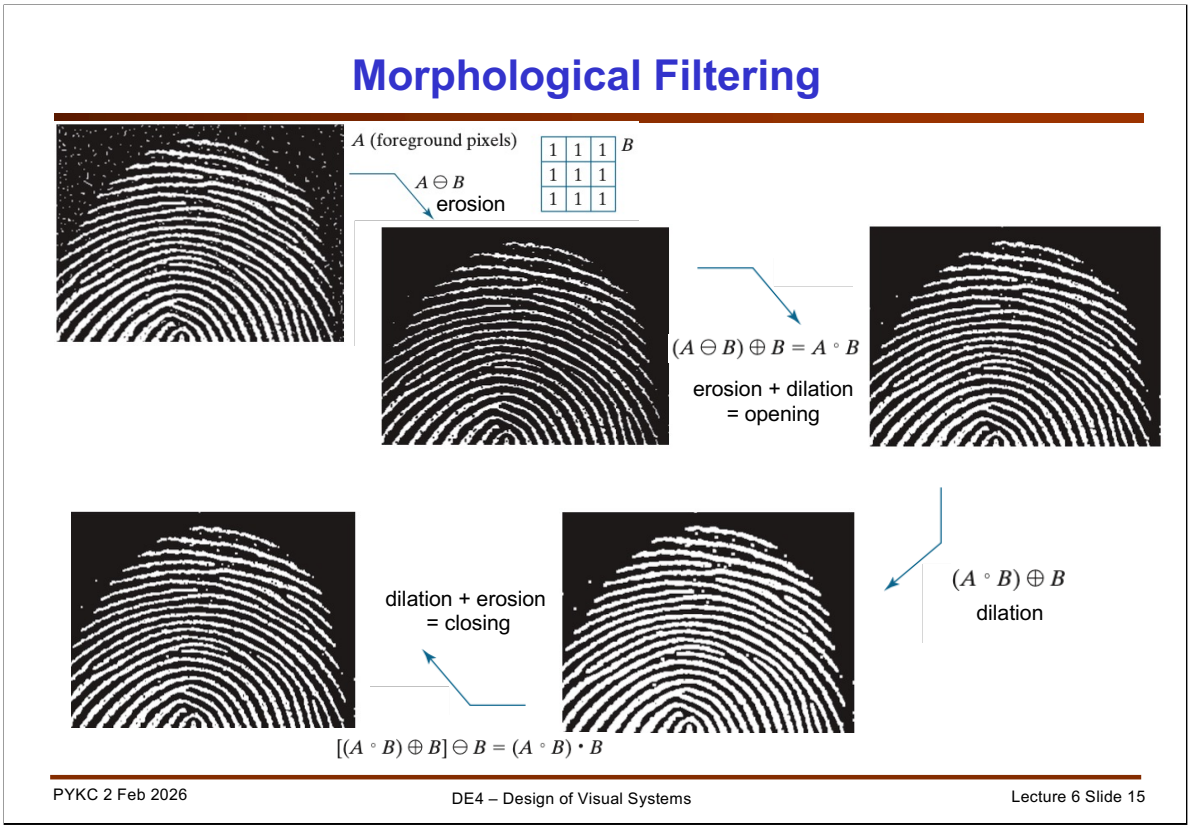
Morphological closing has the following properties:

(a) A is a subset of $A \cdot B$.

(b) If C is a subset of D, then $C \cdot B$ is a subset of $D \cdot B$.

(c) $(A \cdot B) \cdot B = A \cdot B$

The last property means that multiple closing of a set is the same as closing it once. Subsequent closing operations have no effect.



Morphological operations can be used to construct filters similar in concept to the spatial filters in Lecture 5. The original binary image shows a section of a fingerprint corrupted by noise. *A* is the set of all foreground (white) pixels, which includes objects of interest (the fingerprint ridges) as well as white specks of random noise. The background is black. The noise manifests itself as white specks on a dark background and dark specks on the white components of the fingerprint.

The objective is to eliminate the noise and its effects on the print, while distorting it as little as possible. A morphological filter consisting of an **opening** followed by a **closing** can be used to accomplish this objective.

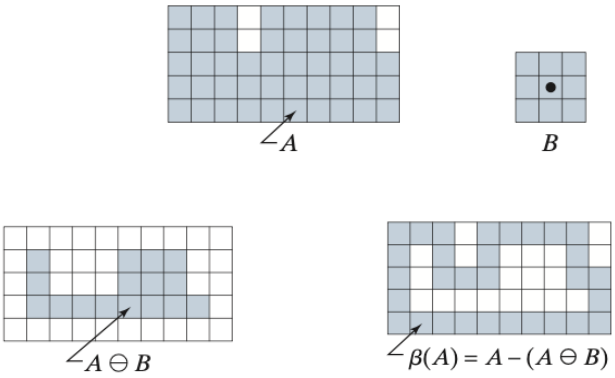
The structuring element is B, a 3x3 square of 1's. The rest of figures in the slide shows the sequence of steps in the filtering operation.

1. First is the result of eroding *A* by *B*. The white speckled noise in the background was eliminated almost completely in the erosion stage of opening because in this case most noise components are smaller than the structuring element. The size of the noise elements (dark spots) contained within the fingerprint actually increased in size. The reason is that these elements are inner boundaries that increase in size as objects are eroded.
2. Next step is to perform dilation on the results. This enlargement is countered by performing dilation. The two operations just described constitute the opening of *A* by *B*. The net effect of opening was to reduce all noise components in both the background and the fingerprint itself. However, new gaps between the fingerprint ridges were created.
3. To counter this undesirable effect, we perform a dilation on the opening. Most of the breaks were restored, but the ridges were thickened.
4. This condition is remedied by erosion. This final result is remarkably clean of noise specks, but it still shows some specks of noise that appear as single pixels.

Boundary Extraction

- Boundary of a set A of foreground pixels denoted by $\beta(A)$, can be obtained by first eroding A by a suitable structuring element B, and then performing the set difference between A and its erosion.

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



The boundary of a set A of foreground pixels, denoted by $\beta(A)$, can be obtained by first eroding A by a suitable structuring element B, and then performing the set difference between A and its erosion $(A \ominus B)$. That is,

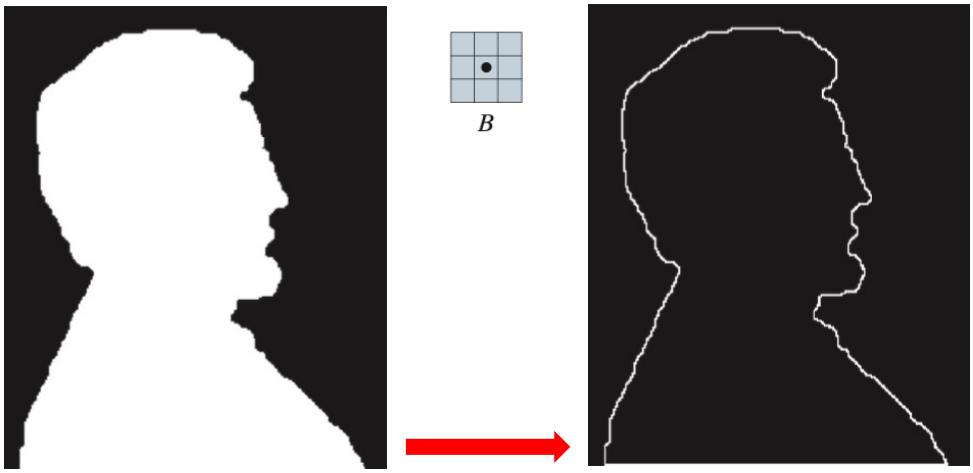
$$\beta(A) = A - (A \ominus B)$$

The slide above illustrates the mechanics of boundary extraction. It shows a simple binary object, a structuring element B, and the result of using the $\beta(A)$ operation.

The structuring element B in the example here is among the most frequently used, but it is not unique. For example, using a 5×5 structuring element of 1's would result in a boundary between 2 and 3 pixels thick.

It is assumed that the image contain A was padded with a border of background elements, and that the results were cropped back to the original size after the morphological operations were completed.

Boundary Extraction Example



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

This example further illustrates the use of the boundary extraction equation $\beta(A) = A - (A \ominus B)$, using a 3 × 3 structuring element of 1's.

The input image is assumed to be a black and white image, where we show foreground pixels (1's) in white and background pixels (0's) in black. The elements of the SE, which are 1's, also are treated as white. Because of the size of the structuring element used being 3x3, the boundary is one pixel thick.

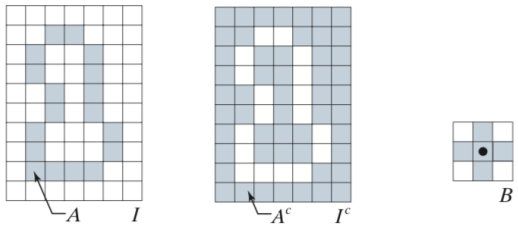
Hole filling

- A *hole* is defined as a background region surrounded by a **connected border** of foreground pixels.
- The formal definition of the hole filling algorithm is:

$$X_k = (X_{k-1} \oplus B) \cap I^c, \qquad k = 1,2,3, \ldots.$$

where X_0 is the first pixel known to be in the hole and X_k are the other hole pixels.

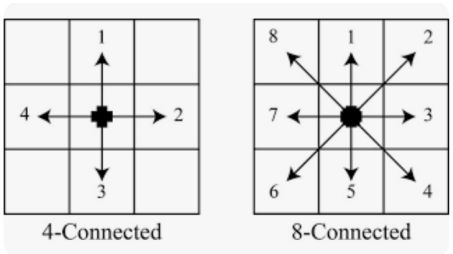
- Consider this example:

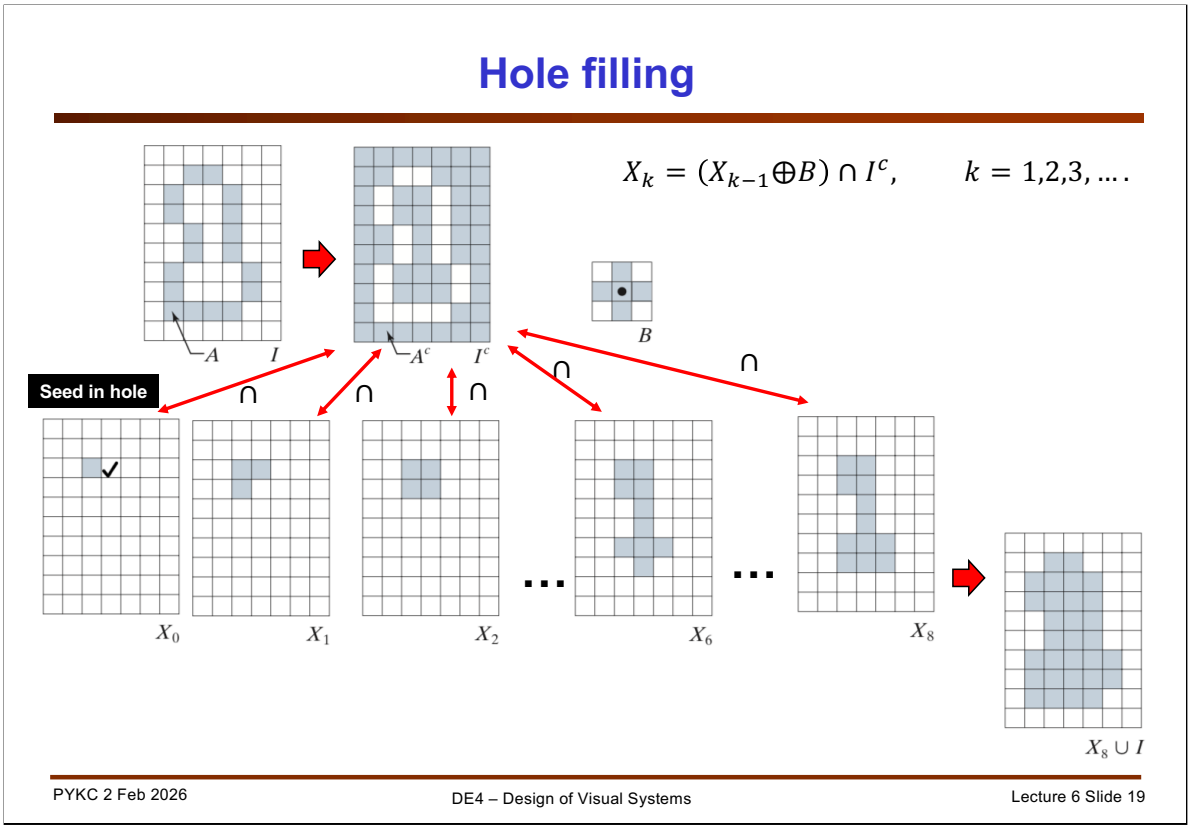


A hole may be defined as a background region surrounded by a connected border of foreground pixels. The hole filling algorithm is based on set **dilation**, **complementation**, and **intersection**.

Let A denote a set whose elements are 8-connected boundaries, with each boundary enclosing a background region (i.e., a hole). Given a point in each hole, the objective is to fill all the holes with foreground elements (1's).

Note the difference between 4- and 8-connected boundaries:





We begin by forming an array, X_0 , of 0's (the same size as I, the image containing A), except at locations in X_0 that correspond to pixels that are known to be holes, which we set to 1. Then, the following procedure fills all the holes with 1's:

$$X_k = (X_{k-1} \oplus B) \cap I^c, \qquad k = 1,2,3, \ldots.$$

where B is the symmetric structuring element in the slide.

The algorithm terminates at iteration step k if $X_k = X_{k-1}$. Then, X_k contains all the filled holes.

The set union of X_k and I contains all the filled holes and their boundaries.

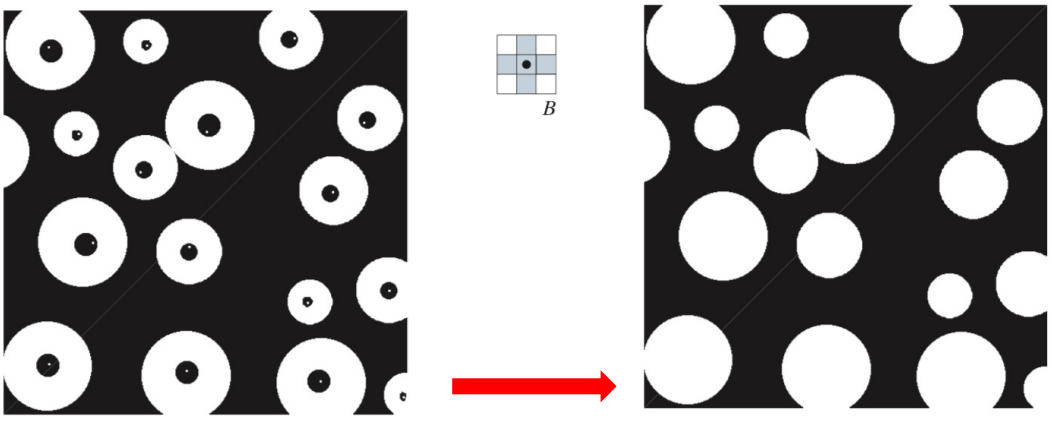
The dilation $(X_{k-1} \oplus B)$ would fill the entire area if left unchecked, but the intersection at each step with I^c limits the result to inside the region of interest. This process is appropriately called conditional dilation.

The rest of the slide illustrates further the mechanics of hole filing equation above.

This example only has one hole, but the concept applies to any finite number of holes, assuming that a point inside each hole is given.

Example of Hole Filling

Thresholded image of ball bearings



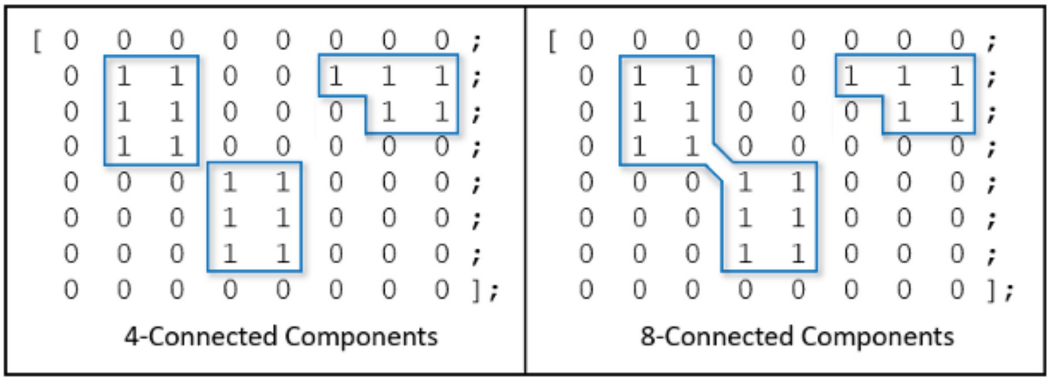
$$X_k = (X_{k-1} \oplus B) \cap I^c, \qquad k = 1,2,3, \ldots.$$

This figure shows an image of white circles with black holes. An image such as this might result from thresholding into two levels a scene containing polished spheres (e.g., ball bearings).

The dark circular areas inside the spheres would result from reflections. The objective is to eliminate the reflections by filling the holes in the image. The right image shows the result of filling all the spheres. Because it must be known whether black points are background points or sphere inner points (i.e., holes), fully automating this procedure requires that additional “intelligence” be built into the algorithm.

What are Connected Components?

- A **connected component** is a set of adjacent pixels in a binary image.
- Two possible definitions of what is "connected":
  - 4-connectivity — Pixels are connected if their edges touch.
  - 8-connectivity — Pixels are connected if their edges or corners touch.



PYKC 2 Feb 2026 DE4 – Design of Visual Systems Lecture 6 Slide 21

A *connected component*, or an object, in a binary image is a set of adjacent pixels. Determining which pixels are adjacent depends on how pixel connectivity is defined. For a two-dimensional image, there are two standard connectivities:

- 4-connectivity — Pixels are connected if their edges touch. Two adjoining pixels are part of the same object if they are both on and are connected along the horizontal or vertical direction.
- 8-connectivity — Pixels are connected if their edges or corners touch. Two adjoining pixels are part of the same object if they are both on and are connected along the horizontal, vertical, or diagonal direction.

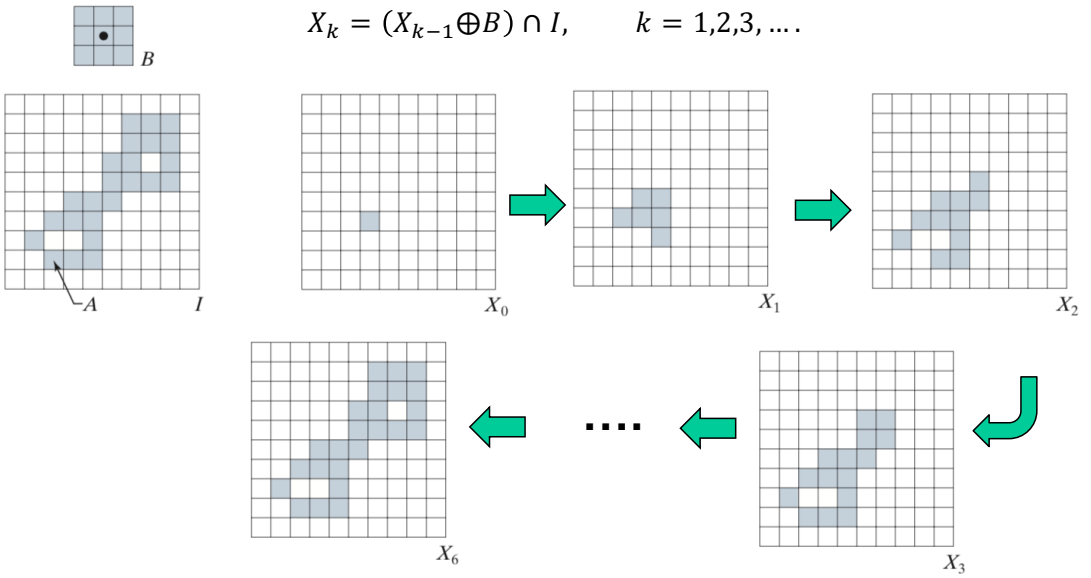
The diagram above shows two identical matrices that represent a binary image. On each matrix is an overlay highlighting the connected components using 4-connectivity and 8-connectivity, respectively. There are three connected components using 4-connectivity, but only two connected components using 8-connectivity.

We can use morphological operators to extract connected components in a binary image as demonstrated in the following slides.

Extraction of Connected Components

- A connected component is a set of adjacent pixels in a binary image.

$$X_k = (X_{k-1} \oplus B) \cap I, \qquad k = 1,2,3, \ldots.$$



Being able to extract connected components from a binary image is central to many automated image analysis applications. Let *A* be a set of foreground pixels consisting of one or more connected components and form an image X_0 (of the same size as *I*, the image containing *A*) whose elements are 0's (background values), except at each location known to correspond to a point in each connected component in *A*, which we set to 1 (foreground value). The objective is to start with X_0 (the seed) and find all the connected components in *I*. The following iterative procedure accomplishes this:

$$X_k = (X_{k-1} \oplus B) \cap I^c, \qquad k = 1,2,3, \ldots.$$

where B is the SE with 3x3 1's.

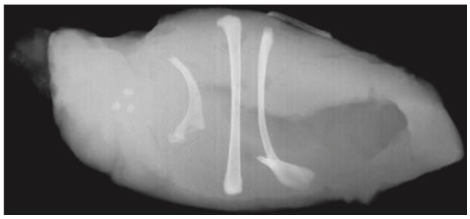
The procedure terminates when $X_k = X_{k-1}$, with X_k containing all the connected components of foreground pixels in the image.

The equation for connected components is very similar to that for hole filling (slide 18), except that the intersection is with I, and not I^c. This is because here we are looking for foreground points, while the objective hole filling is to find background points.

Note that the shape of the structuring element used is based on 8-connectivity between pixels. As in the hole-filling algorithm, this algorithm is applicable to any finite number of connected components contained in I, **assuming that a point is known in each component** (i.e. a known seed for each).

Example application of Connected Components

- X-ray image of a chicken fillet with bone fragments embedded.



thresholding

Connected component	No. of pixels in connected comp
01	11
02	9
03	9
04	39
05	133
06	1
07	1
08	743
09	7
10	11
11	11
12	9
13	9
14	674
15	85

15 connected components found with sizes in pixels

$X_k = (X_{k-1} \oplus B) \cap I$

Erosion with 5x5 SE of 1's

Connected components are used frequently for automated inspection. Above shows an X-ray image of a chicken breast that contains bone fragments. It is important to be able to detect such foreign objects in processed foods before shipping.

In this application, the density of the bones is such that their nominal intensity values are significantly different from the background. This makes extraction of the bones from the background a simple matter by using a single threshold. The result is the binary image on top right.

The most significant feature in this figure is the fact that the points that remain after thresholding are clustered into objects (bones), rather than being scattered. We can make sure that only objects of "significant" size are contained in the binary image by eroding its foreground.

In this example, we define as significant any object that remains after erosion with a 5 × 5 SE of 1's. The bottom right image shows the result of erosion.

The next step is to analyse the size of the objects that remain. We label (identify) these objects by extracting the connected components in the image. The table on the bottom left lists the results of the extraction. There are 15 connected components, with four of them being dominant in size.

This is enough evidence to conclude that significant, undesirable objects are contained in the original image.