Morphological Filtering

김성영교수
금오공과대학교
컴퓨터공학과

학습 내용

- Morphological Filtering 개요
- Basic set theory
- Morphological operations for binary images
- Morphological operations for gray-scale images
- OpenCV를 사용한 Morphological operations 구현

Morphological filtering 개요

- *Morphology* relates to the structure or form of objects
- Useful for smoothing out object outlines, filling small holes, eliminating small projection, etc.
- Simplify a segmented image to facilitate the search for objects of interest
- Applicable to gray-level images as well as binary images

Basic Set Theory

Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be sets in Z^2

a is an **element** of $\boldsymbol{A}$ ➡ $a \in \boldsymbol{A}$

a is **not an element** of $\boldsymbol{A}$ ➡ $a \notin \boldsymbol{A}$

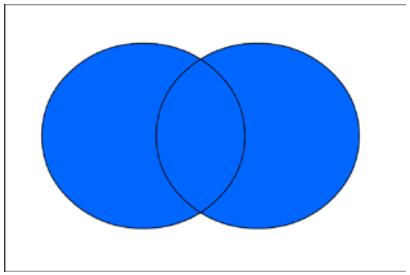
$\boldsymbol{A}$ is a **subset** of $\boldsymbol{B}$ ➡ $\boldsymbol{A} \subseteq \boldsymbol{B}$

The **union** of A and B ➡ $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

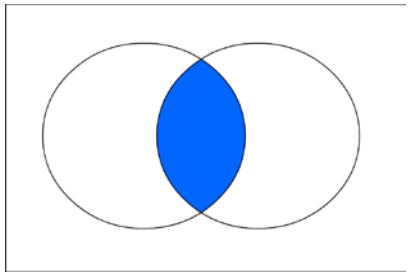
The **intersection** of A and B ➡ $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$

The **complement** of A ➡ $A^c = \{x \mid x \notin A\}$

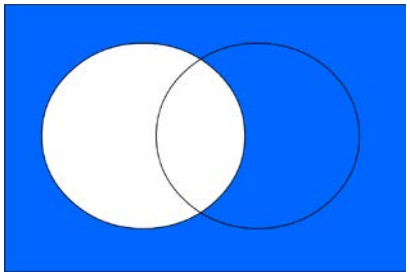
The **difference** of A and B ➡ $A - B = \{x \mid x \in A \text{ and } x \notin B\}$



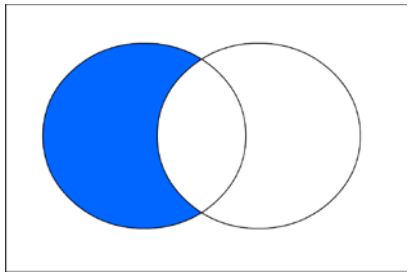
$A \cup B$



$A \cap B$



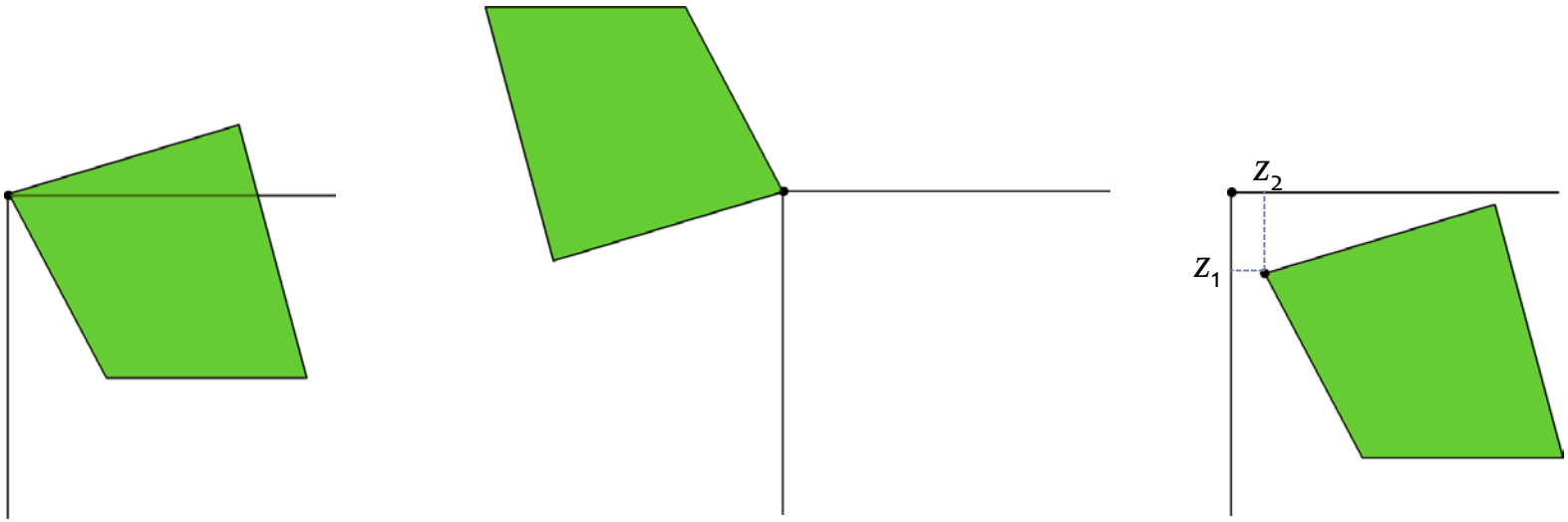
A^c



A-B

The **reflection** of $\boldsymbol{A}$ ➡ $\hat{\boldsymbol{A}} = \{x \mid x = -b, \text{ for } b \in \boldsymbol{A}\}$

The **translation** of $\boldsymbol{A}$ ➡ $(\boldsymbol{A})_z = \{c \mid c = b + z, \text{ for } b \in \boldsymbol{A}\}$



reflection translation

Morphological operations for binary images

- Two principal morphological operations

  Dilation & Erosion

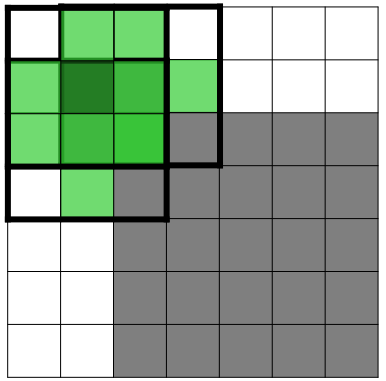
- Can be customized by the proper selection of the structuring element, which determines exactly how the object will be dilated or eroded
- Performed by laying the structuring element on the image and sliding it across the image in a manner similar to convolution

Dilation

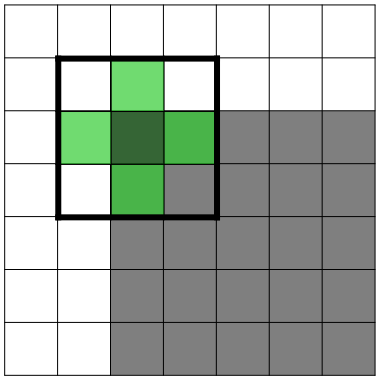
Expand objects, thus potentially filling in small holes and connecting disjoint objects

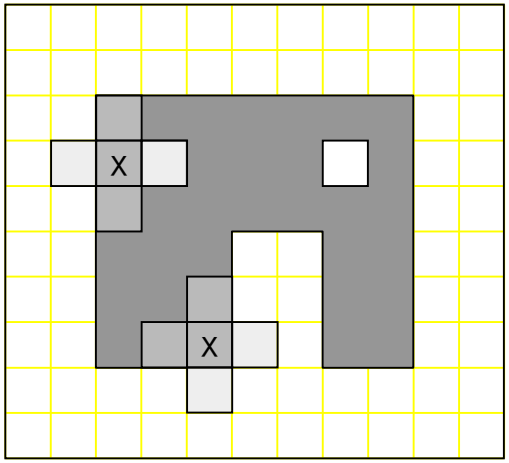
The dilation of A by the structuring element B,

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

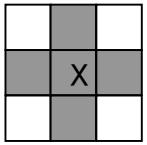


1. If the origin of the structuring element coincides with a ‘0’ in the image, there is no change; move to the next pixel.
2. Otherwise (in case of ‘1’), perform the OR logic operation on all pixels within the structuring element



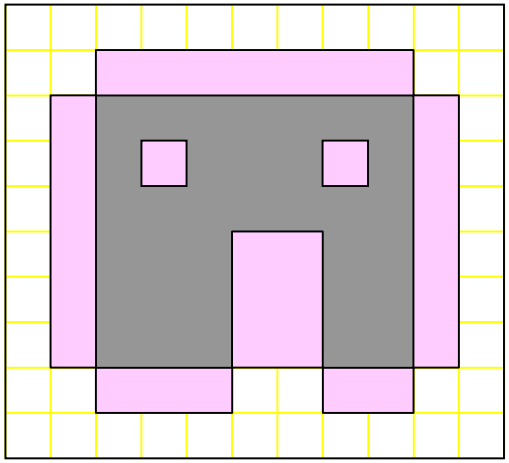


Original Image



Structuring Element

x : origin



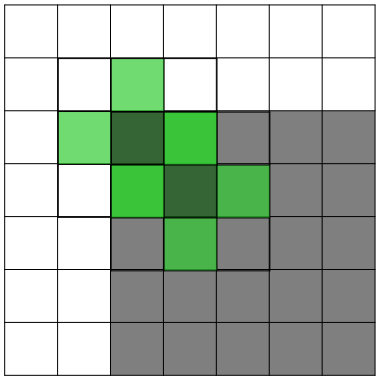
After Dilation

Erosion

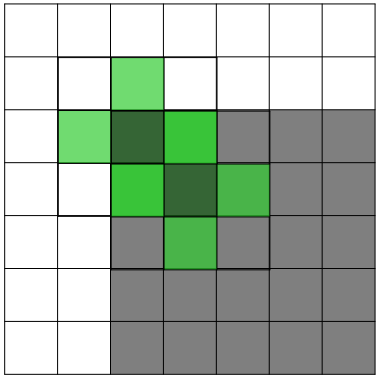
Shrinks objects by etching away(eroding) their boundaries

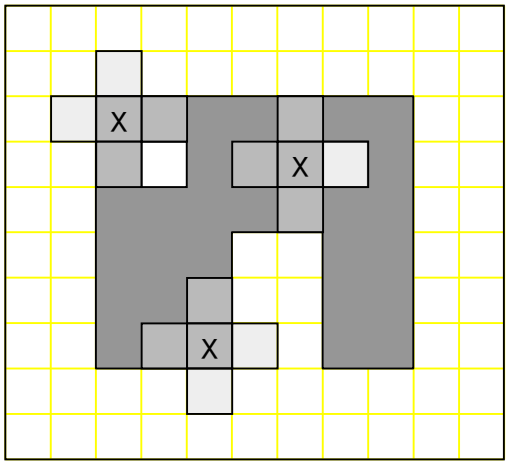
The erosion of A by B,

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

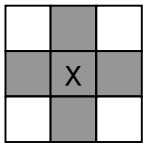


1. If the origin of the structuring element coincides with a ‘0’ in the image, there is no change; move to the next pixel
2. If the origin of the structuring element coincides with a ‘1’ in the image, and any of the ‘1’ pixels in the structuring element extend beyond the object(‘1’ pixels), then change the ‘1’ pixel in the image to a ‘0’



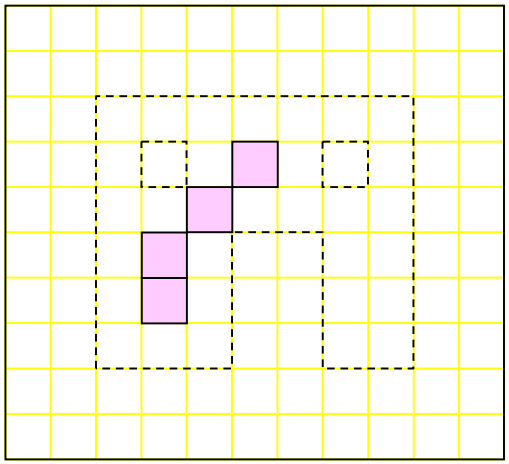


Original Image



Structuring Element

x : origin

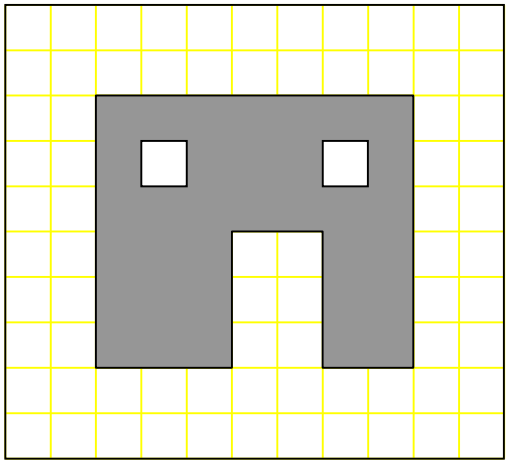


After Erosion

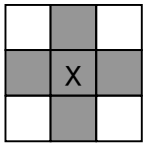
Opening

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

- Erosion followed by a Dilation
- Eliminate all pixels in regions that are too small to contain the structuring element

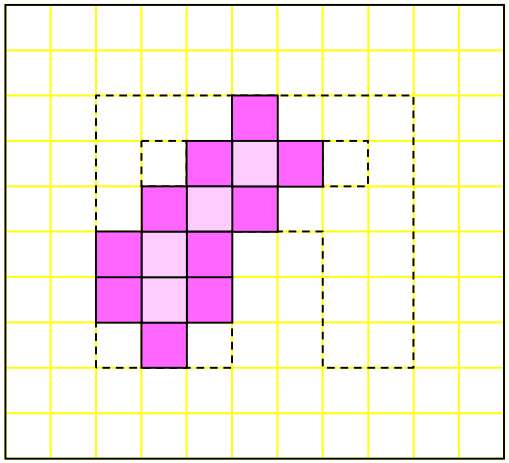


Original Image



Structuring Element

x : origin

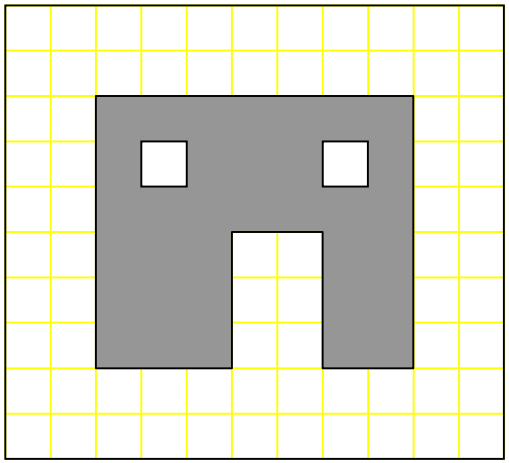


After Opening

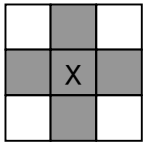
Closing

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

- Dilation followed by a Erosion
- Fill in hole and small gaps

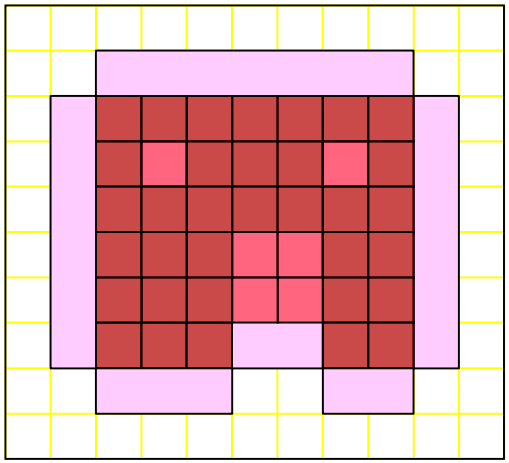


Original Image



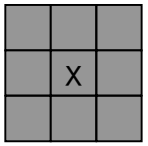
Structuring Element

x : origin

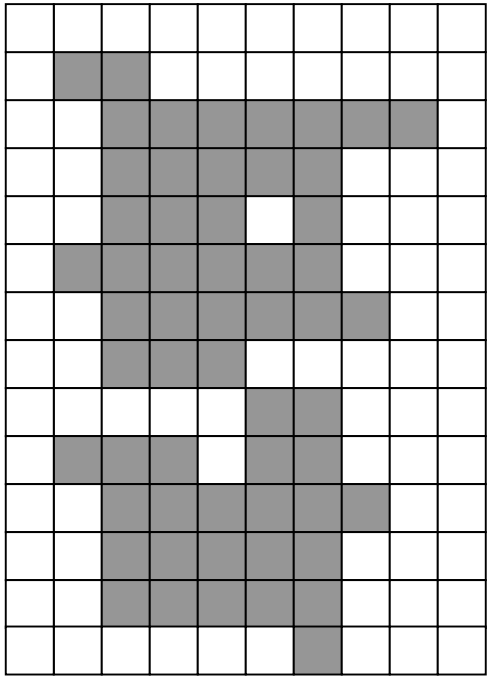


After Closing

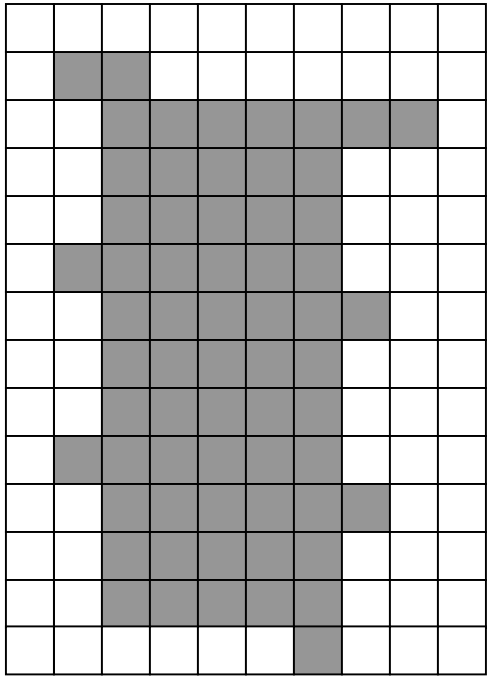
example



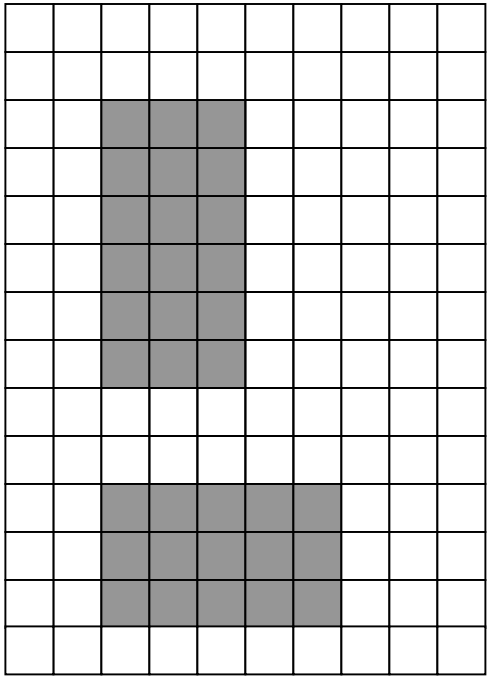
Structuring Element



Original Image



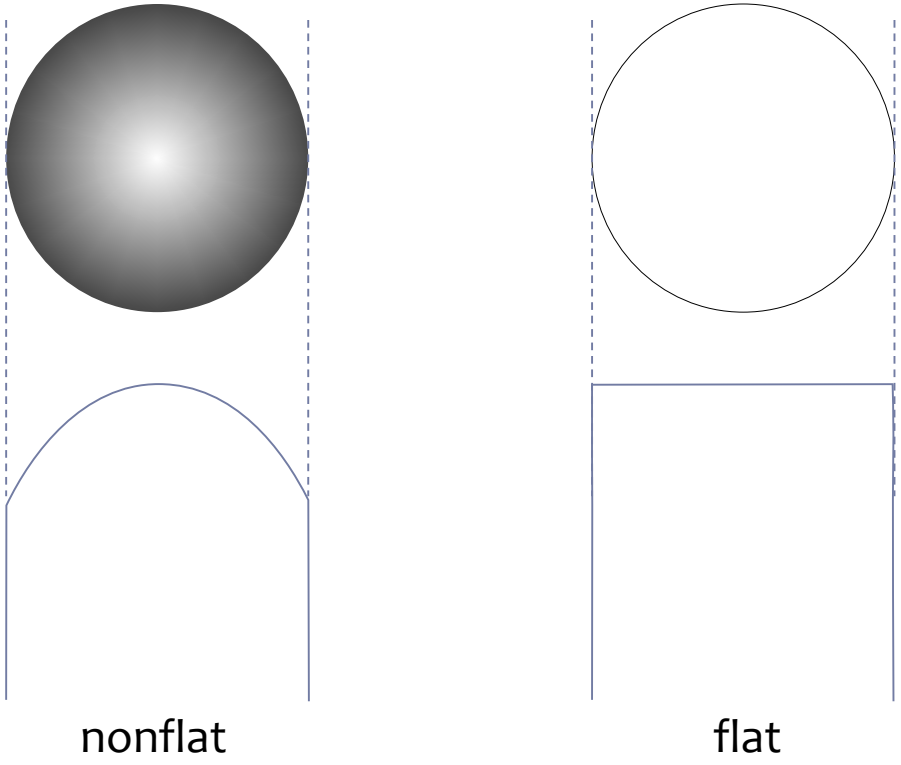
After Closing



After Opening

Operations for gray-scale images

Two types of a structuring element



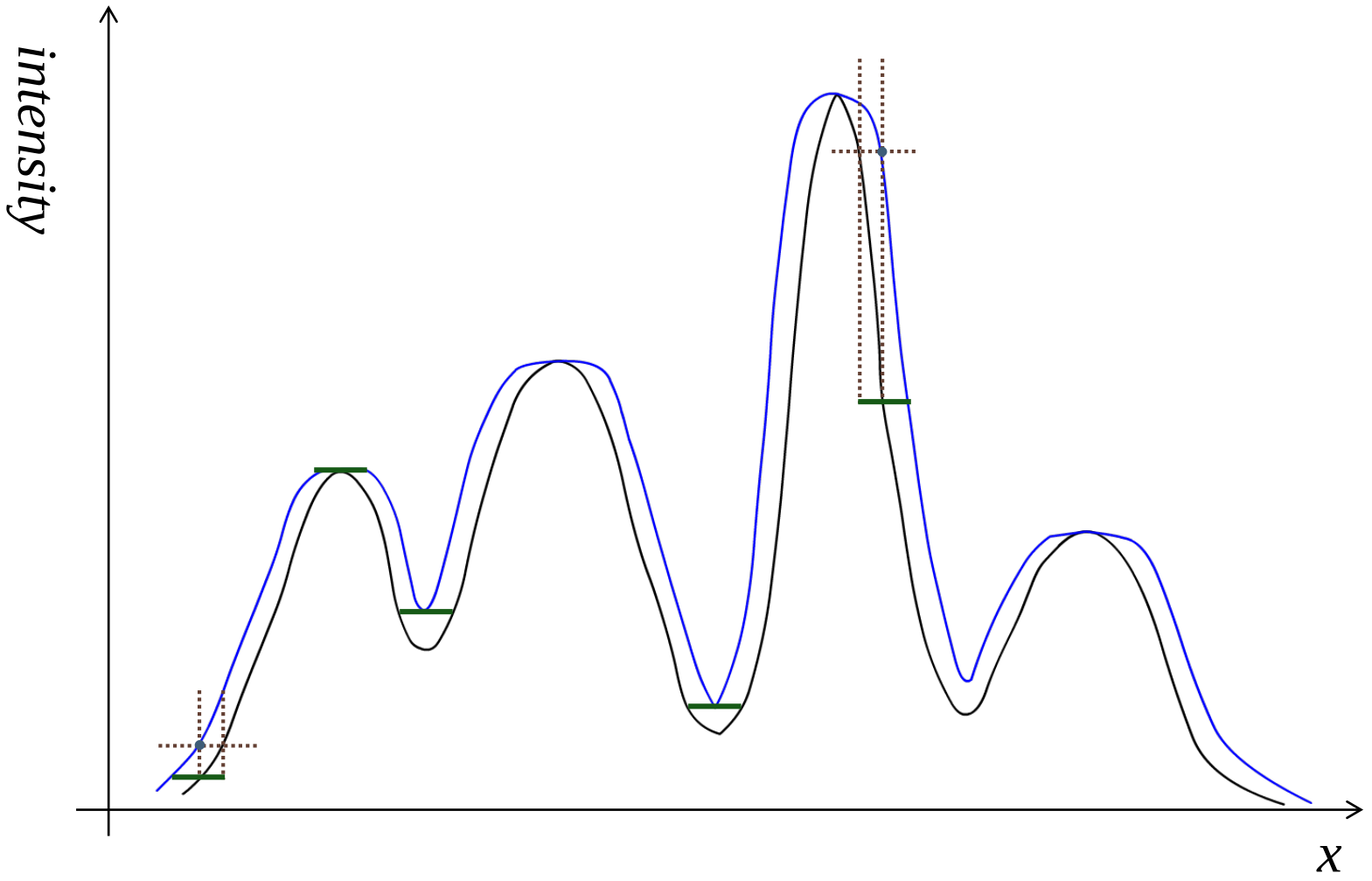
Dilation of *f* by *k*

$$(f \oplus k)(s,t) = \max_{(s,t)\in D_k} \{f(x-s, y-t) + k(s,t)\}$$

Erosion of *f* by *k*

$$(f \ominus k)(s,t) = \min_{(s,t)\in D_k} \{f(x+s, y+t) - k(s,t)\}$$

Dilation with a flat structuring element



OpenCV를 사용한 Morphological operations 구현

```
Mat   getStructuringElement( … )
void  dialte(  …  )
void  erode(  …  )
void  morphologyEx(  …  )
```

```
Mat getStructuringElement(
      int shape,
      Size ksize,
      Point anchor=Point(-1,-1)
)
```

shape

- MORPH_RECT: a rectangular structuring element
- MORPH_ELLIPSE: an elliptic structuring element,

  Rect(0, 0, ksize.width, ksize.height)
- MORPH_CROSS: a cross-shaped structuring element
- CV_SHAPE_CUSTOM: custom structuring element

```
void dilate(
      InputArray src,
      OutputArray dst,
      InputArray element,
      Point anchor=Point(-1,-1),
      int iterations=1,
      int borderType=BORDER_CONSTANT,
      const Scalar& borderValue =
                        morphologyDefaultBorderValue()
)
```

$$\texttt{dst}(x,y) = \max_{(x',y'):\,\texttt{element}(x',y')\neq 0} \texttt{src}(x+x',y+y')$$

```
void erode(
      InputArray src,
      OutputArray dst,
      InputArray element,
      Point anchor=Point(-1,-1),
      int iterations=1,
      int borderType=BORDER_CONSTANT,
      const Scalar& borderValue =
                        morphologyDefaultBorderValue()
)
```

$$\texttt{dst}(x,y) = \min_{(x',y'):\,\texttt{element}(x',y')\neq 0} \texttt{src}(x+x',y+y')$$

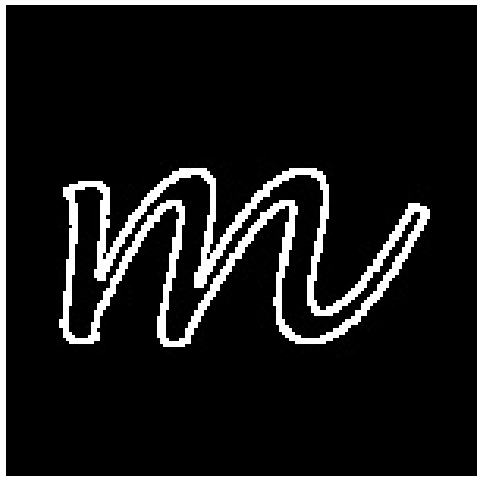
```
void morphologyEx(
      InputArray src,
      OutputArray dst,
      int op,
      InputArray kernel,
      Point anchor=Point(-1,-1),
      int iterations=1,
      int borderType=BORDER_CONSTANT,
      const Scalar& borderValue =
                        morphologyDefaultBorderValue()
)
```

op

- MORPH_OPEN - an opening operation
- MORPH_CLOSE - a closing operation
- MORPH_GRADIENT - a morphological gradient
  $(f \oplus b) - (f \ominus b)$
- MORPH_TOPHAT - "top hat" $f - (f \circ b)$
- MORPH_BLACKHAT - "black hat" $(f \bullet b) - f$

dilation

erosion



gradient

dilation

erosion

```
void Dilation( const Mat &image, Mat &result, int type=0, int size=3 );
void Erosion( const Mat &image, Mat &result, int type=0, int size=3 );
void Morphology( const Mat &image, Mat &result, int op, int type=0, int size=3 );

int main(void){
     Mat image = imread( "text_m.bmp", -1 );
     if( image.data == NULL ) return -1;

     Mat dilation;
     Dilation( image, dilation );

     Mat erosion;
     Erosion( image, erosion );

     Mat morphology;
     Morphology( image, morphology, 2 );

     // Display the images
     namedWindow( "Image" );
     namedWindow( "Dilation" );
     namedWindow( "Erosion" );
     namedWindow( "Morphology" );
     imshow( "Image", image );
     imshow( "Dilation", dilation );
     imshow( "Erosion", erosion );
     imshow( "Morphology", morphology );

     waitKey();

     return 0;
}
```

```
void Dilation( const Mat &image, Mat &result, int type, int size )
{
        // allocate if necessary
       result.create( image.size(), image.type() );

       int dilation_type;
       if( type == 0 )
              dilation_type = MORPH_RECT;
       else if( type == 1 )
              dilation_type = MORPH_CROSS;
       else if( type == 2)
              dilation_type = MORPH_ELLIPSE;

       Mat element = getStructuringElement(
                        dilation_type, Size(size, size) );

       // Apply the dilation operation
       dilate( image, result, element );
}
```

```
void Erosion( const Mat &image, Mat &result, int type, int size )
{
        // allocate if necessary
       result.create( image.size(), image.type() );

       int erosion_type;
       if( type == 0 )
                erosion_type = MORPH_RECT;
       else if( type == 1 )
                erosion_type = MORPH_CROSS;
       else if( type == 2)
                erosion_type = MORPH_ELLIPSE;

       Mat element = getStructuringElement(
                       erosion_type, Size(size, size) );

       // Apply the dilation operation
       erode( image, result, element );
}
```

```
void Morphology( const Mat &image, Mat &result, int op, int type, int size )
{
      // allocate if necessary
     result.create( image.size(), image.type() );

     int operation;
     switch( op )
     {
     case 0:
          operation = MORPH_OPEN;
          break;
     case 1:
          operation = MORPH_CLOSE;
          break;
     case 2:
          operation = MORPH_GRADIENT;
          break;
     case 3:
          operation = MORPH_TOPHAT;
          break;
     case 4:
          operation = MORPH_BLACKHAT;
          break;
     }
```

```
        int mor_type;
        if( type == 0 )
                mor_type = MORPH_RECT;
        else if( type == 1 )
                mor_type = MORPH_CROSS;
        else if( type == 2)
                mor_type = MORPH_ELLIPSE;

        Mat element = getStructuringElement( mor_type, Size(size, size) );

        // Apply the specified morphology operation
        morphologyEx( image, result, operation, element );
}
```

Reference

- R. Gonzalez, R. Woods, **Digital Image Processing (2nd Edition),** Prentice Hall, 2002
- Scott E Umbaugh, **Computer Imaging**, CRC Press, 2005
- R. Laganière, **OpenCV2 Computer Vision: Application Programming Cookbook,** PACKT Publishing, 2011
- http://docs.opencv.org