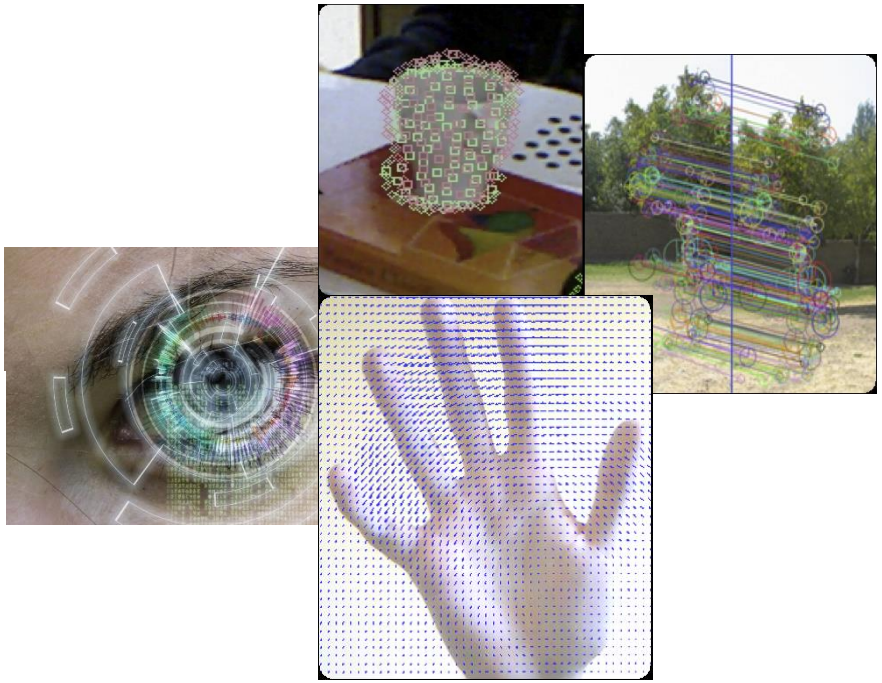
IVPL
Intelligent Vision Processing Lab

2023 Fall

COMPUTER VISION 비젼 프로그래밍



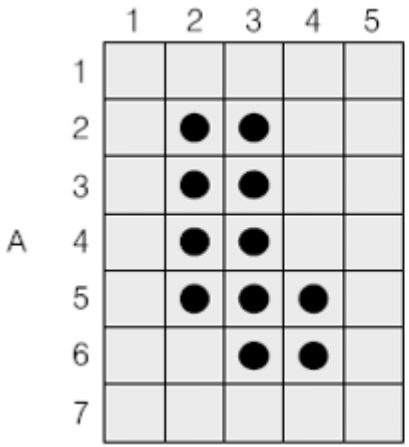
7강. Morphological processing (Practice)

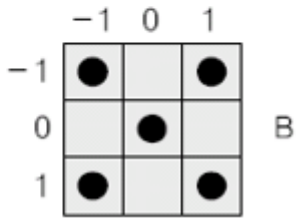
Simple Review of Morphological Operations

- 침식(erosion): 교환/결합 법칙 성립?

  A-B(A by B) = B-A (B by A)

  => No!!!!





- 팽창(dilation): 교환/결합 법칙 성립?

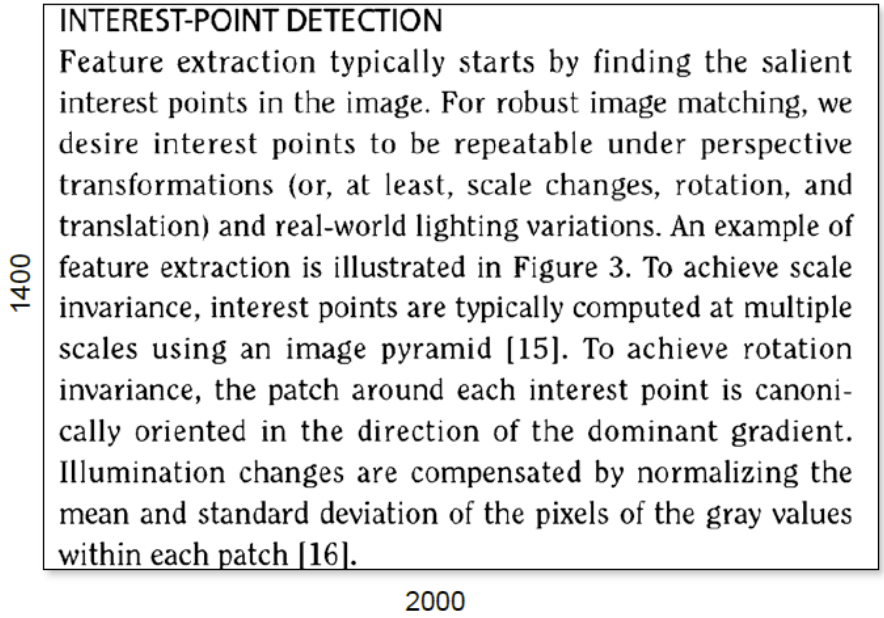
$$A \oplus B = B \oplus A$$

  => Yes !!!!

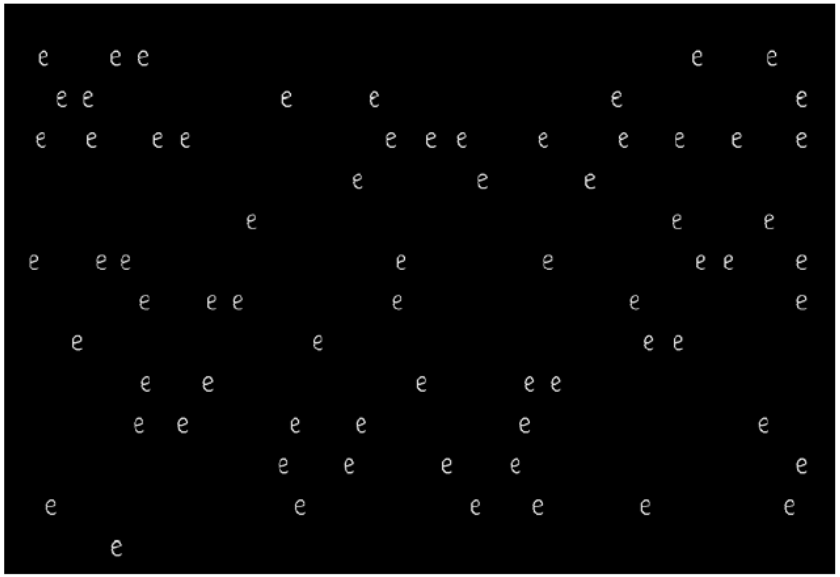
Some Applications

- Recognition by erosion

Binary image f



$$open\left(NOT\left[f\right],W\right) = dilate\left(erode\left(NOT\left[f\right],W\right),W\right)$$

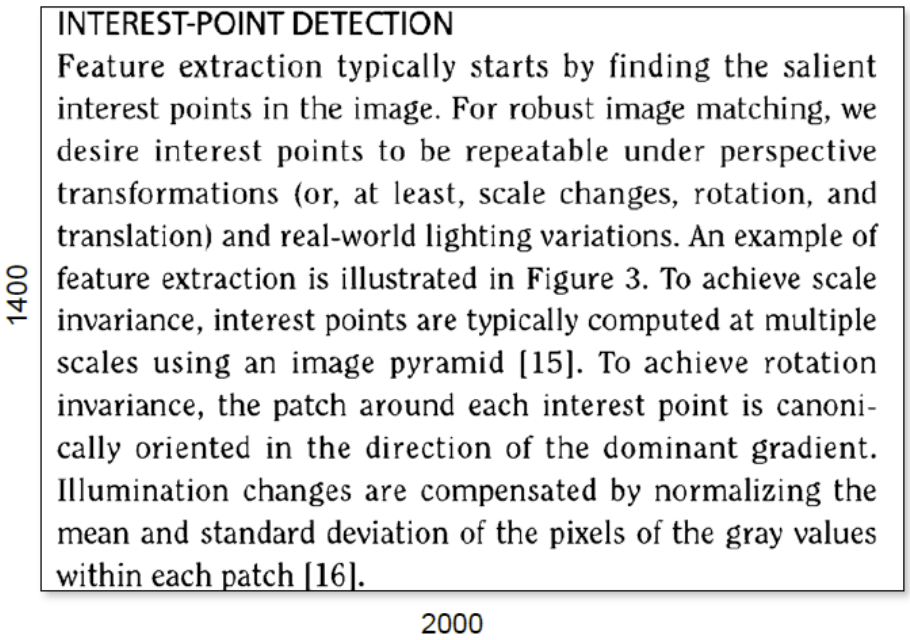


Structuring element W

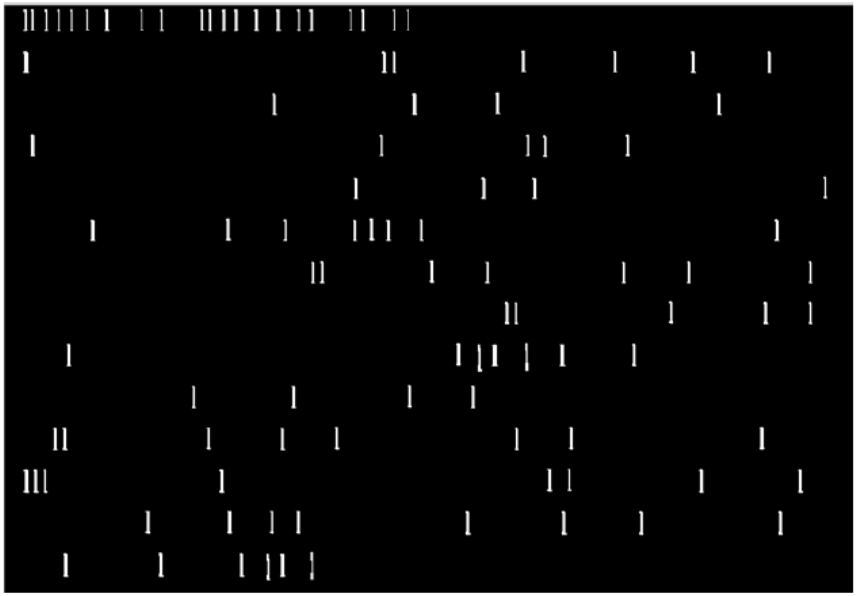
Some Applications

- Recognition by erosion

Binary image f



$$open\left(NOT\left[f\right],W\right) = dilate\left(erode\left(NOT\left[f\right],W\right),W\right)$$



Structuring element W

Some Applications

- Flat dilation with different structuring elements

Original


Diamond


Disk

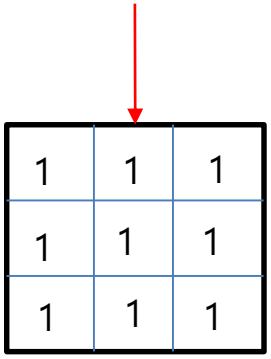

20 degree line


9 points

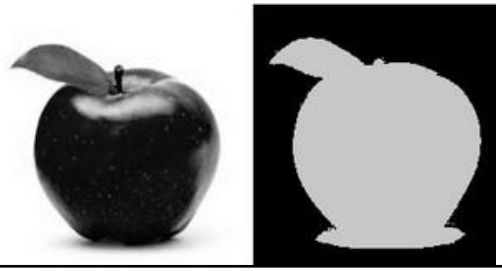

2 horizontal lines

Morphology (형태학)처리의 필요한 기본 요소

- 두 가지 필수 요소
  - 처리할 영상 → 많은 경우 **이진화된 영상** : threshold API 사용 가능
  - **형태소(structuring element) 정의** 필요 : Mat element(3,3, CV_8U, Scalar(1));



- Threshoding: 영상의 영역을 n개로 분할
  - double threshold(InputArray **src**, OutputArray **dst**,
    double **thresh**, double **maxval**, int **type**)



Parameters:

•src – input array (single-channel, 8-bit or 32-bit floating point).
•dst – output array of the same size and type as src.
•thresh – threshold value.
•maxval – maximum value to use with the THRESH_BINARY and THRESH_BINARY_INV thresholding types.
•type – thresholding type (see the details below).→**THRESH_BINARY 사용**

Morphology (형태학)처리의 필요한 기본 요소

- Thresholding: 영상의 영역을 n개로 분할
  - void adaptiveThreshold(InputArray **src**, OutputArray **dst**, double **maxValue**, int **adaptiveMethod**, int **thresholdType**, int **blockSize**, double **C**)

Parameters:	•src – Source 8-bit single-channel image. •dst – Destination image of the same size and the same type as src . •maxValue – Non-zero value assigned to the pixels for which the condition is satisfied. See the details below. •**adaptiveMethod** – Adaptive thresholding algorithm to use, ADAPTIVE_THRESH_MEAN_C or ADAPTIVE_THRESH_GAUSSIAN_C . See the details below. •thresholdType – Thresholding type that must be either THRESH_BINARY or THRESH_BINARY_INV . •blockSize – Size of a pixel neighborhood that is used to calculate a threshold value for the pixel: 3, 5, 7, and so on. •C – Constant subtracted from the mean or weighted mean (see the details below). Normally, it is positive but may be zero or negative as well.

Morphology (형태학)처리의 필요한 기본 요소

- threshold() 기본 예제
  - Trackbar를 활용하자.....!!!

```
///--- Global variables----///
int threshold_value = 0;
int threshold_type = 3;
int const max_value = 255;
int const max_type = 4;
int const max_BINARY_value = 255;

Mat image, src_gray, dst;
char* window_name = "Threshold Demo";

char* trackbar_type = "Type: ₩n 0: Binary ₩n 1: Binary Inverted ₩n 2: Truncate ₩n 3: To Zero ₩n 4: To Zero Inverted";
char* trackbar_value = "Value";

void Threshold_Demo( int, void* );

int main( int argc, char** argv )
{
          /// Load image
           image = imread( "Desert.bmp", 1);    // Read the file

            if (image.empty()){                              // Check for invalid input
                     cout << "Could not open or find the image" << std::endl;
                     return -1;
            }
            namedWindow("Display window", WINDOW_AUTOSIZE);  // Create a window for display.
            imshow("Display window", image );

         /// Convert the image to Gray
           cvtColor( image, src_gray, CV_BGR2GRAY );
```

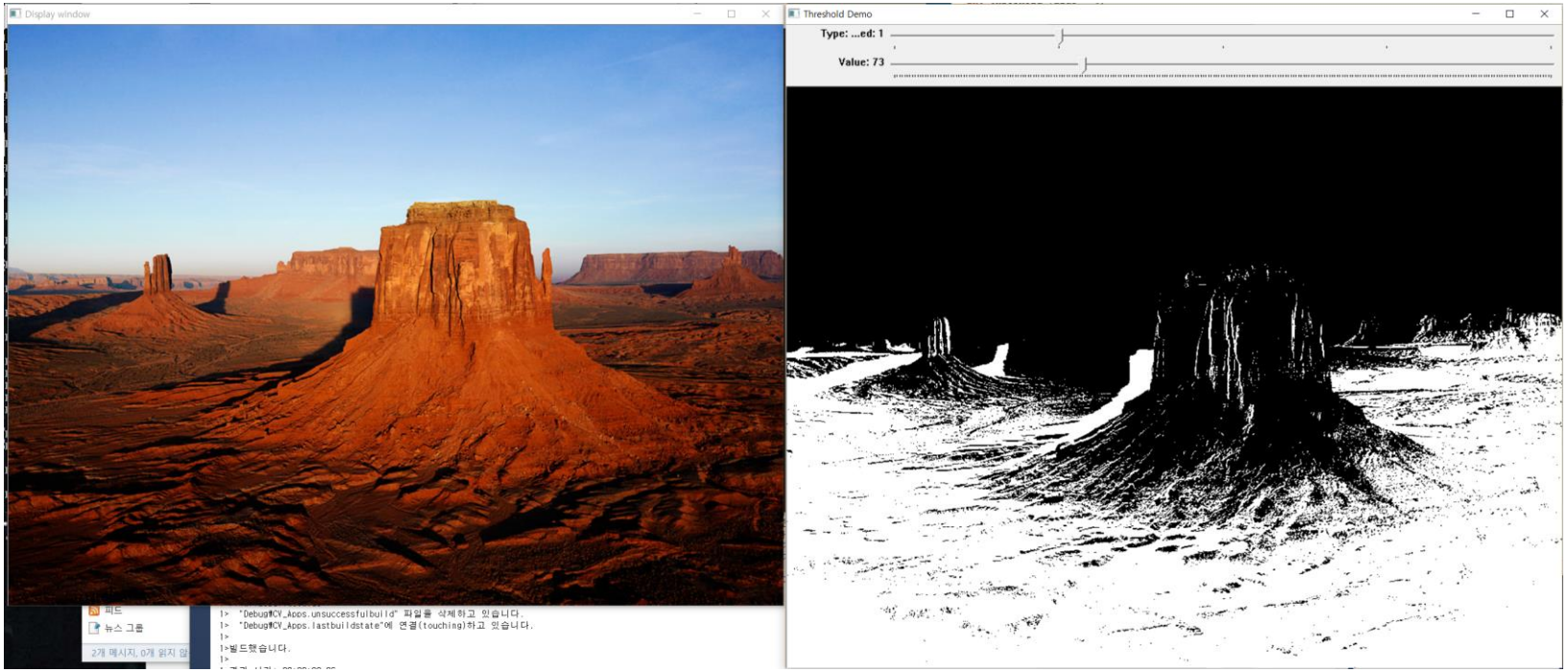
(계속)

Morphology (형태학)처리의 필요한 기본 요소

```
        /// Create a window to display results
          namedWindow( window_name, CV_WINDOW_AUTOSIZE );

          /// Create two Trackbars to choose type of Threshold on one window
          createTrackbar( trackbar_type,    window_name, &threshold_type,    max_type, Threshold_Demo );

          createTrackbar( trackbar_value,    window_name, &threshold_value,    max_value, Threshold_Demo );

          /// Call the function to initialize
          Threshold_Demo( 0, 0 );

          /// Wait until user finishes program
          while(true)
          {
                int c;
                c = waitKey( 20 );
                if( (char)c == 27 )
                                { break; }

        }
}
void Threshold_Demo( int, void* )
{
  /* 0: Binary      1: Binary Inverted      2: Threshold Truncated      3: Threshold to Zero  4: Threshold to Zero Inverted  */

  threshold( src_gray, dst, threshold_value, max_BINARY_value, threshold_type );

  imshow( window_name, dst );
}
```

Morphology (형태학)처리의 필요한 기본 요소

- 실행 결과: trackbar 스크롤에 따라서 이진화 영상을 출력함



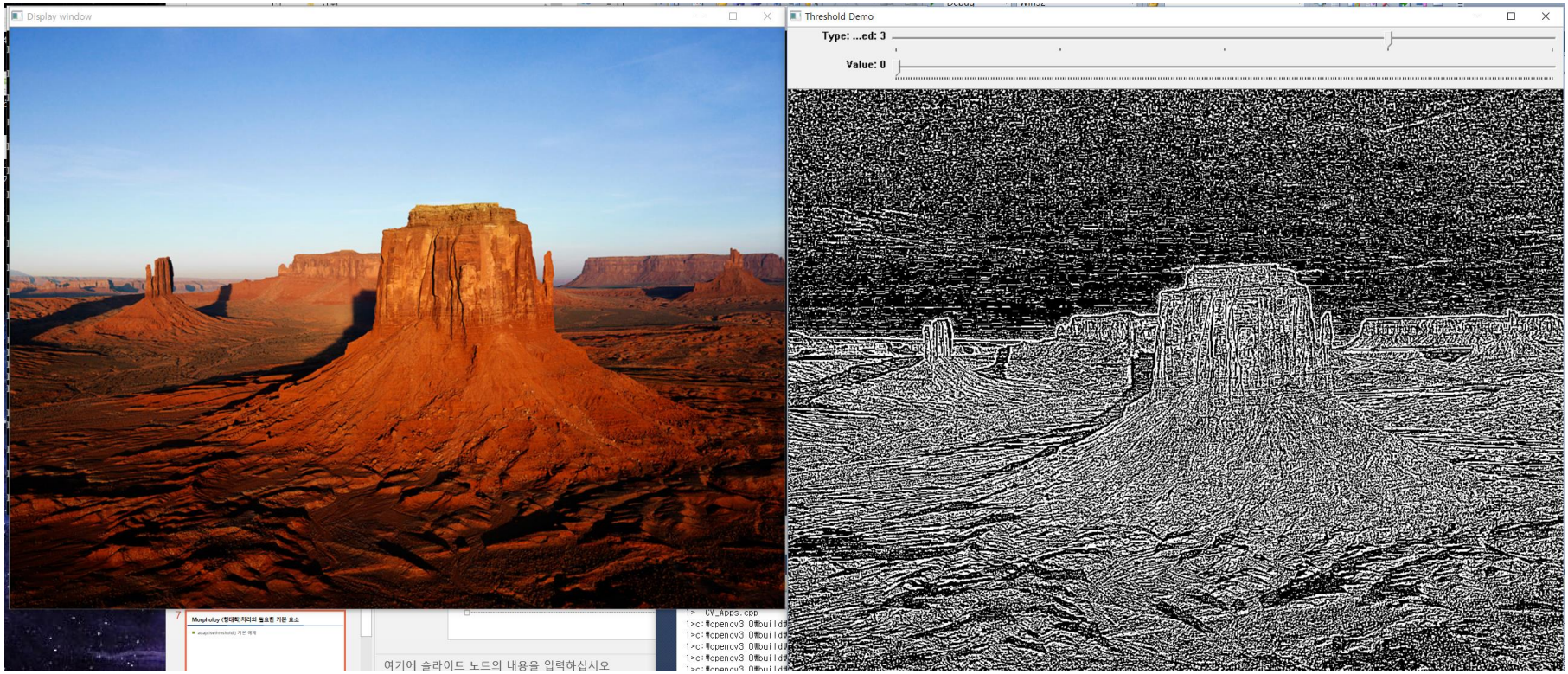
Morphology (형태학)처리의 필요한 기본 요소

- adaptivethreshold() 기본 예제

```
        (동일) ~~

      /// Create a window to display results
        namedWindow( window_name, CV_WINDOW_AUTOSIZE );

        /// Create Trackbar to choose type of Threshold
        createTrackbar( trackbar_type,    window_name, &threshold_type,    max_type, Threshold_Demo );

        createTrackbar( trackbar_value,    window_name, &threshold_value,    max_value, Threshold_Demo );

        /// Call the function to initialize
        Threshold_Demo( 0, 0 );

        /// Wait until user finishes program
        while(true)
        {
                int c;
                c = waitKey( 20 );
                if( (char)c == 27 )
                                { break; }
        }
}
void Threshold_Demo( int, void* )
{
  /* 0: Binary      1: Binary Inverted      2: Threshold Truncated      3: Threshold to Zero  4: Threshold to Zero Inverted  */

  //threshold( src_gray, dst, threshold_value, max_BINARY_value,threshold_type );
   adaptiveThreshold(src_gray, dst, max_value, ADAPTIVE_THRESH_MEAN_C, THRESH_BINARY, 5, 0);

  imshow( window_name, dst );
}
```

Morphology (형태학)처리의 필요한 기본 요소

- 실행 결과: 일정 블록 크기의 평균값을 이용하여 영상 이진화 수행(스크롤 바는 영향 없음)



Morphology (형태학)처리 연산들

if Mat element(7,7,CV_8U, Scalar(1));, then ← structure element(형태소)

- Erosion(침식)
  - Erode(cv::Mat **src**, cv::Mat **dst**, element);
  - Erode(cv::Mat **src**, cv::Mat **dst**, Mat()); ← default로 3x3으로 셋팅됨

- Dilation(팽창)
  - dilate(cv::Mat **src**, cv::Mat **dst**, element);

- 반복 횟수 지정 가능
  - erode(cv::Mat **src**, cv::Mat **dst**, Mat(), Point(-1,-1), **3**); //3x3으로 세 번 수행함

Morphology (형태학)처리 연산들: Dilation(팽창)

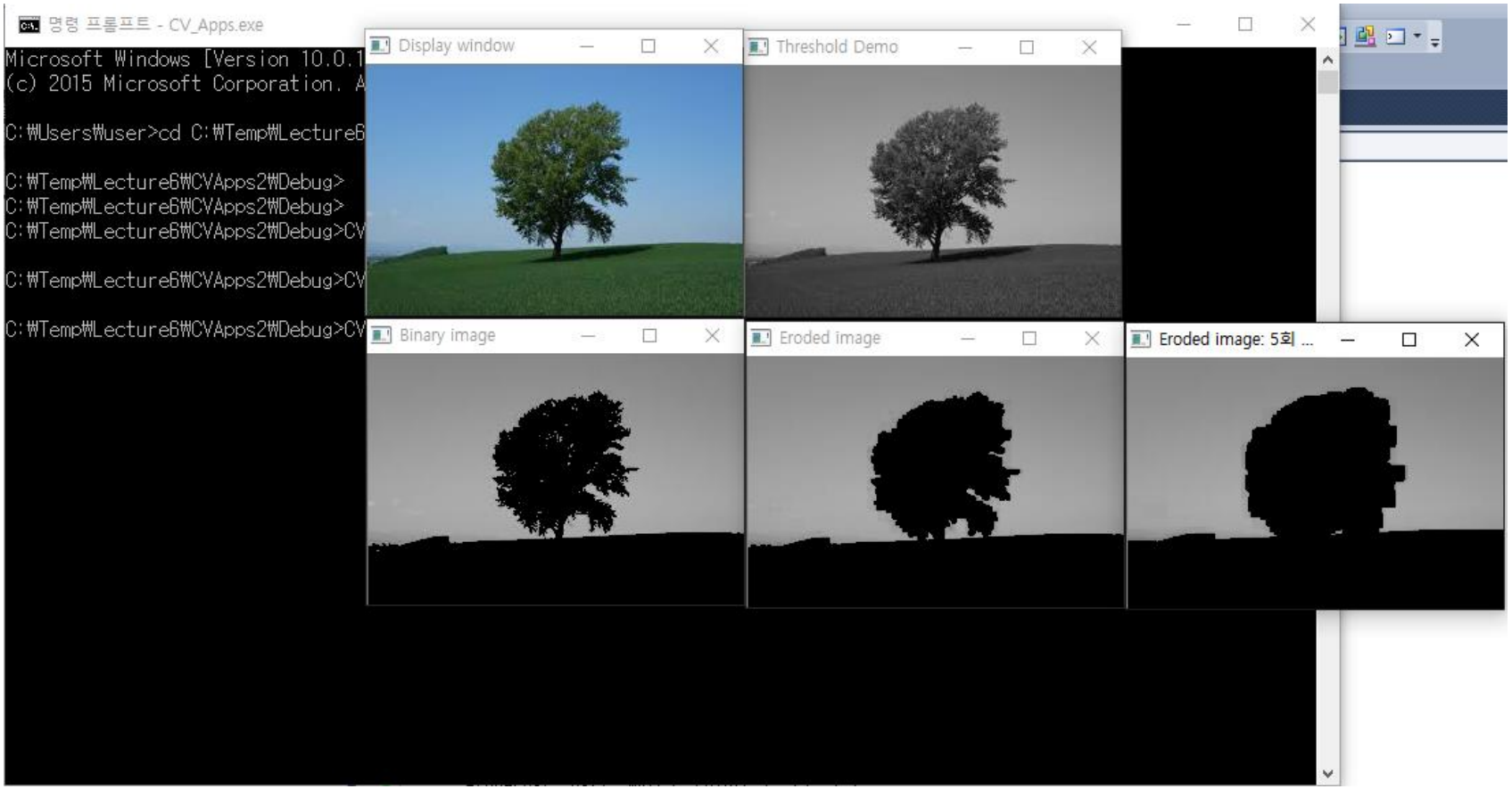
- Erosion 기본 예제

(동일) ~~

```
/// Convert the image to Gray
cvtColor( image, src_gray, CV_BGR2GRAY );

/// Create a window to display results
namedWindow( window_name, CV_WINDOW_AUTOSIZE );
imshow(window_name, src_gray);

threshold( src_gray, dst, threshold_value=120, max_BINARY_value,threshold_type );

namedWindow( "Binary image", CV_WINDOW_AUTOSIZE );
imshow("Binary image", dst);

erode(dst, dst1, Mat());
namedWindow( "Eroded image", CV_WINDOW_AUTOSIZE );
imshow("Eroded image", dst1);

erode(dst, dst1, Mat(), Point(-1,-1), 5 );
namedWindow( "Eroded image: 5회 수행", CV_WINDOW_AUTOSIZE );
imshow("Eroded image: 5회 수행", dst1);

waitKey(0);
return 0;
}
```

Morphology (형태학)처리 연산들: Dilation(팽창)

- 실행 결과



Morphology (형태학)처리 연산들: Erosion(침식)

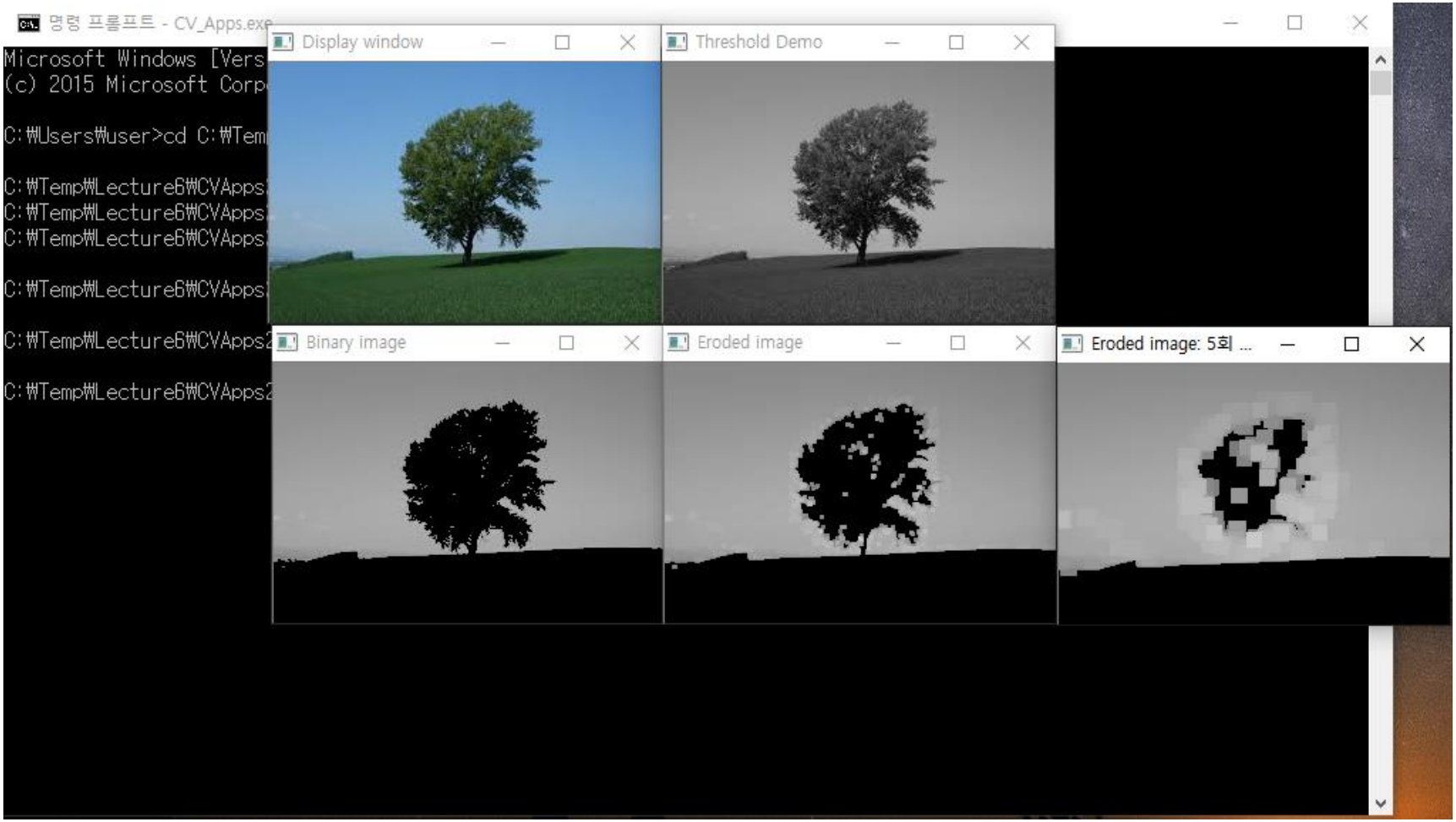
- Dilate 기본 예제

(동일) ~~

```
/// Convert the image to Gray
cvtColor( image, src_gray, CV_BGR2GRAY );

/// Create a window to display results
namedWindow( window_name, CV_WINDOW_AUTOSIZE );
imshow(window_name, src_gray);

threshold( src_gray, dst, threshold_value=120, max_BINARY_value,threshold_type );

namedWindow( "Binary image", CV_WINDOW_AUTOSIZE );
imshow("Binary image", dst);

dilate(dst, dst1, Mat());
namedWindow( "Eroded image", CV_WINDOW_AUTOSIZE );
imshow("Eroded image", dst1);

dilate(dst, dst1, Mat(), Point(-1,-1), 5 );
namedWindow( "Eroded image: 5회 수행", CV_WINDOW_AUTOSIZE );
imshow("Eroded image: 5회 수행", dst1);

waitKey(0);
return 0;
}
```

Morphology (형태학)처리 연산들: Erosion(침식)

- 실행 결과



Morphology (형태학)처리 연산들: Opening(열림)

- 고수준의 형태학적 함수 사용 가능

```
// Open the image
cv::Mat element3(3,3,CV_8U,cv::Scalar(1));
cv::Mat opened;
cv::morphologyEx(image,opened,cv::MORPH_OPEN,element3);

 // Display the opened image
cv::namedWindow("Opened Image");
cv::imshow("Opened Image", opened);
```

Morphology (형태학)처리 연산들: Closing(닫힘)

- 고수준의 형태학적 함수 사용 가능

```
// Close the image
cv::Mat element5(5,5,CV_8U,cv::Scalar(1));
cv::Mat closed;
cv::morphologyEx(image, closed, cv::MORPH_CLOSE, element5);
// Display the opened image
cv::namedWindow("Closed Image");
cv::imshow("Closed Image", closed);
```

Morphology (형태학)처리 응용: thinning(세선화) (1)

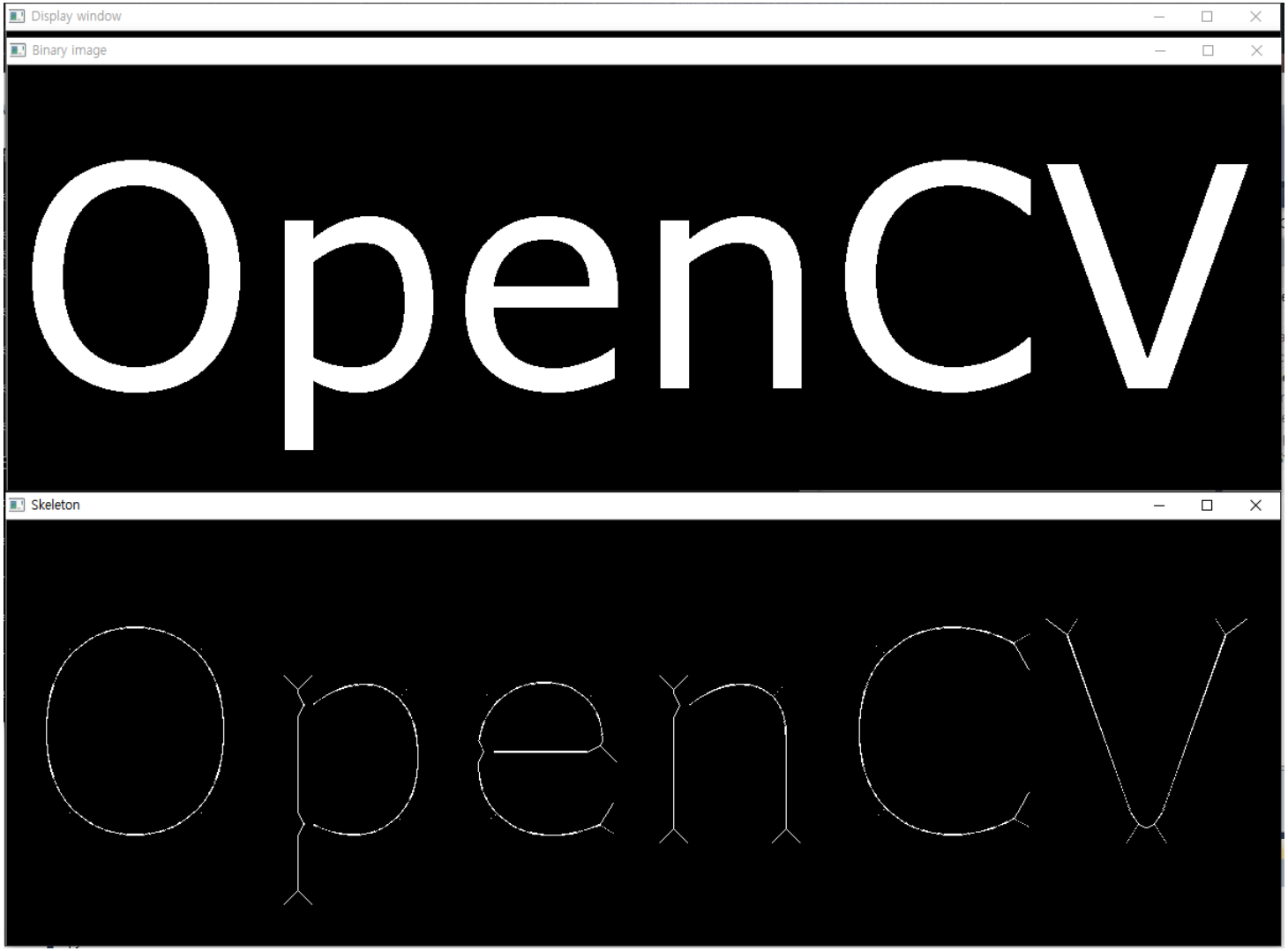
```
Int main(arg, arg*){
        (원본 영상 열기) ~~

      threshold( image, image, threshold_value=120, max_BINARY_value,THRESH_BINARY );

        namedWindow( "Binary image", CV_WINDOW_AUTOSIZE );
        imshow("Binary image", image);

        Mat skel(image.size(), CV_8UC1, cv::Scalar(0));
        Mat temp(image.size(), CV_8UC1);

        cv::Mat element = cv::getStructuringElement(cv::MORPH_CROSS, cv::Size(3, 3));

        bool done;
        do{
                cv::morphologyEx(image, temp, cv::MORPH_OPEN, element);
                cv::bitwise_not(temp, temp);
                cv::bitwise_and(image, temp, temp);
                cv::bitwise_or(skel, temp, skel);
                cv::erode(image, image, element);

                double max;
                cv::minMaxLoc(image, 0, &max);
                done = (max == 0);
        } while (!done);
```

Morphology (형태학)처리 응용: thinning(세선화) (1)

```
    (계속)
    imshow("Skeleton", skel);

    waitKey(0);

    return 0;

}
```

Morphology (형태학)처리 응용: thinning(세선화)

- 수행 결과: 뼈대만 구할 수 있음



Morphology (형태학)처리 응용: thinning(세선화) (2)

- Ximgproc API 사용 방법

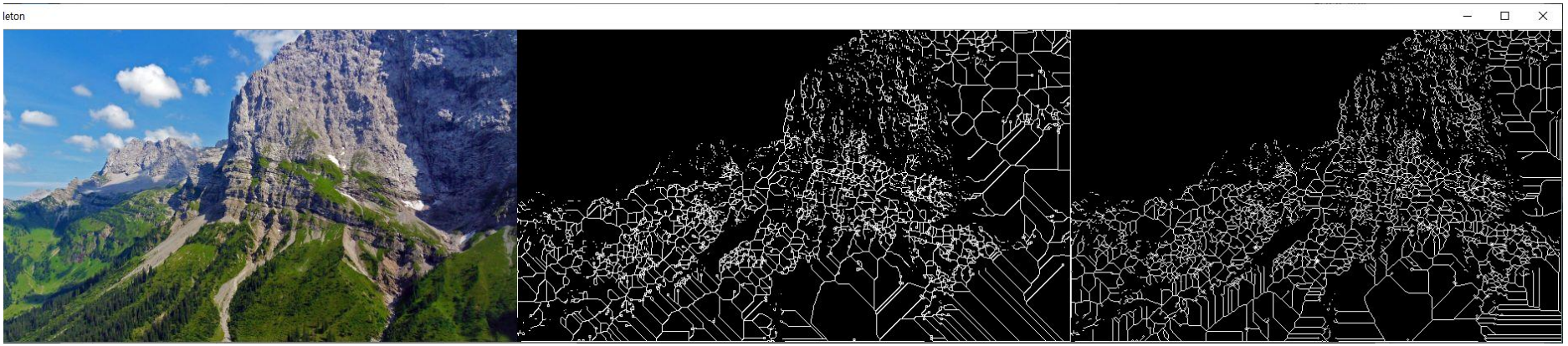
```
#include <opencv2/core.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/xfeatures2d.hpp>
#include <opencv2/imgproc.hpp>
#include "opencv2/ximgproc.hpp"
#include <iostream>
#include <stdio.h>

using namespace cv;
using namespace std;

int main(int argc, char** argv) {
//--OpenCV 2.x 구현 --//
Mat src, dst, tmp;

char window_name[20] = "Pyramids Demo";

char* filename = argv[1];

//image = imread("Desert.bmp", IMREAD_COLOR);   // Read the file
src = imread(filename, IMREAD_COLOR);   // Read the file

/// Threshold the input image
Mat img_grayscale, img_binary;

cvtColor(src, img_grayscale, COLOR_BGR2GRAY);
threshold(img_grayscale, img_binary, 0, 255, THRESH_OTSU | THRESH_BINARY_INV);
```

Morphology (형태학)처리 응용: thinning(세선화) (2)

```
    /// Apply thinning to get a skeleton
    Mat img_thinning_ZS, img_thinning_GH;
    ximgproc::thinning(img_binary, img_thinning_ZS, ximgproc::THINNING_ZHANGSUEN);
    ximgproc::thinning(img_binary, img_thinning_GH, ximgproc::THINNING_GUOHALL);

    /// Make 3 channel images from thinning result
    Mat result_ZS(src.rows, src.cols, CV_8UC3), result_GH(src.rows, src.cols, CV_8UC3);

    Mat in[] = { img_thinning_ZS, img_thinning_ZS, img_thinning_ZS };
    Mat in2[] = { img_thinning_GH, img_thinning_GH, img_thinning_GH };
    int from_to[] = { 0,0, 1,1, 2,2 };
    mixChannels(in, 3, &result_ZS, 1, from_to, 3);
    mixChannels(in2, 3, &result_GH, 1, from_to, 3);

    /// Combine everything into a canvas
    Mat canvas(src.rows, src.cols * 3, CV_8UC3);
    src.copyTo(canvas(Rect(0, 0, src.cols, src.rows)));
    result_ZS.copyTo(canvas(Rect(src.cols, 0, src.cols, src.rows)));
    result_GH.copyTo(canvas(Rect(src.cols * 2, 0, src.cols, src.rows)));

    /// Visualize result
    imshow("Skeleton", canvas); waitKey(0);

    return 0;
}
```

Morphology (형태학)처리 응용: thinning(세선화) (2)

- result

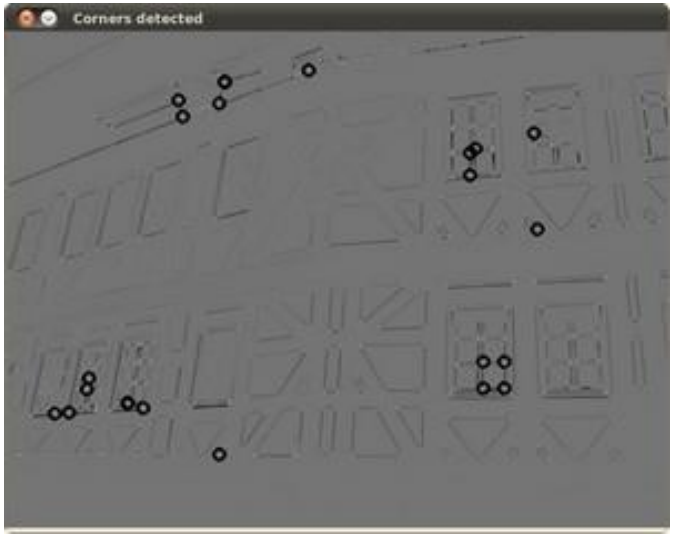



```
void cv::ximgproc::thinning ( InputArray  src,
                              OutputArray dst,
                              int         thinningType = THINNING_ZHANGSUEN
                            )
```

thinningType = { THINNING_ZHANGSUEN or 0
THINNING_GUOHALL or 1

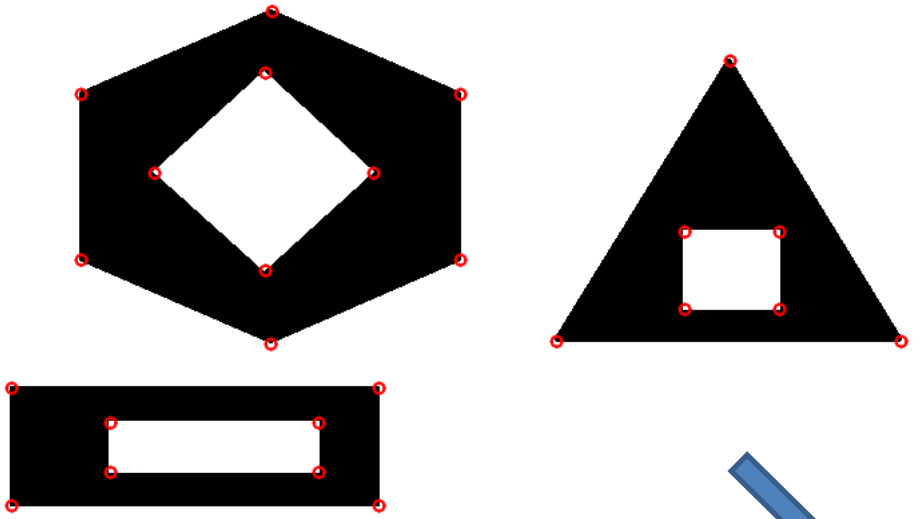
Morphology (형태학)처리 응용: Corner Detection (코너 검출)

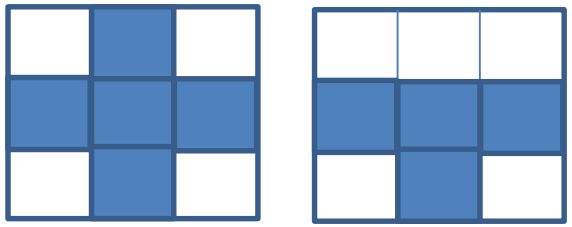
- Corner의 정의
  - **A corner can be defined as the intersection of two edges**. A corner can also be defined as a point for which there are two dominant and different edge directions in a local neighbourhood of the point.
  - Basically, **edge line or object boundary** information is needed to get the corner points.



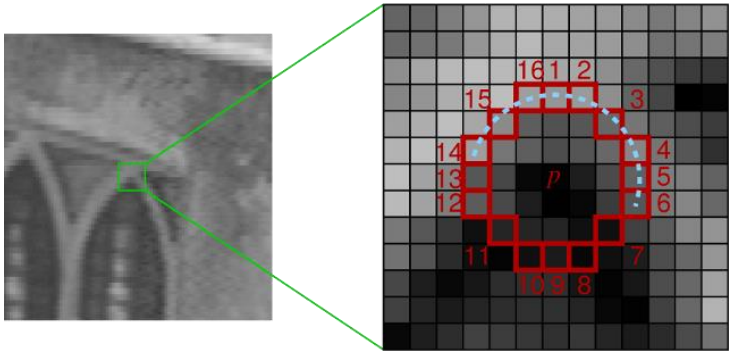
Morphology (형태학)처리 응용: Corner Detection (코너 검출)

- Corner의 다양한 형태





structure element?



Morphology (형태학)처리 응용: Corner Detection (코너 검출)

- Morphology (형태학) 필터를 이용한 코너 검출
  - 이진 영상 사용

```
//--Class for Corner detection ---//
class MorphoFeatures {

        private:

        // threshold to produce binary image
        int threshold;
        // structuring elements used in corner detection
        cv::Mat cross;
        cv::Mat diamond;
        cv::Mat square;
        cv::Mat x;

        void applyThreshold(cv::Mat& result) {

                // Apply threshold on result
                if (threshold>0)
                        cv::threshold(result, result, threshold, 255, cv::THRESH_BINARY_INV);
                }
(계속)
```

Morphology (형태학)처리 응용: Corner Detection (코너 검출)

```
public:

    MorphoFeatures() : threshold(-1), cross(5,5,CV_8U,cv::Scalar(0)),
                                diamond(5,5,CV_8U,cv::Scalar(1)),
                               square(5,5,CV_8U,cv::Scalar(1)),
                               x(5,5,CV_8U,cv::Scalar(0)){

            // Creating the cross-shaped structuring element
            for (int i=0; i<5; i++) {
                               cross.at<uchar>(2,i)= 1;
                               cross.at<uchar>(i,2)= 1;
            }

            // Creating the diamond-shaped structuring element
            diamond.at<uchar>(0,0)= 0;
            diamond.at<uchar>(0,1)= 0;
            diamond.at<uchar>(1,0)= 0;
            diamond.at<uchar>(4,4)= 0;
            diamond.at<uchar>(3,4)= 0;
            diamond.at<uchar>(4,3)= 0;
            diamond.at<uchar>(4,0)= 0;
            diamond.at<uchar>(4,1)= 0;
            diamond.at<uchar>(3,0)= 0;
            diamond.at<uchar>(0,4)= 0;
            diamond.at<uchar>(0,3)= 0;
            diamond.at<uchar>(1,4)= 0;

            // Creating the x-shaped structuring element
            for (int i=0; i<5; i++) {

                        x.at<uchar>(i,i)= 1;
                        x.at<uchar>(4-i,i)= 1;

        }
}
```

(계속)

Morphology (형태학)처리 응용: Corner Detection (코너 검출)

```
void setThreshold(int t) {

        threshold= t;
}

int getThreshold() const {

        return threshold;
}

cv::Mat getEdges(const cv::Mat &image) {

        // Get the gradient image
        cv::Mat result;
        cv::morphologyEx(image,result,cv::MORPH_GRADIENT,cv::Mat());

        // Apply threshold to obtain a binary image
        applyThreshold(result);

        return result;
}
```

(계속)

Morphology (형태학)처리 응용: Corner Detection (코너 검출)

```
cv::Mat getCorners(const cv::Mat &image) {

        cv::Mat result;

        // Dilate with a cross
        cv::dilate(image,result,cross);

        // Erode with a diamond
        cv::erode(result,result,diamond);

        cv::Mat result2;
        // Dilate with a X
        cv::dilate(image,result2,x);

        // Erode with a square
        cv::erode(result2,result2,square);

        // Corners are obtained by differencing
        // the two closed images
        cv::absdiff(result2,result,result);

        // Apply threshold to obtain a binary image
        applyThreshold(result);

        return result;
  }

void drawOnImage(const cv::Mat& binary, cv::Mat& image) {

        cv::Mat_<uchar>::const_iterator it= binary.begin<uchar>();
        cv::Mat_<uchar>::const_iterator itend= binary.end<uchar>();

        // for each pixel
        for (int i=0; it!= itend; ++it,++i) {
                if (!*it)
                cv::circle(image,cv::Point(i%image.step,i/image.step),5,cv::Scalar(255,0,0));
        }
  }
};
```

Morphology (형태학)처리 응용: Corner Detection (코너 검출)

- Main() 함수 구현

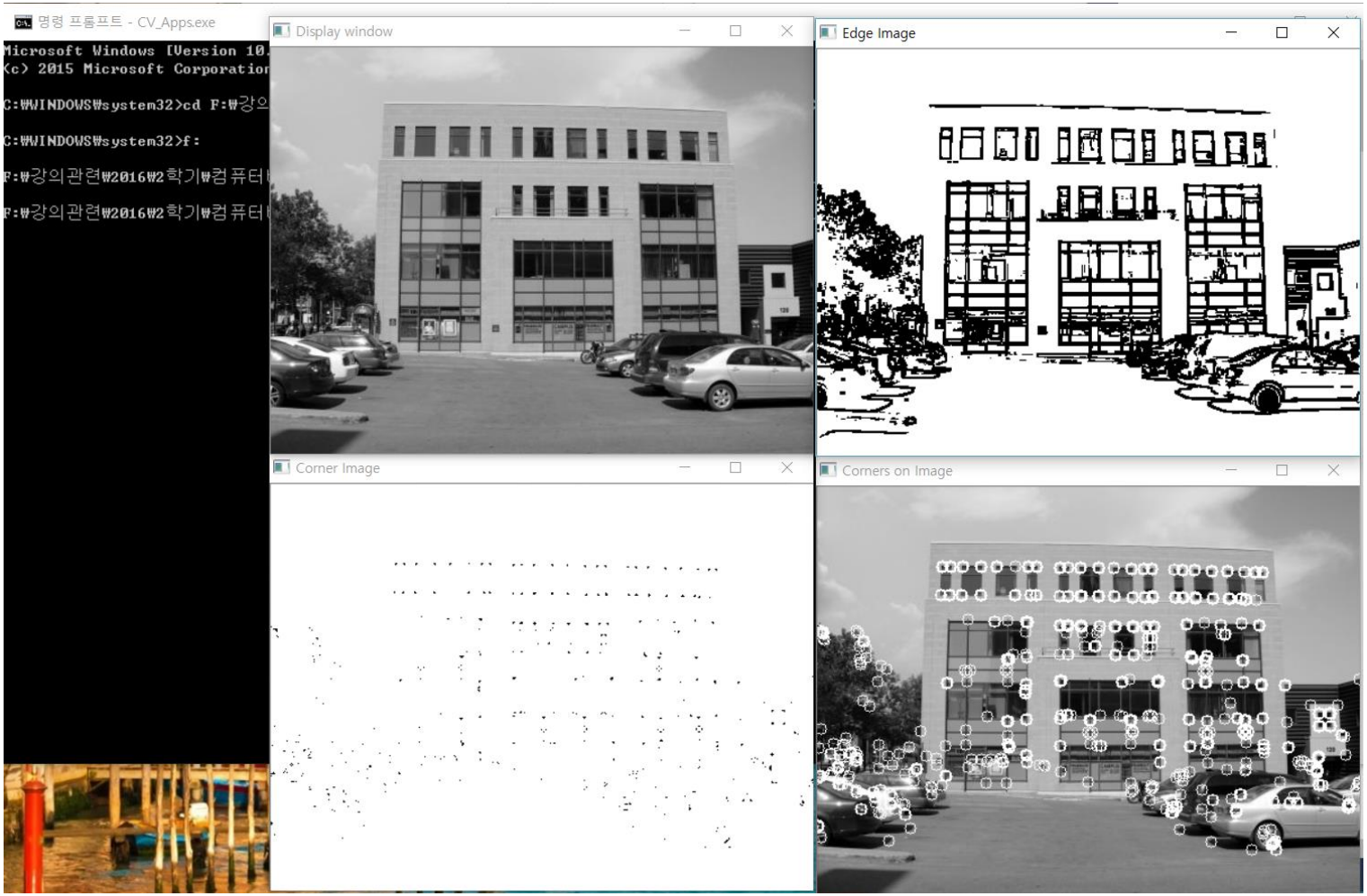
```
int main( int argc, char** argv )
{

        /// Load image
        image = imread( "building.jpg", 0);   // Read the file

        if (image.empty()){                   // Check for invalid input
        cout << "Could not open or find the image" << std::endl;
        return -1;
        }
        // Create a window for display.
        namedWindow("Display window", WINDOW_AUTOSIZE);
        imshow("Display window", image );

        //--클래스 객체 선언 --//
        MorphoFeatures morpho;

        morpho.setThreshold(40);


        //--에지 가져오기 --//
        // Get the edges
    Mat edges;
    edges= morpho.getEdges(image);

    // Display the edge image
    namedWindow("Edge Image");
    imshow("Edge Image",edges);

    //-- Get the corners--//
    morpho.setThreshold(-1);
    cv::Mat corners;
    corners= morpho.getCorners(image);
    cv::morphologyEx(corners,corners,cv::MORPH_TOPHAT,cv::Mat());
    cv::threshold(corners, corners, 40, 255, cv::THRESH_BINARY_INV);
```

(계속)

```
// Display the corner image
cv::namedWindow("Corner Image");
cv::imshow("Corner Image",corners);

// Display the corner on the image
morpho.drawOnImage(corners,image);
cv::namedWindow("Corners on Image");
cv::imshow("Corners on Image",image);

waitKey(0);

return 0;
}
```

Morphology (형태학)처리 응용: Corner Detection (코너 검출)

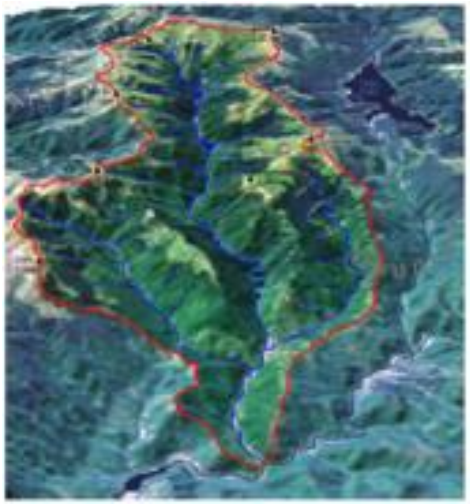
- 수행 결과: 주요 코너 지점을 검출 함

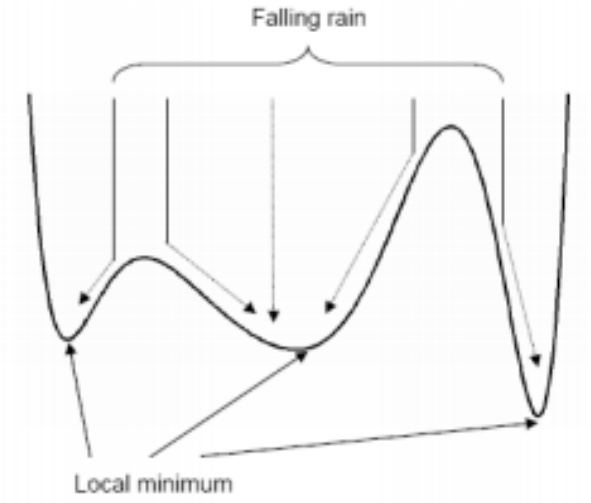


Morphology (형태학)처리 응용: Watershed 분할 기법

- Watershed(워터세드) segmentation concept: 영상을 지형에 맵핑

비가 온다면?

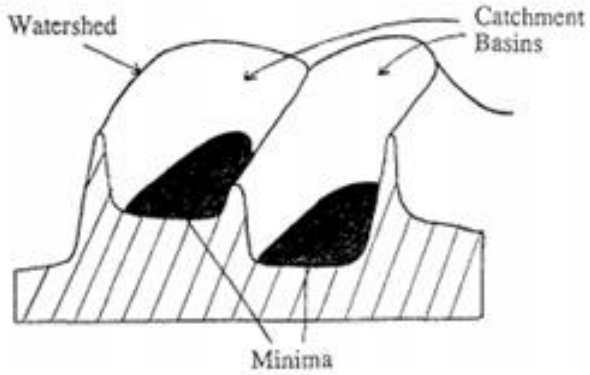




- 높은 값을 갖는 픽셀들은 봉우리(peak) 또는 워터세드 라인(**watershed line**)으로 표현
- •낮은 값을 갖는 픽셀들은 골짜기(valley)또는 국부 최소값(**regional minimum**)으로 표현
- •영상 내의 픽셀들의 집합을 하나의 지형으로 간주하고 높낮이를 분석하는 방법

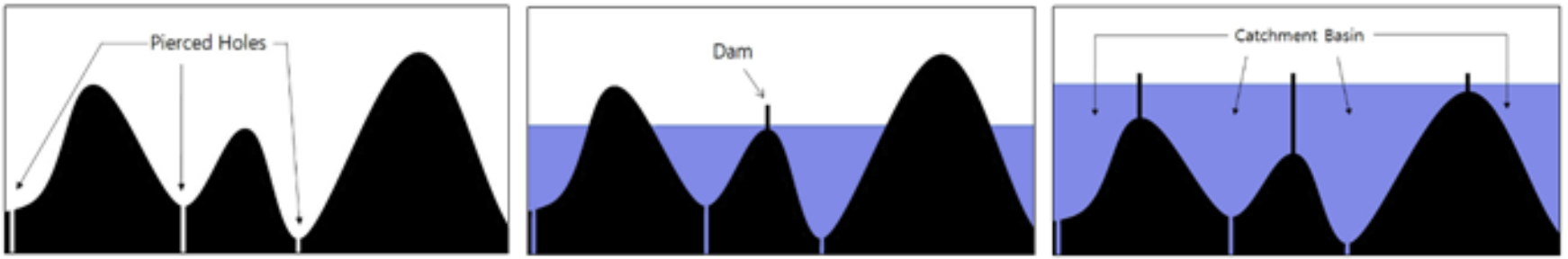
Morphology (형태학)처리 응용: Watershed 분할 기법

Rainfall 방식



- 기울기 계산에 의한 watershed line과 minia로 구별가능

Water flooding 방식



- 각각의 물 웅덩이의 국부 최소지점에 구멍을 뚫고 이 구멍으로부터 서서히 물을 채우는 기법.
- 두 지점이 만나면 댐(dam)을 생성해 주고 계속하여 최고 봉우리에 댐이 생성되면 종료

Morphology (형태학)처리 응용: Watershed 분할 기법

- Watershed 분할 예시 코드
  - void **watershed**(InputArray **image**, InputOutputArray **markers**)

Parameters:	• image – Input 8-bit 3-channel image. • markers – Input/output 32-bit single-channel image (map) of markers. It should have the same size as image .

Morphology (형태학)처리 응용: Watershed 분할 기법

- Watershed 분할 기법의 예제 코드

```
//----Watershed segmentation class --//
class WatershedSegmenter {

        private:

        cv::Mat markers;

        public:

      void setMarkers(const cv::Mat& markerImage) {

            // Convert to image of ints
            markerImage.convertTo(markers,CV_32S);
        }

        cv::Mat process(const cv::Mat &image) {

            // Apply watershed
            cv::watershed(image, markers);

            return markers;
        }
```

(계속)

```
      // Return result in the form of an image
        cv::Mat getSegmentation() {

            cv::Mat tmp;
            // all segment with label higher than 255
            // will be assigned value 255
            markers.convertTo(tmp,CV_8U);

            return tmp;
        }

        // Return watershed in the form of an image
        cv::Mat getWatersheds() {

            cv::Mat tmp;
            markers.convertTo(tmp,CV_8U,255,255);

            return tmp;
        }
};
```

Morphology (형태학)처리 응용: Watershed 분할 기법

```
int main( int argc, char** argv )
{
        Mat image, gray, binary;

        /// Load image
        image = imread( "test2.jpg", 1);    // Read the file

         if (image.empty()){                                    // Check for invalid input
         cout << "Could not open or find the image" << std::endl;
         return -1;
         }
         namedWindow("Display window", WINDOW_AUTOSIZE);
         imshow("Display window", image );

         cvtColor(image,gray,COLOR_BGR2GRAY);

         threshold(gray,binary, 120,255,THRESH_BINARY_INV);

      // Display the binary image
      cv::namedWindow("Binary Image");
      cv::imshow("Binary Image",binary);
```

Morphology (형태학)처리 응용: Watershed 분할 기법

(계속)

```
// Eliminate noise and smaller objects
cv::Mat fg;
cv::erode(binary, fg, cv::Mat(),cv::Point(-1,-1),6);

// Display the foreground image
cv::namedWindow("Foreground Image");
cv::imshow("Foreground Image", fg);

// Identify image pixels without objects
cv::Mat bg;
cv::dilate(binary, bg,cv::Mat(),cv::Point(-1,-1), 6);
cv::threshold(bg,bg,1,128,cv::THRESH_BINARY_INV);

// Display the background image
cv::namedWindow("Background Image");
cv::imshow("Background Image",bg);

// Show markers image
cv::Mat markers(binary.size(),CV_8U,cv::Scalar(0));
markers= fg+bg;
cv::namedWindow("Markers");
cv::imshow("Markers",markers);
```

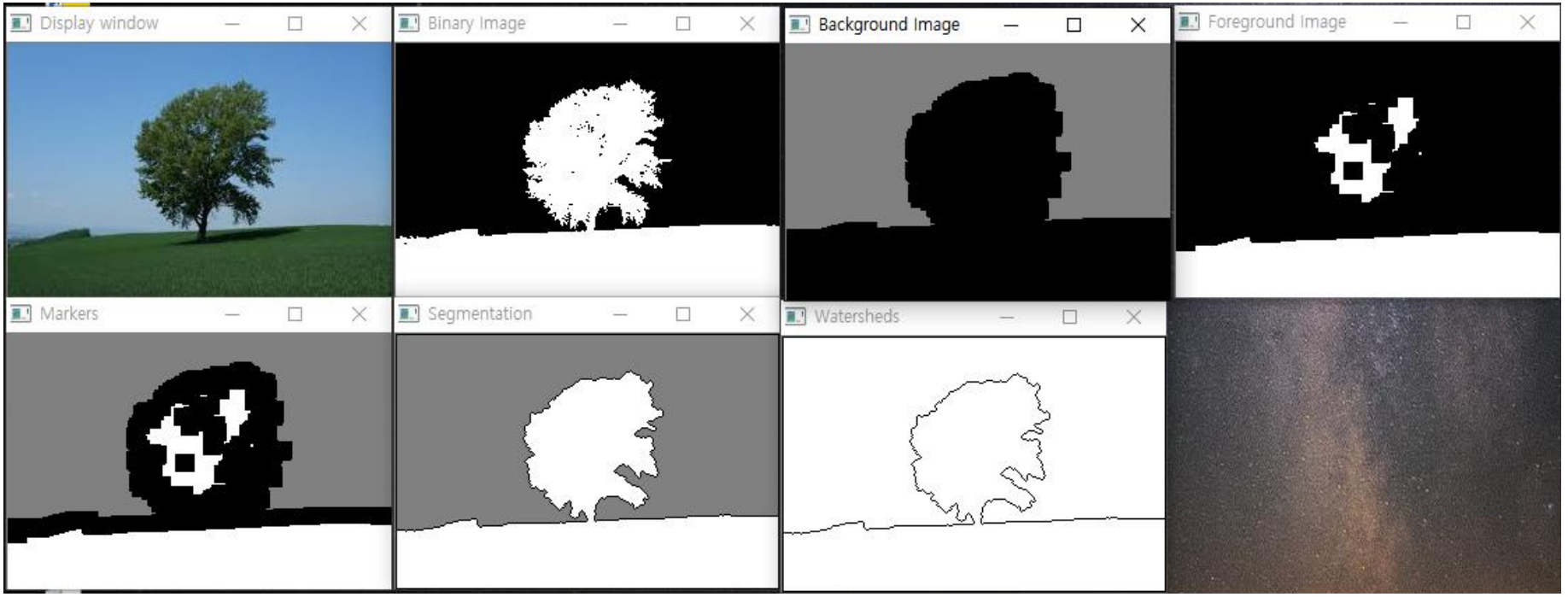
Morphology (형태학)처리 응용: Watershed 분할 기법

(계속)

```
// Create watershed segmentation object
WatershedSegmenter segmenter;

// Set markers and process
segmenter.setMarkers(markers);
Mat res = segmenter.process(image);

// Display segmentation result
//cv::namedWindow("Pre-Segmentation");
//cv::imshow("Pre-Segmentation",res);

// Display segmentation result
cv::namedWindow("Segmentation");
cv::imshow("Segmentation",segmenter.getSegmentation());

// Display watersheds
cv::namedWindow("Watersheds");
cv::imshow("Watersheds",segmenter.getWatersheds());

waitKey(0);

return 0;
}
```

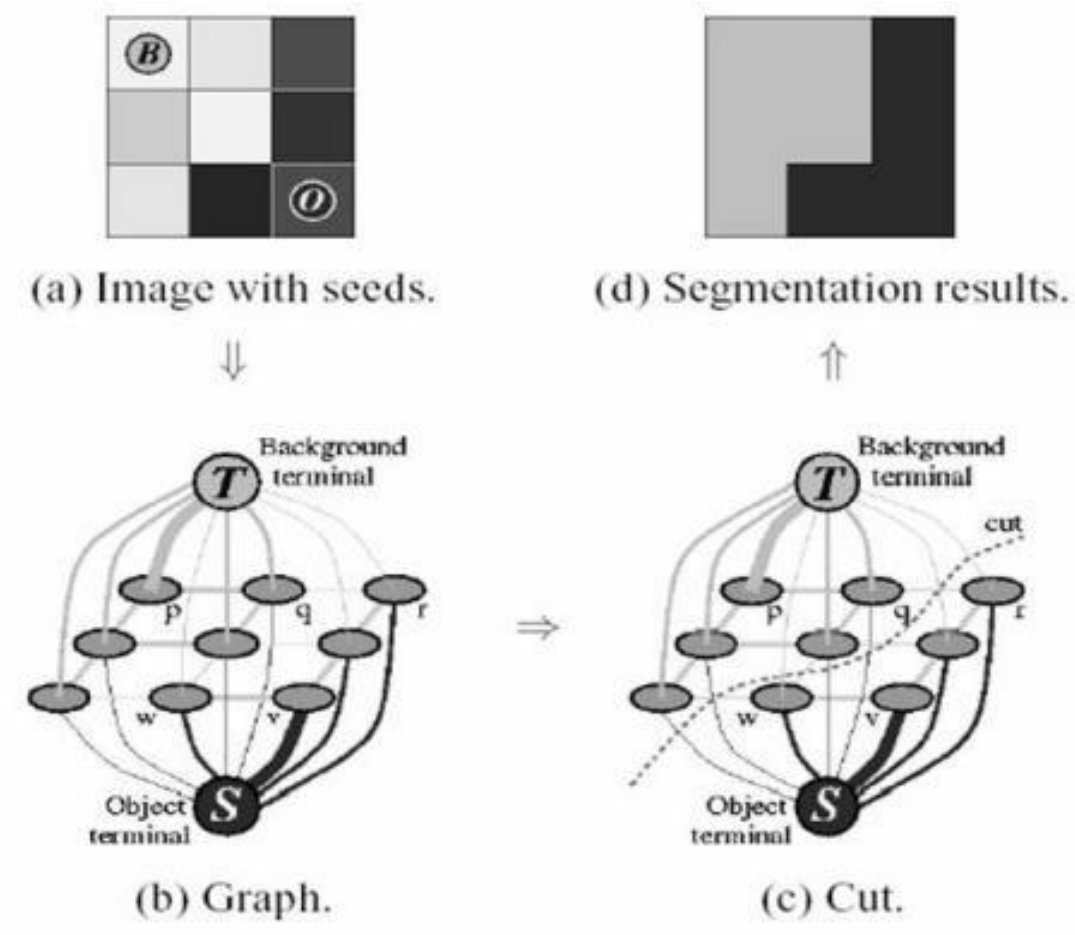
Morphology (형태학)처리 응용: Watershed 분할 기법

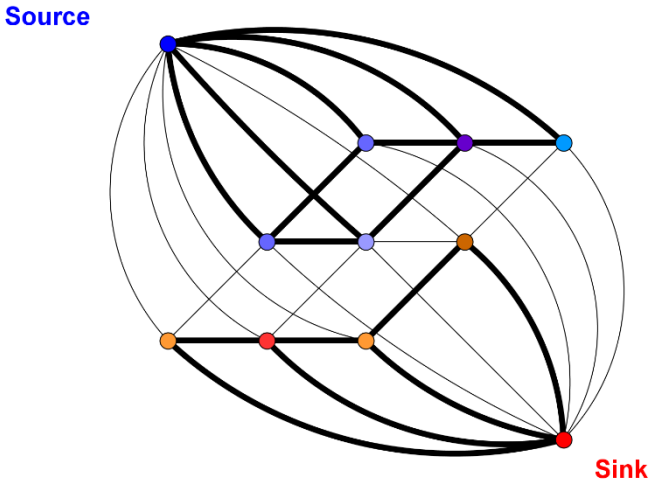
- 수행 결과



Morphology (형태학)처리 응용: GrabCut 분할 기법

- GrabCut concept: 개별 화소 간의 관계(유사도)를 그래프로 표현 후 cut 수행





Morphology (형태학)처리 응용: GrabCut 분할 기법

- grabCut API
  - void **grabCut**(InputArray **img**, InputOutputArray **mask**, Rect **rect**, InputOutputArray **bgdModel**, InputOutputArray **fgdModel**, int **iterCount**, int **mode**=GC_EVAL)

Parameters:	•img – Input 8-bit 3-channel image. •mask – Input/output 8-bit single-channel mask. The mask is initialized by the function when mode is set to GC_INIT_WITH_RECT. Its elements may have one of following values: • GC_BGD defines an obvious background pixels. • GC_FGD defines an obvious foreground (object) pixel. • GC_PR_BGD defines a possible background pixel. • GC_PR_FGD defines a possible foreground pixel. •rect – ROI containing a segmented object. The pixels outside of the ROI are marked as "obvious background". The parameter is only used when mode==GC_INIT_WITH_RECT . •bgdModel – Temporary array for the background model. Do not modify it while you are processing the same image. •fgdModel – Temporary arrays for the foreground model. Do not modify it while you are processing the same image. •iterCount – Number of iterations the algorithm should make before returning the result. Note that the result can be refined with further calls with mode==GC_INIT_WITH_MASK or mode==GC_EVAL . •**mode** – Operation mode that could be one of the following: • **GC_INIT_WITH_RECT** The function initializes the state and the mask using the provided rectangle. After that it runs iterCount iterations of the algorithm. • **GC_INIT_WITH_MASK** The function initializes the state using the provided mask. Note that GC_INIT_WITH_RECT and GC_INIT_WITH_MASK can be combined. Then, all the pixels outside of the ROI are automatically initialized with GC_BGD . • **GC_EVAL** The value means that the algorithm should just resume.

Morphology (형태학)처리 응용: GrabCut 분할 기법

- grabCut 예제 코드
  - **Bgmodel**과 **fgmodel**을 설정이 필요함

```
int main( int argc, char** argv )
{
        Mat image, gray, binary;

        /// Load image
        image = imread( "group.jpg", 1);   // Read the file

         if (image.empty()){                              // Check for invalid input
                cout << "Could not open or find the image" << std::endl;
                return -1;
         }
         namedWindow("Display window", WINDOW_AUTOSIZE);  // Create a window for display.
         imshow("Display window", image );

         // define bounding rectangle --전경(객체 부분) 정의
        cv::Rect rectangle2(10,100,380,180);

        cv::Mat bkgModel, fgrModel; // the models (internally used)
        cv::Mat result; // segmentation result (4 possible values)
        cv::Mat foreground(image.size(),CV_8UC3,cv::Scalar(255,255,255));
```

(계 속)

Morphology (형태학)처리 응용: GrabCut 분할 기법

```
// GrabCut segmentation
cv::grabCut(image,  // input image
                    result, // segmentation result
                    rectangle2,bkgModel,fgrModel,5,cv::GC_INIT_WITH_RECT);

// Get the pixels marked as likely foreground
result= result&1;
foreground.create(image.size(),CV_8UC3);
foreground.setTo(cv::Scalar(255,255,255));
image.copyTo(foreground, result); // bg pixels not copied

// draw rectangle on original image
cv::rectangle(image, rectangle2, cv::Scalar(255,255,255),1);
cv::namedWindow("Image 2");
cv::imshow("Image 2",image);

// display result
cv::namedWindow("Foreground objects");
cv::imshow("Foreground objects", foreground);

waitKey(0);

return 0;
}
```

Morphology (형태학)처리 응용: GrabCut 분할 기법

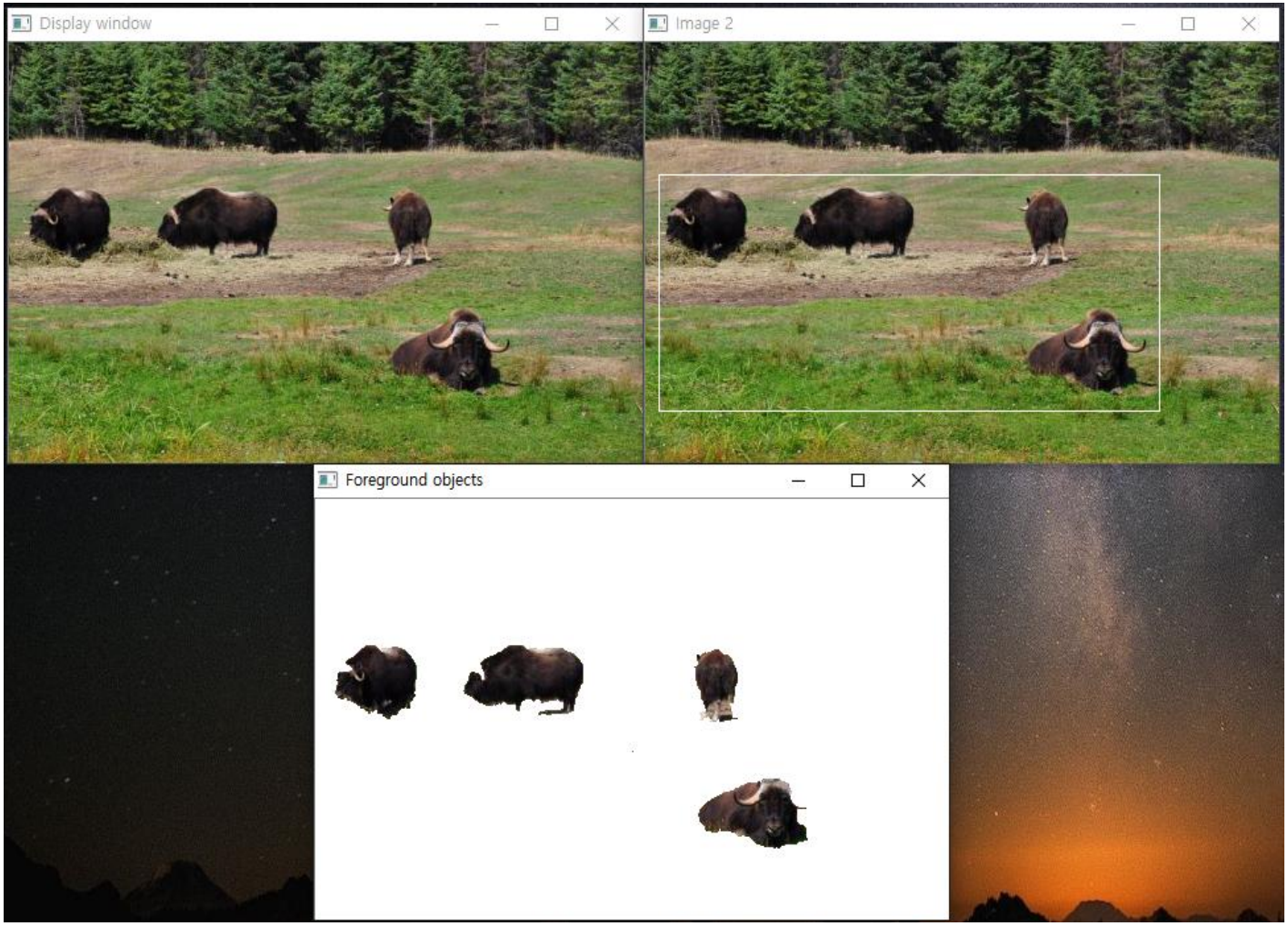
- x.**copyTo()**: Copies the matrix to another one (x=source image).

void Mat::**copyTo**(OutputArray **m**)
void Mat::**copyTo**(OutputArray **m**, InputArray **mask**)

Parameters:	•m – Destination matrix. If it does not have a proper size or type before the operation, it is reallocated. •mask – Operation mask. Its non-zero elements indicate which matrix elements need to be copied. Keep in mind that the mask needs to be of type CV_8U and can have 1/multiple channels.

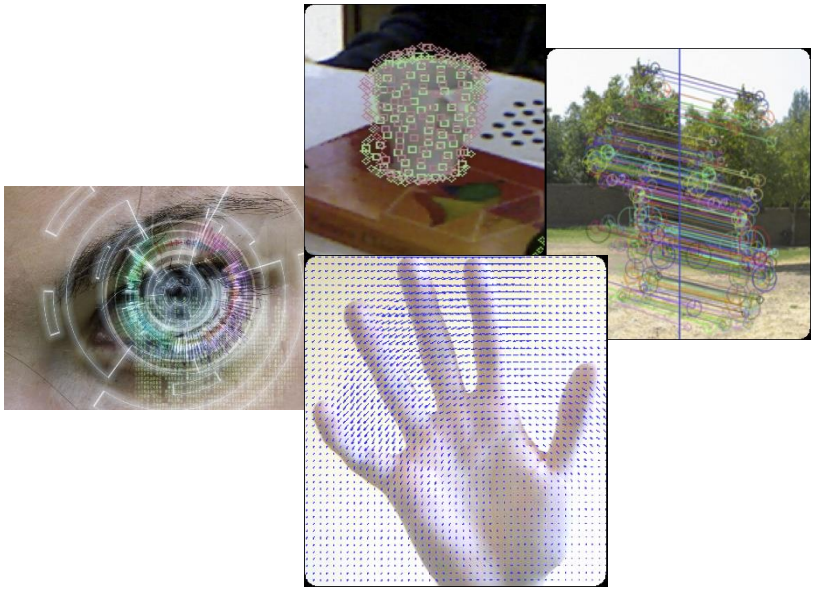
Morphology (형태학)처리 응용: GrabCut 분할 기법

- 수행 결과: 4마리의 버팔로(물소) 추출함



COMPUTER VISION 비젼 프로그래밍

Thank you and Question?



IVPL
Intelligent Vision Processing Lab