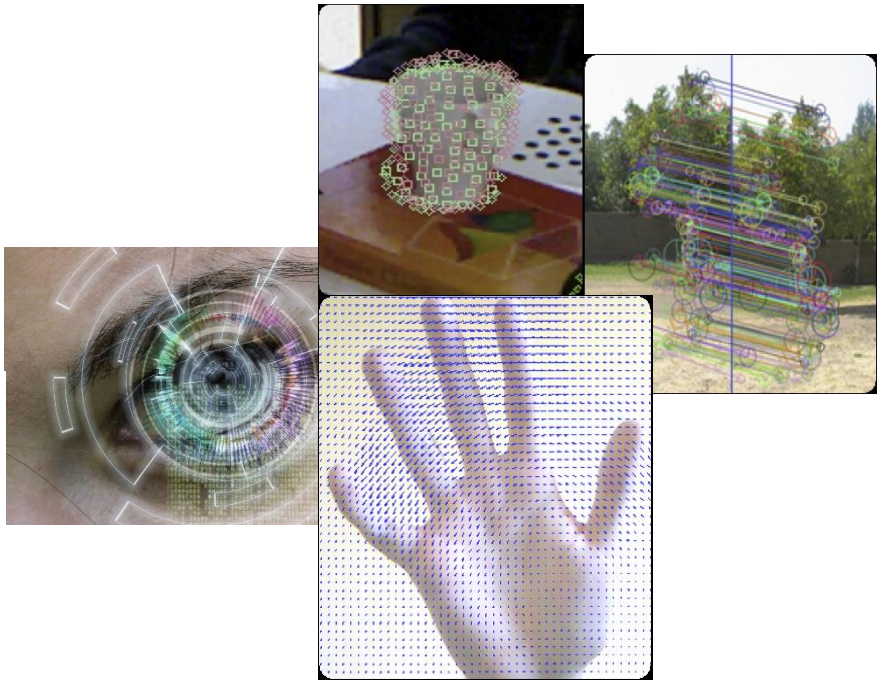
IVPL
Intelligent Vision Processing Lab

2023 Fall

COMPUTER VISION 비젼 프로그래밍



4장. 유용한 기본 기능들 (실습)

Contents

- OpenCV에서 유용한 기본 기능들
  - Mouse Event
  - Drawing Objects: Circle, rectangles, lines and so on
  - Text drawing
  - TrackBar

common API concepts

- cv namespace
  - All the OpenCV classes and functions are placed into the **`cv`** namespace. Therefore, to access this functionality from your code, use the **`cv::`** specifier or **`using namespace cv;`** directive

```
#include "opencv2/core.hpp"
...
cv::Mat H = cv::findHomography(points1, points2, CV_RANSAC, 5);
 ...
```

or

```
#include "opencv2/core.hpp"
using namespace cv;
...
Mat H = findHomography(points1, points2, CV_RANSAC, 5 );
...
```

WaitkEY() - 입력 대기 관련 함수

- waitkey(n)함수
  - N msec.만큼 임의의 키 입력을 대기함
- 사용법
  - Char ch = waitKey(10); // 10ms동안 입력을 대기 후 키 입력은 ch에 ASCII 코드로 맵핑
  - If(ch == 27) break; // 27 == ESC key
  - If(ch == 32) // 32 == SPACE key
  - waitkey() 또는 waitkey(0): 키 입력까지 무한 대기함

namedWindow()

- namedWindow()
  - Creates a window
- 사용법
  - void namedWindow(const string& **winname**, int **flags**=WINDOW_AUTOSIZE)

Parameters:	•name – Name of the window in the window caption that may be used as a window identifier. •flags – Flags of the window. The supported flags are: • WINDOW_NORMAL If this is set, the user can resize the window (no constraint). • WINDOW_AUTOSIZE If this is set, the window size is automatically adjusted to fit the displayed image (see imshow()), and you cannot change the window size manually. • WINDOW_OPENGL If this is set, the window will be created with OpenGL support.

imshow()/destroyAllWindows

- imshow("윈도우 이름", 영상데이터 버퍼)
  - Displays an image in the specified window
- 사용법
  - void imshow(const string& **winname**, InputArray **mat**)

Parameters:	• **winname** – Name of the window. • **image** – Image to be shown.

- destroyAllWindows
  - Destroys all of the HighGUI windows.

Mouse Event 구현하기: setMouseCallback 함수

- Mouse callback 함수 활용
  - **void setMouseCallback(const string& winname, MouseCallback onMouse, void* userdata = 0)**
    - **winname** - Name of the OpenCV window. All mouse events related to this window will be registered
    - **onMouse** - Name of the callback function. Whenever mouse events related to the above window occur, this callback function will be called. This function should have the signature like the following
      - » **void FunctionName(int event, int x, int y, int flags, void* userdata)**
      - » **event** - Type of the mouse event. These are the entire list of mouse events
        - EVENT_MOUSEMOVE
        - EVENT_LBUTTONDOWN
        - EVENT_RBUTTONDOWN
        - EVENT_MBUTTONDOWN
        - EVENT_LBUTTONUP
        - EVENT_RBUTTONUP
        - EVENT_MBUTTONUP
        - EVENT_LBUTTONDBLCLK
        - EVENT_RBUTTONDBLCLK
        - EVENT_MBUTTONDBLCLK

Mouse Event 구현하기: setMouseCallback 함수

» **x** - x coordinate of the mouse event

» **y** - y coordinate of the mouse event

» **flags** - Specific condition whenever a mouse event occurs. See the next OpenCV example code for the usage of this parameter. Here is the entire list of enum values which will be possesed by "flags"

- EVENT_FLAG_LBUTTON
- EVENT_FLAG_RBUTTON
- EVENT_FLAG_MBUTTON
- EVENT_FLAG_CTRLKEY
- EVENT_FLAG_SHIFTKEY
- EVENT_FLAG_ALTKEY

» **userdata** - Any pointer passes to the "**setMouseCallback"** function as the 3rd parameter (see below)

– **userdata** - This pointer will be passed to the callback function

Mouse Event 구현하기: button event 기능 구현

- 실습 코드 이해

```
#include <opencv2/core/core.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2₩opencv.hpp>
#include <iostream>
#include <string>

using namespace cv;
using namespace std;

void my_mouse_callback(int event, int x, int y, int flags, void* param); // 함수 선언

int main(int argc, char** argv)
{
        //--- 마우스 이벤트 구현 ---//
        int i,j,k;

        Mat image;
        image.create(500,500,CV_8UC3);

        namedWindow("Main");

        for(i=0;i<500;i++){
            for(j=0;j<500;j++){
                for(k=0;k<3;k++){
                    Vec3b &intensity = image.at<Vec3b>(j,i);
                    intensity.val[k]=0;
                }
            }
        }
```

Mouse Event 구현하기: button event 기능 구현

- 실습 코드 이해

```
    (계속)
    setMouseCallback( "Main", my_mouse_callback, &image );

    imshow("Main",image);
    waitKey(0);                  // Wait for any user's key stroke....!!!!!

   return 0;
}

//--- 마우스 이벤트  구현함 --//
void my_mouse_callback( int event, int x, int y, int flags, void* param ) {
    int thickness = -1;
    int lineType = 8;

    if(event==EVENT_LBUTTONDOWN){
      cout << "Left button has been clicked ("<< x <<","<< y <<")" << std::endl;

    }else if(event==EVENT_RBUTTONDOWN){
      cout << "Right button has been clicked ("<< x <<","<< y <<")" << std::endl;
    }else if (event==EVENT_MOUSEMOVE){
       cout << "Mouse has been moved ("<< x <<","<< y <<")" << std::endl;
    }
}
```

Mouse Event 구현하기: button event 기능 구현

- 수행 결과: 마우스 이동 시, 또는 버튼 클릭 시 좌표를 표시해 줌

Mouse Event 구현하기: 마우스 클릭 위치의 rgb 값 출력

- 실습 코드의 이해

```
#include <opencv2/core/core.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2₩opencv.hpp>
#include <iostream>
#include <string>

using namespace cv;
using namespace std;

void mouseEvent(int evt, int x, int y, int flags, void* param){
    Mat* rgb = (Mat*) param;
    if (evt == EVENT_LBUTTONDOWN)
    {
        printf("%d %d: %d, %d, %d₩n",
        x, y,
        (int)(*rgb).at<Vec3b>(y, x)[0],
        (int)(*rgb).at<Vec3b>(y, x)[1],
        (int)(*rgb).at<Vec3b>(y, x)[2]);
    }
}

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

        //--OpenCV 2.x 구현 --//
        Mat image, result;

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

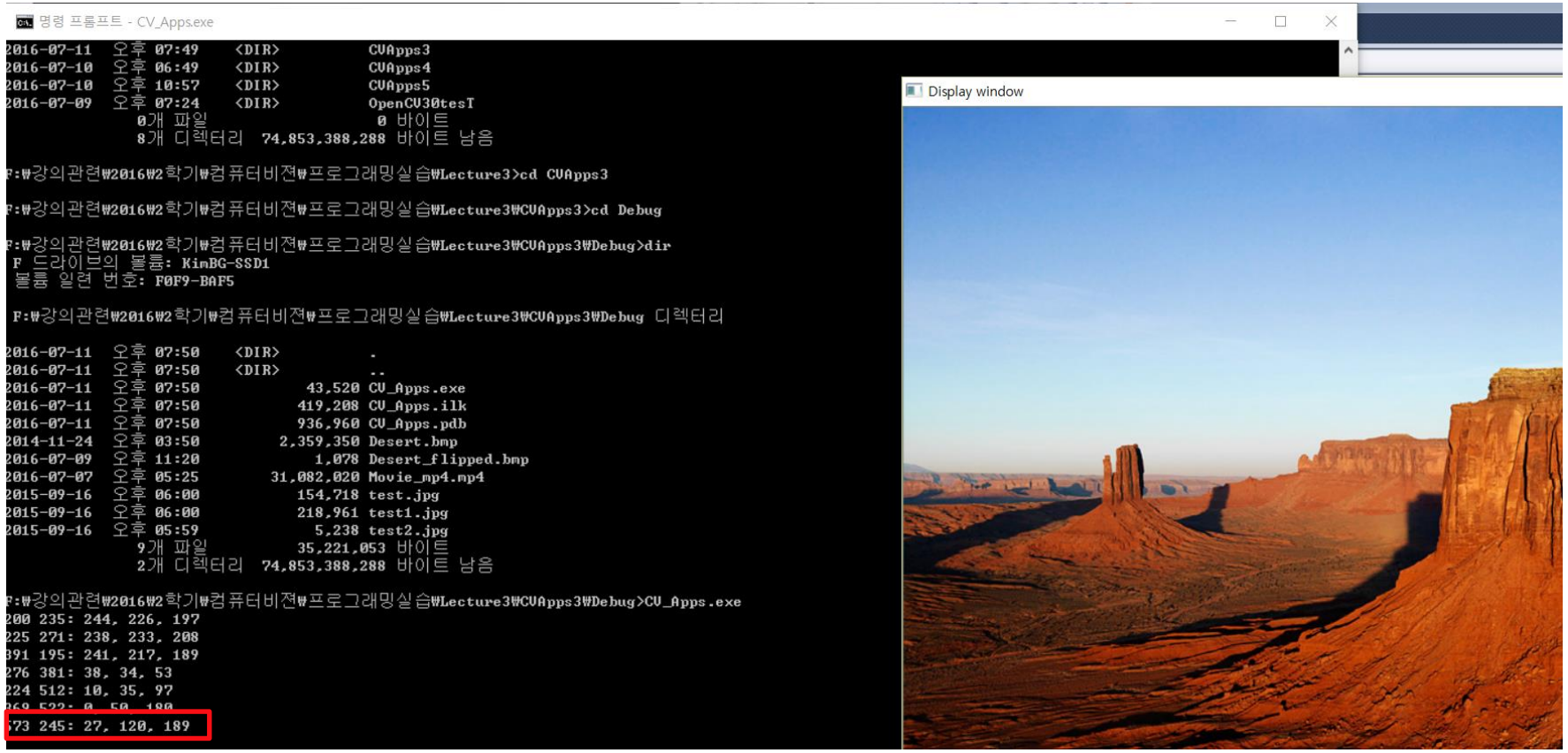
Mouse Event 구현하기: 마우스 클릭 위치의 rgb 값 출력

- 실습 코드의 이해

```
  (계속)
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.

//set the callback function for any mouse event
 setMouseCallback("Display window", mouseEvent, &image);

imshow("Display window", image);                    // Show our image inside it.

waitKey(0);                                        // Wait for a keystroke in the window


return 0;
}
```

Mouse Event 구현하기: 마우스 클릭 위치의 rgb 값 출력

- 실행 결과: 영상의 임의 지점을 클릭하면 해당 위치와 R,G,B 값이 출력됨



Draw somethings…!!: point/Scalar

Point API

- Point p1; → p1.x, p1.y의 2개 성분으로 2차원 공간 좌표 설정
- Point3 p1; → p1.x, p1.y, p1.z 의 3차원 공간 좌표 설정
- Point pt = Point(10, 8); ← pt.x=10, pt.y=8로 설정

Scalar

- Represents a 4-element vector. The type Scalar is widely used in OpenCV for passing pixel values (color values).
- Scalar(a, b, c) : We would be defining a RGB color such as: *Red = c*, *Green = b* and *Blue = a.*

Draw somethings...!!: line 그리기

- line API
  - Draws a line segment connecting two points
  - void **line**(InputOutputArray **img**, Point **pt1**, Point **pt2**, const Scalar& **color**, int **thickness**=1, int **lineType**=LINE_8, int **shift**=0)

Parameters:	•img – Image. •pt1 – First point of the line segment. •pt2 – Second point of the line segment. •color – Line color. •thickness – Line thickness. •lineType – Type of the line: • LINE_8 (or omitted) - 8-connected line. • LINE_4 - 4-connected line. • LINE_AA - antialiased line. •shift – Number of fractional bits in the point coordinates.

Draw somethings...!!: Circle 그리기

- Circle API
  - void **circle**(InputOutputArray **img**, Point **center**, int **radius**, const Scalar& **color**, int **thickness**=1, int **lineType**=LINE_8, int **shift**=0)

Parameters:	•img – Image where the circle is drawn. •center – Center of the circle. •radius – Radius of the circle. •color – Circle color. •thickness – Thickness of the circle outline, if positive. Negative thickness means that a filled circle is to be drawn. •lineType – Type of the circle boundary. See the line() description. •shift – Number of fractional bits in the coordinates of the center and in the radius value.

Draw somethings...!!: Rectangle 구현

- rectangle API
  - Draws a simple, thick, or filled up-right rectangle.
  - void **rectangle**(InputOutputArray **img**, Point **pt1**, Point **pt2**, const Scalar& **color**, int **thickness**=1, int **lineType**=LINE_8, int **shift**=0)

Parameters:	•img – Image. •pt1 – Vertex of the rectangle. •pt2 – Vertex of the rectangle opposite to pt1 . •rec – Alternative specification of the drawn rectangle. •color – Rectangle color or brightness (grayscale image). •thickness – Thickness of lines that make up the rectangle. Negative values, like CV_FILLED , mean that the function has to draw a filled rectangle. •lineType – Type of the line. See the line() description. •shift – Number of fractional bits in the point coordinates.

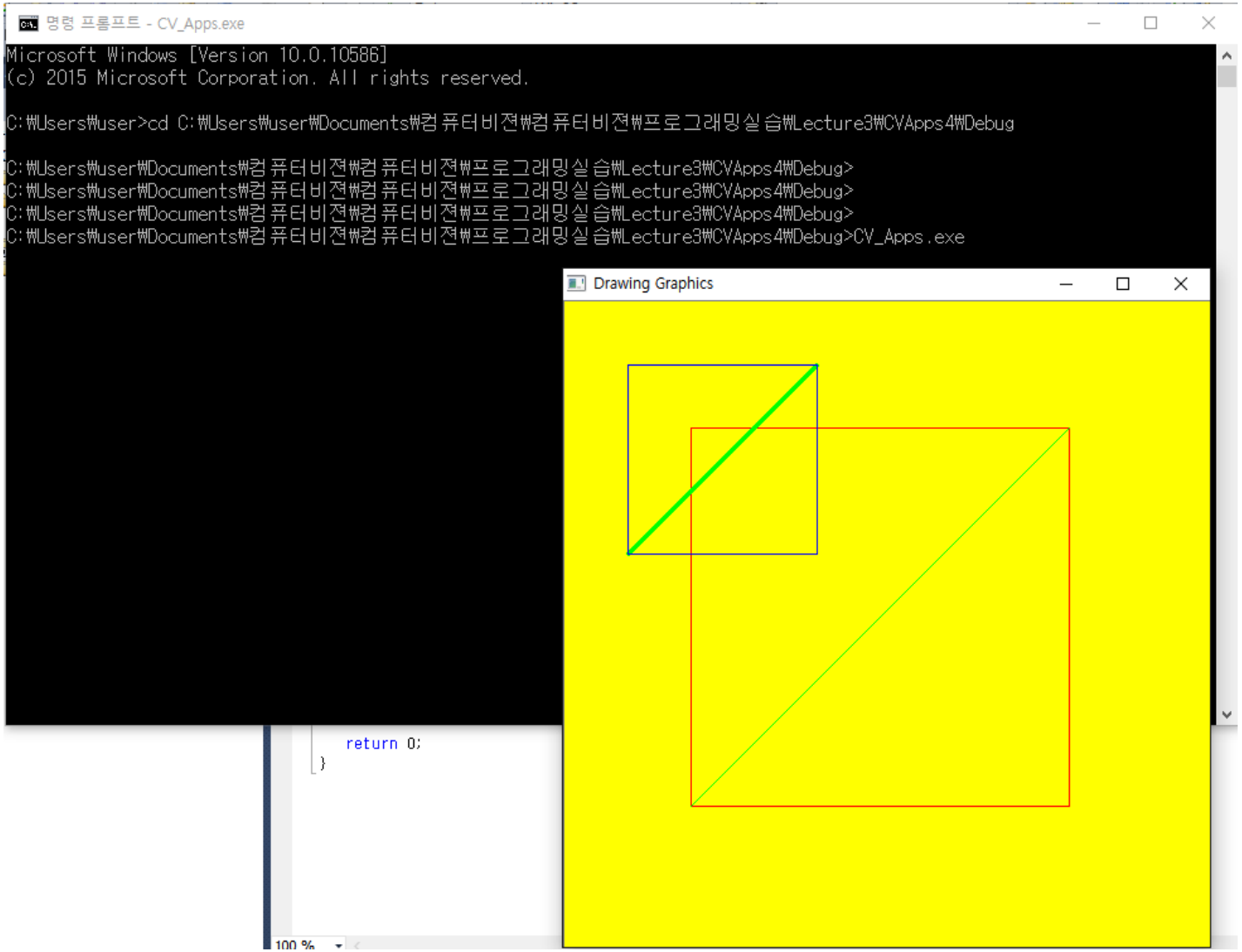
Draw somethings...!!

- 실습 예제 이해

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

      //--OpenCV 2.x 구현 --//
      Mat image(512,512,CV_8UC3, Scalar(0,255,255)), result;

       rectangle(image, Point(100, 100), Point(400, 400), CV_RGB(255, 0, 0));
       line(image, Point(400, 100), Point(100, 400), Scalar(0, 255, 0));
       line(image, Point(400, 100), Point(100, 400), Scalar(0, 255, 0), 2, 8, 1);
       rectangle(image, Point(400/2, 100/2), Point(100/2, 400/2), CV_RGB(0, 0, 255));

       namedWindow("Drawing Graphics", CV_WINDOW_AUTOSIZE);
       imshow("Drawing Graphics", image);

       waitKey(0);// Wait for a keystroke in the window

       cvDestroyWindow("Drawing Graphics");

         return 0;
}
```

Draw somethings...!!

- 실행 결과: 사각형(색깔 변화), 선 그리기

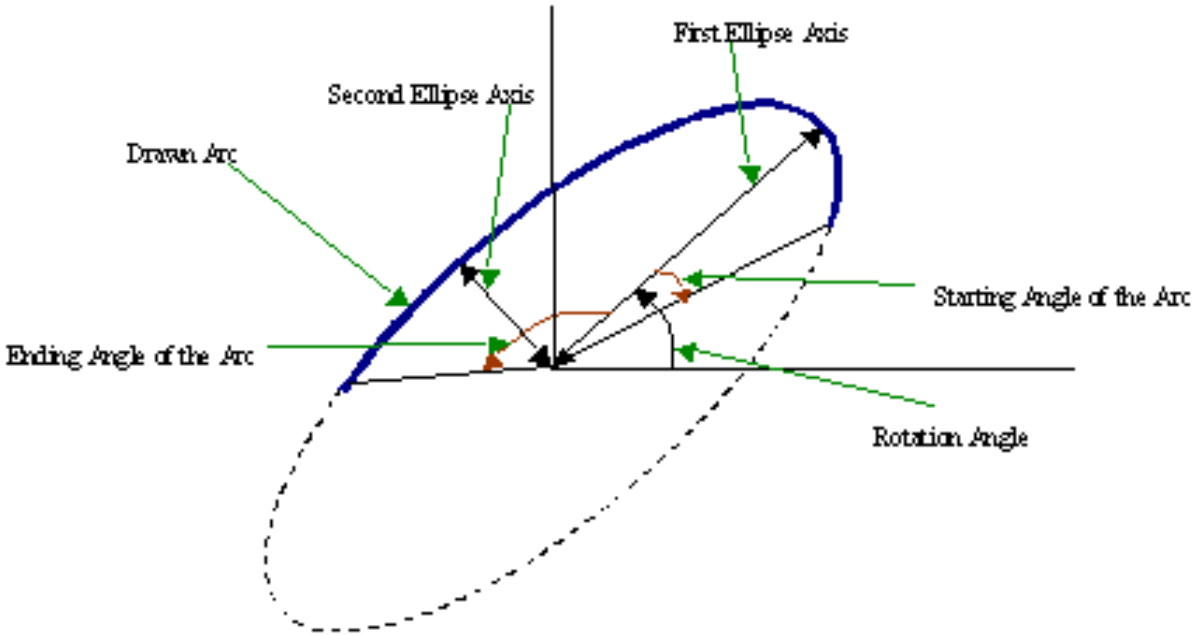


Draw somethings...!!: ellipse 구현

- ellipse API
  - Draws a simple or thick elliptic arc or fills an ellipse sector.
  - void **ellipse**(InputOutputArray **img**, Point **center**, Size **axes**, double **angle**, double **startAngle**, double **endAngle**, const Scalar& **color**, int **thickness**=1, int **lineType**=LINE_8, int **shift**=0)

Parameters:	•**img** – Image. •**center** – Center of the ellipse. •**axes** – Half of the size of the ellipse main axes. •**angle** – Ellipse rotation angle in degrees. •**startAngle** – Starting angle of the elliptic arc in degrees. •**endAngle** – Ending angle of the elliptic arc in degrees. •**box** – Alternative ellipse representation via RotatedRect or CvBox2D. This means that the function draws an ellipse inscribed in the rotated rectangle. •**color** – Ellipse color. •**thickness** – Thickness of the ellipse arc outline, if positive. Otherwise, this indicates that a filled ellipse sector is to be drawn. •**lineType** – Type of the ellipse boundary. See the line() description. •**shift** – Number of fractional bits in the coordinates of the center and values of axes.

Draw somethings...!!: ellipse 구현



Draw somethings...!!: polylines구현

Polylines API

- Draws several polygonal curves.
- void **polylines**(Mat& img, const Point* const* pts, const int* npts, int ncontours, bool isClosed, const Scalar& color, int thickness=1, int lineType=LINE_8, int shift=0)

Parameters:	•img – Image. •pts – Array of polygonal curves. •npts – Array of polygon vertex counters. •ncontours – Number of curves. •isClosed – Flag indicating whether the drawn polylines are closed or not. If they are closed, the function draws a line from the last vertex of each curve to its first vertex. •color – Polyline color. •thickness – Thickness of the polyline edges. •lineType – Type of the line segments. See the line() description. •shift – Number of fractional bits in the vertex coordinates.

Draw somethings...!!

- **실습 예제의 이해**

```
 (중략)
using namespace cv;
using namespace std;


int w=512;// 생성 영상의 크기 설정
void MyLine( Mat img, Point start, Point end );
void MyEllipse( Mat img, double angle );
void MyFilledCircle( Mat img, Point center );

int main(int argc, char** argv)
{
	int k;
	//--OpenCV 2.x 구현 --//
	/// Windows names

	char atom_window[] = "Drawing 1: Atom";

	/// Create black empty images
	Mat atom_image = Mat::zeros( w, w, CV_8UC3 );
	namedWindow(atom_window);
```

Draw somethings...!!

```
//-- 메뉴 표시하기 ---//
cout << "1 - 타원 그리기" << std::endl;
cout << "2 - 채워진 원 그리고" << std::endl;
cout << "3 - 사각형 그리기" << std::endl;
cout << "4 - 라인 그리기" << std::endl;

/// 1. Draw a simple atom:
k=waitKey(0);

if (k==49){
        /// 1.a. Creating ellipses
        MyEllipse( atom_image, 90 );
        MyEllipse( atom_image, 0 );
        MyEllipse( atom_image, 45 );
        MyEllipse( atom_image, -45 );
}else if(k==50){
        /// 1.b. Creating circles
        MyFilledCircle( atom_image, Point( w/2.0, w/2.0) );
}else if (k==51){
        /// 2.b. Creating rectangles
        rectangle( atom_image, Point( 0, 7*w/8.0 ), Point( w, w), Scalar( 0, 255, 255 ),
            -1,    8 );
```

Draw somethings...!!

```
}else if (k==52){
        /// 2.c. Create a few lines
        MyLine( atom_image, Point( 0, 15*w/16 ), Point( w, 15*w/16 ) );
        MyLine( atom_image, Point( w/4, 7*w/8 ), Point( w/4, w ) );
        MyLine( atom_image, Point( w/2, 7*w/8 ), Point( w/2, w ) );
        MyLine( atom_image, Point( 3*w/4, 7*w/8 ), Point( 3*w/4, w ) );
}

imshow(atom_window,atom_image);

 waitKey(0);// Wait for a keystroke in the window

 cvDestroyWindow("Drawing Graphics");

    return 0;
}
```

Draw somethings...!!

```
}else if (k==52){
        /// 2.c. Create a few lines
        MyLine( atom_image, Point( 0, 15*w/16 ), Point( w, 15*w/16 ) );
        MyLine( atom_image, Point( w/4, 7*w/8 ), Point( w/4, w ) );
        MyLine( atom_image, Point( w/2, 7*w/8 ), Point( w/2, w ) );
        MyLine( atom_image, Point( 3*w/4, 7*w/8 ), Point( 3*w/4, w ) );
}

imshow(atom_window,atom_image);

 waitKey(0);// Wait for a keystroke in the window


    return 0;
}
```

Draw somethings...!!

- 사용자 함수 구현

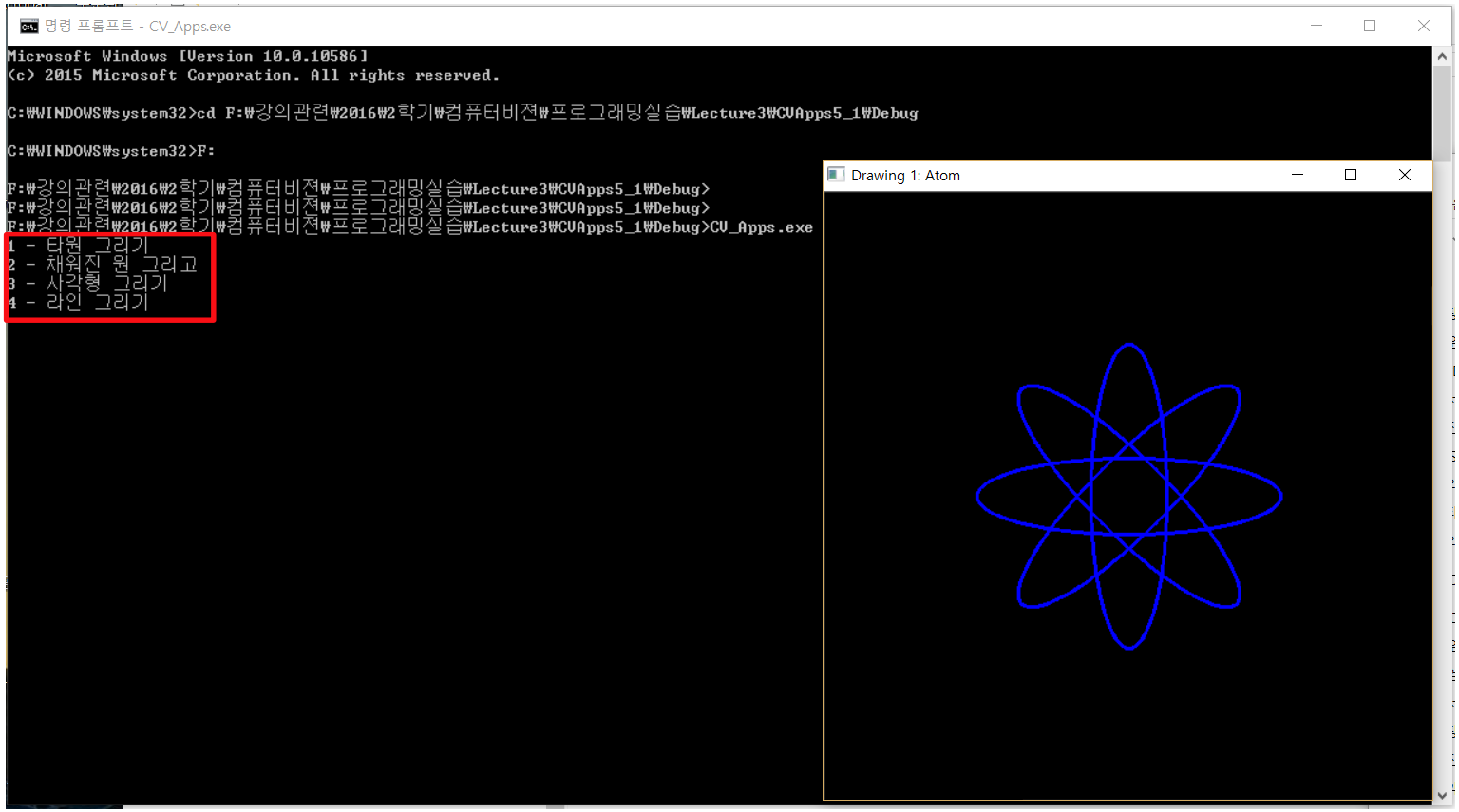
```
void MyLine( Mat img, Point start, Point end )
{
    int thickness = 2;
    int lineType = 8;
    line( img, start, end,
          Scalar( 0, 0, 255 ),
          thickness,
          lineType );
}
void MyFilledCircle( Mat img, Point center )
{
    int thickness = -1;
    int lineType = 8;

    circle( img,  center, w/6.0, Scalar( 0, 0, 255 ), thickness, lineType );
}
```

Draw somethings...!!

- 사용자 함수 구현(계속)

```
void MyEllipse( Mat img, double angle )
{
    int thickness = 2;
    int lineType = 8;

    ellipse( img,
        Point( w/2.0, w/2.0 ),
        Size( w/4.0, w/16.0 ),
        angle,
        0,
        360,
        Scalar( 255, 0, 0 ),
        thickness,
        lineType );
}
```

- 실행 결과: 메뉴에 따라 타원, 채워진 원, 사각형(색깔 변화), 선 그리기



Mouse Event 구현하기: 드래그로 사각형 그리기

```
  (중략)
using namespace cv;
using namespace std;


//---전역 변수 선언 ---//
Mat src,img,ROI;
Rect cropRect(0,0,0,0);
Point P1(0,0);
Point P2(0,0);

const char* winName="Crop Image";
bool clicked=false;
int i=0;
char imgName[15];

void checkBoundary();
void onMouse( int event, int x, int y, int f, void* );
void showImage();
```

Mouse Event 구현하기: 드래그로 사각형 그리기

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

    cout<<"Click and drag for Selection"<<endl<<endl;
    cout<<"------> Press 's' to save"<<endl<<endl;

    cout<<"------> Press '8' to move up"<<endl;
    cout<<"------> Press '2' to move down"<<endl;
    cout<<"------> Press '6' to move right"<<endl;
    cout<<"------> Press '4' to move left"<<endl<<endl;

    cout<<"------> Press 'w' increas top"<<endl;
    cout<<"------> Press 'x' increas bottom"<<endl;
    cout<<"------> Press 'd' increas right"<<endl;
    cout<<"------> Press 'a' increas left"<<endl<<endl;

    cout<<"------> Press 't' decrease top"<<endl;
    cout<<"------> Press 'b' decrease bottom"<<endl;
    cout<<"------> Press 'h' decrease right"<<endl;
    cout<<"------> Press 'f' decrease left"<<endl<<endl;

    cout<<"------> Press 'r' to reset"<<endl;
    cout<<"------> Press 'Esc' to quit"<<endl<<endl;
```

Mouse Event 구현하기: 드래그로 사각형 그리기

```
src=imread(argv[1],1);

namedWindow(winName,WINDOW_AUTOSIZE);
setMouseCallback(winName,onMouse,NULL );
imshow(winName,src);

while(1){
    char c=waitKey();
    if(c=='s'&& !ROI.empty()){
            //sprintf(imgName,"%d.jpg",i++);
            imwrite("croppedImg.jpg",ROI);
            cout<<"   Saved "<<imgName<<endl;
     }
     if(c=='6') cropRect.x++;
    if(c=='4') cropRect.x--;
    if(c=='8') cropRect.y--;
    if(c=='2') cropRect.y++;

    if(c=='w') { cropRect.y--; cropRect.height++;}
    if(c=='d') cropRect.width++;
    if(c=='x') cropRect.height++;
    if(c=='a') { cropRect.x--; cropRect.width++;}
```

Mouse Event 구현하기: 드래그로 사각형 그리기

```
            if(c=='t') { cropRect.y++; cropRect.height--;}
            if(c=='h') cropRect.width--;
            if(c=='b') cropRect.height--;
            if(c=='f') { cropRect.x++; cropRect.width--;}

            if(c==27) break;
            if(c=='r') {
                  cropRect.x=0;cropRect.y=0;
                 cropRect.width=0;cropRect.height=0;
            }

    showImage();

     } // end of "while()"


    return 0;
}
```

Mouse Event 구현하기: 드래그로 사각형 그리기

- Callback 함수 등 구현

```
void onMouse( int event, int x, int y, int f, void* ){

    switch(event){

        case   EVENT_LBUTTONDOWN   :
                                    clicked=true;

                                    P1.x=x;
                                    P1.y=y;
                                    P2.x=x;
                                    P2.y=y;
                                    break;

       case   EVENT_LBUTTONUP     :
                                    P2.x=x;
                                    P2.y=y;
                                    clicked=false;
                                    break;
       case   EVENT_MOUSEMOVE     :
                                    if(clicked){
                                    P2.x=x;
                                    P2.y=y;
                                    }
                                    break;

       default                    :    break;
    }
```

Mouse Event 구현하기: 드래그로 사각형 그리기

- Callback 함수 등 구현(계속)

```
 (계속)
if(clicked){
    if(P1.x>P2.x){ cropRect.x=P2.x;
                   cropRect.width=P1.x-P2.x; }
      else {        cropRect.x=P1.x;
                   cropRect.width=P2.x-P1.x; }

      if(P1.y>P2.y){ cropRect.y=P2.y;
                   cropRect.height=P1.y-P2.y; }
      else {        cropRect.y=P1.y;
                   cropRect.height=P2.y-P1.y; }

    }


   showImage();


}
```

Mouse Event 구현하기: 드래그로 사각형 그리기

- 영상 경계 계산 함수 등 구현

```
void checkBoundary(){
        //check croping rectangle exceed image boundary
        if(cropRect.width>img.cols-cropRect.x)
          cropRect.width=img.cols-cropRect.x;

        if(cropRect.height>img.rows-cropRect.y)
          cropRect.height=img.rows-cropRect.y;

         if(cropRect.x<0)
          cropRect.x=0;

        if(cropRect.y<0)
          cropRect.height=0;
}
```

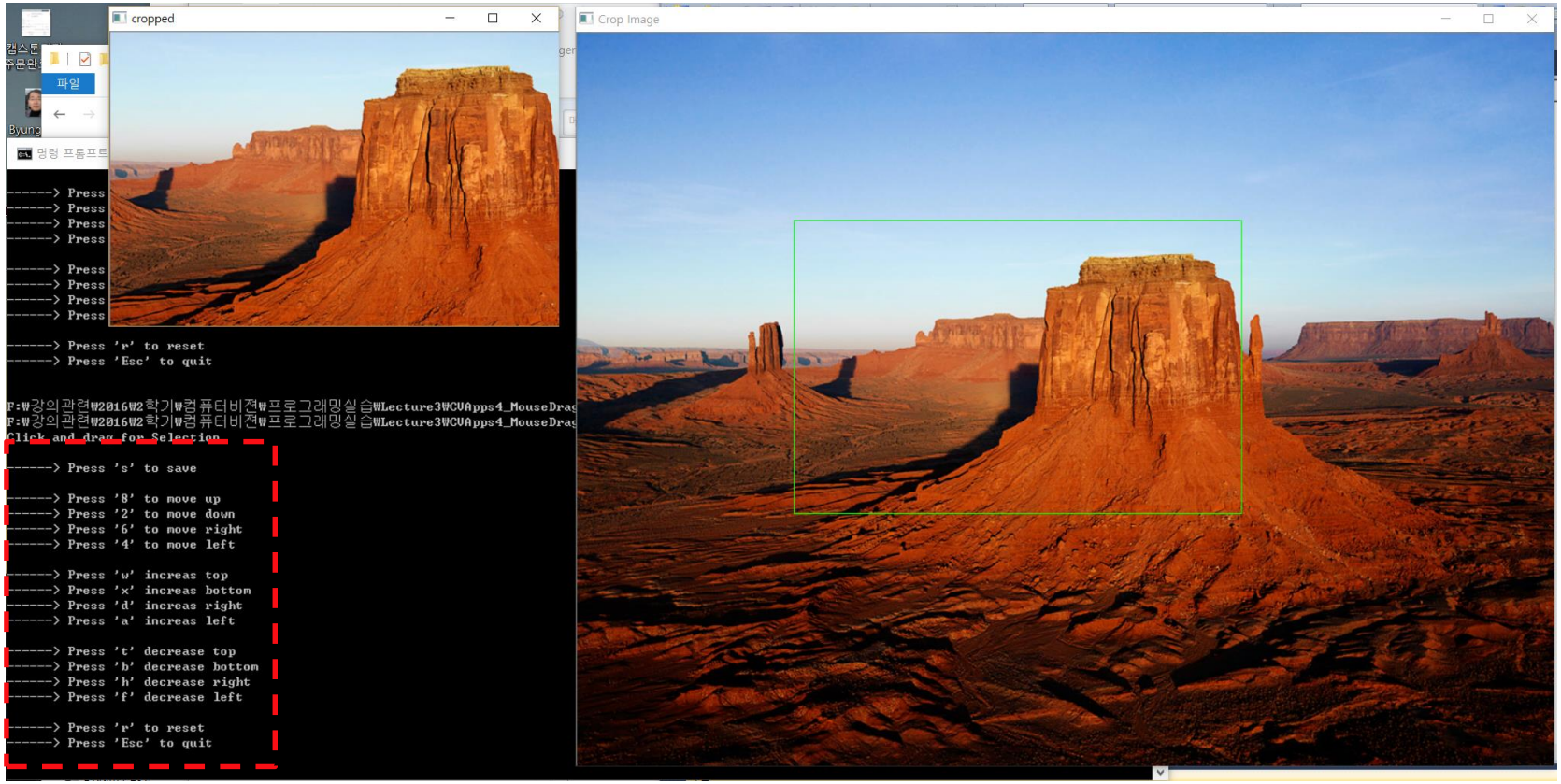
Mouse Event 구현하기: 드래그로 사각형 그리기

- 영상 보여주기 구현

```
void showImage(){
    img=src.clone();
    checkBoundary();
    if(cropRect.width>0&&cropRect.height>0){
        ROI=src(cropRect);
        imshow("cropped",ROI);
    }

    rectangle(img, cropRect, Scalar(0,255,0), 1, 8, 0 );
    imshow(winName,img);
}
```

Mouse Event 구현하기: 드래그로 사각형 그리기

- 실행 결과: 드래그 된 영역을 별도의 윈도우 화면에 그려 줌. 파일 저장 기능도 있음



Text 관련 기능들

- putText
  - Draws a text string.
  - void **putText**(**img**, const string& **text**, Point **org**, int **fontFace**, double **fontScale**, Scalar **color**, int **thickness**=1, int **lineType**=8, bool **bottomLeftOrigin**=false)

Parameters:	•**img** – Image. •**text** – Text string to be drawn. •**org** – Bottom-left corner of the text string in the image. •**font** – CvFont structure initialized using InitFont(). •**fontFace** – Font type. One of FONT_HERSHEY_SIMPLEX, FONT_HERSHEY_PLAIN, FONT_HERSHEY_DUPLEX, FONT_HERSHEY_COMPLEX, FONT_HERSHEY_TRIPLEX, FONT_HERSHEY_COMPLEX_SMALL, FONT_HERSHEY_SCRIPT_SIMPLEX, or FONT_HERSHEY_SCRIPT_COMPLEX, where each of the font ID's can be combined with FONT_ITALIC to get the slanted letters. •**fontScale** – Font scale factor that is multiplied by the font-specific base size. •**color** – Text color. •**thickness** – Thickness of the lines used to draw a text. •**lineType** – Line type. See the line for details. •**bottomLeftOrigin** – When true, the image data origin is at the bottom-left corner. Otherwise, it is at the top-left corner.

Text 관련 기능들

getTextSize

- Calculates the width and height of a text string.
- Size getTextSize(_**text**, int **fontFace**, double **fontScale**, int **thickness**, int* **baseLine**)

Parameters:	**text** – Input text string. **text_string** – Input text string in C format. **fontFace** – Font to use. See the putText() for details. **fontScale** – Font scale. See the putText() for details. **thickness** – Thickness of lines used to render the text. See putText() for details. **baseLine** – Output parameter - y-coordinate of the baseline relative to the bottom-most text point. **baseline** – Output parameter - y-coordinate of the baseline relative to the bottom-most text point. **font** – Font description in terms of old C API. **text_size** – Output parameter - The size of a box that contains the specified text.

Text 관련 기능들

- 영상 위에 임의의 text 표시해 주기-예제 코드 이해

```
int main(int argc, char** argv)
{
        string text = "Funny text inside the box";
        int fontFace = FONT_HERSHEY_SCRIPT_SIMPLEX;
        double fontScale = 2;
        int thickness = 3;

        Mat img(600, 800, CV_8UC3, Scalar::all(0));

        int baseline=0;
        Size textSize = getTextSize(text, fontFace, fontScale, thickness, &baseline);
        baseline += thickness;

        // center the text
        Point textOrg((img.cols - textSize.width)/2, (img.rows + textSize.height)/2);

        // draw the box
        rectangle(img, textOrg + Point(0, baseline), textOrg + Point(textSize.width, -textSize.height), Scalar(0,0,255));
        // ... and the baseline first
        line(img, textOrg + Point(0, thickness),textOrg + Point(textSize.width, thickness),  Scalar(0, 0, 255));

        // then put the text itself
        putText(img, text, textOrg, fontFace, fontScale, Scalar::all(255), thickness, 8);

        namedWindow("Display text");
        imshow("Display text",img);

         waitKey(0);// Wait for a keystroke in the window

       cvDestroyWindow("Drawing Graphics");

      return 0;
}
```

Text 관련 기능들

- 실행 결과:

Trackbar 기능의 활용

- Trackbar?
  - To give some values to increase or decrease on the image window linearly.
  - For example, if we want to vary a threshold to binarize image?

(a) 원본 (b) Th=120 (c) Th=160

- In OpenCV, trackbar is available.

Trackbar 기능의 활용

- Trackbar API
  - int createTrackbar(_**trackbarname**, const string& **winname**, int* **value**, int **count**, TrackbarCallback **onChange**=0, void* **userdata**=0)

Parameters:	•**trackbarname** – Name of the created trackbar. •**winname** – Name of the window that will be used as a parent of the created trackbar. •**value** – Optional pointer to an integer variable whose value reflects the posit ion of the slider. Upon creation, the slider position is defined by this variable. •**count** – Maximal position of the slider. The minimal position is always 0. •**onChange** – Pointer to the function to be called every time the slider chang es position. This function should be prototyped as void Foo(int,void*); , where the first parameter is the trackbar position and the second parameter is the u ser data (see the next parameter). If the callback is the NULL pointer, no callb acks are called, but only value is updated. •**userdata** – User data that is passed as is to the callback. It can be used to h andle trackbar events without using global variables.

Trackbar 기능의 활용

- int getTrackbarPos(const string& **trackbarname**, const string& **winname**)

Parameters:

- **trackbarname** – Name of the trackbar.
- **winname** – Name of the window that is the parent of the trackbar.

Trackbar 기능의 활용

- 실습 코드의 이해

```
   (중략~)
// Global Variables
const int alpha_slider_max = 100;
int alpha_slider;
double alpha;
double beta;

Mat image1, image2, dst;

void on_trackbar( int, void* );

int main(int argc, char** argv)
{
        image1=imread("pepper.bmp",1);
        image2=imread("sailboat.bmp",1);

        namedWindow("Display Blend");
        /// Create Trackbars
         char TrackbarName[50];
         sprintf( TrackbarName, "Alpha x %d", alpha_slider_max );

         createTrackbar( TrackbarName, "Display Blend", &alpha_slider, alpha_slider_max, on_trackbar );

         /// Show some stuff
         on_trackbar( alpha_slider, 0 );

         waitKey(0);// Wait for a keystroke in the window

        destroyWindow("Display Blend");// 또는 destroyWindowall();

       return 0;
}
```

Trackbar 기능의 활용

- On_trackbar 함수 정의

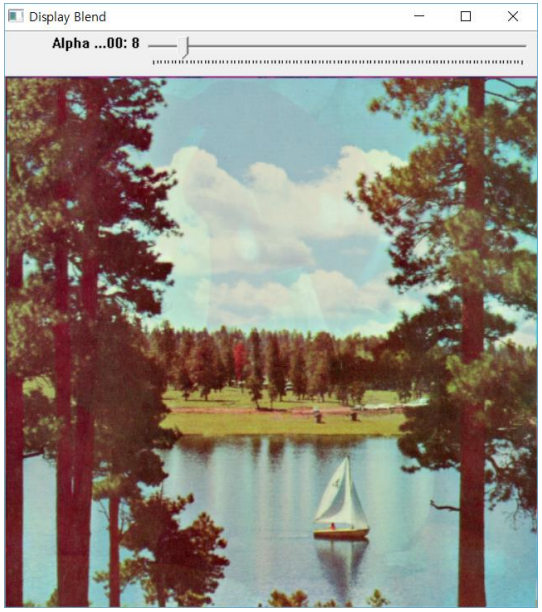
```
  /**
 * @function on_trackbar
 * @brief Callback for trackbar
 */
void on_trackbar( int, void* )
{
 //alpha = (double) alpha_slider/alpha_slider_max ;
 //beta = ( 1.0 - alpha );

 //addWeighted( image1, alpha, image2, beta, 0.0, dst);

  threshold(image1,dst, alpha_slider, 255, THRES_BINARY);
  imshow( "Display Blend", dst );
}
```

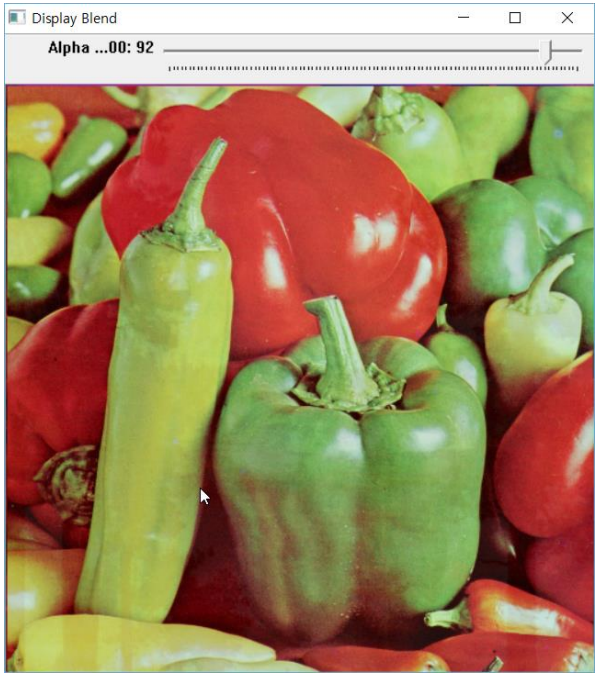
두 영상을 blending

주어진 영상을 이진화

Trackbar 기능의 활용

- 실행 결과: 온라인 스크롤 변수에 따라서 두 장의 영상이 합성됨



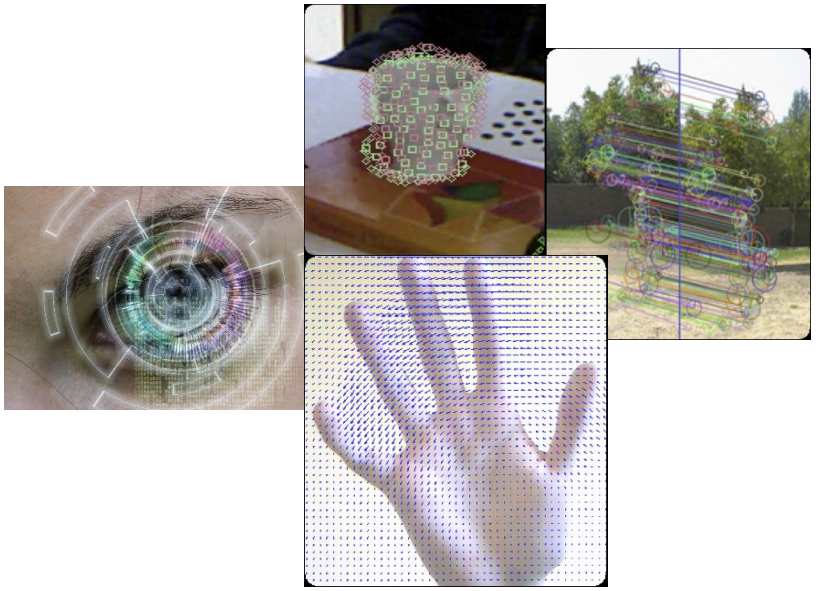


부록: Reference source for your self study...!!!

- 관련 소스
  - http://docs.opencv.org/3.0-beta/doc/tutorials/core/basic_geometric_drawing/basic_geometric_drawing.html
  - http://docs.opencv.org/2.4/doc/tutorials/highgui/trackbar/trackbar.html?highlight=trackbar

COMPUTER VISION 비젼 프로그래밍

Thank you and Question?



IVPL
Intelligent Vision Processing Lab