<OpenCV 4.0.1 기준>

행렬의 데이터타입, 깊이, 몇 차원?

OpenCVExample.cpp*

OpenCVExample (전역 범위)

```cpp
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

int main()
{
    Mat A1(1, 2, DataType<uchar>::type); // row, col, type

    A1.at<uchar>(0, 0) = 1; // A1.at() == A1[]
    A1.at<uchar>(0, 1) = 2;

    cout << "A1" << A1 << endl;
    cout << "depth=" << A1.depth() << ", " << "channels=" << A1.channels() << endl;
    //depth : 자료형 크기. 예를 들면 char은 1byte이므로 2^0 이므로 depth = 0;
    //channel : 행렬의 차원을 뜻함. 여기서는 1차원이므로 channel = 1;


    return 0;
}
```

Point 템플릿 클래스

```cpp
int main()
{
    Point2f pt1(0.1f, 0.2f), pt2(0.3f, 0.4f); //실수형 좌표(2차원)
    //or Point2f pt1 = Point2f(0.1f, 0.2f)

    Point pt3 = (pt1 + pt2)*10.0f; //정수형 좌표(2차원)

    cout << "pt3 : "<<pt3 << endl;
    cout << "x : " << pt3.x << ", y: " << pt3.y << endl; //.x, .y를 통한 각 좌표 확인

    cout << "pt1&pt2 내적 : " << pt1.dot(pt2) << endl; // 두 좌표의 내적 계산.
    // result < 0 : 예각
    // result > 0 : 둔각, = 0이면 직각.

    cout << "원점 <-> pt1 사이의 거리 " << norm(pt1) << endl; // 원점에서의 거리 계산.

    Rect rect(0, 0, 10, 10); //Left_X, Left_Y, width, height
    Point pt(5, 5);

    if (pt.inside(rect)) // 해당 좌표가 해당 사각형 내부의 점인지 확인.
        cout << " pt is inside in rect" << endl;
    else
        cout << " pt is not inside in rect" << endl;
    return 0;
}
```

OpenCVExample (전역 범위)

```cpp
using namespace std;
using namespace cv;

int main()
{
    Point3f pt1(1.0f, 0.0f, 0.0f);
    Point3f pt2(0.0f, 1.0f, 0.0f);

    cout << "pt1&pt2 내적 : " << pt1.dot(pt2) << endl; // 두 좌표의 내적 계산.

    cout << "원점 <-> pt1 사이의 거리 " << norm(pt1) << endl; // 원점에서의 거리 계산.

    cout << "pt1에서 pt2의 외적 : " << pt1.cross(pt2) << endl;
    cout << "pt2에서 pt1의 외적 : " << pt2.cross(pt1) << endl;

    return 0;
}
```

//내적 : 두 벡터가 만드는 평면-> 스칼라 값.

//외적 : 두 벡터가 만드는 평면이 아닌 위치에 두 벡터에 직교하는 제3의 벡터가 존재한다 ->벡터 값

Size 클래스

```
enCVExample                                    (전역 범위)
 1  #include "opencv2/opencv.hpp"
 2
 3  using namespace std;
 4  using namespace cv;
 5
 6  int main()
 7  {
 8      Size rectangle1(30, 20), rectangle2(20, 20);
 9
10      Size Plus = rectangle1 + rectangle2;
11      cout << "각 면적의 가로 세로 길이를 더한 면적 : " << Plus << endl;
12      //Plus.width = 가로 , Plus.height = 세로
13
14      cout << "rectangle1의 넓이 : " << rectangle1.area() << endl;
15
16      return 0;
17  }
18
```

Rect 클래스

```
OpenCVExample                                  (전역 범위)
 1  #include "opencv2/opencv.hpp"
 2
 3  using namespace std;
 4  using namespace cv;
 5
 6  int main()
 7  {
 8      Rect rt1(0, 0, 10, 10), rt2(5, 5, 10, 10);
 9
10      Point pt1(5, 5);
11      Size sz1(20, 20);
12
13      Rect rt3 = rt1 + pt1; // 중심좌표 변경
14      Rect rt4 = rt1 + sz1; // width, height값 변경
15
16      cout << "rt1 : " << rt1 << endl;
17      cout << "rt3 : " << rt3 << endl;
18      cout << "rt4 : " << rt4 << endl;
19      /*
20      rt1: [10 x 10 from(0, 0)]
21      rt3 : [10 x 10 from(5, 5)]
22      rt4 : [30 x 30 from(0, 0)]
23      */
24      return 0;
25  }
26
```

//Rect rt(Top_Left_X, Top_Left_Y, width, height)

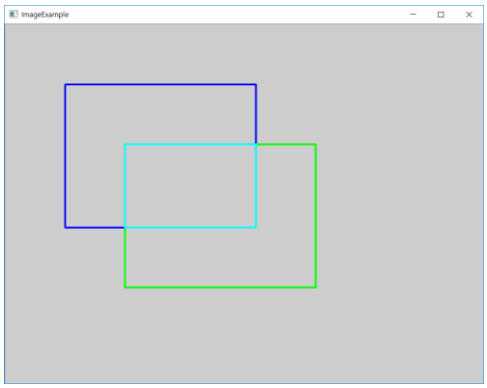
OpenCVExample (전역 범위)

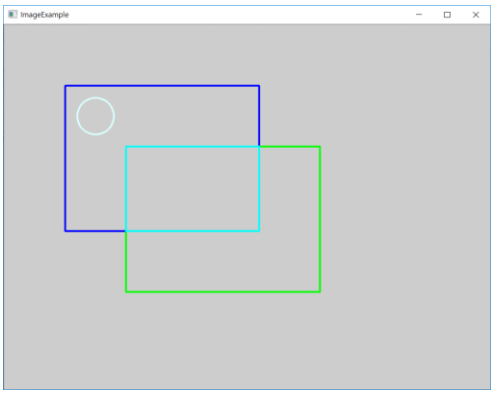
```cpp
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

int main()
{
    Rect rt1(10, 10, 50, 40), rt2(5, 5, 10, 10);

    Point pt1 = rt1.tl(); // top-left 좌표
    Point pt2 = rt1.br(); //bottom-right 좌표

    cout << pt1 << " , " << pt2<<endl;

    Point pt3(20, 20);
    if (rt1.contains(pt3)) // Point.inside()와는 인자가 반대.
        cout << "pt3 is inside in rt1" << endl;

    Rect rt3 = rt1 & rt2;
    Rect rt4 = rt1 | rt2;
    cout << "rt1과 rt2 겹치는 사각형: " << rt3 << endl;
    cout << "rt1과 rt2 를 포함하는 최소크기 사각형: " << rt4 << endl;

    return 0;
}
```

OpenCVExample.cpp*

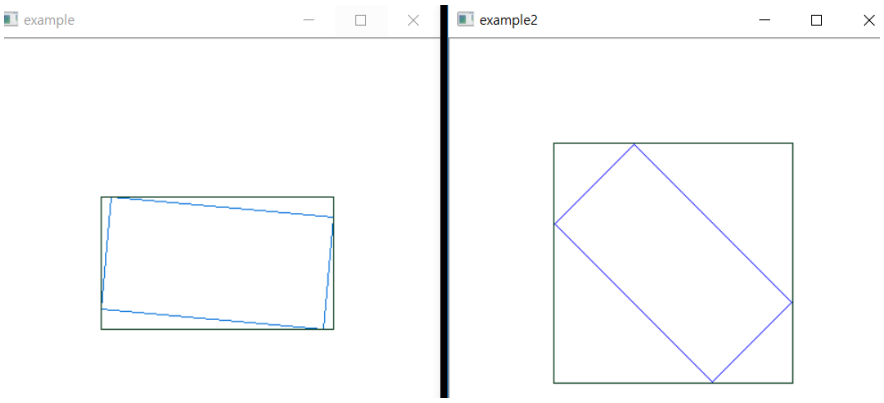
OpenCVExample (전역 범위)

```cpp
int main()
{
    Rect rt1(100, 100, 320, 240), rt2(200, 200, 320, 240);

    Rect rt3 = rt1 & rt2;
    Rect rt4 = rt1 | rt2;

    //그림그리기

    Mat img(600, 800, CV_8UC3); // 8bit 깊이의 uchar 자료형의 3채널 자료형

    namedWindow("ImageExample", WINDOW_AUTOSIZE);

    rectangle(img, rt1, Scalar(255, 0, 0), 2); // 그림그릴곳, 그릴 것, 색상, 두께
    rectangle(img, rt2, Scalar(0, 255, 0), 2);
    rectangle(img, rt3, Scalar(255, 255, 0), 2);
    imshow("ImageExample", img); //imshow("그릴 윈도우 창 이름", 그림이 그려진 행렬 변수)
    waitKey();
    circle(img, Point(150, 150), 30, Scalar(255, 255, 220), 2);

    imshow("ImageExample", img);
    waitKey();

    return 0;
}
```





```cpp
    Point center(200, 200);
    Size sz(100, 200); // (가로, 세로)

    RotatedRect rt1(center, sz, 95.0f); // (사각형의 중심 좌표, 가로&세로, 각도)
    RotatedRect rt2(center, sz, 135.0f);
    //cout << "rt1의 중심 : " << rt1.center.x << ", " << rt1.center.y << endl;
    //cout << "rt1의 회전각도 : " << rt1.angle << endl;

    Point2f points_ex1[4], points_ex2[4];
    rt1.points(points_ex1); //타입은 2f로 해야함, rt1의 꼭지점을 points_ex 변수에 입력.
    rt2.points(points_ex2);
    Rect rt3 = rt1.boundingRect(); // rt2는 rt1을 감싸는 사각형을 만드는 좌표 변수가 저장됨.
    Rect rt4 = rt2.boundingRect();

    Mat img(400, 400, CV_8UC3, Scalar(255, 255, 255));
    Mat img2(400, 400, CV_8UC3, Scalar(255, 255, 255));
    for (int i = 0; i < 4; i++)
    {
        line(img, points_ex1[i], points_ex1[(i + 1) % 4], Scalar(218, 110, 0)); // (그리는 행렬, 시작점, 끝점, 색상)
        line(img2, points_ex2[i], points_ex2[(i + 1) % 4], Scalar(255, 0, 0));
    }

    rectangle(img, rt3, Scalar(22, 52, 0)); // 사각형 그리기
    rectangle(img2, rt4, Scalar(22, 52, 0));
    imshow("example", img);
    imshow("example2", img2);
    waitKey();
```



Matx클래스

```cpp
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

int main()
{

    Matx<float, 2, 3> A(1, 2, 3, 4, 5, 6); //Matx<type, row, col> varName(values...);
    // Matx23f A(1, 2, 3, 4, 5, 6);

    cout << "A : " << A << endl; //[1,2,3;
                                 //  4,5,6]
    Matx13f A_row = A.row(0); // row(index), index에 해당하는 행을 그대로 저장. 열은 .col(index)

    Matx22f A_2by2 = A.get_minor<2, 2>(0, 1); // 부분행렬만들기. <row_size, col_size>(부분행렬 만들 시작 원소 위치)
    Matx22f A_all10 = Matx22f::all(10.0f); //모든 요소값 하나로 통일

    //덧셈,뺄셈, 곱셈은 A+B, A-B, A*B로 가능
    //행렬 A의 각 원소에 5를 곱하려면 A*5
    //A.mul(B)는 같은 위치의 원소의 곱을 말함. 스칼라 곱이 아님.
    //A.dot(B)는 A.mul(B)의 모든 원소의 합
    //A.t()는 A의 전치행렬로, 행과 열은 바꾼 행렬을 말함

    return 0;
}
```

```cpp
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

int main()
{
    Matx33f A = Matx33f::zeros(); // 0으로 초기화
    Matx33f B = Matx33f::ones(); // 1로 초기화
    Matx33f C = Matx33f::eye(); // 행렬의 주대각선(좌상단->우하단) 요소가 모두 1로 초기화

    Matx23f E(1, 2, 3, 4, 5, 6);
    Matx16f D = E.reshape<1, 6>(); // E를 1x6행렬로 변환

    cout << (Mat)D;
    return 0;
}
```

```cpp
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

int main()
{
    Matx22f A(1, -1,
        4, 5);
    Matx22f B(0, 0,
        1, 1);

    Matx22f A_inverse = A.inv(DECOMP_CHOLESKY); // DECOMP_LU 옵션이랑 동일하게 역행렬 계산하는 방법. .inv()는 역행렬 반환

    Matx22f X = A.solve(B); //AX=B의 연립방정식 해인 X 행렬을 반환.
    /*
    Matx22f X;
    solve((Mat)A, (Mat)B, X);랑 같은 의미
    */
    return 0;
}
```

Vec 클래스

```cpp
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

int main()
{
    Vec<float, 3> X(1, 0, 0); // Vec3f X(1,0,0);
    Vec<float, 3> Y(0, 1, 0);

    //Vec3f Z = X.cross(Y);  X와 Y의 외적.
    //Z = X.mul(Y); X와 Y의 각 요소끼리 곱한걸 반환
    //sum(Z); Z의 모든 요소를 더한 값
    //Z = Vec3f::all(0.0); 하나의 값으로 초기화
}
```

sum(Z)는 Scalar로 반환. (0,0,0,0)

Scalar 클래스

```
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

int main()
{
    Scalar X = Vec<float, 4>(1, 2, 3, 4); //벡터로 초기화
    Scalar Y = Scalar(10, 20, 30); // (10,20,30,0)을 의미. 출력하면 0까지 나옴.
    //그밖에
    //Scalar_<자료형>을 통해 uchar, int, float, double 정의 가능.

    return 0;
}
```

Range 클래스

```
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

int main()
{
    Matx33d A(1, 2, 3, 4, 5, 6, 7, 8, 9);

    Mat B(A); //A행렬 그대로 복사.

    B(Range(0, 1), Range(1, 2)); // B(행 범위, 열 범위)
    //Range(int start, int end);는 for(int i=start; i<end; i++)에 해당하는 원소들을 의미함.

    return 0;
}
```

4.0에서는 CvMat, Ptr 대신 cv::Mat, std::vector, cv::threshold()

주로 사용.

Mat 클래스

```cpp
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

int main()
{
    Mat A(2, 2, CV_8UC2); // 채널 수에 따라 원소에 들어각는 값이 달라짐. 만약 c=3이면, (205 205 205 205 205 205 ) -> 1행
    /*
    [ 205, 205, 205, 205,
      205, 205, 205, 205 ]
    */

    Mat B(2, 2, CV_8UC1, Scalar(0)); // 0 으로 초기화

    float data[] = { 1,2,3,4,5,6 };
    Mat C(3, 2, CV_32FC1, data); // 변수를 통해서 초기화

    cout << C;
    return 0;
}
```

```cpp
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

int main()
{
    Vec<float, 3> V(1, 0, 0);
    Mat V1(V); // 3x1 행렬
    //Mat V1(Vec<float, 3> V(1, 0, 0)); 동일

    Matx<float, 3, 3> A(1, 2, 3, 4,5,6,7,8,9);
    Mat A1(A); // Mat A1 = A;

    Mat A2(A1, Range(0, 1), Range::all()); // Mat 변수(대상 행렬, row범위, col범위)
    Mat A3(A1, Rect(0, 0, 2, 1)); // 사각범위를 통해서도 copy가능

    return 0;
}
```

```cpp
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

int main()
{
    int sizes[] = { 2,3,4 };

    Mat A(3, sizes, CV_32FC1, Scalar(0)); // (n차원, 행렬크기, 데이터타입, 초기값)
    //(2x3x4 크기의 3차원 행렬

    cout << A.at<float>(1, 1, 1);
    //3차원 행렬 원소 접근법
    return 0;
}
```