

<OpenCV 4.0.1 기준>

행렬의 데이터타입, 깊이, 몇 차원?

```
OpenCVExample.cpp* x
OpenCVExample (전역 범위)
1 #include "opencv2/opencv.hpp"
2
3 using namespace std;
4 using namespace cv;
5
6 int main()
7 {
8     Mat A1(1, 2, DataType_uchar::type); // row, col, type
9
10    A1.at<uchar>(0, 0) = 1; // A1.at() == A1[]
11    A1.at<uchar>(0, 1) = 2;
12
13    cout << "A1" << A1 << endl;
14    cout << "depth=" << A1.depth() << ", " << "channels=" << A1.channels() << endl;
15    //depth : 자료형 크기. 예를 들면 char은 1byte이므로 2^0 이므로 depth = 0;
16    //channel : 행렬의 차원을 뜻함. 여기서는 1차원이므로 channel = 1;
17
18    return 0;
19 }
20
21
```

Point 템플릿 클래스

```
5
6 int main()
7 {
8     Point2f pt1(0.1f, 0.2f), pt2(0.3f, 0.4f); //실수형 좌표(2차원)
9     //or Point2f pt1 = Point2f(0.1f, 0.2f)
10
11    Point pt3 = (pt1 + pt2)*10.0f; //정수형 좌표(2차원)
12
13    cout << "pt3 : " << pt3 << endl;
14    cout << "x : " << pt3.x << ", y: " << pt3.y << endl; // .x, .y를 통한 각 좌표 확인
15
16    cout << "pt1&pt2 내적 : " << pt1.dot(pt2) << endl; // 두 좌표의 내적 계산.
17    // result < 0 : 예각
18    // result > 0 : 둔각, = 0이면 직각.
19
20    cout << "원점 <=> pt1 사이의 거리 " << norm(pt1) << endl; // 원점에서의 거리 계산.
21
22    Rect rect(0, 0, 10, 10); //Left_X, Left_Y, width, height
23    Point pt(5, 5);
24
25    if (pt.inside(rect)) // 해당 좌표가 해당 사각형 내부의 점인지 확인.
26        cout << " pt is inside in rect" << endl;
27    else
28        cout << " pt is not inside in rect" << endl;
29    return 0;
30 }
31
```

```
OpenCVExample (선택 범위)
2
3 using namespace std;
4 using namespace cv;
5
6 int main()
7 {
8     Point3f pt1(1.0f, 0.0f, 0.0f);
9     Point3f pt2(0.0f, 1.0f, 0.0f);
10
11    cout << "pt1&pt2 내적 : " << pt1.dot(pt2) << endl; // 두 좌표의 내적 계산.
12
13    cout << "원점 <=> pt1 사이의 거리 " << norm(pt1) << endl; // 원점에서의 거리 계산.
14
15    cout << "pt1에서 pt2의 외적 : " << pt1.cross(pt2) << endl;
16    cout << "pt2에서 pt1의 외적 : " << pt2.cross(pt1) << endl;
17
18    return 0;
19 }
20
```

//내적 : 두 벡터가 만드는 평면-> 스칼라 값.

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

Size 클래스

```
OpenCVExample (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
```

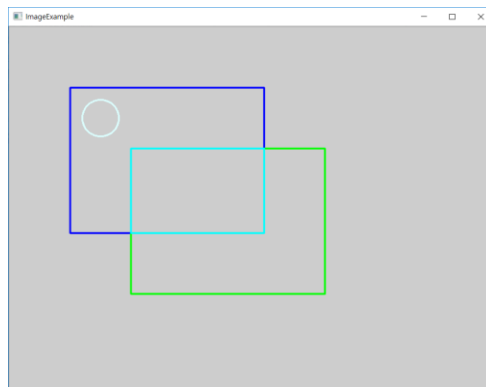
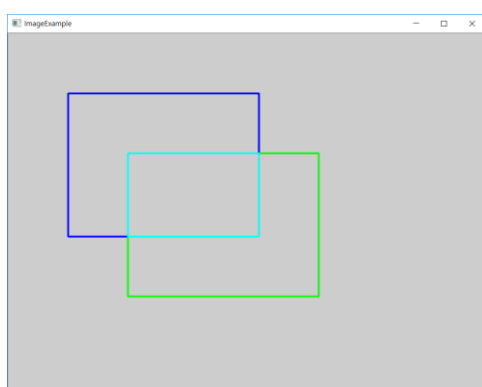
Rect 클래스

```
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)
1 #include "opencv2/opencv.hpp"
2
3 using namespace std;
4 using namespace cv;
5
6 int main()
7 {
8     Rect rt1(10, 10, 50, 40), rt2(5, 5, 10, 10);
9
10    Point pt1 = rt1.tl(); // top-left 좌표
11    Point pt2 = rt1.br(); //bottom-right 좌표
12
13    cout << pt1 << " , " << pt2<<endl;
14
15    Point pt3(20, 20);
16    if (rt1.contains(pt3)) // Point.inside()와는 인자가 반대.
17        cout << "pt3 is inside in rt1" << endl;
18
19    Rect rt3 = rt1 & rt2;
20    Rect rt4 = rt1 | rt2;
21    cout << "rt1과 rt2 겹치는 사각형: " << rt3 << endl;
22    cout << "rt1과 rt2 를 포함하는 최소크기 사각형: " << rt4 << endl;
23
24    return 0;
25 }
26
```

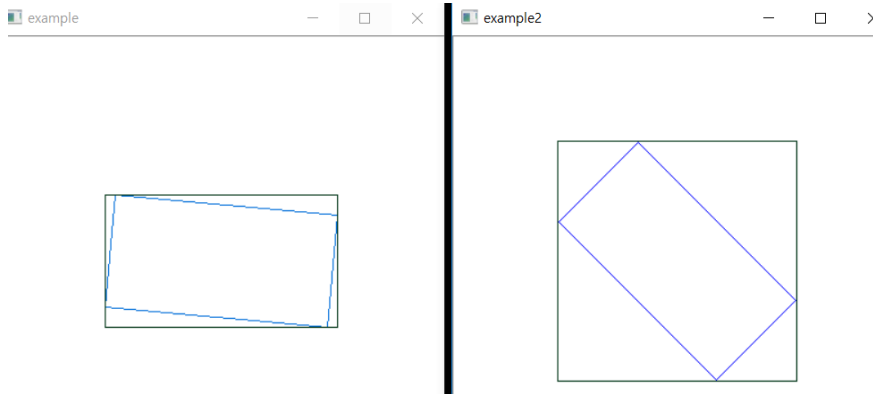
```
OpenCVExample.cpp* x
OpenCVExample (전역 범위)
5
6 int main()
7 {
8     Rect rt1(100, 100, 320, 240), rt2(200, 200, 320, 240);
9
10    Rect rt3 = rt1 & rt2;
11    Rect rt4 = rt1 | rt2;
12
13    //그림그리기
14
15    Mat img(600, 800, CV_8UC3); // 8bit 깊이의 uchar 자료형의 3채널 자료형
16
17    namedWindow("ImageExample", WINDOW_AUTOSIZE);
18
19    rectangle(img, rt1, Scalar(255, 0, 0), 2); // 그림그릴곳, 그릴 것, 색상, 두께
20    rectangle(img, rt2, Scalar(0, 255, 0), 2);
21    rectangle(img, rt3, Scalar(255, 255, 0), 2);
22    imshow("ImageExample", img); //imshow("그릴 윈도우 창 이름", 그림이 그려진 행렬 변수)
23    waitKey();
24    circle(img, Point(150, 150), 30, Scalar(255, 255, 220), 2);
25
26    imshow("ImageExample", img);
27    waitKey();
28
29    return 0;
30 }
31
```



```

8   Point center(200, 200);
9   Size sz(100, 200); // (가로, 세로)
10
11   RotatedRect rt1(center, sz, 95.0f); // (사각형의 중심 좌표, 가로&세로, 각도)
12   RotatedRect rt2(center, sz, 135.0f);
13   //cout << "rt1의 중심 : " << rt1.center.x << ", " << rt1.center.y << endl;
14   //cout << "rt1의 회전각도 : " << rt1.angle << endl;
15
16   Point2f points_ex1[4], points_ex2[4];
17   rt1.points(points_ex1); //타입은 2f로 해야함, rt1의 꼭지점을 points_ex 변수에 입력.
18   rt2.points(points_ex2);
19   Rect rt3 = rt1.boundingRect(); // rt2는 rt1을 감싸는 사각형을 만드는 좌표 변수가 저장됨.
20   Rect rt4 = rt2.boundingRect();
21
22   Mat img(400, 400, CV_8UC3, Scalar(255, 255, 255));
23   Mat img2(400, 400, CV_8UC3, Scalar(255, 255, 255));
24   for (int i = 0; i < 4; i++)
25   {
26       line(img, points_ex1[i], points_ex1[(i + 1) % 4], Scalar(218, 110, 0)); // (그리는 행렬, 시작점, 끝점, 색상)
27       line(img2, points_ex2[i], points_ex2[(i + 1) % 4], Scalar(255, 0, 0));
28   }
29
30   rectangle(img, rt3, Scalar(22, 52, 0)); // 사각형 그리기
31   rectangle(img2, rt4, Scalar(22, 52, 0));
32   imshow("example", img);
33   imshow("example2", img2);
34   waitKey();

```



Matx클래스

```

1   #include "opencv2/opencv.hpp"
2
3   using namespace std;
4   using namespace cv;
5
6   int main()
7   {
8
9       Matx<float, 2, 3> A(1, 2, 3, 4, 5, 6); //Matx<type, row, col> varName(values...);
10      // Matx23f A(1, 2, 3, 4, 5, 6);
11
12      cout << "A : " << A << endl; // [1,2,3;
13                                     // 4,5,6]
14      Matx13f A_row = A.row(0); // row(index), index에 해당하는 행을 그대로 저장, 열은 .col(index)
15
16      Matx22f A_2by2 = A.get_minor<2, 2>(0, 1); // 부분행렬만들기. <row_size, col_size>(부분행렬 만들 시작 원소 위치)
17      Matx22f A_all10 = Matx22f::all(10.0f); // 모든 요소값 하나로 통일
18
19      //덧셈, 뺄셈, 곱셈은 A+B, A-B, A*B로 가능
20      //행렬 A의 각 원소에 5를 곱하려면 A*5
21      //A.mul(B)는 같은 위치의 원소의 곱을 말함. 스칼라 곱이 아님.
22      //A.dot(B)는 A.mul(B)의 모든 원소의 합
23      //A.t()는 A의 전치행렬로, 행과 열은 바꾼 행렬을 말함
24
25      return 0;
26  }
27

```

```

1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      Matx33f A = Matx33f::zeros(); // 0으로 초기화
9      Matx33f B = Matx33f::ones(); // 1로 초기화
10     Matx33f C = Matx33f::eye(); // 행렬의 주대각선(좌상단->우하단) 요소가 모두 1로 초기화
11
12     Matx23f E(1, 2, 3, 4, 5, 6);
13     Matx16f D = E.reshape<1, 6>(); // E를 1x6행렬로 변환
14
15     cout << (Mat)D;
16     return 0;
17 }
18

```

```

1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      Matx22f A(1, -1,
9      4, 5);
10     Matx22f B(0, 0,
11     1, 1);
12
13     Matx22f A_inverse = A.inv(DECOMP_CHOLESKY); // DECOMP_LU 옵션이랑 동일하게 역행렬 계산하는 방법. .inv()는 역행렬 반환
14
15     Matx22f X = A.solve(B); // AX=B의 연립방정식 해인 X 행렬을 반환.
16     /*
17     Matx22f X;
18     solve((Mat)A, (Mat)B, X);랑 같은 의미
19     */
20     return 0;
21 }
22

```

Vec 클래스

```

1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      Vec<float, 3> X(1, 0, 0); // Vec3f X(1,0,0);
9      Vec<float, 3> Y(0, 1, 0);
10
11     //Vec3f Z = X.cross(Y); X와 Y의 외적.
12     //Z = X.mul(Y); X와 Y의 각 요소끼리 곱한걸 반환
13     //sum(Z); Z의 모든 요소를 더한 값
14     //Z = Vec3f::all(0.0); 하나의 값으로 초기화
15 }
16

```

sum(Z)는 Scalar로 반환. (0,0,0,0)

Scalar 클래스

```
1      #include "opencv2/opencv.hpp"
2
3      using namespace std;
4      using namespace cv;
5
6      int main()
7      {
8          Scalar X = Vec<float, 4>(1, 2, 3, 4); //벡터로 초기화
9          Scalar Y = Scalar(10, 20, 30); // (10,20,30,0)을 의미. 출력하면 0까지 나옴.
10         //그밖에
11         //Scalar_<자료형>을 통해 uchar, int, float, double 정의 가능.
12
13         return 0;
14     }
15
```

Range 클래스

```
1      #include "opencv2/opencv.hpp"
2
3      using namespace std;
4      using namespace cv;
5
6      int main()
7      {
8          Matx33d A(1, 2, 3, 4, 5, 6, 7, 8, 9);
9
10         Mat B(A); //A행렬 그대로 복사.
11
12         B(Range(0, 1), Range(1, 2)); // B(행 범위, 열 범위)
13         //Range(int start, int end);는 for(int i=start; i<end; i++)에 해당하는 원소들을 의미함.
14
15         return 0;
16     }
17
```

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

Mat 클래스

```
1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      Mat A(2, 2, CV_8UC2); // 채널 수에 따라 원소에 들어가는 값이 달라짐. 만약 c=3이면, (205 205 205 205 205 205 ) -> 1행
9      /*
10     [ 205, 205, 205, 205,
11       205, 205, 205, 205 ]
12     */
13
14     Mat B(2, 2, CV_8UC1, Scalar(0)); // 0 으로 초기화
15
16     float data[] = { 1,2,3,4,5,6 };
17     Mat C(3, 2, CV_32FC1, data); // 변수를 통해서 초기화
18
19     cout << C;
20     return 0;
21 }
22
```

```
1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      Vec<float, 3> V(1, 0, 0);
9      Mat V1(V); // 3x1 행렬
10     //Mat V1(Vec<float, 3> V(1, 0, 0)); 동일
11
12     Matx<float, 3, 3> A(1, 2, 3, 4,5,6,7,8,9);
13     Mat A1(A); // Mat A1 = A;
14
15     Mat A2(A1, Range(0, 1), Range::all()); // Mat 변수(대상 행렬, row범위, col범위)
16     Mat A3(A1, Rect(0, 0, 2, 1)); // 사각범위를 통해서도 copy가능
17
18     return 0;
19 }
20
```

```
1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      int sizes[] = { 2,3,4 };
9
10     Mat A(3, sizes, CV_32FC1, Scalar(0)); // (n차원, 행렬크기, 데이터타입, 초기값)
11     //(2x3x4 크기의 3차원 행렬
12
13     cout << A.at<float>(1, 1, 1);
14     //3차원 행렬 원소 접근법
15     return 0;
16 }
17
```

```

1      #include "opencv2/opencv.hpp"
2
3      using namespace std;
4      using namespace cv;
5
6      int main()
7      {
8          Mat A(2,3, CV_32FC1, Scalar(0)); // (row, col, type, init_var)
9
10         A.create(Size(3,3), CV_8UC1); //이전 메모리 해제 및 메모리 재할당
11         cout << A;
12
13         Mat B;
14         int sizes[] = { 3,3 };
15         B.create(2, sizes, CV_8UC1); // (dimension, size_var, type)
16         cout << B;
17
18         return 0;
19     }
20

```

```

1      #include "opencv2/opencv.hpp"
2
3      using namespace std;
4      using namespace cv;
5
6      int main()
7      {
8          Mat A;
9          A.create(512, 512, CV_8UC3); // row, col, type
10
11          for (int i = 0; i < A.rows; i++)
12              for (int j = 0; j < A.cols; j++)
13                  A.at<Vec3b>(i, j) = Vec3b(255, 255, 255);
14
15          imshow("example", A);
16          waitKey();
17
18          return 0;
19     }
20

```



```

1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      Mat A;
9      A.create(4, 5, CV_32FC3);
10     //Mat B = A -> A와 B가 같은 메모리 공유
11     A.at<Vec3f>(0, 0) = Vec3f(0.75, 1.0, 10.0);
12
13     cout << *(float*)(A.data+4) << endl; // .data는 주소값을 반환, 32bit = 4byte
14     cout << A.isContinuous() << endl;
15     cout << A.total() << endl; // 모든 요소의 개수
16     cout << A.elemSize() << endl; // 요소 하나의 크기. 여기서는 float x 3ch = 12byte
17     cout << A.elemSize1() << endl; // 한 채널의 요소 하나의 크기. float = 4byte
18     cout << A.type() << endl;
19
20     cout << A.step << endl; // dpth x elemSize() = 5x12=60
21     cout << A.step1() << endl; // step / elemSize1() = 60/4=15
22     cout << A.empty() << endl; // false이면 0
23     cout << A.size() << endl; // [row x col]
24
25
26     return 0;
27 }

```

```

1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      Mat A(3, 3, CV_32F);
9      for (int i = 0; i < A.rows; i++)
10     {
11         float* ptr = A.ptr<float>(i); // 지정된 행의 시작주소를 반환
12         for (int j = 0; j < A.cols; j++)
13             ptr[j] = i * A.cols + j;
14     }
15
16     cout << A;
17
18     return 0;
19 }

```

```

1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      Mat A(3, 3, CV_32F);
9      for (int i = 0; i < A.rows; i++)
10     {
11         float* ptr = A.ptr<float>(i);
12         for (int j = 0; j < A.cols; j++)
13             ptr[j] = i * A.cols + j;
14     }
15
16     A.row(2).copyTo(A.row(1)); // 2열에 있는 값을 1열로 복사. = op로는 copy가 되지 않음
17
18     cout << A;
19
20     return 0;
21 }

```

```

1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      Mat A(3, 3, CV_32F);
9      for (int i = 0; i < A.rows; i++)
10     {
11         for (int j = 0; j < A.cols; j++)
12             A.at<float>(i,j) = float(i * A.cols + j);
13     }
14
15     Mat B = A.clone(); //복사
16     A.copyTo(B); //복사할 대상.copyTo(복사받을 대상)
17     A.assignTo(B, CV_8U); //타입은 생략 가능
18
19     Mat C;
20     A.convertTo(C, CV_8U, 10.0, 1.0); //형전환. (전환값 저장 대상, 타입, 곱셈할 변수_생략가능, 덧셈할 변수_생략가능)
21
22     Mat D = A.reshape(0, 1); // 채널 =0, 행 개수 = 1 개로 행렬변환
23
24     A.setTo(Scalar::all(0)); // 모든 값 0으로 변경
25
26     return 0;
27 }
28

```

```

1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      Mat A(3, 3, CV_32F);
9      for (int i = 0; i < A.rows; i++) { ... }
10
11      A.resize(5, Scalar::all(0)); // 행의 개수를 5로 변경, (변경할 크기, 새로 추가될 행렬 요소의 값)
12
13      A.reserve(10); // 10개의 행에 대한 메모리 공간을 확보. 이미 충분하면 아무일도 하지 않음.
14
15      A.release(); // A.data = Null로 함. 메모리 명시적 해제.
16
17      return 0;
18  }
19
20
21
22
23

```

```

1  #include "opencv2/opencv.hpp"
2
3  using namespace std;
4  using namespace cv;
5
6  int main()
7  {
8      Mat A(3, 3, CV_32F);
9      for (int i = 0; i < A.rows; i++) { ... }
10
11      Mat B = A(Range(5, 8), Range(3, 6));
12
13      Size wholeSize; Point ofs;
14      B.locateROI(wholeSize, ofs);
15      //wholeSize에는 원본행렬의 크기, ofs에는 [열,행]으로 부분행렬이 시작하는 위치 저장
16      //0행 0열.
17
18      B.adjustROI(1, 1, 1, 1); // 행렬을 상하좌우로 각각 변수만큼 크기를 증가시킴
19
20      return 0;
21  }
22
23
24
25
26

```