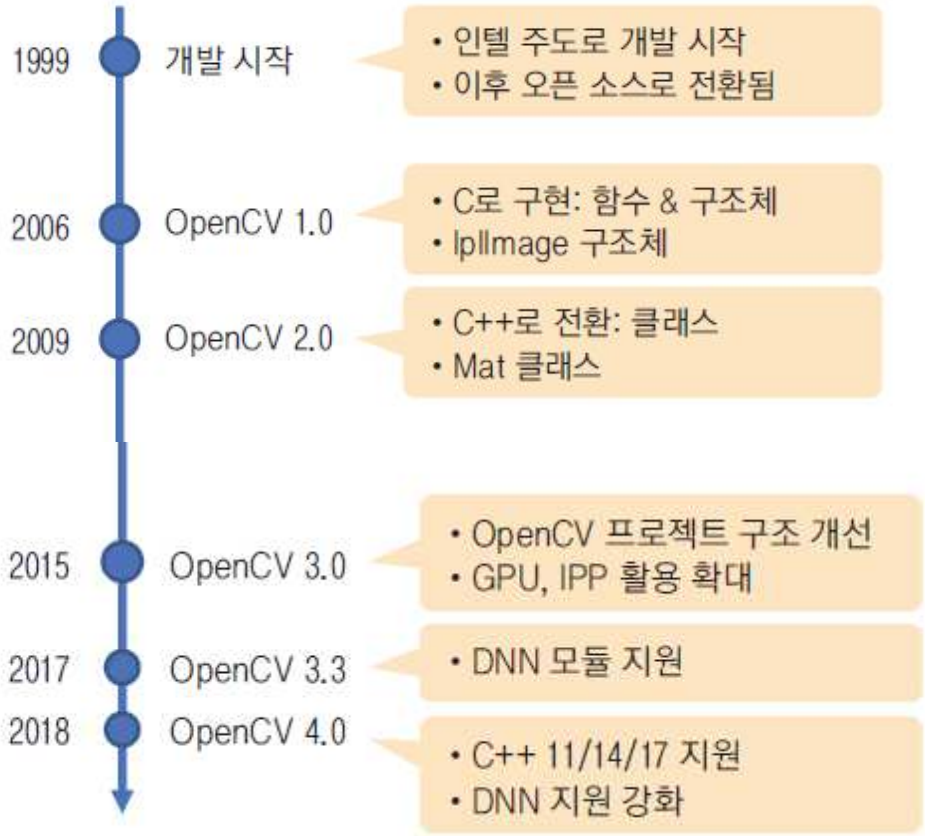
OpenCV 개요

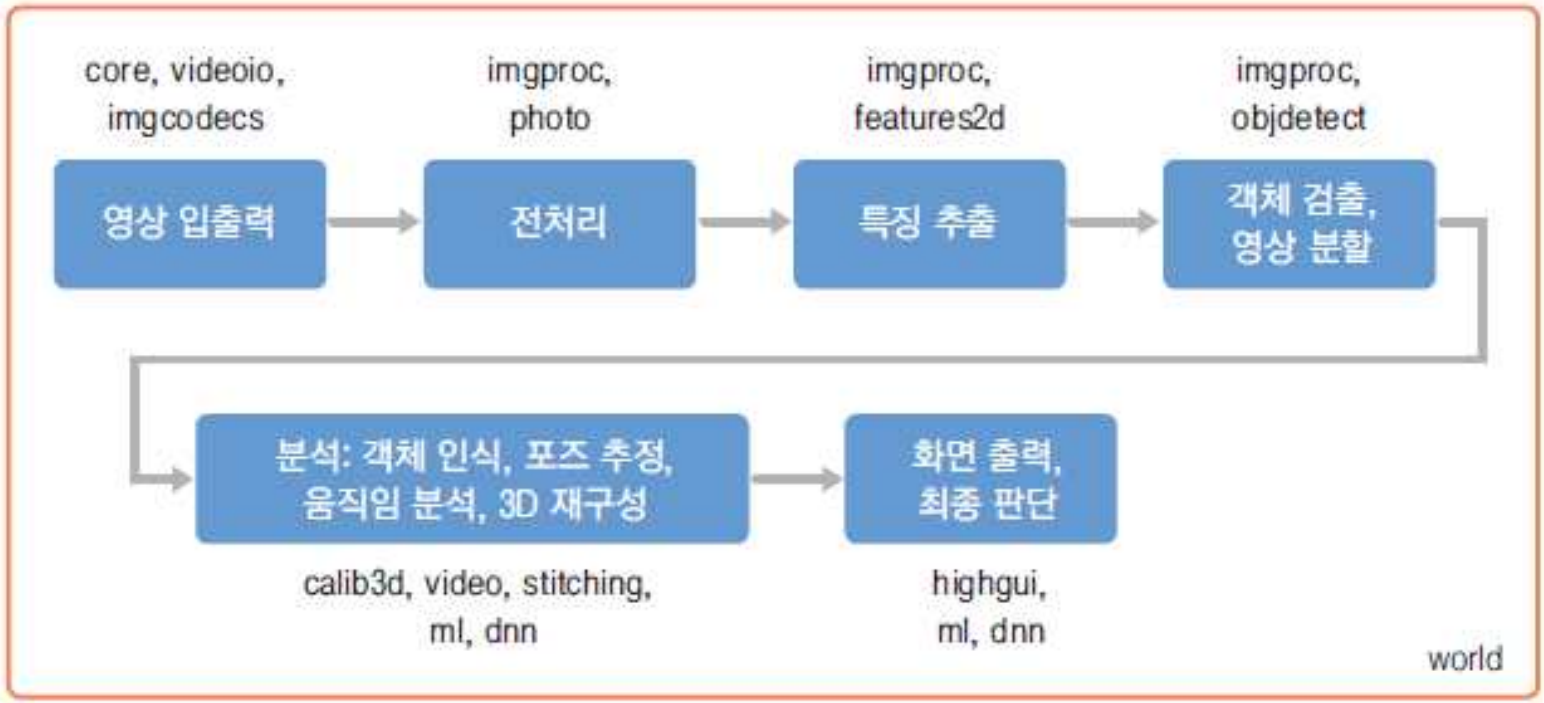
OpenCV

- Version History



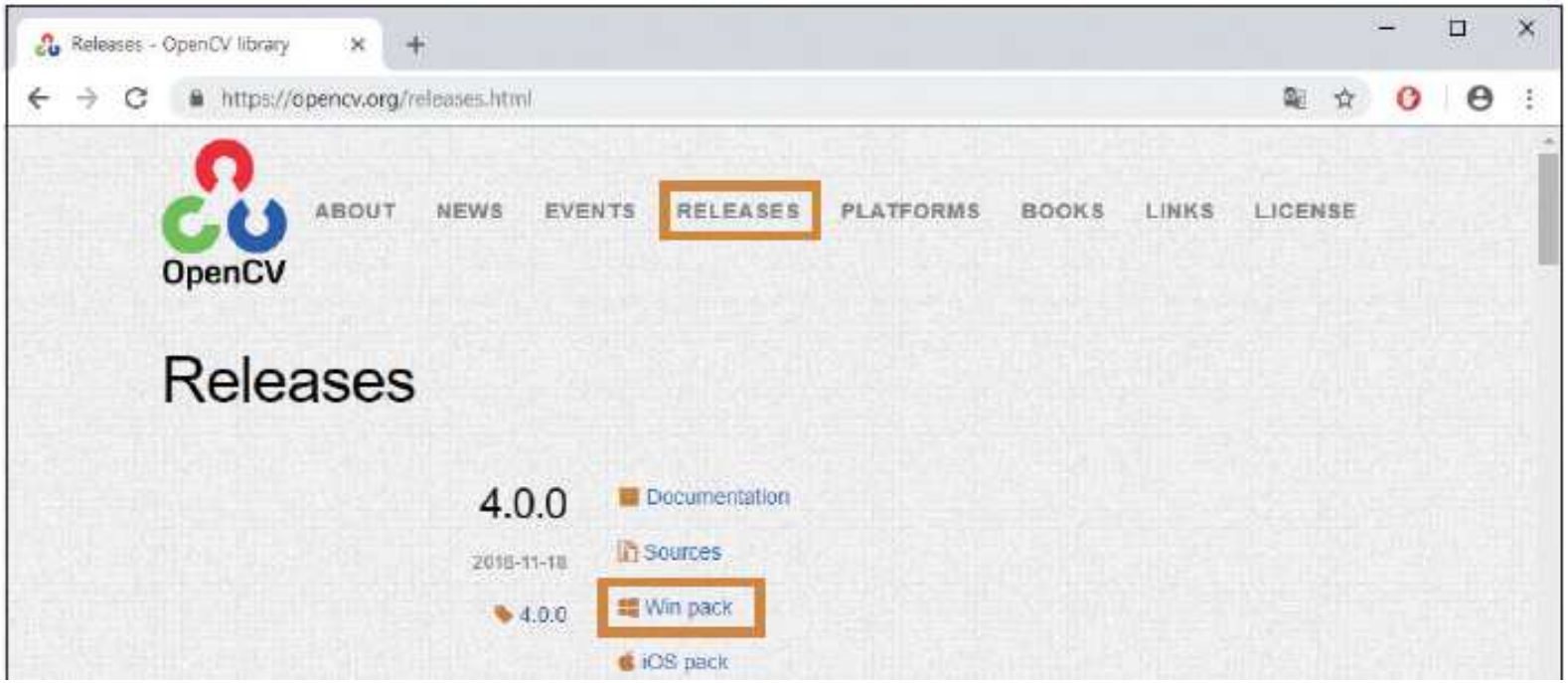
OpenCV

- 주요 module



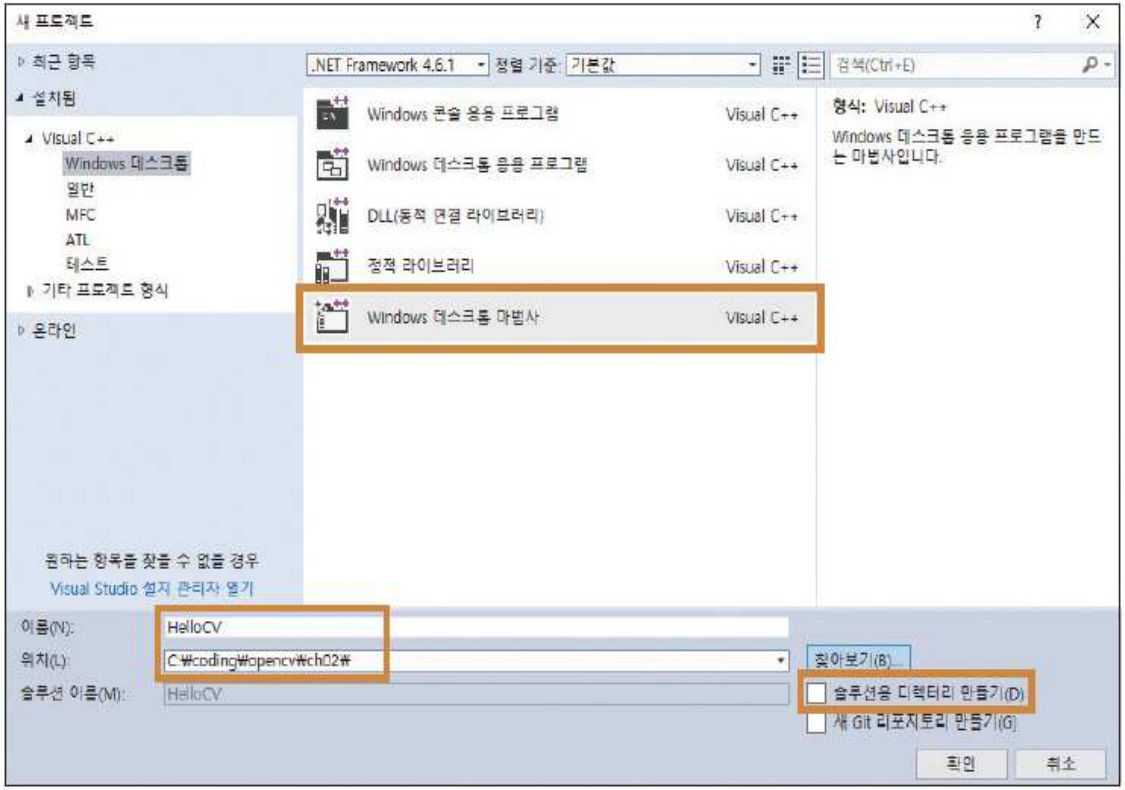
설치

- OpenCV 공식 사이트
  - https://opencv.org/



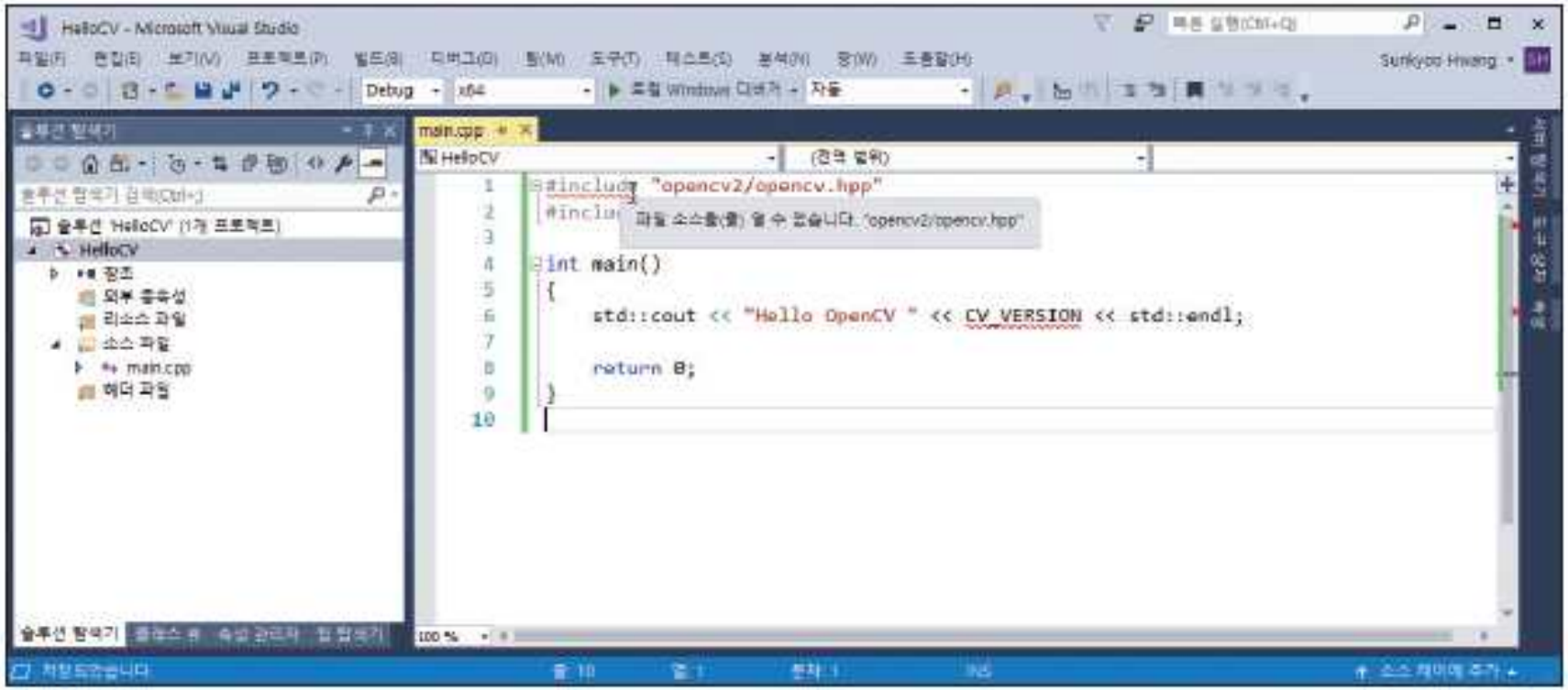
Visual Studio

- HelloCV 프로젝트



Visual Studio

- Main.cpp

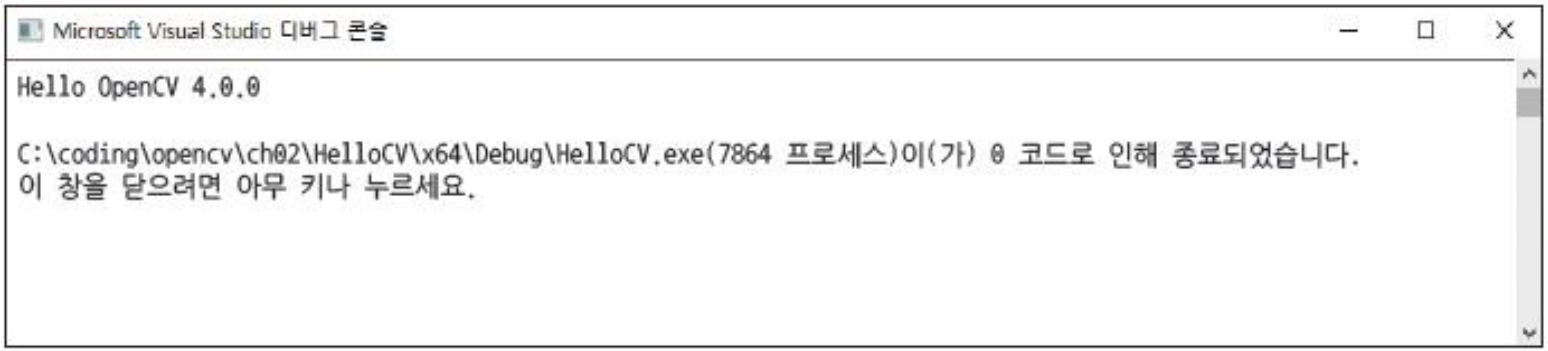


Visual Studio

- Project Property (속성)

	Debug 모드	Release 모드
추가 포함 디렉터리	$(OPENCV_DIR)\include	
추가 라이브러리 디렉터리	$(OPENCV_DIR)\x64\vc15\lib	
추가 종속성	opencv_world400d.lib	opencv_world400.lib

- 실행결과

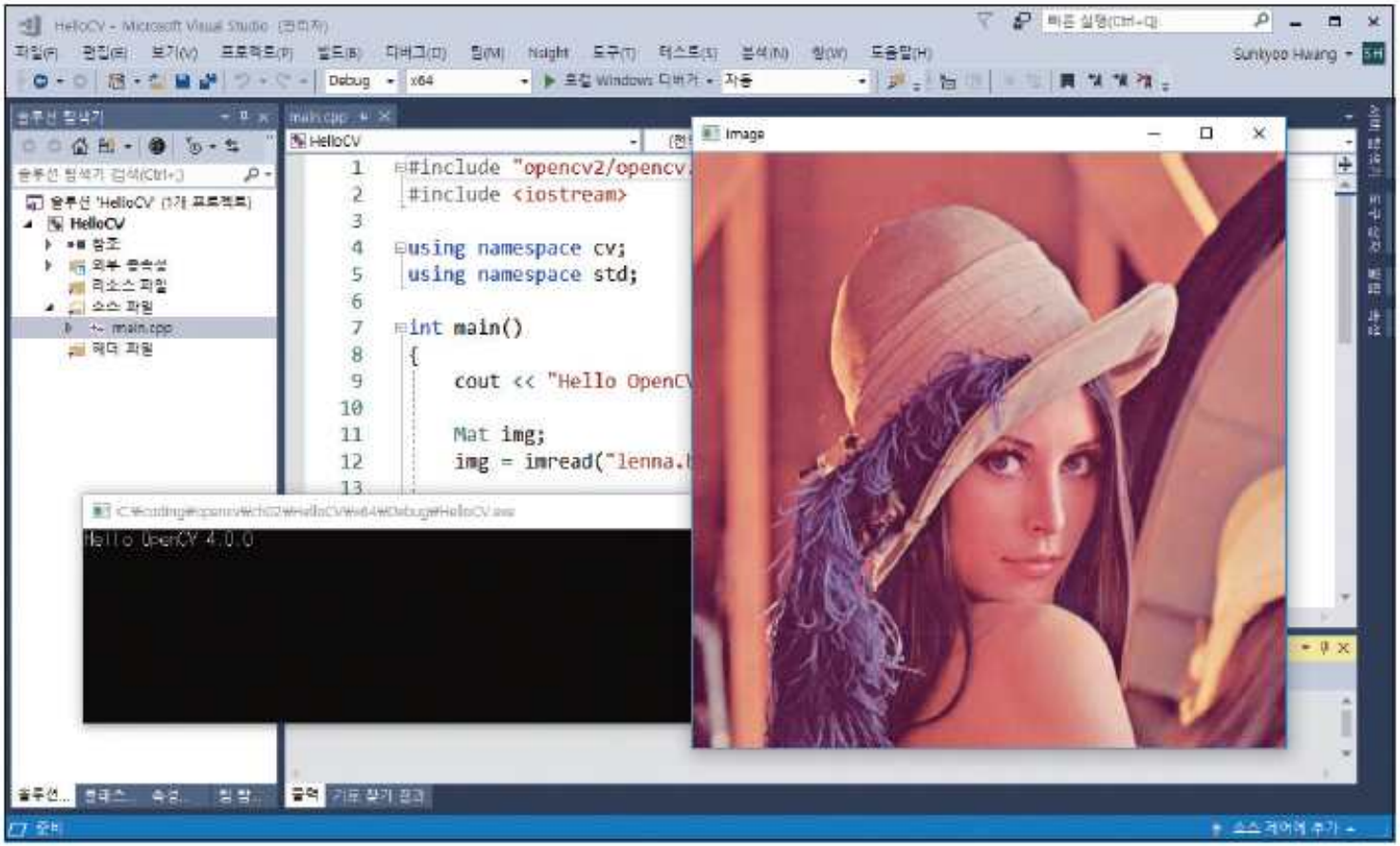


영상출력 예제

코드 2-3 using namespace 구문을 사용한 HelloCV 소스 코드 [ch02/HelloCV]

```
01  #include "opencv2/opencv.hpp"
02  #include <iostream>
03
04  using namespace cv;
05  using namespace std;
06
07  int main()
08  {
09      cout << "Hello OpenCV " << CV_VERSION << endl;
10
11      Mat img;
12      img = imread("lenna.bmp");
13
14      if (img.empty()) {
15          cerr << "Image load failed!" << endl;
16          return -1;
17      }
18
19      namedWindow("image");
20      imshow("image", img);
21
22      waitKey(0);
23      return 0;
24  }
```

영상출력 예제


#include "opencv2/opencv.
#include <iostream>
using namespace cv;
using namespace std;
int main()
{
cout << "Hello OpenCV
Mat img;
img = imread("lenna.
Hello OpenCV 4.0.0
Image

기본 클래스

- Point_

코드 3-1 간략화한 Point_ 클래스 정의와 이름 재정의

```
01  template<typename _Tp> class Point_
02  {
03  public:
04      Point_();
05      Point_(_Tp _x, _Tp _y);
06      Point_(const Point_& pt);
07
08      Point_& operator = (const Point_& pt);
09
10      _Tp dot(const Point_& pt) const;
11      double ddot(const Point_& pt) const;
12      double cross(const Point_& pt) const;
13      bool inside(const Rect_<_Tp>& r) const;
14      ...
15
16      _Tp x, y;
17  };
18
19  typedef Point_<int>    Point2i;
20  typedef Point_<int64>  Point2l;
21  typedef Point_<float>  Point2f;
22  typedef Point_<double> Point2d;
23  typedef Point2i        Point;
```

기본 클래스

- Point_

```
Point pt1;                  // pt1 = (0, 0)
pt1.x = 5; pt1.y = 10;      // pt1 = (5, 10)
Point pt2(10, 30);          // pt2 = (10, 30)

// pt1 = [5, 10], pt2 = [10, 30]
Point pt3 = pt1 + pt2;              // pt3 = [15, 40]
Point pt4 = pt1 * 2;                // pt4 = [10, 20]
int d1 = pt1.dot(pt2);              // d1 = 350
bool b1 = (pt1 == pt2);             // b1 = false
```

기본 클래스

- Size_

코드 3-2 간략화한 Size_ 클래스 정의와 이름 재정의

```
01  template<typename _Tp> class Size_
02  {
03  public:
04      Size_();
05      Size_(_Tp _width, _Tp _height);
06      Size_(const Size_& sz);
07
08      Size_& operator = (const Size_& sz);
09
10      _Tp area() const;
11      bool empty() const;
12
13      _Tp width, height;
14  };
15
16  typedef Size_<int>    Size2i;
17  typedef Size_<int64>  Size2l;
18  typedef Size_<float>  Size2f;
19  typedef Size_<double> Size2d;
20  typedef Size2i        Size;
```

기본 클래스

- Size_

```
// sz1 = [5 x 10], sz2 = [10 x 20]
Size sz3 = sz1 + sz2;    // sz3 = [15 x 30]
Size sz4 = sz1 * 2;      // sz4 = [10 x 20]
int area1 = sz4.area(); // area1 = 200
```

```
cout << "sz3: " << sz3 << endl;
cout << "sz4: " << sz4 << endl;
```

```
sz3: [15 x 30]
sz4: [10 x 20]
```

기본 클래스

- Rect_

코드 3-3 간략화한 Rect_ 클래스 정의와 이름 재정의

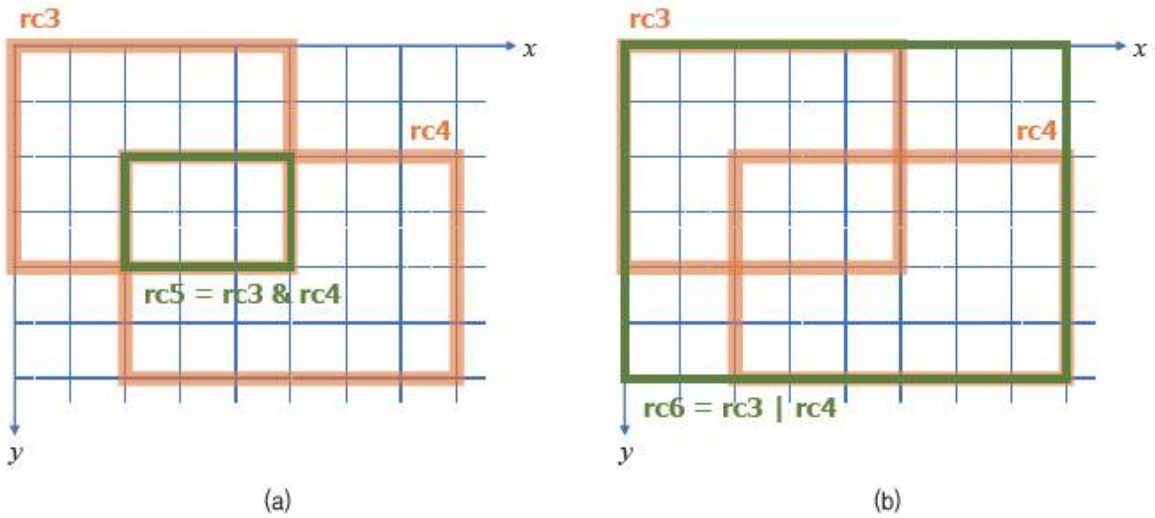
```
01  template<typename _Tp> class Rect_
02  {
03  public:
04      Rect_();
05      Rect_(_Tp _x, _Tp _y, _Tp _width, _Tp _height);
06      Rect_(const Rect_& r);
07      Rect_(const Point_<_Tp>& org, const Size_<_Tp>& sz);
08      Rect_(const Point_<_Tp>& pt1, const Point_<_Tp>& pt2);
09
10      Rect_& operator = ( const Rect_& r );
11
12      Point_<_Tp> tl() const;
13      Point_<_Tp> br() const;
14      Size_<_Tp> size() const;
15      _Tp area() const;
16      bool empty() const;
17      bool contains(const Point_<_Tp>& pt) const;
18
19      _Tp x, y, width, height;
20  };
21
22  typedef Rect_<int>    Rect2i;
23  typedef Rect_<float>  Rect2f;
24  typedef Rect_<double> Rect2d;
25  typedef Rect2i        Rect;
```

기본 클래스

- Rect_

```
// rc1 = [0 x 0 from (0, 0)], rc2 = [60 x 40 from (10, 10)]
Rect rc3 = rc1 + Size(50, 40);    // rc3 = [50 x 40 from (0, 0)]
Rect rc4 = rc2 + Point(10, 10);   // rc4 = [60 x 40 from (20, 20)]


// rc3 = [50 x 40 from (0, 0)], rc4 = [60 x 40 from (20, 20)]
Rect rc5 = rc3 & rc4;             // rc5 = [30 x 20 from (10, 10)]
Rect rc6 = rc3 | rc4;             // rc6 = [80 x 60 from (0, 0)]
```

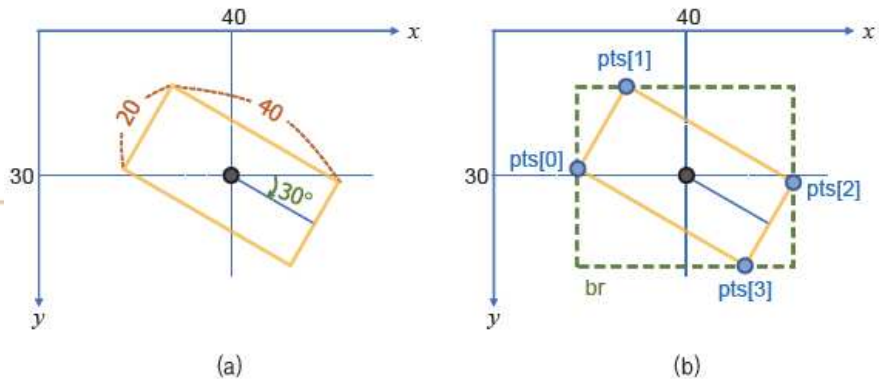


기본 클래스

- RotatedRect

코드 3-4 간략화한 RotatedRect 클래스 정의

```
01  class RotatedRect
02  {
03  public:
04      RotatedRect();
05      RotatedRect(const Point2f& _center, const Size2f& _size, float _angle);
06      RotatedRect(const Point2f& point1, const Point2f& point2, const Point2f& point3);
07
08      void points(Point2f pts[]) const;
09      Rect boundingRect() const;
10      Rect_<float> boundingRect2f() const;
11
12      Point2f center;
13      Size2f size;
14      float angle;
15  };
```



Mat 클래스

- 클래스 정의

코드 3-6 간략화한 Mat 클래스 정의

```
01  class Mat
02  {
03  public:
04      Mat();
05      Mat(int rows, int cols, int type);
06      Mat(Size size, int type);
07      Mat(int rows, int cols, int type, const Scalar& s);
08      Mat(Size size, int type, const Scalar& s);
09      Mat(const Mat& m);
10      ~Mat();
11
12      void create(int rows, int cols, int type);
13      bool empty() const;
14
15      Mat clone() const;
16      void copyTo(OutputArray m) const;
17      Mat& setTo(InputArray value, InputArray mask=noArray());
18
19      static MatExpr zeros(int rows, int cols, int type);
20      static MatExpr ones(int rows, int cols, int type);
21
22      Mat& operator = (const Mat& m);
23      Mat operator()( const Rect& roi ) const;
24
25      template<typename _Tp> _Tp* ptr(int i0 = 0);
26      template<typename _Tp> _Tp& at(int row, int col);
27
28      int dims;
29      int rows, cols;
30      uchar* data;
31      MatSize size;
32      ...
33  };
```

Mat 클래스

- Type

```
#define CV_8U   0       // uchar, unsigned char
#define CV_8S   1       // schar, signed char
#define CV_16U  2       // ushort, unsigned short
#define CV_16S  3       // signed short
#define CV_32S  4       // int
#define CV_32F  5       // float
#define CV_64F  6       // double
#define CV_16F  7       // float16_t
```

- Mat 객체 생성

```
Mat img2(480, 640, CV_8UC1);      // unsigned char, 1-channel
Mat img3(480, 640, CV_8UC3);      // unsigned char, 3-channels

Mat img4(Size(640, 480), CV_8UC3);      // Size(width, height)

Mat img5(480, 640, CV_8UC1, Scalar(128));          // initial values, 128
Mat img6(480, 640, CV_8UC3, Scalar(0, 0, 255));    // initial values, red
```

Blue, Green, Red

Vec 클래스

- 벡터 데이터 표현

코드 3-14 간략화한 Matx와 Vec 클래스 정의

```
01  template<typename _Tp, int m, int n> class Matx
02  {
03  public:
04      ...
05      _Tp val[m*n]; //< matrix elements
06  };
07
08  template<typename _Tp, int cn> class Vec : public Matx<_Tp, cn, 1>
09  {
10  public:
11      ...
12      /*! element access */
13      _Tp& operator[](int i);
14  };
```

(ex)
```
Vec<uchar, 3> p1, p2(0, 0, 255);
p1.val[0] = 100;
p1[0] = 100;
```

Vec 클래스

- 재정의

```
typedef Vec<uchar, 2> Vec2b;
typedef Vec<uchar, 3> Vec3b;
typedef Vec<uchar, 4> Vec4b;

typedef Vec<short, 2> Vec2s;
typedef Vec<short, 3> Vec3s;
typedef Vec<short, 4> Vec4s;

typedef Vec<ushort, 2> Vec2w;
typedef Vec<ushort, 3> Vec3w;
typedef Vec<ushort, 4> Vec4w;
```

```
typedef Vec<int, 2> Vec2i;
typedef Vec<int, 3> Vec3i;
typedef Vec<int, 4> Vec4i;
typedef Vec<int, 6> Vec6i;
typedef Vec<int, 8> Vec8i;

typedef Vec<float, 2> Vec2f;
typedef Vec<float, 3> Vec3f;
typedef Vec<float, 4> Vec4f;
typedef Vec<float, 6> Vec6f;

typedef Vec<double, 2> Vec2d;
typedef Vec<double, 3> Vec3d;
typedef Vec<double, 4> Vec4d;
typedef Vec<double, 6> Vec6d;
```

(ex) `Vec3b p1, p2(0, 0, 255);`

Scalar 클래스

- 영상의 픽셀값 표현
  - 원소 크기 : 4개 이하

```
Scalar(밝기)
Scalar(파란색, 녹색, 빨간색)
Scalar(파란색, 녹색, 빨간색, 투명도)        // PNG 형식
```

Scalar 클래스

코드 3-16 Scalar 클래스 사용법 [ch03/ScalarOp]

```
01  void ScalarOp()
02  {
03      Scalar gray = 128;
04      cout << "gray: " << gray << endl;
05
06      Scalar yellow(0, 255, 255);
07      cout << "yellow: " << yellow << endl;
08
09      Mat img1(256, 256, CV_8UC3, yellow);
10
11      for (int i = 0; i < 4; i++)
12          cout << yellow[i] << endl;
13  }
```

(출력)

```
gray: [128, 0, 0, 0]
yellow: [0, 255, 255, 0]
0
255
255
0
```

InputArray, OutputArray

- 정의

```
typedef const _InputArray& InputArray;
typedef const _OutputArray& OutputArray;
```

  - Mat, vector<T> 등 다양한 타입을 표현할 수 있는 인터페이스 클래스

코드 3-17 InputArray 클래스를 이용한 사용자 함수 정의 [ch03/InputArrayOp]

```
01  void InputArrayOp()
02  {
03      uchar data1[] = { 1, 2, 3, 4, 5, 6 };
04      Mat mat1(2, 3, CV_8U, data1);
05      printMat(mat1);
06
07      vector<float> vec1 = { 1.2f, 3.4f, -2.1f };
08      printMat(vec1);
09  }
10
11  void printMat(InputArray _mat)
12  {
13      Mat mat = _mat.getMat();
14      cout << mat << endl;
15  }
```

(출력)

```
[  1,   2,   3;
   4,   5,   6]
[1.2, 3.4000001, -2.0999999]
```