Ch03 OpenCV 주요 클래스

3.1 기본 자료형 클래스

3.1.1 Point_ 클래스

; 2차월 평면 위에 점의 좌표를 표현하는 템플릿 클래스.

- X와 y라는 멤버 변수를 가짐.
- Point::dot() : 두 점 사이의 내적(dot product)를 반환
- Point::ddot() : 두 점 사이의 내적(dot product)를 실수형으로 계산, double 자료형으로 반환
- Point::cross() : 두 점사이의 외적(cross product)를 반환
- Point::inside() : 점의 좌표가 사각형 r 영역 내에 위치하면 true를 반환.

- Point_ 클래스는 템플릿 클래스이므로 사용시 어떤 자료형으로 좌표를 표현할 것인지를 명시 해야함.

  Ex) Point_<int> : 정수 좌표를 표현하기 위한 Point_ 클래스 명시

- 하지만, OpenCV에서는 자주 사용하는 자료형에 대해 Point_ 클래스 이름을 재정의하여 제공함.
  - Int 자료형 좌표 표현 : Point2i 클래스
  - Float 자료형 좌표 표현 : Point2f 클래스
  - Double 자료형 좌표 표현 : Point2d 클래스
  - 정수형 좌표 표현 : Point
    - 정수형 좌표를 표현하는 경우가 많기 때문에 정수형 좌표 클래스를 좀 더 일반적으로 Point 클래스로 이름을 다시 정의하여 제공함.
    - 즉, OpenCV에서 Point 클래스는 2차원 정수 좌표계에서 좌표를 표현하는 자료형.

[Point_ 클래스 변수의 생성]

1) 기본 생성자
   - Ex) Point pt1;
   - 아무런 추가 구문없이 선언. X=0, y=0으로 초기화됨.
2) (_x, _y)좌표를 인자로 받는 생성자.
   - Ex) Point pt2(10, 30);
   - 선언할 변수 옆에 괄호로 초기화할 좌표를 전달. X=_x, y=_y로 초기화됨.
3) 복사 생성자.
   - Ex) Point pt3 = pt1+pt2;
   - 다른 Point_ 클래스 변수의 좌표를 복사하여 초기화됨.
   - X=pt.x, y=pt.y로 초기화.

[Point_ 클래스 변수 활용]

- Point_ 클래스는 다양한 연산자에 대해 연산자 재정의가 되어있어 다양한 좌표 연산 가능
  - Point pt3 = pt1 + pt2;
    - 덧셈 연산 : x좌표와 y좌표를 각각 더하여 새로운 좌표 생성.
  - Point pt4 = pt1 * 2;
    - 곱셈 연산 : x, y좌표에 각각 정수를 곱한 좌표 생성.
- Point::dot() 함수로 두 점의 내적을 계산 가능
  - Int d1 = pt1.dot(pt2);
    - Pt1과 pt2의 내적의 결과를 반환
- Point 객체 끼리 == 또는 != 연산자를 이용하여 같은 지 다른 지 검사 가능.
  - Bool b1 = (pt1 == pt2)
- C++ 표준 출력 지원. Std::cout과 <<연산자를 이용하여 Point_객체 좌표 출력 가능.
  - Cout << "pt1: " << pt1 << endl;

3.1.2 Size_ 클래스

; 영상 또는 사각형 영역의 크기를 표현할 때 사용.

- 사각형 영역의 가로와 세로 크기를 나타내는 width와 height 멤버 변수를 가짐.
- Size::area() : 사각형 크기에 해당하는 면적(width x height)를 반환
- Size::empty() : 유효하지 않을 크기이면 true를 반환.

- Size_ 클래스는 템플릿 클래스로 정의되어 있음.
- 따라서 다양한 자료형에 대해 이름이 재정의 되어있음.
  - Int 자료형 표현 : Size2i 클래스
  - Float 자료형 표현 : Size2f 클래스
  - Double 자료형 표현 : Size2d 클래스
  - 정수형 크기 표현 : Size 클래스
    - 즉, Size클래스는 정수형 멤버 변수 width와 height를 가지는 사각형 크기 표현 클래스.

3.1.3 Rect_ 클래스

; 사각형의 위치와 크기 정보를 표현할 때 사용.

- 좌측 상단 점의 좌표를 나타내는 x, y 멤버 변수 와
- 사각형의 가로 및 세로 크기를 나타내는 width, height 멤버 변수를 가지고 있음.
- Rect::tl() : 사각형의 좌측 상단 점의 좌표 반환
- Rect::br() : 사각형의 우측 하단 점의 좌표를 반환
- Rect::area() : 사각형의 면적(width x height)을 반환
- Rect::empty() : 유효하지 않은 사각형이면 true 반환
- Rect::contains() : 인자로 전달된 pt점이 사각형 내부에 있으면 true 반환.

- Rect_ 클래스는 템플릿 클래스 -> 다양한 자료형에 대해 이름이 재정의

- Int 자료형 : Rect2i 클래스
- Float 자료형 : Rect2f 클래스
- Double 자료형 : Rect2d 클래스
- 정수형 사각형 표현 : Rect 클래스
  - 즉, Rect 클래스는 정수형 멤버 변수 x, y, width, height를 가진 사각형 표현 클래스.

[Rect_ 클래스 활용]

- Size_ 클래스 또는 Point_ 클래스 객체와의 산술 연산자 재정의
  - Rect rc3 = rc1 + Size(50, 40);
  - Rect_ 객체와 Size_ 객체를 서로 더하면 사각형의 가로와 세로 크기가 변경됨.
  - Rect rc4 = rc2 + Point(10, 10);
  - Rect_ 객체와 Point_ 객체를 더하거나 빼면 사각형 위치가 변경됨.
- 크기(width, height)가 모두 0인 사각형은 유효하지 않은 사각형임.
  - Rect::empty()의 반환 값이 true임.
- Rect_ 객체끼리 서로 &, | 연산자를 이용한 논리 연산 수행 가능
  - 두 개의 사각형 객체끼리 & 연산 : 두 사각형이 교차하는 사각형 영역
  - 두 개의 사각형 객체끼리 | 연산 : 두 사각형을 모두 포함하는 최소 크기 사각형.

3.1.4 RotatedRect 클래스

; 회전된 사각형을 표현하는 클래스

- 사각형의 중심 좌표를 나타내는 center,
- 사각형의 가로 및 세로크기를 나타내는 size,
- 회전 각도 정보를 나타내는 angle 을 멤버 변수로 가짐.
- RotateRect::points() : 회전된 사각형의 네 꼭지점 좌표를 pts 인자에 저장.
- RotateRect::boundingRect() : 회전된 사각형을 포함하는 최소 크기의 사각형 정보를 반환

(정수단위)

- RotateRect::boundingRect2f() : 회전된 사각형을 포함하는 최소 크기의 사각형정보를 반환 (실수 단위)

- RotateRect는 Point_, Size_, Rect_ 클래스와 달리 템플릿 클래스가 아님.
- 모든 정보를 flaot 자료형으로 표현.
  - 중심점 좌표 : Point2f 클래스 사용
  - 크기 정보 : Size2f 클래스 사용.
  - 회전각도 : float 자료형 사용

[RotateRect 클래스 활용]

- 중심 좌표가 (40,30), 크기는 40x20, 시계 방향으로 30°만큼 회전된 사각형 객체 생성
  - RotateRect rr1(Point2f(40, 30), Size2f(40, 20), 30.f);
- 회전된 사각형 객체의 네 꼭지점 좌표를 알고 싶다면
  - RotateRect::points() 함수 사용
  - 이 함수에 Point2f 자료형의 배열 이름을 전달
    - Point2f pts[4]
    - Rr1.points(pts);
  - 코드 실행이 회전된 사각형의 네 꼭지점 좌표가 pts 배열에 저장됨.
  - 이 함수는 사각형의 좌측 하단 꼭지점부터 시계 방향으로 꼭지점 좌표를 추출함.
- 회전된 사각형을 감싸는 최소 크기의 사각형 정보 -> RotateRect::boundingRect() 함수 이용.
  - Rect br = rr1.boundingRect(); -> 반환 자료형은 Rect 형
  - 실수 단위의 좌표를 알고 싶다면 RotateRect::boundingRect2f() 함수 사용.

3.1.5 Range 클래스

; 범위 또는 구간을 표현하는 클래스

- 범위의 시작과 끝을 나타내는 start와 end 멤버 변수를 가짐.
- Range::size() : 범위 크기(end – start)를 반환
- Range::empty() : start와 end가 같으면 true를 반환
- Range::all() : start=INT_MIN, end = INT_MAX로 설정한 Range 객체를 반환

- start부터 end까지의 정수 단위 범위를 표현.
- Start는 포함되며, end는 포함되지 않음.

3.1.6 String 클래스

[String 클래스 활용]

- 이중 따옴표로 감싸진 문자열로부터 간단하게 생성할 수 있음.
  - String str1 = "Hello";
- 덧셈 연산자를 이용하여 여러 문자열을 이어서 하나의 문자열을 만들 수 있음.
  - String str3 = str1 + " " + str2;
- 두 문자열 객체의 내용 비교
  - Std::string::compare 함수 또는 == 연산자 재정의를 사용할 수 있음.
    - Bool ret = (str2 == "WORLD");
    - 이때, == 연산자는 대 소문자를 구분함.
- 특정한 형식의 문자열 만들고 싶다면 format() 함수 사용

String format(const char* fmt, ...);
• fmt : 형식 문자열 • ... : 가변인자. • 반환값 : 지정한 형식으로 생성된 문자열

- 사용법은 C언어의 printf() 함수와 비슷함.

```
Mat imgs[3];
For(int i=0; i<3; i++) {
        String filename = format("test%02d.bpm", i+1);
        Imgs[i] = imread(filename);
}
```

3.3 Vect과 Scaler 클래스

3.3.1 Vec 클래스

Matx 클래스 -> 작은 크기의 행렬을 표현하기 위한 템플릿 클래스

- 원소 데이터를 val이라는 배열에 저장

Vec 클래스 -> Matx 클래스를 상속, 열 개수가 1개로 특화된 벡터 표현 클래스

- Vec 클래스도 템플릿 클래스 -> <>안에 자료형과 데이터 개수를 명시해야 함.
- Ex) 정수형 데이터 4개 : Vec<int, 4>
- Ex) uchar 자료형 3개 : Vec<uchar, 3>
  - Vec<uchar, 3>은 3채널 컬러 영상 픽셀 값 표현에 주로 사용됨.
  - Vec<uchar, 3> 형식의 변수 p1, p2를 선언하는 예제코드
    - Vec<uchar, 3> p1, p2(0, 0, 255)
- 매번 Vec<uchar, 3> 형태로 입력하기 힘듦 -> 이름 재정의 제공.
  - Vec<num-of-data>{b|s|w|i|f|w}
    - <num-of-data> : 2, 3, 4등 작은 숫자 지정
    - {b|s|w|i|f|w}
      - B ; uchar
      - S : ushort
      - I : int

        - F : float
        - D : double
    - Ex) Vec2b, Vec3s, Vec4i ….

- 따라서 컬러 영상의 픽셀 값을 표현하고 싶을 때
    - Vec<uchar, 3> => Vec3b 사용
        - Vec2b p1, p2(0, 0, 255);
        - 이와 같이 변수 선언하면 p1, p2에는 각각 내부에 uchar val[3]; 형식의 멤버 변수를 가지고 있음.
        - P1의 경우 p1.val 배열 원소가 모두 0으로 초기화되고,
        - P2의 경우 p2.val[0] = 0, p2.val[1] = 0, p2.val[2] = 255로 초기화 됨.
        - P1의 첫번째 원소를 100으로 변경하고 싶다면 아래와 같이 코드작성
            - P1.val[0] = 100;
- 외에도 다양한 멤버 함수와 연산자 재정의를 제공함.