실습 3주차 숙제

- 매칭 결과 그리기
  - 이번 숙제에서는 함수 MatchDescriptor( )에서 매칭된 keypoint 들을 서로 연결하여 보이는 프로그램을 작성한다.
  - 이를 위하여 먼저 게시한 swLab19f_3_SIFT_HW_Std.zip의 압축을 푼다.
  - 폴더에는 이미 세팅이 완료된 VS 프로젝트가 있는데, 여기서 파일 snnnnnnL03HW.cpp를 찾을 수 있을 것이다.
  - 이 파일의 이름에서 nnnnnn을 자신의 학번 뒤 6자리로 바꾸고, 이에 따라 솔루션 탐색기를 조정한다.
  - snnnnnnL03HW.cpp의 함수를 프로그래밍하여 해결, 테스트한 후, 사이버 캠퍼스의 해당 제출함에 이 파일만 기한 내 제출한다.
  - Late 제출은 허용하지 않는다.

파일 snnnnnnH03.cpp

- 이 파일에는 다음과 같이 세 개의 함수가 있다.

```
int MatchDescriptor(const Mat &descriptor1,
    const Mat &descriptor2, vector<int> &matchingIdx);
  // Keypoint 매칭 함수로 실습 시간에 작성한 코드를
  // 그대로 사용한다.

Mat stack_imgs(const Mat &im1, const Mat &im2);
  // 입력 이미지 im1과 im2을 좌우 수평으로 연결하여 하나의
  // 이미지로 만들어 반환하는 함수(작성한다)

Mat DrawMatches(const Mat &im1, const Mat &im2,
      vector<KeyPoint> &keypoints1,
      vector<KeyPoint> &keypoints2,
      vector<int> &matchingIdx);
  // 매칭 결과인 matchingIdx에 근거하여 두 이미지 간의
  // 매칭된 keypoint들을 서로 연결하는 함수(작성한다)
  // 이 함수 내에서 함수 stack_imgs( )를 호출한다
```

함수 작성에 필요한 SIFT 함수 1

Mat Initializer

```
cv::Mat::zeros(int rows, int cols, int type);
  // 모두 0으로 초기화한 Mat 배열 생성(all black)
cv::Mat::ones(int rows, int cols, int type);
  // 모두 1(all white)
```

Mat 복사

- 복사는 pointer만 복사하는 shallow copy와 메모리를 다시 할당 받아 내용을 복사하는 deep copy가 있는데, 여기서는 deep copy를 사용한다.

```
void cv::Mat::copyTo(OutputArray m) const  // (1,2)
```

복사 예:

```
im.copyTo(copy); // im 데이터를 copy로 복사
```

(1) https://m.blog.naver.com/PostView.nhn?blogId=kkspp&logNo=220898679507&navType=tl

(2) const는 인수를 함수 내에서 수정할 수 없다는 의미이다.

✦ 클래스 Rect (1,2,3)

- ◆ 이를 사용하여 이미지의 특정 영역을 선택할 수 있다.
- ◆ Instance 설정

```
Rect_ (_Tp _x, _Tp _y, _Tp _width, _Tp _height)
  // x,y : 좌상귀 좌표,
  // width : 영역의 너비, height : 영역의 높이
```

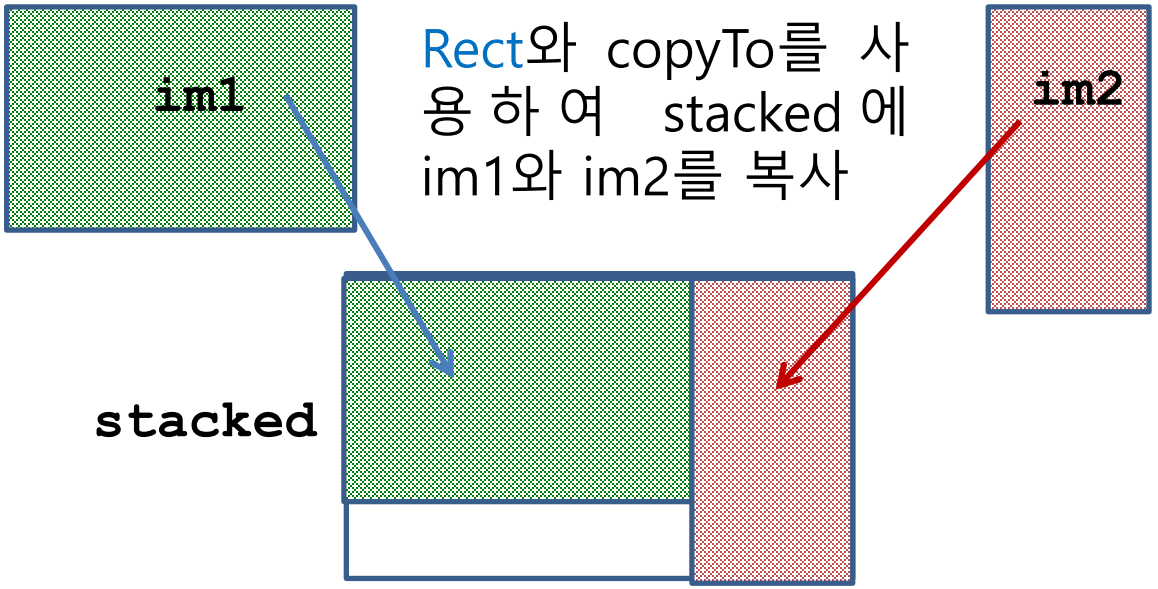
- ◆ 예: 이미지 im에서의 특정 영역 선택

```
im(Rect(x, y, width, height));
```

(1) https://docs.opencv.org/4.1.1/d2/d44/classcv_1_1Rect__.html
(2) #include <opencv2/core/types.hpp> 필요
(3) Rect는 Rect_의 typedef

함수 **stack_imgs( )** 작성

- 두 이미지를 합쳐 **stacked**라는 Mat 배열에 저장한다.
- 합칠 이미지가 **im1**, **im2**이므로 stacked의 폭은 **im1**과 **im2**의 폭을 합친 것과 같아야 하고, 높이는 **im1**과 **im2** 높이 중 큰 값이어야 한다.
- 아래 그림과 같이 이미지를 합칠 것이므로 앞에서 소개한 **copyTo**와 **Rect**를 이용하여 **im1**과 **im2**를 **stacked**에 복사한다.



함수 작성에 필요한 SIFT 함수 2

- 두 가지 도형 그리는 함수를 사용한다

```
void circle(Mat& img, Point center, int radius,
      const Scalar& color, int thickness=1,
      int lineType=8, int shift=0);
   // img : 원을 그릴 이미지
   // center : 원의 중점(x, y),  radius : 반지름
   // color : 원의 색(컬러면 Scalar(B,G,R)
   //                        B,G,R은 0 ~ 255사이의 정수값),
   //              (그레이 스케일이면 0 ~ 255사이의 정수 값)
   // thickness : 굵기 (기본 값 1, -1이면 원 안쪽을 채움)
   // lineType : 선 종류(선을 표현하는 거칠기 정도)
   //    8(or omitted): 8-connected line.
   //    4: 4-connected line, CV_AA: antialiased line.
   // shift : 좌표에 대한 비트 시프트 연산(기본값 0)
```

✦도형 그리는 함수(계속)

```
void line(Mat& img, Point pt1, Point pt2,
      const Scalar& color, int thickness=1,
      int lineType=8, int shift=0)
  // img : 선을 그릴 이미지
  // pt1, pt2  : 선의 시작점과 끝점
  // thickness : 선 굵기 (기본 값 1)
  // color, lineType, shift : circle과 동일
```

함수 DrawMatches( ) 작성

- 두 이미지를 합친 이미지를 얻었으면
  - 매칭된 keypoint 위치에 원을 그려 표시하고,
  - 매칭된 두 keypoint간에 선으로 연결한다.
  - 이때, im2의 어떤 keypoint 좌표가 (x, y)라면, 이 좌표을 좌측으로 img1.cols 만큼 수평이동하여야 한다.
  - 아래 최종적으로 출력된 이미지 쌍을 보인다.

```
D:₩Test>swLab19f_3_SIFT_HWD box.bmp box3_30.png
Image 1 size :[324 x 223], Type:8UC3
Image 2 size :[217 x 238], Type:8UC3
IMG1 Num kpt : 604
IMG2 Num kpt : 320
matches:137
```