KANGWON NATIONAL UNIVERSITY

컴퓨터비전 실습

실습9 | Feature Distance

실습과제 이루리 내 제출

CVMIPLAB @ KNU

실습 9-1 | Feature Distance

문제

주어진 코드를 활용하여 “florence-1.bmp”, “florence-2.bmp” 파일을 각각 **흑백으로** 읽은 뒤,

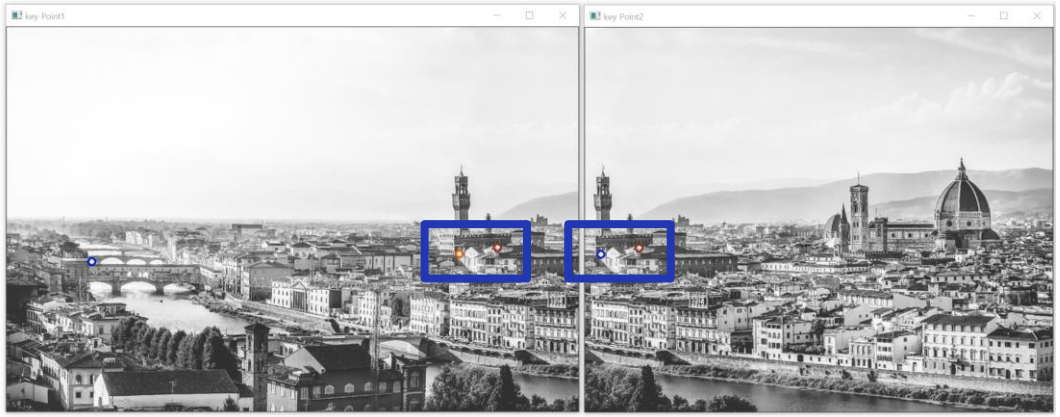
SIFT 기술자를 추출한 뒤 특징 벡터들의 Euclidean Distance를 구합니다.

요구 결과

Euclidean Distance를 구한 결과 이미지 두 장을 각각

화면에 표시 후 “florence-1-fv.bmp”, “florence-2-fv.bmp”로 저장합니다.

저장된 두 영상과 “w10-1.cpp” 총 세 개의 파일을 압축하여 제출합니다.



실습 9-1 | Feature Distance

설명자료

다음은 Euclidean Distance (L2-Distance)의 공식입니다.

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

실습 9-1 | Feature Distance

설명자료

아래 빈칸을 채워 Euclidean Distance 계산식을 구현하세요.

```
// Euclidean Distance (L2 Distance)를 구한다.
// 실습 PPT 3페이지의 수식을 참고하여 구현하도록 한다.
// ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **




// ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
```

실습 9-1 | Feature Distance

설명자료

아래 빈칸을 채워 Scale를 구하세요.

```
// 특징벡터 2개를 비교하여 구한 Euclidean Distance가
// 1.0보다 작을 경우에만 distance_list에 추가하도록 한다.
// 모든 값을 넣고, 추후에 이 distance_list를 distance순으로
// 오름차순 정렬하여 출력해보도록 한다.
if (distance < 1.0) {

    // 현재 키포인트의 Octave의 제곱에 해당하는 Scale을 구한다.
    // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **


    // ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **

    // 구한 Keypoint 리스트에서 key_i(첫번째 이미지), key_j(두번째 이미지)
    // 각 인덱스에 해당하는 키포인트로부터 좌표를 구하여 이를 firstImgPtr와 secondImgPtr에 저장한다.
    cv::Point firstImgPtr = cv::Point(keypoints_first[key_i].x, keypoints_first[key_i].y) * scale_first;
    cv::Point secondImgPtr = cv::Point(keypoints_second[key_j].x, keypoints_second[key_j].y) * scale_second;
```

실습 9-1 | Feature Distance

설명자료

아래 빈칸을 채워 Euclidean Distance 계산식을 구현하세요.

```
// Distance가 짧은 순으로,
// 상위 3개만 원을 그려서 출력한다.
for (int i = 0; i < distance_pair.size() && i < 3; i++)
{
    PointSet& point_set = distance_pair[i];

    std::cout << "PointSet #" + std::to_string(i + 1) << "\tDistance: " << point_set.distance << std::endl;
    std::cout << "\tFirst Image Point " << point_set.firstImgPtr.x << ", " << point_set.firstImgPtr.y << " (x, y)" << std::endl;
    std::cout << "\tSecond Image Point " << point_set.secondImgPtr.x << ", " << point_set.secondImgPtr.y << " (x, y)" << std::endl;

    // point_set으로부터 좌표를 받아와 first_img_output과 second_img_output에
    // 각각 원을 그린다. 색상은 페어를 구분하기 위해 상단의 color 변수를
    // 첫번째 이미지와 두번째 이미지에 동일하게 이용하도록 한다.
    //
    //     void circle(InputOutputArray img, Point center, int radius,
    //                 const Scalar& color, int thickness = 1,
    //                 int lineType = LINE_8, int shift = 0);
    //
    // 상단의 함수를 이용하여 구현하도록 한다. (또는 sift.cpp의 DrawKeyPoints 참조)
    // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **



    // ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
}
```

실습 9-1 | Feature Distance

결과영상 (원의 색상은 다를 수 있음)