메디치소프트 기술연구소

CH43. OpenCV: 이미지 특성 매칭하기

MEDICISOFT

43. 이미지 특성 매칭하기

- 43.1 테스트 환경

순 번	제 목	설치 버전
1	운영체제	Windows 10 64bit
2	프로그래밍언어	Python3.6.5

▪ 43.2 [43-01 example.py] - ORB를 이용한 BF 매칭

- 두 개의 이미지에서 특성을 비교하면 두 이미지가 얼마나 비슷한 형태를 띄고 있는지, 또는 유사도가 얼마인지를 추측할 수 있다. 더 나아가 두 이미지에서 동일한 이미지 부분을 추출할 수도 있다.

- 두 이미지의 특성을 비교하는 가장 단순한 방법은 전수조사 방법으로 매칭(Brute-Force 매칭, 줄여서 BF 매칭)하는 것이다.

- 이미지 A, B가 있다고 하면, 이미지 A에서 하나의 특성 디스크립터를 취하고, 이 디스크립터를 이미지 B의 모든 특성 디스크립터와 거리 계산 방법을 이용해 하나하나 비교한다.
  이렇게 해서 나온 결과중 가장 비슷한 값을 리턴한다.
  이와 같은 방법으로 이미지 A의 모든 특성 디스크립터에 대해 계산한다.

▪ 43.2 [43-01 example.py] - ORB를 이용한 BF 매칭

- OpenCV는 BF 매칭을 제공하는 cv2.BFMatcher()라는 메쏘드를 제공한다.
  이 메쏘드는 두 개의 인자를 취할 수 있는데, 하나는 normType이고 또 다른 하나는 crossCheck 이다.
- normType은 BF 매칭에 사용할 거리 계산 방법을 지정한다.
  보통 SIFT나 SURF인 경우 디폴트로 cv2.NORM_L2를 사용한다.

- crossCheck는 True 또는 False 값이 지정되며, 디폴트는 False이다.
  crossCheck가 True로 설정되면 두 이미지를 서로 BF 매칭하여 가장 일치하는 부분만을 리턴한다.

▪ 43.2 [43-01 example.py] - ORB를 이용한 BF 매칭

- 다음은 43-01 example.py 예제의 code이다.

```
import numpy as numpy
import cv2

def featureMatching():
    img1 = cv2.imread('images/mybook1.jpg', cv2.IMREAD_GRAYSCALE)
    img2 = cv2.imread('images/mybook1-1.jpg', cv2.IMREAD_GRAYSCALE)
    res = None

    orb = cv2.ORB_create()
    kp1, des1 = orb.detectAndCompute(img1, None)
    kp2, des2 = orb.detectAndCompute(img2, None)
```

▪ 43.2 [43-01 example.py] - ORB를 이용한 BF 매칭

- 다음은 43-01 example.py 예제의 code이다.

```
    bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
    matches = bf.match(des1, des2)

    matches = sorted(matches, key = lambda x:x.distance)
    res = cv2.drawMatches(img1, kp1, img2, kp2, matches[:30], res, flags=0)

    cv2.imshow('feature Matching', res)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

featureMatching()
```

▪ 43.2 [43-01 example.py] - ORB를 이용한 BF 매칭

>> LINE 9~11)

```
orb = cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)
```

- ORB 객체를 생성하고, 이미지1과 이미지2의 키포인트들과 디스크립터를 계산한다.

>> LINE 13~14)

```
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(des1, des2)
```

- Brute-Force 매칭 객체를 normType의 값을 cv2.NORM_HAMMING, crossCheck 값을 True로 설정하고 생성한다.
  그리고 두 이미지의 디스크립터들을 매칭하여 그 결과를 matches 변수에 담는다.

▪ 43.2 [43-01 example.py] - ORB를 이용한 BF 매칭

>> LINE 16~17)

```
matches = sorted(matches, key=lambda x:x.distance)
res = cv2.drawMatches(img1, kp1, img2, kp2, matches[:30], res, flags=0)
```

- matches의 요소들을 x.distance의 값으로 정렬한다. 이는 결국 두 이미지의 특성 포인트들을 가장 일치하는 순서대로 정렬하는 것이다.
- drawMatches()를 이용해 matches 멤버에서 처음 30개만 화면에 출력한다.
  flags =0 으로 설정한 것은 매칭 여부와 관계없이 두 이미지에서 찾은 특성 포인트들을 모두 화면에 표시하라는 의미이다.

▪ 43.3 [43-02 example.py] - SIFT를 이용한 BF 매칭

- 다음은 43-02 example.py 예제의 code이다.

```
import numpy as numpy
import cv2

def featureMatching():
    img1 = cv2.imread('images/mybook1.jpg', cv2.IMREAD_GRAYSCALE)
    img2 = cv2.imread('images/mybook1-1.jpg', cv2.IMREAD_GRAYSCALE)
    res = None

    sift = cv2.xfeatures2d.SIFT_create()
    kp1, des1 = sift.detectAndCompute(img1, None)
    kp2, des2 = sift.detectAndCompute(img2, None)
```

▪ 43.3 [43-02 example.py] - SIFT를 이용한 BF 매칭

- 다음은 43-02 example.py 예제의 code이다.

```
    bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
    matches = bf.match(des1, des2)

    matches = sorted(matches, key = lambda x:x.distance)
    res = cv2.drawMatches(img1, kp1, img2, kp2, matches[:30], res, flags=0)

    cv2.imshow('feature Matching', res)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

featureMatching()
```

▪ 43.3 [43-02 example.py] - SIFT를 이용한 BF 매칭

>> LINE 13) **bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)**

- SIFT는 Brute-Force 매칭 객체를 normType의 값으로 cv2.NORM_L2로 설정하고 생성해야 좋은 결과를 도출할 수 있다.