summary

1. 코너검출: `cornerHarris()`
2. 크기 불변 특징점 검출과 기술: `detect()`, `compute()`, `detectAndCompute()`
3. 특징점 매칭: `DescriptorMatcher클래스`, `BFMatcher클래스` `FlannBasedMathcer클래스`
4. 영상 이어 붙이기: `stitch()`

1. 코너검출

1. 특징이란?
    - 영상으로부터 추출할 수 있는 유용한 정보
    - 예를들면, 평균 밝기, 히스토그램, 에지, 직선성분, 코너 등
    - 지역특징: 일부 영역에서 추출할 수 있는 특징 ↔ 전역특징
2. 코너
    - 에지의 방향이 급격하게 변하는 부분
    - 다른 지역 특징에 비해 분별력이 높고, 골고루 분포해서 영상 분석에 유용함
3. 특징점
    - 한 점의 형태로 표현할 수 있는 특징
    - 키포인트 또는 관심점이라고 함
4. 해리스 코너 검출 방법
    - 특정 위치에서 Δx, Δy 만큼 떨어진 픽셀과의 밝기차
    -

$$R = \mathrm{Det}(\mathbf{M}) - k \cdot \mathrm{Tr}(\mathbf{M})^2$$

$$E(\Delta x, \Delta y) = \sum_{x,y} w(x,y)\left[I(x+\Delta x, y+\Delta y) - I(x,y)\right]^2$$

    - E(Δx, Δy) 값이 크면 주변부 픽셀과 차이가 큰것이므로 엣지임 / w = 웨이트 / [...] = 밝기차
    - 위 수식에서 w(x,y) : 가우시안 형태의 가중치를 갖는 윈도우
    - E 값이 모든 방향으로 크게 나타나면 x,y는 코너이고, 주변부 픽셀과 밝기 차가 많이 난다는 의미
        - 테일러 급수와 고유값 분석등으로 코너 응답 함수 R을 유도함

```
det(M)=λ1λ2
trace(M)=λ1+λ2λ1,
λ2 are the eigenvalues of M
```

    - λ1, λ2가 둘다 크면 곱이 상당히 커지고, 합은 곱에 비해서는 조금 커짐

- R이 양수이면: 코너
- R이 0에 가까운 실수면 : 평평한 부분
- R이 음수면 : 엣지
- `cornerHarris()`함수 사용
  - 적절한 임계값 연산을 용하면 코너 찾기 가능
  - 하나의 코너위치에서 임계값보다 큰 픽셀이 여러개 발생함
    - 비최대억제를 수행해 지역 최댓값의 위치만 코너로 판별하는게 best

5. FAST 코너 검출 방법

- 특징
  - 단순한 픽셀 값 비교하여 코너 검출
  - FAST하게 동작하는 코너 검출하는 방법
  - 픽셀을 둘러싸고 있는 16개의 주변 픽셀과 밝기를 비교하여 코너 여부 판별
    - 16개 중 충분히 밝거나 충분히 어두운 픽셀이 9개 연속으로 존재하면 코너로 정의
  - 점 p에서의 밝기는 Ip라고 할 때
    - Ip+t보다 큰 픽셀이 9개 이상 존재 : 어두운 영역이 돌출되어 있는 코너
    - Ip-t보다 작은 픽셀이 9개 이상 존재 : 밝은 영역이 돌출되어 있는 코너
  - 비최대 억제 작업 수행하는 이유: 코너 주변 픽셀도 함께 코너로 검출되는 것을 방지 가능

2. 크기 불변 특징점 검출과 기술

1. 특징

- 코너 = 회전 불변 특징점
  - 단 영상의 크기가 변경되면 코너는 더이상 코너로 검출되지 않을 수도 있음

2. SIFT (크기불변 특징점 알고리즘)

- 알고리즘 설명
  - DoG영상 집합에서 인접한 DoG영상을 고려한 지역 극값 위치 특징점으로 사용함 이후 에지 성분이 강하거나 명암비가 낮은 지점은 특징점에서 제외
- 특징
  - 입력영상으로부터 스케일 스페이스를 구성하여, 크기 변화와 무관하게 특징점 추출이 가능
  - 스케일 스페이스: 다양한 표준편차를 이용한 가우싱안 블러링을 적용해 구성한 영상 집합

그림 14-6 SIFT 특징점 검출을 위한 가우시안 피라미드와 DoG 구성

  - 위 그림의 블러링된 영상 6개가 스케일 스페이스이고, 각 한줄씩을 옥타브라고 부름

        - 이후에 입력영상의 크기를 가로, 세로로 반씩 줄여가며 여러 개의 옥타브를 구성함
        - 옥타브 내에서 DoG(가우시안 블러링 영상 간의 차영상)를 구하며 특징점을 찾음
    - SIFT 특징점 검출
        - 특징점 주변의 픽셀값을 이용한 기술자 계산 방법 포함
            - 특징점 기술자(=특징벡터): 특징점 주변 영상의 특성을 여러개 실수값으로 표현
            - 서로 같은 특징점에서 추출된 기술자는 실수값 구성이 서로 일치해야 함
            - 특징점 근처에서 그래디언트 방향 히스토그램을 추출해 기술자로 사용함
                - 특징점 근방으로부터 특징점의 주된 방향 성분을 계산함
                - 해당 방향만큼 128개의 빈으로 구성된 그래디언트 방향 히스토그램을 계산함
            - 크기, 회전 등의 변환 뿐만 아니라 촬영 시점 변화에도 충분히 강인하게 동작함
    - 활용
        - 객체인식, 파노라마 영상 이어붙이기, 3차원 장면 인식 등..

3. SIFT, SURF, KAZE 방법에 대해

    - 특징
        - 복잡한 연산 수행 때문에 실시간 응용에서는 사용하기 어려움
        - 기술자는 128개 또는 64개의 실수값으로 구성되어 있음
        - 메모리 사용량이 많고 특징점 사이의 거리 계산 오래 걸림
        - 2010년 전후로 특징점 검출이 매우 빠르고 이진수로 구성된 기술자를 사용하는 알고리즘 발표됨
            - 대표적인예: **ORB**

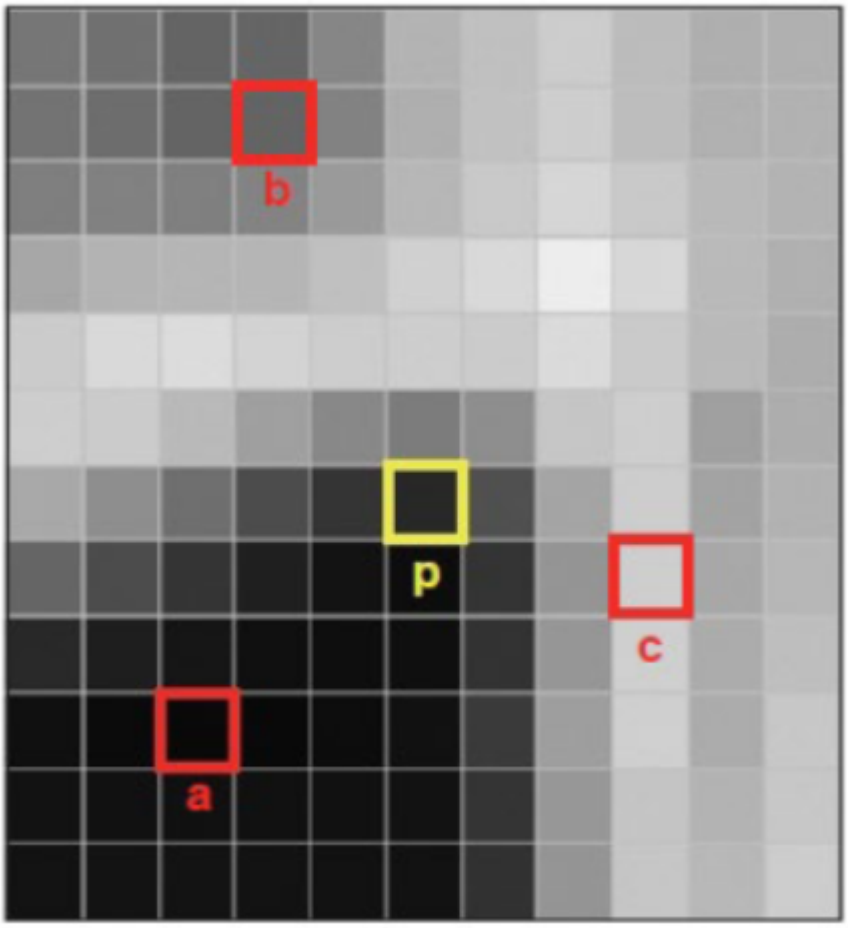
4. ORB

    - 개념: FAST 코너 검출 방법을 이용해 특징점 추출하지만, FAST 크알고리즘 자체가 영상 크기 변화에 취약함 -> 따라서, ORB알고리즘은 입력 영상의 크기를 점진적으로 축소한 피라미드 영상을 구축해 특징점 추출 - 각 특징점에서 주된 방향 성분 계산하고 - 방향을 고려한 BRIEF 알고리즘으로 이진기술자 계산

    - BRIEF

        - 이진 기술자 : 특징점 주변 정보를 이진수 형태로 표현하는 기술
        - 특징점 기술자만을 생성하는 알고리즘
        - 특징점 주변 또는 주변의 픽셀 쌍을 미리 정하고, 해당 픽셀 값 크기를 비교해 0 또는 1로 특징 기술함

    - 예시




```
 1)특징점 주변 픽셀 a, b, c를 미리 정하고
 2) f(a, b), f(b, c), f(c, a) = 110
   a < b
   b < c
   c > a
3) 기본적으로 256개의 크기 비교 픽셀 쌍을 사용해 이진 기술자를 구성
   - 하나의 특징점은 256 비트로 표현 가능
4) SIFT: 512바이트
   SURF : 256바이트
   ORB : 32바이트로 특징점 기술
```

5. 해밍 거리

- 개념: 이진 기술자로 표현된 특징점 사이의 거리 계산은 해밍 거리 방법 사용
  - 두 기술자에서 서로 값이 다른 비트의 개수를 세는 방식
    - XOR 연산 사용, 비트 값이 1인 개수 세는 방식으로 빠르게 계산함

6. 이외에도 brisk, akaze, freak등 다양한 알고리즘 존재함

7. openCV 특징점 검출과 기술

- KeyPoint 클래스

pt: 특징점 좌표	size : 특징점 크기	angle: 특징점의 주된 방향	response: 반응성(좋은 특징점 선발하는 용도)	옥타브

- openCV에서 특징점 관련 클래스는 모두 Feature2D 클래스를 상속받아 만들어짐
  - detect(), compute(), detectAndCompute()라는 가상 멤버 함수를 가짐
    - `dect()`: 영상에서 키포인트 컴출
    - `compute()`: 검출된 키포인트를 표현하는 기술자를 생성
    - `detectAndCompute()`: 키포인트 검출과 기술자 생성을 동시에 수행
    - 특징점만 찾는 알고리즘: compute 사용불가 (FAST)
    - 기술자만 생성하는 알고리즘: detect 사용불가 (BRIEF)
    - SIFT, SURF, KAZA, ORB: detect(), compute(), detectAndCompute()함수를 모두 사용할 수 있음
- `drawKeypoints()`: 특징점 검출한 후 입력 영상이 어느 위치에서 특징점이 검출됐는지 영상위에 직접 표시해 확인함
  - 특징점 좌표만 보이도록 표현, 특징점 검출시 고려한 크기 성분과 주된 방향 성분 함께 표시 가능
  - DrawMatchesFlags 열거형 상수
    - DEFAULT
    - DRAW_OVER_OUTIMG
    - NOT_DRAW_SINGLE_POINTS
    - DRAW_RICE_KEYPOINTS

3. 특징점 매칭

1. 개념

- 두 영상에서 추출한 특징점 기술자를 비교해 서로 비슷한 특징점을 찾는 작업을 의미함
- 크기 불변 특징점으로부터 구한 기술자를 매칭하면 크기와 회전에 강인한 영상 매칭 수행 가능
- DMatch클래스는 한 장의 영상에서 추출한 특징점과 다른 한 장의 영상, 또는 여러 영상에서 추출한 특징점 사이의 매칭 정보를 표현할 수 있음

queryIdx: 질의 기술자 번호	trainIdx: 훈련 기술자 번호	imgIdx: 훈련 영상 번호	distance: 두 기술자 사이의 거리

- distance 계산 방식은 다차원 벡터의 유클리드 거리로 주로 계산함
- 이진 기술자끼리 비교하는 경우 해밍거리 사용
- openCV 특징점 매칭 클래스는 DescriptorMatcher클래스를 상속받아 만들어짐
  - match(): 비슷한 가장 기술자 쌍을 하나 찾음
  - knnMatch(): 비슷한 기술자 쌍 k개를 찾음

- radiusMatch(): 지정 거리 반경 안에 있는 기술자 쌍을 모두 찾아 반환함

2. BFMatcher 클래스

- Brute-Force매칭 수행
- 모든 기술자와 훈련 기술자 집합에 있는 모든 기술자 사이의 거리 계산하여 이 중 가장 거리가 작은 기술자를 찾아 매칭하는 방식
- 특징점 개수가 늘어날수록 매칭 연산량이 크게 늘어나므로, FlannBasedMathcer클래스 사용하는 게 효율적임

3. FlannBasedMathcer 클래스

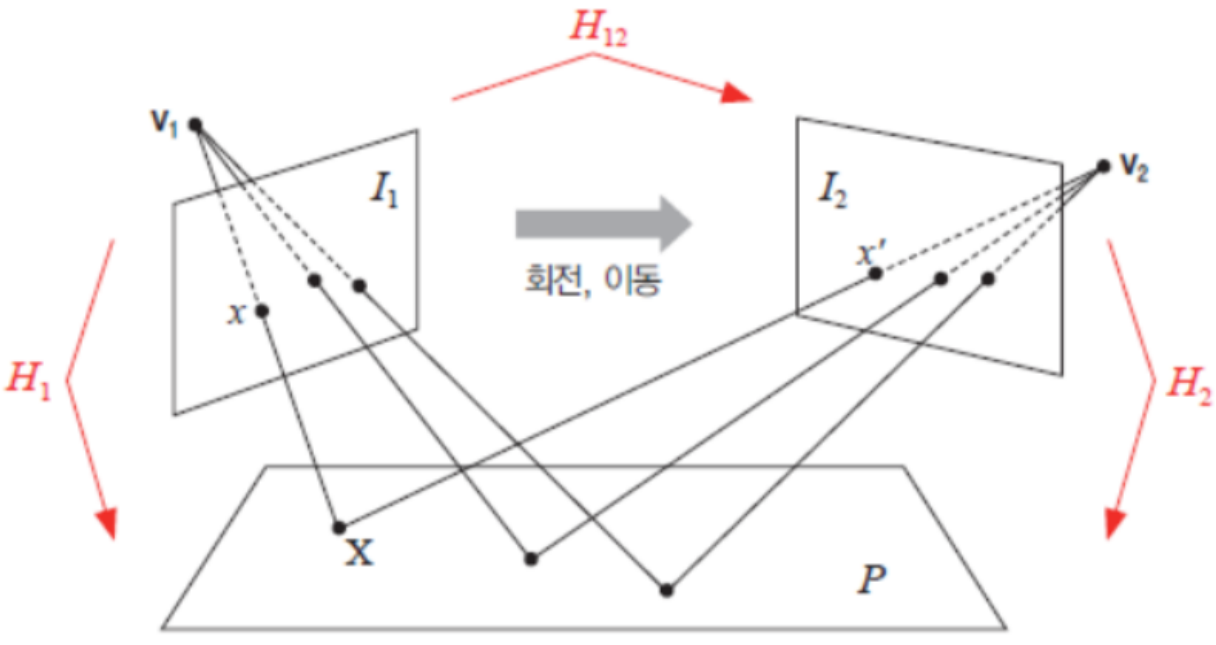
- 근사회된 최근방이웃 알고리즘
- 가장 거리가 작은 특징점을 찾지 못할 수 있지만 매우 빠르게 동작
- L2 norm 거리 측정 방식을 사용해 해밍 거리를 사용하는 이진 기술자에 대해서는 사용할 수 없음

```
 orb = cv.ORB_create()

keypoints1, desc1 = orb.detectAndCompute(img1, None)
keypoints2, desc2 = orb.detectAndCompute(img2, None)

matcher = cv.BFMatcher_create(cv.NORM_HAMMING)
matches = matcher.match(desc1, desc2)
```

4. `drawMatches()`: 매칭 입력 영상을 가로로 이어붙이는 함수

- 각 영상에서 추출한 특징점, 매칭 결과를 랜덤한 다양한 색상으로 표시한 결과 영상 생성

5. 호모그래피 영상 매칭



- 실제 연산 관점에서 볼 때, 투시 변환과 같음

        - 3x3 실수 행렬로 표현
        - 4 개의 점에 대응되는 이동 정보가 있으며 호모그래피 행렬 구하기 가능
        - 특징점 매칭 정보로부터 호모그래피 구하는 경우 서로 대응되는 점의 개수가 4개보다 훨씬 많음
        - 에러 최소화되는 형태의 호모그래피 행렬 구해야 함
        - `findHomograpy()` 함수 사용
            - method로 0, LMEDS, RANSAC, RHO

    - 특징

        - 주요 특징점이 찾아지는 부분만 보존되면 잘 찾을 수 있음
        - 한 개를 찾을 때 유용함

4. 영상 이어 붙이기

1. 개념
    - 여러 장의 영상을 서로 이어붙여 하나의 영상 만드는 것
    - 파노라마 영상이라고 부름
2. 제한점
    - 서로 일정 비율 이상으로 겹치는 영역이 존재해야함
    - 서로 같은 위치를 분간하도록 유효한 특징점이 많이 있어야 함
3. 블렌딩처리
    - 영상의 밝기를 적절히 보정해 파노라마 영상이 자연스러워 보이도록 처리

```
stitcher = cv.Stitcher_create()
status, dst = stitcher.stitch(imgs)

cv.imshow('result', dst) #결과물은 투시 변환이라서 검은부분이 발생하기도함
```