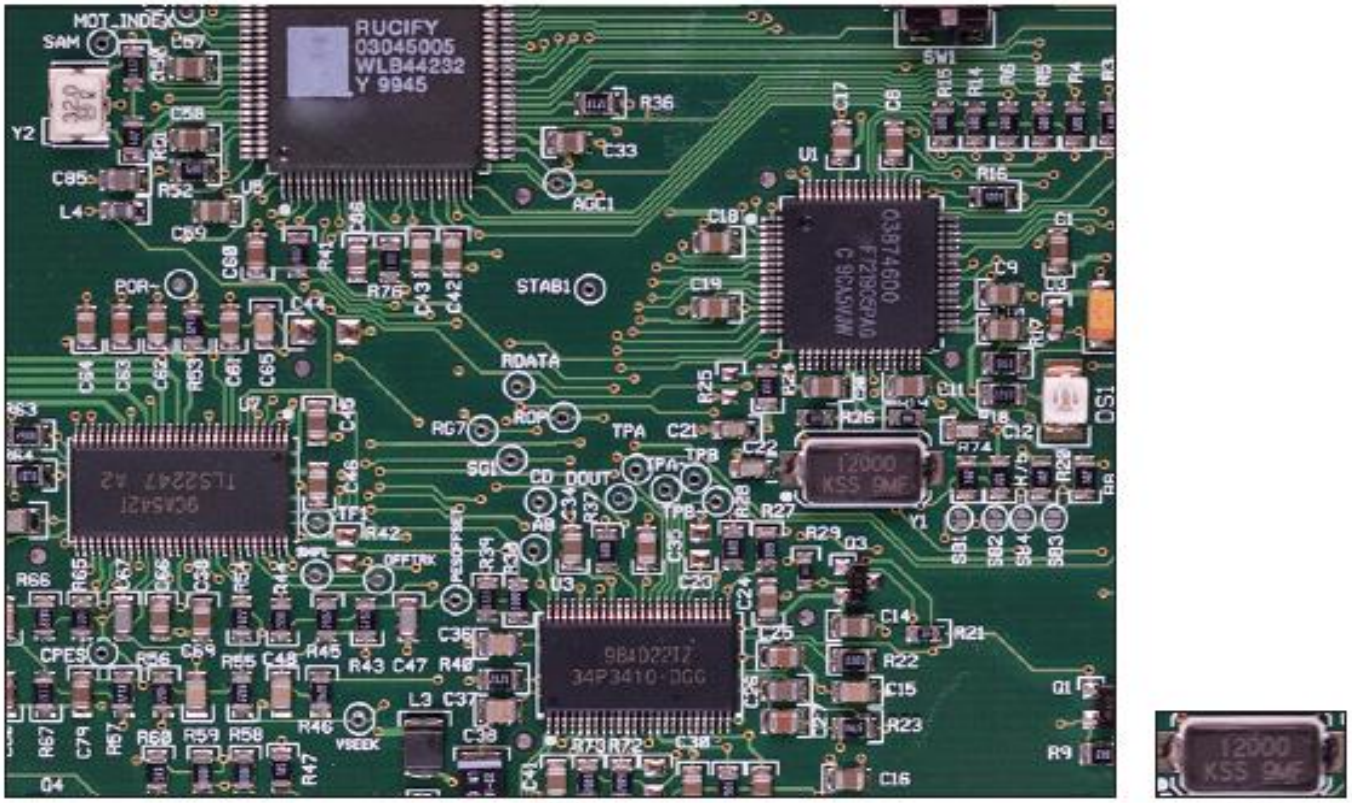
가톨릭대학교
THE CATHOLIC UNIVERSITY OF KOREA

Matching, Stitching

미디어기술콘텐츠학과
강호철

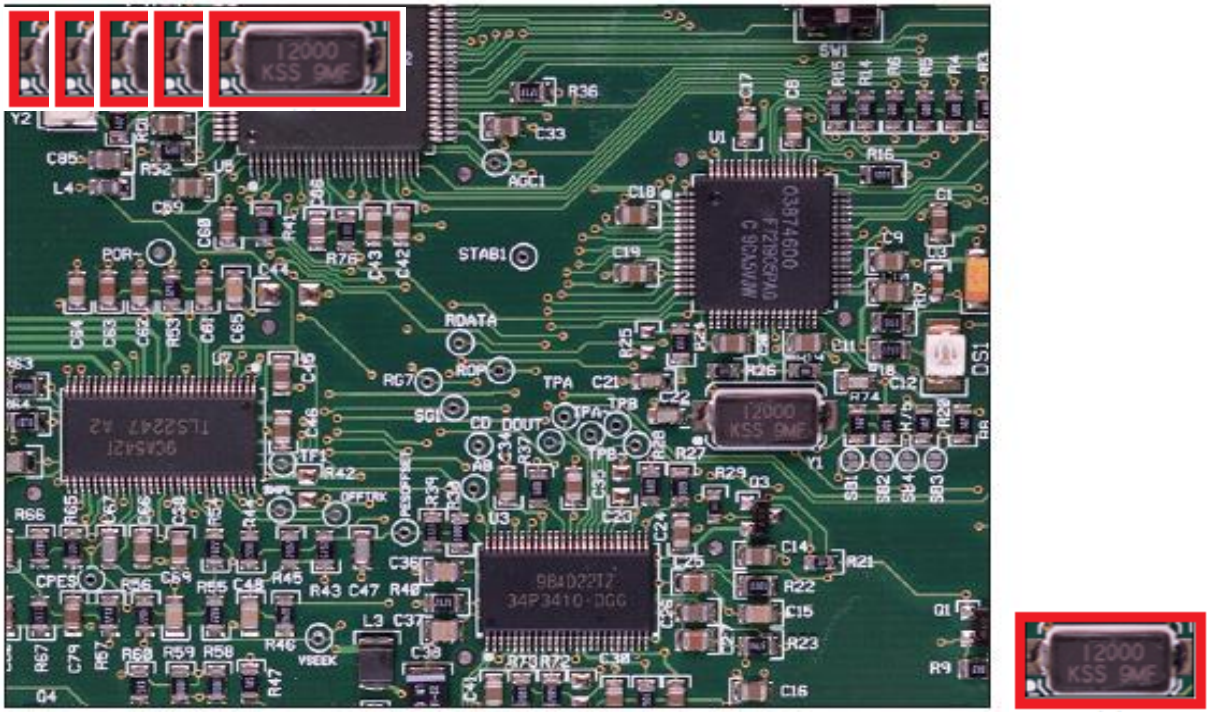
(Review) 매칭

- 객체 찾기
  - 템플릿이 있는 경우



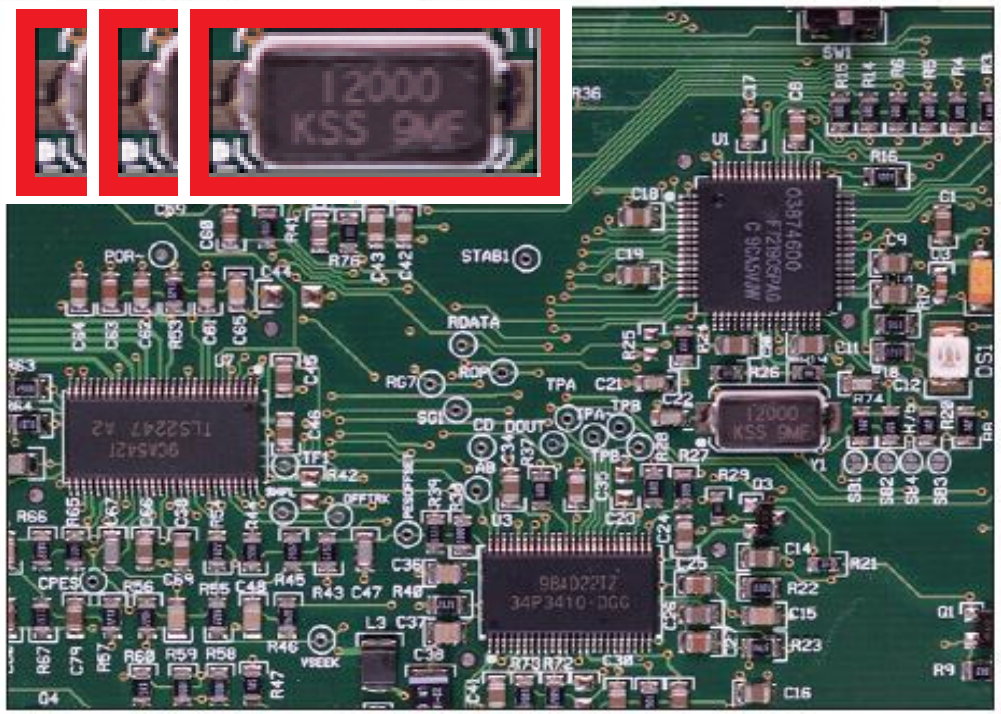
(Review) 매칭

- 객체 찾기
  - 템플릿이 있는 경우
    - Block matching



(Review) 매칭

- 객체 찾기
  - 템플릿이 있는 경우
    - 한계점

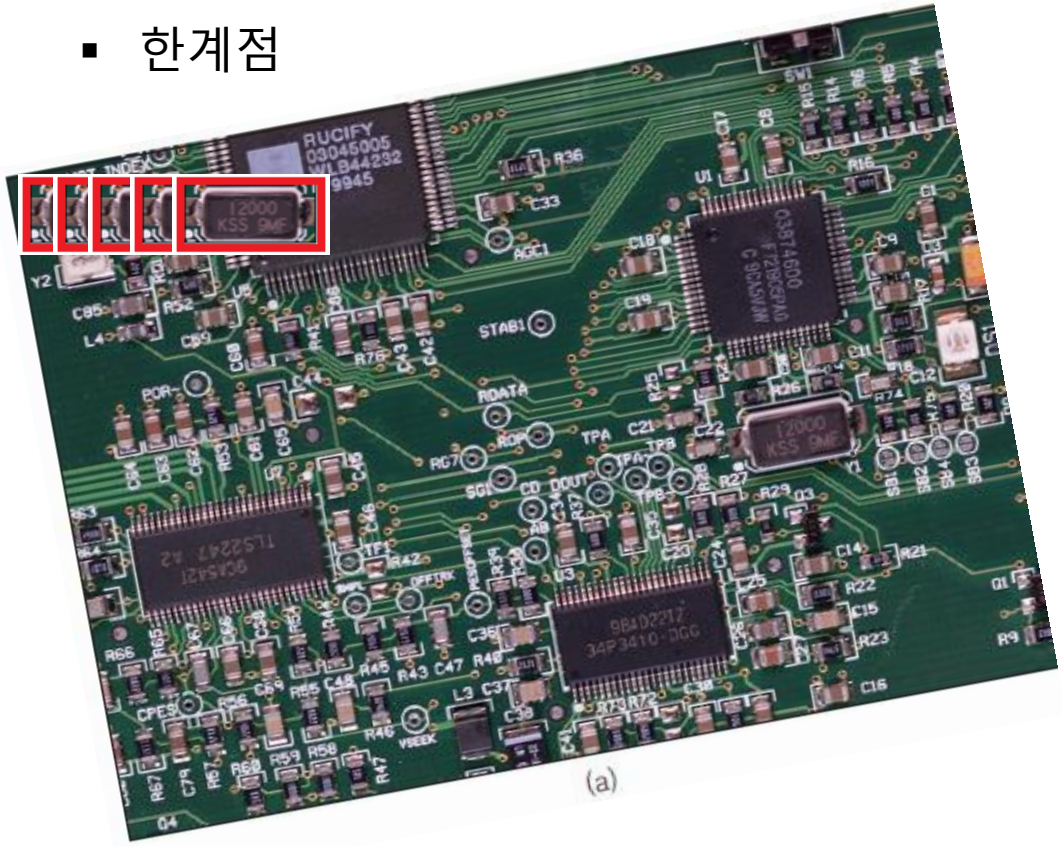


(a)

?

(Review) 매칭

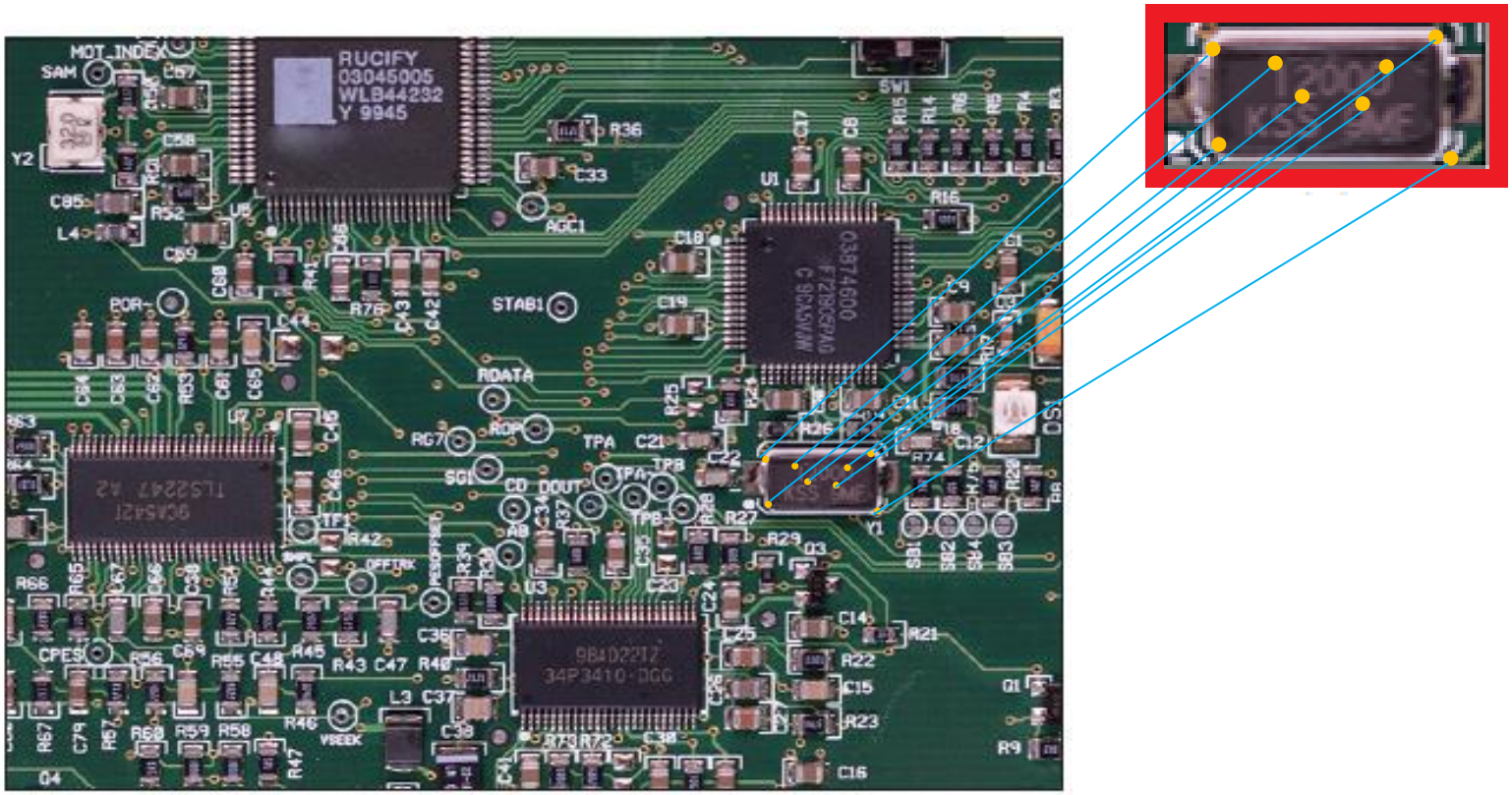
- 객체 찾기
  - 템플릿이 있는 경우
    - 한계점



?

(Review) 매칭

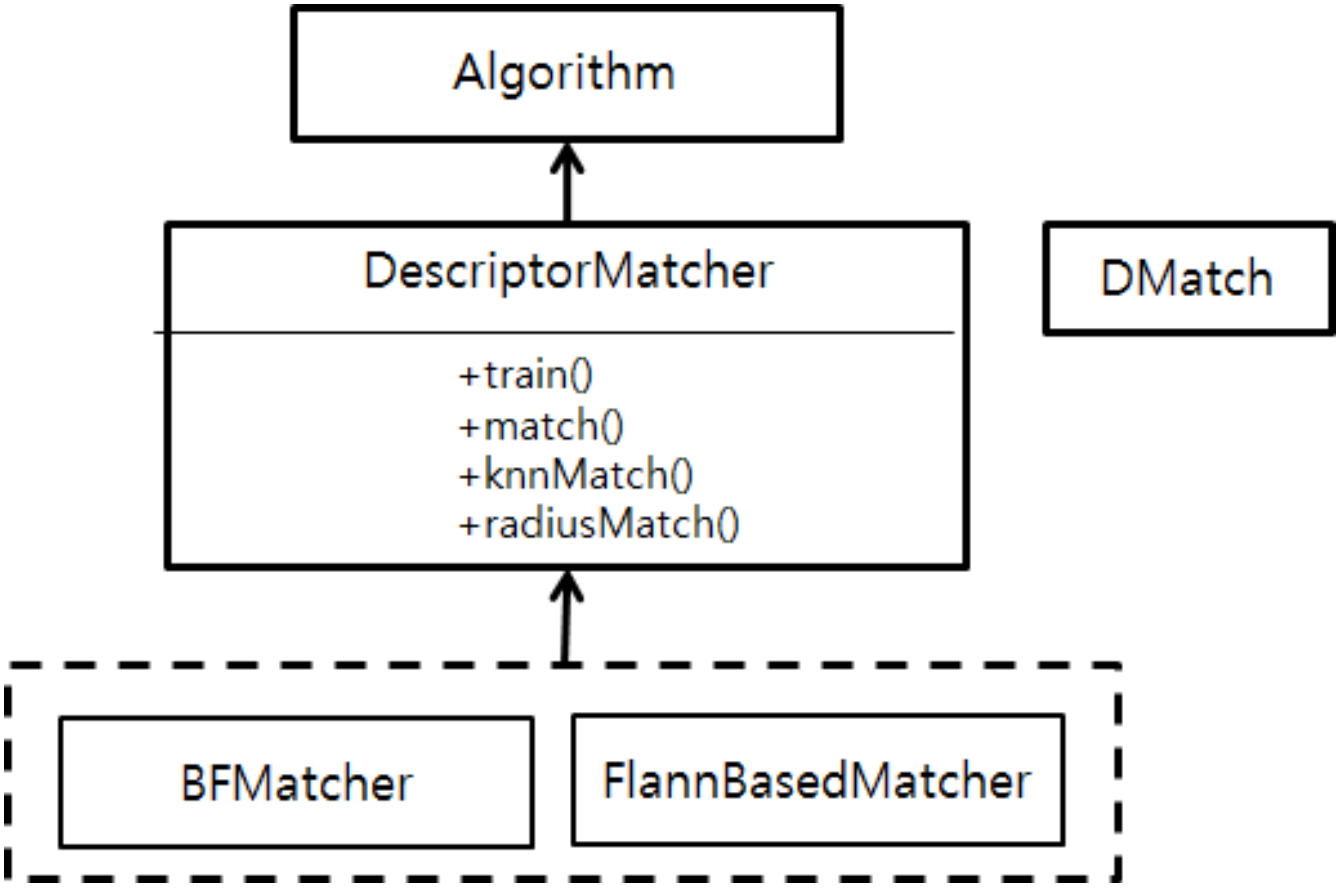
- 객체 찾기
  - 특징점을 이용한 매칭



(a)

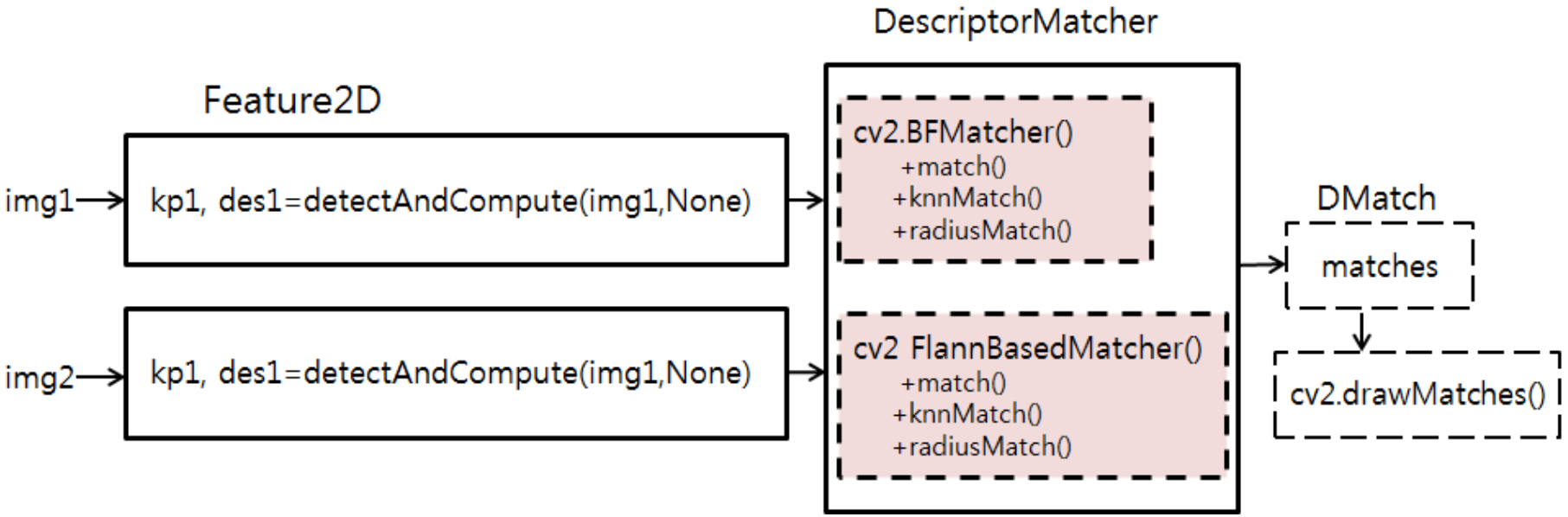
디스크립터를 이용한 매칭

- 특징 디스크립터 매칭 클래스 구조



디스크립터를 이용한 매칭

- 디스크립터를 이용한 매칭 과정



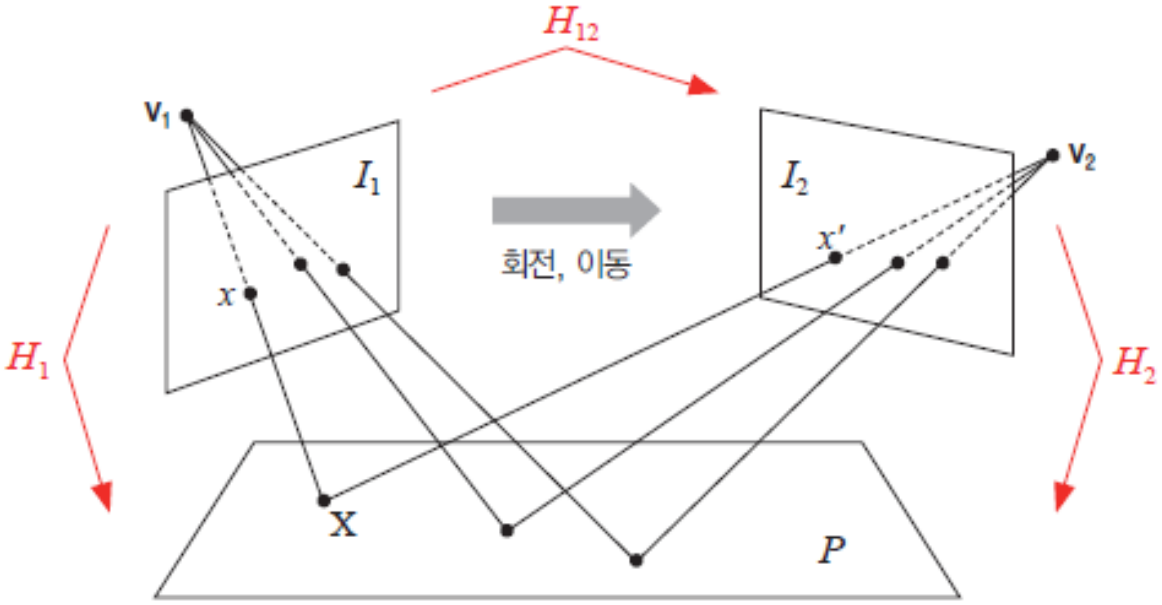
디스크립터를 이용한 매칭

- BFMatcher
  - 디스크립터를 일일이 하나씩 모두 검사하여 가장 가까운 디스크립터를 찾는 방법

- FlannBasedMatcher
  - Fast Library for Approximate Nearest Neighbors
  - 가장 가까운 이웃의 근사값으로 매칭 수행
  - 큰 데이터 셋과 고차원 특성에서 빠른 성능을 보임

호모그래피와 매칭

- 개념
  - 3차원 공간상의 평면을 서로 다른 시점에서 바라봤을 때 획득되는 영상 사이의 관계를 나타냄
  - 수학적으로는 하나의 평면을 다른 평면으로 투시 변환 했을 때 대응점 간 관계를 나타냄



호모그래피와 매칭

- 호모그래피 행렬 계산
  - 호모그래피는 투시변환이기 때문에 3x3 행렬로 표현
  - 네 개의 대응되는 점의 좌표 이동 정보가 있으면 행렬 계산 가능
  - 특징점 매칭 정보로부터 호모그래피를 구하는 경우 서로 대응되는 점 개수가 네 개 보다 많기 때문에 투시 변환 시 에러가 최소가 되는 형태의 호모그래피 행렬을 구해야 함

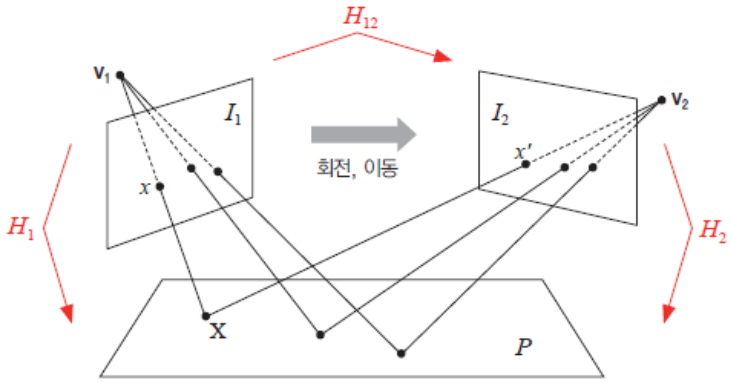
호모그래피와 매칭

- 호모그래피 행렬 계산
  - findHomography()함수를 이용하여 호모그래피 행렬 계산
  - 비용 함수가 최소가 되는 호모그래피 H 구하기

비용 함수 = $$\sum_i \left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 - \left(y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2$$

$$s_i \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

H

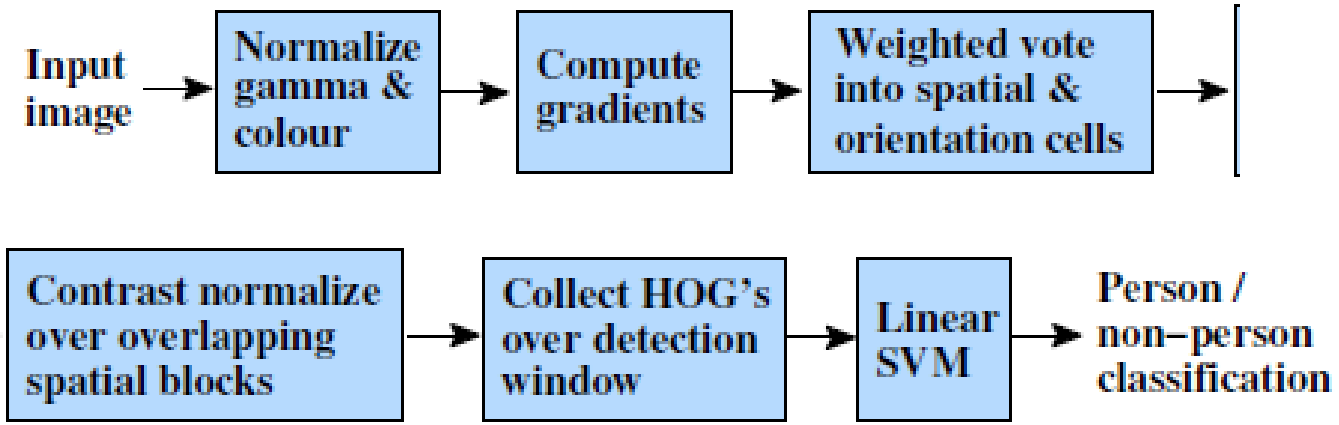


호모그래피와 매칭

- 호모그래피
  - 호모그래피 행렬 계산을 위해 findHomography()에서 method 설정
    - LMEDS
      - 이상치(outlier)가 50% 이하인 경우 잘 작동
    - RANSAC, RHO
      - 일반적으로 잘 작동
      - 이상치가 아니라는 판단을 위한 임계값 설정

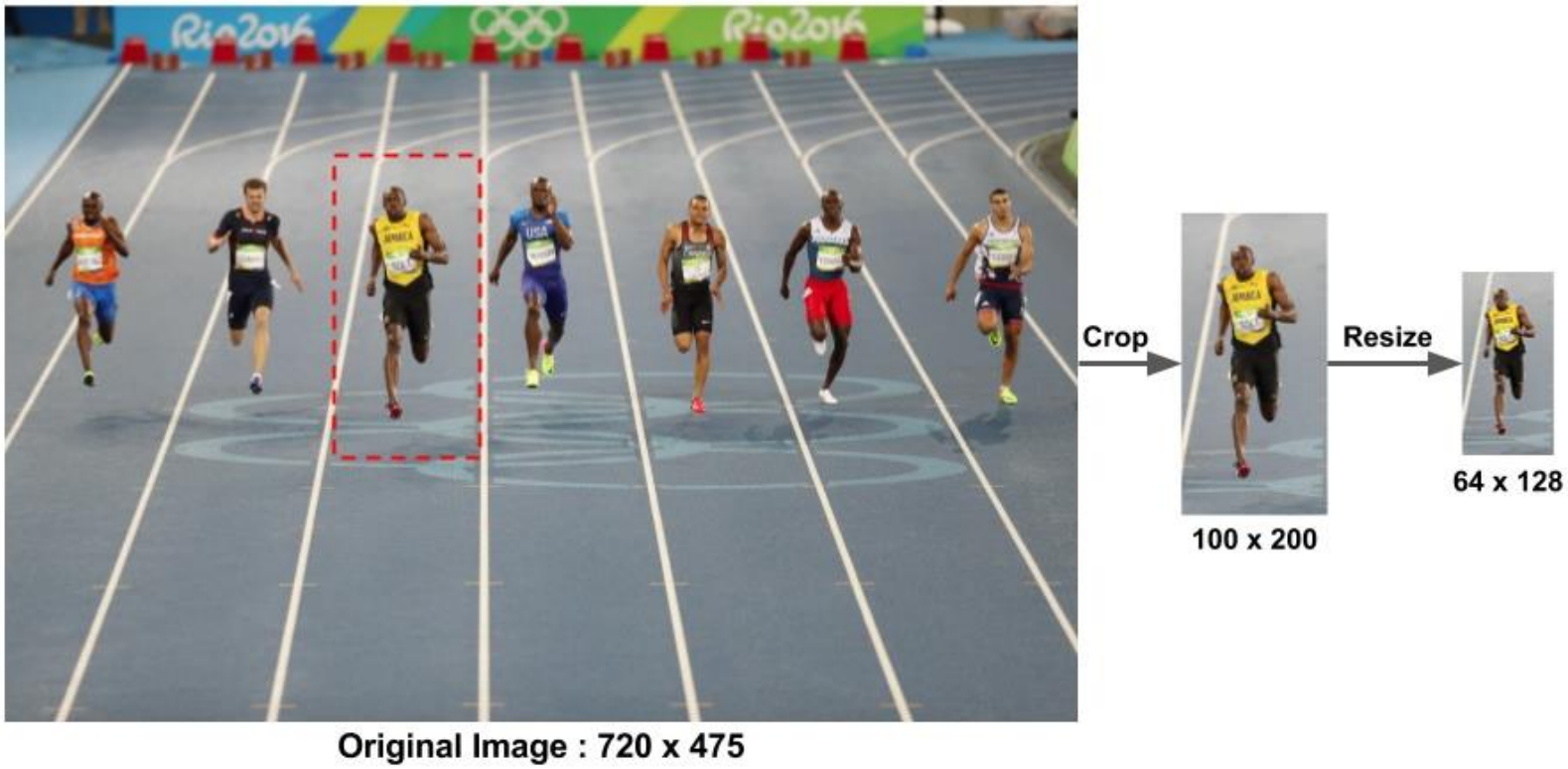
디스크립터

- HOG (Histogram of Oriented Gradient)
  - 물체 인식에 많이 사용되는 디스크립터
  - 보행자 검출이나 사람의 형태에 대한 검출 및 추적에 많이 사용됨
  - Local gradient 를 특징으로 사용



HOG 실행 과정

- 입력 영상 resize
  - Image ROI crop 후 종횡비가 1:2가 되도록 resize (예. w=64, h=128)



출처:
http://blog.naver.com/PostView.nhn?blogId=tommybee&logNo=221173056260&parentCategoryNo=&categoryNo=57&viewDate=&isShowPopularPosts=true&from=search

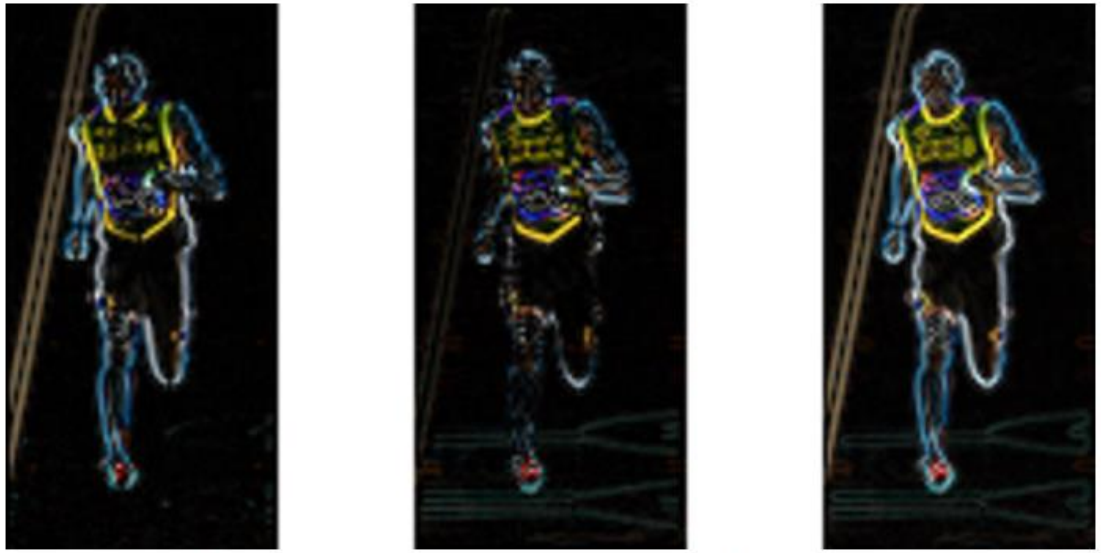
HOG 실행 과정

- Gradient 계산
  - 수평, 수직 방향의 gradient
  - 이를 이용하여 gradient histogram 계산

-1	0	1

-1
0
1

$$g = \sqrt{g_x^2 + g_y^2}$$
$$\theta = \arctan \frac{g_y}{g_x}$$

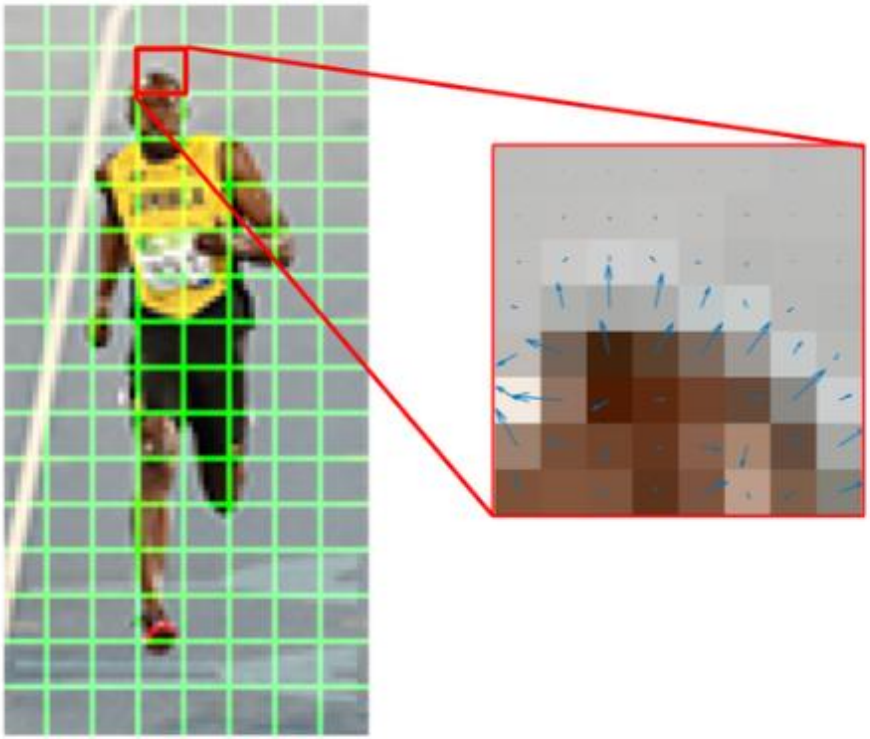


왼쪽: x- 그라디언트의 절대 값. 중앙: y- 그라디언트의 절대 값. 오른쪽 : 그라디언트의 크기.

출처:
http://blog.naver.com/PostView.nhn?blogId=tommybee&logNo=221173056260&parentCategoryNo=&categoryNo=57&viewDate=&isShowPopularPosts=true&from=search

HOG 실행 과정

- 8x8 셀 내의 histogram of gradients 계산
  - 8 x 8 크기의 패치로 나눔



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

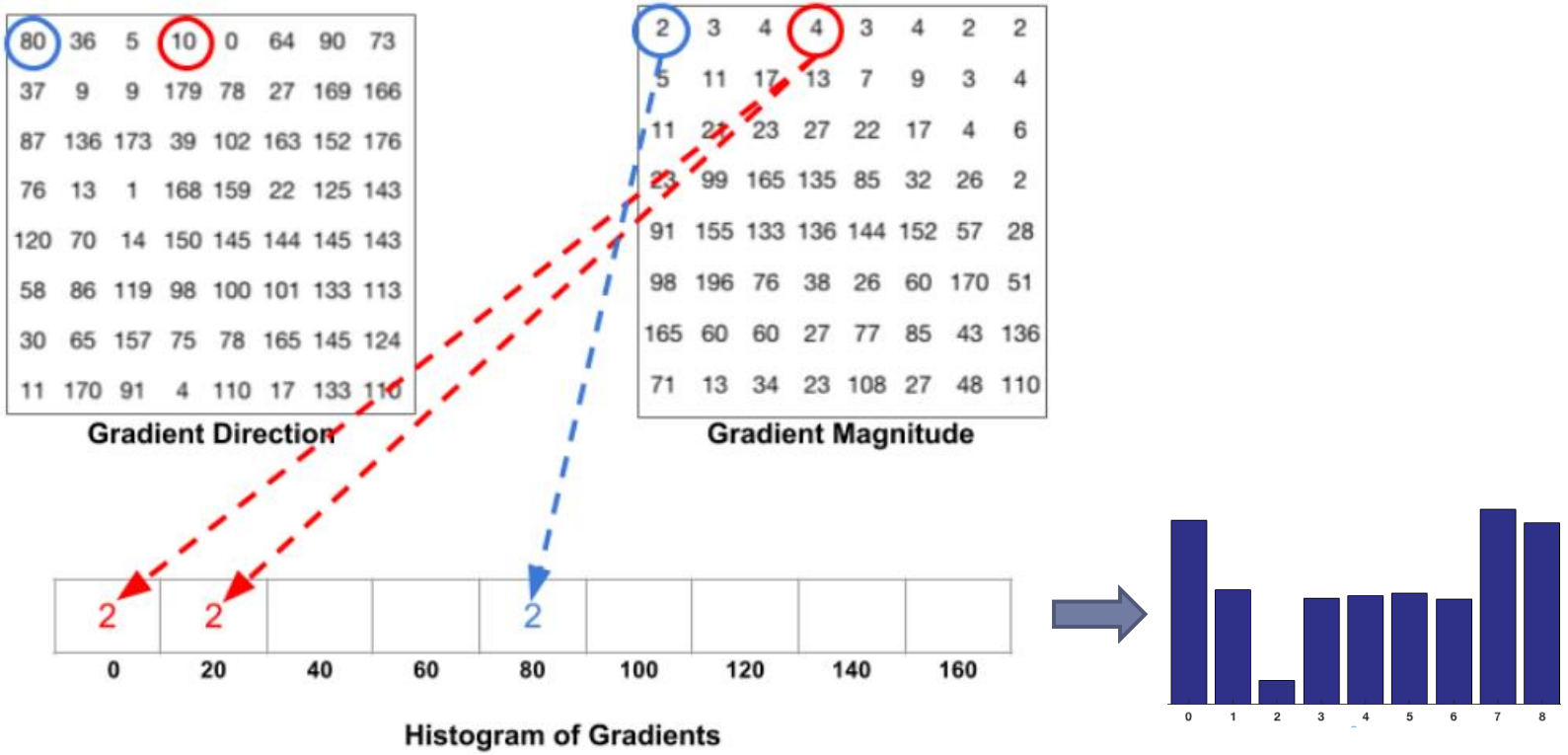
Gradient Direction

가운데 : 화살표를 사용하여 RGB 패치와 그라디언트를 나타냅니다. 오른쪽 : 동일한 패치의 그라디언트가 숫자로 표시됩니다.

출처: http://blog.naver.com/PostView.nhn?blogId=tommybee&logNo=221173056260&parentCategoryNo=&categoryNo=57&viewDate=&isShowPopularPosts=true&from=search

HOG 실행 과정

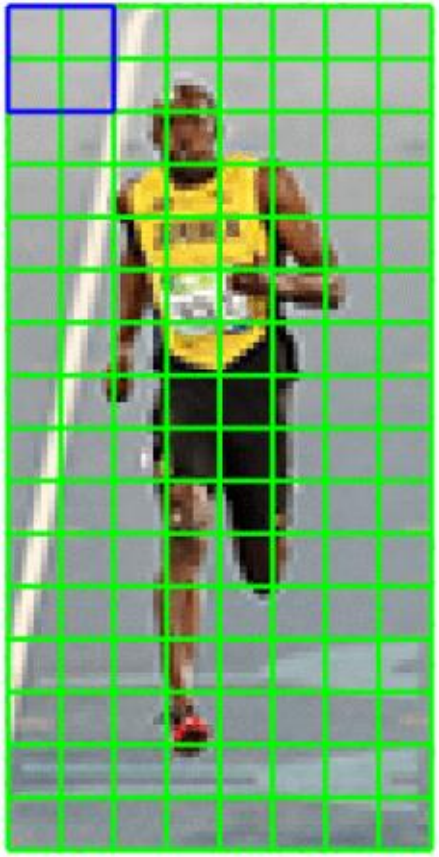
- 8x8 셀 내의 histogram of gradients 계산
  - 히스토그램 계산



출처: http://blog.naver.com/PostView.nhn?blogId=tommybee&logNo=221173056260&parentCategoryNo=&categoryNo=57&viewDate=&isShowPopularPosts=true&from=search

HOG 실행 과정

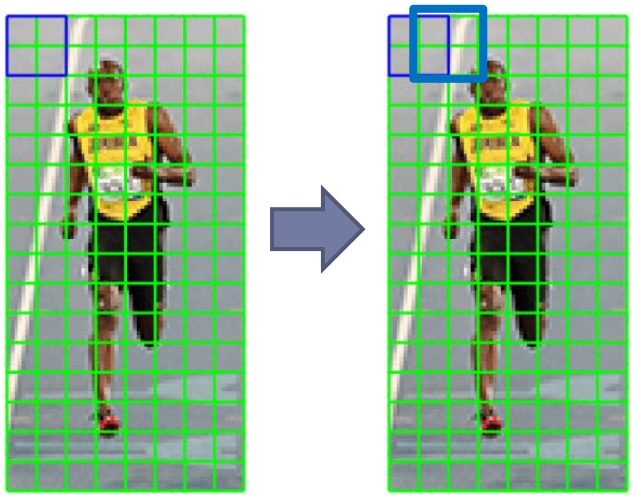
- 16x16 블록 정규화
  - Gradient histogram은 조명에 민감함
  - 최대한 조명 변화에 덜 민감하도록 히스토그램 정규화
    - 벡터 길이를 이용하여 정규화 벡터 생성
  - 16 x 16으로 총 36개(4x9)의 특징을 정규화



출처: http://blog.naver.com/PostView.nhn?blogId=tommybee&logNo=221173056260&parentCategoryNo=&categoryNo=57&viewDate=&isShowPopularPosts=true&from=search

HOG 실행 과정

- Descriptor 벡터 계산
  - 패치 당 36x1 벡터
  - 슬라이딩 윈도우를 이용하여 총 블록의 개수 구함
    - 7 x 15 = 105개
    - 36 x 105 = 3780 차원 벡터



출처: http://blog.naver.com/PostView.nhn?blogId=tommybee&logNo=221173056260&parentCategoryNo=&categoryNo=57&viewDate=&isShowPopularPosts=true&from=search

HOG 실행 과정

- Object detection
  - SVM 알고리즘 이용

영상 스티칭

- 개념
  - 여러 장의 영상을 서로 이어 붙여 하나의 큰 영상으로 만드는 기법
  - 스티칭 결과 영상을 파노라마 영상이라 함
  - 디지털 카메라, 스마트폰에서 기능 제공
  - 서로 겹치는 영역이 존재해야 함
  - 유의미한 특징점이 많을 수록 유리함

- 실행 단계
  - 입력영상에서 특징점 검출
  - 호모그래피(homography) 계산
  - 호모그래피 행렬을 기반으로 입력 영상을 변형하여 서로 이어 붙이는 작업 수행
  - 이어 붙인 부분을 자연스럽게 보이기 위해 블렌딩 처리

영상 스티칭

참고자료

- Python으로 배우는 OpenCV 프로그래밍
  - 김동근 지음
  - 가메 출판사, 2018
- OpenCV4 로 배우는 컴퓨터 비전과 머신러닝
  - 황선규 지음
  - 길벗, 2019