



국립공주대학교

KONG JU NATIONAL UNIVERSITY

[REPORT 2]



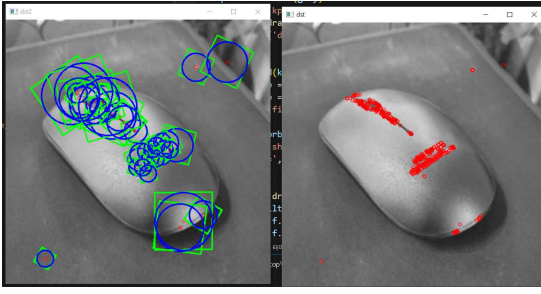
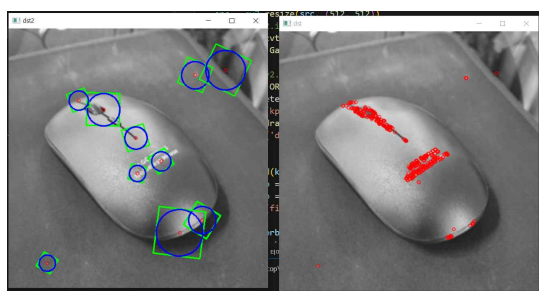
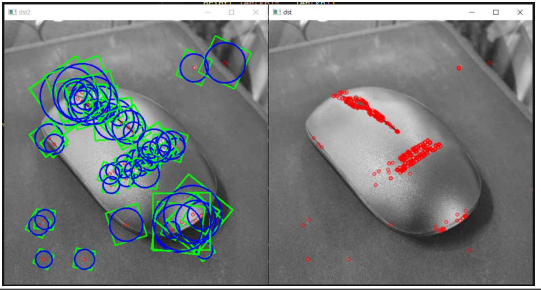
■	과	목	명	:	영상처리
■	학		과	:	컴퓨터공학
■	학		번	:	201905153
■	성		명	:	오세훈

영상처리 Report 2

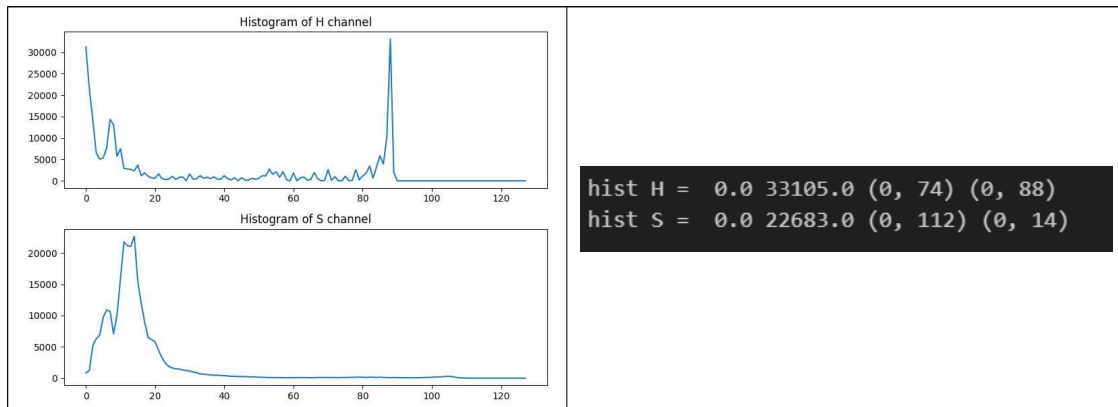
1. 각 영상에 대해 하나 또는 여러 개의 (Histogram, HOG, ORB, BRISK, KAZE, SIFT 등) 특징을 계산합니다.

1-1 ORB

Harris의 코너 반응값을 사용하여 특징점 계산

dstE=10	dstE=50
 <pre> len(kp)= 416 len(filtered_kp)= 38 des.shape= (38, 32) des= [[39 255 172 ... 119 46 255] [222 159 240 ... 247 198 95] [206 253 240 ... 247 143 119] ... [127 32 33 ... 47 59 116] [199 68 167 ... 117 177 100] [105 40 157 ... 168 82 40]] </pre>	 <pre> len(kp)= 416 len(filtered_kp)= 10 des.shape= (10, 32) des= [[28 141 31 237 176 202 67 76 120 89 137 9 62 233 231 [105 237 122 70 42 89 87 245 196 106 204 42 236 45 98 205 [61 255 15 15 217 87 103 248 234 238 237 210 253 63 125 39 </pre>
<p>blur x, dstE=10</p>  <pre> len(kp)= 480 len(filtered_kp)= 57 des.shape= (57, 32) des= [[2 144 180 ... 12 157 215] [130 184 155 ... 0 161 171] [120 167 184 ... 64 135 227] ... [80 169 153 ... 34 7 42] [127 32 33 ... 39 59 116] [105 40 157 ... 168 82 40]] </pre>	<p>dstE = 10 416개의 특징점이 검출되었고 dstE 이내의 특징점을 제거하여 38인 filtered가 생성</p> <p>dstE = 50 416개의 특징점이 검출되었고 dstE 이내의 특징점을 제거하여 10인 filtered가 생성</p> <p>blur 처리를 안할 경우 더 많은 특징점이 검출 BRICK 보다는 정확성이 높음</p>

1-2 Histogram







HSV로 변환하여 H, S의 값을 histogram 연산을 통해 유사 이미지 검색

```
query = hists[0]
methods = {'CORREL': cv2.HISTCMP_CORREL, 'CHISQR': cv2.HISTCMP_CHISQR,
           'INTERSECT': cv2.HISTCMP_INTERSECT}

for j, (name, flag) in enumerate(methods.items()):
    print('%-10s'%name, end='\t')
    for i, (hist, img) in enumerate(zip(hists, imgs)):
        #--@ 각 메서드에 따라 img1과 각 이미지의 히스토그램 비교
        ret = cv2.compareHist(query, hist, flag)
        if flag == cv2.HISTCMP_INTERSECT: #교차 분석인 경우
            ret = ret/np.sum(query) #비교대상으로 나누어 1로 정규화
        print("img%d:%7.2f"% (i+1, ret), end='\t')
    print()
plt.show()
```

- * cv2.HISTCMP_CORREL: 상관관계 (1: 완전 일치, -1: 완전 불일치, 0: 무관계)
- * cv2.HISTCMP_CHISQR: 카이제곱 (0: 완전 일치, 무한대: 완전 불일치)
- * cv2.HISTCMP_INTERSECT: 교차 (1: 완전 일치, 0: 완전 불일치 - 1로 정규화한 경우)

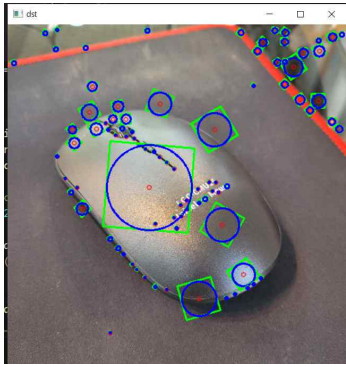
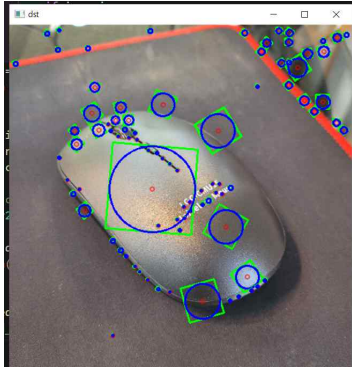
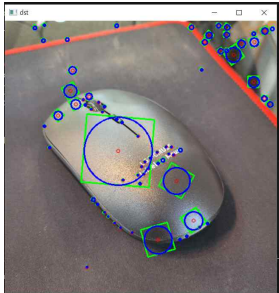
img1 = 입력 이미지, img1 포함 나머지 비교 이미지				
	img1	img2	img3	img4
				
bin = 32				
CORREL	img1: 1.00	img2: 0.87	img3: 0.16	img4: 0.08
CHISQR	img1: 0.00	img2: 3.73	img3: 1683.49	img4: 50237.99
INTERSECT	img1: 1.00	img2: 0.63	img3: 0.21	img4: 0.24

bin = 128					
CORREL	img1:	1.00	img2:	0.75	img3: 0.09 img4: 0.05
CHISQR	img1:	0.00	img2:	138.73	img3:1355.65 img4:98757.78
INTERSECT	img1:	1.00	img2:	0.75	img3: 0.17 img4: 0.31

bin 크기가 작을수록 유사도에 가깝게 나오는 경우도 있고 멀게 나오는 경우도 있는 걸 확인할 수 있다. 특히 img는 bin의 크기가 커질수록 유사하다는 것을 알 수 있는데 img4는 이와 반대로 유사하지 않는다는 것을 알 수 있다.

1-3 SIFT

특징점 주위의 영역에서 영상 그레디언트와 크기와 방향을 계산하고, 가우시안 함수로 가중 필터링 4x4 영역으로 나누어 방향에 따라 크기를 histogram으로 계산한다. gradient의 크기와 방향을 디스크립터로 사용, 임계 값이 작을수록 검출되는 특징점이 적어지는 것을 알 수 있다.

threshold 80	threshold 20
	
<pre>len(kp)= 256 len(filtered_kp)= 88 des.shape= (88, 128) des.dtype= float32 des= [[0. 1. 2. ... 0. 2. 6.] [16. 0. 0. ... 0. 0. 1.] [0. 0. 1. ... 1. 0. 24.] ... [24. 4. 1. ... 7. 2. 4.] [77. 41. 1. ... 19. 2. 16.] [45. 105. 32. ... 10. 0. 2.]]</pre>	<pre>len(kp)= 220 len(filtered_kp)= 84 des.shape= (84, 128) des.dtype= float32 des= [[0. 1. 2. ... 0. 2. 6.] [16. 0. 0. ... 0. 0. 1.] [0. 0. 1. ... 1. 0. 24.] ... [24. 4. 1. ... 7. 2. 4.] [77. 41. 1. ... 19. 2. 16.] [45. 105. 32. ... 10. 0. 2.]]</pre>
threshold x	
	<pre>len(kp)= 169 len(filtered_kp)= 72 des.shape= (72, 128) des.dtype= float32 des= [[0. 1. 2. ... 0. 2. 6.] [16. 0. 0. ... 0. 0. 1.] [0. 0. 1. ... 1. 0. 24.] ... [24. 4. 1. ... 7. 2. 4.] [77. 41. 1. ... 19. 2. 16.] [45. 105. 32. ... 10. 0. 2.]]</pre>

SIFT와 BFMatcher를 이용해 이미지 유사도 확인, 입력 이미지와 비슷할수록 먼저 출력 해당 이미지들로 테스트를 진행했습니다.

```
[path + 'flower1.jpg', path + 'flower2.jpg', path + 'flower3.jpg', path + 'flower4.jpg', path + 'flower5.jpg',
path + 'umb1.jpg', path + 'umb2.jpg', path + 'umb3.jpg', path + 'umb4.jpg', path + 'umb5.jpg',
path + 'mouse1.jpg', path + 'mouse2.jpg', path + 'mouse3.jpg', path + 'mouse4.jpg', path + 'mouse5.jpg']
```

```
# data 이미지와 입력 이미지 간에 유사도 확인
for image_path in database_image_paths:
    # data 이미지 불러오기
    database_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    database_image = cv2.resize(database_image, imgSize)

    # 특징점과 디스크립터 검출
    _, database_descriptors = sift.detectAndCompute(database_image, None)

    # data 이미지와 query 이미지 확인
    matches = bf.match(query_descriptors, database_descriptors)

    # distnace를 이용해 유사도 거리 계산
    similarity_score = sum([match.distance for match in matches])

    # 유사성 점수 및 경로 저장
    similarity_scores.append(similarity_score)
    similar_image_paths.append(image_path)
```

<p>src = umb6.jpg</p> 	<p>src = mouse6.jpg</p> 
<p>src = flower6.jpg</p> 	<p>bf.matcher()를 통해 이미지 검색을 진행했는데 우산이랑 꽃은 잘 찾아내는데 마우스는 5개중에 2개만 찾은 것을 확인</p>

2. 여러개의 디스크립터를 계산할 경우, VLAD (Vector of Locally Aggregated Descriptors)를 이용하여 하나의 벡터로 통합 계산합니다.

VLAD 구현 함수

```
def compute_vlad(image, des, labels, centers):
    # SIFT 특징점 검출과 디스크립터 계산을 한 번에 수행
    sift = cv2.SIFT_create(edgeThreshold=80)
    _, des = sift.detectAndCompute(image, None)

    if des is None:
        return None

    # VLAD 벡터 초기화
    vlad = np.zeros((centers.shape[0], des.shape[1]), dtype=np.float32)

    # 누적합
    for i in range(des.shape[0]):
        vlad[labels[i]] += des[i] - centers[labels[i]]

    # VLAD 벡터 정규화
    vlad = cv2.normalize(vlad, None).flatten()
    vlad /= np.linalg.norm(vlad)
    return vlad

# k-means clustering
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
flags = cv2.KMEANS_RANDOM_CENTERS
_, labels, centers = cv2.kmeans(category_des, k, None, criteria, 5, flags)
```

이미지를 vlad로 연산하여 각 카테고리 별로 입력 이미지와 비교하여 유사도가 높은 이미지 카테고리 출력, 0에 가까울수록 입력 영상과 카테고리의 유사도가 높음

```
# 입력 이미지
# query_image_path = path + 'flower6.jpg'
# query_image_path = path + 'mouse6.jpg'
query_image_path = path + 'umb6.jpg'

# 분류할 카테고리 및 해당 카테고리의 이미지들을 딕셔너리로 정의
categories = {
    'mouse': [cv2.imread(path + 'mouse1.jpg'), cv2.imread(path + 'mouse2.jpg'), cv2.imread(path + 'mouse3.jpg'),
              cv2.imread(path + 'mouse4.jpg'), cv2.imread(path + 'mouse5.jpg')],
    'flower': [cv2.imread(path + 'flower1.jpg'), cv2.imread(path + 'flower2.jpg'), cv2.imread(path + 'flower3.jpg'),
              cv2.imread(path + 'flower4.jpg'), cv2.imread(path + 'flower5.jpg')],
    'umbrella': [cv2.imread(path + 'umb1.jpg'), cv2.imread(path + 'umb2.jpg'), cv2.imread(path + 'umb3.jpg'),
               cv2.imread(path + 'umb4.jpg'), cv2.imread(path + 'umb5.jpg')]
}
```

결과

flower6.jpg	mouse6.jpg	umb6.jpg
k = 16	k = 16	k = 16
0에 가까울 수록 입력 이미지와 유사하다. 카테고리: flower / 유사도: 0.18 카테고리: mouse / 유사도: 0.84 카테고리: umbrella / 유사도: 0.88	0에 가까울 수록 입력 이미지와 유사하다. 카테고리: mouse / 유사도: 0.37 카테고리: umbrella / 유사도: 0.45 카테고리: flower / 유사도: 0.67	0에 가까울 수록 입력 이미지와 유사하다. 카테고리: umbrella / 유사도: 0.32 카테고리: mouse / 유사도: 0.40 카테고리: flower / 유사도: 0.74

k = 2	k = 2	k = 2
0에 가까운 수록 입력 이미지와 유사하다. 카테고리: flower / 유사도: 0.15 카테고리: mouse / 유사도: 0.99 카테고리: umbrella / 유사도: 1.13	0에 가까운 수록 입력 이미지와 유사하다. 카테고리: mouse / 유사도: 0.37 카테고리: umbrella / 유사도: 0.59 카테고리: flower / 유사도: 0.79	0에 가까운 수록 입력 이미지와 유사하다. 카테고리: umbrella / 유사도: 0.37 카테고리: mouse / 유사도: 0.52 카테고리: flower / 유사도: 1.00

3. 영상의 유사도는 Faiss: A library for efficient similarity search을 사용합니다.

```

similarity_scores = {}
for category, images in category_images.items():
    category_vlads = []
    for image in images:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, (300, 300))
        _, des = sift.detectAndCompute(image, None)
        if des is not None:
            category_vlads.append(compute_vlad(image, des, labels, centers))

    category_vlads = np.array(category_vlads)

    # Faiss Index 생성
    d = query_vlad.shape[0]
    index = faiss.IndexFlatL2(d)
    index.add(category_vlads)

    # Query 이미지를 Faiss Index에 검색
    distances, _ = index.search(query_vlad.reshape(1, -1), 10)
    similarity_scores[category] = np.mean(distances)

# 0에 가까운 것이 유사도가 가장 높으므로 오름차순 정렬
sorted_categories = sorted(similarity_scores.items(), key=lambda x: x[1])
return sorted_categories

```

카테고리별로 data 이미지를 평균 낸 category_vlads를 통해 입력 영상과 비교하여 유사도 판단, 0에 가까울수록 해당 카테고리에 속함

```

입력 이미지: c:/data/temp/flower1.jpg
제일 유사한 카테고리: ('flower', 0.010839691)
제일 유사하지 않은 카테고리: ('shirt', 1.2408476)

0에 가까운 수록 입력 이미지와 유사하다.
카테고리: flower / 유사도: 0.01
카테고리: muffin / 유사도: 0.36
카테고리: balloon / 유사도: 0.37
카테고리: book / 유사도: 0.48
카테고리: mouse / 유사도: 0.54
카테고리: game / 유사도: 0.56
카테고리: box / 유사도: 0.64
카테고리: umbrella / 유사도: 0.67
카테고리: scissors / 유사도: 0.89
카테고리: shirt / 유사도: 1.24

```


4. 영상은 스마트폰으로 촬영한 10종류(각각 10장 이상) 이상의 영상에 대해 실험합니다.
해당 과제를 수행하는데 있어 사용한 이미지 모든 이미지의 1번은 query 이미지 2~11번의 이미지는 data 이미지로 사용, 이미지 사이즈는 300x300 크기로 resize 수행

이미지 유사도 확인 테스트

<p>입력 이미지: c:/data/temp/flower1.jpg 제일 유사한 카테고리: ('flower', 0.010839691) 제일 유사하지 않은 카테고리: ('shirt', 1.2408476)</p> <p>0에 가까운 수록 입력 이미지와 유사하다.</p> <p>카테고리: flower / 유사도: 0.01 카테고리: muffin / 유사도: 0.36 카테고리: balloon / 유사도: 0.37 카테고리: book / 유사도: 0.48 카테고리: mouse / 유사도: 0.54 카테고리: game / 유사도: 0.56 카테고리: box / 유사도: 0.64 카테고리: umbrella / 유사도: 0.67 카테고리: scissors / 유사도: 0.89 카테고리: shirt / 유사도: 1.24</p>	<p>입력 이미지: c:/data/temp/mouse1.jpg 제일 유사한 카테고리: ('mouse', 0.18214953) 제일 유사하지 않은 카테고리: ('flower', 2.3962483)</p> <p>0에 가까운 수록 입력 이미지와 유사하다.</p> <p>카테고리: mouse / 유사도: 0.18 카테고리: balloon / 유사도: 0.27 카테고리: umbrella / 유사도: 0.34 카테고리: box / 유사도: 0.40 카테고리: book / 유사도: 0.46 카테고리: shirt / 유사도: 0.46 카테고리: muffin / 유사도: 0.54 카테고리: game / 유사도: 0.55 카테고리: scissors / 유사도: 0.82 카테고리: flower / 유사도: 2.40</p>	<p>입력 이미지: c:/data/temp/umb1.jpg 제일 유사한 카테고리: ('umbrella', 0.096167654) 제일 유사하지 않은 카테고리: ('flower', 1.0954467)</p> <p>0에 가까운 수록 입력 이미지와 유사하다.</p> <p>카테고리: umbrella / 유사도: 0.10 카테고리: balloon / 유사도: 0.32 카테고리: book / 유사도: 0.34 카테고리: shirt / 유사도: 0.40 카테고리: game / 유사도: 0.40 카테고리: mouse / 유사도: 0.40 카테고리: box / 유사도: 0.43 카테고리: scissors / 유사도: 0.43 카테고리: muffin / 유사도: 0.46 카테고리: flower / 유사도: 1.10</p>
<p>입력 이미지: c:/data/temp/muff1.jpg 제일 유사한 카테고리: ('muffin', 0.03737215) 제일 유사하지 않은 카테고리: ('scissors', 0.70272446)</p> <p>0에 가까운 수록 입력 이미지와 유사하다.</p> <p>카테고리: muffin / 유사도: 0.04 카테고리: balloon / 유사도: 0.14 카테고리: game / 유사도: 0.18 카테고리: mouse / 유사도: 0.20 카테고리: box / 유사도: 0.20 카테고리: book / 유사도: 0.23 카테고리: umbrella / 유사도: 0.24 카테고리: flower / 유사도: 0.33 카테고리: shirt / 유사도: 0.49 카테고리: scissors / 유사도: 0.70</p>	<p>입력 이미지: c:/data/temp/book1.jpg 제일 유사한 카테고리: ('book', 0.078539036) 제일 유사하지 않은 카테고리: ('flower', 0.7638437)</p> <p>0에 가까운 수록 입력 이미지와 유사하다.</p> <p>카테고리: book / 유사도: 0.08 카테고리: scissors / 유사도: 0.22 카테고리: umbrella / 유사도: 0.27 카테고리: balloon / 유사도: 0.34 카테고리: game / 유사도: 0.36 카테고리: box / 유사도: 0.39 카테고리: mouse / 유사도: 0.43 카테고리: muffin / 유사도: 0.48 카테고리: shirt / 유사도: 0.62 카테고리: flower / 유사도: 0.76</p>	<p>입력 이미지: c:/data/temp/box1.jpg 제일 유사한 카테고리: ('box', 0.075382374) 제일 유사하지 않은 카테고리: ('scissors', 0.8066489)</p> <p>0에 가까운 수록 입력 이미지와 유사하다.</p> <p>카테고리: box / 유사도: 0.08 카테고리: mouse / 유사도: 0.14 카테고리: game / 유사도: 0.20 카테고리: shirt / 유사도: 0.20 카테고리: balloon / 유사도: 0.21 카테고리: muffin / 유사도: 0.22 카테고리: book / 유사도: 0.29 카테고리: umbrella / 유사도: 0.37 카테고리: flower / 유사도: 0.64 카테고리: scissors / 유사도: 0.81</p>
<p>입력 이미지: c:/data/temp/game1.jpg 제일 유사한 카테고리: ('game', 0.070570804) 제일 유사하지 않은 카테고리: ('scissors', 0.6481677)</p> <p>0에 가까운 수록 입력 이미지와 유사하다.</p> <p>카테고리: game / 유사도: 0.07 카테고리: box / 유사도: 0.16 카테고리: muffin / 유사도: 0.17 카테고리: balloon / 유사도: 0.24 카테고리: shirt / 유사도: 0.25 카테고리: book / 유사도: 0.26 카테고리: mouse / 유사도: 0.28 카테고리: umbrella / 유사도: 0.29 카테고리: flower / 유사도: 0.51 카테고리: scissors / 유사도: 0.65</p>	<p>입력 이미지: c:/data/temp/scil.jpg 제일 유사한 카테고리: ('scissors', 0.22910385) 제일 유사하지 않은 카테고리: ('shirt', 1.014128)</p> <p>0에 가까운 수록 입력 이미지와 유사하다.</p> <p>카테고리: scissors / 유사도: 0.23 카테고리: book / 유사도: 0.26 카테고리: umbrella / 유사도: 0.39 카테고리: balloon / 유사도: 0.46 카테고리: game / 유사도: 0.53 카테고리: muffin / 유사도: 0.53 카테고리: flower / 유사도: 0.55 카테고리: mouse / 유사도: 0.65 카테고리: box / 유사도: 0.73 카테고리: shirt / 유사도: 1.01</p>	<p>입력 이미지: c:/data/temp/shirt1.jpg 제일 유사한 카테고리: ('shirt', 0.016824102) 제일 유사하지 않은 카테고리: ('flower', 1.3635536)</p> <p>0에 가까운 수록 입력 이미지와 유사하다.</p> <p>카테고리: shirt / 유사도: 0.02 카테고리: box / 유사도: 0.28 카테고리: game / 유사도: 0.28 카테고리: umbrella / 유사도: 0.37 카테고리: mouse / 유사도: 0.41 카테고리: book / 유사도: 0.48 카테고리: balloon / 유사도: 0.52 카테고리: muffin / 유사도: 0.58 카테고리: scissors / 유사도: 0.82 카테고리: flower / 유사도: 1.36</p>

전체적으로 query 이미지를 data 이미지와 비교하여 유사한 카테고리를 잘 찾는 것을 확인할 수 있다.

5. 참고 문헌

<https://github.com/jorjasso/VLAD>

<https://github.com/facebookresearch/faiss>

책, LMS강의영상

6. 느낀점

인터넷으로 찾아보고 책이랑 영상 보면서 했는데 많이 어렵기도 했지만 완성하고 나니 생각보다 재밌고 신기합니다.