# 차량비전시스템

## Assignment 5


학번 5533082
전공 컴퓨터공학전공
이름 최승환

# Topic. 특징점 추출 및 이미지 매칭

Python: 3.6.13
Opencv: 3.4.2.15

## (1) stereo dataset 선택

왼쪽에서 찍은 이미지(view1)와 오른쪽에서 찍은 이미지(view5) 그리고 view1의 석고상 부분을 잘라내어 template 이미지로써 저장하였다.

left image


right image


Template 이미지는 이미지 매칭에서 수행될 때 사용된다.

template image

### (2) 좌/우 이미지로부터 특징점 추출 – SIFT/SURF/ORB

1. SIFT/SURF/ORB 서술자 추출기 생성

SIFT: cv2.xfeatures2d.SIFT_create()
SURF: cv2.xfeatures2d.SURF_create()
ORB: cv2.ORB_create()

2. 각 영상에 대한 키포인트와 서술자 추출

키포인트와 서술자 계산: detector.detectAndCompute()
키포인트 이미지 생성: cv2.drawKeypoints()

```
# SIFT 서술자 추출기 생성 ---①
sift = cv2.xfeatures2d.SIFT_create()
surf = cv2.xfeatures2d.SURF_create()
orb = cv2.ORB_create()

# 각 영상에 대해 키 포인트와 서술자 추출 ---②
# 1. SIFT
keypoints1, descriptors1 = sift.detectAndCompute(left_gray, None)
keypoints2, descriptors2 = sift.detectAndCompute(right_gray, None)
sift_desc_image1 = cv2.drawKeypoints(left_img, keypoints1, None,
flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
sift_desc_image2 = cv2.drawKeypoints(right_img, keypoints2, None,
flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)

# 2. SURF
keypoints, descriptors = surf.detectAndCompute(left_gray, None)
keypoints2, descriptors2 = surf.detectAndCompute(right_gray, None)
surf_desc_image1 = cv2.drawKeypoints(left_img, keypoints, None,
flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
surf_desc_image2 = cv2.drawKeypoints(right_img, keypoints2, None,
flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)

# 3. ORB
keypoints, descriptors = orb.detectAndCompute(left_gray, None)
keypoints2, descriptors2 = orb.detectAndCompute(right_gray, None)
orb_desc_image1 = cv2.drawKeypoints(left_img, keypoints, None,
flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
orb_desc_image2 = cv2.drawKeypoints(right_img, keypoints2, None,
flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
```

2-1. left SIFT

2-1. right SIFT

2-2. left SURF

2-2. right SURF

2-3. left ORB

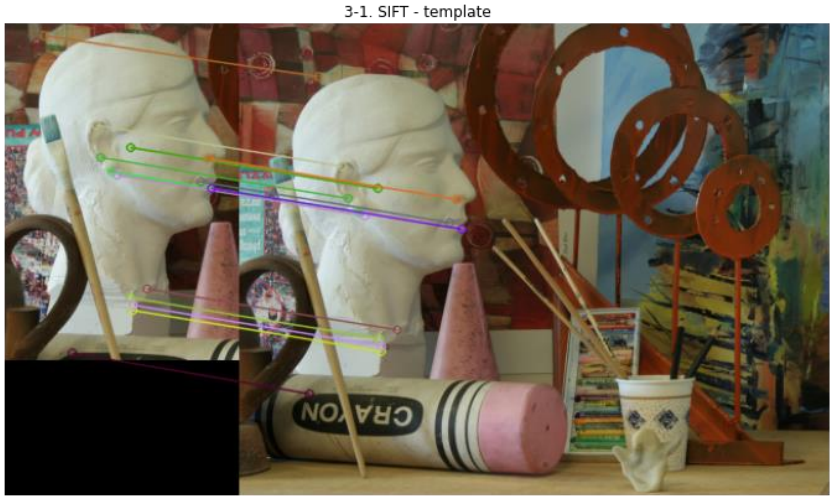
2-3. right ORB

### (3) 이미지 매칭 – template 기반 매칭

target 이미지(view5)와 template 이미지(view1)의 이미지 매칭 수행 결과,
SIFT와 SURF를 사용한 특징점 매칭은 원활했지만 ORB는 view1에 대한 template 이미지 이외에는 효과적이지 못했다.

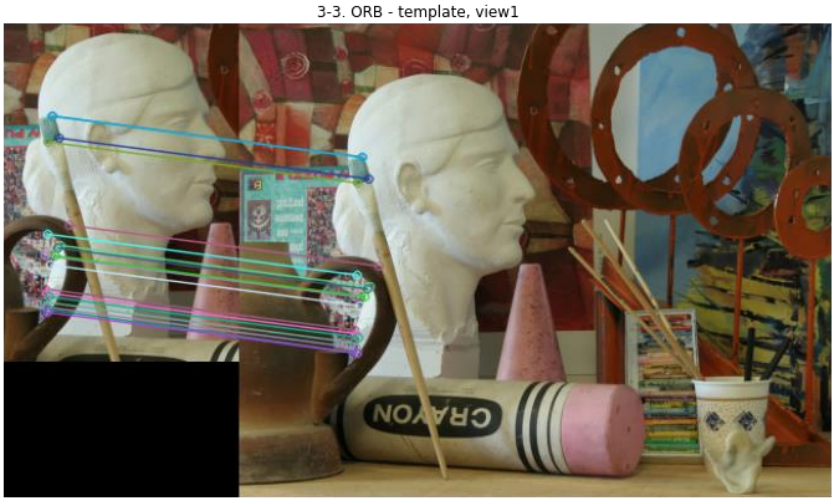
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the main image and template image
image = cv2.imread('./img/Art/view5.png')
template = cv2.imread('./template.png')

sift = cv2.xfeatures2d.SIFT_create()

gray_template = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# SIFT feature extraction
kp1, des1 = sift.detectAndCompute(gray_template, None)
kp2, des2 = sift.detectAndCompute(gray_image, None)

flann_matcher = cv2.DescriptorMatcher_create(cv2.DescriptorMatcher_FLANNBASED)
knn_match = flann_matcher.knnMatch(des1, des2, k=2)

# Apply ratio test to filter good matches
T=0.45
good_match = []
for nearest1, nearest2 in knn_match:
    if (nearest1.distance/nearest2.distance)<T:
        good_match.append(nearest1)

# Draw matches on the image
match_image = cv2.drawMatches(template, kp1, image, kp2, good_match, None,
                                      flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

# Display the matched image
plt.figure(figsize=(15,7))
plt.imshow(cv2.cvtColor(match_image, cv2.COLOR_BGR2RGB)), plt.title("3-1. SIFT - template"),
plt.axis("off")
plt.show()
```

3-1. SIFT - template

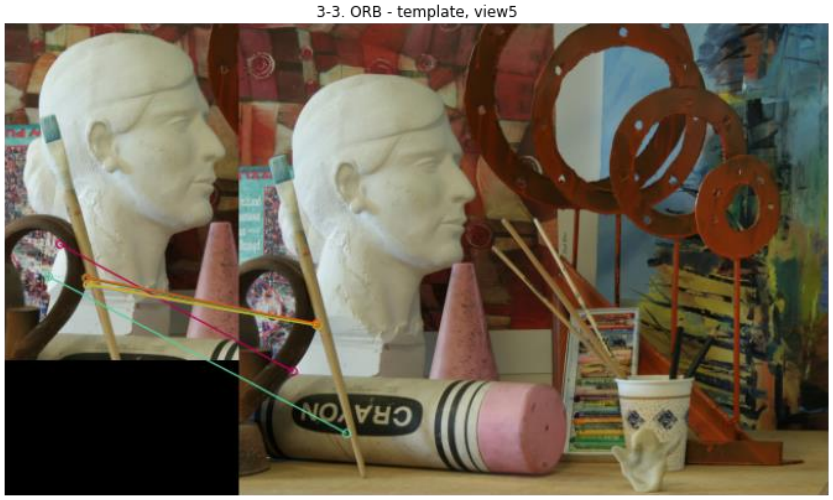



3-2. SURF - template


3-3. ORB - template, view1




3-3. ORB - template, view5

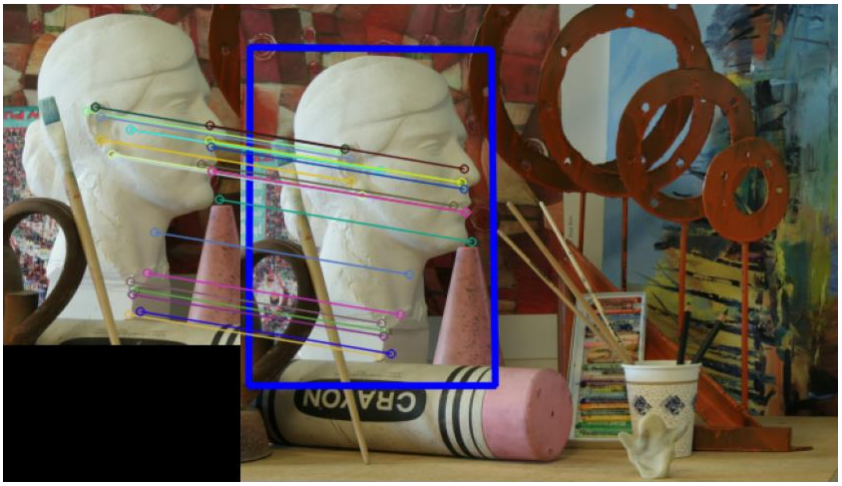


### (3) 이미지 매칭 – RANSAC 기반 매칭

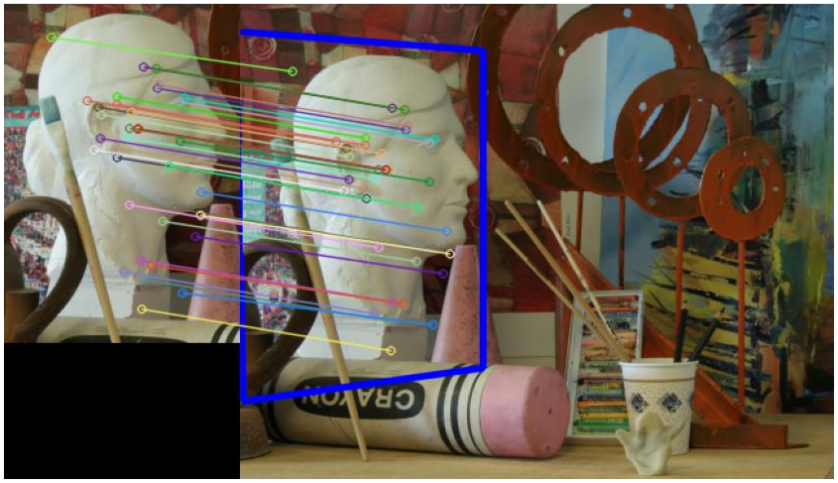
target 이미지(view5)와 template 이미지(view1)의 이미지 매칭 수행 결과,
SIFT와 SURF를 사용한 특징점 매칭은 원활했지만 ORB는 view1에 대한 template 이미지 이외에는 효과적이지 못했다.

```
detector = cv2.xfeatures2d.SIFT_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)

matcher = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
matches = matcher.match(desc1, desc2)
matches = sorted(matches, key=lambda x:x.distance)

# 매칭점으로 원근 변환 및 영역 표시
src_pts = np.float32([ kp1[m.queryIdx].pt for m in matches ])
dst_pts = np.float32([ kp2[m.trainIdx].pt for m in matches ])

# RANSAC 으로 변환 행렬 근사 계산
mtrx, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC)
h,w = img1.shape[:2]
pts = np.float32([ [[0,0]],[[0,h-1]],[[w-1,h-1]],[[w-1,0]] ])
dst = cv2.perspectiveTransform(pts,mtrx)
img2 = cv2.polylines(img2,[np.int32(dst)],True,255,3, cv2.LINE_AA)

# 정상치 매칭만 그리기
matchesMask = mask.ravel().tolist()
res2 = cv2.drawMatches(img1, kp1, img2, kp2, matches, None, \
                   matchesMask = matchesMask,
                   flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
```

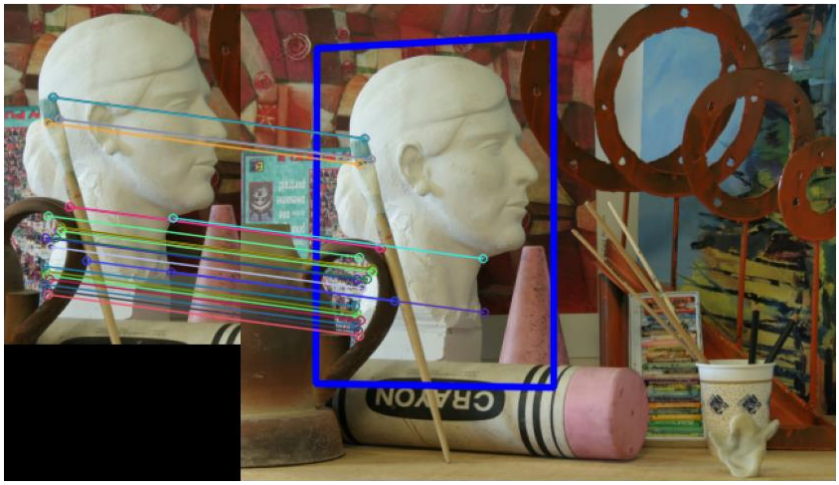
3-4. SIFT - RANSAC



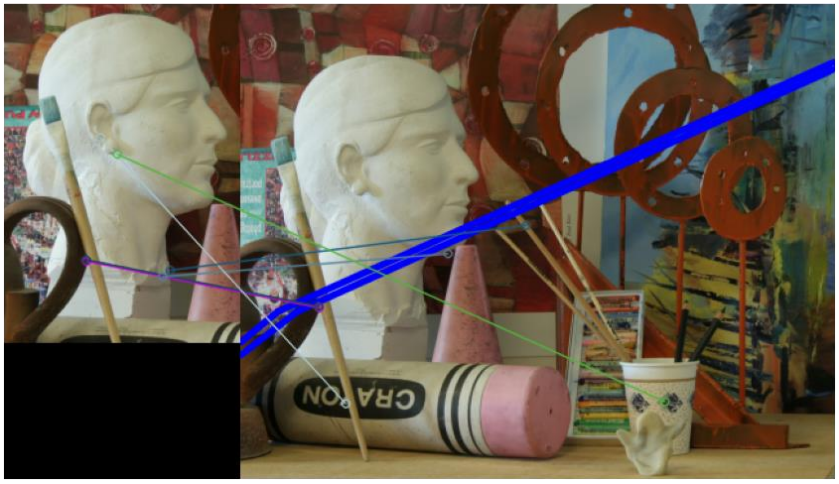

3-5. SURF - RANSAC




3-6. ORB - RANSAC, view1




3-6. ORB - RANSAC, view5



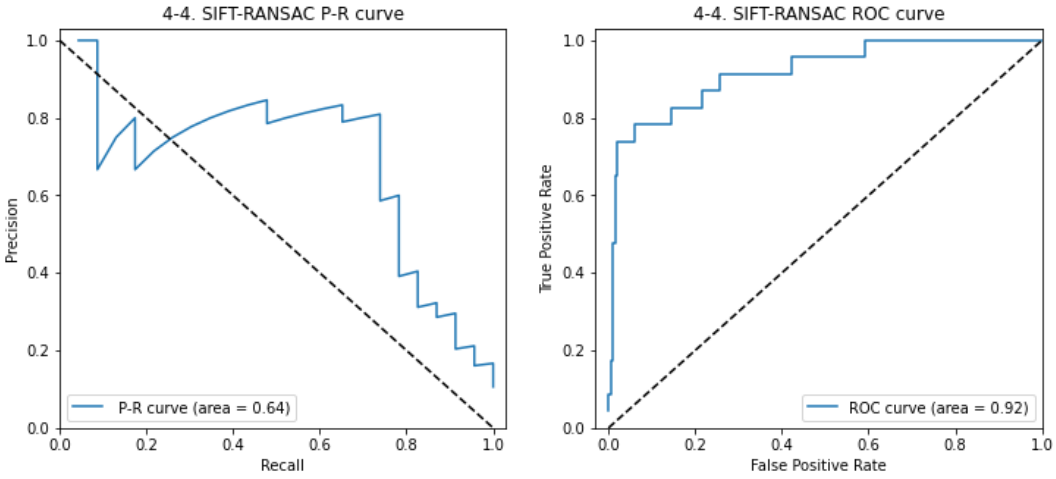
### (4) 매칭 정확도: P-R curve, ROC curve

```
def compute_tpr_fpr(mask):
    tp=[]
    fn=[]
    fp=[]
    tn=[]

    tp_cnt = 0      # 1의 개수 (상향)
    fp_cnt = 0      # 0의 개수 (상향)
    for i, m in enumerate(mask):
        if m == 1:
            tp_cnt += 1
            tp.append(tp_cnt)
            fp.append(fp_cnt)
        else:
            fp_cnt += 1
            tp.append(tp_cnt)
            fp.append(fp_cnt)

    fn_cnt = tp_cnt     # 1의 개수 (하향)
    tn_cnt = fp_cnt     # 0의 개수 (하향)
    for i, m in enumerate(mask):
        if m == 1:
            fn_cnt -= 1
            fn.append(fn_cnt)
            tn.append(tn_cnt)
        else:
            tn_cnt -= 1
            fn.append(fn_cnt)
            tn.append(tn_cnt)

    precision = [a/(a+b) for (a, b) in zip(tp, fp)]      #      tp/tp+fp
    recall = [a/(a+b) for (a, b) in zip(tp, fn)]         # tpr: tp/tp+fn
    specificity = [a/(a+b) for (a, b) in zip(tn, fp)]    # tnr: tn/tn+fp
    fpr = [1-a for a in specificity]                     # fpr: 1-specificity

    return precision, recall, fpr
```

**1. SIFT(Scale-Invariant Feature Transform):**

크기와 회전에 불변하는 특징점을 검출할 수 있다는 강력한 특징이 있다. ROC 곡선을 보면 알 수 있듯이 변형, 변환, 회전, 부분 가려짐 등 다양한 조건에서 높은 일치율을 보인다.

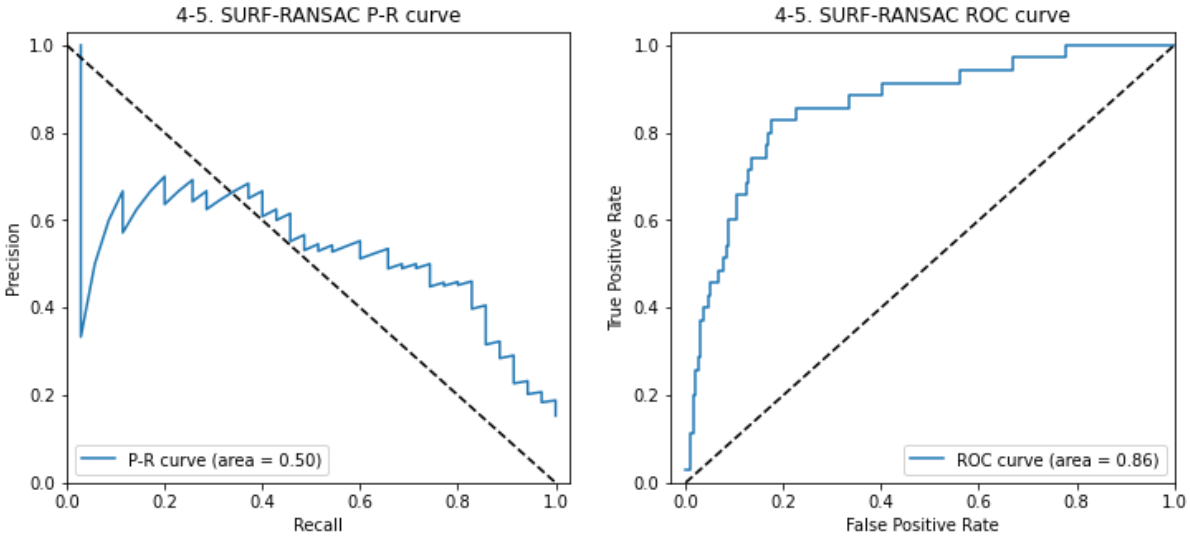
하지만 상대적으로 큰 특징점 집합을 생성하고 저장하는 메모리 요구량이 커 계산적으로 비용이 많이 드는 알고리즘이기 때문에 실시간 응용에는 제한적일 수 있다.




**2. SURF(Speeded-Up Robust Features):**

SIFT 와 마찬가지로 크기와 회전에 대해 불변성을 유지하는 특징을 갖고 있다. 또한, SIFT 보다 계산적으로 더 효율적이며, SIFT 에 비해 좀 더 빠른 특징점 검출과 매칭을 제공하기 때문에 실시간 응용에 적합하다.

하지만 SIFT 에 비해 약간의 일치율 손실이 있을 수 있고, 일부 환경에서는 성능이 SIFT 보다 약간 낮을 수 있다.



### 3. ORB(Oriented Fast and Rotated BRIEF):

다양한 플랫폼에서 사용 가능한 경량화되어 매우 빠른 계산 속도를 가진다는 장점이 있지만 상대적으로 SIFT와 SURF보다 일치율이 낮을 수 있다. 특히 큰 변형, 회전, 크기 조절에 대해서는 성능이 떨어질 수 있다.

따라서 ORB 는 작은 크기의 변형, 회전, 크기 조절에 대해서 성능을 기대할 수 있으며, 계산적인 측면에서 매우 효율적이기 때문에 SURF 와 같이 실시간 응용 및 자원 제한된 환경에서 사용하기에 적합하다.