# 영상기술특론 **HOMEWORK #2**

연세대학교 공학대학원 인공지능학과

2023450135 김연경

## 주어진 두 이미지를 이용하여 Feature matching을 실행하고, matching point를 예측

```
# Match features using the descriptors
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
matches = bf.match(descriptors1, descriptors2)

# Extract the matched keypoints
src_pts = np.float32([keypoints1[m.queryIdx].pt for m in matches]).reshape(-1, 1, 2)
dst_pts = np.float32([keypoints2[m.trainIdx].pt for m in matches]).reshape(-1, 1, 2)
```

*Feature Matching: 두 이미지에서 찾은 특징점을 서로 비교하여, 두 이미지 간의 일치하는 특징점을 찾아냅니다. 일반적으로는, 최근접 이웃(Nearest Neighbor) 알고리즘을 사용하여 가장 근접한 특징점을 찾습니다

## 위에서 예측된 Matching point를 이용하여 두 이미지의 실제 픽셀(pixel) 좌표 사이의 기하학적 관계인 fundamental matrix를 계산

```
# Compute the fundamental matrix using RANSAC
F, mask = cv2.findFundamentalMat(src_pts, dst_pts, cv2.FM_RANSAC, ransacReprojThreshold=distance_threshold, confidence=confidence, maxIters=num_trials)

# Extract the inlier matches
inlier_matches = [matches[i] for i in range(len(matches)) if mask[i]]
# Extract the outlier matches
outlier_matches = [matches[i] for i in range(len(matches)) if not mask[i]]

# Draw the inlier matches
img_inlier_matches = cv2.drawMatches(img1, keypoints1, img2, keypoints2, inlier_matches, None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
# Draw the outlier matches
img_outlier_matches = cv2.drawMatches(img1, keypoints1, img2, keypoints2, outlier_matches, None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

# Display the image with the inlier matches
cv2.imwrite('./inlier_matches.png', img_inlier_matches)
# Display the image with the outlier matches
```

cv2.imwrite('./outlier_matches.png', img_outlier_matches)

*Fundamental Matrix 계산: Fundamental Matrix는 두 이미지 간의 상대적인 위치 및 방향을 설명하는 행렬입니다. 일반적으로는 8개의 matching point를 사용하여 8개의 방정식을 구성하여 계산합니다. 이 방정식은 일반적으로 특이값 분해 (Singular Value Decomposition)을 사용하여 해결됩니다.

## 계산한 fundamental matrix의 정확도를 측정 및 분석

```
##
# Write a code that compute the accuracy of the estimated fundamental matrix
# Note that the fundamental matrix F and all pairs of corresponding points holds:
# (x')^T F x = 0. (Lecture2 p.28)

# Compute the average reprojection error
avg_error = 0
for i in range(len(src_pts)):
    x1 = np.array([src_pts[i][0][0], src_pts[i][0][1], 1])
    x2 = np.array([dst_pts[i][0][0], dst_pts[i][0][1], 1])
    error = abs(np.dot(np.dot(x2, F), x1.T))
    avg_error += error
avg_error /= len(src_pts)

# Print the average reprojection error
print('Average reprojection error:', avg_error)
```

*Fundamental Matrix 정확도 측정: 에러를 측정하여 결정됩니다.

이미지에서 찾은 모든 matching point들의 재투영 오차(Reprojection Error)를 계산하여 평균 에러를 측정합니다.

# RANSAC의 parameter 변경에 따른 fundamental matrix 정확도 분석

## *NumTrials

RANSAC에서 생성할 무작위 샘플의 수를 결정하는 매개변수

이 값을 증가시켰을 때 더 많은 샘플을 고려하므로 fundamental matrix의 정확도가 향상되었습니다. 그러나 이 값이 너무 높으면 계산 시간이 증가했습니다.

## *DistanceThreshold

inlier로 간주되는 최소 거리를 결정하는 매개변수

이 값이 낮을수록 더 많은 점들이 inlier로 인식되므로 fundamental matrix의 정확도가 높아졌습니다. 그러나 이 값이 너무 낮으면 노이즈가 많은 이미지에서는 잘못된 결과를 가져왔습니다.

## *Confidence

RANSAC에서 모델을 생성할 때 필요한 inlier 비율을 결정하는 매개변수

이 값을 높게 설정하면 fundamental matrix 정확도가 높아졌지만, 계산 시간이 늘어났습니다.

# fundamental matrix 정확도 분석에 사용한 코드 설명

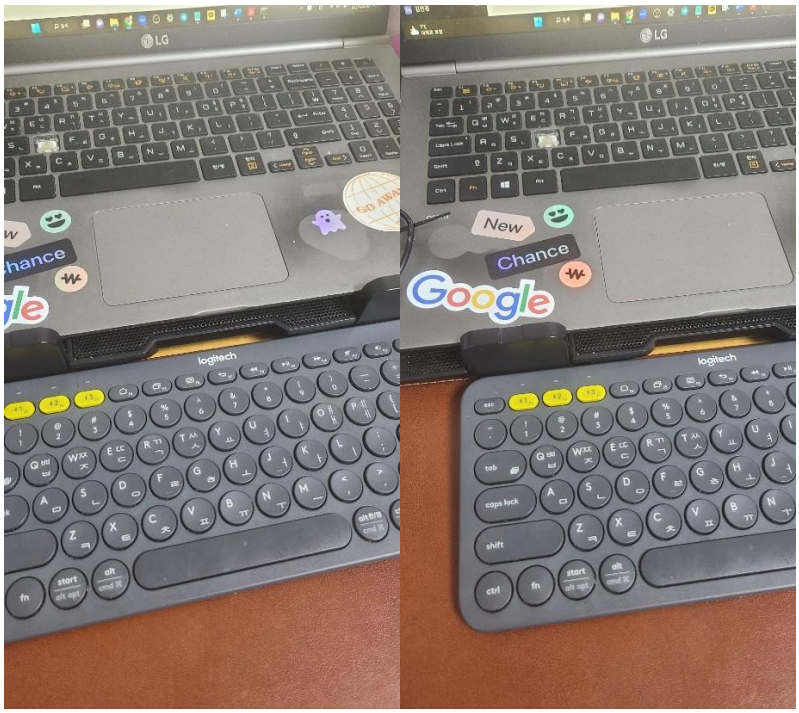
```
# Compute the average reprojection error
avg_error = 0
for i in range(len(src_pts)):
    x1 = np.array([src_pts[i][0][0], src_pts[i][0][1], 1])
    x2 = np.array([dst_pts[i][0][0], dst_pts[i][0][1], 1])
    error = abs(np.dot(np.dot(x2, F), x1.T))
    avg_error += error
avg_error /= len(src_pts)

# Print the average reprojection error
print('Average reprojection error:', avg_error)
```

위 코드에서는 추정된 fundamental matrix F와 대응하는 모든 점 쌍 (x, x')에 대해서 다음과 같은 계산을 수행합니다.

1. 대응하는 점 쌍 (x, x')를 각각 3차원 벡터로 변환합니다. 이때, x와 x' 각각의 마지막 원소는 1로 설정합니다.
2. Fundamental matrix F와 x, x'를 사용하여 다음과 같은 값(error)을 계산합니다: abs(x'^T F x)
3. 대응하는 모든 점 쌍의 error 값을 더하고, 대응하는 점 쌍의 수로 나누어 평균 error 값을 계산합니다.
4. 평균 error 값을 출력합니다.
5. 평균 error 값이 작을수록 fundamental matrix의 추정이 더 정확하다고 판단할 수 있습니다.

## 주어진 두 이미지 외 추가로 본인이 직접 촬영한 두 이미지를 이용해 실험 후 결과를 분석




```
# Display the image with the inlier matches
cv2.imwrite('./inlier_matches.png', img_inlier_matches)
# Display the image with the outlier matches
cv2.imwrite('./outlier_matches.png', img_outlier_matches)

##
# Write a code that compute the accuracy of the estimated fundamental matrix
# Note that the fundamental matrix F and all pairs of corresponding points holds:
# (x')^T F x = 0. (Lecture2 p.28)

# Compute the average reprojection error
avg_error = 0
for i in range(len(src_pts)):
    x1 = np.array([src_pts[i][0][0], src_pts[i][0][1], 1])
    x2 = np.array([dst_pts[i][0][0], dst_pts[i][0][1], 1])
    error = abs(np.dot(np.dot(x2, F), x1.T))
    avg_error += error
avg_error /= len(src_pts)

# Print the average reprojection error
print('Average reprojection error:', avg_error)
```

```
Average reprojection error: 0.3794570963245937
```

차이가 크지 않아서인지 오차가 적게 나옴