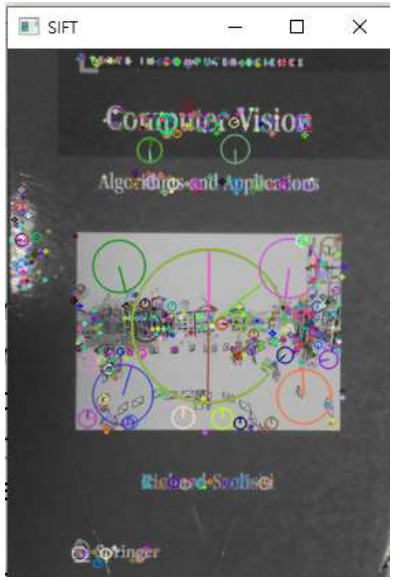
# 매칭 실습

- 물체인식

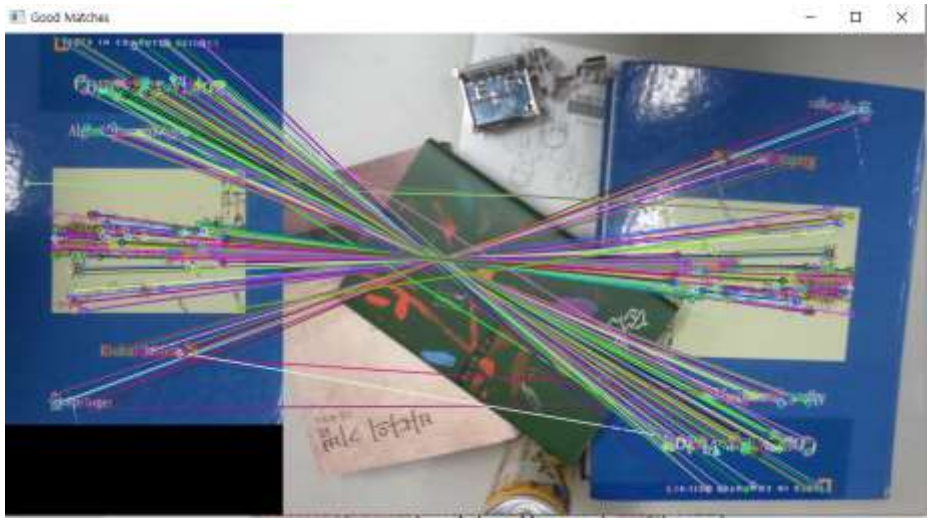
# 1. SIFT 키포인트 검출과 기술자 추출

```
import cv2 as cv
import numpy as np
#%%
img1=cv.imread('model3.jpg')
gray1=cv.cvtColor(img1, cv.COLOR_BGR2GRAY)
img2=cv.imread('scene.jpg')
gray2=cv.cvtColor(img2, cv.COLOR_BGR2GRAY)
cv.imshow("IMG!", img1);cv.imshow("IMG2", img2)
cv.waitKey(); cv.destroyAllWindows()
#%% detect and compute Keypoints
sift=cv.SIFT_create()
kp1, des1 =sift.detectAndCompute(gray1, None)
kp2, des2 =sift.detectAndCompute(gray2, None)
print("특징점 개수", len(kp1), len(kp2))
#%% Keypoints of img1
gray1 = cv.drawKeypoints(gray1, kp1, None, flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv.imshow("SIFT", gray1)
cv.waitKey(); cv.destroyAllWindows()
```



# 2. 빠른 매칭

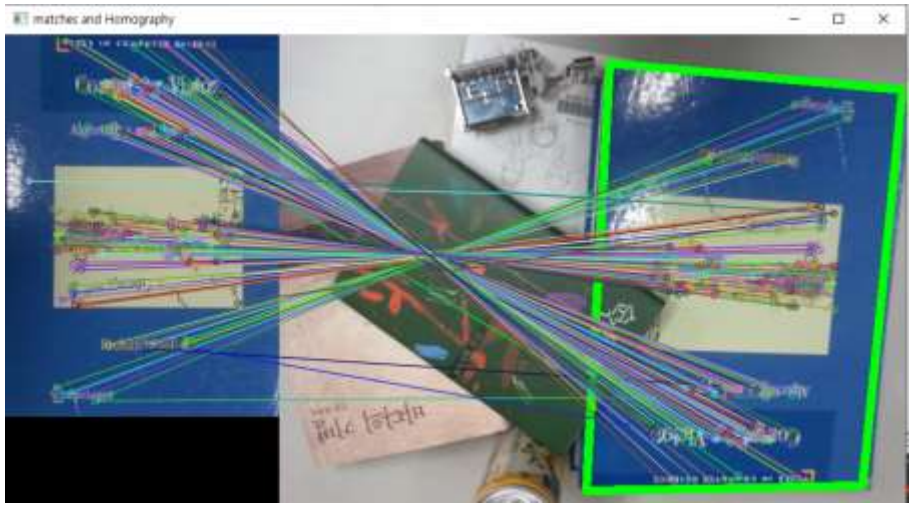
```
#%% Matches
flann_matcher=cv.DescriptorMatcher_create(cv.DescriptorMatcher_FLANNBASED)
knn_match=flann_matcher.knnMatch(des1, des2, k=2)

T=0.7
good_match=[]
for nearest1, nearest2 in knn_match:
    if(nearest1.distance/nearest2.distance) < T:
        good_match.append(nearest1)
#%%
print(len(good_match)); type(good_match)
print(good_match[0].queryIdx, good_match[0].trainIdx,good_match[0].distance, good_match[0].imgIdx)
#%% DrawMatches
img_mat=np.empty((max(img1.shape[0], img2.shape[0]), img1.shape[1]+img2.shape[1], 3), dtype=np.uint8)
cv.drawMatches(img1, kp1, img2, kp2, good_match, img_mat,
flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

cv.imshow("Good Matches", img_match_1)
cv.waitKey(); cv.destroyAllWindows()
```



# 3.변환행렬 추정과 원근변환

```
#%% Instance recognition
points1=np.float32([kp1[gm.queryIdx].pt for gm in good_match])
points2=np.float32([kp2[gm.trainIdx].pt for gm in good_match])
H, mask=cv.findHomography(points1, points2, cv.RANSAC)

h1, w1 = img1.shape[0], img1.shape[1]
h2, w2 = img2.shape[0], img2.shape[1]

box1=np.float32([[0,0], [0, h1-1], [w1-1, h1-1], [w1-1,0]]).reshape(4,1,2)
box2=cv.perspectiveTransform(box1,H)

img2=cv.polylines(img2, [np.int32(box2)], True, (0,255,0),8)

img_match=np.empty((max(h1, h2), w1+w2, 3), dtype=np.uint8)
cv.drawMatches(img1, kp1, img2, kp2, good_match, img_match,
flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

cv.imshow("matches and Homography", img_match)
cv.waitKey(); cv.destroyAllWindows()
```

```
void cv::perspectiveTransform(
    cv::InputArray    src,   // N×1 입력 배열(2 또는 3채널)
    cv::OutputArray   dst,   // N×1 출력 배열(2 또는 3채널)
    cv::InputArray    mtx    // 변환 행렬(3×3 또는 4×4)
```



# 파노라마 영상 제작

- 제작 과정

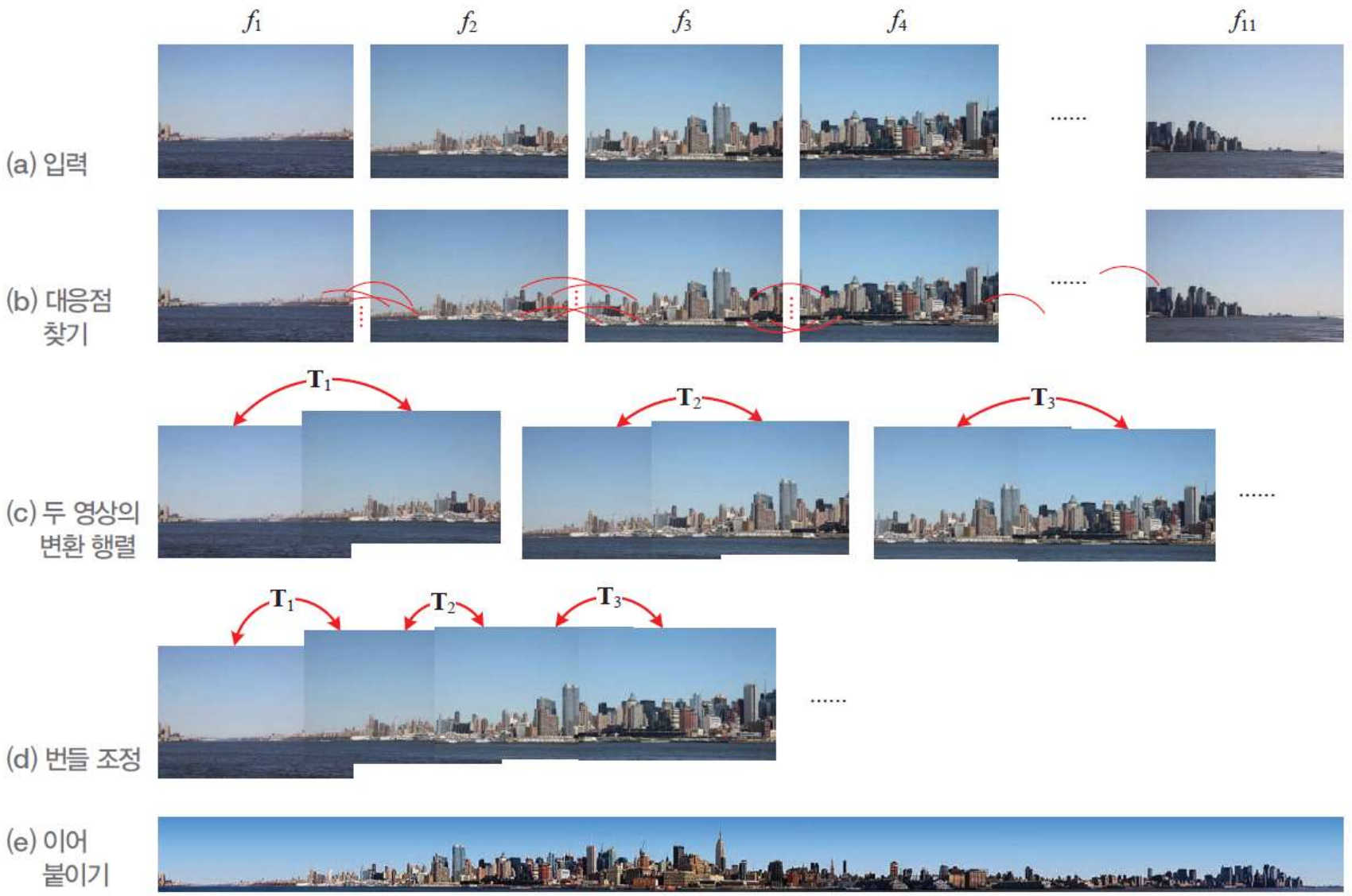



그림 7-16 파노라마 제작 과정

```
import cv2 as cv
img_files=["parliament1.bmp","parliament2.bmp", "parliament3.bmp"]
#image load
images=[]; i=1
for file in img_files:
    image=cv.imread(file)
    cv.imshow(f"Parliament{i}", image )
    cv.waitKey()
    images.append(image); i+=1
#Stitching
stitcher=cv.Stitcher_create()
status, stitched_image=stitcher.stitch(images)
if status==cv.Stitcher_OK:
    cv.imwrite("panorama.jpg", stitched_image)
    print("Panorama saved sucessfully")
else:
    print("Stitching failed")
cv.imshow("Panorama", stitched_image)
cv.waitKey(); cv.destroyAllWindows()
```

# 과제

- 자신의 모습이 들어간 영상 3장을 시점을 달리해서 찍어서 파노라마 영상을 만드시오. 영상 크기가 크면 resize()를 이용해서 영상 크기를 적절하게 줄인 후에 작업하시오. 3장의 사진을 따로 출력하고 파노라마 결과 영상을 코드와 함께 제출하시오.