프로그래밍 과제 (2)

산업 컴퓨터비전 실제

2021. 11. 24

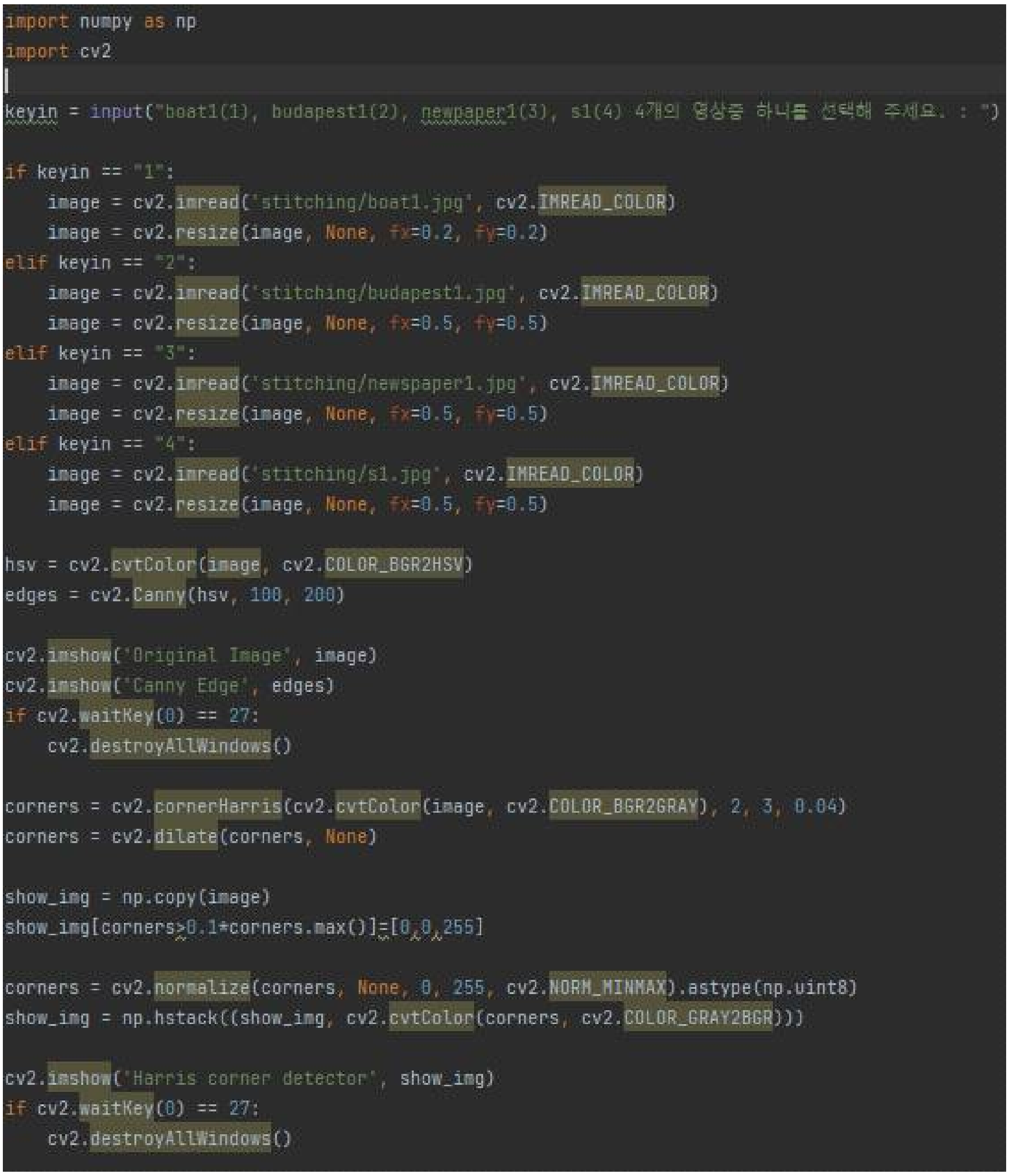
2020254014

임 동 민

I. Feature Detection

stitching.zip에서 4장의 영상(boat1, budapest1, newpaper1, s1)을 선택한 후에 Canny Edge와 Harris Corner를 검출해서 결과를 출력하는 코드를 작성하시오.

(1) Source Code



```python
import numpy as np
import cv2

keyin = input("boat1(1), budapest1(2), newpaper1(3), s1(4) 4개의 영상중 하나를 선택해 주세요. : ")

if keyin == "1":
    image = cv2.imread('stitching/boat1.jpg', cv2.IMREAD_COLOR)
    image = cv2.resize(image, None, fx=0.2, fy=0.2)
elif keyin == "2":
    image = cv2.imread('stitching/budapest1.jpg', cv2.IMREAD_COLOR)
    image = cv2.resize(image, None, fx=0.5, fy=0.5)
elif keyin == "3":
    image = cv2.imread('stitching/newspaper1.jpg', cv2.IMREAD_COLOR)
    image = cv2.resize(image, None, fx=0.5, fy=0.5)
elif keyin == "4":
    image = cv2.imread('stitching/s1.jpg', cv2.IMREAD_COLOR)
    image = cv2.resize(image, None, fx=0.5, fy=0.5)

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
edges = cv2.Canny(hsv, 100, 200)

cv2.imshow('Original Image', image)
cv2.imshow('Canny Edge', edges)
if cv2.waitKey(0) == 27:
    cv2.destroyAllWindows()

corners = cv2.cornerHarris(cv2.cvtColor(image, cv2.COLOR_BGR2GRAY), 2, 3, 0.04)
corners = cv2.dilate(corners, None)

show_img = np.copy(image)
show_img[corners>0.1*corners.max()]=[0,0,255]

corners = cv2.normalize(corners, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)
show_img = np.hstack((show_img, cv2.cvtColor(corners, cv2.COLOR_GRAY2BGR)))

cv2.imshow('Harris corner detector', show_img)
if cv2.waitKey(0) == 27:
    cv2.destroyAllWindows()
```

(2) 결과

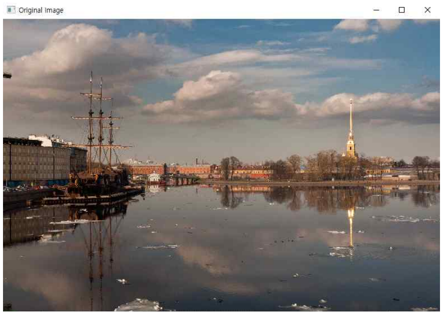
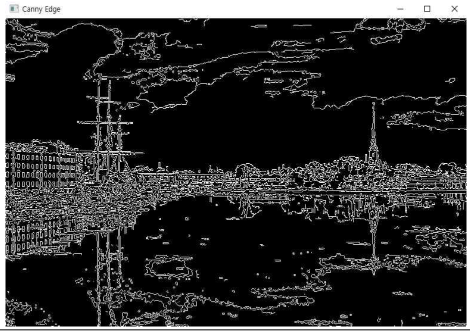
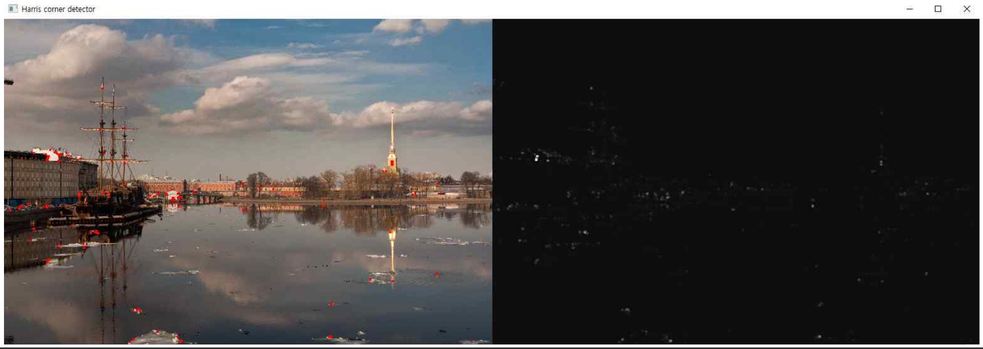
image	Original	Canny Edge
boat1		
		

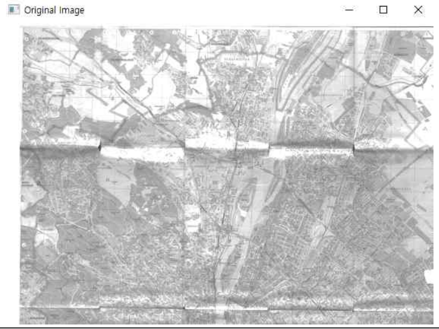
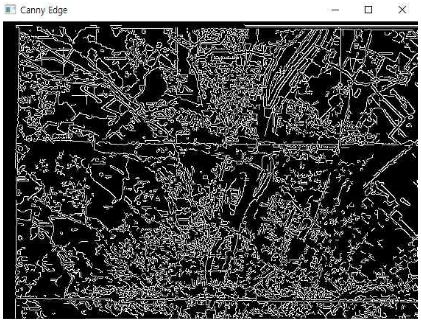
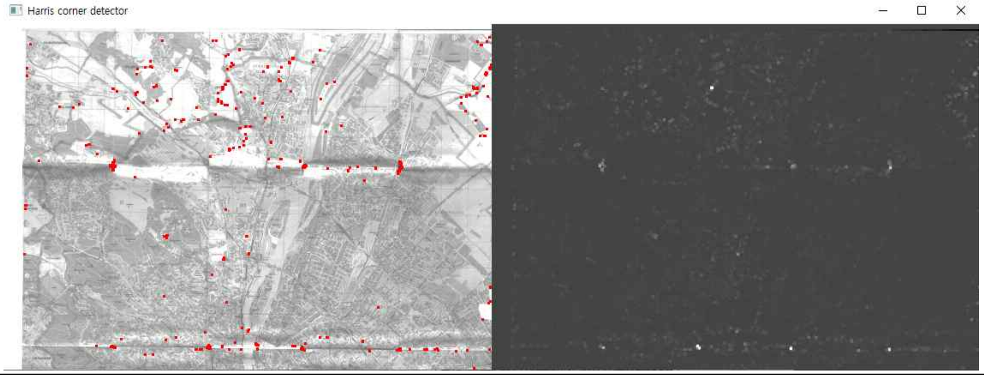
image	Original	Canny Edge
budapest1		
		

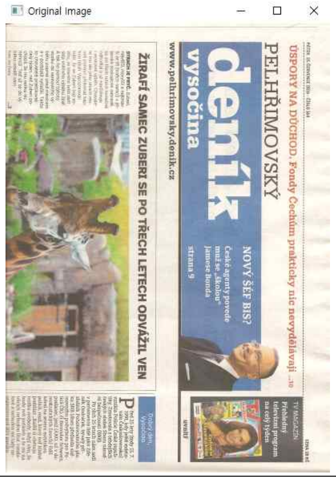
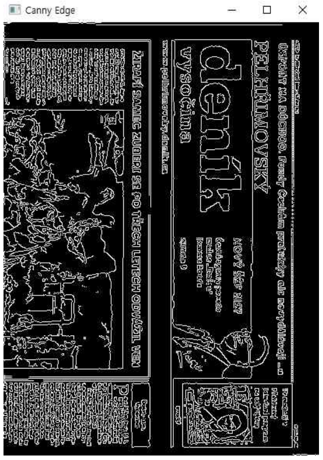
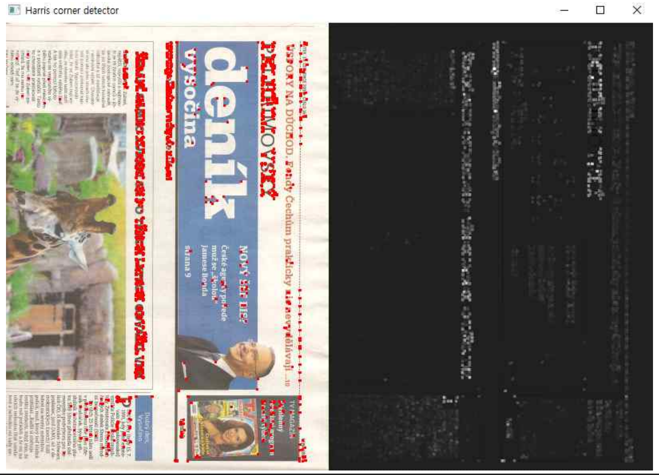
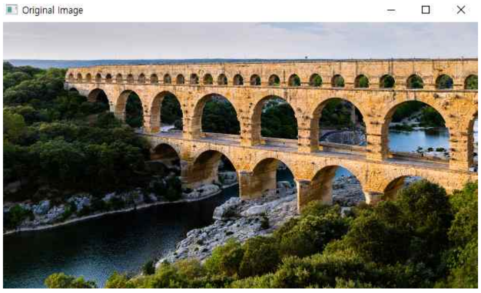
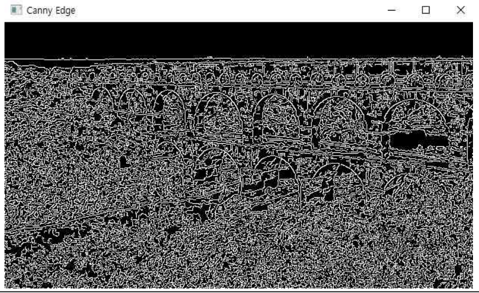
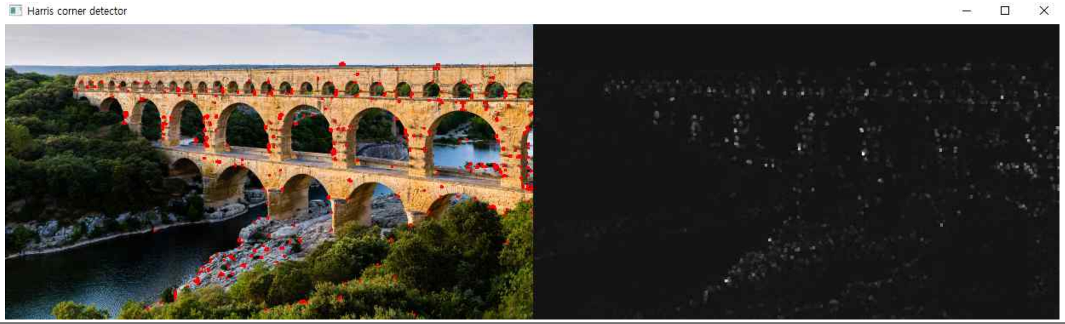
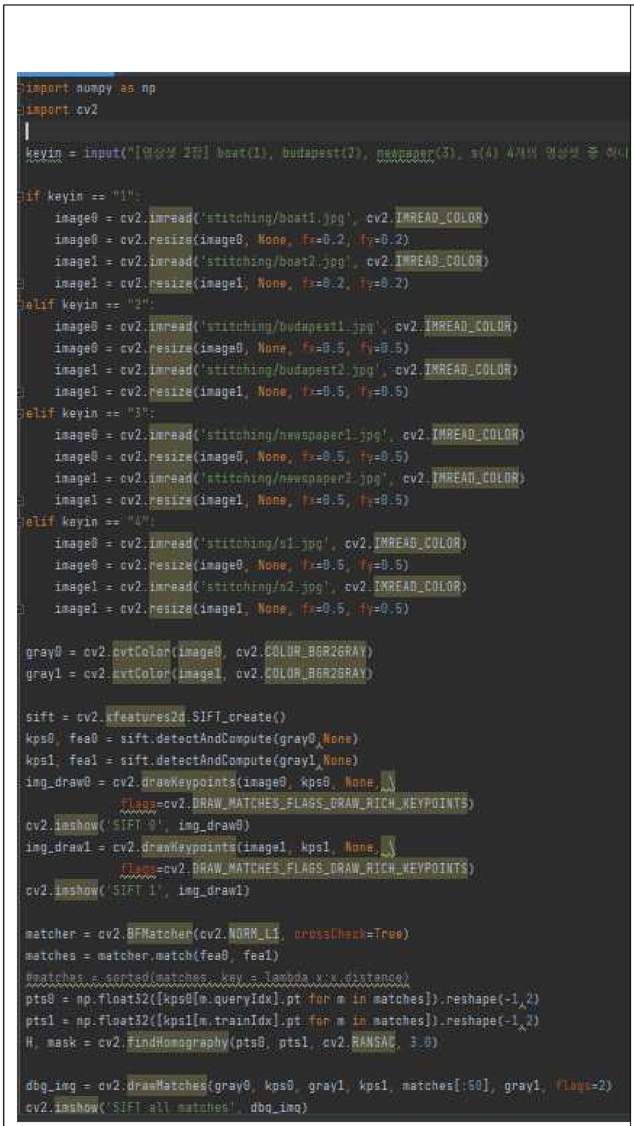
image	Original	Canny Edge
newpaper1		
		

image	Original	Canny Edge
s1		
		

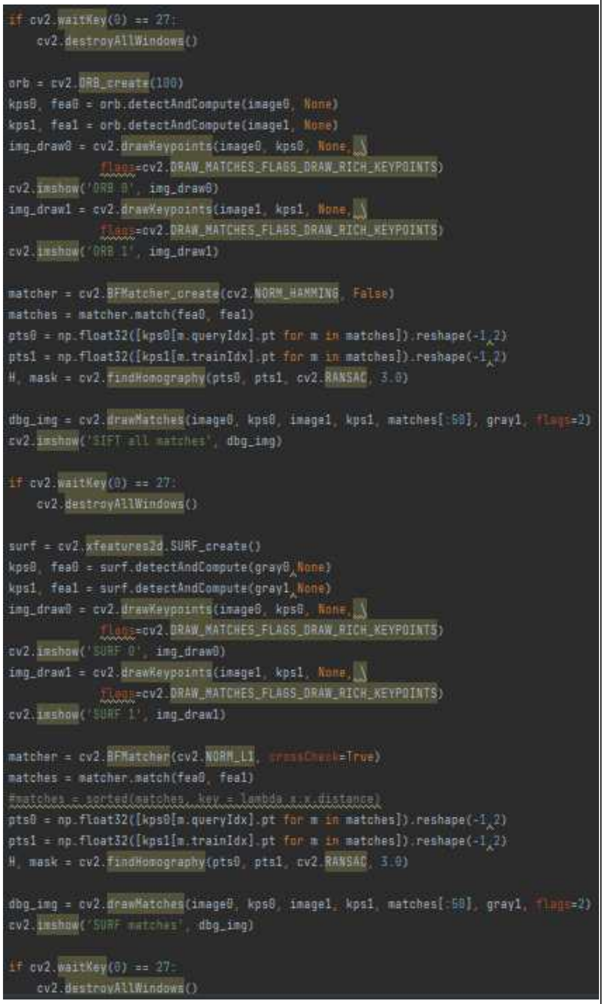
II. Matching

stitching.zip에서 각 영상셋(boat, budapest, newpaper, s1~s2)에서 두 장을 선택하고 각 영상에서 각각 SIFT, SURF, ORB를 추출한 후에 매칭 및 RANSAC을 통해서 두 장의 영상간의 homography를 계산하고, 이를 통해한 장의 영상을 다른 한 장의 영상으로 warping 하는 코드를 작성하시오. (SURF는 Lib build 이슈로 적용 못 하였습니다.)

(1) Source Code



```
import numpy as np
import cv2

keyin = input("[영상셋 2장] boat(1), budapest(2), newpaper(3), s(4) 4개의 영상셋 중 하나

if keyin == "1":
    image0 = cv2.imread('stitching/boat1.jpg', cv2.IMREAD_COLOR)
    image0 = cv2.resize(image0, None, fx=0.2, fy=0.2)
    image1 = cv2.imread('stitching/boat2.jpg', cv2.IMREAD_COLOR)
    image1 = cv2.resize(image1, None, fx=0.2, fy=0.2)
elif keyin == "2":
    image0 = cv2.imread('stitching/budapest1.jpg', cv2.IMREAD_COLOR)
    image0 = cv2.resize(image0, None, fx=0.5, fy=0.5)
    image1 = cv2.imread('stitching/budapest2.jpg', cv2.IMREAD_COLOR)
    image1 = cv2.resize(image1, None, fx=0.5, fy=0.5)
elif keyin == "3":
    image0 = cv2.imread('stitching/newspaper1.jpg', cv2.IMREAD_COLOR)
    image0 = cv2.resize(image0, None, fx=0.5, fy=0.5)
    image1 = cv2.imread('stitching/newspaper2.jpg', cv2.IMREAD_COLOR)
    image1 = cv2.resize(image1, None, fx=0.5, fy=0.5)
elif keyin == "4":
    image0 = cv2.imread('stitching/s1.jpg', cv2.IMREAD_COLOR)
    image0 = cv2.resize(image0, None, fx=0.5, fy=0.5)
    image1 = cv2.imread('stitching/s2.jpg', cv2.IMREAD_COLOR)
    image1 = cv2.resize(image1, None, fx=0.5, fy=0.5)

gray0 = cv2.cvtColor(image0, cv2.COLOR_BGR2GRAY)
gray1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

sift = cv2.xfeatures2d.SIFT_create()
kps0, fea0 = sift.detectAndCompute(gray0,None)
kps1, fea1 = sift.detectAndCompute(gray1,None)
img_draw0 = cv2.drawKeypoints(image0, kps0, None, \
            flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow('SIFT 0', img_draw0)
img_draw1 = cv2.drawKeypoints(image1, kps1, None, \
            flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow('SIFT 1', img_draw1)

matcher = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
matches = matcher.match(fea0, fea1)
#matches = sorted(matches, key = lambda x:x.distance)
pts0 = np.float32([kps0[m.queryIdx].pt for m in matches]).reshape(-1,2)
pts1 = np.float32([kps1[m.trainIdx].pt for m in matches]).reshape(-1,2)
H, mask = cv2.findHomography(pts0, pts1, cv2.RANSAC, 3.0)

dbg_img = cv2.drawMatches(gray0, kps0, gray1, kps1, matches[:50], gray1, flags=2)
cv2.imshow('SIFT all matches', dbg_img)
```



```
if cv2.waitKey(0) == 27:
    cv2.destroyAllWindows()

orb = cv2.ORB_create(100)
kps0, fea0 = orb.detectAndCompute(image0, None)
kps1, fea1 = orb.detectAndCompute(image1, None)
img_draw0 = cv2.drawKeypoints(image0, kps0, None, \
            flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow('ORB 0', img_draw0)
img_draw1 = cv2.drawKeypoints(image1, kps1, None, \
            flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow('ORB 1', img_draw1)

matcher = cv2.BFMatcher_create(cv2.NORM_HAMMING, False)
matches = matcher.match(fea0, fea1)
pts0 = np.float32([kps0[m.queryIdx].pt for m in matches]).reshape(-1,2)
pts1 = np.float32([kps1[m.trainIdx].pt for m in matches]).reshape(-1,2)
H, mask = cv2.findHomography(pts0, pts1, cv2.RANSAC, 3.0)

dbg_img = cv2.drawMatches(image0, kps0, image1, kps1, matches[:50], gray1, flags=2)
cv2.imshow('SIFT all matches', dbg_img)

if cv2.waitKey(0) == 27:
    cv2.destroyAllWindows()

surf = cv2.xfeatures2d.SURF_create()
kps0, fea0 = surf.detectAndCompute(gray0,None)
kps1, fea1 = surf.detectAndCompute(gray1,None)
img_draw0 = cv2.drawKeypoints(image0, kps0, None, \
            flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow('SURF 0', img_draw0)
img_draw1 = cv2.drawKeypoints(image1, kps1, None, \
            flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow('SURF 1', img_draw1)

matcher = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
matches = matcher.match(fea0, fea1)
#matches = sorted(matches, key = lambda x:x.distance)
pts0 = np.float32([kps0[m.queryIdx].pt for m in matches]).reshape(-1,2)
pts1 = np.float32([kps1[m.trainIdx].pt for m in matches]).reshape(-1,2)
H, mask = cv2.findHomography(pts0, pts1, cv2.RANSAC, 3.0)

dbg_img = cv2.drawMatches(image0, kps0, image1, kps1, matches[:50], gray1, flags=2)
cv2.imshow('SURF matches', dbg_img)

if cv2.waitKey(0) == 27:
    cv2.destroyAllWindows()
```

(2) 결과

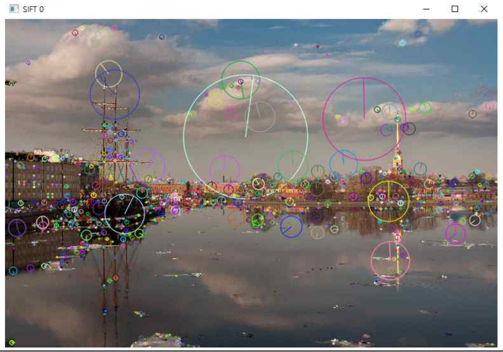
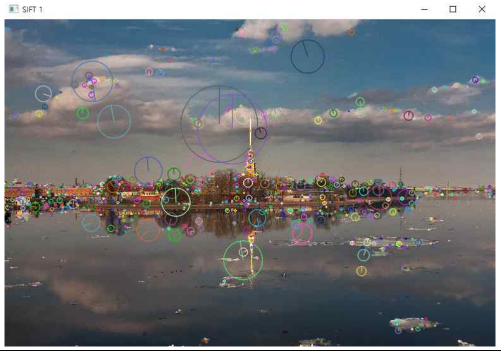
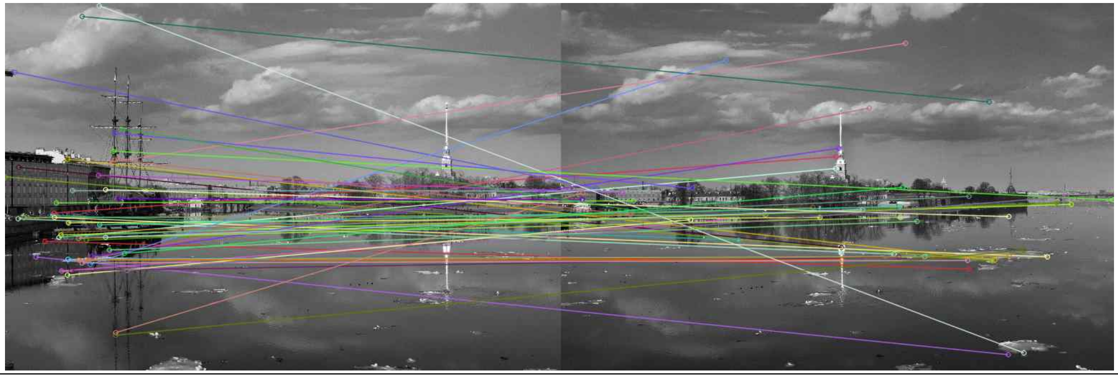
image	SIFT 0	SIFT 1
boat		
		

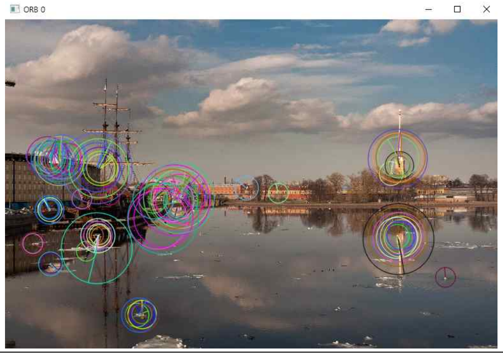
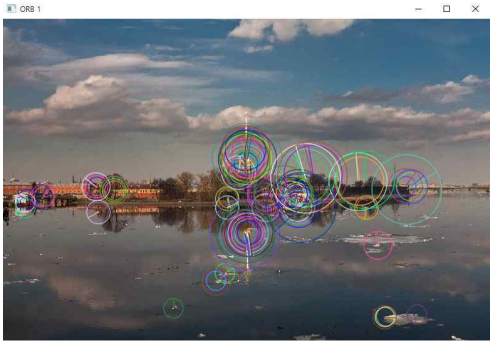
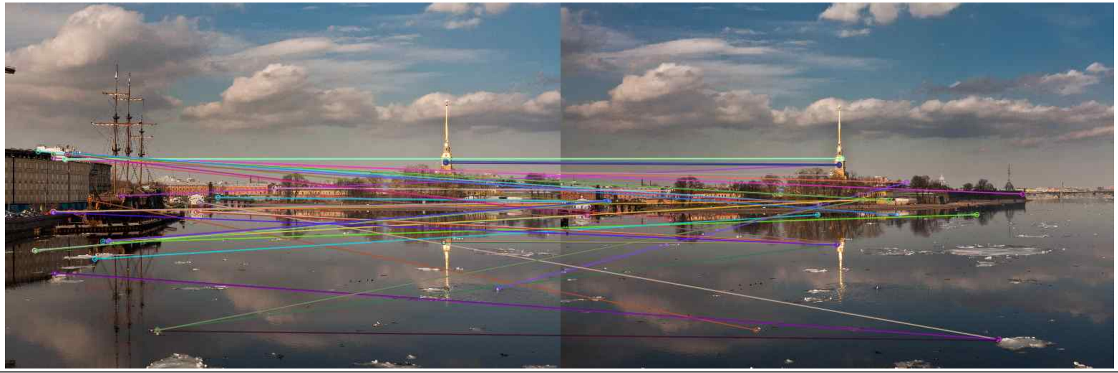
image	ORB 0	ORB 1
boat		
		

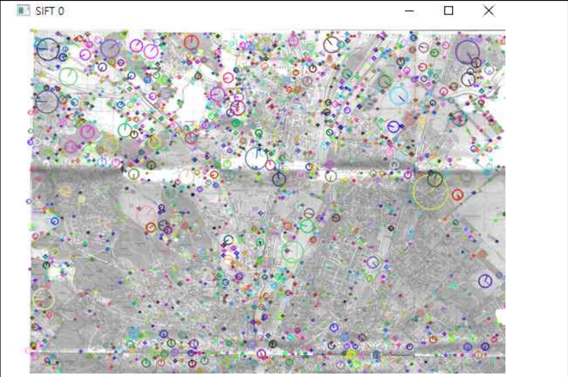
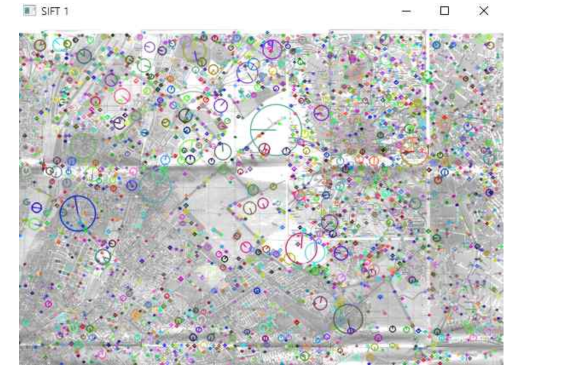
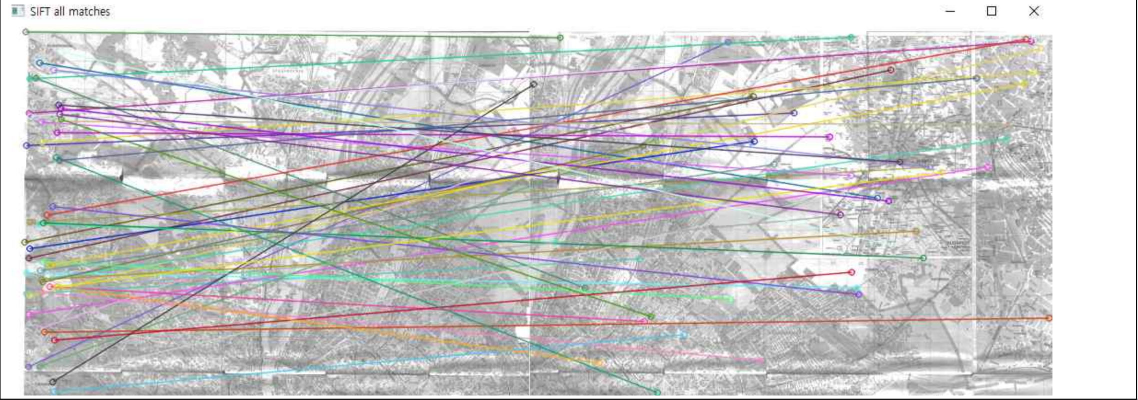
image	SIFT 0	SIFT 1
budapest		
		

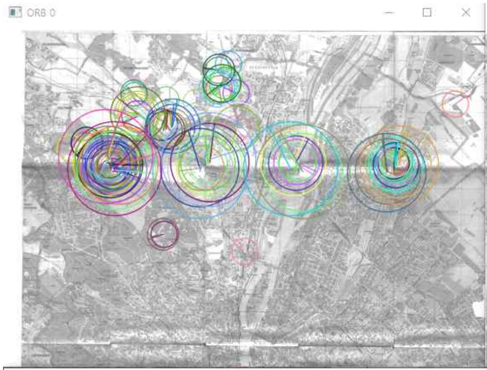
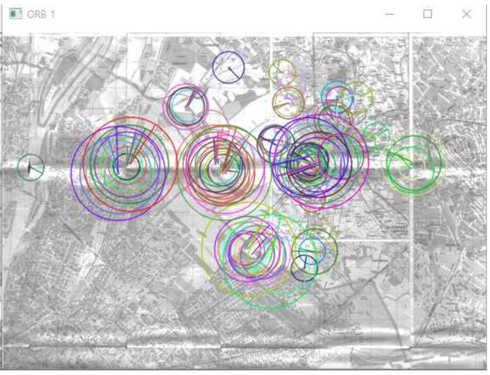
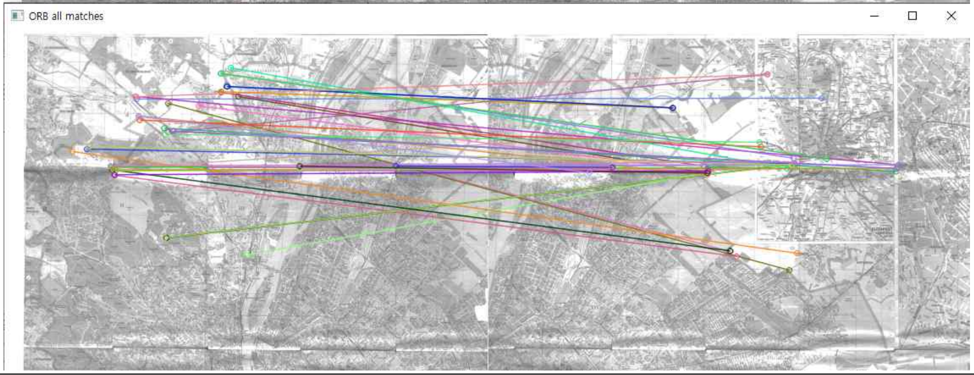
image	ORB 0	ORB 1
budapest		
		

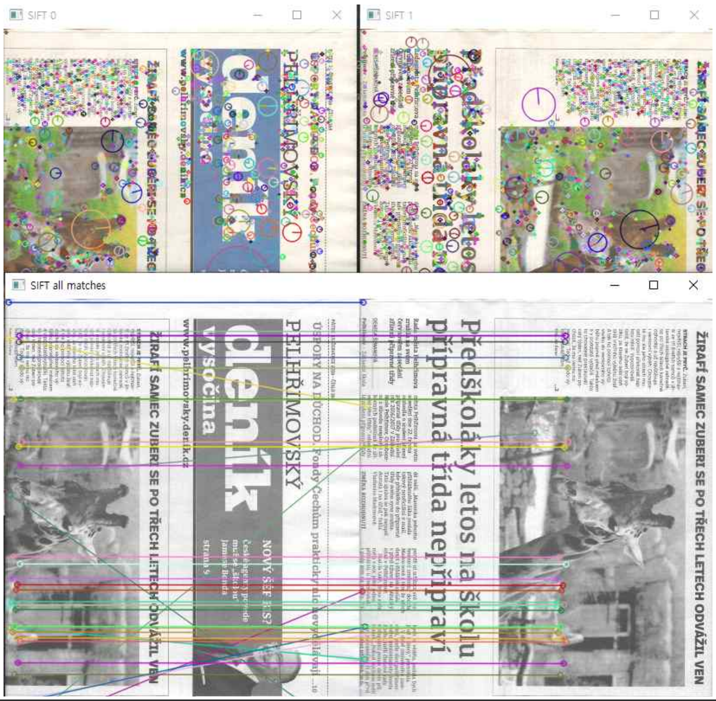
image	SIFT 0	SIFT 1
newpaper		

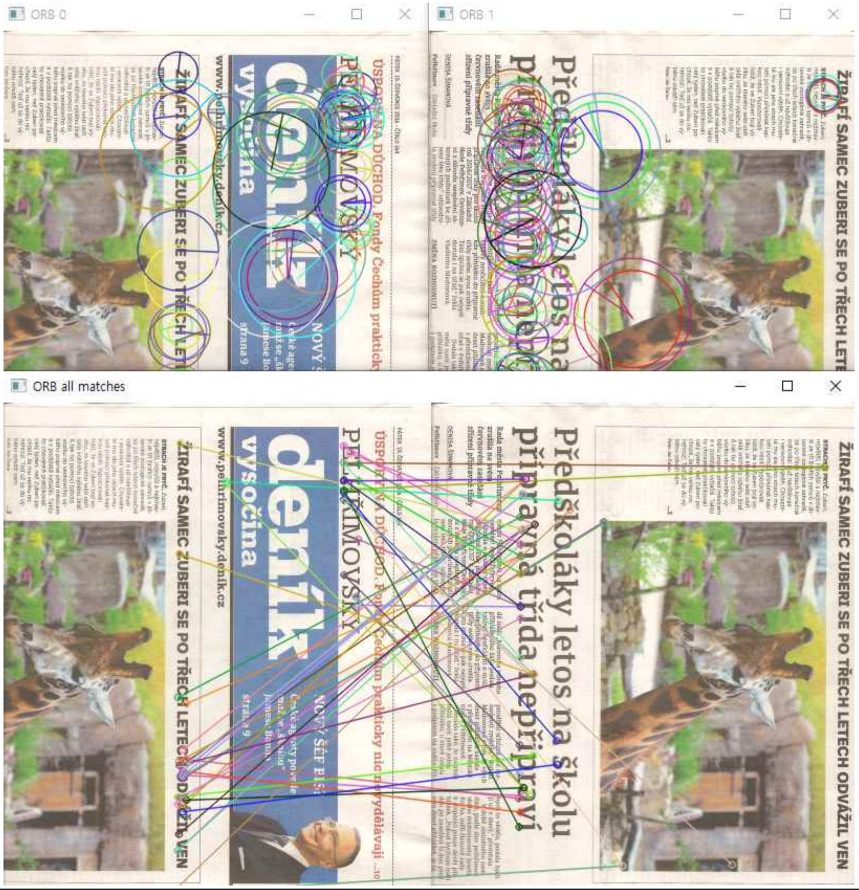
image	ORB 0	ORB 1
newpaper		

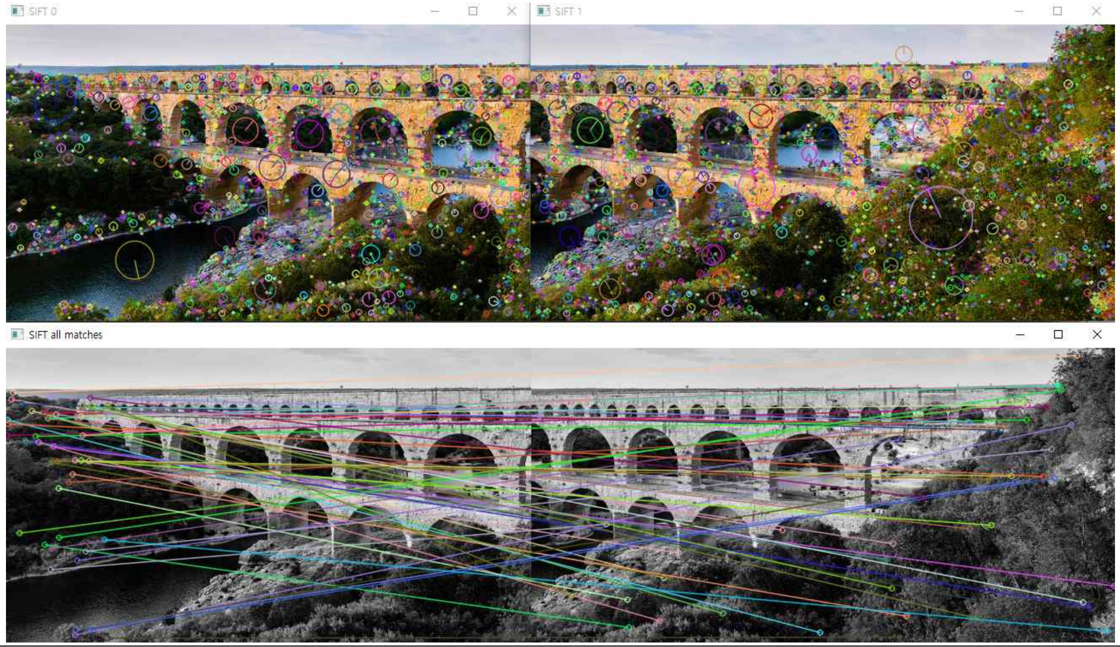
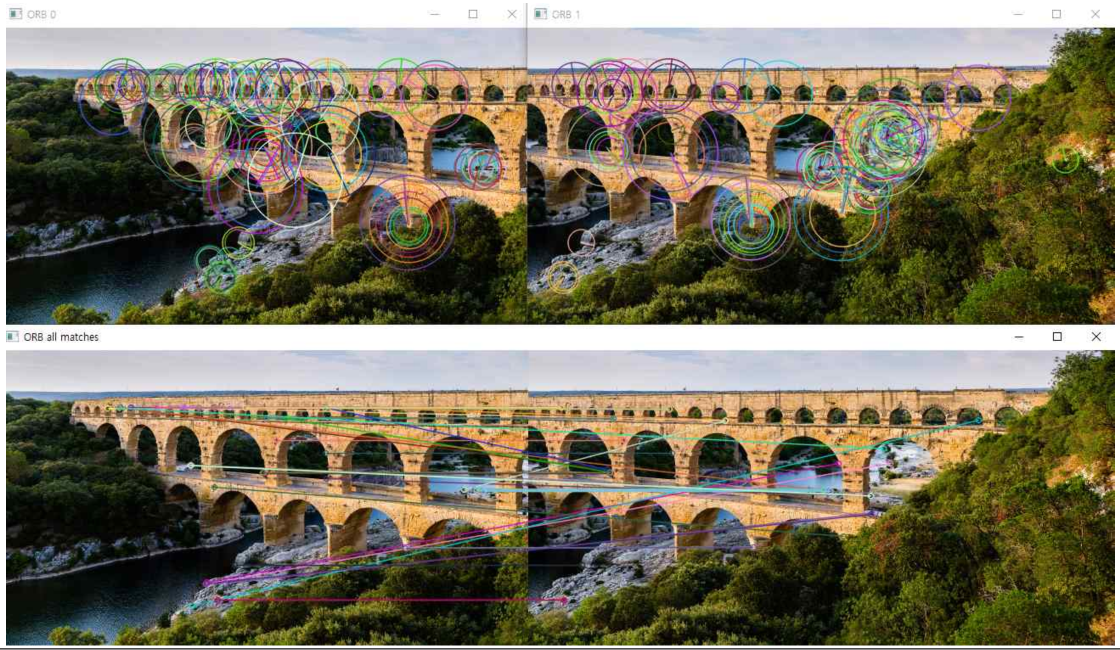
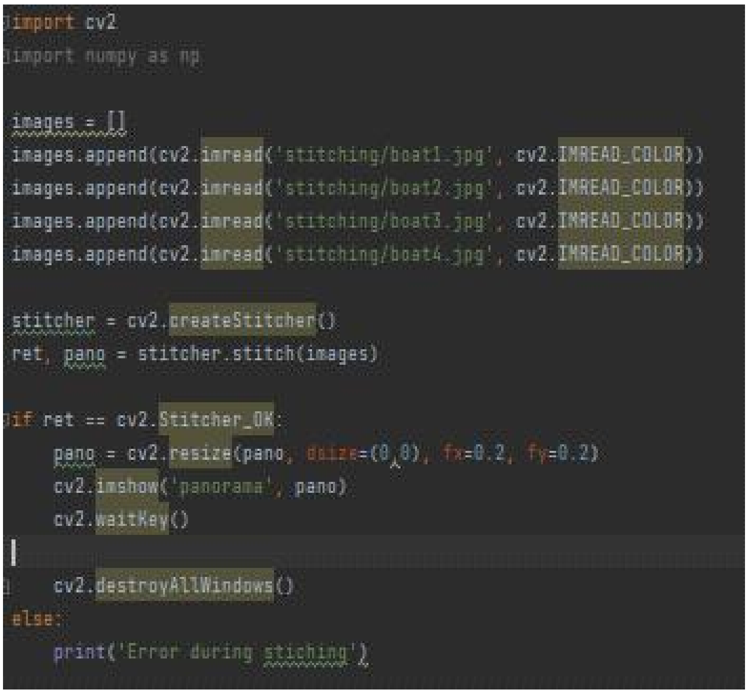
image	SIFT 0	SIFT 1
s	 	

image	SIFT 0	SIFT 1
s	 	

III. Panorama

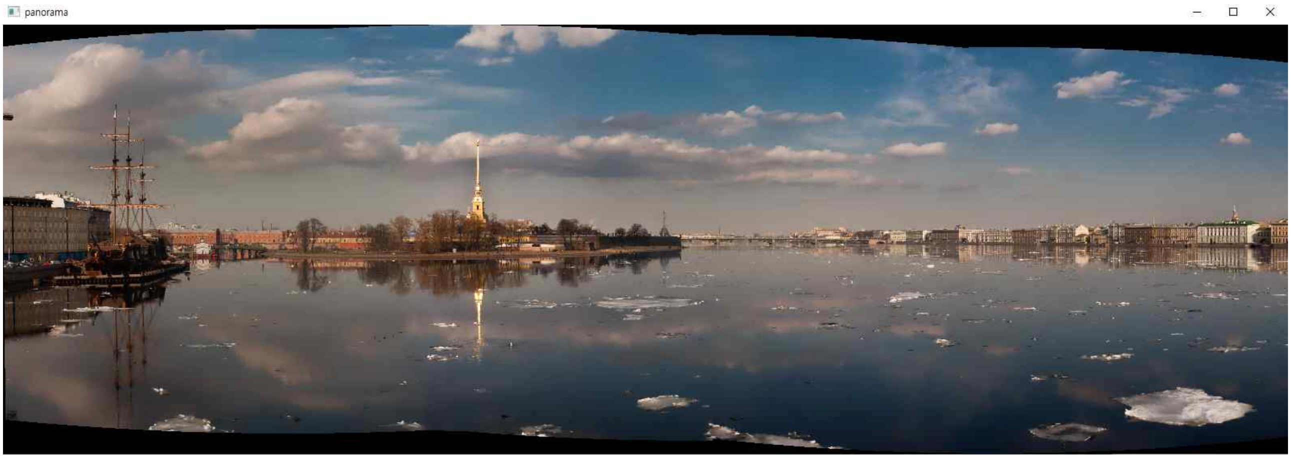
CreaterStitcher 함수를 이용하여 4개의 영상 셋에 대해서 파노라마 이미지를 만드는 방법을 구현하시오.

(1) Source Code



```
import cv2
import numpy as np


images = []
images.append(cv2.imread('stitching/boat1.jpg', cv2.IMREAD_COLOR))
images.append(cv2.imread('stitching/boat2.jpg', cv2.IMREAD_COLOR))
images.append(cv2.imread('stitching/boat3.jpg', cv2.IMREAD_COLOR))
images.append(cv2.imread('stitching/boat4.jpg', cv2.IMREAD_COLOR))

stitcher = cv2.createStitcher()
ret, pano = stitcher.stitch(images)

if ret == cv2.Stitcher_OK:
    pano = cv2.resize(pano, dsize=(0,0), fx=0.2, fy=0.2)
    cv2.imshow('panorama', pano)
    cv2.waitKey()

    cv2.destroyAllWindows()
else:
    print('Error during stiching')
```

(2) 결과

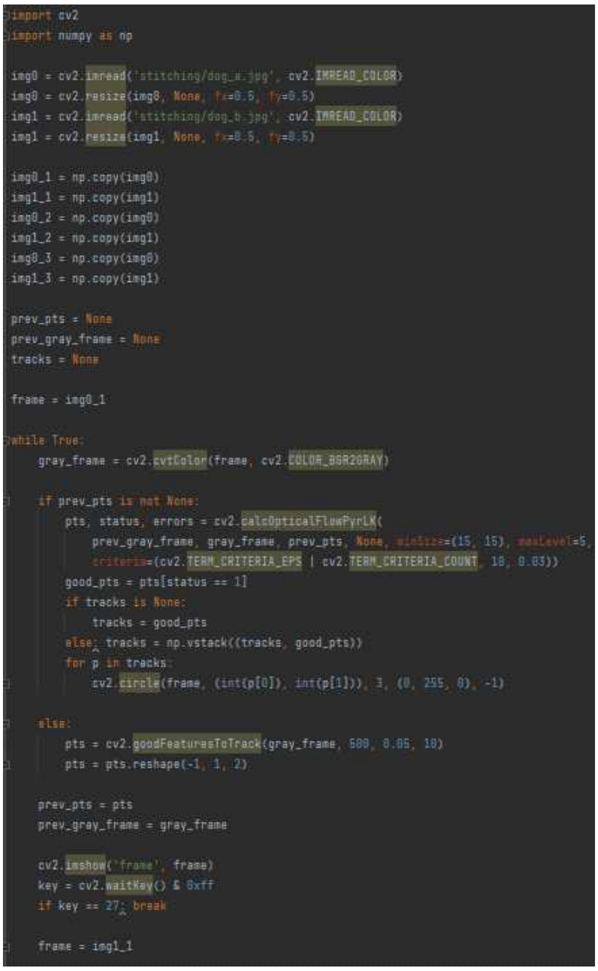


IV. Optical Flow

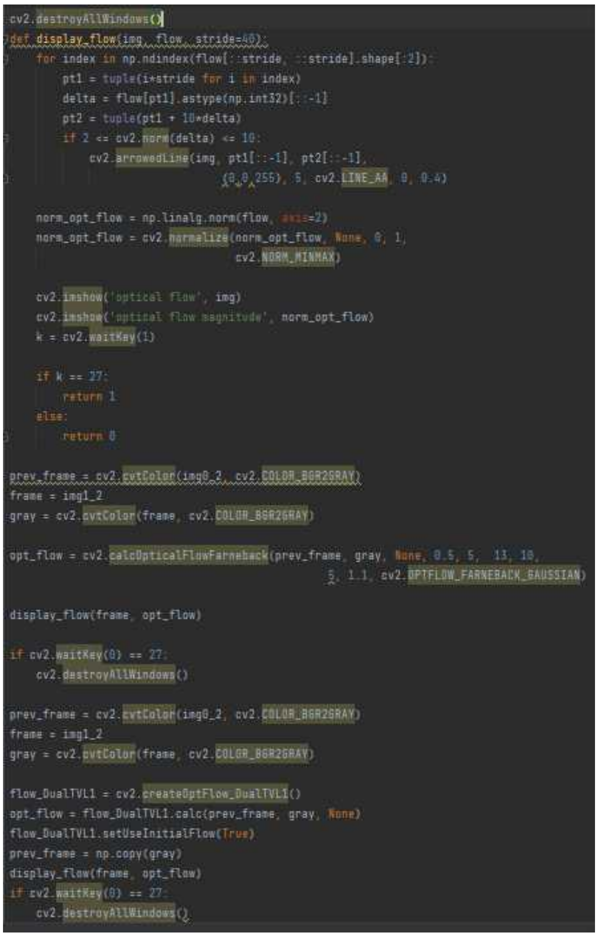
(1) stitching.zip에서 dog_a, dog_b 두 사진을 이용해서 Good Feature to Tracking을 추출하고 Pyramid Lucas-Kanade 알고리즘을 적용해서 Optical Flow를 구하는 코드를 작성하시오.

(2) stitching.zip에서 dog_a, dog_b 두 사진을 이용해서 Farneback과 DualTVL1 Optical Flow 알고리즘을 구하는 코드를 작성하시오.

(1) Source Code



```python
import cv2
import numpy as np

img0 = cv2.imread('stitching/dog_a.jpg', cv2.IMREAD_COLOR)
img0 = cv2.resize(img0, None, fx=0.5, fy=0.5)
img1 = cv2.imread('stitching/dog_b.jpg', cv2.IMREAD_COLOR)
img1 = cv2.resize(img1, None, fx=0.5, fy=0.5)

img0_1 = np.copy(img0)
img1_1 = np.copy(img1)
img0_2 = np.copy(img0)
img1_2 = np.copy(img1)
img0_3 = np.copy(img0)
img1_3 = np.copy(img1)

prev_pts = None
prev_gray_frame = None
tracks = None

frame = img0_1

while True:
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    if prev_pts is not None:
        pts, status, errors = cv2.calcOpticalFlowPyrLK(
            prev_gray_frame, gray_frame, prev_pts, None, winSize=(15, 15), maxLevel=5,
            criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))
        good_pts = pts[status == 1]
        if tracks is None:
            tracks = good_pts
        else: tracks = np.vstack((tracks, good_pts))
        for p in tracks:
            cv2.circle(frame, (int(p[0]), int(p[1])), 3, (0, 255, 0), -1)

    else:
        pts = cv2.goodFeaturesToTrack(gray_frame, 500, 0.05, 10)
        pts = pts.reshape(-1, 1, 2)

    prev_pts = pts
    prev_gray_frame = gray_frame

    cv2.imshow('frame', frame)
    key = cv2.waitKey() & 0xff
    if key == 27: break

    frame = img1_1
```

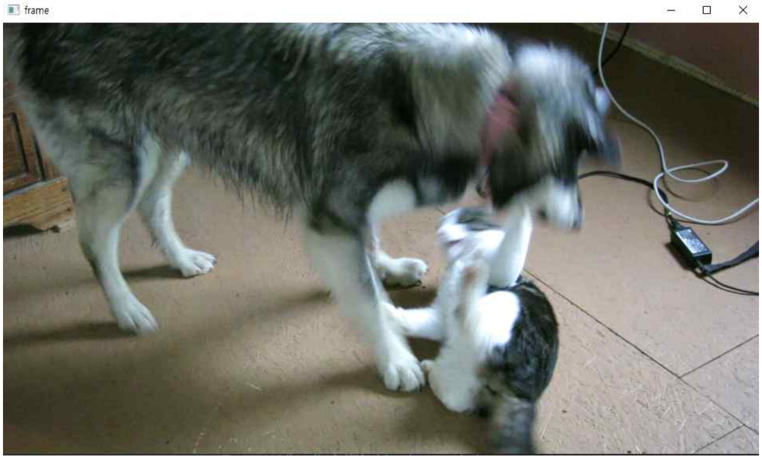


```python
cv2.destroyAllWindows()
def display_flow(img, flow, stride=40):
    for index in np.ndindex(flow[::stride, ::stride].shape[:2]):
        pt1 = tuple(i*stride for i in index)
        delta = flow[pt1].astype(np.int32)[::-1]
        pt2 = tuple(pt1 + 10*delta)
        if 2 <= cv2.norm(delta) <= 10:
            cv2.arrowedLine(img, pt1[::-1], pt2[::-1],
                            (0,0,255), 5, cv2.LINE_AA, 0, 0.4)

    norm_opt_flow = np.linalg.norm(flow, axis=2)
    norm_opt_flow = cv2.normalize(norm_opt_flow, None, 0, 1,
                                  cv2.NORM_MINMAX)

    cv2.imshow('optical flow', img)
    cv2.imshow('optical flow magnitude', norm_opt_flow)
    k = cv2.waitKey(1)

    if k == 27:
        return 1
    else:
        return 0

prev_frame = cv2.cvtColor(img0_2, cv2.COLOR_BGR2GRAY)
frame = img1_2
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

opt_flow = cv2.calcOpticalFlowFarneback(prev_frame, gray, None, 0.5, 5,  13, 10,
                                        5, 1.1, cv2.OPTFLOW_FARNEBACK_GAUSSIAN)

display_flow(frame, opt_flow)

if cv2.waitKey(0) == 27:
    cv2.destroyAllWindows()

prev_frame = cv2.cvtColor(img0_2, cv2.COLOR_BGR2GRAY)
frame = img1_2
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

flow_DualTVL1 = cv2.createOptFlow_DualTVL1()
opt_flow = flow_DualTVL1.calc(prev_frame, gray, None)
flow_DualTVL1.setUseInitialFlow(True)
prev_frame = np.copy(gray)
display_flow(frame, opt_flow)
if cv2.waitKey(0) == 27:
    cv2.destroyAllWindows()
```

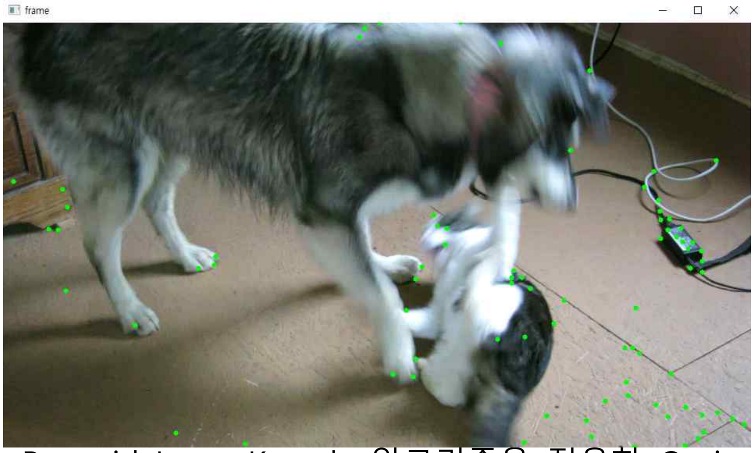
(2) 결과

(1)의 결과

dog_a,



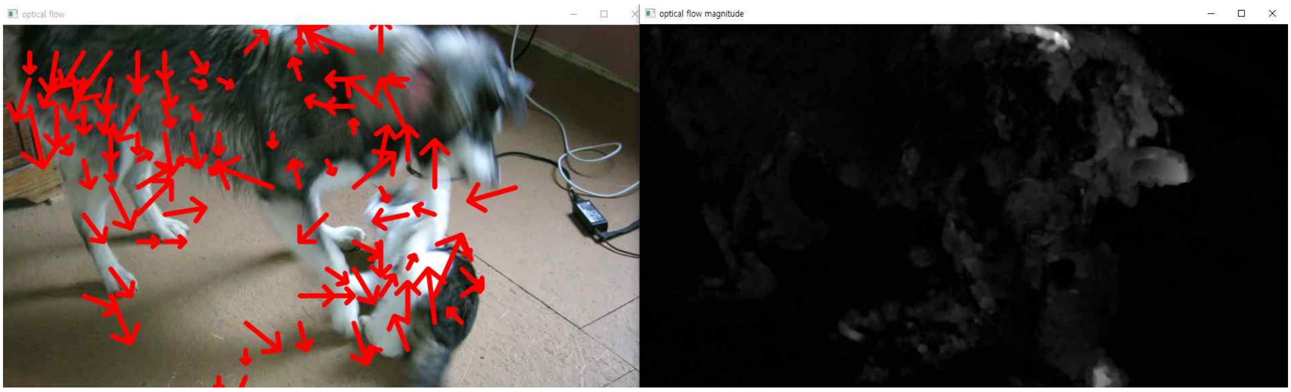
Good Feature to Tracking



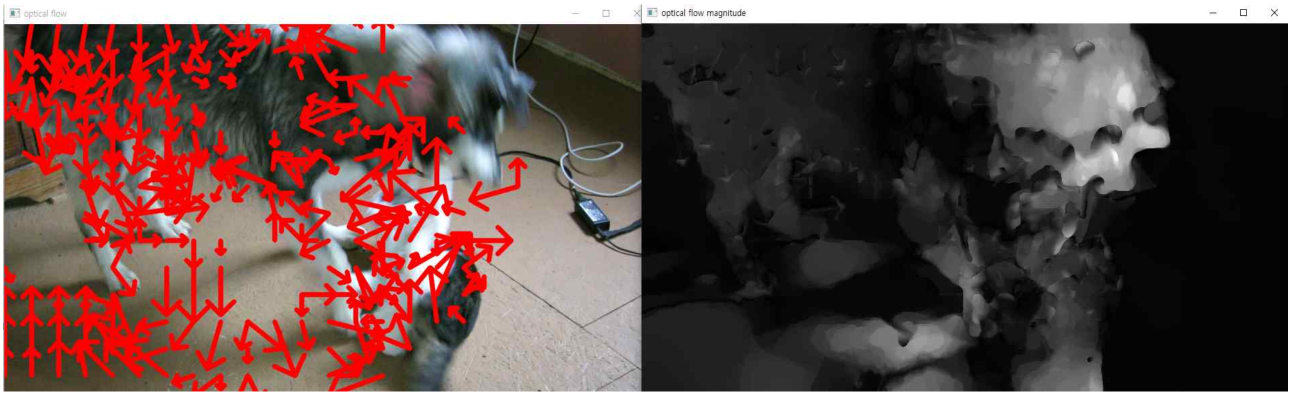
Pyramid Lucas-Kanade 알고리즘을 적용한 Optical Flow

(2)의 결과

Farneback Optical Flow 알고리즘을 구현



DualTVL1 Optical Flow 알고리즘을 구현


optical flow
optical flow magnitude