파이썬 과제물

(이미지 매칭 및 스티칭)

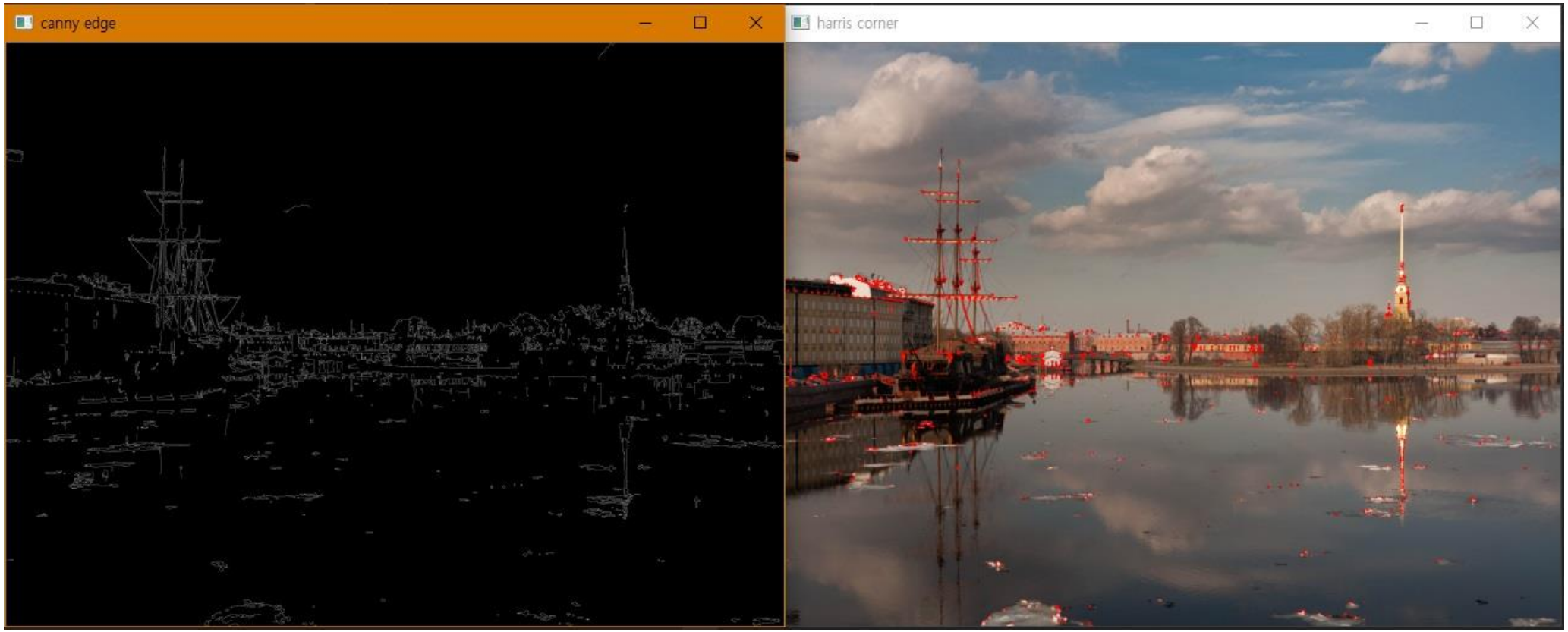
2020. 11. 30

1. 코딩 Coding : Canny Edge와 Harris Corner를 검출

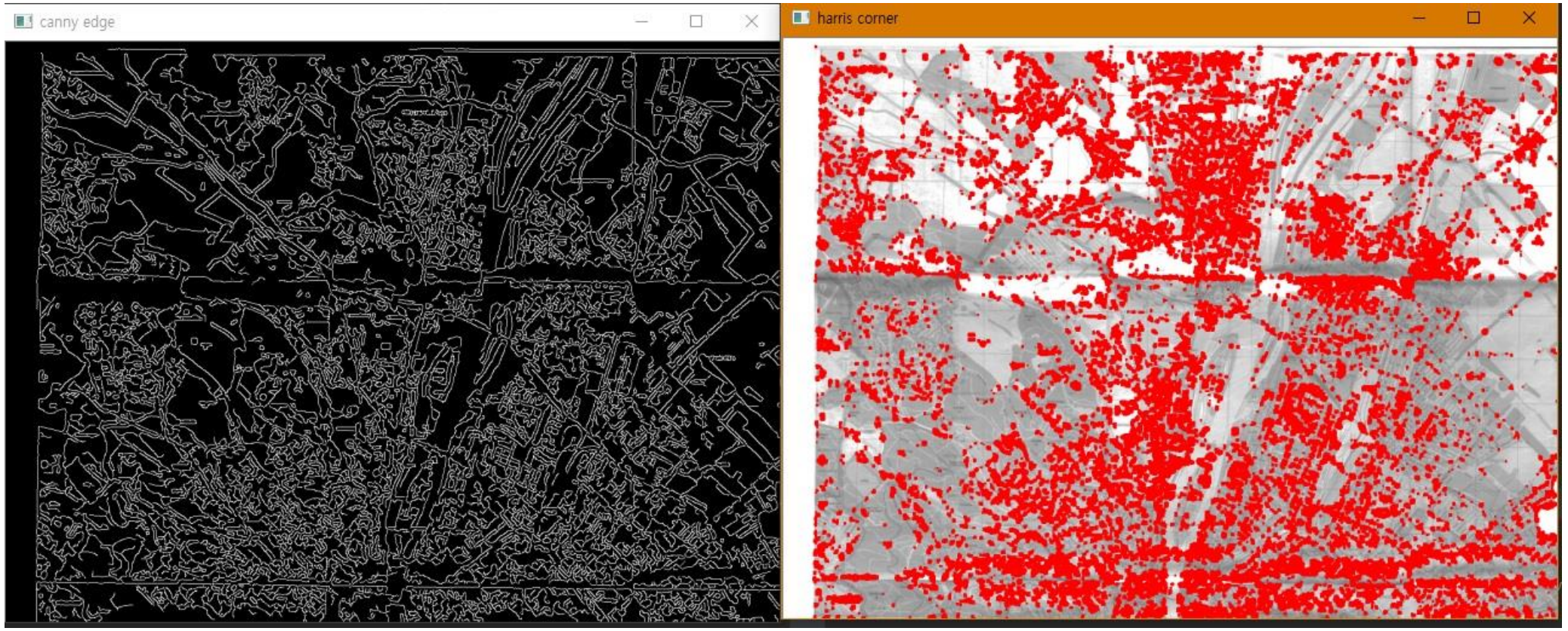
```python
import ...


def histogram_equilization(image):
    enhanced_image = cv2.equalizeHist(image)
    #cv2.imshow("histogram equilization", enhanced_image)
    return enhanced_image


def feature_detector(image):
    # Convert the image color to grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    # Use histogram equilization
    enhanced_grey = histogram_equilization(gray_image)
    # Reduce noise from the image
    blur = cv2.GaussianBlur(enhanced_grey, (9, 9), 0)
    canny = cv2.Canny(blur, 50, 170)
    dst = cv2.resize(canny, dsize=(640, 480), interpolation=cv2.INTER_AREA)
    cv2.imshow("canny edge", dst)
    dst = cv2.cornerHarris(gray_image, 5, 3, 0.04)
    # Non maximal suppression
    dst = cv2.dilate(dst, None)
    image[dst > 0.01 * dst.max()] = [0, 0, 255]
    dst1 = cv2.resize(image, dsize=(640, 480), interpolation=cv2.INTER_AREA)
    cv2.imshow('harris corner', dst1)
```

```python
choice = input("Please input image choices(no.): \n 1. BOAT\n 2. BUDAPEST\n 3. NEWSPAPER\n 4. SCENE\n ")

if (choice == '1'):
    print('Fetching boat images')
    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))
elif (choice == '2'):
    print('Fetching budapest images')
    images = []
    images.append(cv2.imread('stitching/budapest1.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/budapest2.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/budapest3.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/budapest4.jpg', cv2.IMREAD_COLOR))
elif (choice == '3'):
    print('Fetching newspaper images')
    images = []
    images.append(cv2.imread('stitching/newspaper1.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/newspaper2.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/newspaper3.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/newspaper4.jpg', cv2.IMREAD_COLOR))
else:
    print('Fetching scene images')
    images = []
    images.append(cv2.imread('stitching/s1.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/s2.jpg', cv2.IMREAD_COLOR))


feature_detector(images[0])
cv2.waitKey()
cv2.destroyAllWindows()
```

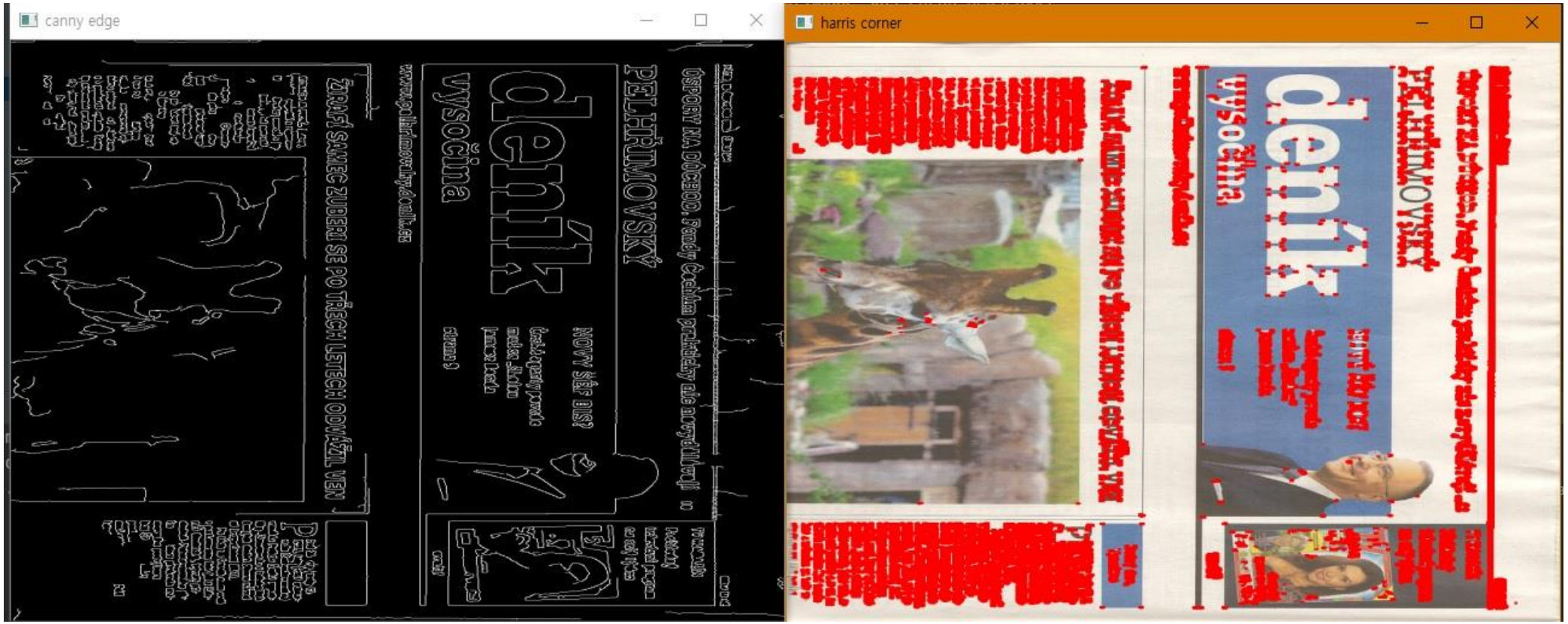
결과 | Image 1



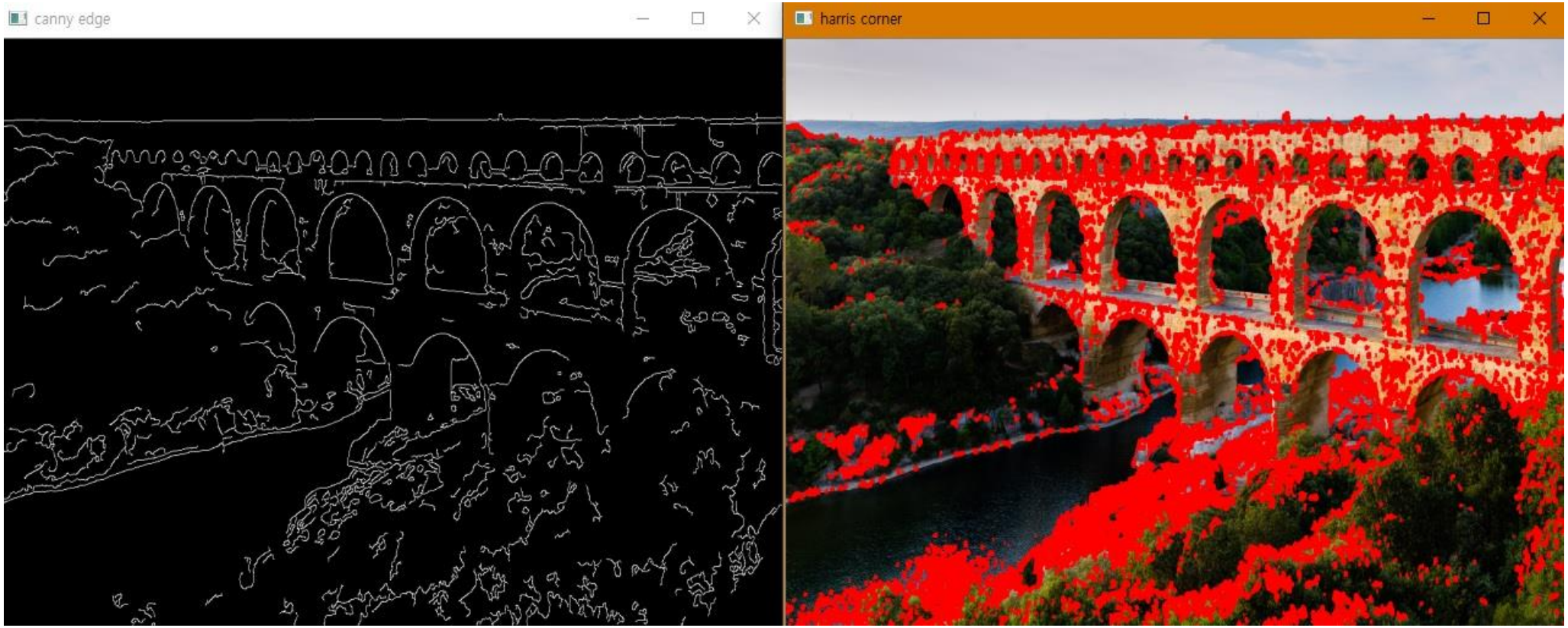
결과 | Image 2



결과 | Image 3



결과 | Image 4



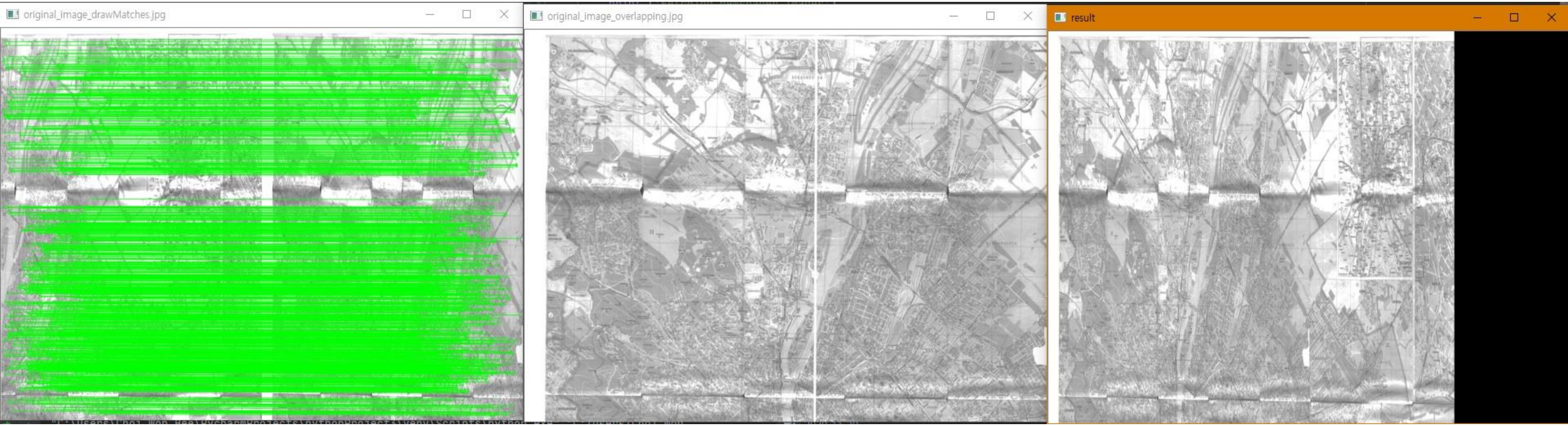
2. 코딩 Coding : RANSAC을 통해서 두 장의 영상간의 homography 계산

```python
def image_warping(image1, image2):
    img1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
    img2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
    sift = cv2.xfeatures2d.SIFT_create()
    # find key points
    kp1, des1 = sift.detectAndCompute(img1, None)
    kp2, des2 = sift.detectAndCompute(img2, None)
    match = cv2.BFMatcher()
    matches = match.knnMatch(des1, des2, k=2)
    good = []
    for m, n in matches:
        if m.distance < 0.3 * n.distance:# initially 0.03
            good.append(m)
    draw_params = dict(matchColor=(0, 255, 0), singlePointColor=None, flags=2)
    img3 = cv2.drawMatches(image1, kp1, image2, kp2, good, None, **draw_params)
    dst2 = cv2.resize(img3, dsize=(640, 480), interpolation=cv2.INTER_AREA)
    cv2.imshow("original_image_drawMatches.jpg", dst2)
    print('no. of points:', len(good))
    MIN_MATCH_COUNT = 1
    if len(good) > MIN_MATCH_COUNT:
        src_pts = np.float32([kp1[m.queryIdx].pt for m in good]).reshape(-1, 1, 2)
        dst_pts = np.float32([kp2[m.trainIdx].pt for m in good]).reshape(-1, 1, 2)
        M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
        h, w = img1.shape
        pts = np.float32([[0, 0], [0, h - 1], [w - 1, h - 1], [w - 1, 0]]).reshape(-1, 1, 2)
        dst = cv2.perspectiveTransform(pts, M)
        img2 = cv2.polylines(img2, [np.int32(dst)], True, 255, 3, cv2.LINE_AA)
        dst3 = cv2.resize(img2, dsize=(640, 480), interpolation=cv2.INTER_AREA)
        cv2.imshow("original_image_overlapping.jpg", dst3)
    else:
        print("Not enough matches are found - %d/%d", (len(good) / MIN_MATCH_COUNT))
    result = cv2.warpPerspective(image1, M, (image1.shape[1] + image2.shape[1], image1.shape[0]))
    result[0:image2.shape[0], 0:image2.shape[1]] = image2
    dst4 = cv2.resize(result, dsize=(640, 480), interpolation=cv2.INTER_AREA)
    cv2.imshow('result', dst4)


choice = input("Please input image choices(no.): \n 1. BOAT\n 2. BUDAPEST\n 3. NEWSPAPER\n 4. SCENE\n ")

if (choice == '1'):
    print('Fetching boat images')
    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))
elif (choice == '2'):
    print('Fetching budapest images')
    images = []
    images.append(cv2.imread('stitching/budapest1.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/budapest2.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/budapest3.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/budapest4.jpg', cv2.IMREAD_COLOR))
elif (choice == '3'):
    print('Fetching newspaper images')
    images = []
    images.append(cv2.imread('stitching/newspaper1.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/newspaper2.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/newspaper3.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/newspaper4.jpg', cv2.IMREAD_COLOR))
else:
    print('Fetching scene images')
    images = []
    images.append(cv2.imread('stitching/s1.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/s2.jpg', cv2.IMREAD_COLOR))


image_warping(images[1], images[0])
cv2.waitKey()
cv2.destroyAllWindows()
```

결과 | Image 1


original_image_drawMatches.jpg
original_image_overlapping.jpg
result

결과 | Image 2


original_image_drawMatches.jpg
original_image_overlapping.jpg
result

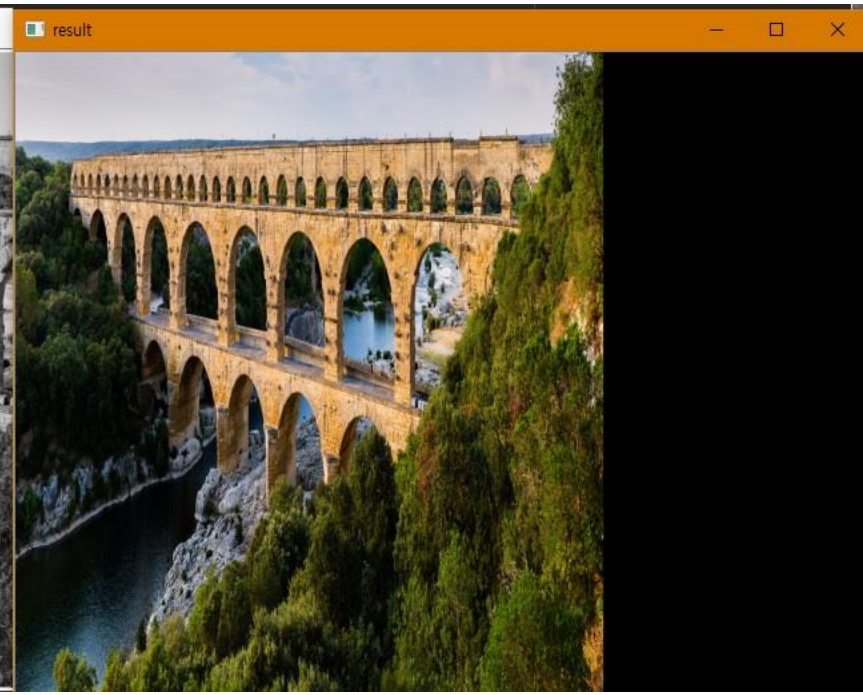
결과 | Image 3


original_image_drawMatches.jpg
original_image_overlapping.jpg
result
ÚSPORY NA DŮCHOD. Fondy Čechům prakticky nic nevydělávají ...10
PELHŘIMOVSKÝ
deník
vysočina
www.pelhrimovsky.denik.cz
NOVÝ ŠÉF BIS?
České agenty povede muž se „školou" Jamese Bonda
strana 9
ŽIRAFÍ SAMEC ZUBERI SE PO TŘECH LETECH ODVÁŽIL VEN
TV MAGAZÍN
Přehledový televizní program na celý týden
Předškoláky letos na školu připravná třída nepřipraví

결과 | Image 4


original_image_drawMatches.jpg


original_image_overlapping.jpg


result

3. 코딩 Coding : CreaterStitcher 함수를 이용하여 4개의 영상 셋에 대해서 파노라마 이미지

```python
#!/usr/bin/env python
import ...


def image_stitching(images):
    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)
    else:
        print('Error during stiching')
        exit(1)
    return pano

choice = input("Please input image choices(no.): \n 1. BOAT\n 2. BUDAPEST\n 3. NEWSPAPER\n 4. SCENE\n ")

if (choice == '1'):
    print ('Fetching boat images')
    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))
```

```python
elif (choice == '2'):
    print ('Fetching budapest images')
    images = []
    images.append(cv2.imread('stitching/budapest1.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/budapest2.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/budapest3.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/budapest4.jpg', cv2.IMREAD_COLOR))
elif (choice == '3'):
    print ('Fetching newspaper images')
    images = []
    images.append(cv2.imread('stitching/newspaper1.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/newspaper2.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/newspaper3.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/newspaper4.jpg', cv2.IMREAD_COLOR))
else:
    print ('Fetching scene images')
    images = []
    images.append(cv2.imread('stitching/s1.jpg', cv2.IMREAD_COLOR))
    images.append(cv2.imread('stitching/s2.jpg', cv2.IMREAD_COLOR))


pano = image_stitching(images)
cv2.waitKey()
cv2.destroyAllWindows()
```

첨부 | 사이즈 조정 코딩

Main Code

```
import cv2

src = cv2.imread("Image/champagne.jpg", cv2.IMREAD_COLOR)

dst = cv2.resize(src, dsize=(640, 480), interpolation=cv2.INTER_AREA)
dst2 = cv2.resize(src, dsize=(0, 0), fx=0.3, fy=0.7, interpolation=cv2.INTER_LINEAR)

cv2.imshow("src", src)
cv2.imshow("dst", dst)
cv2.imshow("dst2", dst2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Detailed Code

```
dst = cv2.resize(src, dsize=(640, 480), interpolation=cv2.INTER_AREA)
```

`cv2.resize(원본 이미지, 결과 이미지 크기, 보간법)`로 이미지의 크기를 조절할 수 있습니다.

`결과 이미지 크기`는 `Tuple`형을 사용하며, `(너비, 높이)`를 의미합니다. 설정된 이미지 크기로 변경합니다.

`보간법`은 이미지의 크기를 변경하는 경우, 변형된 이미지의 픽셀은 `추정`해서 값을 할당해야합니다.

`보간법`을 이용하여 픽셀들의 값을 할당합니다.

```
dst2 = cv2.resize(src, dsize=(0, 0), fx=0.3, fy=0.7, interpolation=cv2.INTER_LINEAR)
```

`cv2.resize(원본 이미지, dsize=(0, 0), 가로비, 세로비, 보간법)`로 이미지의 크기를 조절할 수 있습니다.

`결과 이미지 크기`가 `(0, 0)`으로 **크기를 설정하지 않은 경우**, `fx`와 `fy`를 이용하여 이미지의 비율을 조절할 수 있습니다.

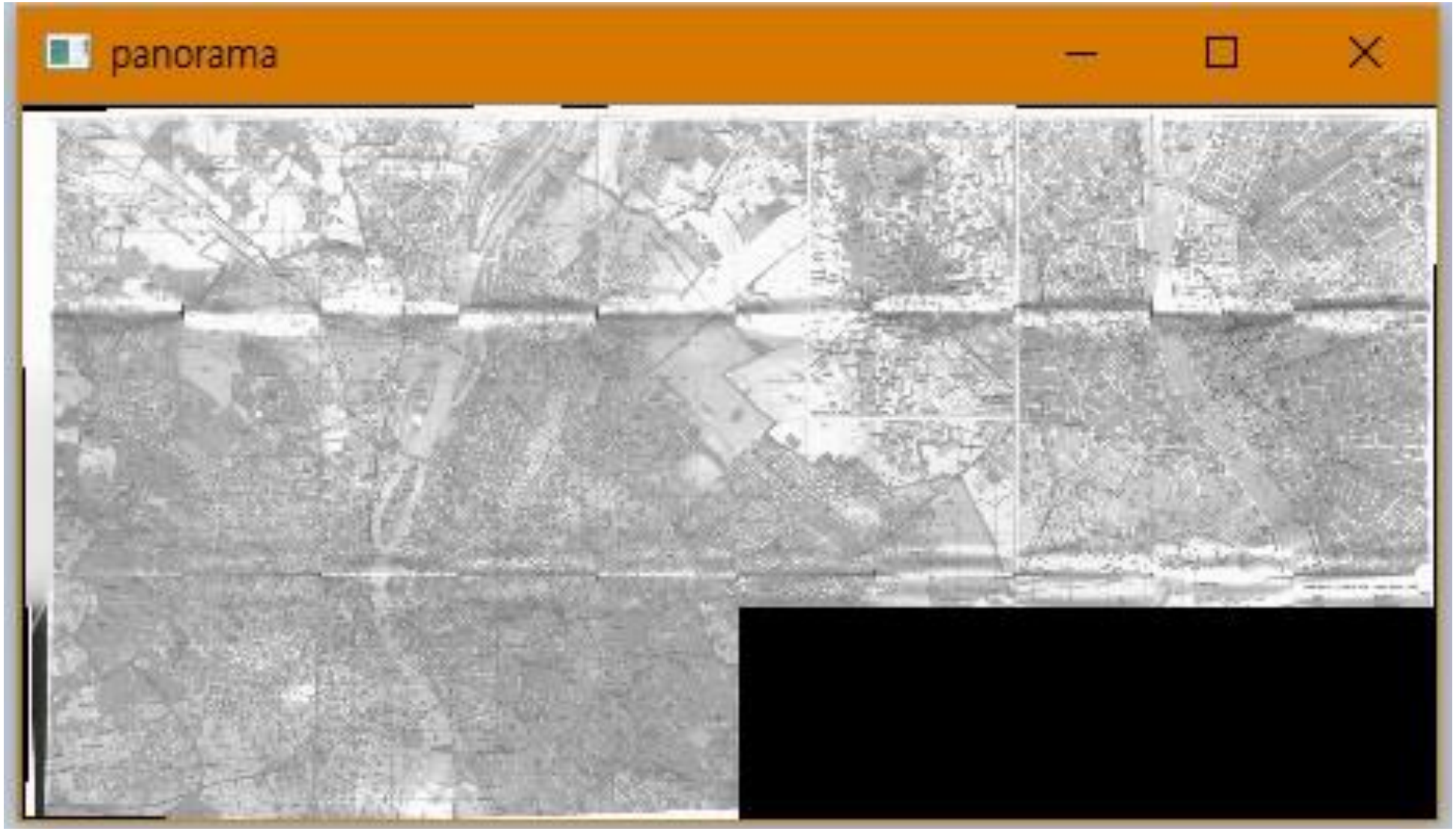
`fx`가 `0.3`인 경우, 원본 이미지 너비의 `0.3배`로 변경됩니다.

`fy`가 `0.7`인 경우, 원본 이미지 높이의 `0.7배`로 변경됩니다.

- Tip : `결과 이미지 크기`와 `가로비`, `세로비`가 모두 설정된 경우, `결과 이미지 크기`의 값으로 이미지의 크기가 조절됩니다.

결과 | Image 1

결과 | Image 2



결과 | Image 3

panorama

ZUBAŘE SAMCE ZLOBÍ OD PO TŘECH LETECH ODVÁŽIL VEK

Předškoláky letos na školu
přípravná třída nepřipraví

결과 | Image 4