Computer Graphics
실습 6.

2020. 10. 21

박 화 종

aqkrghkwhd@naver.com

실습 소개

- **과목 홈페이지**
  - 충남대학교 사이버 캠퍼스 (http://e-learn.cnu.ac.kr)

- **TA 연락처**
  - 박화종
  - 공대 5호관 506호 컴퓨터비전 연구실
  - aqkrghkwhd@naver.com

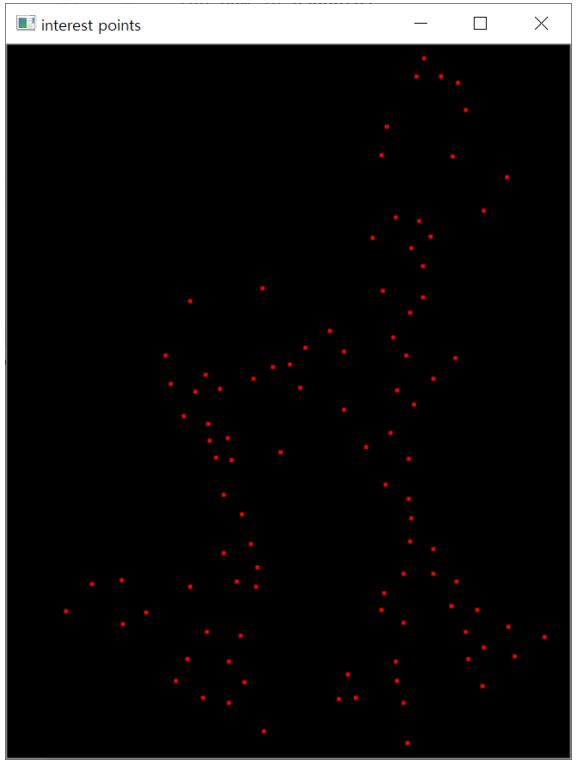
- **실습 튜터**
  - 최수민(00반)
  - eocjstnals12@naver.com
  - 신준호(01반)
  - wnsgh578@naver.com

목 차

4주차 과제 리뷰

- Harris Corner Detection 구현하기

original



my interest points

my harris corner detection

4주차 과제 리뷰

```python
def calc_derivatives(src):
    """
    #ToDo
    3x3 sobel 필터를 사용해서 Ix Iy 구하기
    :param src: 입력 이미지 (흑백)
    :return: Ix, Iy
    """
    '''
    Ix = ???
    Iy = ???
    '''
    # calculate Ix, Iy
    sobel_x, sobel_y = get_my_sobel()
    Ix = my_filtering(src, sobel_x)
    Iy = my_filtering(src, sobel_y)
    return Ix, Iy
```

4주차 과제 리뷰

```python
def HarrisDetector(src, gaus_filter_size = 3, gaus_sigma = 1, alpha = 0.04, threshold_rate = 0.01):
    (h, w) = src.shape
    # calculate Ix, Iy
    Ix, Iy = calc_derivatives(src)
```

• • •

```python
    # Square of derivatives
    IxIx = Ix**2
    IyIy = Iy**2
    IxIy = Ix * Iy
```

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I, \sigma_D))^2] =$$
$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

• • •

```python
    G_IxIx = GaussianFiltering(IxIx, fshape=(gaus_filter_size, gaus_filter_size), sigma=gaus_sigma)
    G_IyIy = GaussianFiltering(IyIy, fshape=(gaus_filter_size, gaus_filter_size), sigma=gaus_sigma)
    G_IxIy = GaussianFiltering(IxIy, fshape=(gaus_filter_size, gaus_filter_size), sigma=gaus_sigma)
```

• • •

```python
    #det = G_IxIx * G_IyIy - (G_IxIy**2)
    #tr = G_IxIx
    #har = det - (alpha * (tr ** 2))
    har = G_IxIx * G_IyIy - (G_IxIy**2) - alpha * (G_IxIx**2 + G_IyIy**2)
```

5주차 과제 추가 설명

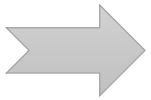
- **Integral image 사용시 속도 차이 확인**

```
start!
src.shape :  (552, 435, 3)
fsize :  5
M_harris time :  6.1276217
make integral image time :  0.8241929999999993
M_harris integral time :  1.1511037999999996
```

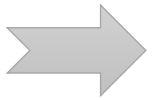
<- 이 속도는 채점 기준에 들어가지 않음

속도를 빠르게 하고 싶다면

1	2	3
4	5	6
7	8	9



1	3	6
5	?	.
12	.	.

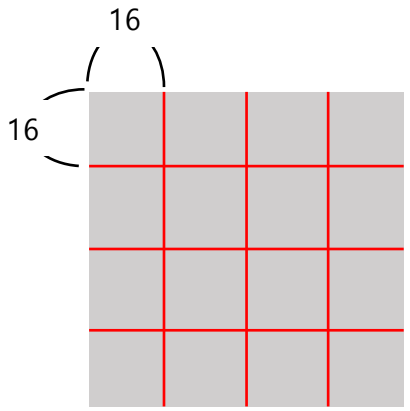


1	3	6
5	12	.
12	.	.

실습

- Orientation Assignment

간단한 실습을 위해...



Image

3. Orientation Assignment

- **Take 16×16 square window around detected feature from blurred image associated with the keypoint's scale**
- **Compute image gradients (magnitude and angle)**
- **Throw out weak edges**
- **Create histogram**

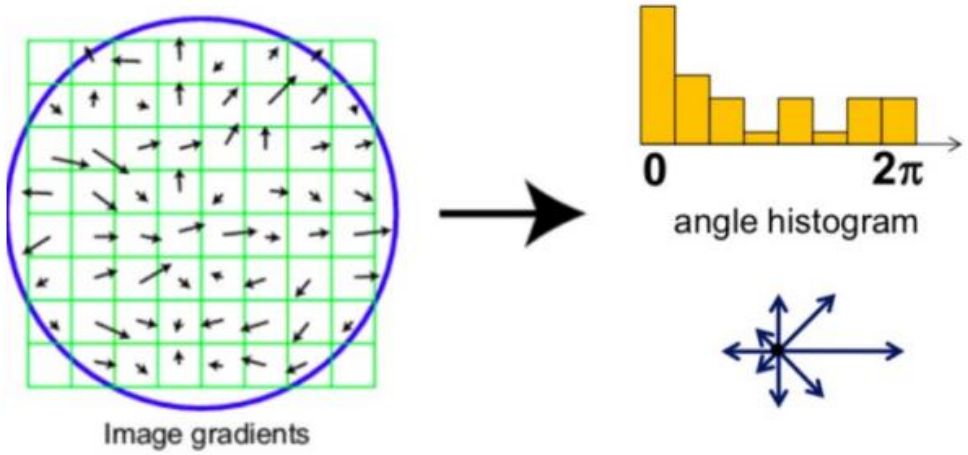


Image gradients

실습

- Orientation Assignment

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

def my_padding(src, filter):
    (h, w) = src.shape
    if isinstance(filter, tuple):
        (h_pad, w_pad) = filter
    else:
        (h_pad, w_pad) = filter.shape
    h_pad = h_pad // 2
    w_pad = w_pad // 2
    padding_img = np.zeros((h+h_pad*2, w+w_pad*2))
    padding_img[h_pad:h+h_pad, w_pad:w+w_pad] = src

    # repetition padding
    # up
    padding_img[:h_pad, w_pad:w_pad + w] = src[0, :]
    # down
    padding_img[h_pad + h:, w_pad:w_pad + w] = src[h - 1, :]
    # left
    padding_img[:, :w_pad] = padding_img[:, w_pad:w_pad + 1]
    # right
    padding_img[:, w_pad + w:] = padding_img[:, w_pad + w - 1:w_pad + w]

    return padding_img
```

실습

- Orientation Assignment

```python
def my_filtering(src, filter):
    (h, w) = src.shape
    (f_h, f_w) = filter.shape

    #filter 확인
    #print('<filter>')
    #print(filter)

    # 직접 구현한 my_padding 함수를 이용
    pad_img = my_padding(src, filter)

    dst = np.zeros((h, w))
    for row in range(h):
        for col in range(w):
            dst[row, col] = np.sum(pad_img[row:row + f_h, col:col + f_w] * filter)

    return dst

def get_my_sobel():
    sobel_x = np.dot(np.array([[1], [2], [1]]), np.array([[-1, 0, 1]]))
    sobel_y = np.dot(np.array([[-1], [0], [1]]), np.array([[1, 2, 1]]))
    return sobel_x, sobel_y

def calc_derivatives(src):
    # calculate Ix, Iy
    sobel_x, sobel_y = get_my_sobel()
    Ix = my_filtering(src, sobel_x)
    Iy = my_filtering(src, sobel_y)
    return Ix, Iy
```

실습

- Orientation Assignment

```python
def calc_angle(Ix, Iy):

    #return np.rad2deg(np.arctan(Iy/(Ix+1E-6)))
    # -180 ~ 180
    angle = np.rad2deg(np.arctan2(Iy, Ix))

    # 0 ~ 360
    return (angle+360) % 360

def calc_magnitude(Ix, Iy):
    magnitude = np.sqrt(Ix**2 + Iy**2)
    return magnitude

def calc_patch_hist(patch_ang, patch_mag, angle_range):
    h, w = patch_ang.shape[:2]
    assert h, w == patch_mag.shape[:2]
    vector_size = 360//angle_range

    vector = np.zeros(vector_size,)
    for row in range(h):
        for col in range(w):
            vector[int(patch_ang[row, col]//angle_range)] += patch_mag[row, col]

    return vector
```

실습

```python
def get_histogram(angle, magnitude, window_size=16, angle_range=30):
    h, w = angle.shape[:2]
    h = h//window_size
    w = w//window_size
    #print(h, w)

    assert 360 % angle_range == 0
    vector_size = 360//angle_range
    patches_vector = np.zeros((h, w, vector_size))
    print('calculate histogram...')
    for row in range(h):
        for col in range(w):
            patch_amg = angle[row*window_size:(row+1)*window_size, col*window_size:(col+1)*window_size]
            patch_mag = magnitude[row*window_size:(row+1)*window_size, col*window_size:(col+1)*window_size]
            patches_vector[row, col] = calc_patch_hist(patch_amg, patch_mag, angle_range)

    return patches_vector

def show_patch_hist(patch_vector):
    index = np.arange(len(patch_vector))
    plt.bar(index, patch_vector)
    plt.title('202050249')
    plt.show()
```

자신의 학번으로 변경(안할 시 1점 감점)

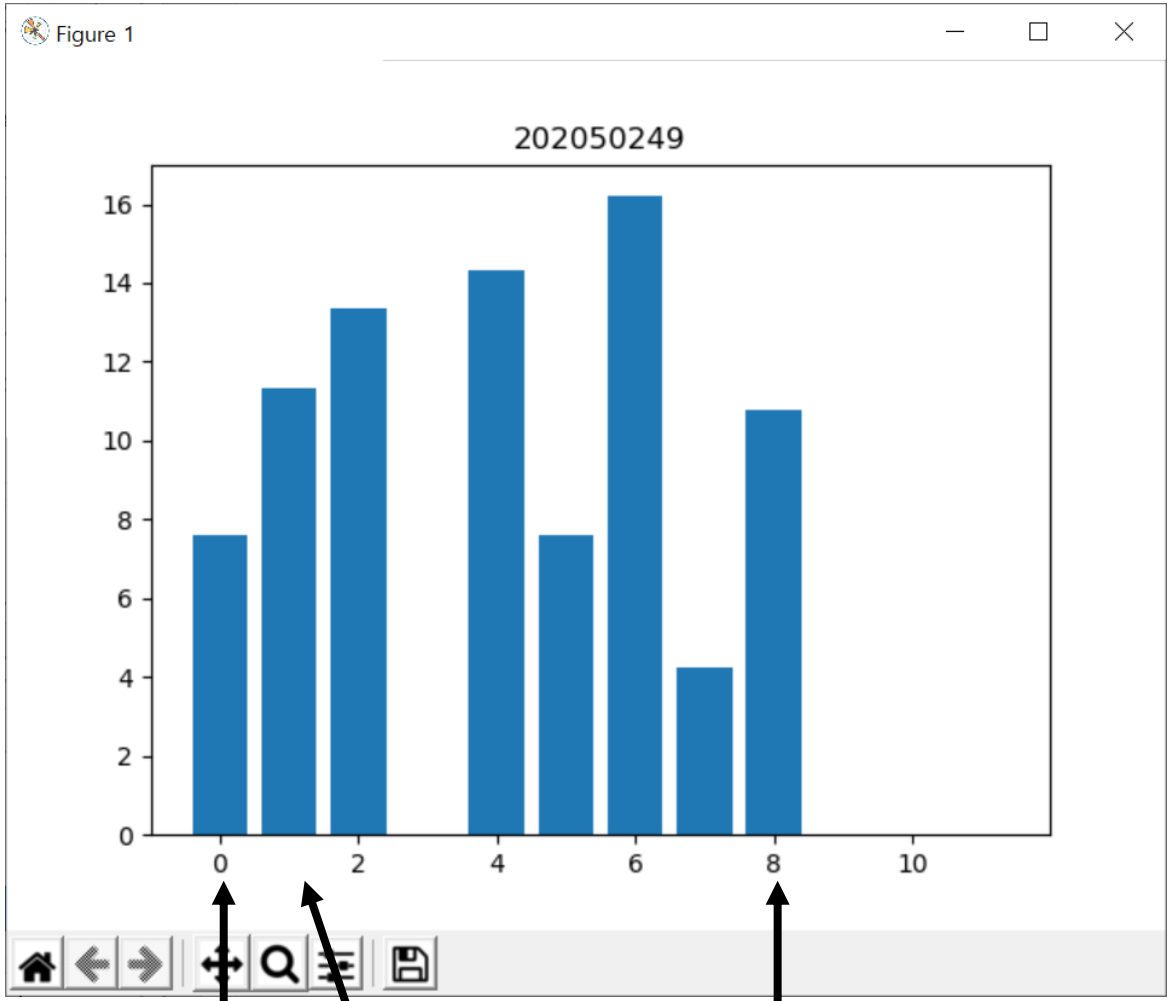
실습

```
get Ix and Iy...
calculate angle and magnitude
calculate histogram...
angle
[[225.         180.         156.80140949  23.19859051]
 [251.56505118 135.         135.          45.        ]
 [ 45.          45.         149.03624347   0.        ]
 [ 75.96375653  78.69006753 191.30993247 246.80140949]]
magnitude
[[ 4.24264069  6.          7.61577311  7.61577311]
 [ 3.16227766  1.41421356  7.07106781  5.65685425]
 [ 1.41421356  4.24264069  5.83095189  0.        ]
 [ 8.24621125  5.09901951 10.19803903  7.61577311]]
vector
[ 7.61577311 11.3137085  13.34523076  0.         14.31623327  7.61577311
 16.19803903  4.24264069 10.77805077  0.          0.          0.        ]

Process finished with exit code -1
```

```python
def main():
    src = cv2.imread('../image/Lena.png')
    gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
    print('get Ix and Iy...')
    Ix, Iy = calc_derivatives(gray)
    print('calculate angle and magnitude')
    angle = calc_angle(Ix, Iy)
    magnitude = calc_magnitude(Ix, Iy)

    patches_vector = get_histogram(angle, magnitude,
                                   window_size=4, angle_range = 30)

    print('angle')
    print(angle[:4, :4])
    print('magnitude')
    print(magnitude[:4, :4])
    print('vector')
    print(patches_vector[0, 0])
    #print(np.max(angle), np.min(angle))

    show_patch_hist(patches_vector[0,0])



if __name__ =='__main__':
    main()
```

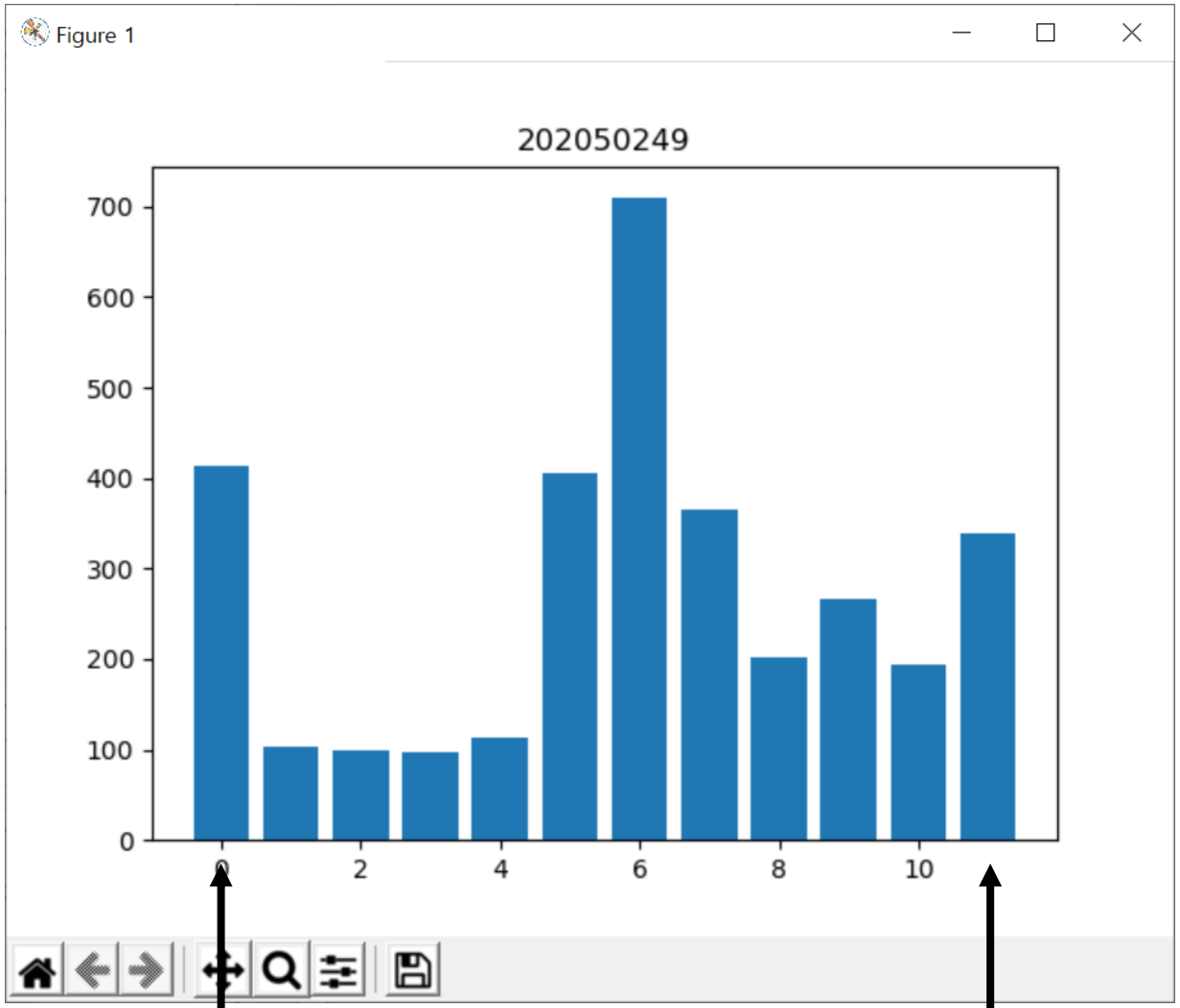
실습

```
    patches_vector = get_histogram(angle, magnitude,
                                   window_size=4, angle_range = 30)

get Ix and Iy...
calculate angle and magnitude
calculate histogram...
angle
[[225.         180.         156.80140949  23.19859051]
 [251.56505118 135.         135.          45.        ]
 [ 45.          45.         149.03624347   0.        ]
 [ 75.96375653  78.69006753 191.30993247 246.80140949]]
magnitude
[[ 4.24264069  6.          7.61577311  7.61577311]
 [ 3.16227766  1.41421356  7.07106781  5.65685425]
 [ 1.41421356  4.24264069  5.83095189  0.        ]
 [ 8.24621125  5.09901951 10.19803903  7.61577311]]
vector
[ 7.61577311 11.3137085  13.34523076  0.         14.31623327  7.61577311
 16.19803903  4.24264069 10.77805077  0.          0.          0.        ]

Process finished with exit code -1
```



실습

```
patches_vector = get_histogram(angle, magnitude,
                               window_size=16, angle_range = 30)
```



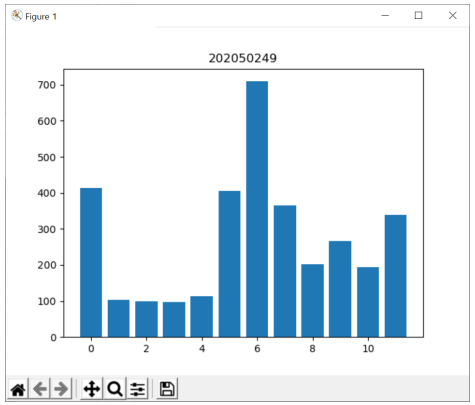
해당 결과를 보고서에 꼭 첨부할 것
pdf에 첨부하지 않으면 2점 감점

0 ~ 30

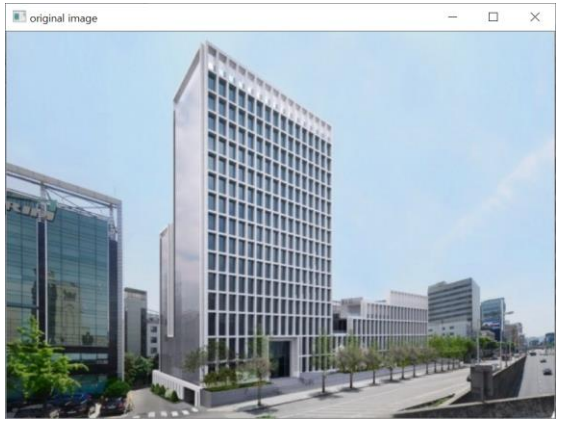
330 ~ 360

과제

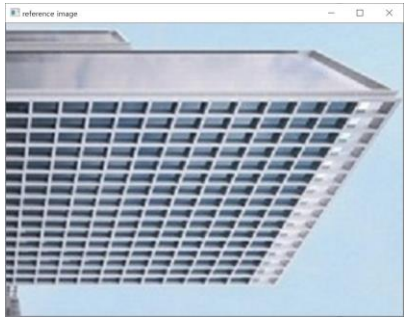
- Determining unknown transformations



실습 진행한 결과도 첨부



original



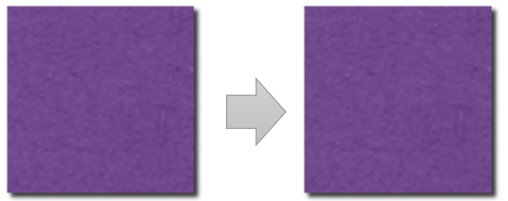
reference

transform
RANSAC

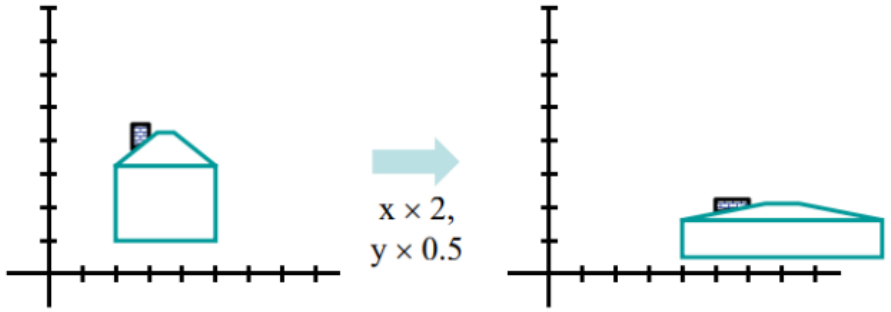
transform

과제

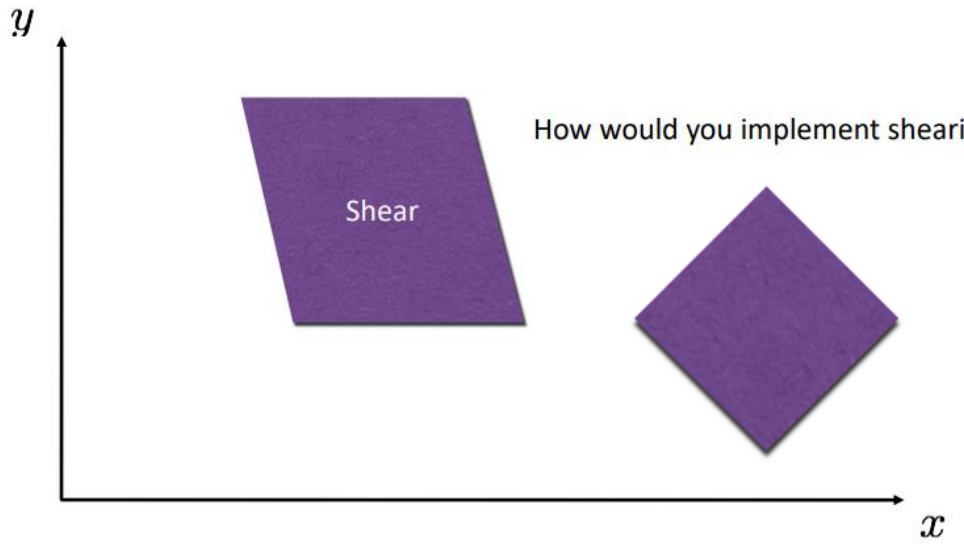
- Determining unknown transformations
  - Affine transform
    - Rotate, shear, translation, scaling 네 가지를 합쳐서 Affine 변환



Translation



Scaling



Shear Rotate

과제

- Determining unknown transformations
  - sift = cv2.xfeatures2d.SIFT_create() 함수 사용
    - sift.detectAndCompute(img, None)을 이용하여 keypoint와 descriptor를 구할 수 있다.

```
sift = cv2.xfeatures2d.SIFT_create(keypoint_num)

kp1, des1 = sift.detectAndCompute(img1, None)
kp2, des2 = sift.detectAndCompute(img2, None)

print(kp1[0].pt)
print(des1[0])
```

```
(304.55364990234375, 112.65033721923828)
[ 28.   7.   3.  40.  79.  42.  19.  16. 110.  20.   3.  59.  16.   5.
   5.  22.  94.  13.  17.  76.  17.   1.   2.  12.  55.   6.   5.  21.
  55.  48.  10.  11.  28.  10.   9.  28.  55.  70.  40.  20.  46.   6.
   7.  75. 100.  43.  43.  37. 110.  20.  17.  65.  27.   5.   2.  21.
  70.  13.  19.  93.  26.   3.   2.   8. 108.  20.   9.  49.  13.   5.
   3.  18.  43.   5.   8.  32. 110. 107.  32.  19.  87.  11.   3.  53.
  58.  42.  55.  62. 103.  15.  16.  87.  31.   1.   0.   9. 110.  15.
   2.  45.  12.   1.   1.  14.  74.  17.  20.  72.  19.   6.   8.  10.
  46.   6.   3.  19. 103.  69.  23.  14.  78.   9.   4.  22.  46.  37.
  25.  32.]
```

keypoint(x,y) &
descriptor – 128Dim

과제

- Determining unknown transformations

```python
def feature_matching(img1, img2, RANSAC=False, threshold = 300, keypoint_num = None, iter_num = 500):

    sift = cv2.xfeatures2d.SIFT_create(keypoint_num)

    kp1, des1 = sift.detectAndCompute(img1, None)
    kp2, des2 = sift.detectAndCompute(img2, None)

    distance = []
    for idx_1, des_1 in enumerate(des1):
        dist = []
        for idx_2, des_2 in enumerate(des2):
            dist.append(L2_distance(des_1, des_2))

        distance.append(dist)

    distance = np.array(distance)

    min_dist_idx = np.argmin(distance, axis=1)
    min_dist_value = np.min(distance, axis=1)
```

L2_distance 함수 완성

[1,2,3]
[4,5,6]
=> 5.19615…
[0,0,0]
[3,4,12]
=> 13

과제

- Determining unknown transformations

img1과 img2의 가장 짧은 거리의 좌표쌍
threshold보다 거리값이 크다면 continue

print(points)

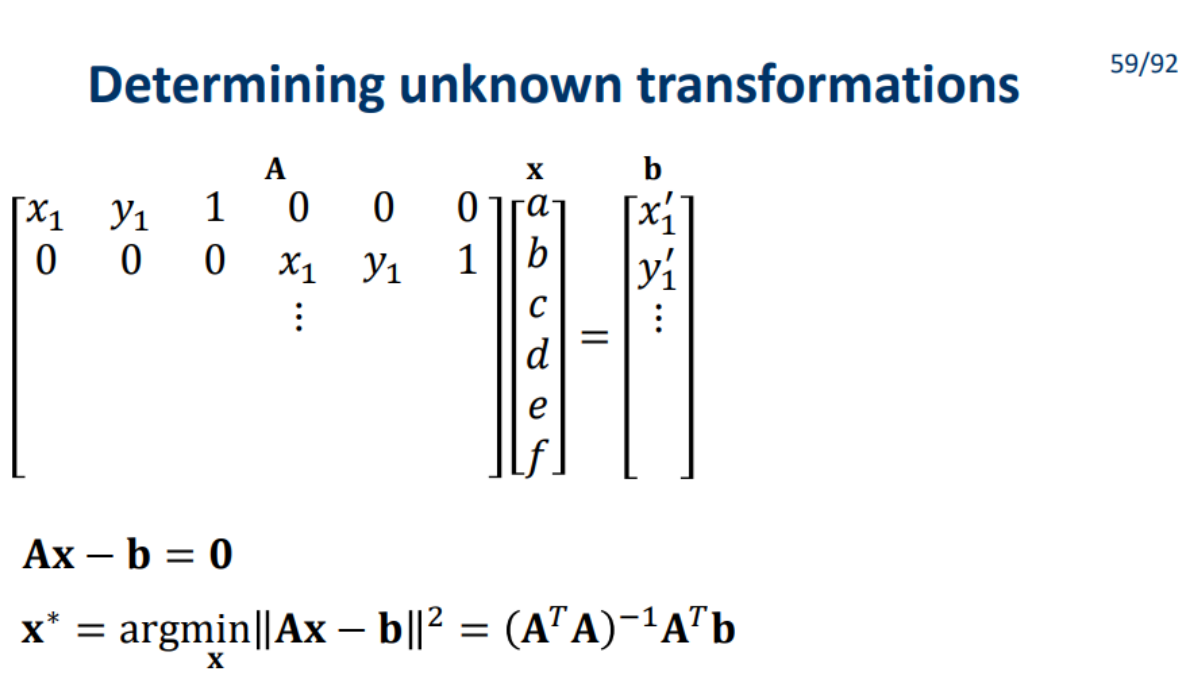
```
[[(305, 113), (430, 268)], [(232, 73), (503, 135)], [(232, 339), (23, 370)],
```

```
points = []
for idx, point in enumerate(kp1):
    if min_dist_value[idx] >= threshold:
        continue

    x1, y1 = point.pt
    x2, y2 = kp2[min_dist_idx[idx]].pt

    x1 = int(np.round(x1))
    y1 = int(np.round(y1))

    x2 = int(np.round(x2))
    y2 = int(np.round(y2))
    points.append([(x1, y1), (x2, y2)])
```

[[(img1의 x좌표, img1의 y좌표), (img2의 x좌표, img2의 y좌표)], [….], …]

과제

- Determining unknown transformations




```python
# no RANSAC
if not RANSAC:

    A = []
    B = []
    for idx, point in enumerate(points):
        '''
        #ToDo
        #A, B 완성
        # A.append(???) 이런식으로 할 수 있음
        # 결과만 잘 나오면 다른방법으로 해도 상관없음
        '''
        ???
        ???

    A = np.array(A)
    B = np.array(B)
```

과제

- Determining unknown transformations

Determining unknown transformations

59/92

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ & & & \vdots & & \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} x_1' \\ y_1' \\ \vdots \end{bmatrix}}_{\mathbf{b}}$$

$$\mathbf{Ax} - \mathbf{b} = \mathbf{0}$$

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{Ax} - \mathbf{b}\|^2 = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

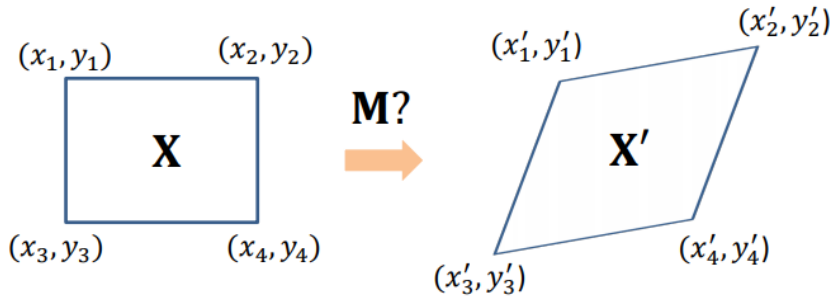
```
'''
#ToDo
#X 완성
#np.linalg.inv(V) : V의 역행렬 구하는것
#np.dot(V1, V2) : V1과 V2의 행렬곱
# V1.T : V1의 transpose
'''
X = ???
```

과제

- Determining unknown transformations

Determining unknown transformations 52/92

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$




```
'''
# ToDo
# 위에서 구한 X를 이용하여 M 완성
'''
M = ???

M = np.array(M)
M_ = np.linalg.inv(M)
```

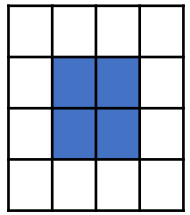
과제

- Determining unknown transformations

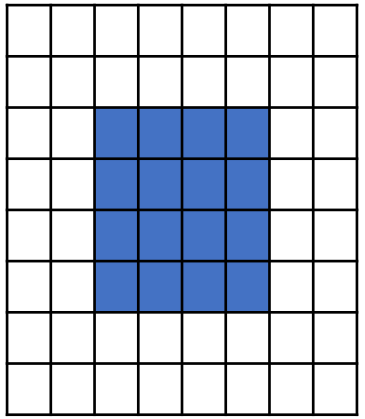
```
'''
# ToDo
# backward 방식으로 dst완성
'''
#Backward 방식
dst = ???
```

과제

- Forward 방식 vs Backward 방식



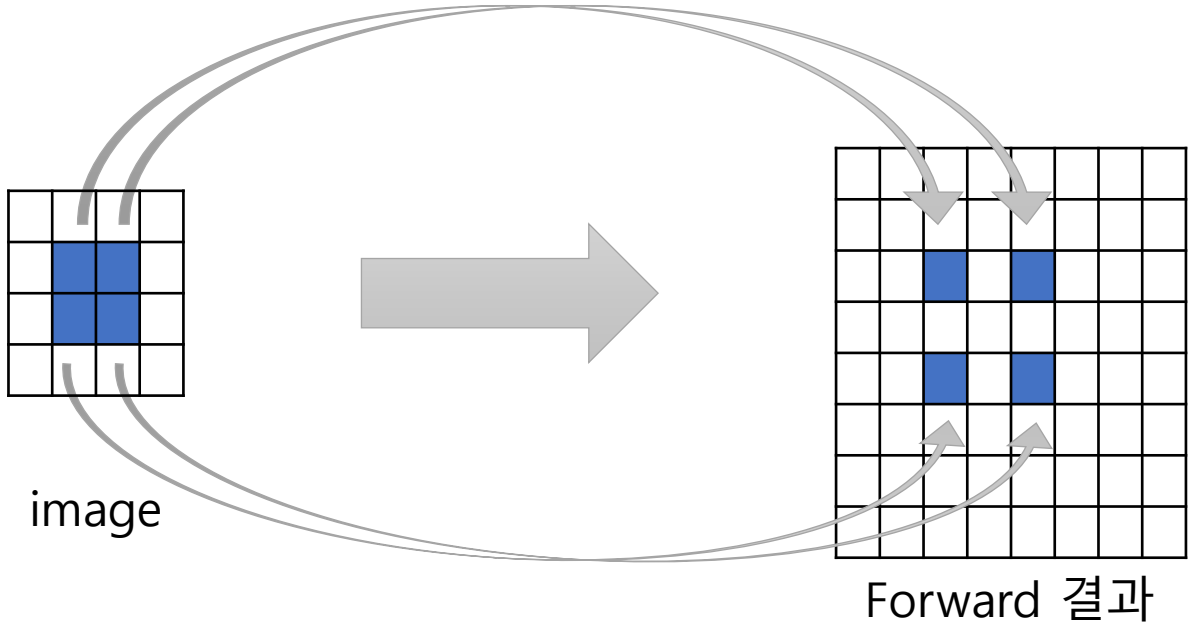
image



결과(희망)

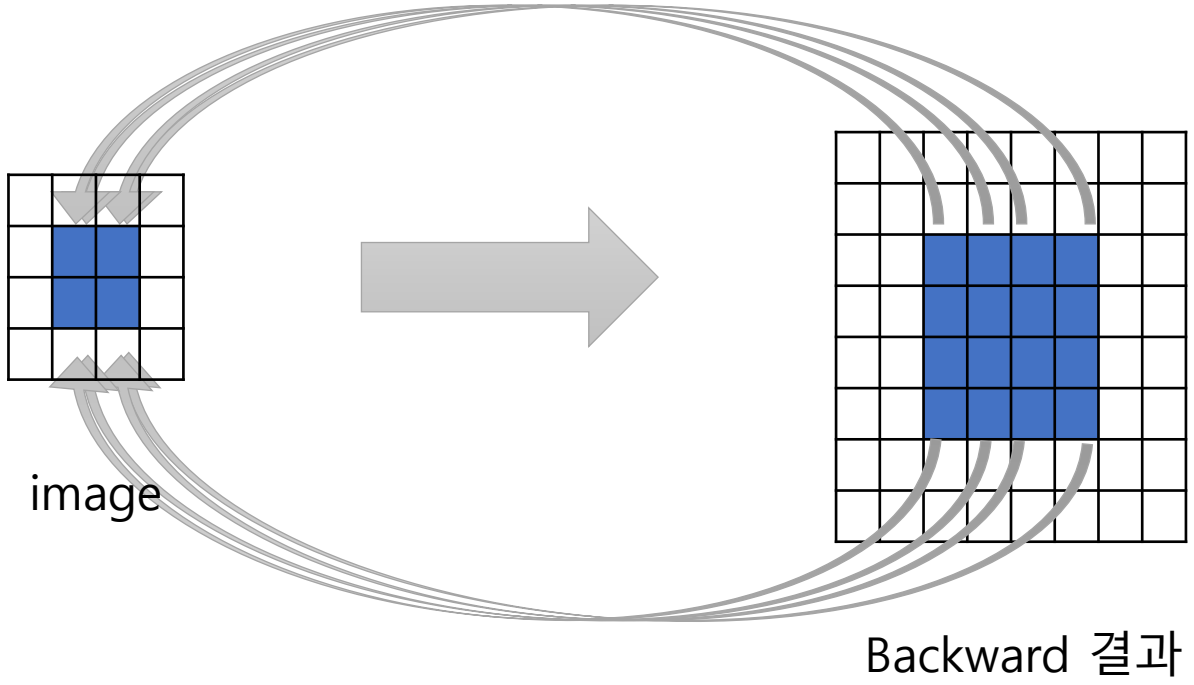
과제

- Forward 방식 vs Backward 방식
  - Forward 방식



과제

- Forward 방식 vs Backward 방식
  - Backward 방식



과제

- Forward 방식 vs Backward 방식

Affine 변환 전 이미지

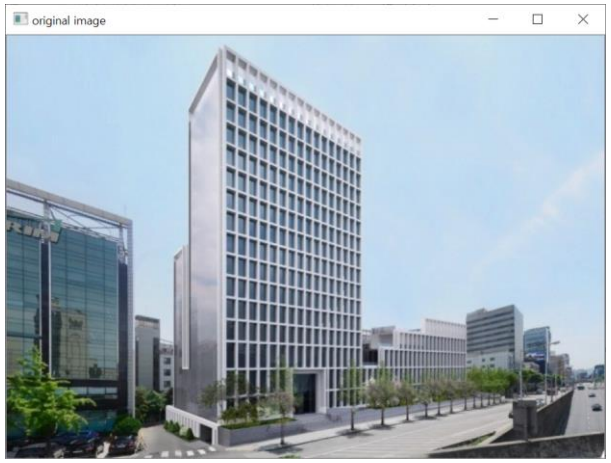
Affine 변환 후 이미지 (희망)

Affine 변환 후 이미지 (Forward)

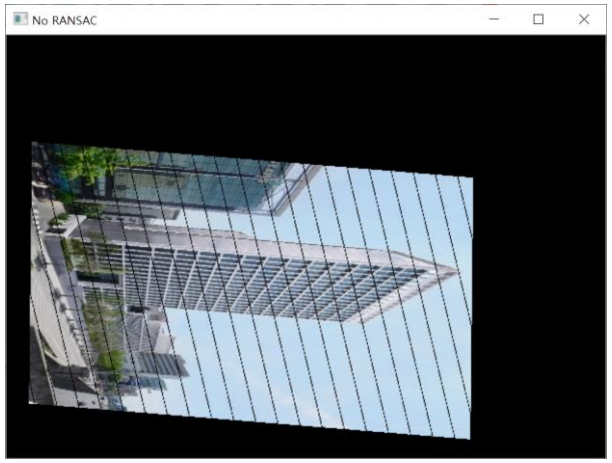
Affine 변환 후 이미지 (Backward)

과제

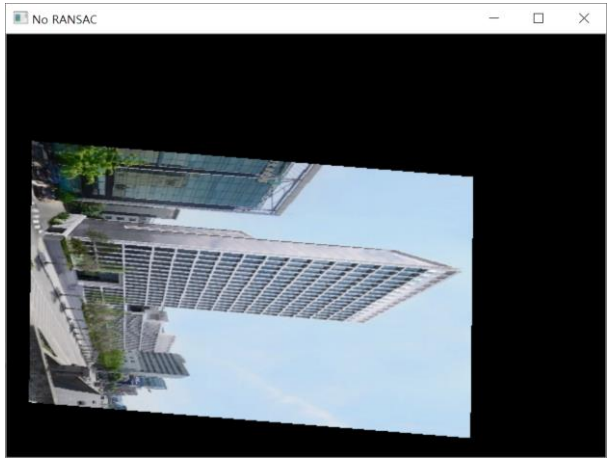
- Forward 방식 vs Backward 방식



original



Forward



Backward

Backward

과제

- Forward 방식 vs Backward 방식
  - Forward 방식
    - 행렬 M을 이용해 MX = x′ 방식으로 좌표를 찾기
    - h, w = src.shape[:2]
    - for row in range(h):
    - for col in range(w):
    - dst[row′,col′] = src[row, col]
  - Backward 방식
    - MX = X′ 이 아니라 X = M^{-1} * X′ 방식으로 좌표를 찾기
    - h, w = dst.shape[:2]
    - for row in range(h):
    - for col in range(w):
    - dst[row′,col′] = src[row, col]

과제

- Determining unknown transformations

Determining unknown transformations

59/92

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ & & & \vdots & & \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} x_1' \\ y_1' \\ \vdots \end{bmatrix}}_{\mathbf{b}}$$

$$\mathbf{Ax} - \mathbf{b} = \mathbf{0}$$

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{Ax} - \mathbf{b}\|^2 = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

```
#use RANSAAC
else:
    points_shuffle = points.copy()

    inliers = []
    M_list = []
    for i in range(iter_num):
        random.shuffle(points_shuffle)
        three_points = points_shuffle[:3]

        A = []
        B = []
        #3개의 point만 가지고 M 구하기
        for idx, point in enumerate(three_points):
            '''
            #ToDo
            #A, B 완성
            # A.append(???) 이런식으로 할 수 있음
            # 결과만 잘 나오면 다른방법으로 해도 상관없음
            '''
            ???
            ???

        A = np.array(A)
        B = np.array(B)
```

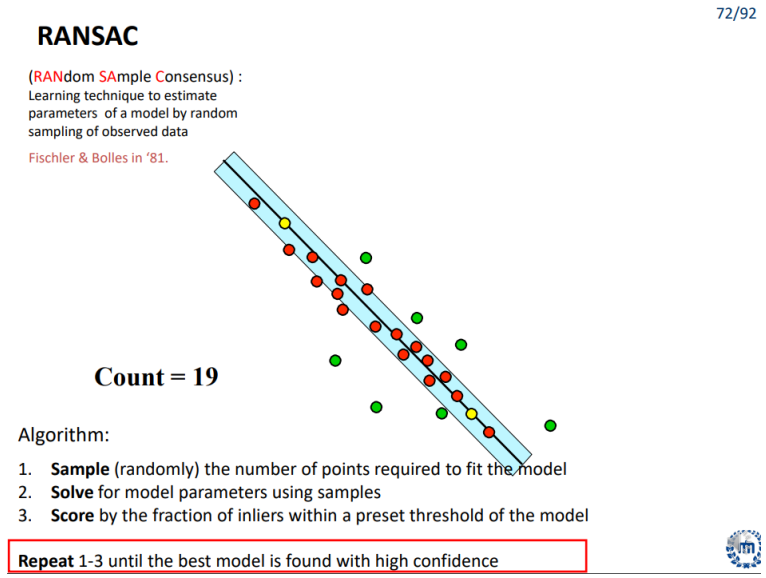
과제

- Determining unknown transformations
  - 역행렬이 구해지지 않는 경우 try, except 를 이용해 해결

```
try:
    '''
    #ToDo
    #X 완성
    #np.linalg.inv(V) : V의 역행렬 구하는것
    #np.dot(V1, V2) : V1과 V2의 행렬곱
    # V1.T : V1의 transpose 단, type이 np.array일때만 가능. type이 list일때는 안됨
    '''
    X = ???

except:
    print('can\'t calculate np.linalg.inv((np.dot(A.T, A)) !!!!!')
    continue

'''
# ToDo
# 위에서 구한 X를 이용하여 M 완성
'''
M = ???

M_list.append(M)
```

과제

- Determining unknown transformations




```
        M_list.append(M)

        count_inliers = 0
        for idx, point in enumerate(points):
            '''
            # ToDo
            # 위에서 구한 M으로(3개의 point로 만든 M) 모든 point들에 대하여 예상 point 구하기
            # 구해진 예상 point와 실제 point간의 L2 distance 를 구해서 threshold_distance보다 작은 값이 있는 경우 inlier로 판단
            '''
            ???(M으로 구한 point)
            ???(실제 point)
            if L2_distance(???(M으로 구한 point), ???(실제 point)) < threshold_distance:
                count_inliers += 1

        inliers.append(count_inliers)

    inliers = np.array(inliers)
    max_inliers_idx = np.argmax(inliers)

    best_M = np.array(M_list[max_inliers_idx])
```

과제

- Determining unknown transformations

Affine 변환 후 이미지 (Forward)

Affine 변환 후 이미지 (Backward)

```
    best_M = np.array(M_list[max_inliers_idx])

    M = best_M
    M_ = np.linalg.inv(M)


    '''
    # ToDo
    # backward 방식으로 dst완성
    '''
    #Backward 방식
    dst = ???

return dst
```

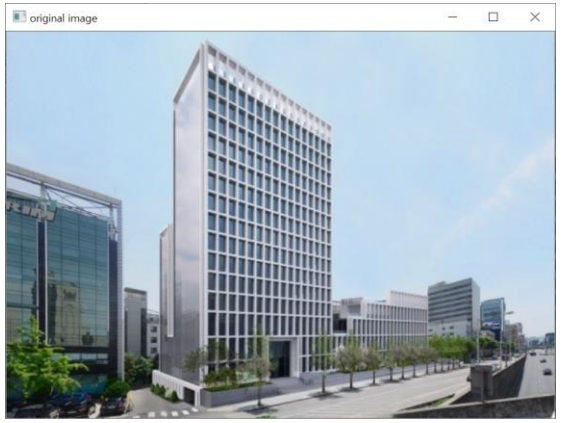
과제

- Determining unknown transformations

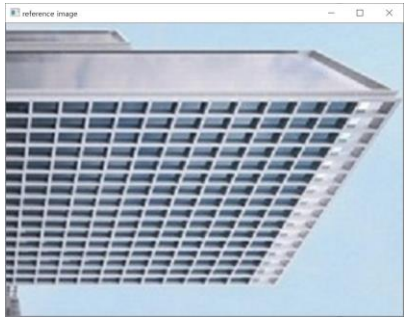
```
def main():
    img = cv2.imread('../image/building.jpg')
    img_ref = cv2.imread('../image/building_temp.jpg')

    threshold = 300
    iter_num = 500
    #속도가 너무 느리면 100과 같이 숫자로 입력
    keypoint_num = None
    #keypoint_num = 50
    threshold_distance = 10

    dst_no_ransac = feature_matching(img, img_ref, threshold=threshold,
    dst_use_ransac = feature_matching(img, img_ref, RANSAC=True, thresh

    cv2.imshow('No RANSAC' + 학번을 입력하세요, dst_no_ransac)
    cv2.imshow('Use RANSAC' + 학번을 입력하세요, dst_use_ransac)

    cv2.imshow('original image' + 학번을 입력하세요, img)
    cv2.imshow('reference image' + 학번을 입력하세요, img_ref)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

과제

- Determining unknown transformations



original



reference

transform
RANSAC

transform

과제 - 구현

- 제공된 코드가 제대로 동작하도록 구현하기
  - 빈칸 채우기
  - 코드 채점 시 error가 발생하면 감점
  - cv2.imshow 할 때 꼭! 자신의 학번 보이도록 변경
  - 이번 과제는 구현 속도에 대한 감점 없음
  - transform 결과 이미지는 ppt에 나온 것과 다를 수있음
  - transform 결과가 보일 때 이미지가 잘려서 보이면 감점

과제 - 보고서

- **보고서**
  - **내용:**
    - 이름, 학번, 학과
    - 구현 코드: 구현한 코드
    - 코드 설명: 구현한 코드에 대한 설명(설명은 1page를 넘기지 말 것, 1줄이어도 상관없음)
    - 이미지: 과제 첫 페이지를 참고하여 이미지 첨부(실습 결과 포함)
    - 느낀 점 : 결과를 보고 느낀 점, 혹은 과제를 하면서 어려웠던 점 등
    - 과제 난이도: 개인적으로 생각하는 난이도 (과제가 너무 쉬운 것 같다 등)

  - .pdf 파일로 제출 (이 외의 파일 형식일 경우 감점)

  - **파일 이름:**
    - [CG]20xxxxxxx_이름_n주차_과제.pdf

과제

- **제출 기한**
  - 11월 03일 23시 59분까지 (최대 점수 10점)
- **추가 제출 기한**
  - 11월 10일 23시 59분까지 (최대 점수 4점, 과제 총점 계산 후 -6점)
  - 11월 11일 00시 00분 이후 (점수 0점)
- **채점**
  - 구현을 못하거나(잘못 구현하거나) 보고서 내용이 빠진 경우 감점
  - 아무것도 구현하지 못해도 과제 제출하면 기본점수 있음
  - 다른 사람의 코드를 copy해서 제출시 보여준 사람, copy한 사람 둘 다 0점
  - 내장함수 사용시 감점(내장함수를 사용해도 된다고 말 한 것 제외)
- **제출 파일**
  - 아래의 파일을 압축해서 [CG]20xxxxxxx_이름_n주차_과제.zip로 제출
    - .py 파일 전부
    - .pdf 보고서 파일

QnA