Computer Graphics
실습 7.

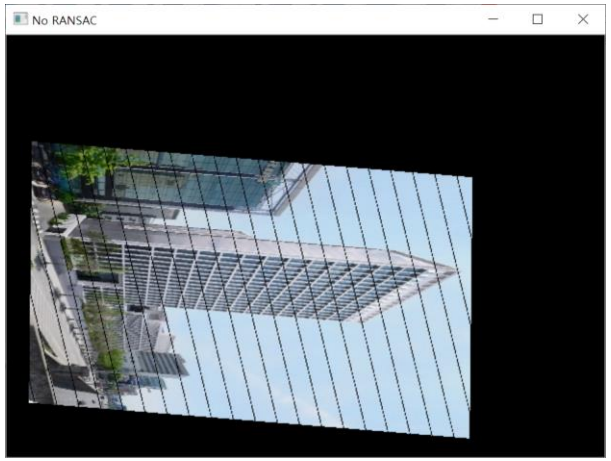
2020. 10. 28

박 화 종

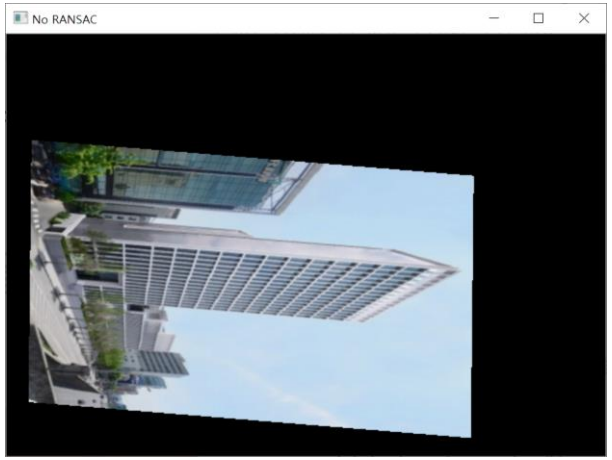
aqkrghkwhd@naver.com

공 지 사 항

- **6주차 과제 1주 연장**
- **Forward Backward 다시 설명 (6주차 Forward, Backward 잘못 구현)**



6주차



7주차

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

5주차 과제 리뷰

```
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
```

- **Integral image 사용시 속도 차이 확인**

자신의 학번을 추가하기

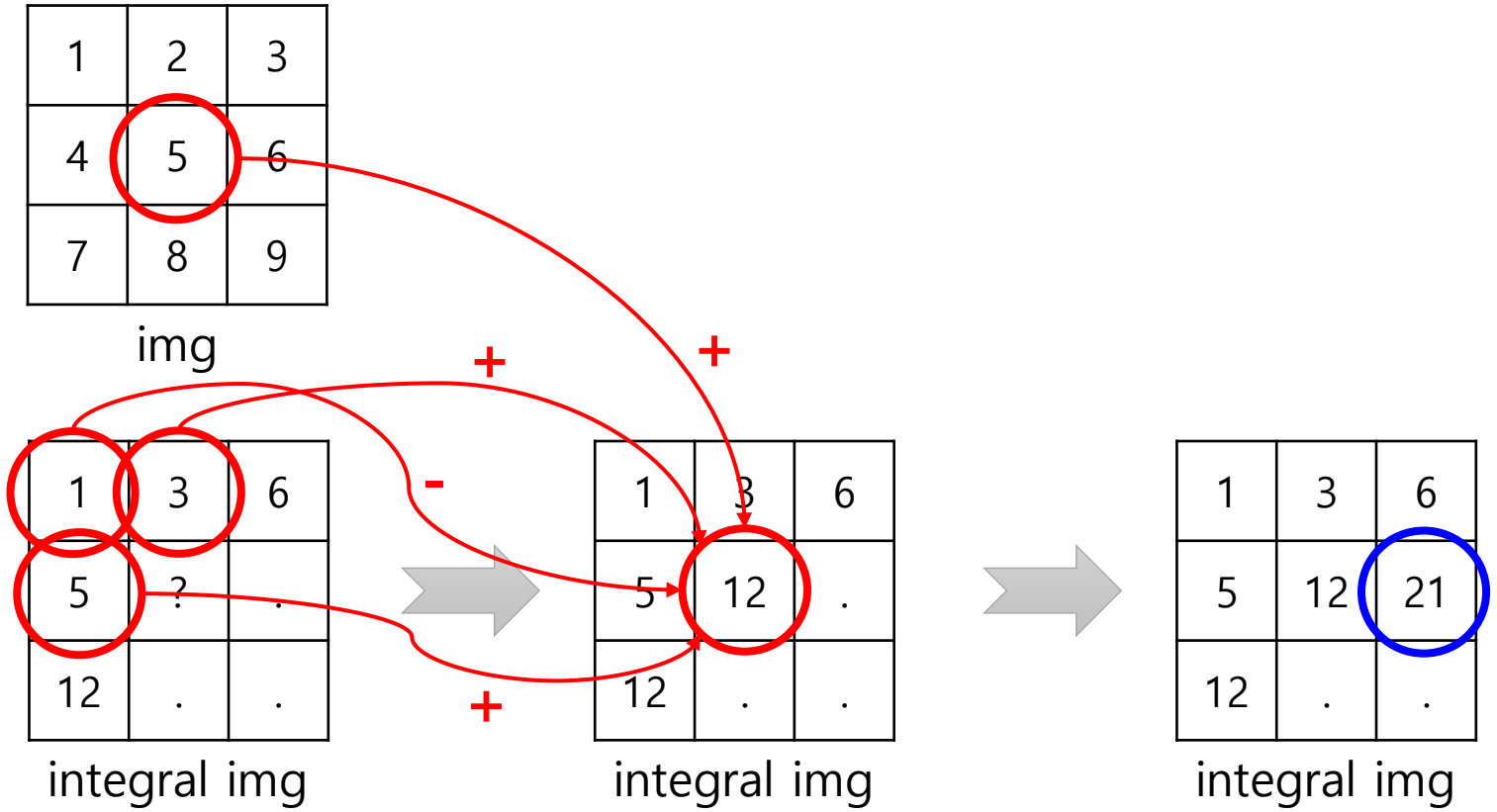
original

harris corner

harris corner
integral image

5주차 과제 리뷰

- **Integral image 사용시 속도 차이 확인**



5주차 과제 리뷰

- **Integral image 사용시 속도 차이 확인**

np.sum을 사용해서 만들어도 상관없음.
– 시간이 느려도 감점 없음

```python
def get_integral_image(src):
    assert len(src.shape) == 2
    h, w = src.shape
    dst = np.zeros(src.shape)
    for row in range(h):
        dst[row, 0] = np.sum(src[0:row+1, 0])

    for col in range(w):
        dst[0, col] = np.sum(src[0, 0:col+1])

    for row in range(1, h):
        for col in range(1, w):
            dst[row, col] = src[row, col] + dst[row-1, col] + dst[row, col-1] - dst[row-1, col-1]
    return dst
```

5주차 과제 리뷰

- Integral image 사용시 속도 차이 확인

```python
def calc_M_harris(IxIx, IxIy, IyIy, fsize = 5):
    assert IxIx.shape == IxIy.shape and IxIx.shape == IyIy.shape
    h, w = IxIx.shape
    M = np.zeros((h, w, 2, 2))
    IxIx_pad = my_padding(IxIx, (fsize, fsize))
    IxIy_pad = my_padding(IxIy, (fsize, fsize))
    IyIy_pad = my_padding(IyIy, (fsize, fsize))

    """for row in range(h):
        for col in range(w):
            M[row, col, 0, 0] = np.sum(IxIx_pad[row:row+fsize, col:col+fsize])
            M[row, col, 0, 1] = np.sum(IxIy_pad[row:row+fsize, col:col+fsize])
            M[row, col, 1, 0] = M[row, col, 0, 1]
            M[row, col, 1, 1] = np.sum(IyIy_pad[row:row+fsize, col:col+fsize])"""

    for row in range(h):
        for col in range(w):
            IxIx_value = 0
            IxIy_value = 0
            IyIy_value = 0
            for f_row in range(fsize):
                for f_col in range(fsize):
                    IxIx_value += IxIx_pad[row+f_row, col+f_col]
                    IxIy_value += IxIy_pad[row+f_row, col+f_col]
                    IyIy_value += IyIy_pad[row+f_row, col+f_col]

            M[row, col, 0, 0] = IxIx_value
            M[row, col, 0, 1] = IxIy_value
            M[row, col, 1, 0] = IxIy_value
            M[row, col, 1, 1] = IyIy_value

    return M
```

5주차 과제 리뷰

- **Integral image 사용시 속도 차이 확인**

```
def harris_detector(src, k = 0.04, threshold_rate = 0.01, fsize=5):
    ...

    R = np.zeros((h, w))
    for row in range(h):
        for col in range(w):
            det_M = M_harris[row, col, 0, 0] * M_harris[row, col, 1, 1] -(M_harris[row, col, 0, 1] * M_harris[row, col, 1, 0])
            trace_M =M_harris[row, col, 0, 0] + M_harris[row, col, 1, 1]
            #R[row, col] = det_M / (trace_M + 1E-8)
            R[row, col] = det_M - k * (trace_M * trace_M)

    # thresholding
    R[R < threshold_rate * np.max(R)] = 0
```

5주차 과제 리뷰

- **Integral image 사용시 속도 차이 확인**

```
def harris_detector_integral(src, k = 0.04, threshold_rate = 0.01, fsize=5):
    ...
    start = time.perf_counter()  # 시간 측정 시작
    M_integral = calc_M_integral(IxIx_integral, IxIy_integral, IyIy_integral, fsize)
    end = time.perf_counter()  # 시간 측정 끝
    print('M_harris integral time : ', end-start)

    R = np.zeros((h, w))
    for row in range(h):
        for col in range(w):
            det_M = M_integral[row, col, 0, 0] * M_integral[row, col, 1, 1] -(M_integral[row, col, 0, 1] * M_integral[row, col, 1, 0])
            trace_M =M_integral[row, col, 0, 0] + M_integral[row, col, 1, 1]
            #R[row, col] = det_M / (trace_M + 1E-8)
            R[row, col] = det_M - k * (trace_M * trace_M)
```

calc_M_integral()는 다음 page에...

5주차 과제 리뷰

- **Integral image 사용시 속도 차이 확인**

```python
def calc_M_integral(IxIx_integral, IxIy_integral, IyIy_integral, fsize = 5):
    assert IxIx_integral.shape == IxIy_integral.shape and IxIx_integral.shape == IyIy_integral.shape
    h, w = IxIx_integral.shape
    M = np.zeros((h, w, 2, 2))

    IxIx_integral_pad = my_padding(IxIx_integral, (fsize, fsize))
    IxIy_integral_pad = my_padding(IxIy_integral, (fsize, fsize))
    IyIy_integral_pad = my_padding(IyIy_integral, (fsize, fsize))

    for row in range(h):
        for col in range(w):
            M[row, col, 0, 0] = calc_local_integral_value(IxIx_integral_pad, (row, col), (row+fsize-1, col+fsize-1))
            M[row, col, 0, 1] = calc_local_integral_value(IxIy_integral_pad, (row, col), (row+fsize-1, col+fsize-1))
            M[row, col, 1, 0] = M[row, col, 0, 1]
            M[row, col, 1, 1] = calc_local_integral_value(IyIy_integral_pad, (row, col), (row+fsize-1, col+fsize-1))

    return M
```

다음 page에 설명

5주차 과제 리뷰

- **Integral image 사용시 속도 차이 확인**

1	2	3
4	5	6
7	8	9

img

1	3	6
5	12	21
12	27	45

integral img

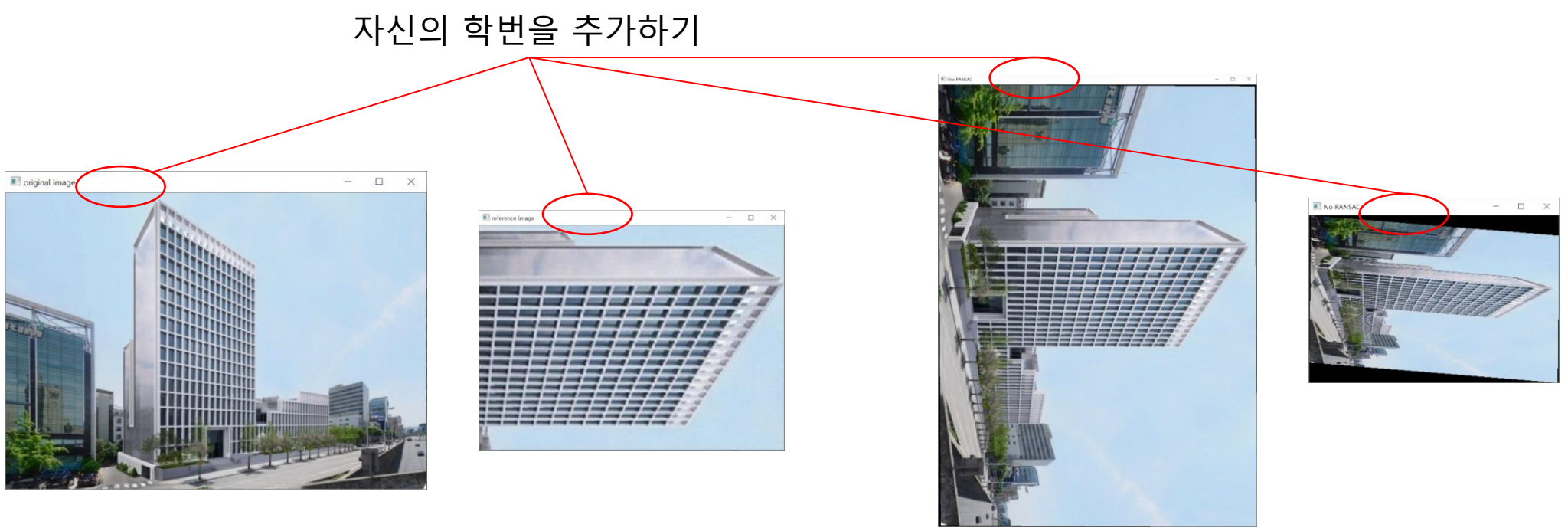
```python
def calc_local_integral_value(src, left_top, right_bottom):
    assert len(left_top) == 2
    assert len(right_bottom) == 2

    lt_row, lt_col = left_top
    rb_row, rb_col = right_bottom

    lt_val = src[lt_row - 1, lt_col - 1]
    lb_val = src[lt_row - 1, rb_col]
    rt_val = src[rb_row, lt_col - 1]
    rb_val = src[rb_row, rb_col]

    if lt_row == 0:
        lt_val = 0
        lb_val = 0
    if lt_col == 0:
        lt_val = 0
        rt_val = 0

    return lt_val - lb_val - rt_val + rb_val
```

6주차 과제 추가 설명

- **6주차 과제에서 Backward에 Bilinear 적용 안하고 제출 시 감점**
  - 이미 6주차 과제를 제출하신 분 중에 Bilinear를 적용하지 않은 경우 수정 부탁드립니다.(죄송합니다)
  - 대신 이번에 실습에서 코드를 많이 공개했습니다. 실습 코드를 참고해서 수정 부탁드립니다.

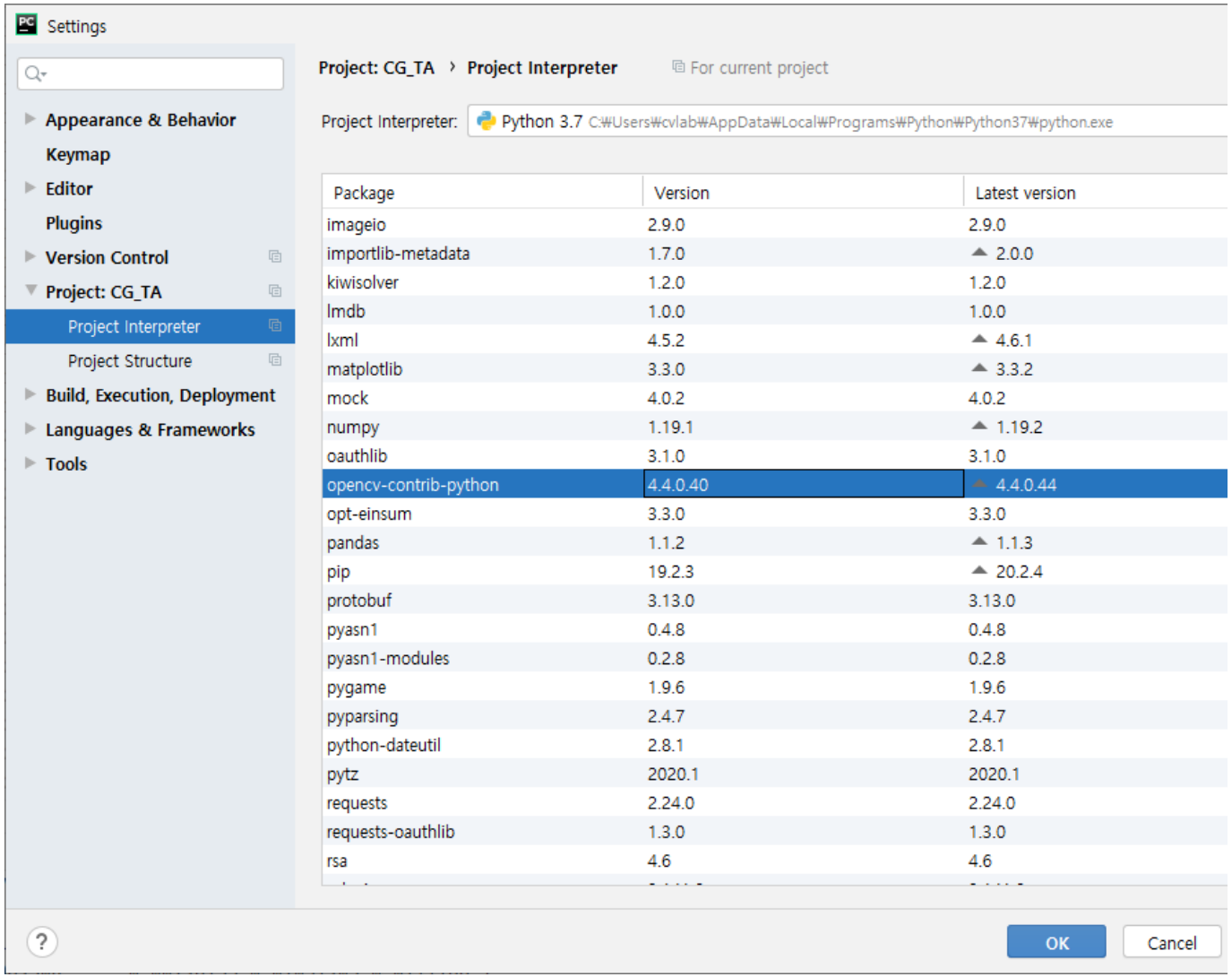
6주차 과제 추가 설명

- **따로 설명이 없어도 과제 보고서 작성 시 학번을 추가하셔야 합니다.**
  - 한글이 들어가면 결과가 제대로 안보이는 경우도 있음(그러니 학번만 추가)



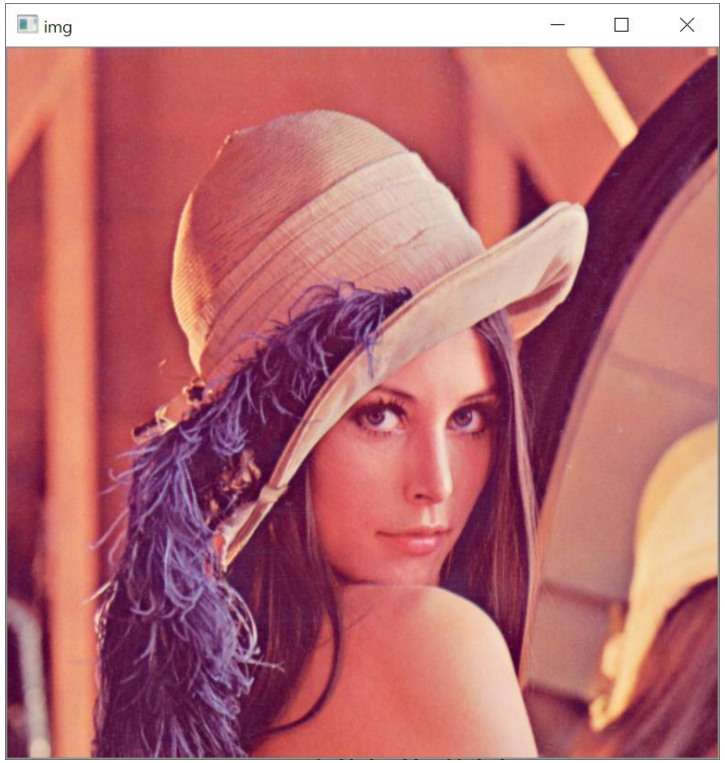
6주차 과제 추가 설명

- **SIFT 사용시 error가 발생하면**
  - opencv 버전을 4.4.0.40으로 버전 변경 후 사용

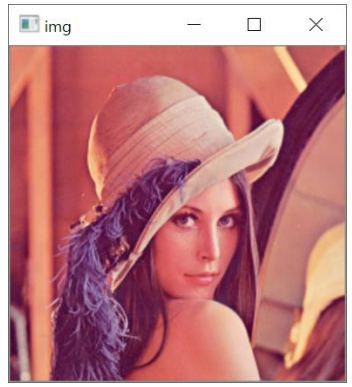


실습

- **Forward vs Backward**
  - Lena 이미지 h, w를 ½로 한 후 실습 진행



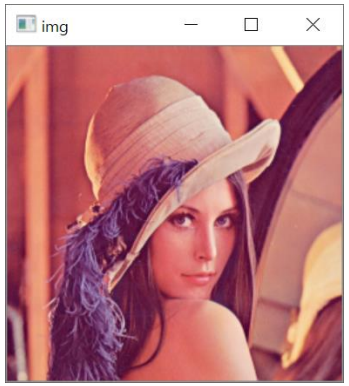
original (512 x 512)



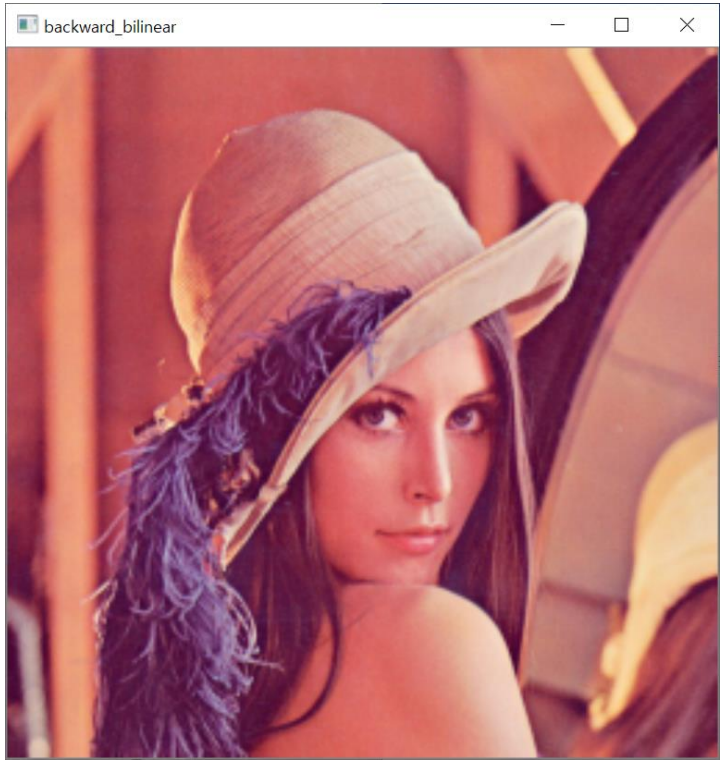
(256 x 256)

실습

- **Forward vs Backward**
  - 목표 Lena image를 2배 키우기(scaling)
    - bilinear interpolation



(256 x 256)



(512 x 512)

실습

- **Forward vs Backward**
  - 목표 Lena image를 2배 키우기(scaling)
    - bilinear interpolation

  - 2배 scaling을 할 때의 M

$$M = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - x', y' 구하기

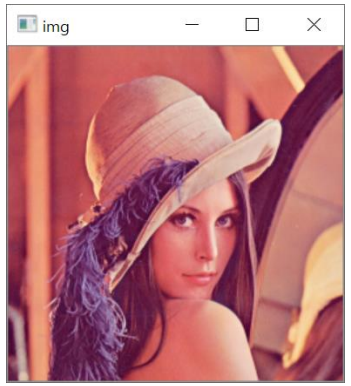
$$\begin{bmatrix} x` \\ y` \\ 1 \end{bmatrix} = \begin{bmatrix} 2, 0, 0 \\ 0, 2, 0 \\ 0, 0, 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

ex) x : 160, y : 160

$$\begin{bmatrix} 320 \\ 320 \\ 1 \end{bmatrix} = \begin{bmatrix} 2, 0, 0 \\ 0, 2, 0 \\ 0, 0, 1 \end{bmatrix} \begin{bmatrix} 160 \\ 160 \\ 1 \end{bmatrix}$$

실습

- **Forward vs Backward**
  - 목표 Lena image를 2배 키우기(scaling)
    - bilinear interpolation

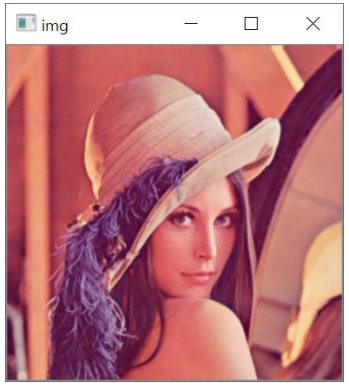


(256 x 256)

```python
def main():
    src = cv2.imread('../image/Lena.png')
    img = cv2.resize(src, dsize=(0, 0), fx=0.5, fy=0.5)
    cv2.imshow('img', img)
    scaling_test(img)



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

실습

- **Forward vs Backward**
  - 목표 Lena image를 2배 키우기(scaling)
    - bilinear interpolation

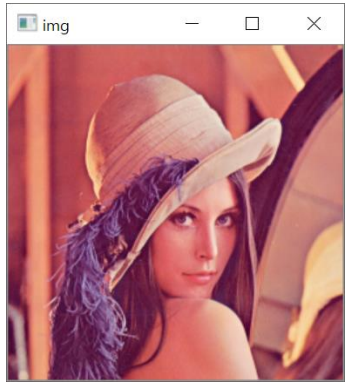


(256 x 256)

```
def scaling_test(src):
    h, w = src.shape[:2]
    rate = 2.0
    dst_for = np.zeros((int(np.round(h*rate)), int(np.round(w*rate)), 3))
    dst_back_bilinear = np.zeros((int(np.round(h*rate)), int(np.round(w*rate)), 3))
    M = np.array([[rate, 0, 0],
                  [0, rate, 0],
                  [0, 0, 1]])

    #FORWARD
    h_, w_ = dst_for.shape[:2]
    count = dst_for.copy()
    for row in range(h):
        for col in range(w):
            vec = np.dot(M, np.array([[col, row, 1]]).T)
            x = vec[0,0]
            y = vec[1,0]
```

실습

- **Forward vs Backward**
  - 목표 Lena image를 2배 키우기(scaling)
    - bilinear interpolation

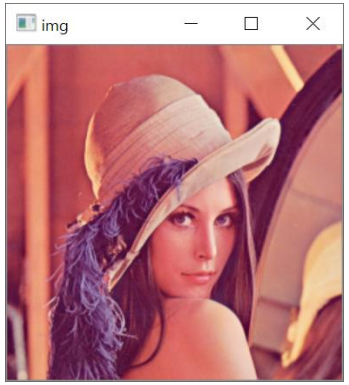


(256 x 256)

```
            x1 = int(np.floor(x))
            x2 = int(np.ceil(x))
            y1 = int(np.floor(y))
            y2 = int(np.ceil(y))

            points_list = [(y1, x1), (y1, x2), (y2, x1), (y2, x2)]
            points = set(points_list) # 중복제거

            for (row_, col_) in points:
                dst_for[min(row_, h_-1), min(col_, w_-1)] += src[row, col]
                count[min(row_, h_-1), min(col_, w_-1)] += 1

dst_for = (dst_for / count).astype(np.uint8)
cv2.imshow('forward', dst_for)
```

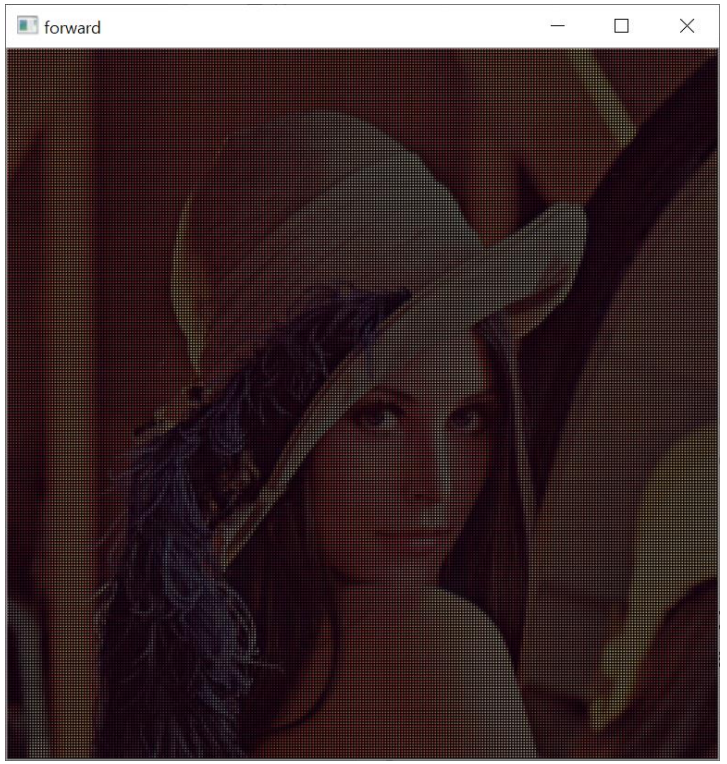
실습

- **Forward vs Backward**
  - 목표 Lena image를 2배 키우기(scaling)
    - bilinear interpolation



(256 x 256)

$$\begin{bmatrix} x` \\ y` \\ 1 \end{bmatrix} = \begin{bmatrix} 2, 0, 0 \\ 0, 2, 0 \\ 0, 0, 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



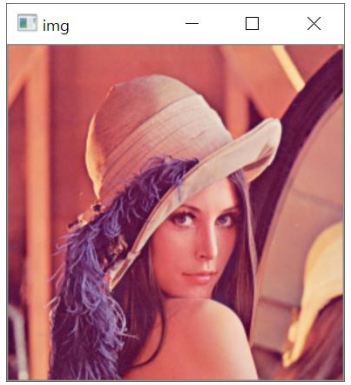
(512 x 512)

```
print('img')
print(src[:3, :3, 0]) #BGR 중 B만 참고

print('forward')
print(dst_for[:6, :6, 0]) #BGR 중 B만 참고
```

```
img
[[127 125 123]
 [126 126 123]
 [125 120 119]]
forward
[[127   0 125   0 123   0]
 [  0   0   0   0   0   0]
 [126   0 126   0 123   0]
 [  0   0   0   0   0   0]
 [125   0 120   0 119   0]
 [  0   0   0   0   0   0]]
```

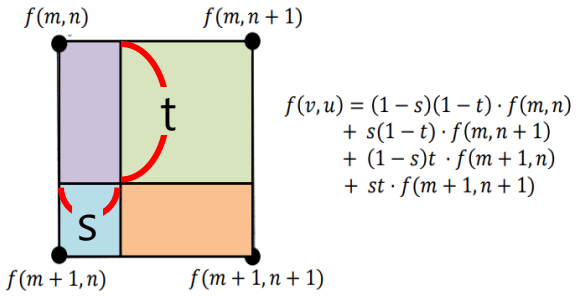
실습

- **Forward vs Backward**
  - 목표 Lena image를 2배 키우기(scaling)
    - bilinear interpolation



(256 x 256)

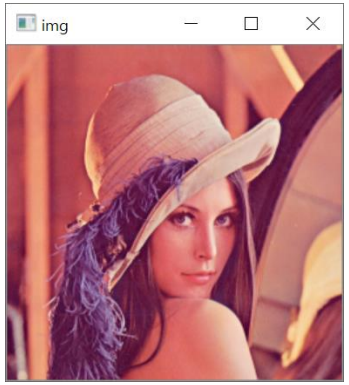
```
M
[[2 0 0]
 [0 2 0]
 [0 0 1]]
M 역행렬
[[0.5 0.  0. ]
 [0.  0.5 0. ]
 [0.  0.  1. ]]
```




```python
#M 역행렬
M_ = np.linalg.inv(M)
print('M')
print(M)
print('M 역행렬')
print(M_)
h_, w_ = dst_back_bilinear.shape[:2]
#BACKWARD
for row_ in range(h_):
    for col_ in range(w_):
        #bilinear
        vec = np.dot(M_, np.array([[col_, row_, 1]]).T)
        c = vec[0,0]
        r = vec[1,0]
        c_left = int(c)
        c_right = min(int(c+1), w-1)
        r_top = int(r)
        r_bottom = min(int(r+1), h-1)
        s = c - c_left
        t = r - r_top
        intensity = (1-s) * (1-t) * src[r_top, c_left] \
                    + s * (1-t) * src[r_top, c_right] \
                    + (1-s) * t * src[r_bottom, c_left] \
                    + s * t * src[r_bottom, c_right]

        dst_back_bilinear[row_, col_] = intensity

dst_back_bilinear = dst_back_bilinear.astype(np.uint8)
```

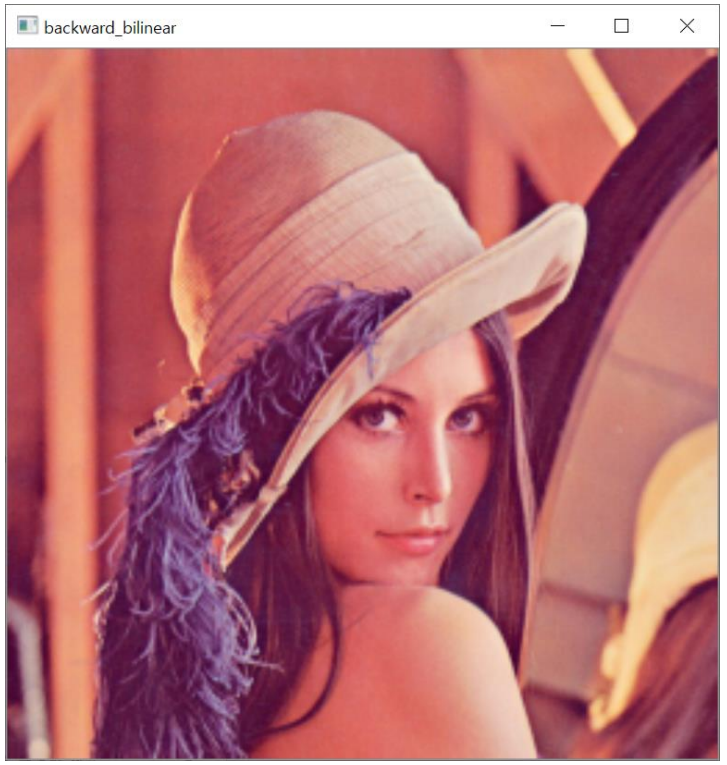
실습

- **Forward vs Backward**
  - 목표 Lena image를 2배 키우기(scaling)
    - bilinear interpolation



(256 x 256)

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x` \\ y` \\ 1 \end{bmatrix}$$



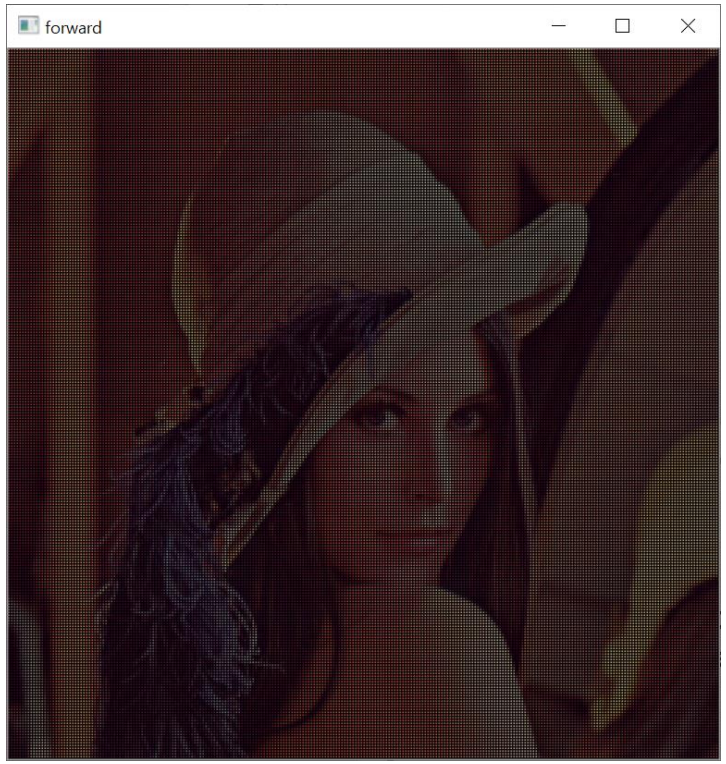
(512 x 512)

```
print('img')
print(src[:3, :3, 0]) #BGR 중 B만 참고

print('backward_bilinear')
print(dst_back_bilinear[:6, :6, 0]) #BGR 중 B만
```

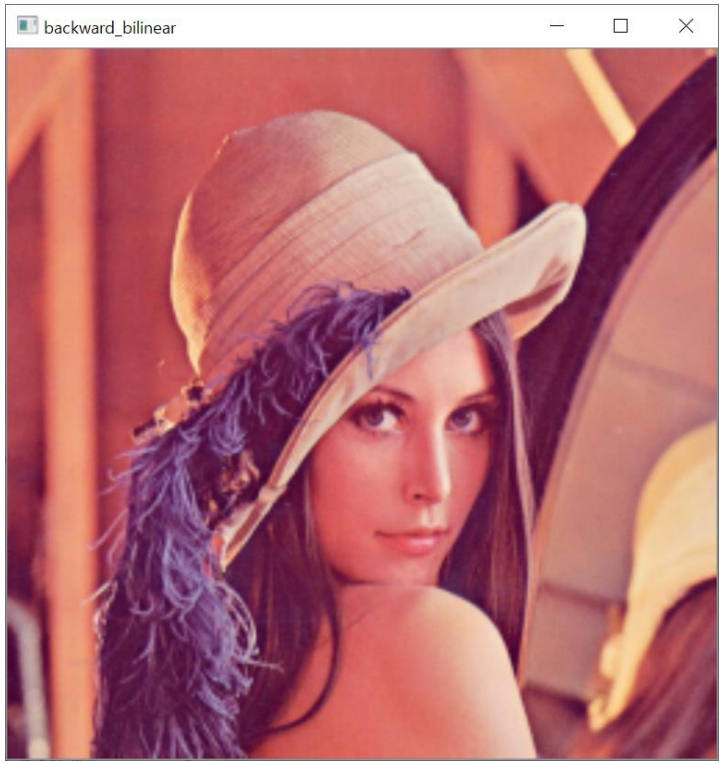
```
img
[[127 125 123]
 [126 126 123]
 [125 120 119]]
backward_bilinear
[[127 126 125 124 123 124]
 [126 126 125 124 123 123]
 [126 126 126 124 123 123]
 [125 124 123 122 121 121]
 [125 122 120 119 119 119]
 [121 119 117 117 117 116]]
```

실습

- **Forward vs Backward**



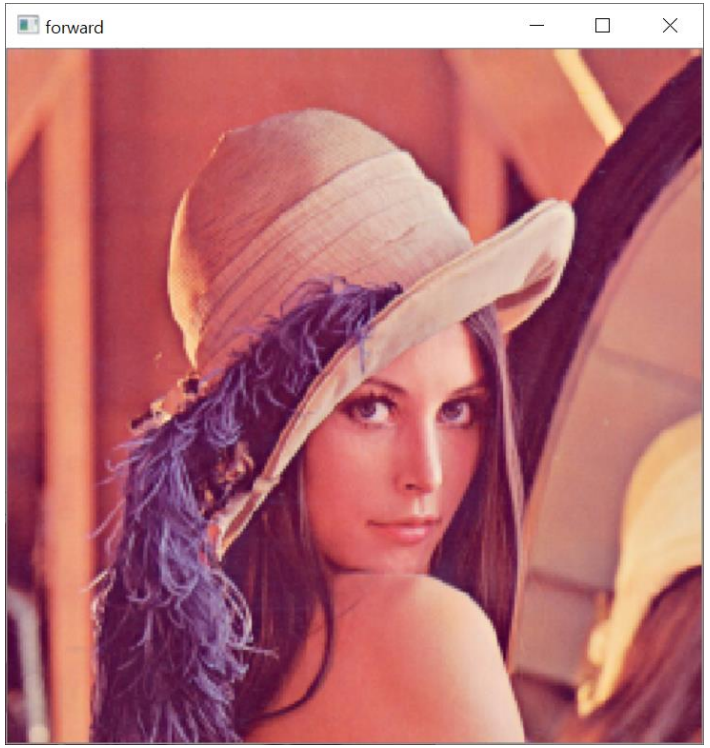
Forward



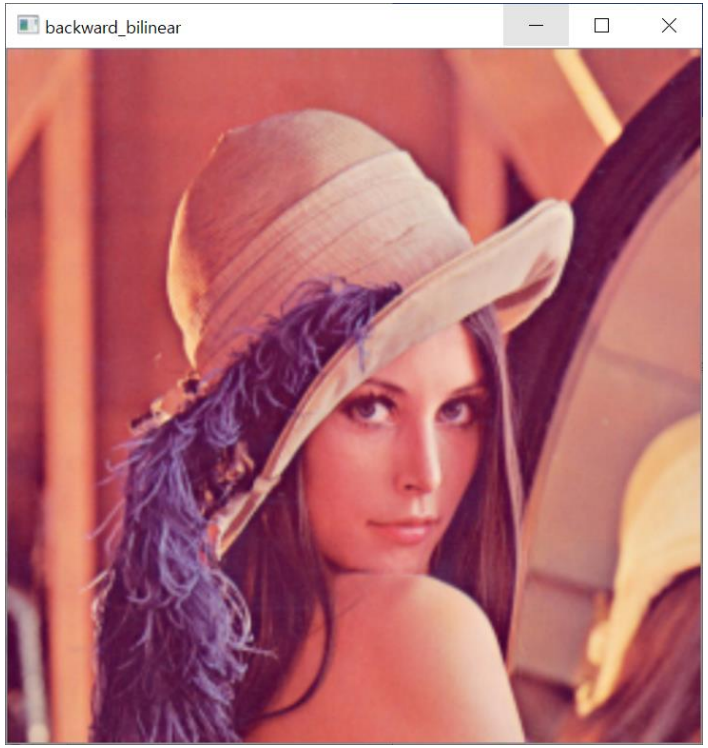
Backward(Bilinear)

실습

- **Forward vs Backward**
  - 2배 확대가 아니라 1.9배 확대의 경우



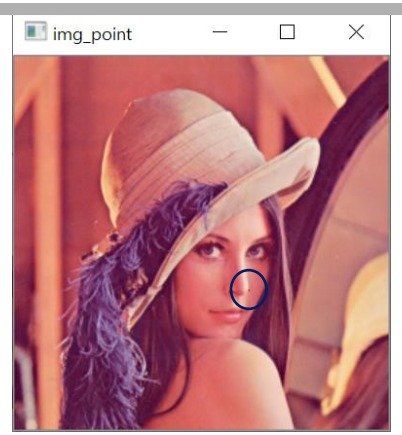
Forward

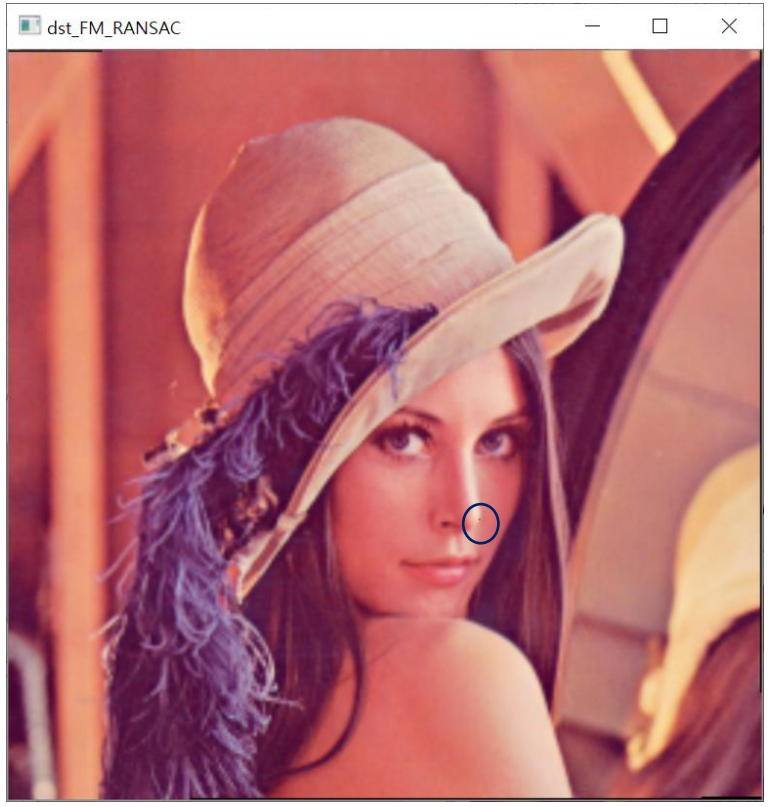


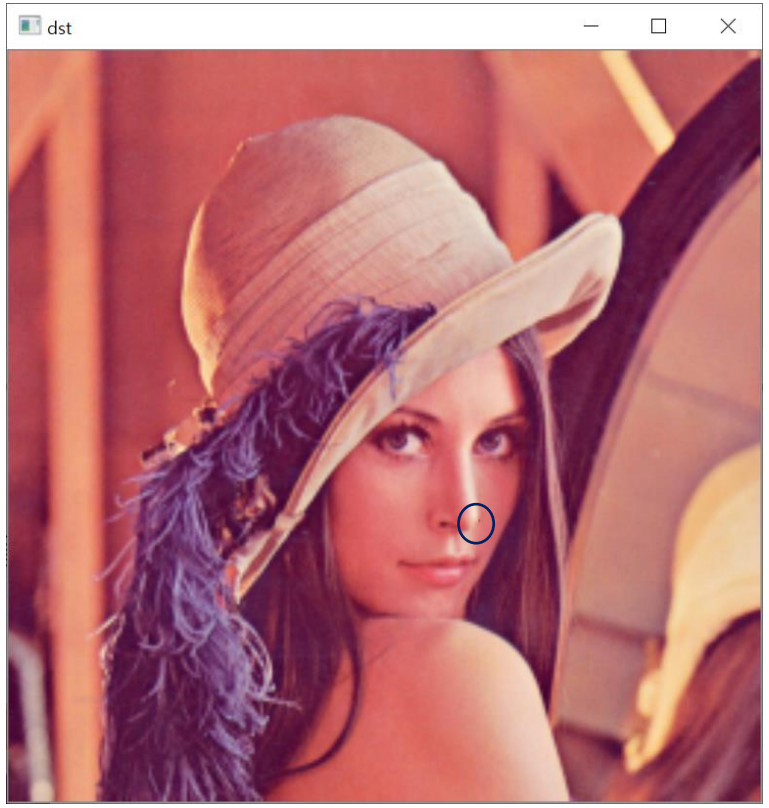
Backward(Bilinear)

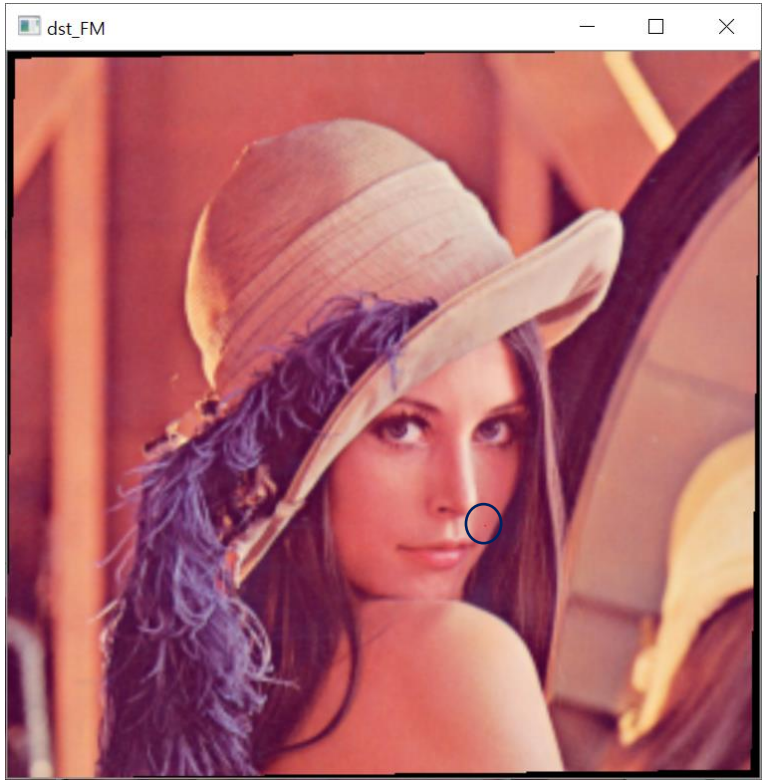
과제

- 행렬 M을 구한 후 점 찍어 보기
  - Lena.png(256x256) 의 160,160에 점 찍기









과제

- 행렬 M을 구한 후 점 찍어 보기
  - Lena.png(256x256) 의 160,160에 점 찍기

```
M
[[2 0 0]
 [0 2 0]
 [0 0 1]]
M 역행렬
[[0.5 0.  0. ]
 [0.  0.5 0. ]
 [0.  0.  1. ]]
```

```
No RANSAC M
[[ 1.94747178 -0.02368967 11.26673841]
 [-0.02221604  1.88221433 18.98483234]
 [ 0.          0.          1.        ]]
RANSAC M
[[ 2.00574896  0.0032471  -0.17763228]
 [-0.00574896  1.9967529   1.17763228]
 [ 0.          0.          1.        ]]
```

```
No RANSAC distance
point :  317 319
3.1622776601683795
Use RANSAC distance
point :  320 321
1.0
```

과제 - 구현

- 제공된 코드가 제대로 동작하도록 구현하기
  - 빈칸 채우기
  - 코드 채점 시 error가 발생하면 감점
  - cv2.imshow 할 때 꼭! 자신의 학번 보이도록 변경
  - distance와 점의 위치 등은 ppt에 첨부된 결과와 다를 수 있음

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(두 번째 page도 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 (6주차 과제도 이번 과제 제출 기한과 동일)**
  - 11월 10일 23시 59분까지 (최대 점수 10점)
- **추가 제출 기한 (6주차 과제도 이번 과제 제출 기한과 동일)**
  - 11월 17일 23시 59분까지 (최대 점수 4점, 과제 총점 계산 후 -6점)
  - 11월 18일 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