Image Processing

실습 5.

2021. 04. 04.

실습 수업 소개

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

- **TA 연락처**
  - 신준호
  - wnsgh578@naver.com

- **튜터 연락처**
  - 19 한승오
  - sh.h4ns@gmail.com

- **실습 중 질문사항**
  - 실시간 수업중 질문 or 메일을 통한 질문
  - 메일로 질문할 때 [IP] 를 제목에 붙여주세요

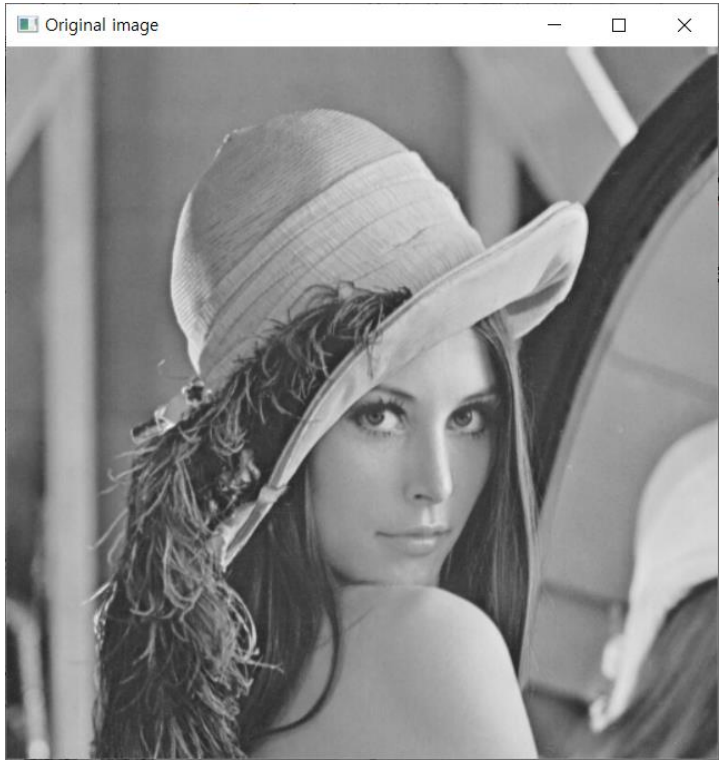
실습 수업 소개

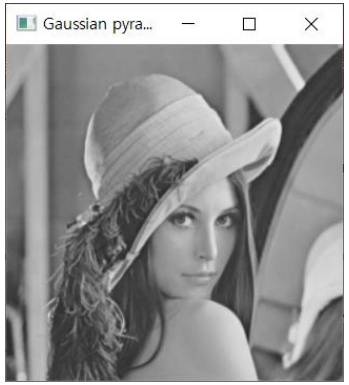
- **실습 출석**
  - 사이버캠퍼스를 통해 Zoom 출석
  - Zoom 퇴장 전 채팅 기록[학번 이름] 남기고 퇴장
  - 위 두 기록을 통해 출석 체크 진행 예정

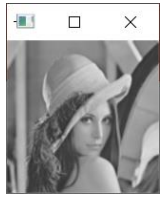
목 차

Image up&down sampling

- **Image pyramid**

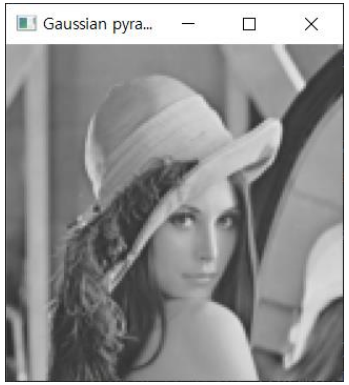


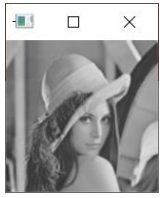


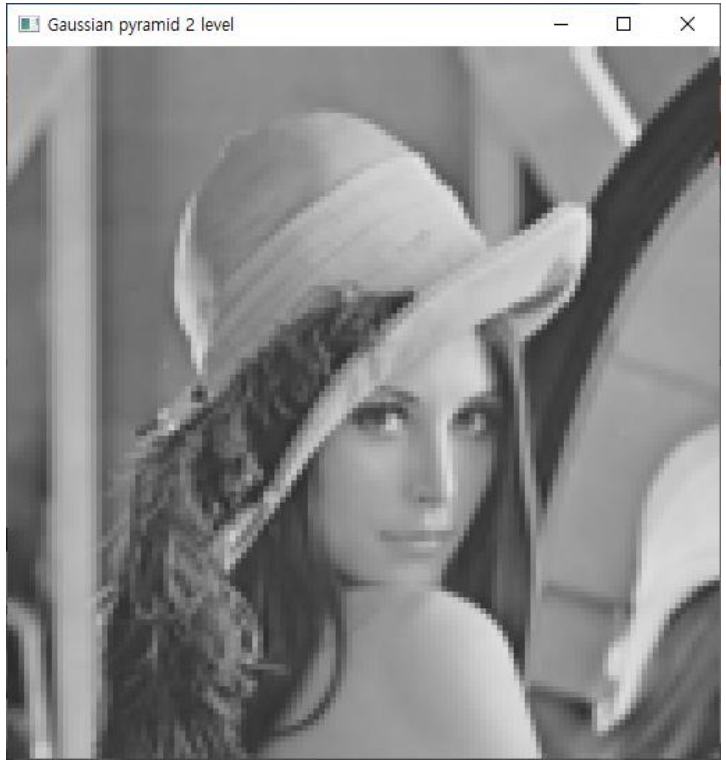


downsampling

Gaussian filter size : 3x3
sigma = 1





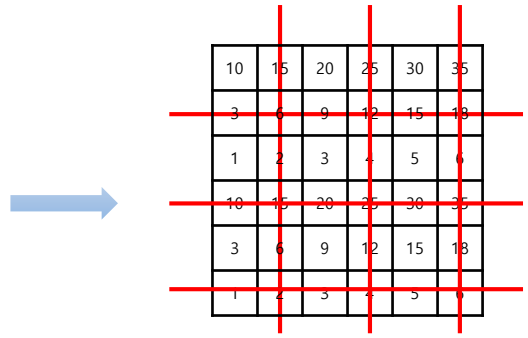


upsampling

Image up&down sampling

- **Gaussian image pyramid downsampling**
  - 이미지에 gaussian filter 적용
  - blur된 이미지에서 가로와 세로를 ½씩 줄임
  - 짝수 번째 행과 열을 지움

10	15	20	25	30	35
3	6	9	12	15	18
1	2	3	4	5	6
10	15	20	25	30	35
3	6	9	12	15	18
1	2	3	4	5	6



10	20	30
1	3	5
3	9	15

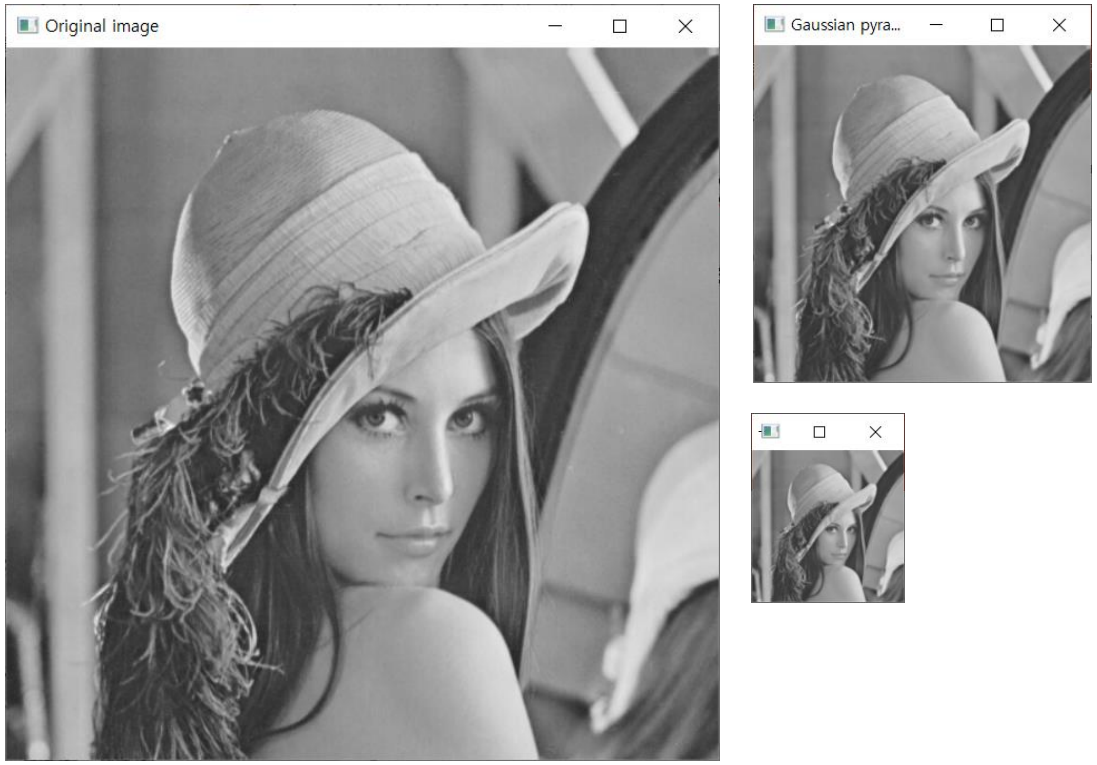


Image up&down sampling

- **Image pyramid upsampling: Nearest interpolation**
  - 이미지에서 가로와 세로를 2(혹은 n)배 늘림
  - 픽셀값을 반복해서 빈 곳을 채움

10	20	30
1	3	5
3	9	15

10		20		30	
1		3		5	
3		9		15	

10	10	20	20	30	30
10	10	20	20	30	30
1	1	3	3	5	5
1	1	3	3	5	5
3	3	9	9	15	15
3	3	9	9	15	15

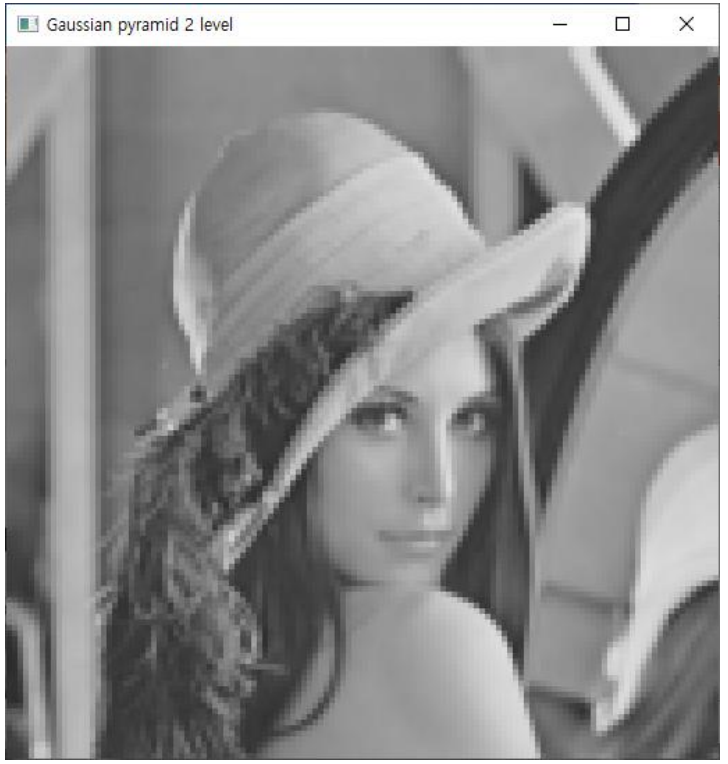
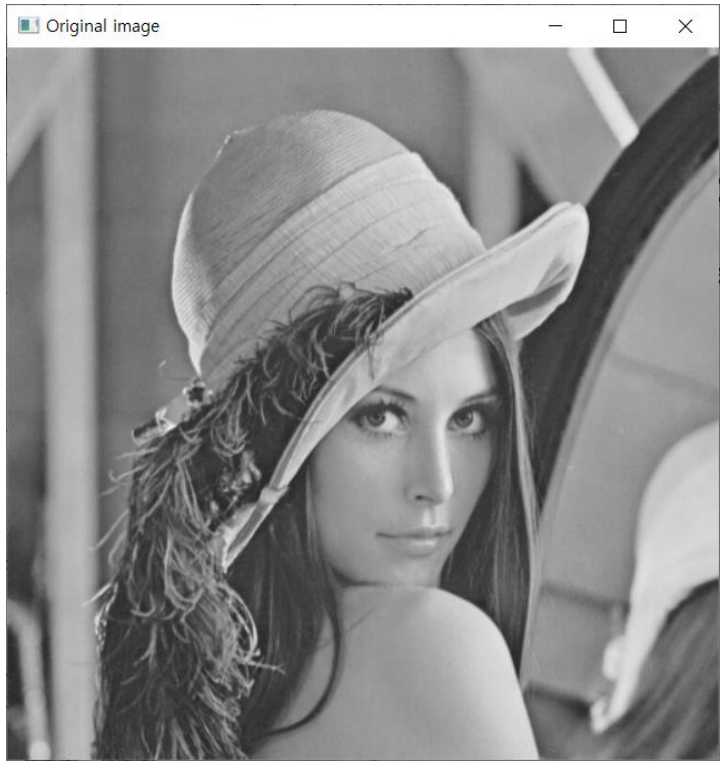
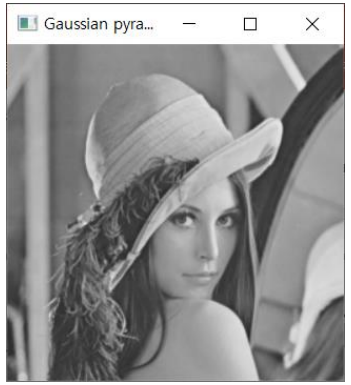


Image up&down sampling

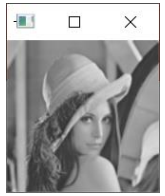
- **Gaussian image pyramid**



Original image



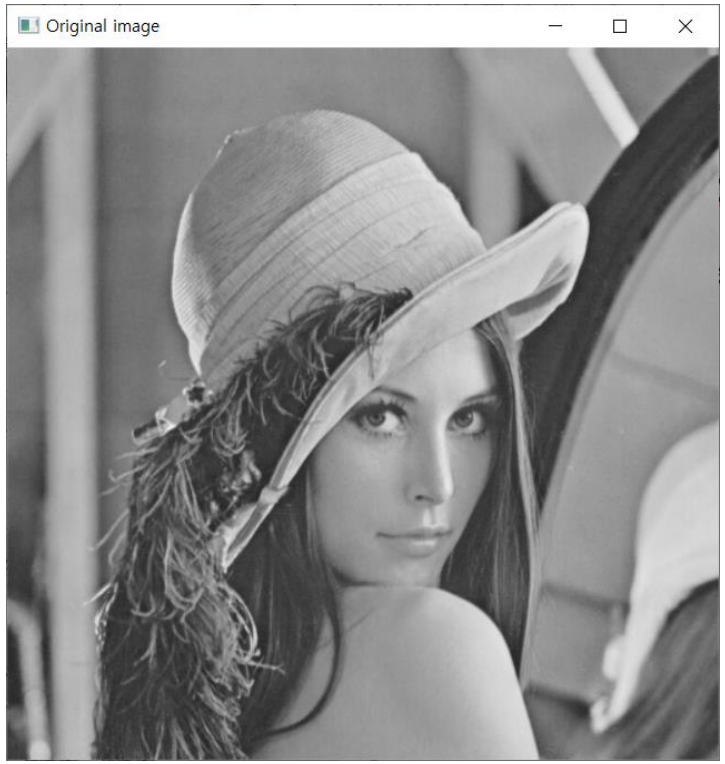
level 1



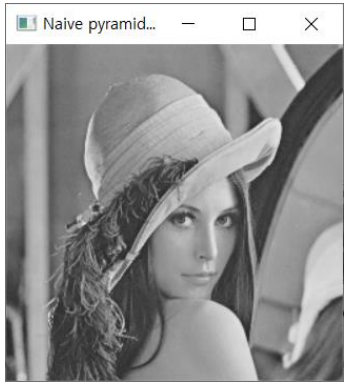
level 2

Image up&down sampling

- **Naïve image pyramid**



Original image

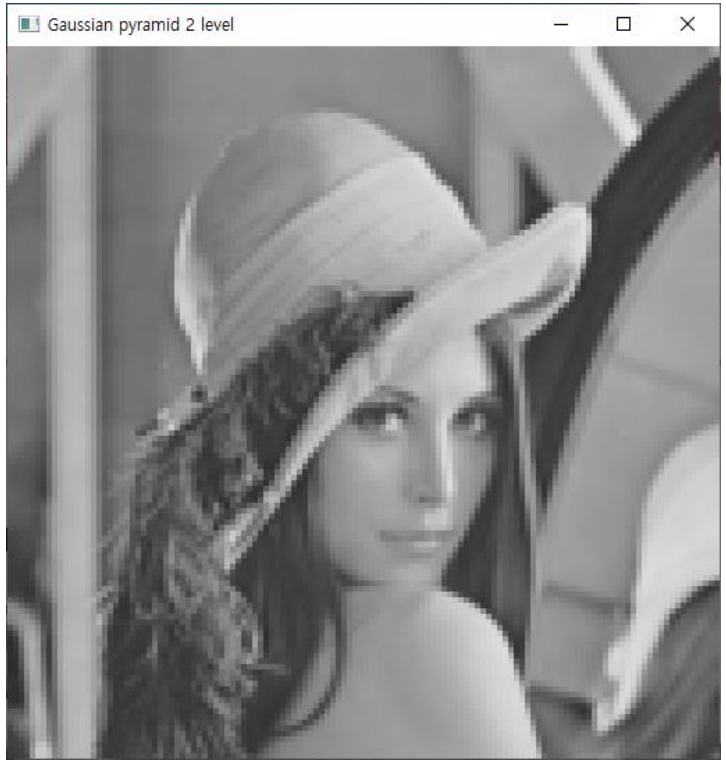


level 1

level 2

Image up&down sampling

- 비교를 위해 Nearest neighbor interpolation으로 확대 후의 차이



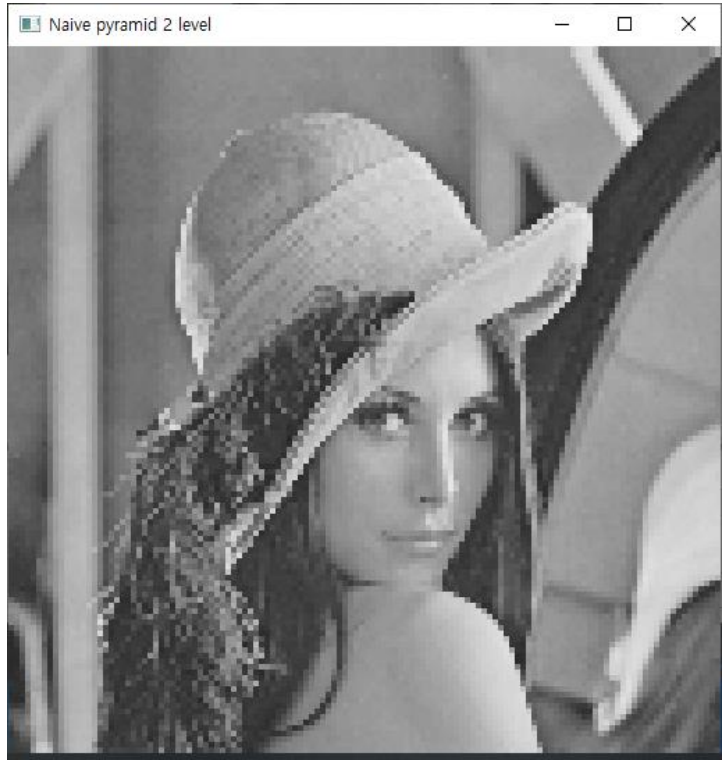


Image up&down sampling

- **Downsample과 upsample을 진행한 후 차이 비교**

```python
if __name__ == '__main__':
    img = cv2.imread('Lenna.png', cv2.IMREAD_GRAYSCALE)
    original_shape = img.shape

    gaussian_pyramid = my_gaussian_pyramid(img, ratio=2, pyramid_len=2, filter_size=3, sigma=1)
    naive_pyramid = my_naive_pyramid(img, ratio=2, pyramid_len=2)

    cv2.imshow('Original image', img)
    for level, g_img in enumerate(gaussian_pyramid[1:]):
        cv2.imshow('Gaussian pyramid {} level'.format(level + 1),
                   g_img)
    for level, n_img in enumerate(naive_pyramid[1:]):
        cv2.imshow('Naive pyramid {} level'.format(level + 1),
                   n_img)

    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Image up&down sampling

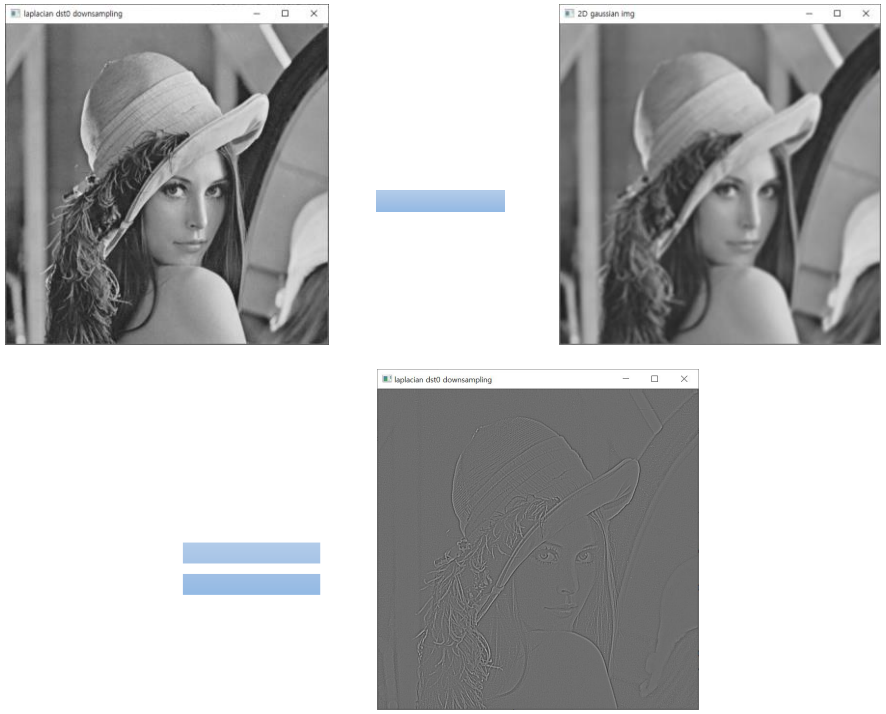
- **Gaussian, Naïve pyramid**

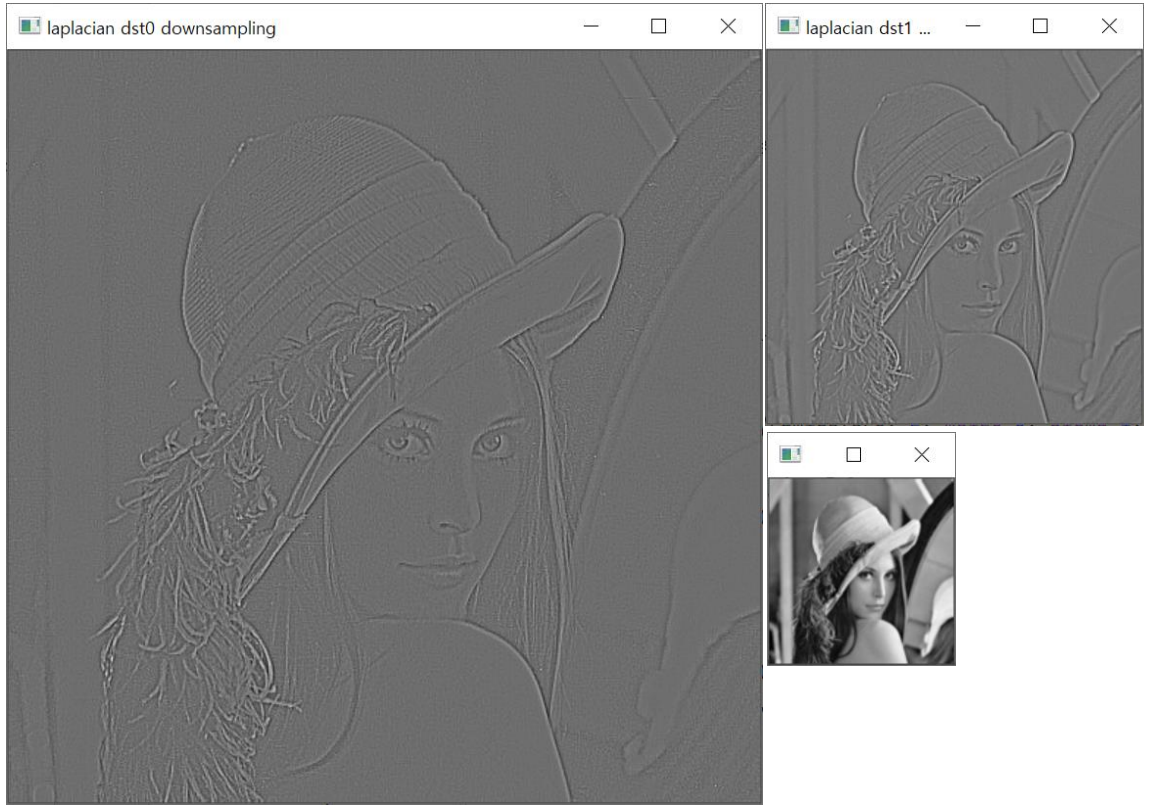
```python
def my_gaussian_downsampling(src, ratio, mask):
    blurred_src = cv2.filter2D(src, -1, mask)
    downsampled_blurred_src = blurred_src[::ratio, ::ratio]
    return downsampled_blurred_src

def my_gaussian_pyramid(src, ratio, pyramid_len, filter_size, sigma):
    pyramid = [src]
    mask = cv2.getGaussianKernel(ksize=filter_size, sigma=sigma)
    mask = mask @ mask.T

    for p in range(pyramid_len):
        pyramid.append(my_gaussian_downsampling(pyramid[-1], ratio, mask))

    return pyramid[1:]
```

```python
def my_naive_downsampling(src, ratio):
    return src[::ratio, ::ratio]

def my_naive_pyramid(src, ratio, pyramid_len):
    pyramid = [src]
    for p in range(pyramid_len):
        pyramid.append(my_naive_downsampling(pyramid[-1], ratio))

    return pyramid[1:]
```

Image up&down sampling

- **Laplacian image pyramid downsampling**
  - 원본 이미지에 gaussian filter 적용
  - 원본 이미지 - blur된 이미지 => residual 로 저장
  - blur된 이미지에서 가로와 세로를 ½씩 줄임
  - 짝수 번째 행과 열을 지움

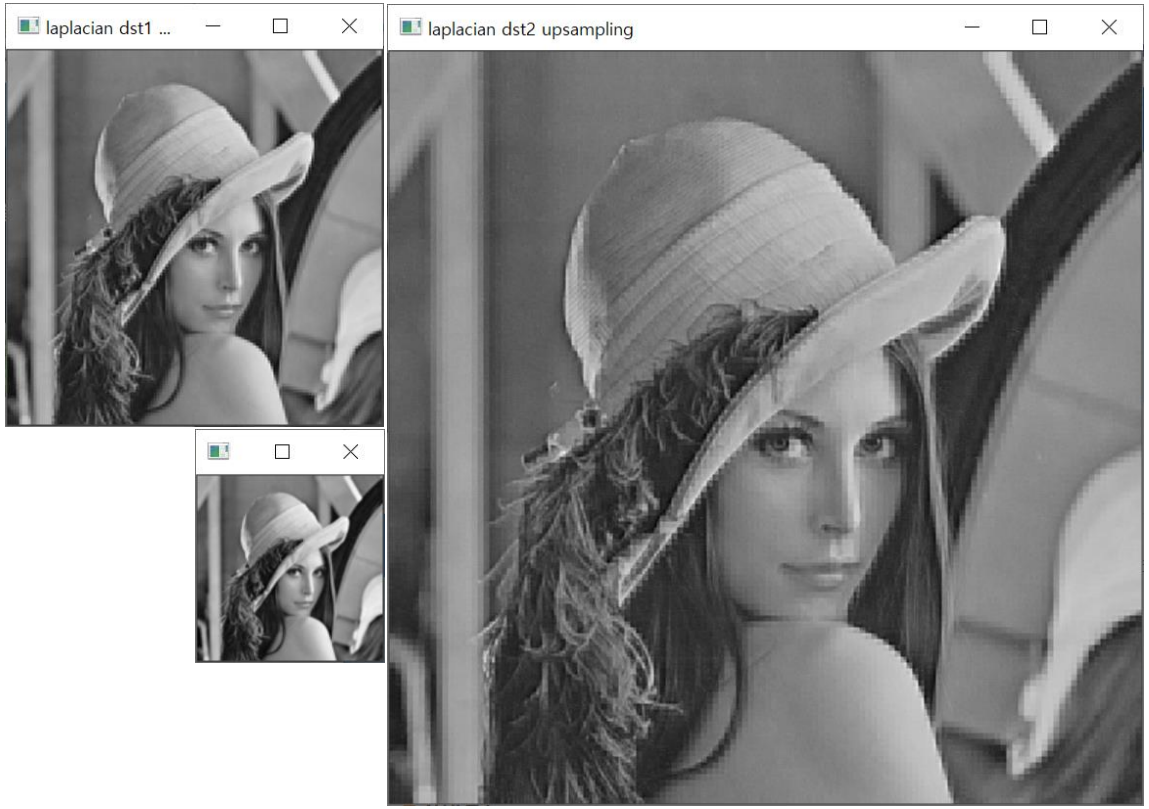




downsampling

Image up&down sampling

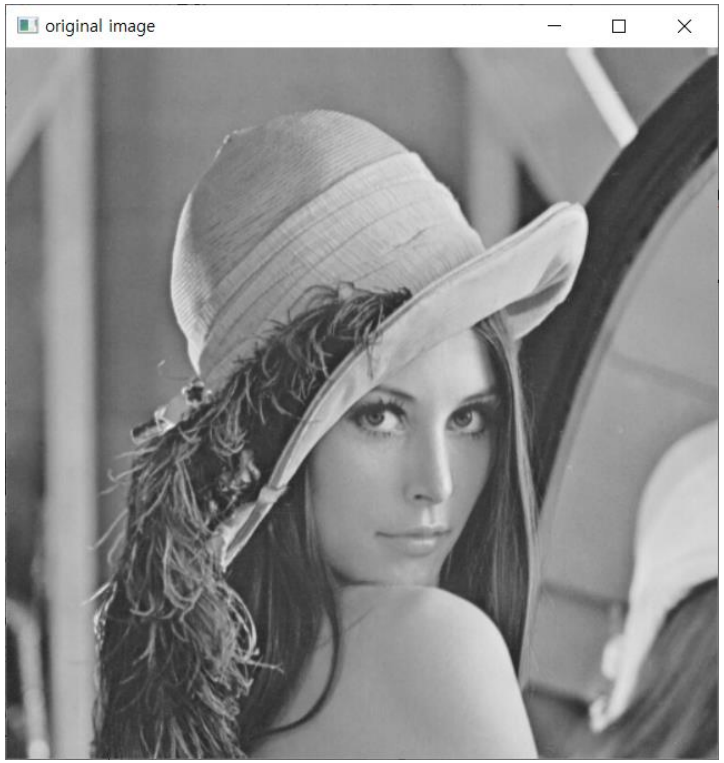
- **Laplacian image pyramid upsampling**
  - 원본 이미지에서 가로와 세로를 2배 늘림
  - 픽셀값을 반복해서 빈 곳을 채움
  - residual을 이미지에 더함



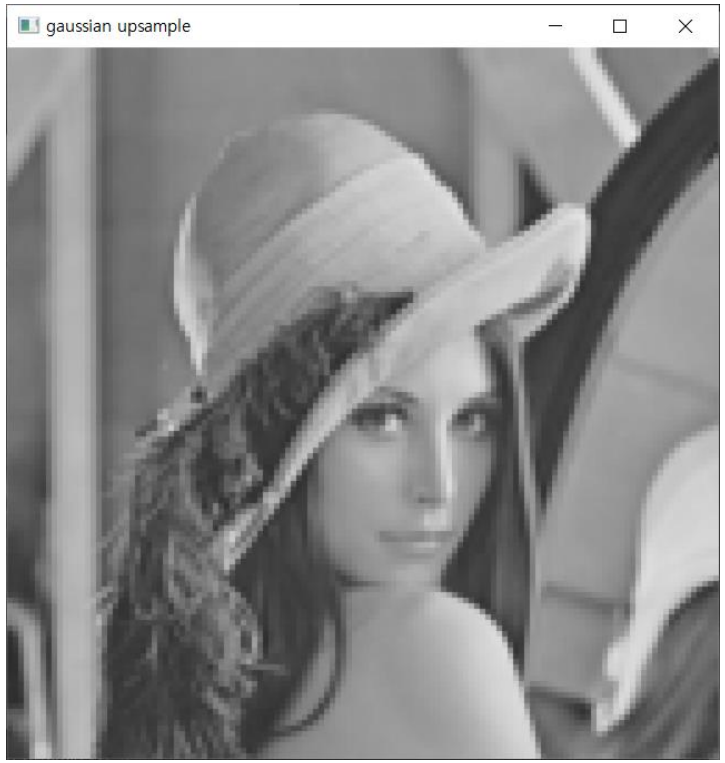
upsampling

Image up&down sampling

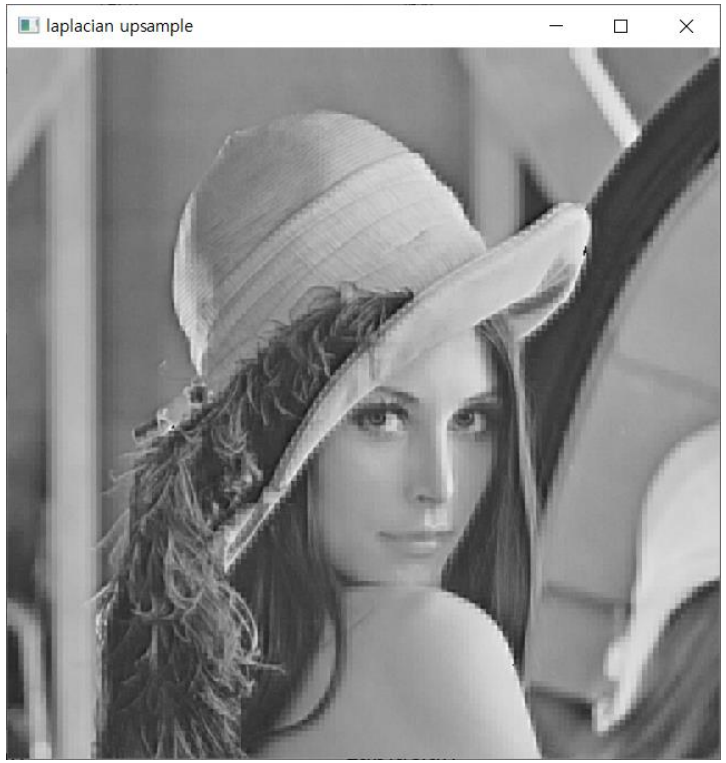
- **Gaussian vs Laplacian**



Original



Gaussian



Laplacian

Image up&down sampling

- **Gaussian vs Laplacian**

```
if __name__ == '__main__':
    img = cv2.imread('Lenna.png', cv2.IMREAD_GRAYSCALE).astype('int')
    original_shape = img.shape

    laplacian = my_laplacian_pyramid(img, ratio=2, pyramid_len=2, filter_size=3, sigma=1)

    laplacian_up = my_laplacian_upsample(laplacian, ratio=2)

    cv2.imshow('original image', img.astype('uint8'))
    cv2.imshow('laplacian upsample', laplacian_up.astype('uint8'))
    cv2.imshow('gaussian upsample', cv2.resize(laplacian[-1],
                    dsize=(0, 0), fx=2*2, fy=2*2, interpolation=cv2.INTER_NEAREST).astype('uint8'))

    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Image up&down sampling

- **Gaussian vs Laplacian**

```python
def my_laplacian_pyramid(src, ratio, pyramid_len, filter_size, sigma):
    laplacian = [src]

    mask = cv2.getGaussianKernel(ksize=filter_size, sigma=sigma)
    mask = mask @ mask.T

    for p in range(pyramid_len):
        last_img = laplacian.pop()

        filtered_last_img = cv2.filter2D(last_img.astype('uint8'), -1, mask).astype('int')
        residual_img = last_img - filtered_last_img

        laplacian.append(residual_img)
        laplacian.append(filtered_last_img[::ratio, ::ratio])

    return laplacian
```

Image up&down sampling

- Gaussian vs Laplacian

```
def my_laplacian_upsample(laplacian, ratio):
    img = laplacian[-1]

    for l in laplacian[-2::-1]:
        img_up = cv2.resize(img, dsize=(0, 0), fx=ratio, fy=ratio, interpolation=cv2.INTER_NEAREST)
        img_up += l
        img = img_up

    return img
```

Image up&down sampling - Interpolation

- **nearest-neighbor Interpolation**
  - 이미지의 크기를 변경하거나 회전할 때 가장 가까운 지점의 픽셀값을 이용

- **bilinear Interpolation**
  - 값이 변할 때 선형적인 변화가 있을 것 이라는 가정 하에 픽셀값을 예측

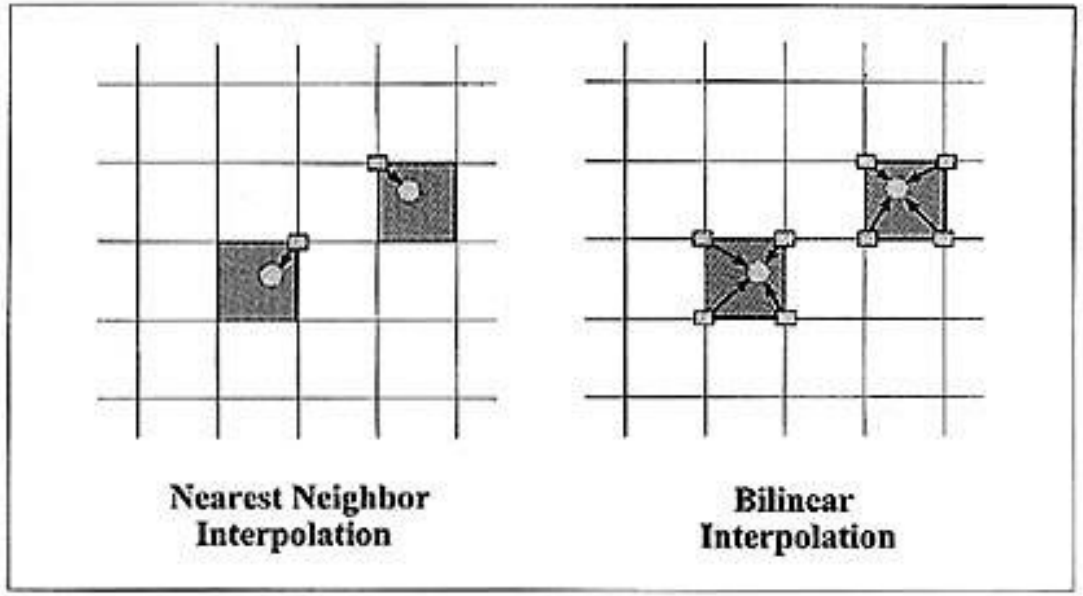
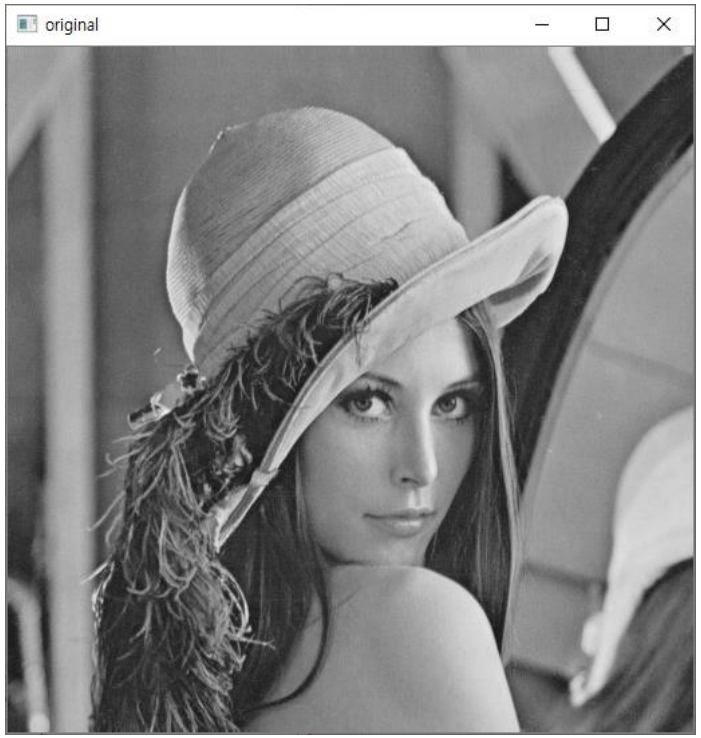


Image up&down sampling - Interpolation

- **nearest-neighbor Interpolation**



original

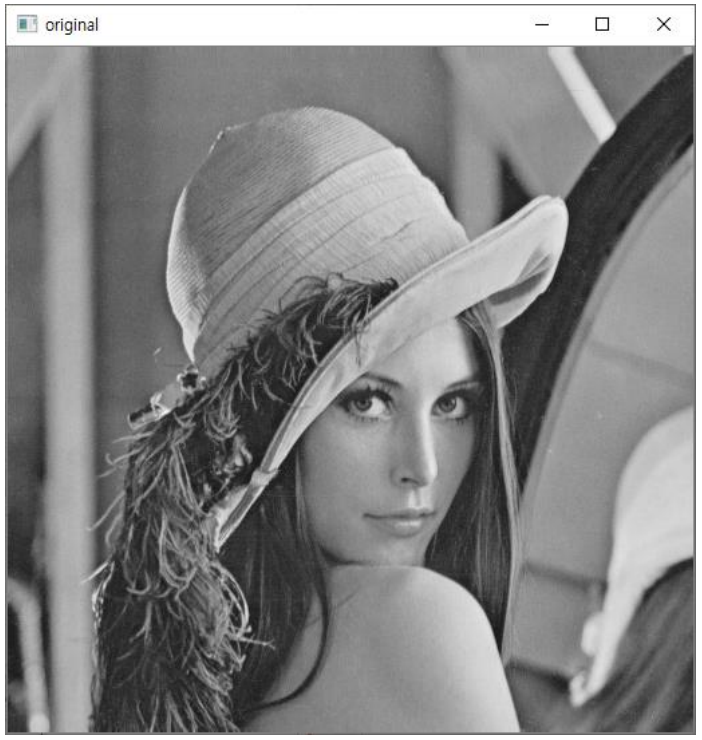
cv2.resize()

my_nearest_neighbor()

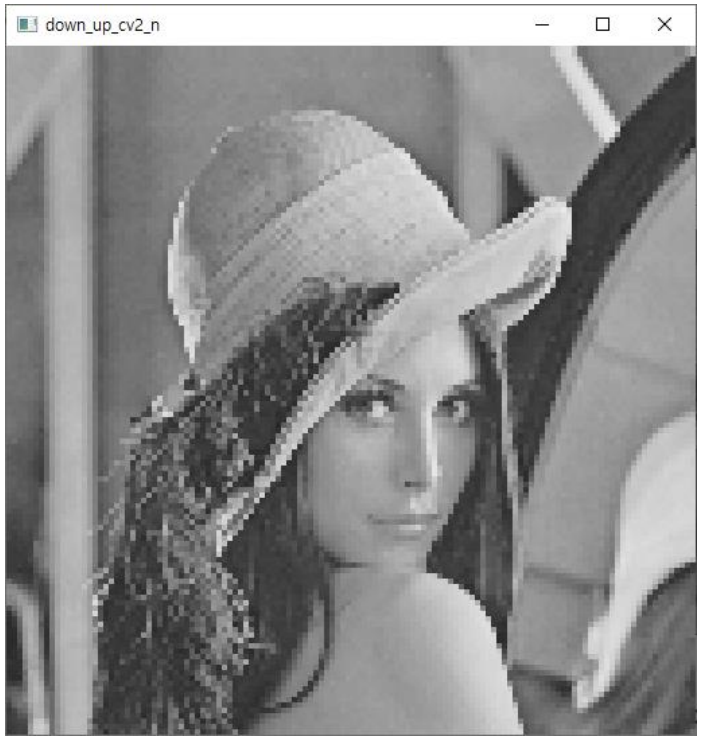
row, col 1/4로 줄임

Image up&down sampling - Interpolation

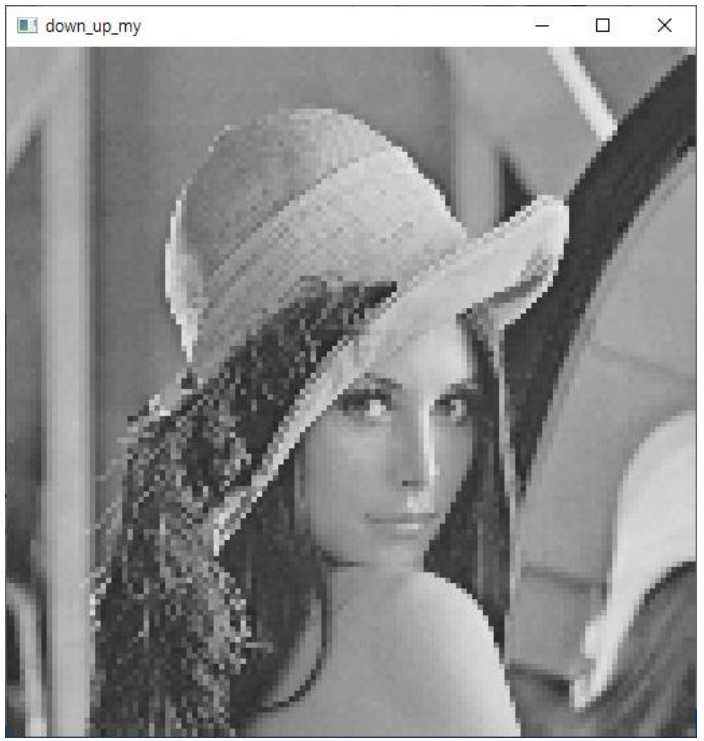
- **nearest-neighbor Interpolation**



original



cv2.resize()



my_nearest_neighbor()

row, col 1/2로 줄인 후 다시 2배 늘림

Image up&down sampling - Interpolation

- **nearest-neighbor Interpolation**

```python
if __name__ == '__main__':
    img = cv2.imread('Lenna.png', cv2.IMREAD_GRAYSCALE)

    down_cv2_n = cv2.resize(img, dsize=(0, 0), fx=0.5, fy=0.5, interpolation=cv2.INTER_NEAREST)
    down_up_cv2_n = cv2.resize(down_cv2_n, dsize=(0, 0), fx=2.0, fy=2.0, interpolation=cv2.INTER_NEAREST)

    down_my = my_resize_nearest_interpolation(img, scale=0.5)
    down_up_my = my_resize_nearest_interpolation(down_my, scale=2.0)

    cv2.imshow('original image', img)
    cv2.imshow('down_cv2_n image', down_cv2_n)
    cv2.imshow('down_up_cv2_n', down_up_cv2_n)
    cv2.imshow('down_my', down_my)
    cv2.imshow('down_up_my', down_up_my)

    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

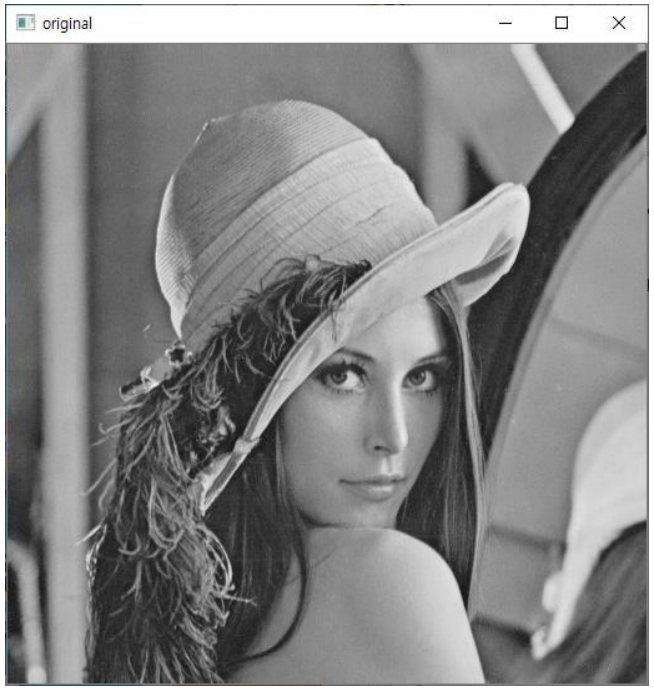
Image up&down sampling - Interpolation

- **nearest-neighbor Interpolation**

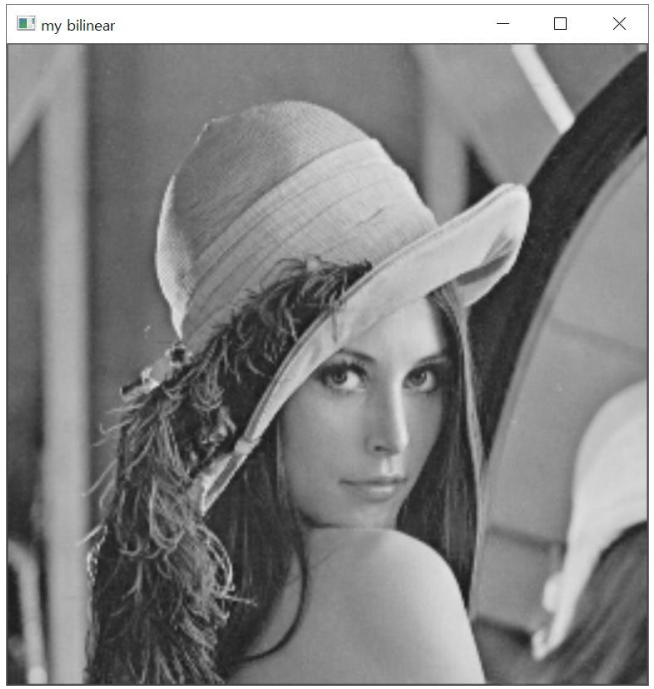
```
def my_resize_nearest_interpolation(src, scale):
    (h, w) = src.shape
    h_dst = int(h * scale + 0.5)
    w_dst = int(w * scale + 0.5)

    dst = np.zeros((h_dst, w_dst), np.uint8)
    for row in range(h_dst):
        for col in range(w_dst):
            r = min(int(row / scale + 0.5), h-1)
            c = min(int(col / scale + 0.5), w-1)
            dst[row, col] = src[r, c]

    return dst
```

과제(Bilinear Interpolation)

- **Bilinear Interpolation**



original

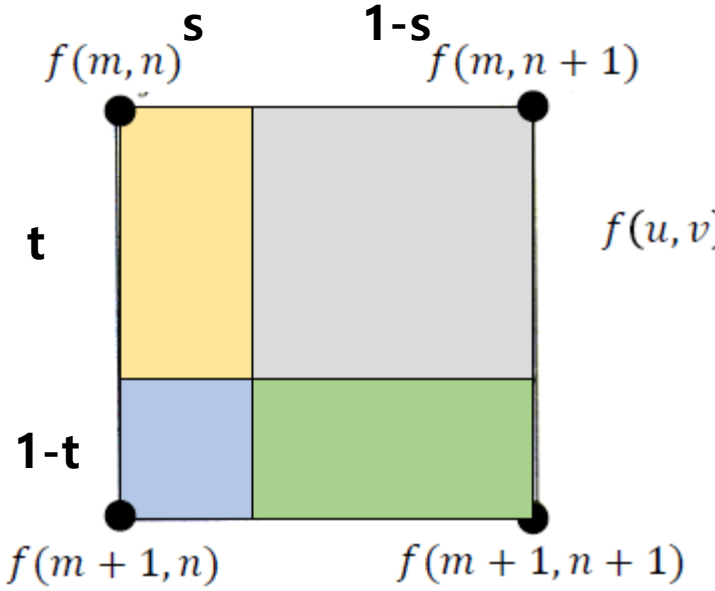


1/2 -> 원상복구

1/7 -> 원상복구

과제(Bilinear Interpolation)

- **Bilinear Interpolation**



$$f(u,v) = (1-s)(1-t)\cdot f(m,n) + s(1-t)\cdot f(m,n+1) + (1-s)t\cdot f(m+1,n) + st\cdot f(m+1,n+1)$$

```python
def my_bilinear(src, scale):
    (h, w) = src.shape
    h_dst = int(h * scale + 0.5)
    w_dst = int(w * scale + 0.5)

    dst = np.zeros((h_dst, w_dst), np.uint8)

    ###############################################
    # TODO                                        #
    # my_bilinear 완성                            #
    ###############################################
    for row in range(h_dst):
        for col in range(w_dst):
            dst[row, col] = ???

    return dst
```

과제

- **my_bilinear 함수 작성**
  - my_bilinear의 dst 행렬을 완성시켜 리턴
  - 보고서에 cv2의 bilinear와 결과에 큰 차이가 있는지 비교 및 설명

과제

- **제출 방법**
  - 코드 파일
    - 구현 결과가 포함된 python 파일(.py)
  - 보고서
    - [IP]201900000_홍길동_2주차_과제.pdf
    - 보고서 양식 사용
    - PDF 파일 형식으로 제출(pdf가 아닌 다른 양식으로 제출시 감점)
  - 제출 파일
    - [IP]201900000_홍길동_2주차_과제.zip
    - .py 파일과 pdf 보고서를 하나의 파일로 압축한 후, 양식에 맞는 이름으로 제출

출석체크

- **Zoom 퇴장 전, [학번 이름]을 채팅창에 올린 후 퇴장해 주시기 바랍니다.**

QnA