머신 비전 시스템 과제 2

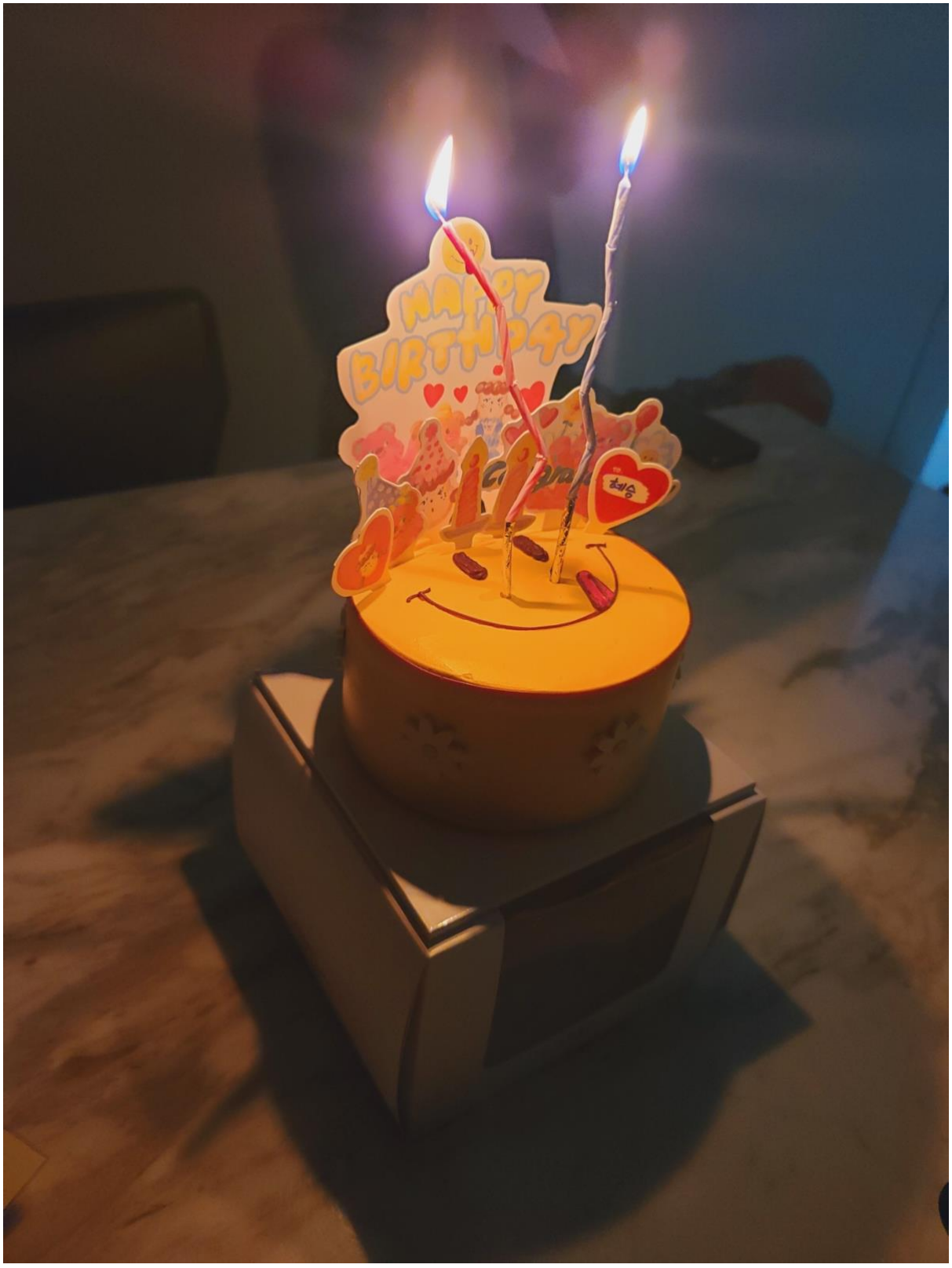
조혜수

1.
(1) Average filtering

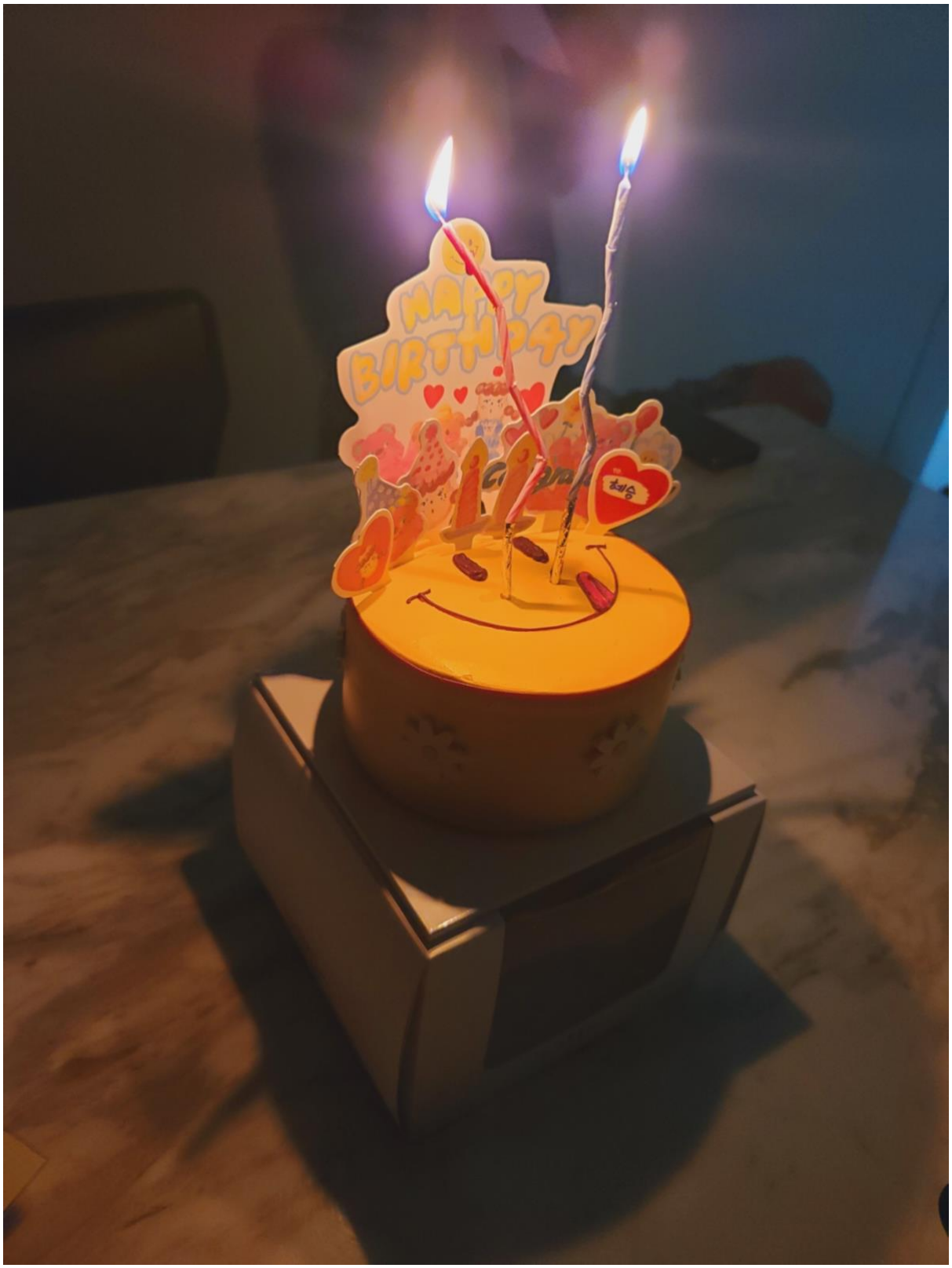
원본영상

결과 영상

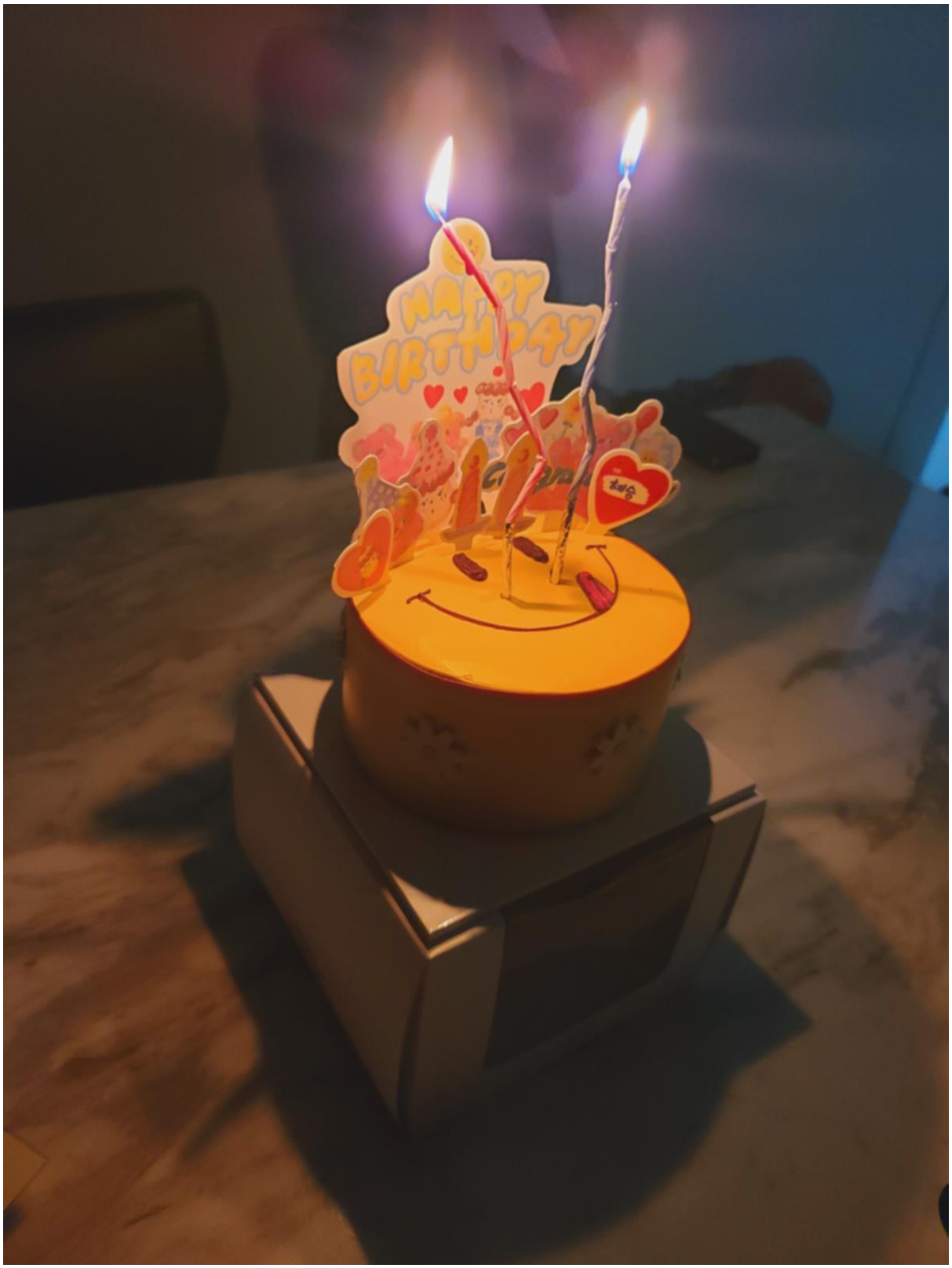
kernel 크기 3x3



kernel 크기 5x5



kernel 크기 7x7



코드 (캡쳐본, 텍스트)

< Average filtering >

- kernel 3x3
- kernel 5x5
- kernel 7x7

```
import cv2 as cv
from google.colab.patches import cv_imshow
original = cv.imread('/content/gdrive/MyDrive/MachineVision/OriginalIMG.jpeg')

blur_3 = cv.blur(original, ksize=(3,3))
blur_5 = cv.blur(original, ksize=(5,5))
blur_7 = cv.blur(original, ksize=(7,7))
cv_imshow(original)
cv_imshow(blur_3)
cv_imshow(blur_5)
cv_imshow(blur_7)
```

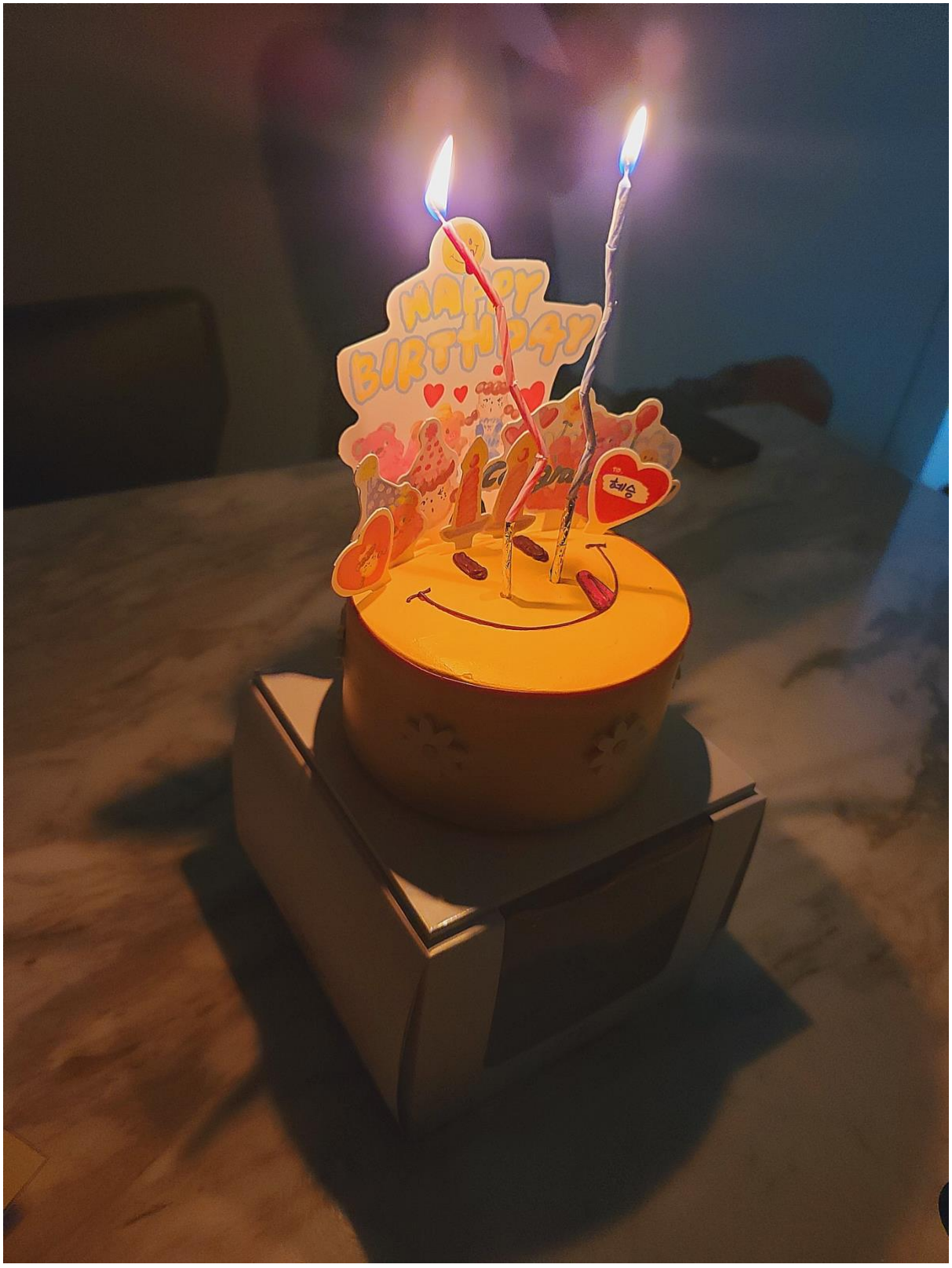

```
import cv2 as cv
from google.colab.patches import cv_imshow
original = cv.imread('/content/gdrive/MyDrive/MachineVision/OriginalIMG.jpeg')

blur_3 = cv.blur(original, ksize=(3,3))
blur_5 = cv.blur(original, ksize=(5,5))
blur_7 = cv.blur(original, ksize=(7,7))
cv_imshow(original)
cv_imshow(blur_3)
cv_imshow(blur_5)
cv_imshow(blur_7)
```

(2) Image Sharpening

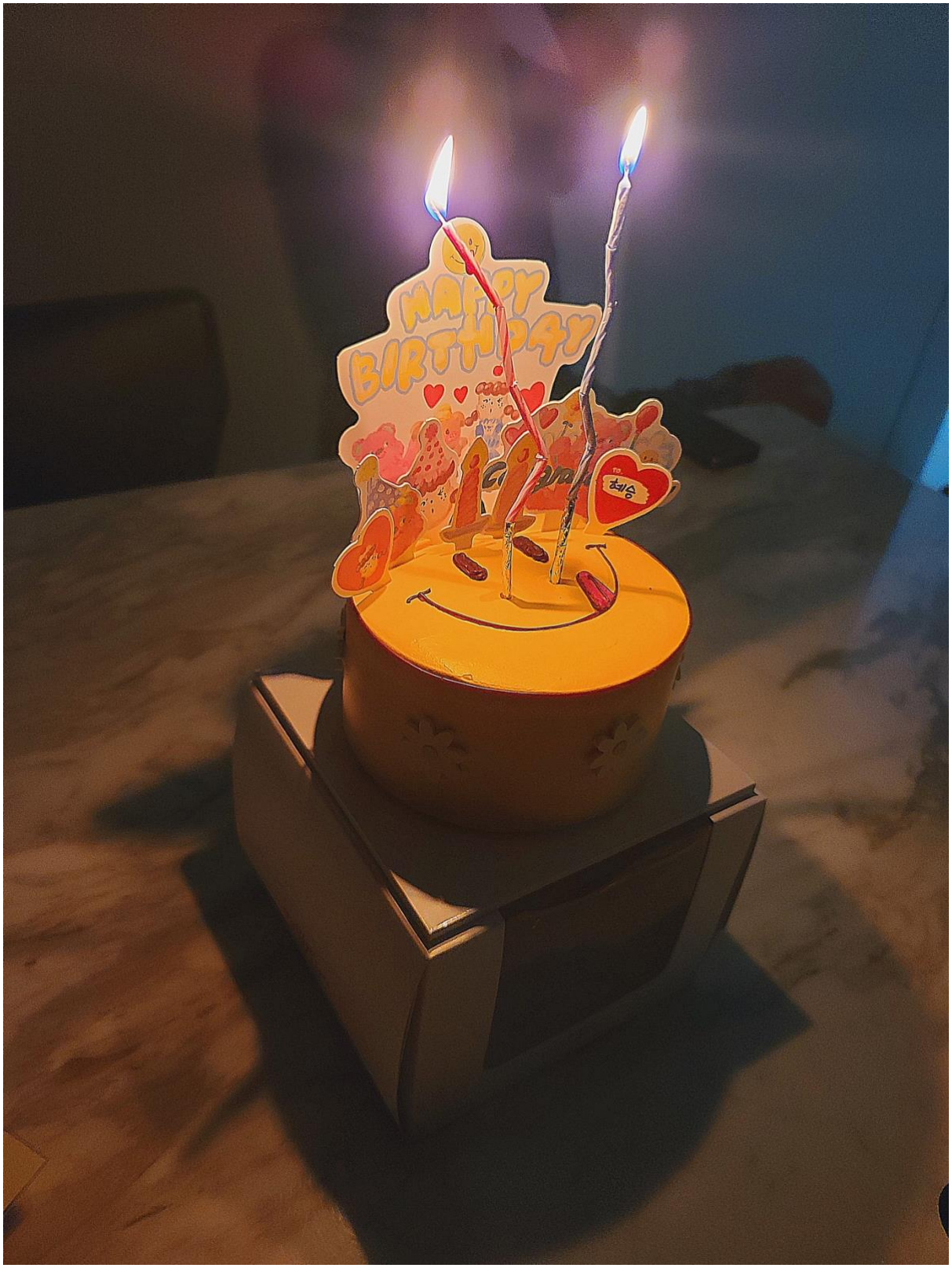
원본 영상 (1 에서의 사진과 동일)

결과 영상

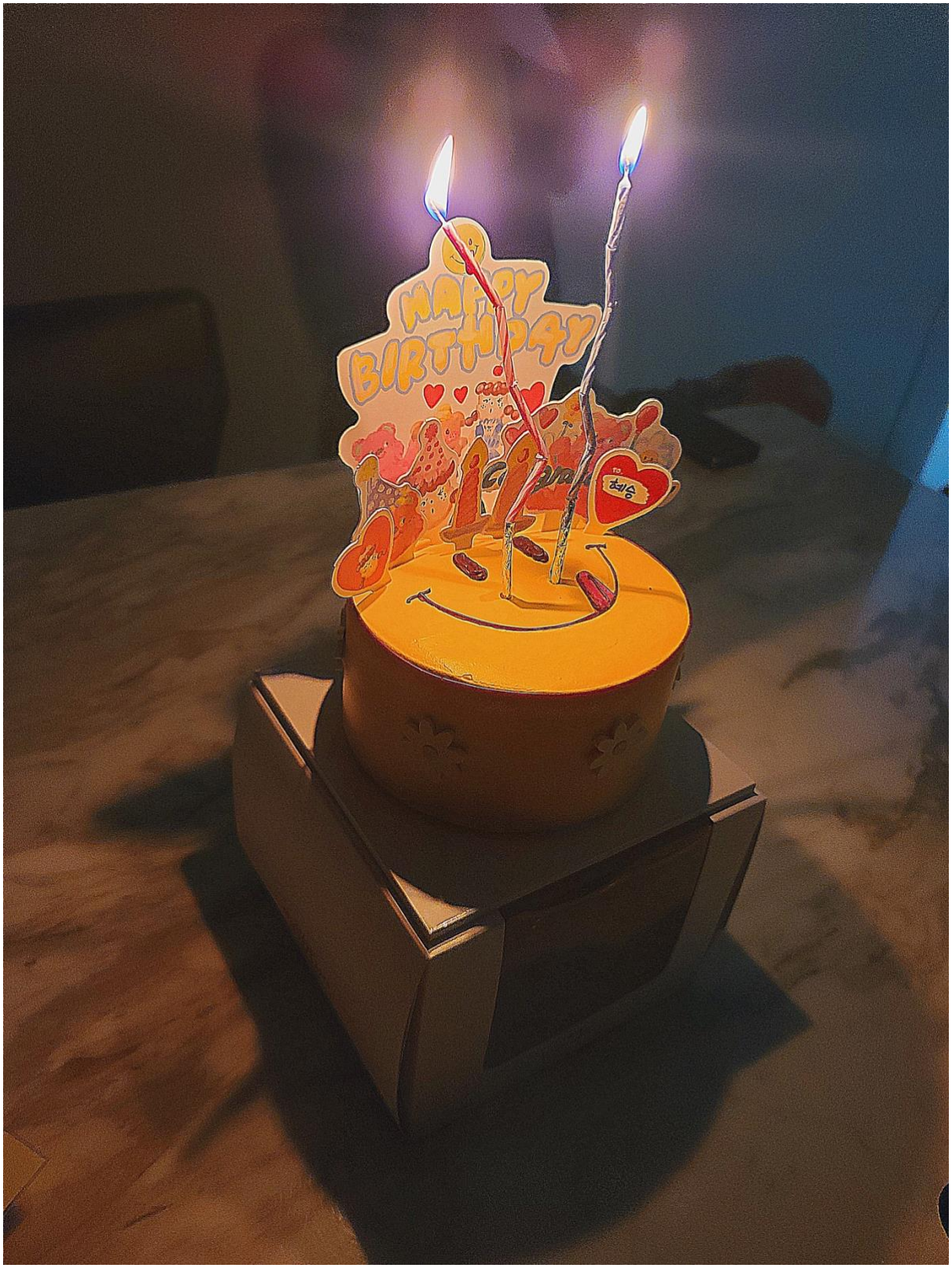
a=2



a=5



a=10



코드 (캡쳐본, 텍스트)

< Image Sharpening >

a = 2, 5, 10

```
import numpy as np

detail = np.int32(original) - np.int32(blur_7)
shapened_2 = np.int32(original) + 2*detail
shapened_5 = np.int32(original) + 5*detail
shapened_10 = np.int32(original) + 10*detail

cv_imshow(shapened_2)
cv_imshow(shapened_5)
cv_imshow(shapened_10)
```

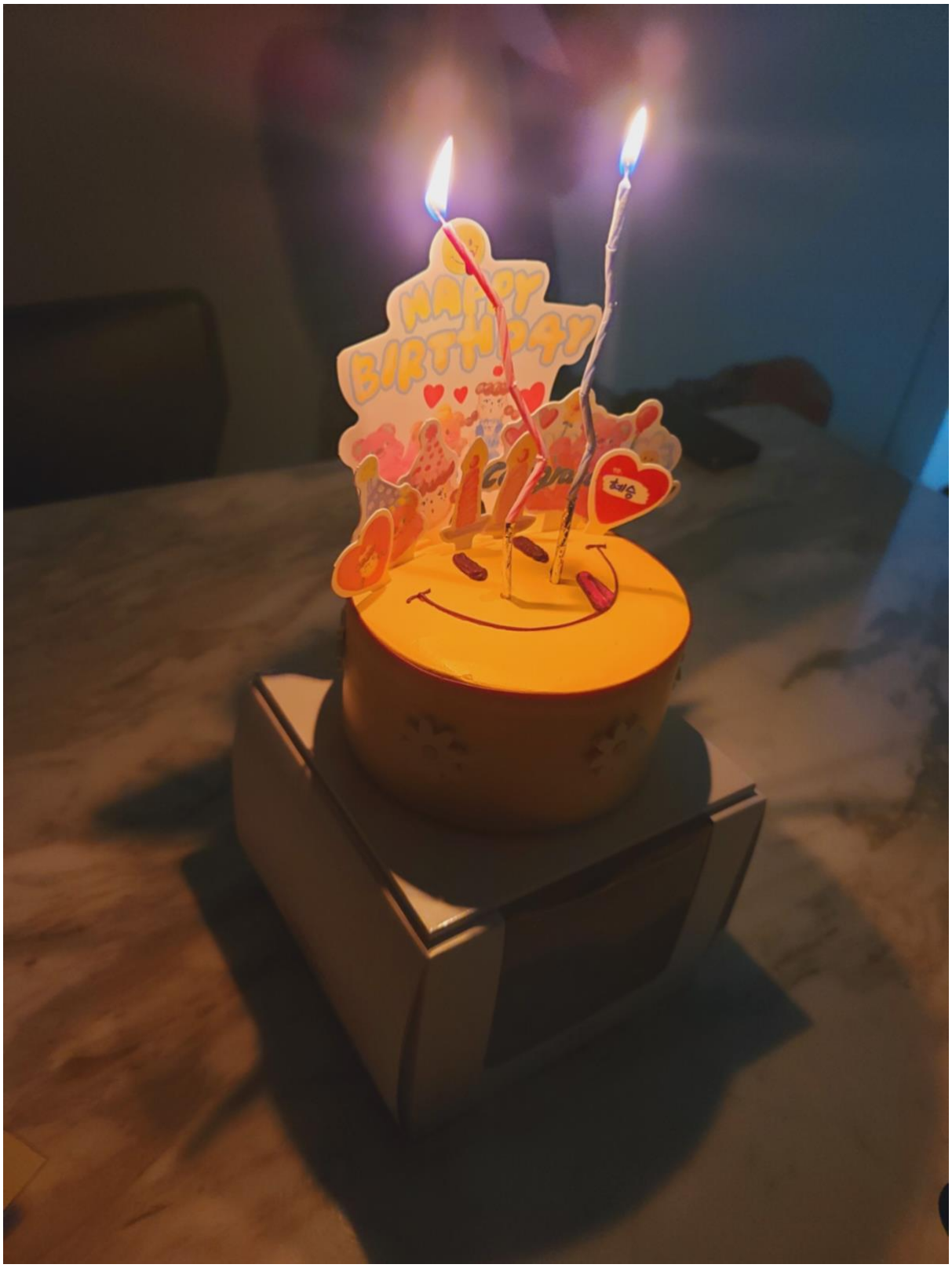

```
import numpy as np

detail = np.int32(original) - np.int32(blur_7)
shapened_2 = np.int32(original) + 2*detail
shapened_5 = np.int32(original) + 5*detail
shapened_10 = np.int32(original) + 10*detail

cv_imshow(shapened_2)
cv_imshow(shapened_5)
cv_imshow(shapened_10)
```

2. Gaussian Filter

원본 영상 (1 에서의 사진과 동일)

결과 영상

2D Gaussian filtering



1D Gaussian filtering

코드 (캡쳐본, 텍스트)

Gaussian filter

```
[9] #1D Gaussian filter 만들기
    kernel1d = cv.getGaussianKernel(7,2)
    print(kernel1d)
```

```
[[0.07015933]
 [0.13107488]
 [0.19071282]
 [0.21610594]
 [0.19071282]
 [0.13107488]
 [0.07015933]]
```

```
#2D Gaussian filter 만들기
kernel2d = np.outer(kernel1d, kernel1d.transpose())
print(kernel2d)
```

```
[[0.00492233 0.00919613 0.01338028 0.01516185 0.01338028 0.00919613
  0.00492233]
 [0.00919613 0.01718062 0.02499766 0.02832606 0.02499766 0.01718062
  0.00919613]
 [0.01338028 0.02499766 0.03637138 0.04121417 0.03637138 0.02499766
  0.01338028]
 [0.01516185 0.02832606 0.04121417 0.04670178 0.04121417 0.02832606
  0.01516185]
 [0.01338028 0.02499766 0.03637138 0.04121417 0.03637138 0.02499766
  0.01338028]
 [0.00919613 0.01718062 0.02499766 0.02832606 0.02499766 0.01718062
  0.00919613]
 [0.00492233 0.00919613 0.01338028 0.01516185 0.01338028 0.00919613
  0.00492233]]
```

```
[11] #2D Gaussian filter 이용하여 필터링
     gaussian_2d = cv.filter2D(original, -1, kernel2d)
     cv_imshow(gaussian_2d)
```

```
[12] #1D Gaussian filter 두번 이용하여 필터링
     gaussian_1d_tmp = cv.filter2D(original, -1, kernel1d)
     gaussian_1d = cv.filter2D(gaussian_1d_tmp, -1, kernel1d.transpose())
     cv_imshow(gaussian_1d)
```

```
#두 필터의 결과가 같은지 비교
difference = cv.subtract(gaussian_1d, gaussian_2d)
print(difference)
```

```
[[[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]

 [[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]

 [[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]

 ...

 [[0 0 0]
```

두 필터의 결과가 같음 픽셀 값 차 없음

```
#1D Gaussian filter 만들기
kernel1d = cv.getGaussianKernel(7,2)
print(kernel1d)
```

```
#2D Gaussian filter 만들기
kernel2d = np.outer(kernel1d, kernel1d.transpose())
print(kernel2d)
```

```
#2D Gaussian filter 이용하여 필터링
```

```
gaussian_2d = cv.filter2D(original, -1, kernel2d)
cv_imshow(gaussian_2d)
```

```
#1D Gaussian filter 두번 이용하여 필터링
gaussian_1d_tmp = cv.filter2D(original, -1, kernel1d)
gaussian_1d = cv.filter2D(gaussian_1d_tmp, -1, kernel1d.transpose())
cv_imshow(gaussian_1d)
```

```
#두 필터의 결과가 같은지 비교
difference = cv.subtract(gaussian_1d, gaussian_2d)
print(difference)
```

3. Median Filter

원본 영상 : 앞에서 쓴 것과 같음

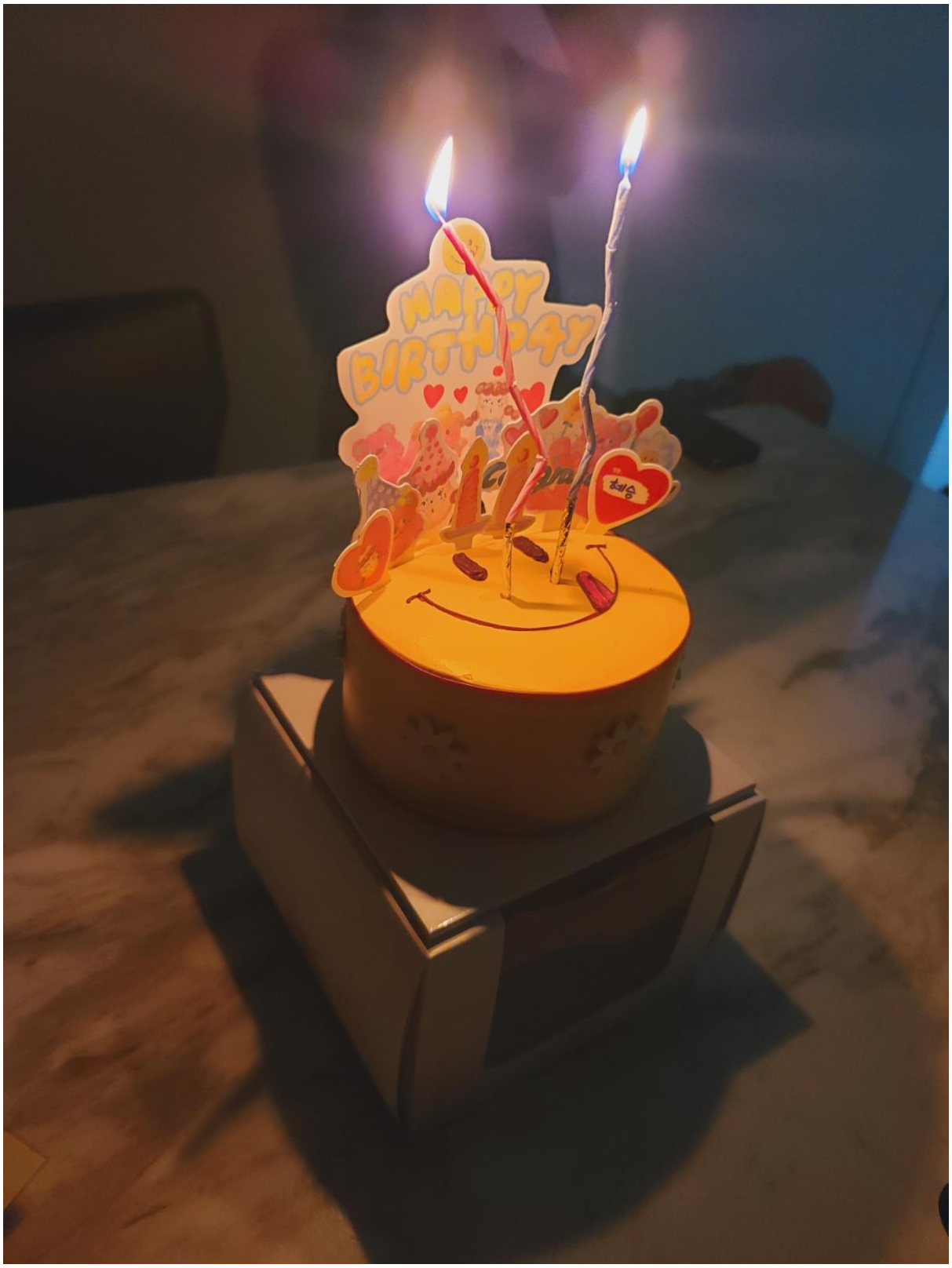
노이즈 추가한 영상

ratio 0.02 일 때

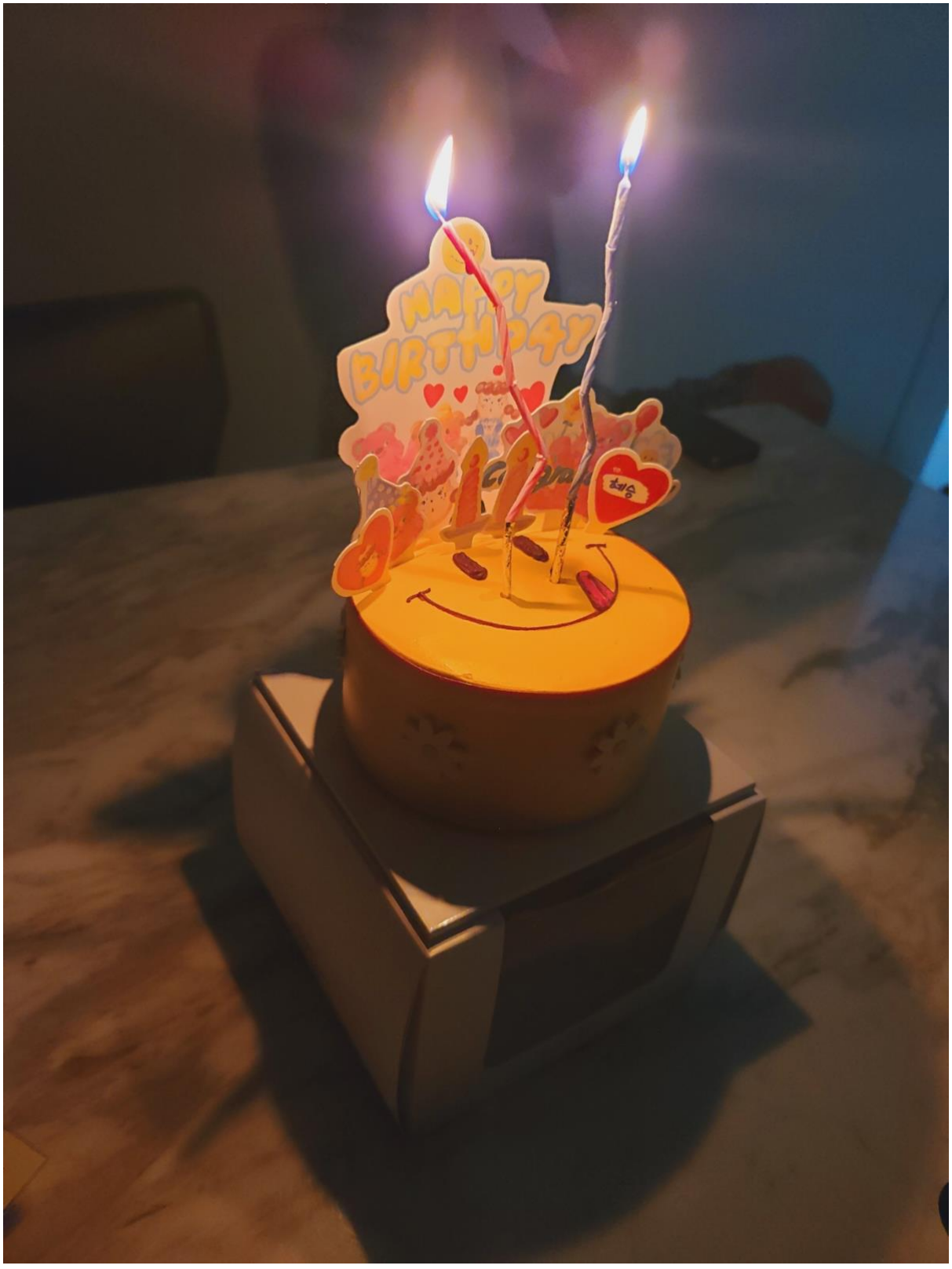
ratio 0.1 일 때

ratio 0.25 일 때

median filter 한 영상
ratio 0.02 일 때
filter size = 3



ratio 0.1 일 때
filter size = 5



ratio 0.25 일 때
filter size = 9

Salt and pepper noise 의 ratio 와 적절한 median filter 의 size 사이 관계

ratio (이하 r)가 0.02 일 경우에는 median filter size(이하 k)가 3 일 때 부터 필터링이 잘 되었다.
k=3 일 때 k=5 일 때 필터링에는 거의 차이가 없었다.
하지만 k=5 일 때 약간 더 뿌옇게 보여서 k=3 을 선택했다.

r=0.1 일때도 위와 같은 맥락으로 선택했다.
k=3 은 다 필터링 하지 못해서 노이즈가 남아있었다.
하지만 k=5 이상은 다 필터를 잘 했지만 가장 선명한 k=5 를 선택했다.

r=0.25 일 때도 마찬가지이다.
k=9 이후 부터 필터링을 잘 했기에 k=9 를 선택했다.

Salt and pepper noise 의 ratio 와 적절한 median filter 의 size 사이의 관계를 생각해보면 ratio 가 클 수록 큰 size 의 필터가 필요하다.
그리고 눈으로 볼 때 노이즈가 안보일 만큼까지만 size 를 키워야 노이즈도 없고 선명한 결과를 얻을 수 있다.

코드(캡쳐, 텍스트)

Median Filter

salt and pepper noise 추가 후 Median filter 적용

```
#salt and pepper noise 추가하는 함수
def addsaltandpeppernoise(image,ratio):
    row,col,ch = image.shape
    out = np.copy(image)
    # Salt mode
    num_salt = np.ceil(image.size * ratio)
    coords = [np.random.randint(0, i - 1, int(num_salt)) for i in image.shape]
    out[coords] = 255
    # Pepper mode
    num_pepper = np.ceil(image.size * ratio)
    coords = [np.random.randint(0, i - 1, int(num_pepper)) for i in image.shape]
    out[coords] = 0
    return out
```

```
addnoise_002 = addsaltandpeppernoise(original, ratio = 0.02)
addnoise_010 = addsaltandpeppernoise(original, ratio = 0.1)
addnoise_025 = addsaltandpeppernoise(original, ratio = 0.25)
cv_imshow(addnoise_002)
cv_imshow(addnoise_010)
cv_imshow(addnoise_025)
```

```
[25] median_002_3 = cv.medianBlur(addnoise_002, ksize=3)
     cv_imshow(median_002_3)
     #median1 = cv.medianBlur(addnoise_002, ksize=5)
     #cv_imshow(median1)

     #Ratio = 0.02일 때 결과 median filter size 비교
     #3>5
     #3, 5 모두 필터링이 잘 되었음 둘다 필터링 잘 되는데 5가 조금 더 블러되어있어서 3으로 선택
```

```
[27] median_010_5 = cv.medianBlur(addnoise_010, ksize=5)
     cv_imshow(median_010_5)
     #Ratio = 0.1일 때 결과 median filter size 비교
     #3<7<5
     #3은 필터 잘 못함 5,7 모두 필터링이 잘 되었음 둘다 필터링 잘 되는데 7가 조금 더 블러되어있어서 5으로 선택
```

```
median_025_9 = cv.medianBlur(addnoise_025, ksize=9)
cv_imshow(median_025_9)
#Ratio = 0.25일 때 결과 median filter size 비교
#3<7<11<9
# 3,7보다 9가 월등하게 성능이 좋았음 9와 11은 비슷했으나 더 선명한 9를 선택
```


```
#salt and pepper noise 추가하는 함수
def addsaltandpeppernoise(image,ratio):
    row,col,ch = image.shape
    out = np.copy(image)
    # Salt mode
    num_salt = np.ceil(image.size * ratio)
    coords = [np.random.randint(0, i - 1, int(num_salt)) for i in image.shape]
    out[coords] = 255
```

```
    # Pepper mode
    num_pepper = np.ceil(image.size * ratio)
    coords = [np.random.randint(0, i - 1, int(num_pepper)) for i in image.shape]
    out[coords] = 0
    return out
```

```
addnoise_002 = addsaltandpeppernoise(original, ratio = 0.02)
addnoise_010 = addsaltandpeppernoise(original, ratio = 0.1)
addnoise_025 = addsaltandpeppernoise(original, ratio = 0.25)
cv_imshow(addnoise_002)
cv_imshow(addnoise_010)
cv_imshow(addnoise_025)
```

```
median_002_3 = cv.medianBlur(addnoise_002, ksize=3)
cv_imshow(median_002_3)
#median1 = cv.medianBlur(addnoise_002, ksize=5)
#cv_imshow(median1)

#Ratio = 0.02 일 때 결과 median filter size 비교
#3>5
#3, 5 모두 필터링이 잘 되었음 둘다 필터링 잘 되는데 5 가 조금 더
블러되어있어서 3 으로 선택
```

```
median_010_5 = cv.medianBlur(addnoise_010, ksize=5)
cv_imshow(median_010_5)
#Ratio = 0.1 일 때 결과 median filter size 비교
#3<7<5
#3 은 필터 잘 못함 5,7 모두 필터링이 잘 되었음 둘다 필터링 잘 되는데 7 가
조금 더 블러되어있어서 5 으로 선택
```

```
median_025_9 = cv.medianBlur(addnoise_025, ksize=9)
cv_imshow(median_025_9)
#Ratio = 0.25 일 때 결과 median filter size 비교
#3<7<11<9
# 3,7 보다 9 가 월등하게 성능이 좋았음 9 와 11 은 비슷했으나 더 선명한 9 를
선택
```