1) ***2 бала*** ***(Розмиття Гауса)*** Розглянемо накладання елементарних цифрових фільтрів на растрове зображення. Для цього введемо кілька понять, які стосуються цифрової обробки зображень.

Похідна зображення. Згідно з математичним означенням похідної, це границя відношення приросту функції до приросту аргументу, коли останній прямує до 0:

$$\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

Якщо ми візьмемо $h = 1$ піксель, а $f(x)$ – це значення пікселя в точці $x(i,j)$, тоді похідна буде рівна $f(x+1)-f(x)$. Таке представлення називають скінченною різницею. Існують три види скінченних різниць:

- Різниця вперед: $f(x+1)-f(x)$;
- Різниця назад: $f(x)-f(x-1)$;
- Центральна різниця: $f(x+1)-f(x-1)$.

Маючи матричне представлення зображення, можемо знайти його похідну за допомогою іншої матриці, яку називають ***маскою***, або ***ядром***. Звідси, маски для похідних у матричному вигляді будуть:

- [1 -1] для різниці вперед: $1\cdot f(x+1)+(-1)\cdot f(x)$;
- [-1 1] для різниці назад: $-1\cdot f(x)+1\cdot f(x-1)$
- [1 0 -1] для центральної різниці: $1\cdot f(x+1)+0\cdot f(x)+(-1)\cdot f(x-1)$.

Загалом для обчислення похідної матриці 2D-зображення по напрямку OX має вигляд

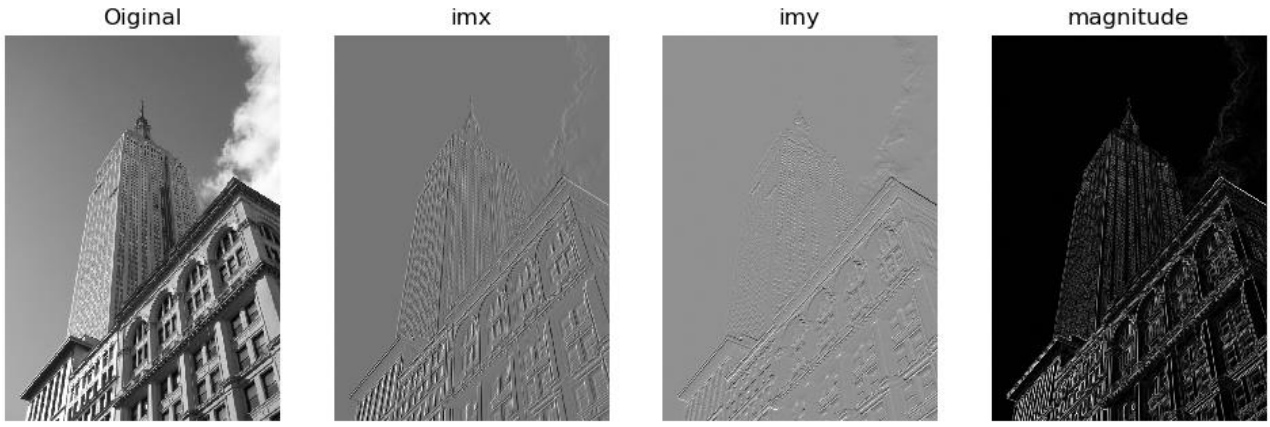
$$\begin{pmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{pmatrix}$$

Аналогічно можна обчислити похідну по напрямку OY. Застосування маски до зображення виглядає так:

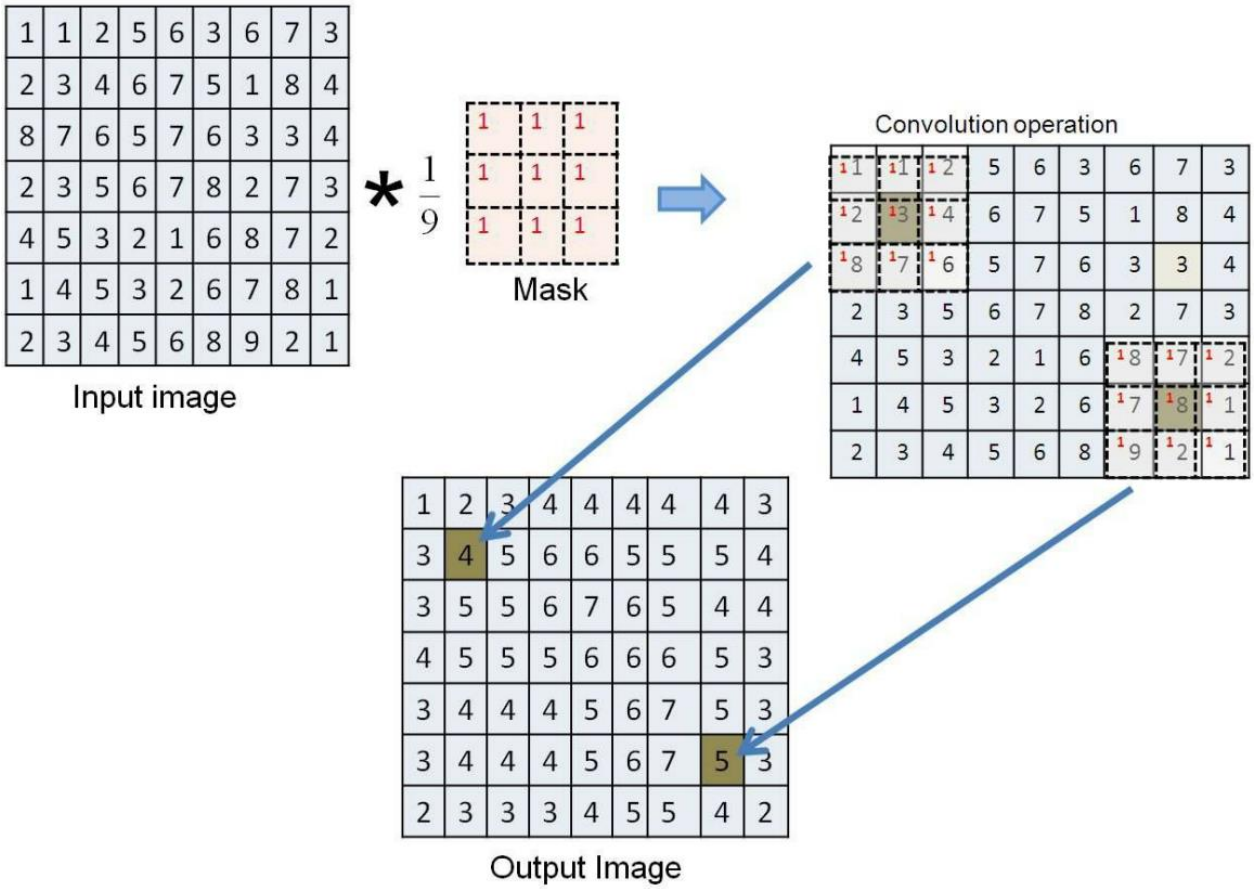
$$\begin{bmatrix} 40 & 50 & 60 & 70 \\ 40 & 50 & 60 & 70 \\ 40 & 50 & 60 & 70 \\ 40 & 50 & 60 & 70 \end{bmatrix} \quad \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 20 & 20 & 0 \\ 0 & 20 & 20 & 20 & 0 \\ 0 & 20 & 20 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Матриця зображення *Похідна маска* *Похідна*

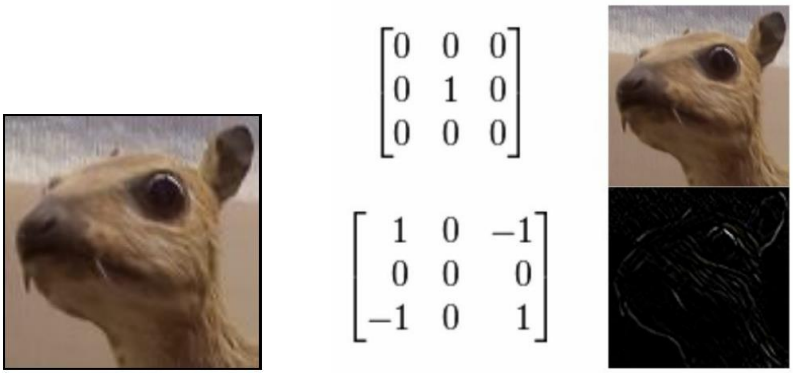
Червоним виділено пікселі, які задіюються в обчисленні значення похідної для відповідного пікселя (2,2) у похідній по зображенню. Демонстрація застосування частинних та звичайної похідної до початкового зображення:



Згортка зображення. У контексті цифрової обробки зображень згортка (convolution) визначається як сума добутків відповідних елементів ядра та матриці зображення.



При згортці реального зображення за допомогою різних масок можемо отримати таке:



Початкове зображення *Ядро* *Згорнуте зображення*

Покращення зображення за допомогою застосування деякої функції до значення пікселів називають ***фільтруванням***. Існують різні типи операцій фільтрування:

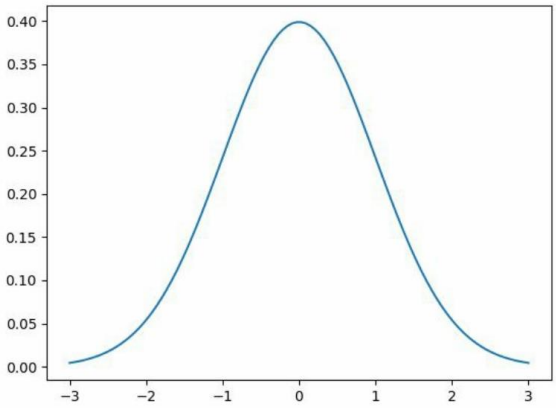
- Розмивання Гауса;
- Медіанний фільтр;

- Дилатація та ерозія;
- Власні фільтри;
- Порогування.

Розмиття Гауса. Один з найбільш часто використовуваних фільтрів для обробки зображень. Він використовує функцію Гаусівського розподілу:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Графічно дана функція представляється так при $\sigma = 1,\ \mu = 0$:



Напишіть власну реалізацію розмивання Гауса, в чому вам може допомогти посилання. Порівняйте отриманий результат із вбудованими можливостями бібліотеки pillow

```
from PIL import Image
from PIL import ImageFilter
img = Image.open("image.png")
blur_img = img.filter(ImageFilter.GaussianBlur(5))
blur_img.show()
```

або skimage

```
from skimage import io
from skimage import filters
img = io.imread("image.png")
out = filters.gaussian(img, sigma=5)
io.imshow(out)
io.show()
```

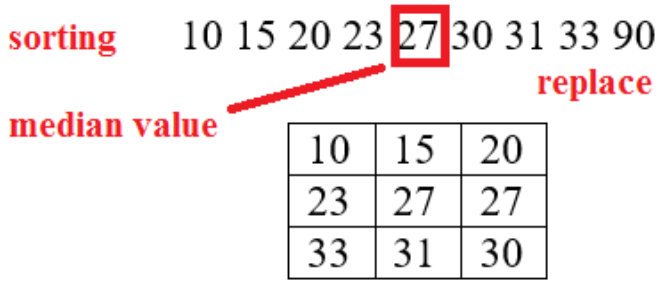
Значення σ обираєте так: номеру списку підгрупи % 5 + 1! *Додайте до звіту відповідні програмні коди та скриншот з порівнянням оригінального*

зображення з результатами розмиття власною та вбудованою реалізаціями.

2) ***2 бала*** Медіанний фільтр повертає медіану (число, яке “поділяє” “навпіл” упорядковану сукупність значень). *Для знайомства з медіанною фільтрацією на графіках візьміть довільну функцію та застосуйте вбудований медіанний фільтр*, як це показано в посиланні.

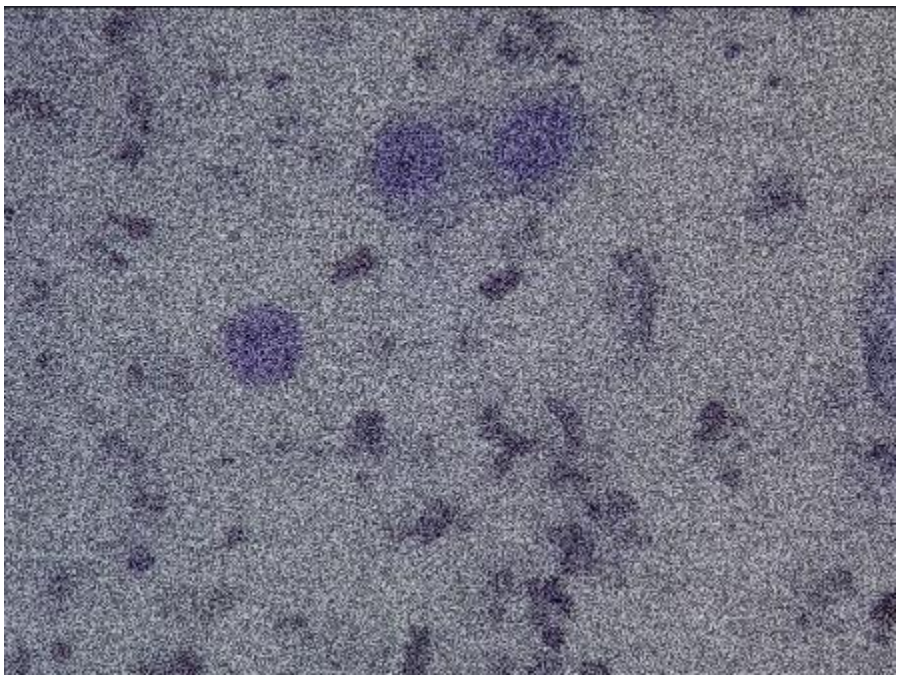
Застосування медіанного фільтру до зображень дозволяє знизити його зашумленість. Береться окіл пікселя (тут – 90 та вісім його сусідів навколо), а в відфільтрованому зображенні його значення заміняється на медіану відсортованої послідовності значень цих 9 пікселів:

10	15	20
23	90	27
33	31	30



10	15	20
23	27	27
33	31	30

Напишіть власну реалізацію медіанного фільтру та застосуйте його до зашумленого зображення, на зразок такого:



Зображення повинно бути унікальним для Вашого варіанту! *Порівняйте отриманий результат від Вашої реалізації та вбудованих можливостей бібліотеки pillow*

```
from PIL import Image
from PIL import ImageFilter
img = Image.open("image.png")
blur_img = img.filter(ImageFilter.MedianFilter(7))
blur_img.show()
```

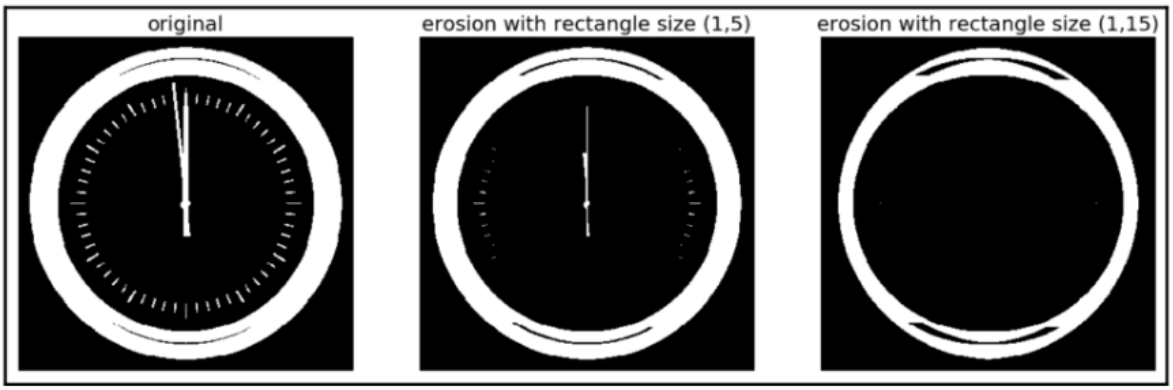
або skimage

```
from skimage import io
from skimage import filters
img = io.imread("image.png")
out = filters.median(img, disk(7))
io.imshow(out)
io.show()
```

3) ***1 бал*** Підберіть чорно-білі зображення для морфологічних операцій та застосуйте до них дилатацію та ерозію, як це показано далі.

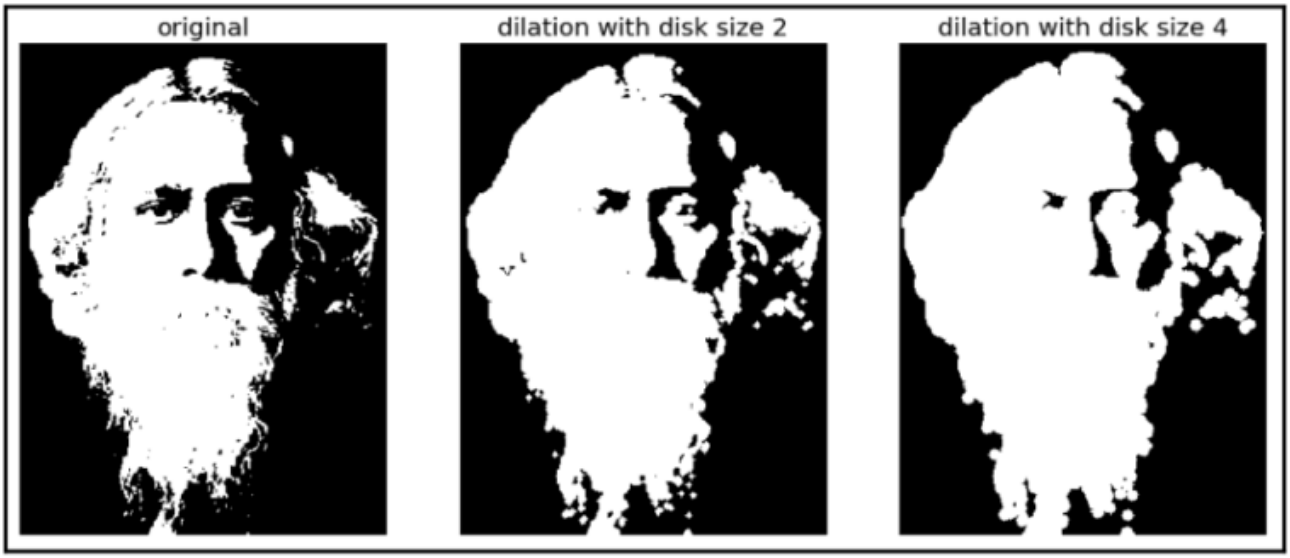
Ерозія:

```
from skimage.io import imread
from skimage.color import rgb2gray
import matplotlib.pylab as pylab
from skimage.morphology import binary_erosion, rectangle

def plot_image(image, title=''):
    pylab.title(title, size=20), pylab.imshow(image)
    pylab.axis('off') # comment this line if you want axis ticks

im = rgb2gray(imread('../images/clock2.jpg'))
im[im <= 0.5] = 0 # create binary image with fixed threshold 0.5
im[im > 0.5] = 1
pylab.gray()
pylab.figure(figsize=(20,10))
pylab.subplot(1,3,1), plot_image(im, 'original')
im1 = binary_erosion(im, rectangle(1,5))
pylab.subplot(1,3,2), plot_image(im1, 'erosion with rectangle size (1,5)')
im1 = binary_erosion(im, rectangle(1,15))
pylab.subplot(1,3,3), plot_image(im1, 'erosion with rectangle size (1,15)')
pylab.show()
```



Дилатація:

```
from skimage.morphology import binary_dilation, disk
from skimage import img_as_float
im = img_as_float(imread('../images/tagore.png'))
im = 1 - im[...,3]
im[im <= 0.5] = 0
im[im > 0.5] = 1
pylab.gray()
pylab.figure(figsize=(18,9))
pylab.subplot(131)
pylab.imshow(im)
pylab.title('original', size=20)
pylab.axis('off')
for d in range(1,3):
    pylab.subplot(1,3,d+1)
    im1 = binary_dilation(im, disk(2*d))
    pylab.imshow(im1)
    pylab.title('dilation with disk size ' + str(2*d), size=20)
    pylab.axis('off')
pylab.show()
```



Додайте до звіту відповідні програмні коди та скріншоти.