

1-

ماتریس با پدینگ reflect

50	50	50	50	50	50
50	50	50	50	50	50
50	50	50	50	50	50
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100

کرنل تبدیل

1	1	1
1	-8	1
1	1	1

نتیجه

0	0	0	0
150	150	150	150
-150	-150	-150	-150
0	0	0	0

2-A

ویژگی ها:

- ۱- خطی بودن
- ۲- دارای مولفه حقیقی و موهومی
- ۳- معکوس پذیری تبدیل
- ۴- عمود بودن بردار های پایه

کاربرد ها:

- ۱- حذف نویز
- ۲- حذف فرکانس های خاص از تصویر
- ۳- تحلیل فرکانسی تصویر

2-B

$$r(x, u) = e^{-j \frac{2\pi}{N} xu}$$

N=4

	real				imaginary			
u=0	1	1	1	1	0	0	0	0
u=1	1	0	-1	0	0	-1	0	1
u=2	1	-1	1	-1	0	0	0	0
u=3	1	0	-1	0	0	1	0	-1

$$T(u) = \sum_{x=0}^{N-1} f(x)r(x, u)$$

$$T(0) = 1 * 1 + 2 * 1 + 3 * 1 + 4 * 1 = 7$$

$$T(1) = 1 * 1 + 2 * (-j) + 3 * (-1) + 4 * j = -2 + 2j$$

$$T(2) = 1 * 1 + 2 * (-1) + 3 * 1 + 4 * (-1) = -2$$

$$T(3) = 1 * 1 + 2 * j + 3 * (-1) + 4 * (-j) = -2 - 2j$$

نتيجة:

7	-2+2j	-2	-2-2j
---	-------	----	-------

## 2-C

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) r(x, y, u, v)$$

$$r(x, y, u, v) = e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

$f(x, y)$ :

0	1	2	1
1	2	3	2
2	3	4	3
1	2	3	2

$T(u, v)$ :

$T(0, 0)$	$T(0, 1)$	$T(0, 2)$	$T(0, 3)$
$T(1, 0)$	$T(1, 1)$	$T(1, 2)$	$T(1, 3)$
$T(2, 0)$	$T(2, 1)$	$T(2, 2)$	$T(2, 3)$
$T(3, 0)$	$T(3, 1)$	$T(3, 2)$	$T(3, 3)$

$$T(0, 0) = \sum_{x=0}^3 \sum_{y=0}^3 f(x, y) = 32$$

$$T(0, 1) = \sum_{x=0}^3 \sum_{y=0}^3 f(x, y) e^{-j2\pi(\frac{y}{4})} = -8$$

$$T(0, 2) = \sum_{x=0}^3 \sum_{y=0}^3 f(x, y) e^{-j2\pi(\frac{2y}{4})} = 0$$

$$T(0, 3) = \sum_{x=0}^3 \sum_{y=0}^3 f(x, y) e^{-j2\pi(\frac{3y}{4})} = -8$$

$$T(1, 0) = \sum_{x=0}^3 \sum_{y=0}^3 f(x, y) e^{-j2\pi(\frac{x}{4})} = -8$$

...

32	-8	0	-8
-8	0	0	0
0	0	0	0
-8	0	0	0

### 3-

```
▶ def median_filter(data, filter_size):
    data_final = []
    data_final = numpy.zeros((len(data),len(data[0])))
    #your code
    radius = filter_size // 2
    pdata = numpy.pad(data,((radius,radius),(radius,radius)),'reflect')
    for i in range(radius,pdata.shape[0]-radius):
        for j in range(radius,pdata.shape[1]-radius):
            pixel=int(numpy.median([pdata[i-radius:i+radius+1,j-radius:j+radius+1]]))
            data_final[i-radius,j-radius]=pixel
    return data_final
```

باید روی پیکسل های تصویر حرکت کنیم و به ازای هر کدام روی پیکسل های اطراف آن در محدوده یک مربع با عرض مشخص محاسباتی را انجام دهیم، از آنجا که پیکسل های مرزی از یک طرف و یا در نقاط گوشی ای از دو طرف فاقد پیکسل های همسایه هستند، با استفاده از متده `pad` از کتابخانه `numpy` برای تصویر حاشیه میسازیم.  
با استفاده دو حلقه `for` تو در تو روی پیکسل های تصویر حرکت میکنیم و روی پیکسل های همسایه آن که در مربعی با عرض مشخص شده توسط `filter_size` قرار دارند، مقدار میانه را به وسیله متده `median` از کتابخانه `numpy` محاسبه میکنیم و جایگزین پیکسل مرکزی میکنیم.

```
▶ #do not change this cell
img = Image.open("noisyimg.png").convert(
    "L")
arr = numpy.array(img)
removed_noise = median_filter(arr, 1)
cv2.imshow(removed_noise)
```



```
▶ #do not change this cell
img = Image.open("noisyimg.png").convert(
    "L")
arr = numpy.array(img)
removed_noise = median_filter(arr, 3)
cv2.imshow(removed_noise)
```

☞



```
▶ #do not change this cell
img = Image.open("noisyimg.png").convert(
    "L")
arr = numpy.array(img)
removed_noise = median_filter(arr, 5)
cv2.imshow(removed_noise)
```

☞



```
▶ #do not change this cell
img = Image.open("noisyimg.png").convert(
    "L")
arr = numpy.array(img)
removed_noise = median_filter(arr, 7)
cv2_imshow(removed_noise)
```

□



```
▶ #do not change this cell
img = Image.open("noisyimg.png").convert(
    "L")
arr = numpy.array(img)
removed_noise = median_filter(arr, 9)
cv2_imshow(removed_noise)
```

□



## مقایسه نتایج:

این فیلتر میتواند نویز نمک و فلفل را به طور موثری برطرف کند  
هر چه ابعاد فیلتر بزرگ تر باشد پیکسل های بیشتری در محاسبه میانه دخالت داده می شوند بنابراین نویز بیشتری حذف می شود ولی از یک جایی به بعد بزرگ تر کردن فیلتر بر روی تصویر اثر نامطلوب دارد و باعث مات شدن تصویر می شود.

## References:

[https://en.wikipedia.org/wiki/Discrete\\_Fourier\\_transform#Applications](https://en.wikipedia.org/wiki/Discrete_Fourier_transform#Applications)