## КУРЕЦ Любовь ИСУ

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

### Устранение шумов на бинарном изображении

#### Алгоритм устранение шумов на бинарном изображении

Алгоритм заключается в обработке изображения локальным окном с записью результата обработки в новое изображение:

- 1) Пусть мы находимся в точке изображения с координатами (I,J) текущий отсчет
- 2) Вокруг текущего отсчета рассматривается локальная окрестность размера NxN
- 3) По значениям точек в лоальной окрестности строится вариационный ряд, который обозначим р. Размер данного ряда N\*N.
- 4) В результирующем изображении текущий отсчет принимает значение по следующему правилу:

$$\operatorname{Im}_{2}(i,j) = \begin{cases} p(k), & \text{if } |p(k) - \operatorname{Im}_{1}(i,j)| < |p(N^{2} - k + 1) - \operatorname{Im}_{1}(i,j)| \\ p(N^{2} - k + 1), & \text{if } |p(k) - \operatorname{Im}_{1}(i,j)| \ge |p(N^{2} - k + 1) - \operatorname{Im}_{1}(i,j)| \end{cases}$$

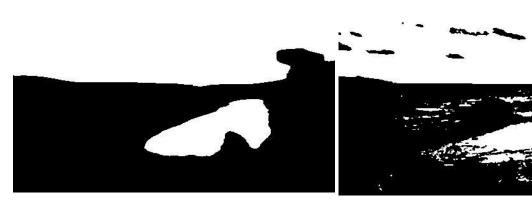
где k — параметр алгоритма, N — нечетное число, lm1 — исходное изображение, lm2 — результирующее изображение.

Если  $k = (N^2+1)/2$  — то есть центр вариационного ряда — данный фильтр становится известным медианным фильтром. В дальнейшем этот параметр будем называть **ОТСТУПОМ.** 

### ВХОДНОЕ ИЗОБРАЖЕНИЕ



## РЕЗУЛЬТАТ:



ОТСТУП 20

ОТСТУП 1



ОТСТУП 10



ОТСТУП 5

# Листинг программы

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
from jupyterthemes import jtplot
jtplot.style()
In [2]:
def show_images(im_dict, cols, shape = (1,1), length = None, size = 8):
 if not length:
    length = len(im_dict)
 else:
    length = max(length, len(im_dict))
 w_size = shape[1] / sum(shape) * size
 h_size = shape[0] / sum(shape) * size
 rows = np.ceil(length / cols)
 fig = plt.figure(figsize=(w_size*cols, h_size*rows))
 i = 0
 for key, value in im_dict.items():
    i += 1
    sub = fig.add_subplot(rows, cols, i)
    sub.set_title(key)
    plt.axis('off')
    plt.imshow(value)
 return fig
In [3]:
fn = 'image.jpg'
img = cv.imread(fn, 0)
img = cv.resize(img, None, fx = 2, fy = 2)
ret,image = cv.threshold(img,127,255,cv.THRESH_BINARY)
row,col = image.shape
```

```
s_vs_p = 0.5
amount = 0.004
out = np.copy(image)
num_salt = np.ceil(amount * image.size * s_vs_p)
coords = [np.random.randint(0, i - 1, int(num_salt)) for i in image.shape]
out[coords] = 1
num_pepper = np.ceil(amount* image.size * (1. - s_vs_p))
coords = [np.random.randint(0, i - 1, int(num_pepper)) for i in
image.shape]
out[coords] = 0
image = cv.cvtColor(image, cv.COLOR_GRAY2RGB)
salt = cv.cvtColor(out, cv.COLOR_GRAY2RGB)
blur = cv.blur(salt,(5,5))
median = cv.medianBim_dict = {'Source':image, 'Salt & Pepper':salt,
'Mean Blur':blur, 'Median Blur':median}
cols = 2
size = 16
fig = show_images(im_dict, cols, shape = img.shape, size =
size)lur(salt,3)
```