МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

«Пороговая обработка изображений»

Отчет по лабораторной работе № 11 (5)

по дисциплине «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1

Кучеренко С. Ю. « » 2023г.

Подпись студента_____

Работа защищена « »_____2023г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2023

Цель работы: изучение алгоритмов порогового преобразования. Рассмотрение методов адаптивного определения порога, нахождение порогового значения Оцу.

Изучение функций cv.threshold , cv.adaptiveThreshold.

Выполнение работы:

Примеры лабораторной работы

Задание 5.1.

Для трех значений порога 70 + No, 140 + No, 210 + No, где No – номер по списку группы (12), провести пороговую обработку полутонового изображения с плавным изменением интенсивности.

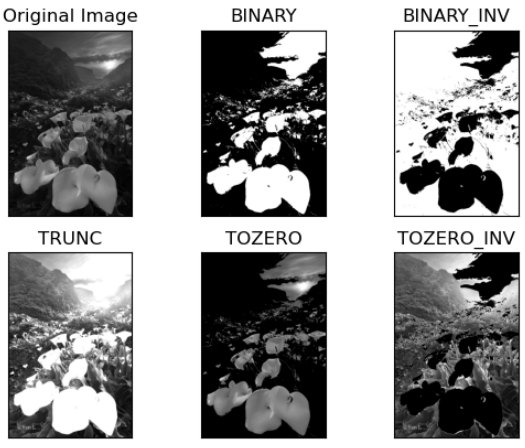
In [1]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
```

In [6]:
```
img = cv2.imread('img/sun.jpg', 0)
```

In [7]:
```
ret, thresh1 = cv2.threshold (img, 82,255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold (img, 82,255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold (img, 82,255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold (img, 82,255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold (img, 82,255, cv2.THRESH_TOZERO_INV)
```

In [8]:
```
title = ['Original Image', 'BINARY', 'BINARY_INV', 'TRUNC', 'TOZERO', 'TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]
```

In [10]:
```
for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(title[i])
    plt.xticks([]),plt.yticks([])
plt.show ()
```



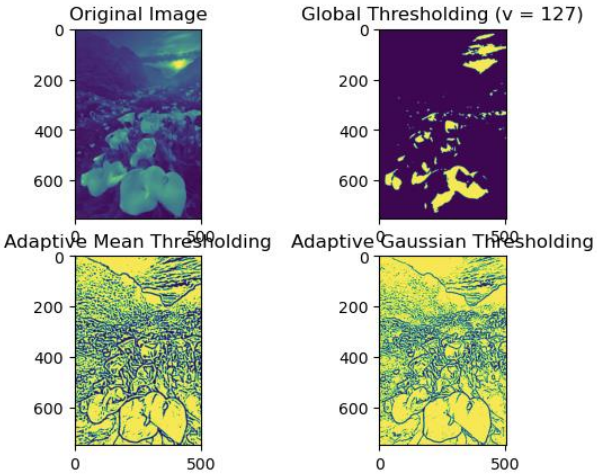
Задание 5.2.

Протестировать функции с адаптивным порогом, задавая последовательно два значения порога, примерно 1/3 и 2/3 от максимума интенсивности.

In [10]:
```
img = cv2.imread('img/sun.jpg', 0)
img = cv2.medianBlur(img,5)
```

In [11]:
```
ret1,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
th2 = cv2.adaptiveThreshold(img,255,
                            cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,11,2)
th3 = cv2.adaptiveThreshold(img,255,
                            cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)
```

In [19]:
```
plt.subplot(221), plt.imshow(img)
plt.title('Original Image')

plt.subplot(222), plt.imshow(th1)
plt.title('Global Thresholding (v = 127)')

plt.subplot(223), plt.imshow(th2)
plt.title('Adaptive Mean Thresholding')

plt.subplot(224), plt.imshow(th3)
plt.title('Adaptive Gaussian Thresholding')
plt.show()
```



Задание 5.3.

Загрузить модули cv2, random, PIL. Создать зашумленное изображение.

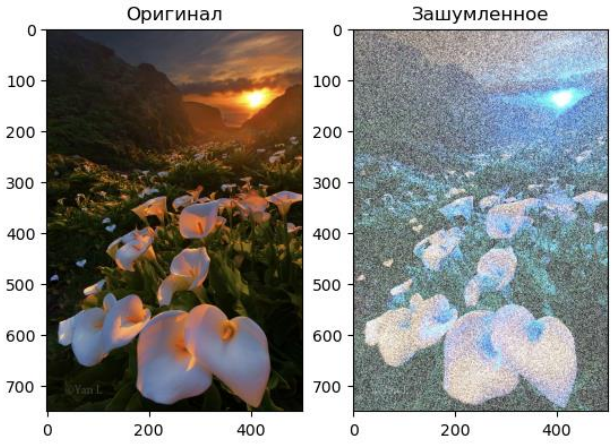
In [20]:
```
import random
from PIL import Image, ImageDraw
```

In [21]:
```
image = Image.open('img/sun.jpg')
```

In [23]:
```
draw = ImageDraw.Draw(image)

width = image.size[0]
height = image.size[1]

pix = image.load()
```

In [24]:
```
for i in range(width):
    for j in range(height):
        rand = random.randint(0, 200)
        a = pix[i, j][0] + rand
        b = pix[i, j][1] + rand
        c = pix[i, j][2] + rand
        if (a > 255):
            a = 255
        if (b > 255):
            b = 255
        if (c > 255):
            c = 255
        draw.point((i, j), (a, b, c))
```

In [29]:
```
image.save("median.png", "JPEG")

imag = cv2.imread('img/sun.jpg')
imag = cv2.cvtColor(imag, cv2.COLOR_BGR2RGB)
img = cv2.imread('median.png')

plt.subplot(121),plt.imshow(imag),plt.title('Оригинал')
plt.subplot(122),plt.imshow(img),plt.title('Зашумленное')
plt.show()
```



Задание 5.4.

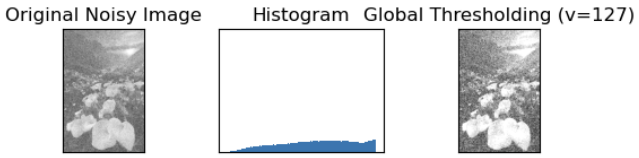
На вход программы пороговой обработки подается зашумленное изображение. Это изображение обрабатывается тремя способами. В первом случае используется глобальный порог со значением 127. Во втором случае напрямую применяется порог Оцу. В третьем случае изображение сначала удаляет шум фильтром с гауссовым ядром 5x5, затем применяется пороговая обработка Оцу. Сделать анализ того, как фильтрация шума улучшает результат.

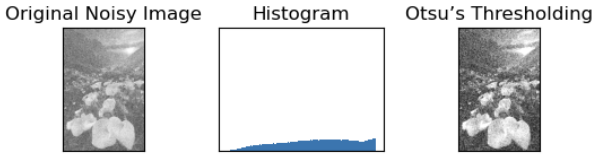
In [32]:
```
img = cv2.imread('img/median.png', 0)
```

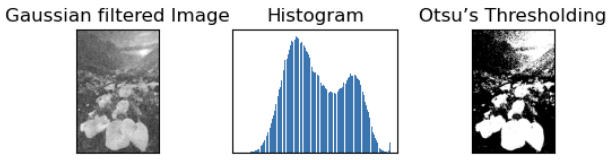
In [33]:
```
ret1,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
blur = cv2.GaussianBlur(img,(5,5),0)
ret3,th3 = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

In [34]:
```
images = [img, 0, th1, img, 0, th2, blur, 0, th3]
titles = ['Original Noisy Image', 'Histogram','Global Thresholding (v=127)',
          'Original Noisy Image', 'Histogram','Otsu's Thresholding',
          'Gaussian filtered Image','Histogram','Otsu's Thresholding']
```

In [35]:
```
for i in range(3):
    plt.subplot(3,3,i*3+1), plt.imshow(images[i*3], "gray")
    plt.title(titles[i*3]), plt.xticks([]),
    plt.yticks([])
    plt.subplot(3,3,i*3+2),plt.hist(images[i*3].ravel(),256)
    plt.title(titles[i*3+1]), plt.xticks([]),plt.yticks([])
    plt.subplot(3,3,i*3+3),plt.imshow(images[i*3+2],"gray")
    plt.title(titles[i*3+2]), plt.xticks([]),plt.yticks([])
    plt.show()
```







Самостоятельное задание

Задано оригинальное изображение. Вывести восстановленное изображение без шума.

In [51]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import random
```

Создадим функцию, которая будет генерировать зашумленное изображение. Для этого можно использовать библиотеку random, чтобы добавлять шум к каждому пикселю изображения.

In [52]:
```
def generate_noisy_image(image):
    noisy_image = np.zeros(image.shape, np.uint8)
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            noise = random.randint(-50, 50)
            noisy_image[i][j] = image[i][j] + noise
    return noisy_image
```

Загружаем изображение и делаем его зашумленным

In [53]:
```
image = cv2.imread('img/mort.jpg')
noisy_image = generate_noisy_image(image)
```

Создаем функцию, которая будет обрабатывать изображение и возвращать восстановленное изображение без шума, где применяем фильтр медианной фильтрации для удаления шума и фильтр Гаусса для сглаживания изображения

In [54]:
```
def restore_image(image):
    restored_image = np.copy(image)
    restored_image = cv2.medianBlur(restored_image, 3)
    restored_image = cv2.GaussianBlur(restored_image, (5, 5), 0)
    return restored_image
```

Восстанавливаем изображение при помощи функции restored_image

In [56]:
```
restored_image = restore_image(noisy_image)
```

In [57]:
```
plt.subplot(131),plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)),
plt.title('Original Image')

plt.xticks([]), plt.yticks([])
plt.subplot(132),plt.imshow(cv2.cvtColor(noisy_image, cv2.COLOR_BGR2RGB)),
plt.title('Noisy Image')

plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(cv2.cvtColor(restored_image, cv2.COLOR_BGR2RGB)),
plt.title('Restored Image')
plt.xticks([]), plt.yticks([])
plt.show()
```