**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**
**Федеральное государственное автономное образовательное учреждение высшего образования**
**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

# Отчет по лабораторной работе № 3.11

# «Пороговая обработка изображений»

**по дисциплине «Технологии распознавания образов»**

Выполнил студент группы
ПИЖ-б-о-21-1
Зиберов Александр
« » мая 2023 г.
Подпись студента_____
Работа защищена
« »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2023

**Цель работы:**

Изучение алгоритмов порогового преобразования. Рассмотрение методов адаптивного определения порога, нахождение порогового значения Оцу.

**Выполнение работы:**

Проработать примеры лабораторной работы в отдельном ноутбуке.

**Задание 5.1.**

Для трех значений порога 70 + №, где № – номер по списку группы (7), провести пороговую обработку полутонового изображения с плавным изменением интенсивности.

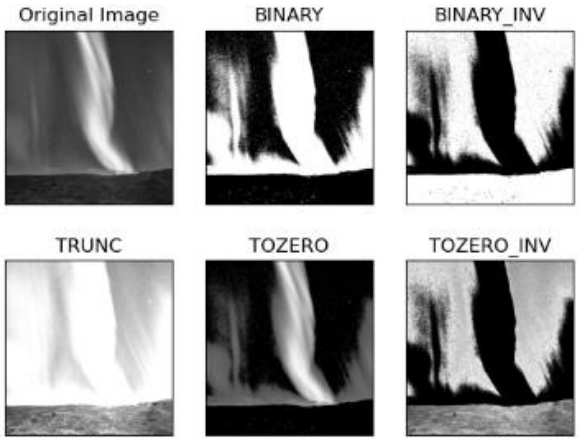
In [1]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

def imgshow(image, conversion=cv2.COLOR_BGR2RGB):
    image = cv2.cvtColor(image, conversion)
    # Show the image
    plt.imshow(image)
    # remove the axis / ticks for a clean looking image
    plt.xticks([])
    plt.yticks([])
    plt.show()
```

In [2]:
```
img = cv2.imread ('pictures/light_sky.jpg')
imgshow(img)
img = cv2.cvtColor(img,  cv2.COLOR_BGR2GRAY)
```

In [3]:
```
ret, thresh1 = cv2.threshold(img, 70+7,255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 70+7,255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img, 70+7,255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img, 70+7,255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img, 70+7,255, cv2.THRESH_TOZERO_INV)

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

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()
```

Рисунок 1 – Пример 1

In [3]:
```
ret, thresh1 = cv2.threshold(img, 70+7,255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 70+7,255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img, 70+7,255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img, 70+7,255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img, 70+7,255, cv2.THRESH_TOZERO_INV)

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

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()
```




In [4]:
```
ret, thresh1 = cv2.threshold(img, 140+7,255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 140+7,255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img, 140+7,255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img, 140+7,255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img, 140+7,255, cv2.THRESH_TOZERO_INV)

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

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()
```

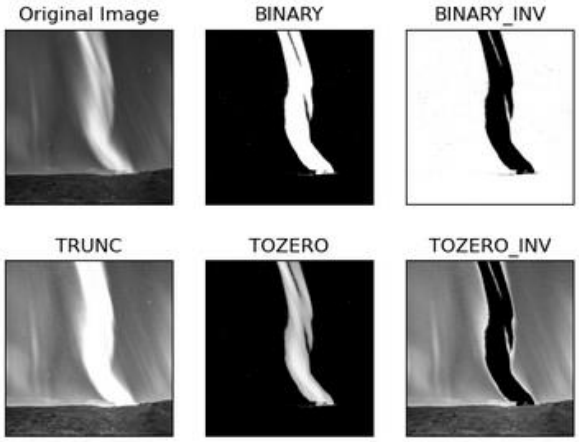



Рисунок 2 – Пример 2

In [5]:
```
ret, thresh1 = cv2.threshold(img, 210+7,255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 210+7,255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img, 210+7,255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img, 210+7,255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img, 210+7,255, cv2.THRESH_TOZERO_INV)

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

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()
```

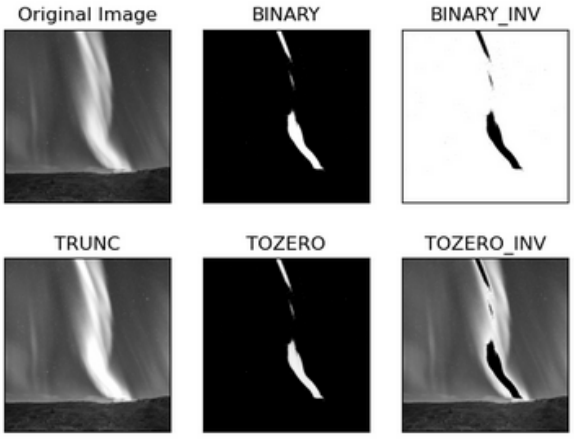



Рисунок 3 – Пример 3

**Задание 5.2.**

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

In [6]:
```
img = cv2.imread('pictures/light_sky.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img = cv2.medianBlur(img,5)
```

In [7]:
```
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 [8]:
```
titles = ['Original Image', 'Global Thresholding (v = 127)',
          'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']

images = [img, th1, th2, th3]

for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

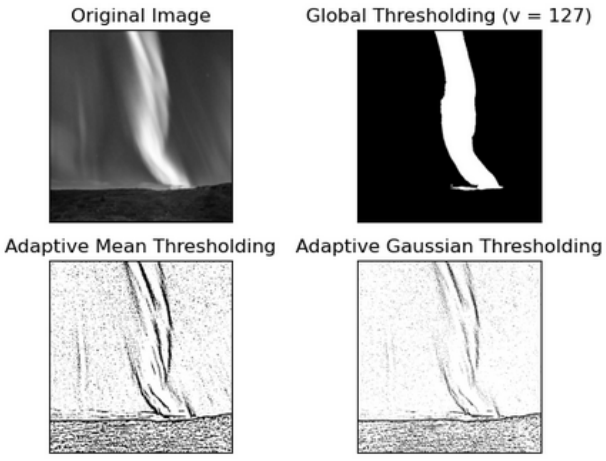



Рисунок 4 – Пример 4

**Задание 5.3.**

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

In [9]:
```
import random
from PIL import Image, ImageDraw
image = Image.open('pictures/light_sky.jpg')
draw = ImageDraw.Draw(image)
```

In [10]:
```
import random
from PIL import Image, ImageDraw
image = Image.open('pictures/light_sky.jpg')
draw = ImageDraw.Draw(image)
width = image.size[0]
height = image.size[1]
pix = image.load()
```

In [11]:
```
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 [12]:
```
image.save("median.png", "JPEG")

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

plt.subplot(121),plt.imshow(imag),plt.title('original')
plt.axis("off")
plt.subplot(122),plt.imshow(img),plt.title('result')
plt.axis("off")
plt.show()
```

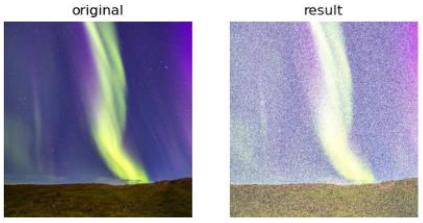



Рисунок 5 – Пример 5

**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 [13]:
```
img = cv2.imread('pictures/light_sky.jpg', 0)
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 [14]:
```
images = [img, 0, th1, img, 0, th2, blur, 0, th3]
titles = ["Original Noisy Image","Histogram","GlobalThresholding (v=127)",
          "Original Noisy Image", "Histogram","Otsu's Thresholding",
          "Gaussian filtered Image","Histogram","Otsu's Thresh-olding"]
```

In [15]:
```
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()
```

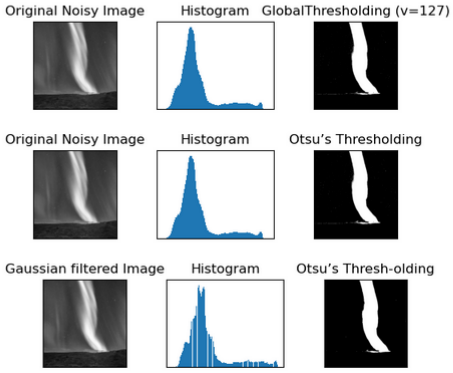



Рисунок 6 – Пример 6

## Индивидуальное задание

**Задание**

Дана фотография чека. Подготовить изображение для распознавания, используя изученные методы пороговой обработки.

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

def imgshow(image, conversion=cv2.COLOR_BGR2RGB):
    image = cv2.cvtColor(image, conversion)
    plt.imshow(image)
    plt.axis("off")
    plt.xticks([])
    plt.yticks([])
    plt.show()
```

Импортируем изображение, преобразуем его в градации серого функцией cv2.COLOR_BGR2GRAY:

In [2]:
```
img = cv2.imread('check2.jpg')
img_original = cv2.imread('check2.jpg')

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_original = cv2.cvtColor(img_original, cv2.COLOR_BGR2RGB)
imgshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB));
```


Рисунок 10 – Индивидуальное задание (1)

В данном случае применим адаптивный порог, так как освещение на фото неравномерное.
Воспользуемся функцией cv2.adaptiveThreshold() с флагом cv2.ADAPTIVE_THRESH_MEAN_C, который в качестве порогового значения берет среднее арифметическое всех пикселей в окрестности выделенного пикселя.

In [3]:
```
adapt_th = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 21, 15)
adapt_th = cv2.cvtColor(adapt_th, cv2.COLOR_BGR2RGB)
imgshow(adapt_th)
```


Рисунок 11 – Индивидуальное задание (2)

Также, можно воспользоваться бинаризацией Оцу, передав в функцию cv2.threshold() флаги cv2.THRESH_BINARY + cv2.THRESH_OTSU:

In [4]:
```
ret,th_otsu = cv2.threshold(img,177,255,cv2.THRESH_BINARY + cv2.THRESH_OTSU)
th_otsu = cv2.cvtColor(th_otsu, cv2.COLOR_BGR2RGB)
imgshow(th_otsu);
```

Рисунок 12 – Индивидуальное задание (3)

Сравним полученные результаты:

In [5]:
```
images = [adapt_th, th_otsu]
titles = ["Адаптивный порог", "Бинаризация Оцу"]
plt.figure(figsize=(15,15))
for i in range(2):
    plt.subplot(2,3,i+1),plt.imshow(images[i])
    plt.xticks([]),plt.yticks([])
    plt.title(titles[i])
plt.show();
```

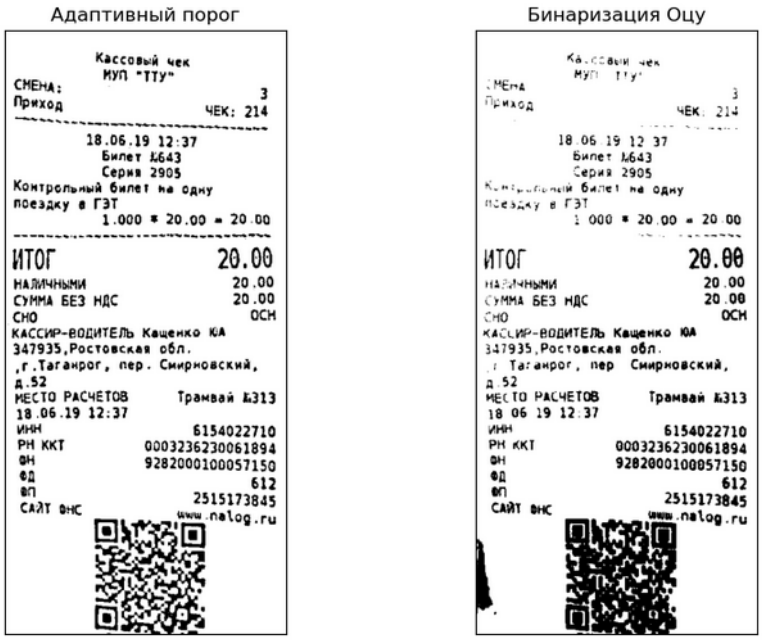



Рисунок 13 – Индивидуальное задание (4)

**Вывод:** В результате выполнения работы были изучены алгоритмы порогового преобразования, рассмотрены методы адаптивного определения порога, нахождение порогового значения Оцу.

**1. Как происходит процесс получения бинарного изображения?**

Бинарное изображение получается путем сравнения интенсивности каждого пикселя с пороговым значением, удаления старого значения интенсивности и присвоения нового значения в зависимости от того, больше или меньше интенсивность пикселя порогового значения.

**2. Что такое пороговая обработка изображений?**

Пороговая обработка – разбиение изображения на две области, одна из которых содержит все пиксели со значением ниже порога, а другая содержит все пиксели со значением выше этого порога. Этот метод занимает важное место в задачах сегментации изображений.

**3. Какими способами можно использовать порог для обработки изображений?**

Порог можно использовать для бинаризации изображения, выделения объектов, удаления шума и т.д., например, функция cv2.threshold может использоваться вместе с пороговыми флагами для различных методов бинаризации. Также пороговые методы могут быть использованы вместе с другими методами обработки изображений, такими как морфологические операции.

**4. Опишите процесс пороговой обработки изображения**

Первым аргумент в функции cv.threshold(img,127, 255, cv.THRESH) – это исходное изображение, которое должно быть в градациях серого. Второй аргумент – это величина порога. Третий аргумент – это значение интенсивности на выходе функции, когда значение пикселя больше порогового значения. В режиме инвертирования меньше порогового значения. Четвертым параметром задаются различные типы порогового значения.

**cv.THRESH_BINARY** – для формирования на выходе бинарного изображения:

$$b(x,y)=\begin{cases}255, \text{если } f(x,y)>p;\\ 0, \text{если } f(x,y)<p.\end{cases}$$

**cv.THRESH_BINARY_INV** – для формирования на выходе инвертированного бинарного изображения:

$$b(x,y)=\begin{cases}0, \text{если } f(x,y)>p;\\ 255, \text{если } f(x,y)<p.\end{cases}$$

**cv.THRESH_TRUNC**, на выходе:

$$b(x,y)=\begin{cases}threshold, \text{если } f(x,y)>p;\\ src(x,y), \text{если } f(x,y)<p.\end{cases}$$

**cv.THRESH_TOZERO**, на выходе:

$$b(x,y)=\begin{cases}src(x,y), \text{если } f(x,y)>p;\\ 0, \text{если } f(x,y)<p.\end{cases}$$

**cv.THRESH_TOZERO_INV**, на выходе:

$$b(x,y)=\begin{cases}0, \text{если } f(x,y)>p;\\ src(x,y), \text{если } f(x,y)<p.\end{cases}$$

**5. Что такое адаптивный порог?**

Адаптивный порог (Adaptive Thresholding) — это метод пороговой обработки изображений, который позволяет автоматически определить пороговое значение для каждого пикселя на основе его окружающего контекста. В отличие от глобального порога, который применяется одинаково ко всем пикселям изображения, адаптивный порог учитывает локальные свойства изображения.

Применение адаптивного порога может быть полезно при задачах бинаризации изображений, выделении объектов на сложных фоновых условиях, распознавании символов или штрих-кодов и в других областях обработки изображений, где необходимо автоматически настроить пороговое значение для каждой области изображения.

**6. В чем отличие функции cv2.adaptiveThreshold с параметром cv2.ADAPTIVE_THRESH_MEAN_ и cv2.ADAPTIVE_THRESH_GAUSSIAN _C?**

Функция cv2.adaptiveThreshold с параметром cv2.ADAPTIVE_THRESH_MEAN_C берет в качестве порогового значения среднее арифметическое всех пикселей в окрестности выделенного пикселя, а

функция cv2.adaptiveThreshold с параметром cv2.ADAPTIVE_THRESH_GAUSSIAN_C берет взвешенную сумму значений окрестностей, где весовые коэффициенты определяются с помощью функции Гаусса.

**7. Бинаризация Оцу**

Если объект отличается по яркости от фона, то можно ввести порог, чтобы разделить изображение на светлый объект и темный фон.

Объект – это множество пикселей, яркость которых превышает порог $I>p$, а фон – множество остальных пикселей, яркость которых ниже порога $I<p$.

Метод Оцу для расчета порога использует гистограмму изображения. Гистограмма показывает, как часто встречается на данном изображении то или иное значение пикселя. Зная яркость каждого пикселя, подсчитаем сколько пикселей имеют такую яркость.