МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет»

Кафедра инфокоммуникаций

Отчёт по практическому занятию №3.11

«Пороговая обработка изображений»

по дисциплине «Теории распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1
Коновалова Виктория
Николаевна « »_____20__г.
Подпись студента _____
Работа защищена « »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Цель: изучение алгоритмов порогового преобразования. Рассмотрение методов адаптивного определения порога, нахождение порогового значения Оцу. Изучение функций cv.threshold, cv.adaptiveThreshold.

Проработаны примеры из методических указаний

Примеры

Задание 5.1.

Для трех значений порога 70 + №, 140 + №, 210 + №, где № – номер по списку группы (21), провести пороговую обработку полутонового изображения с плавным изменением интенсивности.

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

In [2]:
```
img = cv2.imread ('b.jpg')
img = cv2.cvtColor(img,  cv2.COLOR_BGR2GRAY)
```

In [3]:
```
ret, thresh1 = cv2.threshold (img, 76,255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold (img, 76,255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold (img, 76,255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold (img, 76,255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold (img, 76,255, cv2.THRESH_TOZERO_INV)
```

In [4]:
```
title = ['Original Image', 'BINARY', 'BINARY_INV', 'TRUNC', 'TOZERO', 'TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]
```

In [5]:
```
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.

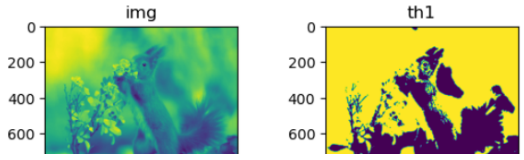
Протестировать функции с адаптивным порогом, задавая последовательно два значения порога, примерно 1/3 и 2/3 от максимума интенсивности. Проанализировать результат пороговой обработки изображения

In [6]:
```
img = cv2.imread('b.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)
titles = ['Original Image', 'Global Thresholding (v = 127)', 'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
```

In [8]:
```
images = [img, th1, th2, th3]

plt.subplot(221), plt.imshow(img)
plt.title('img')
plt.subplot(222), plt.imshow(th1)
plt.title('th1')
plt.subplot(223), plt.imshow(th2)
plt.title('th2')
plt.subplot(224), plt.imshow(th3)
plt.title('th3')
plt.show()
```

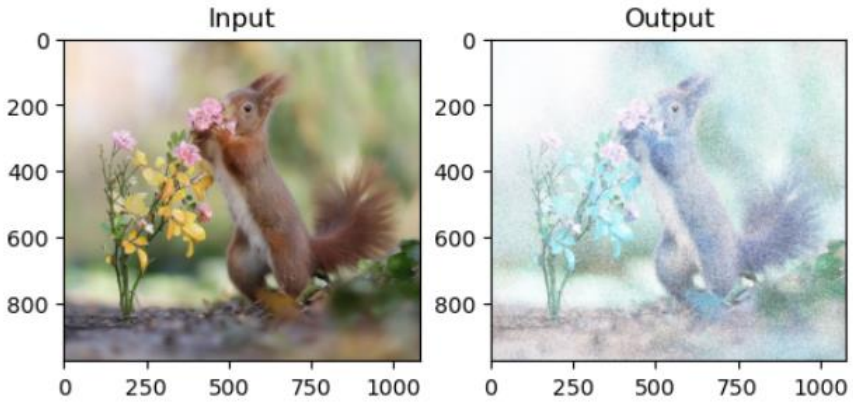


```
        a = pix[i, j][0] + rand
        b = pix[i, j][1] + rand
        c = pix[i, j][2] + rand
        if (a > 255):
            a = 255
        if (b > 255):
            b = 255
        if (c > 255):
            c = 255
        draw.point((i, j), (a, b, c))
```

In [14]:
```
image.save("median.png", "JPEG")

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

plt.subplot(121),plt.imshow(imag),plt.title('Input')
plt.subplot(122),plt.imshow(img),plt.title('Output')
plt.show()
```



ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Индивидуальное задание

Дано изображение рукописного текста в оттенках серого, улучшить читабельность текста

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

In [2]:
```
# Загрузка изображения
image = cv2.imread('3.jpg', 0)
```

In [3]:
```
# Применение медианного фильтра
blurred = cv2.medianBlur(image, 5)
```

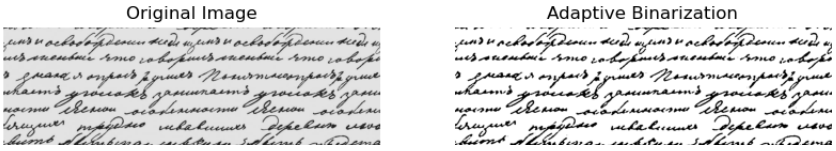
In [4]:
```
# Применение адаптивной бинаризации
binarized = cv2.adaptiveThreshold(blurred, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 11, 2)
```

In [5]:
```
# Отображение изображений
titles = ['Original Image', 'Adaptive Binarization']
images = [image, binarized]

plt.figure(figsize=(10, 5))

for i in range(2):
    plt.subplot(1, 2, i+1)
    plt.imshow(images[i], cmap='gray')
    plt.title(titles[i])
    plt.axis('off')

plt.show()
```



Вопросы для самопроверки

1. 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:

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

2. Адаптивный порог – это?

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.

Процесс адаптивного порога включает следующие шаги:

Разделение изображения на неперекрывающиеся блоки или окна.

Расчет порогового значения для каждого блока на основе свойств пикселей внутри блока.

Применение порогового значения к соответствующему блоку изображения.

Основное преимущество адаптивного порога состоит в том, что он позволяет эффективно обрабатывать изображения с переменной освещенностью или различными фонами. Он способен адаптироваться к изменяющимся условиям освещения и обеспечивать более точную

сегментацию объектов на изображении.

Адаптивный порог имеет несколько вариантов реализации, таких как адаптивный порог с фиксированным размером окна, адаптивный порог с переменным размером окна, методы адаптивного порога на основе среднего значения и гауссовского взвешивания и т. д. Выбор конкретного метода зависит от характеристик изображения и требуемого результата.

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.

3. Функции с адаптивным порогом

Рассмотрим две функции с адаптивным порогом.

cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,11,2) – в качестве порогового значения берется среднее арифметическое всех пикселей в окрестности выделенного пикселя.

cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2) – в качестве порогового значения берется взвешенная сумма значений окрестностей, причем весовые коэффициенты находятся с помощью функции Гаусса.

Первый аргумент в этих функциях – исходное изображение, второй – значение интенсивности на выходе функции, третий указывает, какой метод используется: берется среднее арифметическое всех пикселей в окрестности выделенного пикселя или среднее по Гауссу, пятый определяет размер окрестности 11×11 выделенного пикселя, нужной для вычисления порогового значение. Последний аргумент – постоянная C вычитается из вычисленного среднего или взвешенного среднего, ее применение позволяет точно настроить пороговое значение.

4. Что такое Бинаризация?

Процесс бинаризации – это перевод цветного (или в градациях серого) изображения в двухцветное черно-белое. Главным параметром такого преобразования является порог t – значение, с которым сравнивается яркость каждого пикселя.

5. Как осуществляется выбор способа обработки

Для выбора способа обработки используется функция cv2.threshold(img,127,255,cv2.THRESH_BINARY), но передается дополнительный флаг cv.THRESH_OTSU. Для порогового значения просто введите ноль. Затем алгоритм находит оптимальное пороговое значение и возвращает вас в качестве второго выхода retVal.

Пример

cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_O TSU)# обработка Otsu

cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_ OTSU) # обработка Otsu’s после фильтра Гаусса

6. 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.

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.

8. Библиотека PIL, объект Image

Библиотека PIL (Python Imaging Library) предоставляет возможности для обработки изображений в Python. Она позволяет открывать, создавать и сохранять изображения, а также выполнять различные операции с ними, такие как изменение размера, обрезка, поворот, фильтрация и многое другое.

Основной объект в PIL - это объект изображения (Image), который представляет собой двумерное растровое изображение. С помощью этого объекта можно выполнять различные операции над изображением, включая доступ к пикселям, изменение цветовых каналов, применение фильтров и фильтрации, рисование и т. д.

Для работы с изображениями в PIL также используется объект ImageDraw, который предоставляет функциональность для рисования на изображениях. С его помощью можно создавать графические элементы, такие как линии, окружности, прямоугольники и т. д., а также заполнять их цветом.

В целом, библиотека PIL является мощным инструментом для работы с изображениями в Python и предоставляет множество функций и методов для обработки и модификации изображений.

9. Функция ImageDraw.Draw()

Функция ImageDraw.Draw() является конструктором объекта ImageDraw, который используется для рисования на изображении. Она создает инструмент, который позволяет нам выполнять операции рисования, такие как рисование линий, прямоугольников, окружностей и т. д., на заданном изображении.

При вызове функции ImageDraw.Draw(image), где image - это объект изображения Image, создается и возвращается инструмент draw, связанный с указанным изображением. С помощью этого инструмента мы можем применять различные методы рисования для добавления графических элементов на изображение.

Например, с помощью инструмента draw мы можем вызвать метод line() для рисования линии, метод rectangle() для рисования прямоугольника и так далее. Каждый из этих методов принимает параметры, такие как координаты точек, размеры, цвет и толщину линии, и выполняет соответствующую операцию рисования на изображении.

Таким образом, функция ImageDraw.Draw() позволяет нам создать инструмент для рисования на изображении и начать использовать его для добавления графических элементов на изображение.

10. Как фильтрация шума улучшает результат.

Фильтрация шума помогает улучшить результаты обработки изображения, потому что шум может вносить случайные изменения в пиксели, что приводит к искажениям и потере деталей. Применение подходящего фильтра для удаления шума может сгладить эти случайные изменения и улучшить четкость и качество изображения. Вот несколько способов, как фильтрация шума может положительно влиять на результаты:

Удаление случайных выбросов: Шум может представляться в виде ярких пикселей или пикселей с аномально высоким или низким значением. Фильтры, такие как медианный фильтр или фильтр сглаживания, могут удалить эти выбросы и сгладить изображение.

Восстановление деталей: Шум может размывать тонкие детали и текстуры на изображении. Фильтры, такие как усреднение, гауссовское размытие или фильтры резкости, могут улучшить видимость деталей, сгладить шум и восстановить текстуры.

Улучшение контраста: Шум может снижать контраст изображения, делая его менее четким и менее выразительным. Фильтры, такие как контрастное растяжение или адаптивное увеличение контраста, могут повысить контрастность изображения, улучшить различимость объектов и улучшить восприятие деталей.

Сглаживание шума при обработке: Если в дальнейшей обработке изображения используются алгоритмы компьютерного зрения или анализа данных, шум может негативно влиять на результаты этих алгоритмов. Фильтрация шума помогает создать более чистые данные, что может улучшить точность и надежность последующей обработки.

В целом, фильтрация шума позволяет удалить случайные искажения и повысить качество изображения, делая его более четким, более детализированным и более пригодным для последующей обработки или визуального восприятия.