**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САУ**

отчет

**по лабораторной работе №4**

**по дисциплине «Техническое зрение»**

**Тема: «Пороговые фильтры»**

|  |  |  |
| --- | --- | --- |
| Студентки гр. 6491 |  | Сергиенко С. |
|  |  | Спирова К.О |
| Преподаватель |  | Моклева К.А. |

Санкт-Петербург

2020 г.

**Цель работы**: изучить принцип применения пороговых фильтров для

обработки изображений.

**Задание на лабораторную работу**:

1) Напишите свою “легкую” реализацию cv2.threshold() только для

варианта THRESH\_BINARY. Функция должна принимать значение threshold. Пусть maxVal по умолчанию всегда будет 255.

2) Примените на практике бинаризацию всех типов для следующих

изображений (нескольких каждого типа):

a) фото дорожной разметки белого цвета;

b) изображение отсканированного текстового документа;

c) фото написанного от руки или напечатанного текста.

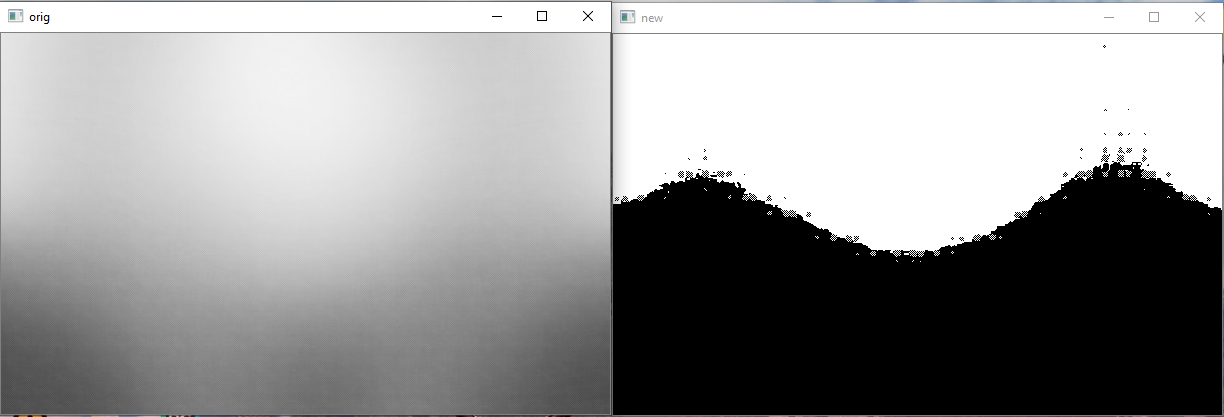
Для каждого типа опишите, какой результат был получен.

**Код программы:**

1. **import** cv2  
   **def** thresh(a):  
    **for** i **in** range(len(img)):  
    **for** j **in** range(len(img[i])):  
    **if** img[i][j] > a:  
    img[i][j] = 255  
    **else**:  
    img[i][j] = 0  
   img = cv2.imread(**'/home/user/Documents/91-1/gradient.jpg'**,cv2.IMREAD\_GRAYSCALE)  
   cv2.imshow(**'orig'**,img)  
   print(thresh(200))  
   cv2.imshow(**'new'**,img)  
   cv2.waitKey(0)

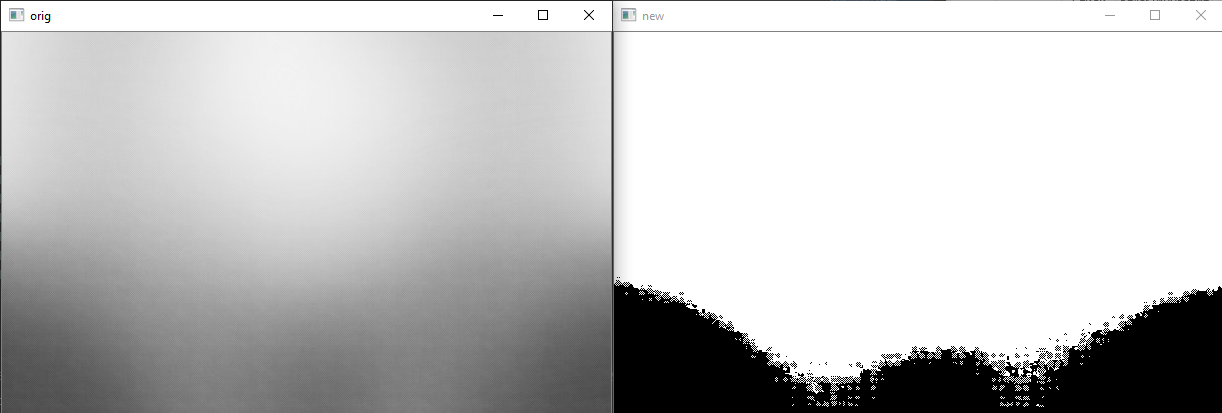
Исходное изображение:





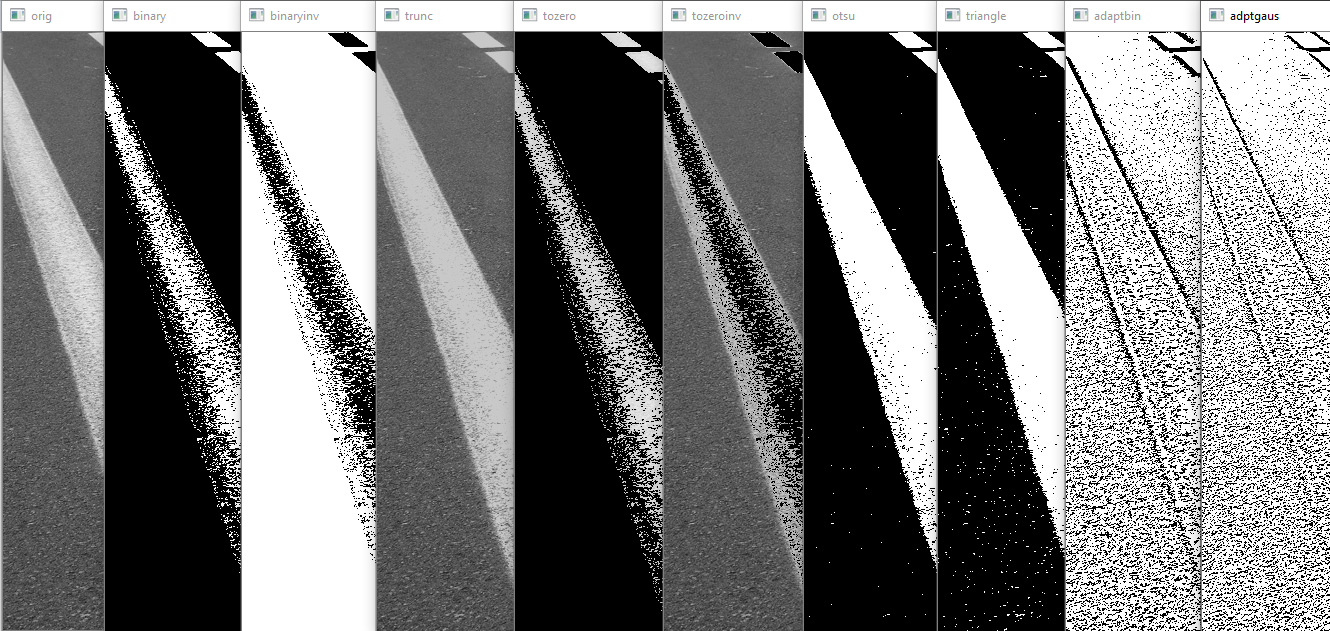
Взяв картинку градиентного перехода цветов, мы увидели, что операция бинарного разложения пороговым значением, взятым нами 200, дала изображение, в котором выделены 2 области.

Если же взять простейшее пороговое значение(127), то заметим на картинке четкое разделение на области, где преобладал более темный и более светлый оттенки.



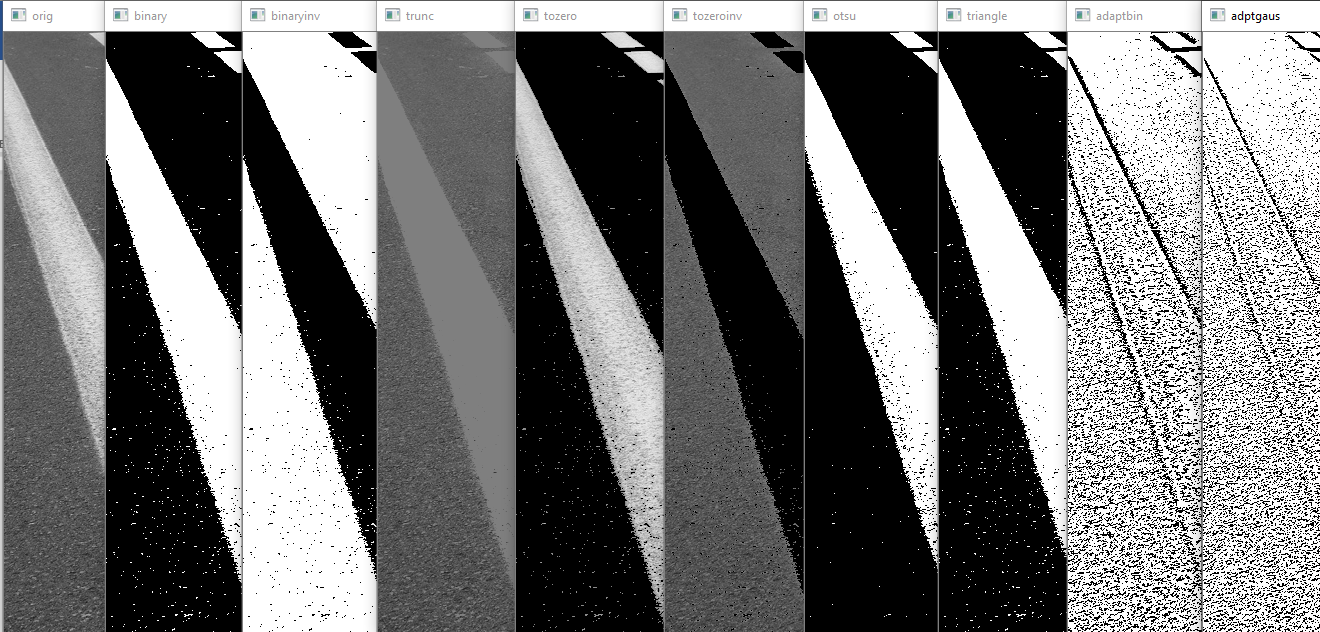
**import** cv2  
img = cv2.imread(**'D:/tzlabki/doroga.jpg'**, cv2.IMREAD\_GRAYSCALE)  
cv2.imshow(**'orig'**,img)  
new\_threshold, new\_img = cv2.threshold(img, 127 ,255, cv2.THRESH\_BINARY)  
print(**'binary'**,new\_threshold)  
cv2.imshow(**'binary'**,new\_img)  
  
new\_threshold, new\_img = cv2.threshold(img, 127 ,255, cv2.THRESH\_BINARY\_INV)  
print(**'binaryinv'**,new\_threshold)  
cv2.imshow(**'binaryinv'**,new\_img)  
  
new\_threshold, new\_img = cv2.threshold(img, 127 ,255, cv2.THRESH\_TRUNC)  
print(**'trunc'**,new\_threshold)  
cv2.imshow(**'trunc'**,new\_img)  
  
new\_threshold, new\_img = cv2.threshold(img, 127 ,255, cv2.THRESH\_TOZERO)  
print(**'tozero'**,new\_threshold)  
cv2.imshow(**'tozero'**,new\_img)  
  
new\_threshold, new\_img = cv2.threshold(img, 127 ,255, cv2.THRESH\_TOZERO\_INV)  
print(**'tozeroinv'**,new\_threshold)  
cv2.imshow(**'tozeroinv'**,new\_img)  
  
new\_threshold, new\_img1 = cv2.threshold(img, 127 ,255, cv2.THRESH\_OTSU)   
cv2.imshow(**'otsu'**,new\_img1)  
print(**'otsu'**,new\_threshold)  
  
new\_threshold, new\_img1a = cv2.threshold(img, 127 ,255, cv2.THRESH\_TRIANGLE)   
cv2.imshow(**'triangle'**,new\_img1a)  
print(**'triangle'**, new\_threshold)  
  
new\_img2 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 11, 9)  
cv2.imshow(**'adaptbin'**,new\_img2)  
  
new\_img3 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 11, 9)  
cv2.imshow(**'adptgaus'**,new\_img3)  
cv2.waitKey(0)

1. фото дорожной разметки белого цвета;



На изображении дороги при естественном освещении можно заметить, что адаптивные методы непригодны, так как добавляют шум, бинарный и другие методы с ручным заданием порога могут неплохо справляться при хорошем подборе порогового значения, но лучше всего в этом случае справляются методы с автоматическим подбором значения(otsu, triangle): разметка видна наиболее четко.

Если задать порог 127, то получим такие изображения:



Видно, что брать серединный порог-хороший вариант, теперь при фильтрации с ручным заданием порога разметка отчетливее, хоть и присутствует шум на изображениях. В любом случае, метод THRESH\_OTSU кажется самым подходящим для данной задачи.

При выводе информации о порогах, можно увидеть такие значения:

binary 127.0

binaryinv 127.0

trunc 127.0

tozero 127.0

tozeroinv 127.0

otsu 150.0

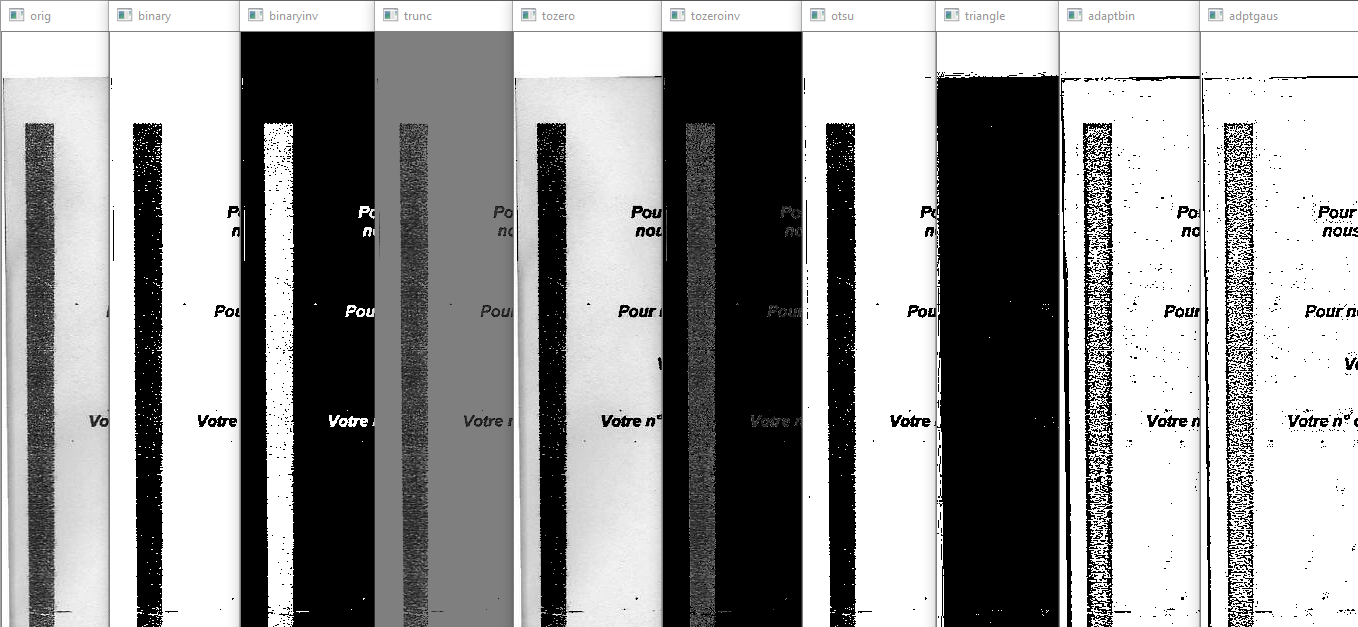
triangle 126.0

1. изображение отсканированного текстового документа;



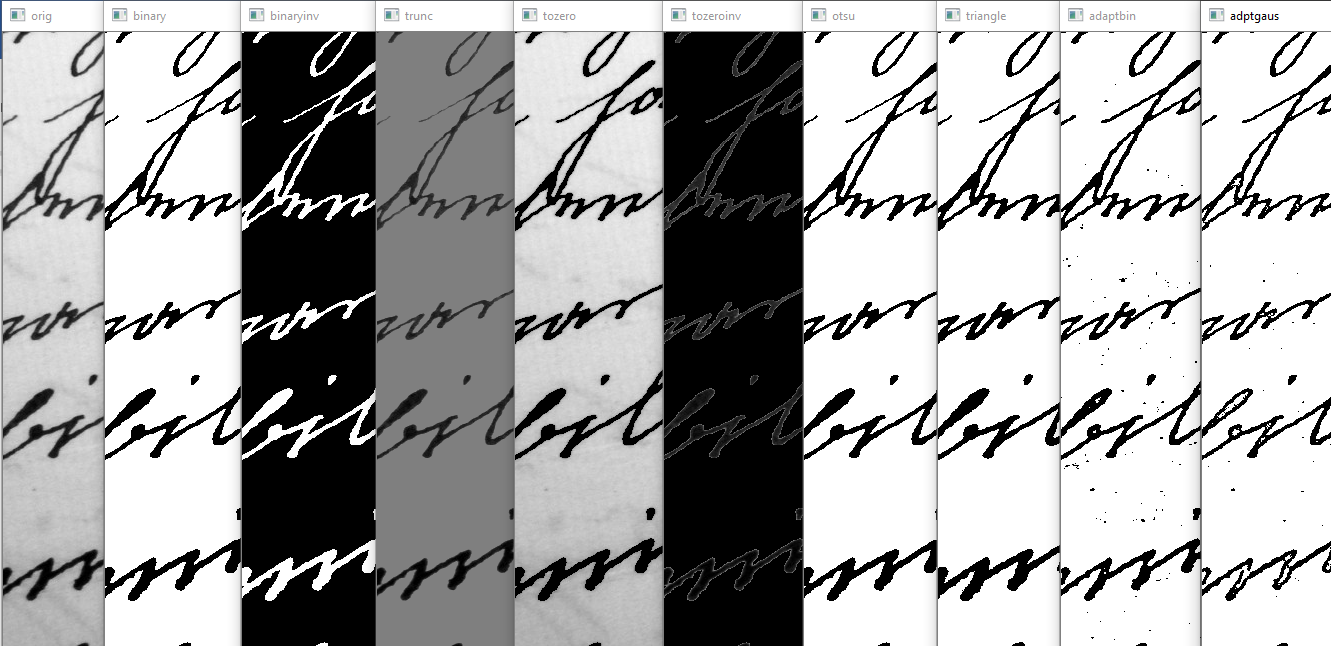
В данном случае лучше всего справляется метод THRESH\_TOZERO, так как яркость примерно на одном уровне, в сравнении с исходником текст виден четче, все остальные методы ухудшают читабельность в разной степени. Метод THRESH\_TRIANGLE добавляет шум на изображение, в то время как другой метод с автоматическим подбором порога THRESH\_OTSU справляется неплохо, но все же ухудшает изображение. В случае адаптивных методов за счет хорошего подбора параметров получили довольно хорошие результаты.

Другой скан:



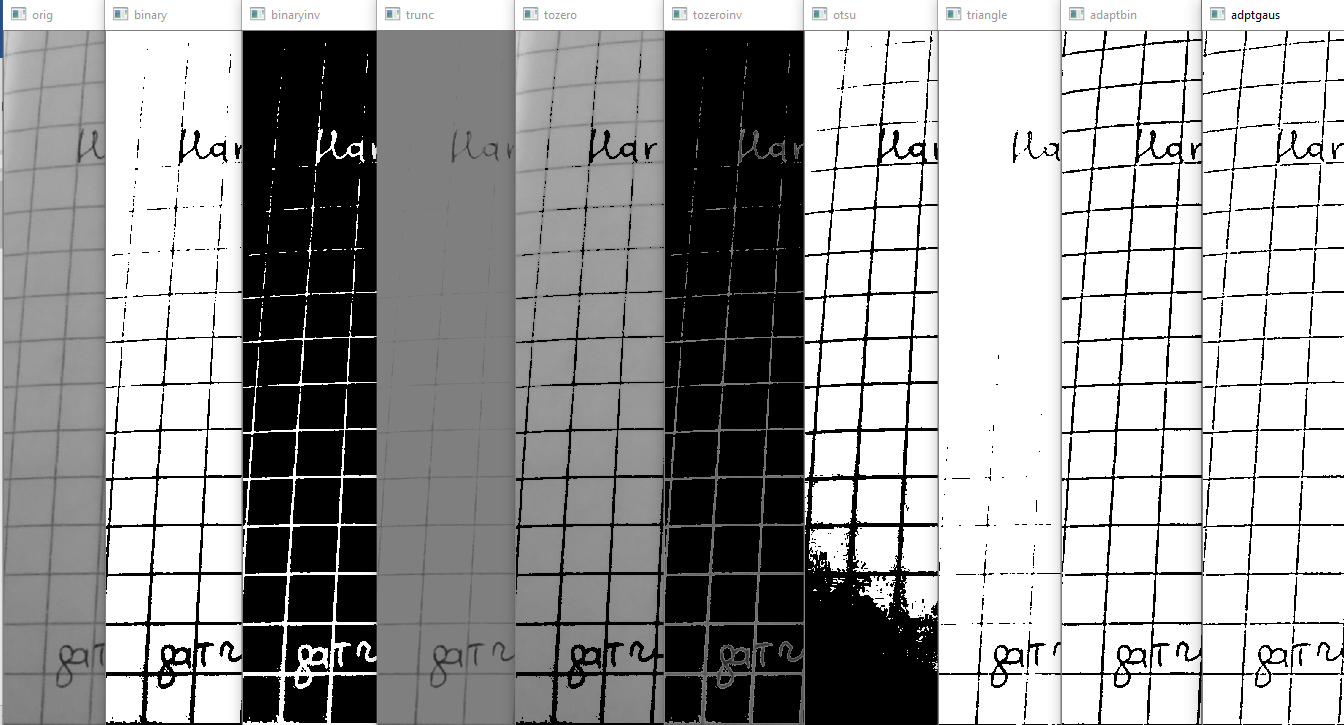
Метод THRESH\_TRIANGLE сделал изображение полностью черным. Адаптивная фильтрация добавляет шум, но все же оставляет читабельность. THRESH\_TOZERO делает ярче и резче, что повышает качество изображения. Остальные методы дают худший результат, буквы становятся сложно воспринимаемыми.

c) фото написанного от руки или напечатанного текста.



Адаптивные методы добавляют шум, хотя эффект неплохой. В целом, можно отметить, что с фото текста все методы справляются хорошо, так как исходное изображение хорошее и на нем не присутствует засветов и прочего.

Cделаем то же самое с фотографией текста при искусственном освещении:



Тут большинство методов не справляются с фильтрацией и появляются участки, перекрывающие текст. Отчетливее всего виден текст при адаптивных методах, которые смогли сохранить и текст, и клетку и THRESH\_TOZERO, хотя и бинарный тип очень отчетливо выделил написанный от руки текст, сохранив его полностью.

**Вывод:** в ходе данной лабораторной работы были изучены пороговые фильтры. Применив разные ко всем типам изображений выяснили, что хуже всего работает метод THRESH\_TRUNC, делая изображения серыми и тусклыми. Адаптивные методы добавляют на изображения шум. Ручное задание порога может давать хорошие результаты при более длительном подборе значений.