**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.

|  |  |  |
| --- | --- | --- |
| Студент гр. 6492 |  | Михайлов Н.К. |
| Преподаватель |  | Моклева К.А. |

Санкт-Петербург

2020

**Цель работы:** изучить принцип применения пороговых фильтров для обработки изображений.

**Задания:**

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, насколько

конкретный тип пороговой фильтрации для него применим и почему.

**Ход работы.**

**import** cv2

**import** numpy **as** np

**def** threshold\_bin(threshold):

**for** x **in** range(img.shape[0]):

**for** y **in** range(img.shape[1]):

**if** img[x,y] > threshold:

img[x,y] = 255

**else**:

img[x,y] = 0

img = cv2.imread('/home/user/Documents/92-2/lab3/img.jpg', cv2.IMREAD\_GRAYSCALE)

threshold\_bin(150)

cv2.imshow('BINARY', img)

cv2.waitKey(0)



Рисунок 1. Исходное изображение.



Рисунок 2. Изображение после применения фильтра.

**import** cv2

**import** numpy **as** np

img = cv2.imread('/home/user/Documents/92-2/lab4/img\_name.jpg', cv2.IMREAD\_GRAYSCALE)

bin\_threshold, bin\_img = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)

otsu\_threshold, otsu\_img = cv2.threshold(img, 0, 255, cv2.THRESH\_OTSU)

tri\_threshold, tri\_img = cv2.threshold(img, 0, 255, cv2.THRESH\_TRIANGLE)

mean\_img = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 5, 15)

gauss\_img = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 5, 15)

cv2.imshow('BINARY', bin\_img)

cv2.imshow('OTSU', otsu\_img)

cv2.imshow('TRIANGLE', tri\_img)

cv2.imshow('MEAN', mean\_img)

cv2.imshow('GAUSS', gauss\_img)

cv2.waitKey(0)



Рисунок 3. Исходное изображение.

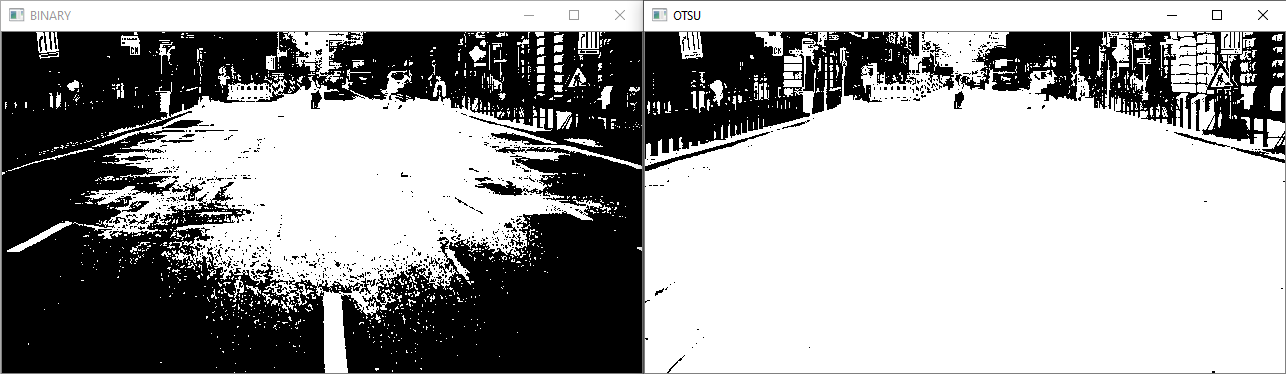




Рисунок 4. Изображение после применения фильтров.



Рисунок 5. Исходное изображение.

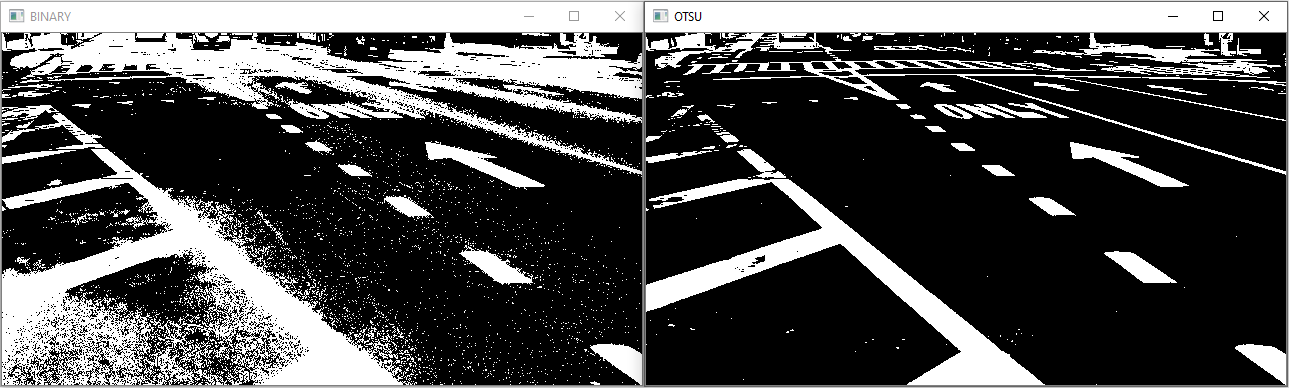




Рисунок 6. Изображение после применения фильтров.



Рисунок 7. Исходное изображение.



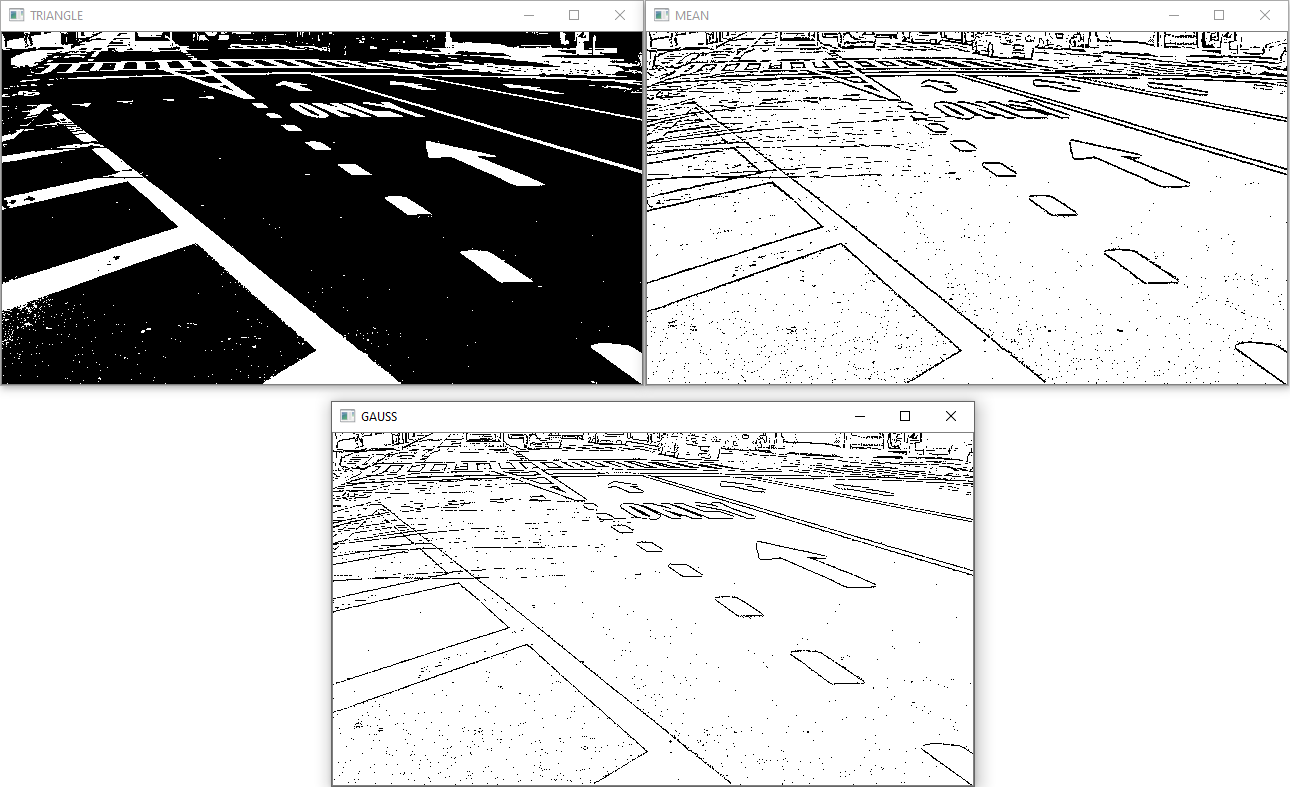


Рисунок 8. Изображение после применения фильтров.

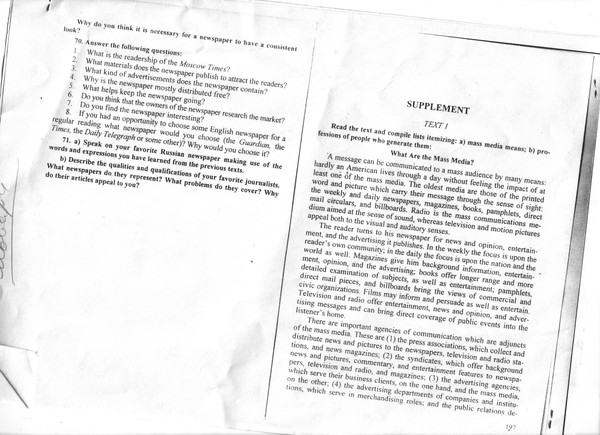
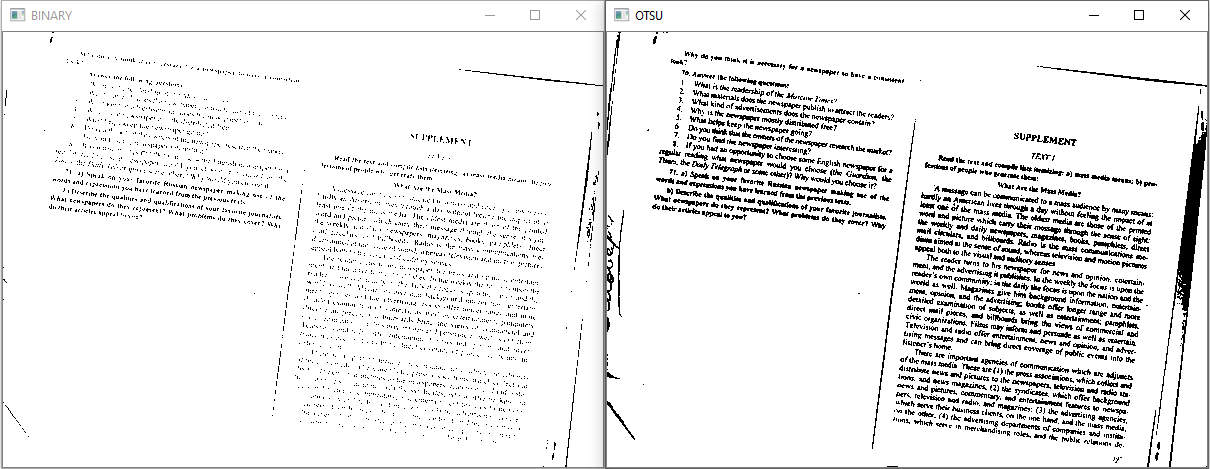


Рисунок 9. Исходное изображение.



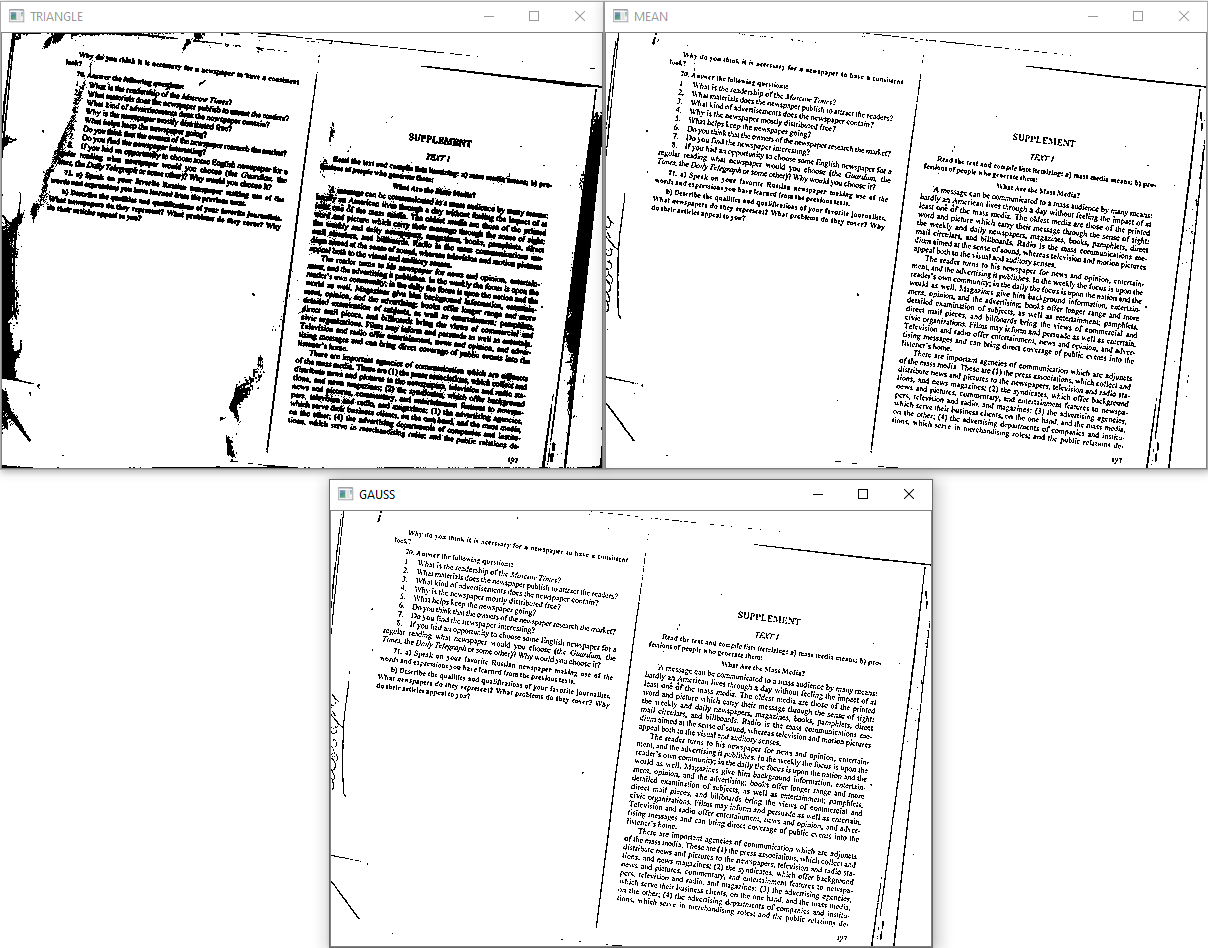


Рисунок 10. Изображение после применения фильтров.

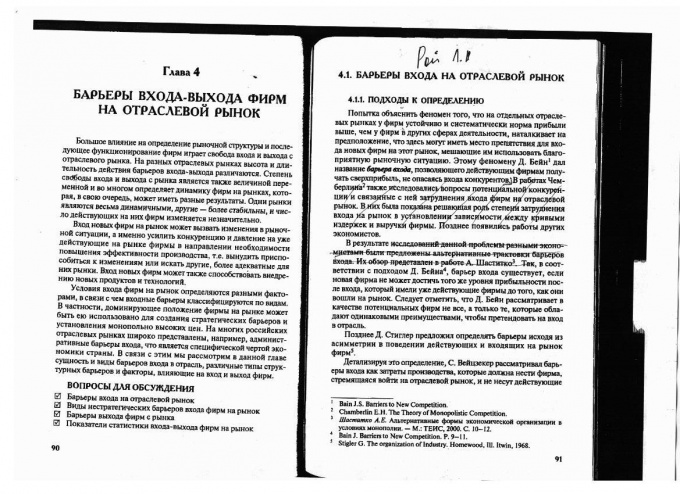
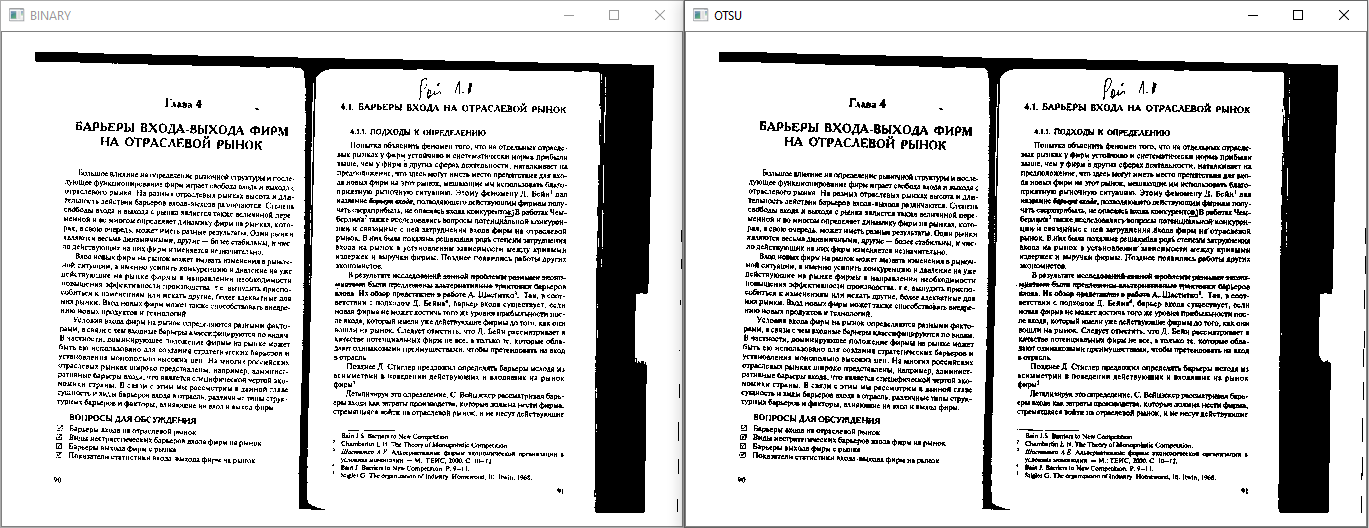
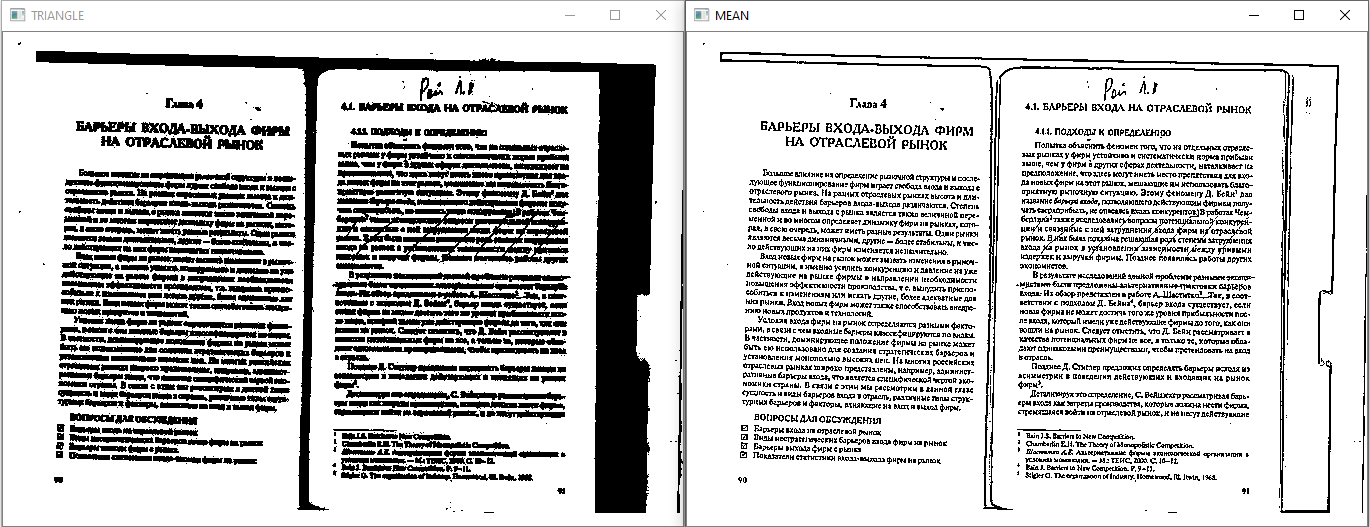


Рисунок 11. Исходное изображение.





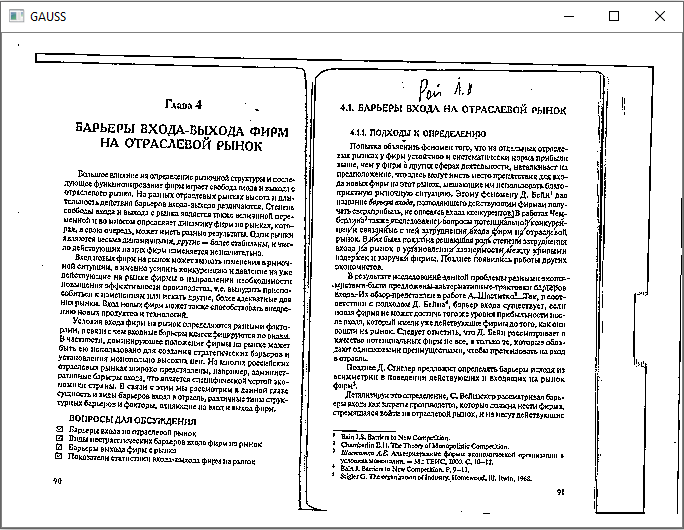


Рисунок 12. Изображение после применения фильтров.

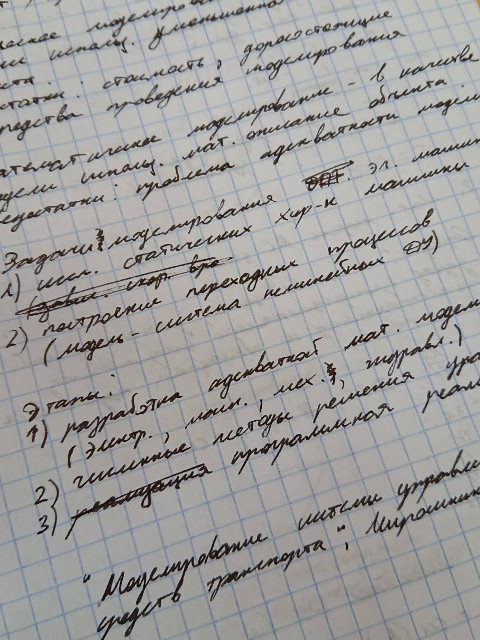
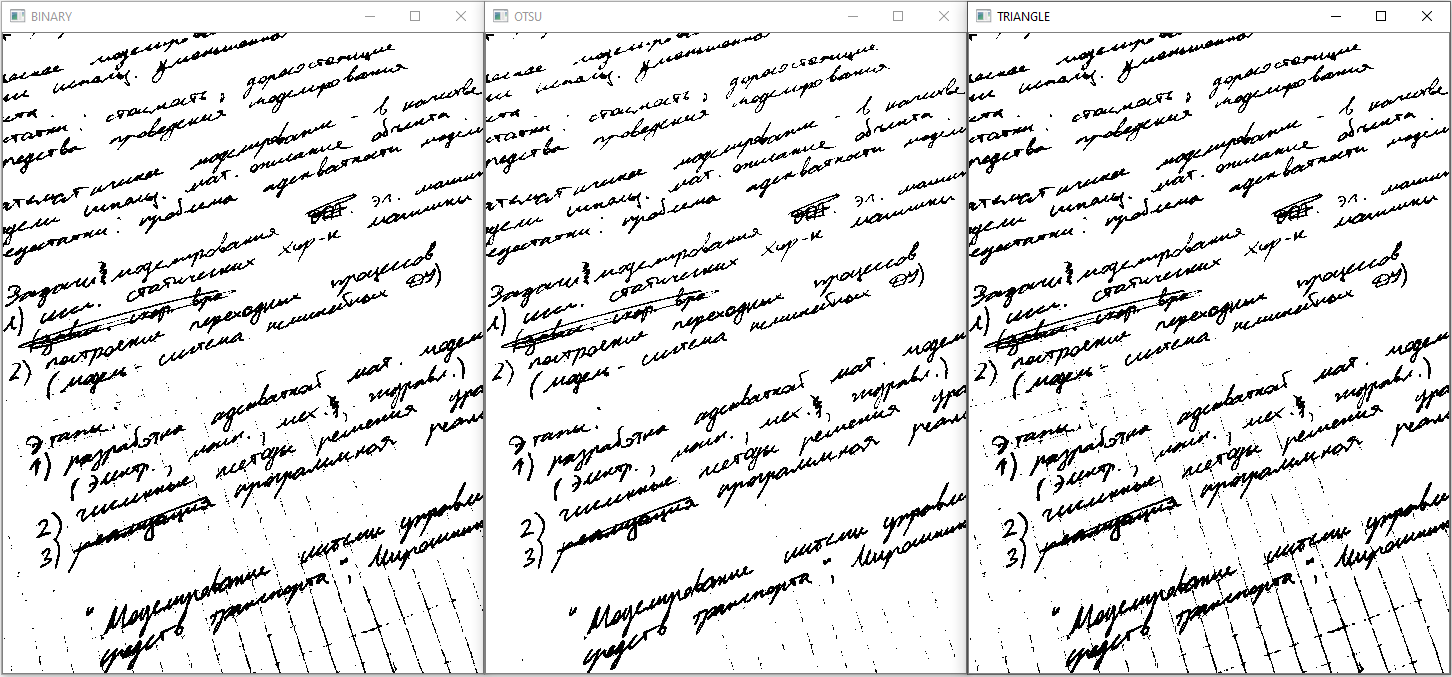


Рисунок 13. Исходное изображение.



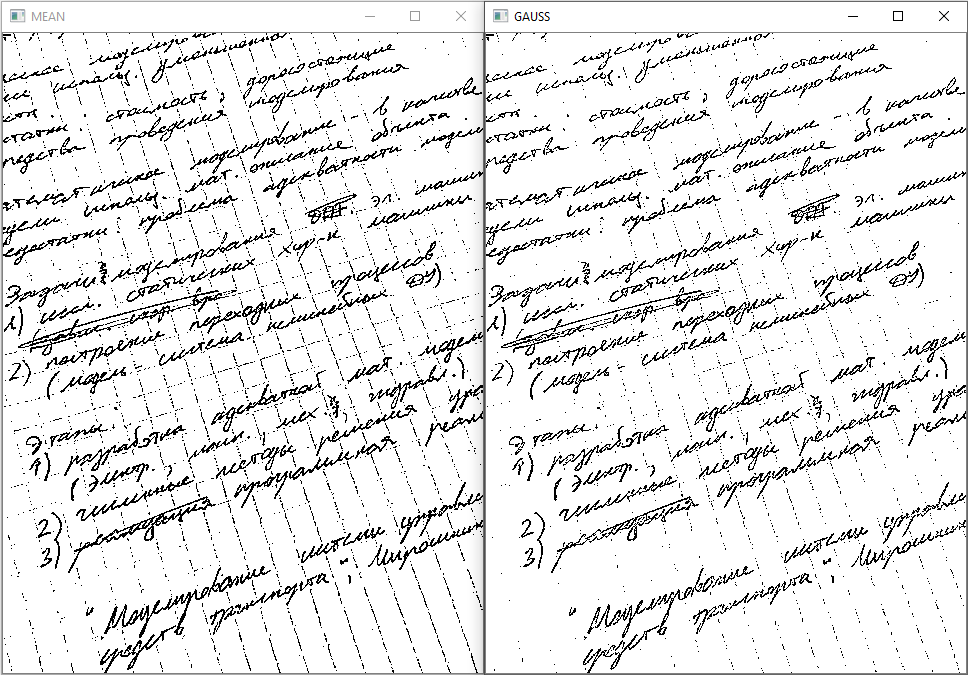


Рисунок 14. Изображение после применения фильтров.

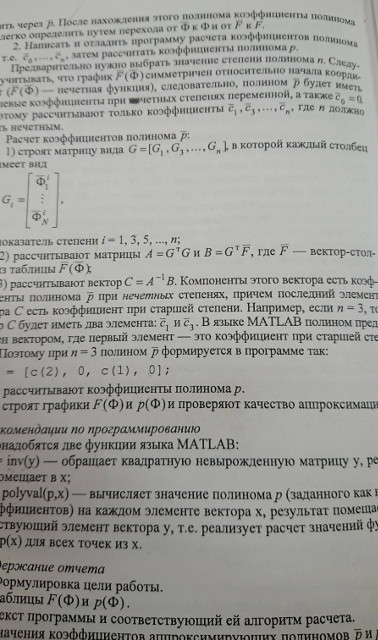
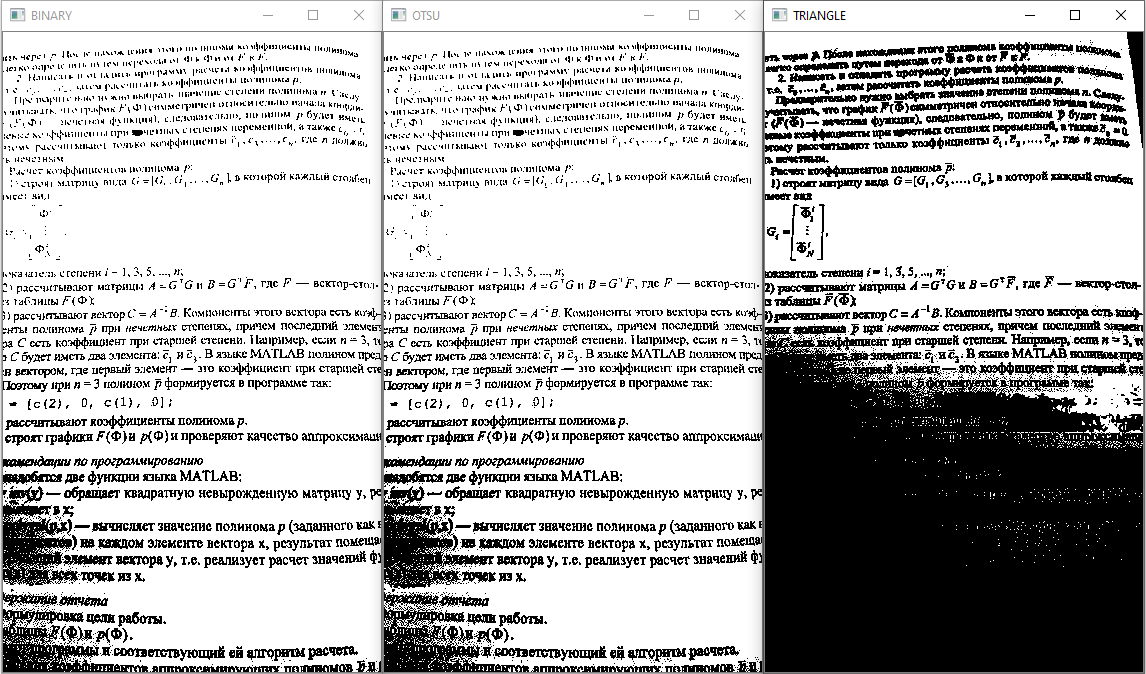


Рисунок 15. Исходное изображение.



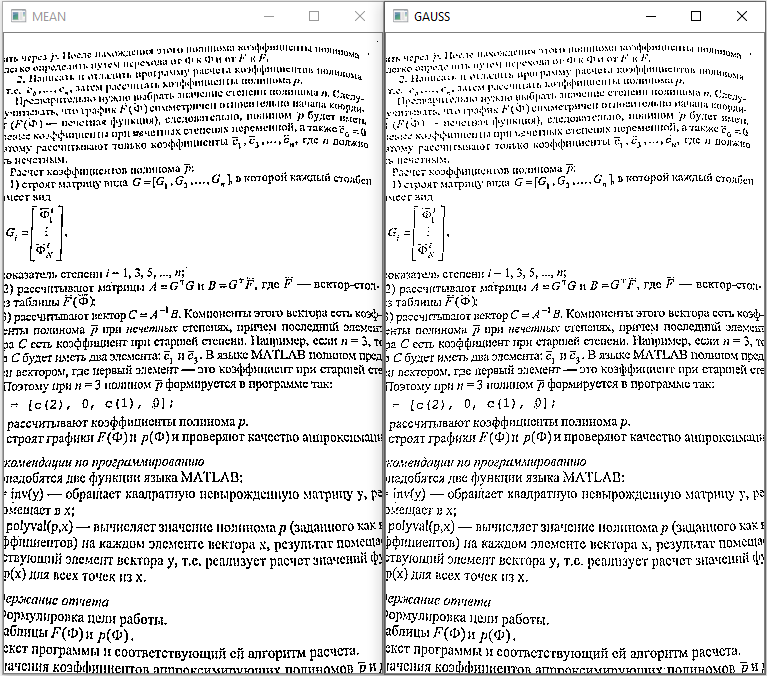


Рисунок 16. Изображение после применения фильтров.

**Вывод:** в ходе выполнения работы была изучена работа пороговых фильтров. Исходя из полученных результатов, можно сделать следующие выводы:

* Для фильтрации изображения дороги лучше всего подходит бинаризация типа TRIANGLE, так как при использовании BINARY или OTSU при недостаточной контрастности изображения разметка сливается с дорогой, а MEAN и GAUSS выделяют только переходы от светлой части исходного изображения к темной, из-за чего на отфильтрованном изображении виден только контур разметки.
* Для фильтрации фотографии или скана текста лучше всего подходят адаптивные методы. При использовании неадаптивных методов текст на затененных участках изображения становится нечитаемым.