

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. 6491

Бузи Дарья

Преподаватель

Моклева К.А.

Санкт-Петербург
2020

Лабораторная работа 4

Пороговые фильтры

Цель работы: изучить принцип применения пороговых фильтров для обработки изображений

Ход работы:

1) Написание “легкой” реализации cv2.threshold() только для варианта THRESH_BINARY. Функция должна принимать значение threshold. Пусть maxVal по умолчанию всегда будет 255.

Код работы программы:

```
import cv2
import numpy

imagecolour = cv2.imread('/Users/dariabusi/Desktop/kirby.jpg',cv2.IMREAD_GRAYSCALE)
cv2.imshow ('KirbyONE', imagecolour)

(x, y) = imagecolour.shape
for i in range (0,x):
    for j in range (0,y):
        if imagecolour[i][j] >= 125:
            imagecolour[i][j]=255
        else:
            imagecolour[i][j]=0

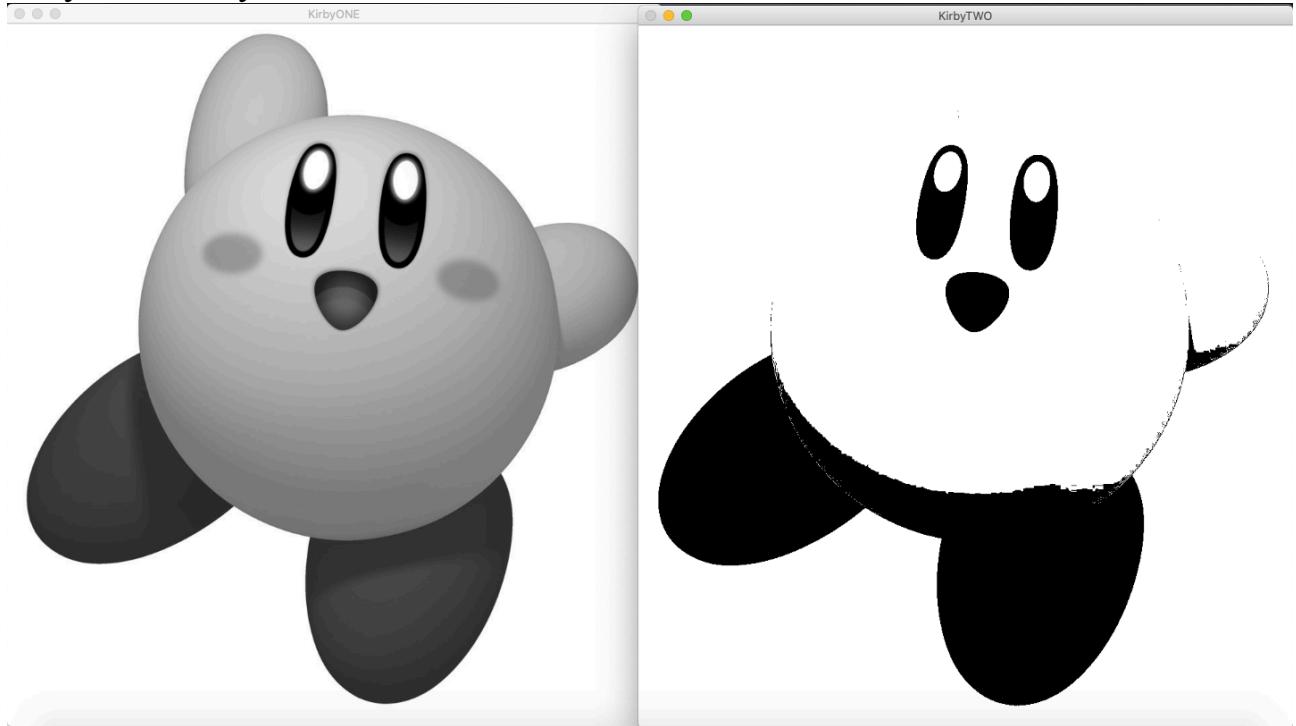
cv2.imshow ('KirbyTWO',imagecolour)

cv2.waitKey(0)
```

Рисунок 1: Код программы

```
1  import cv2
2  import numpy
3
4  imagecolour = cv2.imread('/Users/dariabusi/Desktop/kirby.jpg',cv2.IMREAD_GRAYSCALE)
5  cv2.imshow ('KirbyONE', imagecolour)
6
7  (x, y) = imagecolour.shape
8  for i in range (0,x):
9      for j in range (0,y):
10         if imagecolour[i][j] >= 125:
11             imagecolour[i][j]=255
12         else:
13             imagecolour[i][j]=0
14
15 cv2.imshow ('KirbyTWO',imagecolour)
16
17 cv2.waitKey[0]
```

Рисунок 2: Результат



2) Применение на практике бинаризации 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.

Цель фильтрования следующих картинок: для всех дорог после фильтрования должна выделяться четко дорожные полосы без каких либо пятен, для всех текстов надпись без пятен должна четко вырисовываться.

Простые Пороговые Фильтры

Код работы программы (пример для одной картинки):

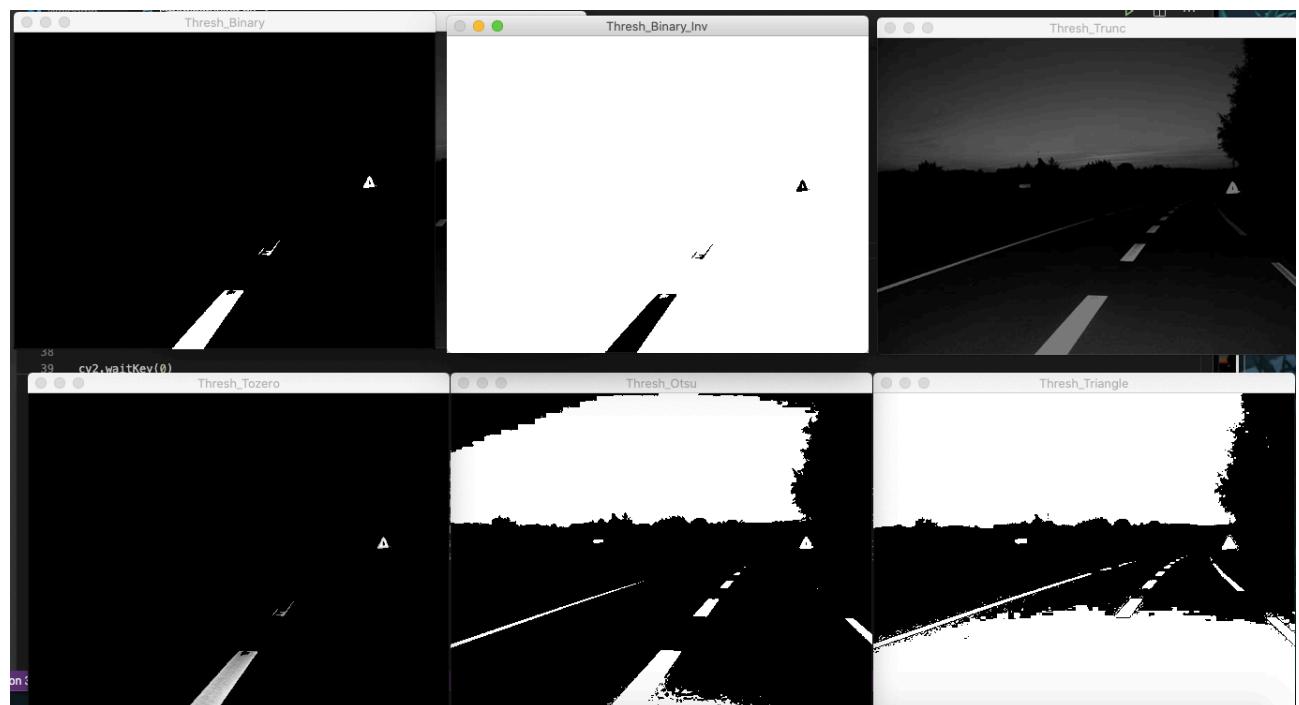
```
1. Dayroad = cv2.imread('/Users/dariabusi/Desktop/  
nightroad.jpg',cv2.IMREAD_GRAYSCALE)#считываем файл, который возвращает #массив с  
данными об изображении, читается в GRAYSCALE (серый оттенок)  
2. cv2.imshow ('INITIAL',Dayroad) #выводим результат  
3.  
4. new_threshold, img = cv2.threshold(Dayroad, 122 , 255 ,  
cv2.THRESH_BINARY)#применение порогового фильтра типа THRESH_BINARY  
5. cv2.imshow ('Thresh_Binary' , img)  
6.  
7. new_threshold, img = cv2.threshold(Dayroad, 122 , 255 ,  
cv2.THRESH_BINARY_INV)#применение порогового фильтра типа #THRESH_BINARY_INV  
8. cv2.imshow ('Thresh_Binary_Inv' , img)  
9.  
10. new_threshold, img = cv2.threshold(Dayroad, 122 , 255 ,  
cv2.THRESH_TRUNC)#применение порогового фильтра типа THRESH_TRUNC  
11. cv2.imshow ('Thresh_Trunc' , img)  
12.  
13. new_threshold, img = cv2.threshold(Dayroad, 100 , 255 ,  
cv2.THRESH_TOZERO)#применение порогового фильтра типа THRESH_TOZERO  
14. cv2.imshow ('Thresh_Tozero' , img)  
15.  
16. new_threshold, img = cv2.threshold(Dayroad, 122 , 255 , cv2.THRESH_OTSU)#применение  
порогового фильтра типа THRESH_OTSU  
17. cv2.imshow ('Thresh_Otsu' , img)  
18.
```

```
19. new_threshold, img = cv2.threshold(Dayroad, 122, 255 ,  
cv2.THRESH_TRIANGLE)#применение порогового фильтра типа #THRESH_TRIANGLE  
20. cv2.imshow ('Thresh_Triangle', img)
```

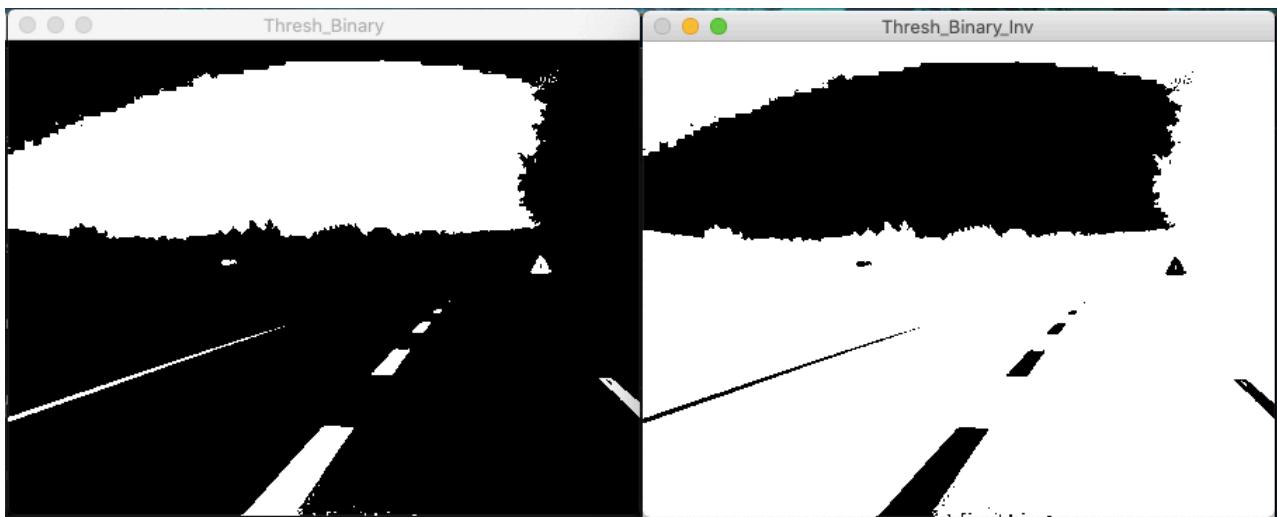
Картинка 1: Ночная дорога



Начальная картинка



Отфильтрованные картинки (threshold=122, MaxValue=225)



Отфильтрованные картинки (threshold=45, MaxValue=225)



Отфильтрованные картинки (threshold=75, MaxValue=225)

Рассуждение: Яркие полосы на темном фоне хорошо выделяются. Для ночной дороги лучше фильтровать через threshold binary и tozero при правильном выборе значения порога (лучше результат при меньшем threshold value), так картинка дороги с полосой получается наиболее четкой. Также лучше использовать OTSU вместо TRIANGLE, так как полученная отфильтрованная картинка имеет меньше шума, ведь TRIANGLE выбрал более высокое пороговое значение.

Картина 2: Дневная дорога (равномерный свет)



Начальная картинка



Отфильтрованные картинки (threshold=122, MaxValue=225)



Отфильтрованные картинки (threshold=200, MaxValue=225)

Рассуждение: При равномерном солнечном свете асфальт слияется с белыми полосами. Tozero и Binary смогли выделить полосу при высоком threshold value, но картинки получились нечеткими, есть шумы. Только Triangle смог выдать картинку без шумов

Картинка 3: Дорога с бликами



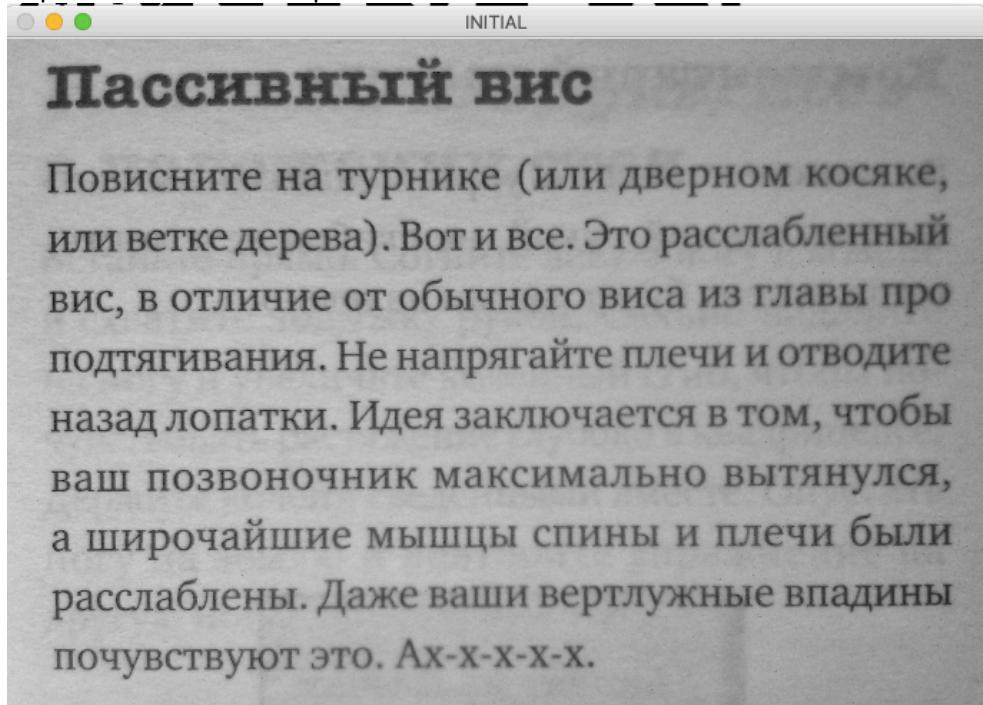
Начальная картинка



Отфильтрованные картинки (threshold=122, MaxValue=225)

Рассуждение: С картинкой третьей невозможно подобрать такой отдельный пороговый фильтр, который позволит четко отделить яркие блики на дороге от белых полос. Необходимо использовать другой тип фильтра.

Картинка 4: Отсканированный текст



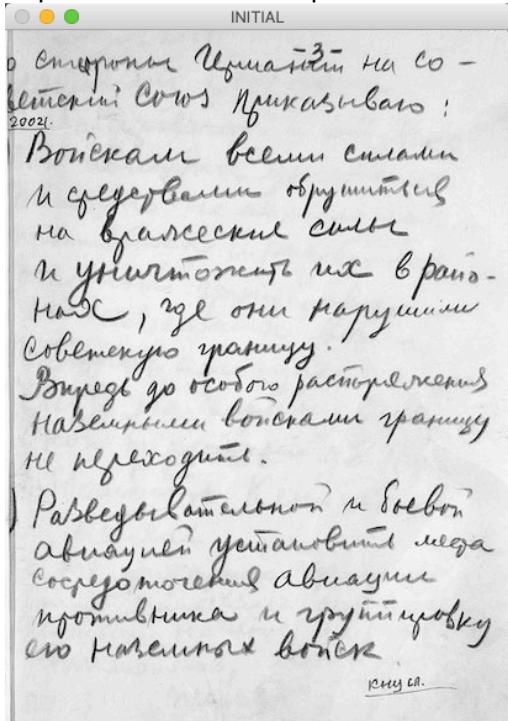
Начальная картинка



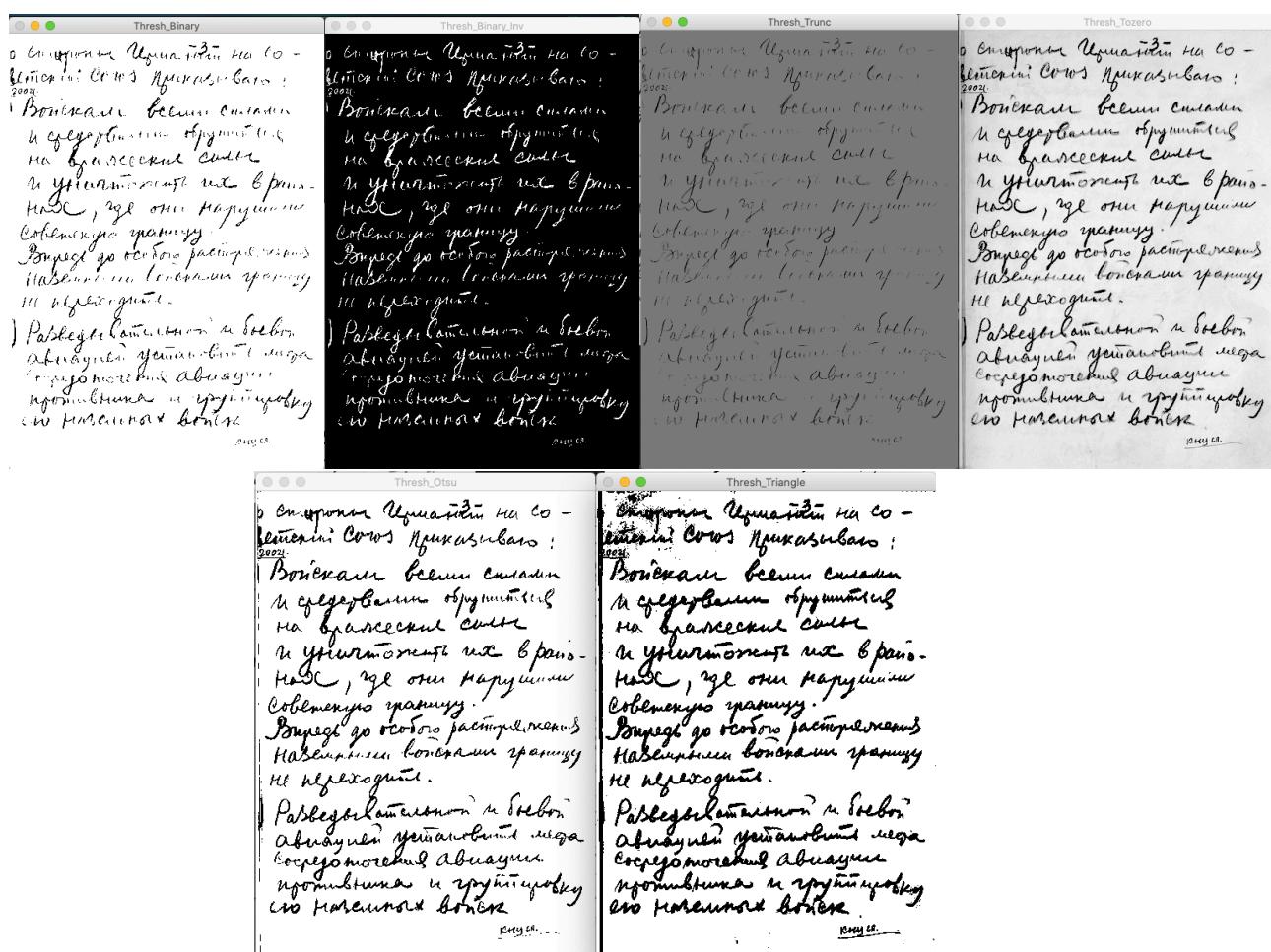
Отфильтрованные картинки (threshold=122, MaxValue=225)

Рассуждение: С помощью Binary можно получить текст с чистым фоном, но надпись нехорошо вырисовывается. Tozero позволяет наиболее ярко выделить текст, поэтому эту версию яснее читать. Через Triangle и OTSU картинки получаются с шумами, но текст читабельный.

Картинка 5: Отсканированный текст написанный от руки



Начальная картинка



Отфильтрованные картинки (threshold=122, MaxValue=225)

Рассуждение: В этом случае чернила не распределены равномерно. Поэтому для Binary фильтра нужно повысить порог. Tozero позволяет выделить эти неявные области, этот фильтр наиболее эффективный. Через Triangle и OTSU картинки получаются с шумами, но текст читабельный.

АДАПТИВНЫЕ ПОРОГОВЫЕ ФИЛЬТРЫ

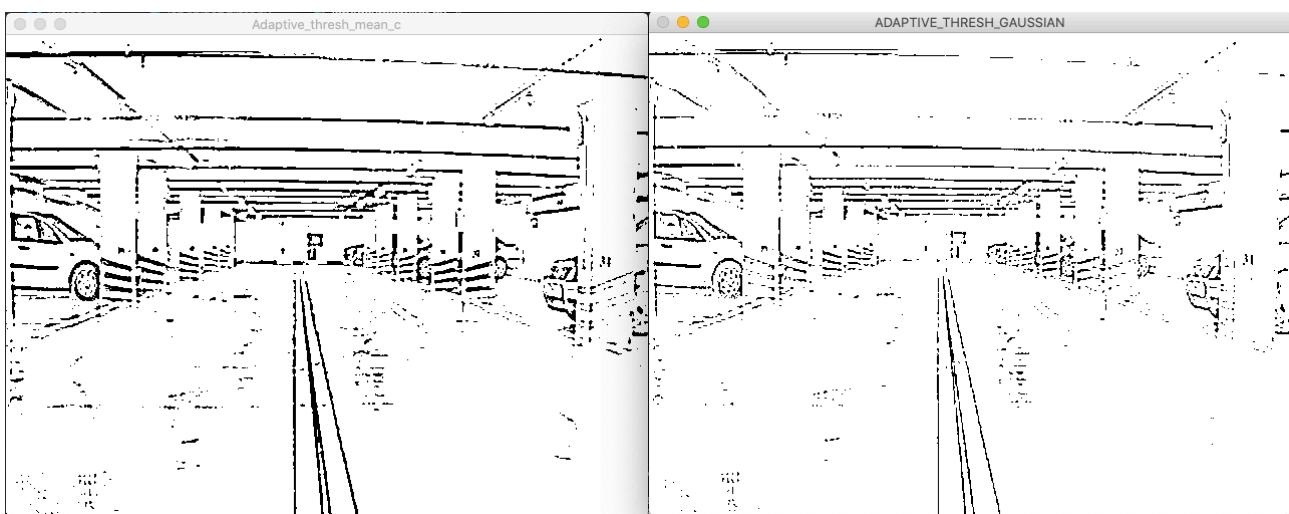
Код работы программы (пример для одной картинки):

```
21. img = cv2.adaptiveThreshold(Dayroad, 255, cv2.ADAPTIVE_THRESH_MEAN_C,  
cv2.THRESH_BINARY, 11, 20) #применение порогового #фильтра типа THRESH_BINARY с  
адаптивным поиском порога типа ADAPTIVE_THRESH_MEAN_C  
22. cv2.imshow ('Adaptive_thresh_mean_c', img)  
23.  
24. img = cv2.adaptiveThreshold(Dayroad, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,  
cv2.THRESH_BINARY, 11, 20) #применение порогового #фильтра типа THRESH_BINARY с  
адаптивным поиском порога типа ADAPTIVE_THRESH_GAUSSIAN_C  
25. cv2.imshow ('ADAPTIVE_THRESH_GAUSSIAN', img)
```

Картина 3: Дорога с бликами



Начальная картинка



Отфильтрованные картинки (Block Size = 11, C = 20)

Рассуждение: Адаптивными фильтрами можно избавиться от бликов, четко вырисовываются полосы, но фон получается с шумами (но меньше, чем с предыдущими фильтрами).

ИТОГОВЫЙ КОД

```
1. import cv2 #подключение библиотеки opencv
2. import numpy
3.
4. imagecolour = cv2.imread('/Users/dariabusi/Desktop/
kirby.jpg',cv2.IMREAD_GRAYSCALE) #читываем файл, который возвращает массив с
данными об изображении, читается в GRAYSCALE (серый оттенок)
5.
6. cv2.imshow ('KirbyONE', imagecolour)#выводим результат
7.
8. #алгоритм порогового фильтра BINARY
9. (x, y) = imagecolour.shape
10. for i in range (0,x):
11.     for j in range (0,y):
12.         if imagecolour[i][j] >= 125:
13.             imagecolour[i][j]=255
14.         else:
15.             imagecolour[i][j]=0
16.
17. cv2.imshow ('KirbyTwo',imagecolour) #выводим результат
18.
19. Dayroad = cv2.imread('/Users/dariabusi/Desktop/
nightroad.jpg',cv2.IMREAD_GRAYSCALE)#читываем файл, который возвращает массив с
данными об изображении, читается в GRAYSCALE (серый оттенок)
20. cv2.imshow ('INITIAL',Dayroad) #выводим результат
21.
22. new_threshold, img = cv2.threshold(Dayroad, 122 , 255 ,
cv2.THRESH_BINARY)#применение порогового фильтра типа THRESH_BINARY
23. cv2.imshow ('Thresh_Binary', img)
24.
25. new_threshold, img = cv2.threshold(Dayroad, 122 , 255 ,
cv2.THRESH_BINARY_INV)#применение порогового фильтра типа #THRESH_BINARY_INV
26. cv2.imshow ('Thresh_Binary_Inv', img)
27.
28. new_threshold, img = cv2.threshold(Dayroad, 122 , 255 ,
cv2.THRESH_TRUNC)#применение порогового фильтра типа THRESH_TRUNC
29. cv2.imshow ('Thresh_Trunc', img)
30.
31. new_threshold, img = cv2.threshold(Dayroad, 100 , 255 ,
cv2.THRESH_TOZERO)#применение порогового фильтра типа THRESH_TOZERO
32. cv2.imshow ('Thresh_Tozero', img)
33.
34. new_threshold, img = cv2.threshold(Dayroad, 122 , 255 , cv2.THRESH_OTSU)#применение
порогового фильтра типа THRESH_OTSU
35. cv2.imshow ('Thresh_Otsu', img)
36.
37. new_threshold, img = cv2.threshold(Dayroad, 122 , 255 ,
cv2.THRESH_TRIANGLE)#применение порогового фильтра типа #THRESH_TRIANGLE
38. cv2.imshow ('Thresh_Triangle', img)
39.
40. img = cv2.adaptiveThreshold(Dayroad, 255 , cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY, 11, 20)#применение порогового #фильтра типа THRESH_BINARY с
адаптивным поиском порога типа ADAPTIVE_THRESH_MEAN_C
41. cv2.imshow ('Adaptive_thresh_mean_c', img)
42.
43. img = cv2.adaptiveThreshold(Dayroad, 255 , cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 11, 20)#применение порогового #фильтра типа THRESH_BINARY с
адаптивным поиском порога типа ADAPTIVE_THRESH_GAUSSIAN_C
44. cv2.imshow ('ADAPTIVE_THRESH_GAUSSIAN', img)
45.
46. cv2.waitKey(0)#ожидание нажатия клавиши
```