МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра: Программной инженерии**

Направление подготовки: «Программная инженерия»

«Обработка изображений»

**Отчёт по лабораторной работе**

|  |  |
| --- | --- |
|  | Выполнил:  студент группы 381908-3  Новожилов Александр Юрьевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  Проверил:  Гетманская А.А  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

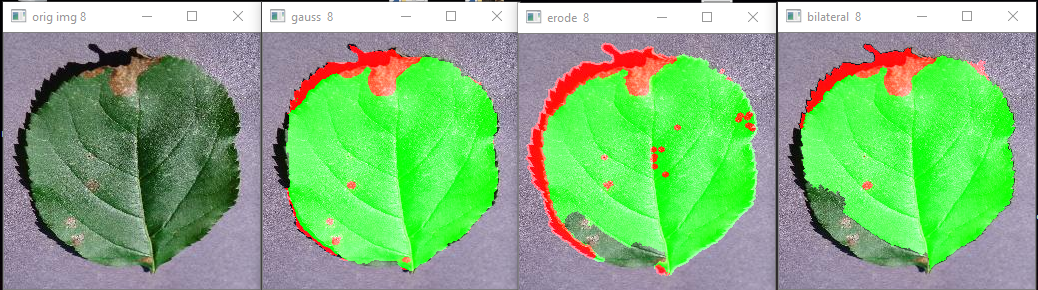
Нижний Новгород  
2021 г.

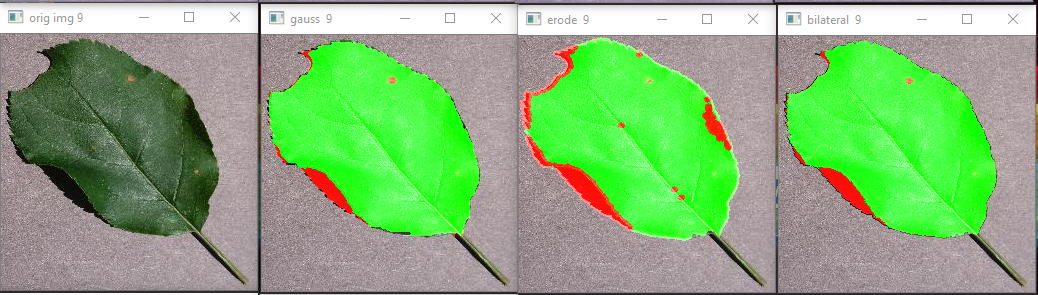
# Задача

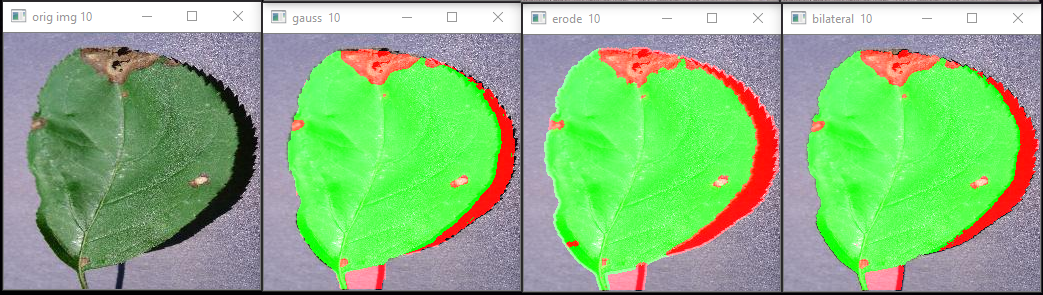
Выделить на изображении здоровую часть листа и повреждения листа. Использовать watershed и фильтры, уменьшающие шум. Написать какие методы, с какими параметрами использовали, какие оказались наилучшими.

# Результаты работы

Для устранения шума на изображении было использовано 3 фильтра: Gaussian filter, Erosion Filter и Bilateral. Размер матрицы был выбран 7\*7. Ниже приведены результаты работы программы с использованием этих фильтров:







Учитывая получившиеся результаты, можно сделать вывод, что фильтр Гаусса лучше справился с поставленной задачей.

**Код программы**

import cv2 as cv  
import numpy as np  
def calc\_of\_damage\_and\_non\_damage(image):  
 hsv\_img = cv.cvtColor(image, cv.COLOR\_BGR2HSV)  
 markers = np.zeros((image.shape[0], image.shape[1]), "int32")  
 markers[90:140, 90:140] = 255  
 markers[236:255, 0:20] = 1  
 markers[0:20, 0:20] = 1  
 markers[0:20, 236:255] = 1  
 markers[236:255, 236:255] = 1  
  
 leafs\_area\_BGR = cv.watershed(image, markers)  
 healthy\_part = cv.inRange(hsv\_img, (33, 25, 25), (86, 255, 255))  
 ill\_part = leafs\_area\_BGR - healthy\_part  
 mask = np.zeros\_like(image, np.uint8)  
 mask[leafs\_area\_BGR > 1] = (0, 255, 0)  
 mask[ill\_part > 1] = (0, 0, 255)  
 return mask  
  
def bilateral\_filter(image):  
 return cv.bilateralFilter(image, 15, 75, 75)  
  
def gaussian\_filter(image):  
 return cv.GaussianBlur(image, (7, 7), cv.BORDER\_DEFAULT)  
  
def erode\_filter(image):  
 kernel = cv.getStructuringElement(cv.MORPH\_ELLIPSE, (7, 7))  
 return cv.erode(image, kernel)  
  
def get\_result\_image(image, mask):  
 return cv.add(image, mask)  
  
  
path = "C:\\Users\\sasha\\PycharmProjects\\ImageProcessingHomeWork2\\test\\"  
img\_list = []  
for i in range(1, 13):  
 filename = ""  
 filename += path + str(i) + ".jpg"  
 img\_list.append(cv.imread(filename))  
for i in range(8, 12):  
 cv.imshow("orig img "+str(i), img\_list[i])  
 bilateral = bilateral\_filter(img\_list[i])  
 erode = erode\_filter(img\_list[i])  
 gauss = gaussian\_filter(img\_list[i])  
 result\_image = get\_result\_image(img\_list[i], calc\_of\_damage\_and\_non\_damage(bilateral))  
 cv.imshow("bilateral " + str(i), result\_image)  
 result\_image = get\_result\_image(img\_list[i], calc\_of\_damage\_and\_non\_damage(erode))  
 cv.imshow("erode " + str(i), result\_image)  
 result\_image = get\_result\_image(img\_list[i], calc\_of\_damage\_and\_non\_damage(gauss))  
 cv.imshow("gauss " + str(i), result\_image)  
  
k = cv.waitKey(0)