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  Имя  Бражник Дмитрий \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  Выполнил:  студент группы 381908-3  Имя  Черемушкин Кирилл \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  Выполнил:  студент группы 381908-3  Имя  Козел Светлана\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  Проверил:  Гетманская А.А  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Нижний Новгород  
2021 г.

Содержание

[Задача 3](#_Toc86255925)

[Проделанная работа 3](#_Toc86255926)

[Анализ полученных результатов 3](#_Toc86255927)

[Библиотеки 4](#_Toc86255928)

[Код программы 4](#_Toc86255929)

[Вывод 6](#_Toc86255930)

# Задача

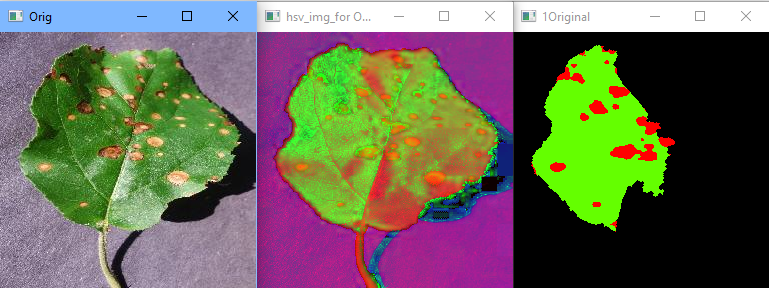
Выделить на изображении здоровую часть листа и повреждения листа. Использовать watershed и фильтры, уменьшающие шум. Написать какие методы, с какими параметрами использовали, какие оказались наилучшими.

# Проделанная работа

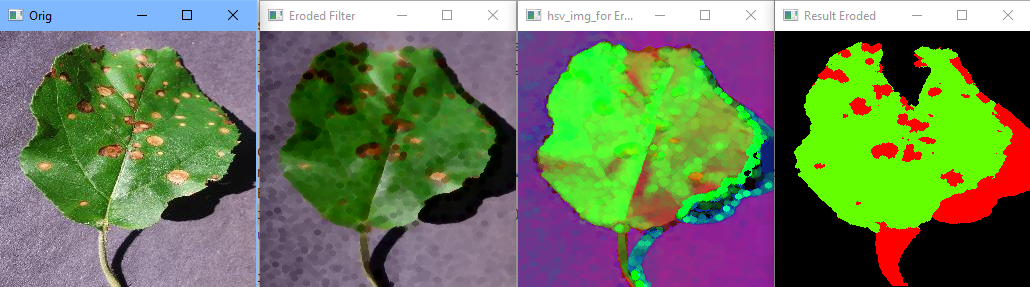
Мы поработали с библиотекой OpenCV. Научились применять матричные фильтры в связке с watershed к изображению для выделения целых и поврежденных частей листа. В ходе работы мы попробовали разные размеры матриц для фильтров (3\*3, 5\*5, 7\*7, 9\*9), наилучший результат дала матрица размерностью 7\*7 с фильтром Гаусса. Остальные фильтры размытия, такие как медианный фильтр и эрозии показали недостаточно хороший конечный результат. Так же попробовали применить watershed без использования фильтра размытия, результат получился некорректным, пол листа не было отображено.

# Анализ полученных результатов

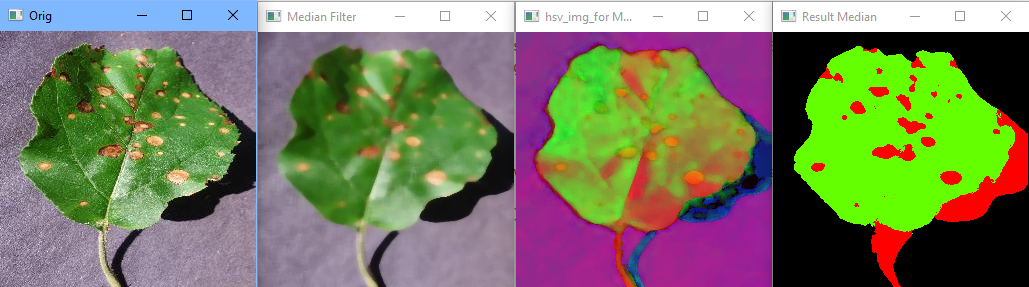
Для эксперимента мы взяли исходное изображение для анализа без использования фильтров размытия. Как можно заметить, больше половины листа не отобразилось.

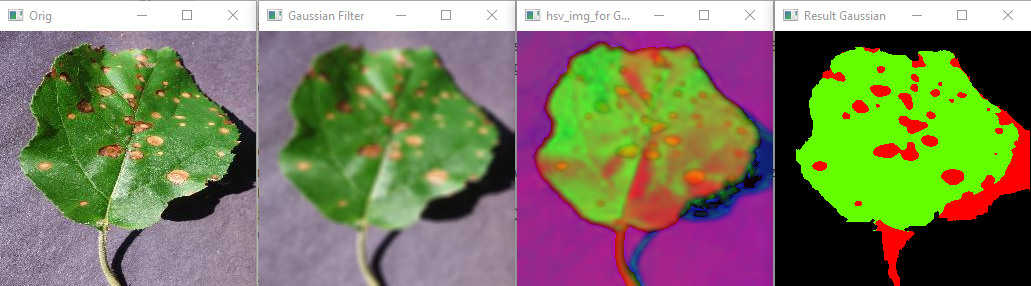


В данном примере мы применяли Erosion Filter, размер ядра – 7\*7. Большинство повреждённых участков отмечены правильно, но верхушка листа “потерялась”, что является не нормой, так же в правой части лица имеются незначительные повреждения, но они не отмечены. Данный алгоритм показал посредственный результат эффективности.



Пример с медианным фильтром показал неплохой результат, на основе предыдущих примеров, но острые края были сглажены и точность поврежденных участков не до конца точна. Для работы фильтра так же использовалась матрица 7\*7.



Последний фильтр показал наилучший результат – это фильтра Гаусса, для работы использовалась матрица 7\*7. Повреждённые и целые зоны отмечены максимально точно. 

# Библиотеки

import sys – Для закрытия приложения в случае если не удалось открыть изображение  
import cv2 as cv – OpenCV для работы с изображениями  
import numpy as np - Для математических вычислений

# Код программы

def CalcOfDamageAndNonDamage(image, name):  
  
 hsv\_img = cv.cvtColor(image, cv.COLOR\_BGR2HSV)  
  
 markers = np.zeros((image.shape[0], image.shape[1]), dtype=**"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, (36, 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, 100)  
 mask[ill\_part > 1] = (0, 0, 255)  
  
 cv.imshow(**"hsv\_img\_for {}"**.format(name), hsv\_img)  
  
 return mask  
  
  
def GaussianFilter(image, sizeMatrix):  
  
 resImage = cv.GaussianBlur(image, (sizeMatrix, sizeMatrix), cv.BORDER\_DEFAULT)  
 cv.imshow(**"Gaussian Filter"**, resImage)  
  
 return resImage  
  
  
def ErodedFilter(image, sizeMatrix):  
  
 kernel = cv.getStructuringElement(cv.MORPH\_ELLIPSE, (sizeMatrix, sizeMatrix))  
 resErode = cv.erode(image, kernel)  
 cv.imshow(**"Eroded Filter"**, resErode)  
  
 return resErode  
  
  
def MedianFilter(image, sizeMatrix):  
  
 imgBlur = cv.medianBlur(image, sizeMatrix, sizeMatrix)  
 cv.imshow(**"Median Filter"**, imgBlur)  
  
 return imgBlur  
  
  
def main():  
  
 *# Constant* NAME\_KEY\_ORIGINAL = **"Original"** NAME\_KEY\_ERODED = **"Eroded"** NAME\_KEY\_GAUSSIAN = **"Gaussian"** NAME\_KEY\_MEDIAN = **"Median"** *# Size matrix for filter* sizeMatrix = 7  
  
 img = cv.imread(**'C:**\\**Users**\\**DIMA**\\**Downloads**\\**data**\\**3.jpg'**)  
 if img is None:  
 sys.exit(**"Could not read the image."**)  
  
 (b, g, r) = img[0, 0]  
 print(**"Red: {}, Green: {}, Blue: {}"**.format(r, g, b))  
 cv.imshow(**"Orig"**, img)  
  
 cv.imshow(**"1"** + NAME\_KEY\_ORIGINAL, CalcOfDamageAndNonDamage(img, NAME\_KEY\_ORIGINAL))  
  
 gaussianFilter = GaussianFilter(img, sizeMatrix)  
 resImg = CalcOfDamageAndNonDamage(gaussianFilter, NAME\_KEY\_GAUSSIAN)  
 cv.imshow(**"Result {}"**.format(NAME\_KEY\_GAUSSIAN), resImg)  
  
 erodedFilter = ErodedFilter(img, sizeMatrix)  
 resImg = CalcOfDamageAndNonDamage(erodedFilter, NAME\_KEY\_ERODED)  
 cv.imshow(**"Result {}"**.format(NAME\_KEY\_ERODED), resImg)  
  
 medianFilter = MedianFilter(img, sizeMatrix)  
 resImg = CalcOfDamageAndNonDamage(medianFilter, NAME\_KEY\_MEDIAN)  
 cv.imshow(**"Result {}"**.format(NAME\_KEY\_MEDIAN), resImg)  
  
 k = cv.waitKey(0)  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

# Вывод

На основе полученных результатов можно сделать вывод. Размытие по Гауссу является наилучшем решением для подобного рода задач. Конкретно к нашей задаче идеально подошла матрица 7\*7. Единственная проблема с которой мы не смогли справиться, выделение тени и палочки листа, как повреждённые участки.