Министерство образования и науки Российской Федерации

Федеральное государственное автономное

Образовательное учреждение высшего образования

«Национальный исследовательский Нижегородский   
государственный университет им. Н.И. Лобачевского»

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**Выделение поврежденных и здоровых зон листьев**

**Выполнил:**

Утюгов Денис  
Студент группы 381906-2

**Преподаватель:**

**Александра Александровна Гетманская**

Нижний Новгород

2021

**Содержание:**

1. Введение
2. Постановка учебно-практической задачи
3. Примененные методы
4. Список литературы
5. Приложение

**Введение**

Мы живем в эпоху данных. Огромное количество организаций занимается изучением изображений. Задача выделения объектов встречается очень часто и применяется во многих сферах.

**Постановка учебно-практической задачи**

В рамках лабораторной работы ставится задачи шумоподавления и сегментирования здоровых и поврежденных участков листьев с использованием библиотеки OpenCV.

**Примененные методы**

Все примеры будут по этому изображению (Рис. 1):



Рисунок 1. Оригинал изображения.

Самой главной проблемой была тень (Рис. 2) - она всегда воспринималась как поврежденный участок. Параметры HSV для выделения самого листика были подобраны изначально вполне успешно (H: [36, 86] S: [25, 255]). Менять я их не стал. Решая проблему с тенью, я решил добавить в маску еще одно условие:   
shadow = cv2.inRange(hsv\_img, (0, 0, 0, 0), (180, 255, 30, 0))  
mask[shadow > 1] = (0,0,0)

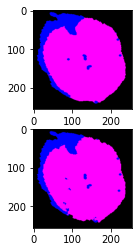
  

Рисунок 2. Три стадии: с тенью, без тени с большой контрастностью (V=30) и без тени с относительно средней контрастностью (V=15). Снизу оригинальные изображения, сверху обработанные шумоподавлением.

Возникла другая проблема - вместе с тенью пропадали куски листа, поэтому пришлось понизить V с 30 до 15. Так удалось убрать тень, но ее контур (в силу контраста) оставался виден. Тем не менее наибольшая составляющая часть проблемы была решена (“поврежденные” куски на фоне отчетливо дают понять, что это вообще не листик), а применение шумоподавления уменьшало или убирало контур тени.

На Рис. 3 мы можем увидеть, где именно программа выделяет тени.



Рисунок 3. Наглядное выделение тени.

Также я пытался выделить эту зону маркером, пометить маркером смог только сам контур (можно увидеть в закомментированной части функции).

Второй по величине проблемой стало шумоподавление. Во многих местах из-за него пропадали пораженные участки (это видно на Рис. 2).

Билатеральный фильтр показывал себя лучше в детектировании пораженных зон, но проигрывал в выделении размера зоны (он их занижал) в сравнении с Non-Local Mean

В самом Non-Local Mean помогло уменьшение параметра h. Я уменьшил его до 7, но возникла еще одна проблема: он сегментировал блики света как пораженную часть.

Медианный фильтр работал относительно хуже предыдущих. Использовал я его с ядром размера 3, иные значения не имели смысла.

В итоге я пришел к выводу, что в большинстве случаев шумоподавление больше мешает, чем помогает.

**Заключение**

В ходе выполнения лабораторной работы поставленная задача была выполнена. пораженные и здоровые участки листьев были сегментированы. Было исследовано несколько шумоподавлений, а также их влияние на сегментацию в данной задаче.

**Литература**

1. Приложенная к заданию методичка
2. Раздел документации [“Image Filtering”](https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#void%20GaussianBlur(InputArray%20src,%20OutputArray%20dst,%20Size%20ksize,%20double%20sigmaX,%20double%20sigmaY,%20int%20borderType)) OpenCV

**Приложение**

import cv2

import sys

import numpy as np

import matplotlib.pyplot as plt

import os

def CalcOfDamageAndNonDamage (image\_name):

'''Использование методов watershed(), выделение марекров.'''

image = image\_name

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (7,7))

image\_erode = cv2.erode(image, kernel)

hsv\_img = cv2.cvtColor(image\_erode, cv2.COLOR\_BGR2HSV)

#gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

#mask = cv2.compare(gray, 5, cv2.CMP\_LT)

#img[mask > 0] = 255

#Создаем массив из нулей - маркеры, после выделяем зоны (фон - 1 и листик - 255).

markers = np.zeros((image.shape[0], image.shape[1]), dtype = "int32")

markers[40:170, 90:140] = 255 #Решили, что лист всегда будет в центре

markers[236:255, 0:20] = 1

markers[0:20, 0:20] = 1

markers[0:20, 236:255] = 1

markers[236:255, 235:255] = 1

#markers[mask > 0] = 2 #Попытка отделить тень в отдельный класс

#Применяем watershed()

leafs\_area\_BGR = cv2.watershed(image\_erode, markers)

healthy\_part = cv2.inRange(hsv\_img, (36, 25, 25), (86, 255, 255)) #Параметры hsv вроде бы выставлены корректно и менять их смыла нет

ill\_part = leafs\_area\_BGR - healthy\_part

shadow = cv2.inRange(hsv\_img, (0, 0, 0, 0), (180, 255, 15, 0))

#Создаем маску

mask = np.zeros\_like(image, np.uint8 )

mask[leafs\_area\_BGR > 1] = (255,0,255)

mask[ill\_part>1] = (0,0,255)

mask[shadow > 1] = (0,0,0)

return mask

img = cv2.imread('8.jpg')

def NlMeans(img):

'''Шумоподавляющий фильтр'''

b,g,r = cv2.split(img)

rgb\_img = cv2.merge([r,g,b])

dst = cv2.fastNlMeansDenoisingColored(img, None, 7, 10, 7, 21) #img, dst, h, tWS, sWS. Есть смысл рассматривать h

b,g,r = cv2.split(dst)

rgb\_dst = cv2.merge([r,g,b])

return rgb\_dst

rgb\_dst = NlMeans(img)

plt.subplot(211), plt.imshow(img)

plt.subplot(212), plt.imshow(rgb\_dst)

plt.show()

a = CalcOfDamageAndNonDamage(rgb\_dst)

b = CalcOfDamageAndNonDamage(img)

plt.subplot(211), plt.imshow(a)

plt.subplot(212), plt.imshow(b)

plt.show()

# Read the image .

img = cv2.imread('8.jpg')

# diameter = 15, sigmaColor = sigmaSpace = 75.

bilateral = cv2.bilateralFilter(img, 3, 75, 75)

plt.imshow(bilateral)

plt.subplot(211), plt.imshow(img)

plt.subplot(212), plt.imshow(bilateral)

plt.show()

a = CalcOfDamageAndNonDamage(rgb\_dst)

b = CalcOfDamageAndNonDamage(bilateral)

plt.subplot(211), plt.imshow(a)

plt.subplot(212), plt.imshow(b)

plt.show()

medBl = cv2.medianBlur(img, 3)

plt.imshow(medBl)

a = CalcOfDamageAndNonDamage(img)

b = CalcOfDamageAndNonDamage(medBl)

plt.subplot(211), plt.imshow(a)

plt.subplot(212), plt.imshow(b)

plt.show()