МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное   
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики  
Кафедра технической кибернетики

**Отчет по лабораторной работе №6**

Дисциплина: «Системы обработки изображений»

Тема: **«Нелинейная фильтрация изображений»**

Выполнил: Пшенин В.И.

Группа: 6133-010402D

Самара 2020

**Задание на лабораторную работу**

1. Осуществить взвешенную медианную фильтрацию входного изображения, совпадающего с одним из тестовых изображений, соответствующим окном. Виды окон по вариантам приведены ниже.
2. Осуществить взвешенную ранговую фильтрацию входного изображения соответствующим окном со значением ранга 1, *N*, *r*, где *N* = *S*∙*S*, где *S* – размер маски, *r* – экспериментально выбранный ранг. Виды окон по вариантам приведены ниже.
3. Вычислить ошибку и сравнить результат с результатом лабораторной работы 5 соответствующего входного изображения.

|  |  |
| --- | --- |
| **№ варианта** | **Вид окна** |
| **2** |  |

**Задание 1 - Осуществить взвешенную медианную фильтрацию входного изображения соответствующим окном.**



Рисунок 1 – Входное изображение

Осуществим взвешенную медианную фильтрацию входного изображения окном . Результат представлен в таблице 1.

Таблица 1 – Результат применения медианной фильтрации

|  |  |
| --- | --- |
| Искаженное изображение (шум p=0.14) | Отфильтрованное изображение |
| C:\Users\otori\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Без названия.png |  |

**Задание 2 - Осуществить взвешенную ранговую фильтрацию входного изображения соответствующим окном со значением ранга 1, N, r, где N = S∙S, где S – размер маски, r – экспериментально выбранный ранг**

Таблица 2 – Результат применения взвешенной ранговой фильтрации

|  |  |
| --- | --- |
| Ранг | Отфильтрованное изображение |
| 1 | C:\Users\otori\AppData\Local\Microsoft\Windows\INetCache\Content.Word\first.png |
| 6 (экспер.) | **C:\Users\otori\AppData\Local\Microsoft\Windows\INetCache\Content.Word\мой ранг.png** |
| 9 | **C:\Users\otori\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ранг 9.png** |

**Задание 3 - Вычислить ошибку и сравнить результат с результатом лабораторной работы 5 соответствующего входного изображения**

Таблица 3 – Ошибка восстановление сигнала

|  |  |  |
| --- | --- | --- |
| Метод фильтрации | Ошибка до фильтрации | Ошибка после фильтрации |
| Медианная | 0.212 | 0.038 |
| Взвешенная, ранг 6 | 0.037 |
| Взвешенная, ранг 1 | 0.333 |
| Взвешенная, ранг 9 | 0.076 |
| КИХ | 0.093 |

Самым простым, эффективным и быстрым методом оказалась медианная фильтрация. Здесь КИХ-фильтрация явно уступает остальным методам, возможно при ином окне КИХ-фильтра результаты будут лучше.

**ПРИЛОЖЕНИЕ А**

**Код программы**

import cv2

import random

from PIL import Image

import numpy as np

from IPython.display import display

# In[2]:

def show(img):

img2 = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

return Image.fromarray(img2)

# In[40]:

def myFilter(img,mask,r=None):

newImage = np.copy(img)

ijStart = mask.shape[0] // 2

img = cv2.copyMakeBorder(img, ijStart, ijStart, ijStart, ijStart, cv2.BORDER\_REPLICATE)

mask = mask.flatten()

for i in range(0,newImage.shape[0]):

for j in range(0,newImage.shape[1]):

frame = img[i:i+ijStart\*2+1,j:j+ijStart\*2+1]

frame = frame.flatten()

mediana = []

for p in range(len(mask)):

for k in range(mask[p]):

mediana.append(frame[p])

mediana.sort()

if r==None:

newImage[i,j] = mediana[len(mediana) // 2]

else:

newImage[i,j] = mediana[r]

return newImage

# In[4]:

def myNoisePer(img,per):

newImage = np.copy(img)

countOfPixels = int(img.shape[0]\*img.shape[1]/100\*per)

for c in range(countOfPixels):

i = random.randint(0,img.shape[0]-1)

j = random.randint(0,img.shape[1]-1)

newImage[i,j] = [random.choice([0,255]),random.choice([0,255]),random.choice([0,255])]

return newImage

# In[5]:

def getError(img1, img2):

img1 = img1/255

img2 = img2/255

error = (np.sum((np.array(img1.flatten()) - np.array(img2.flatten())) \*\* 2) / (img1.shape[0] \* img2.shape[1] \* 3)) \*\* 0.5

return error

# In[41]:

def doLab(img):

kernel = np.array([

[2, 1, 2],

[1, 1, 1],

[2, 1, 2]])

imgNoisy = myNoisePer(img, 14)

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

bNoise, gNoise, rNoise = cv2.split(imgNoisy)

rOut = myFilter(rNoise, kernel,9)

gOut = myFilter(gNoise, kernel,9)

bOut = myFilter(bNoise, kernel,9)

newImg = cv2.merge((bOut, gOut, rOut))

display(show(img),show(imgNoisy),show(newImg))

print("Error (noise image): ", getError(img, imgNoisy))

print("Error (after filter): ", getError(img, newImg))