Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий  
Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Отчет

По лабораторной работе №6

«Фильтрация фотоизображения.   
Подавление шумов на фотографиях»

по дисциплине «Компьютерная графика»

Студент группы ПИ-02 Чередов Р.А.

Преподаватель Андреева А.Ю.

Барнаул 2023

**Задание:**

Для произвольного фотоизображения реализовать:

* искусственное наложение шума (точки, линии, окружности);
* сравнение фильтров шумоподавления со сменой аппертуры (по вариантам): равномерный и фильтр гаусса
* сравнение методов повышения резкости;

Применение спецэффектов (по вариантам): акварелизация

Равномерный фильтр:

Суть:

Каждый пиксель поглощает, что-то из цвета, соседнего пикселя

До



После



Фильтр Гаусса:

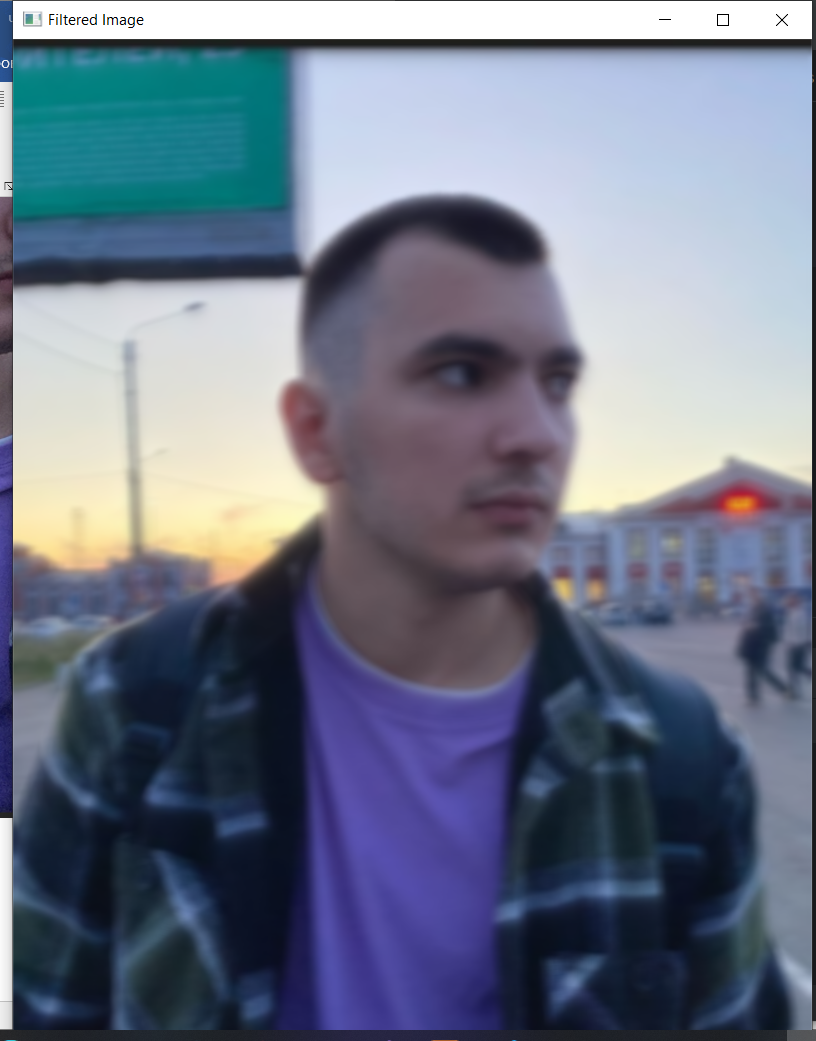
Суть:

Влияние пикселей на обрабатываемый, уменьшается с расстоянием

Было



Стало

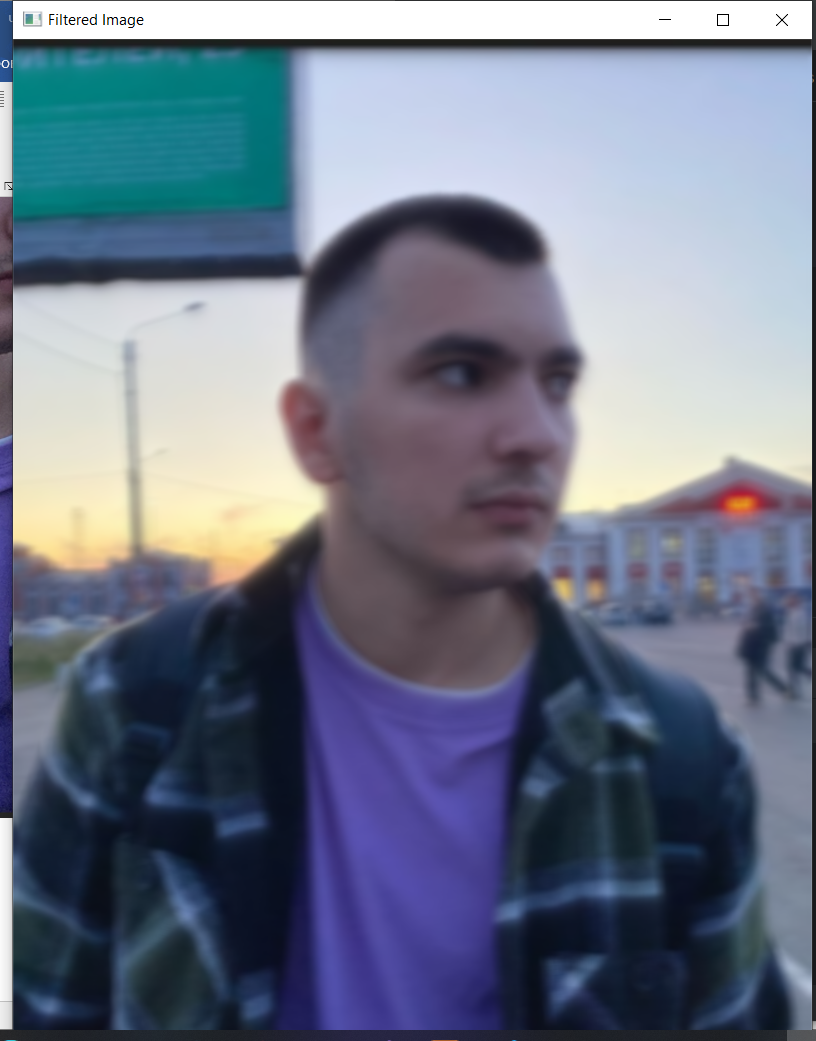


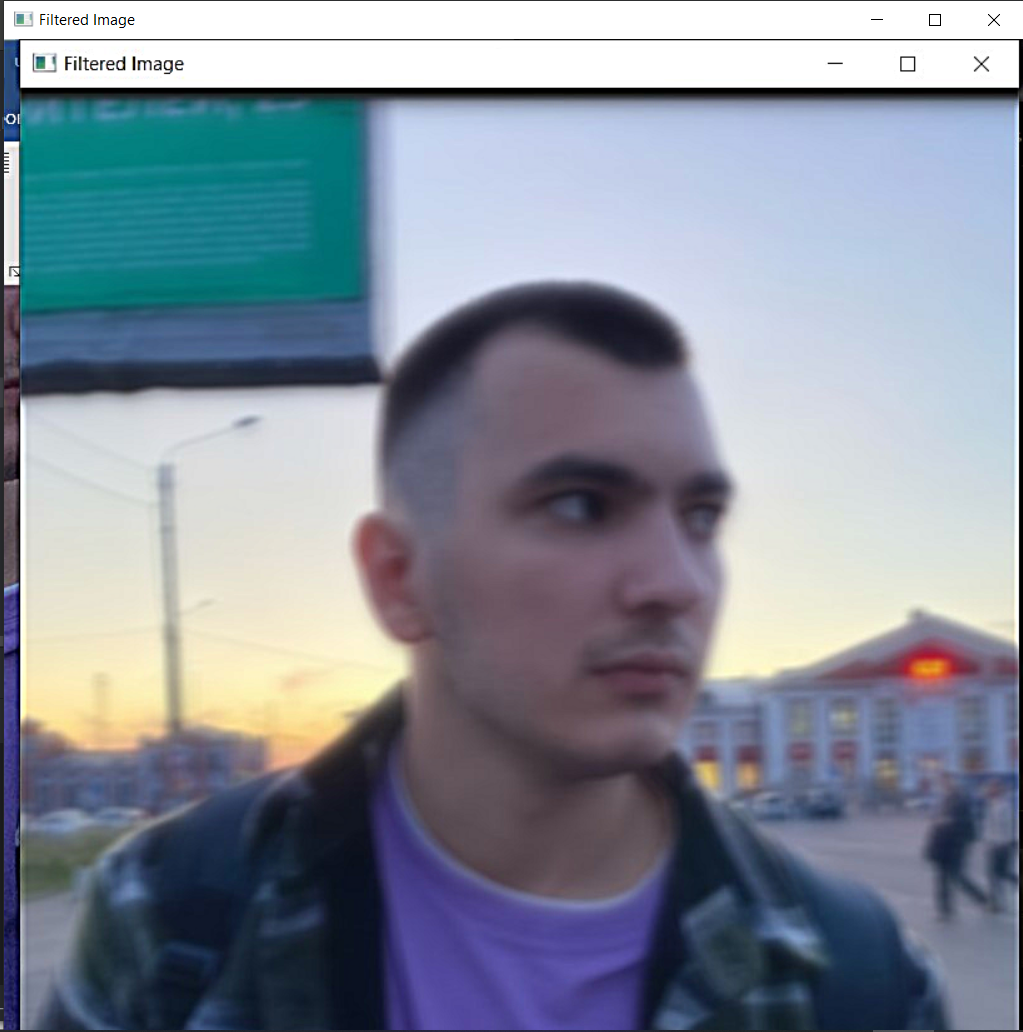
Методы повышения резкости:

Суть:

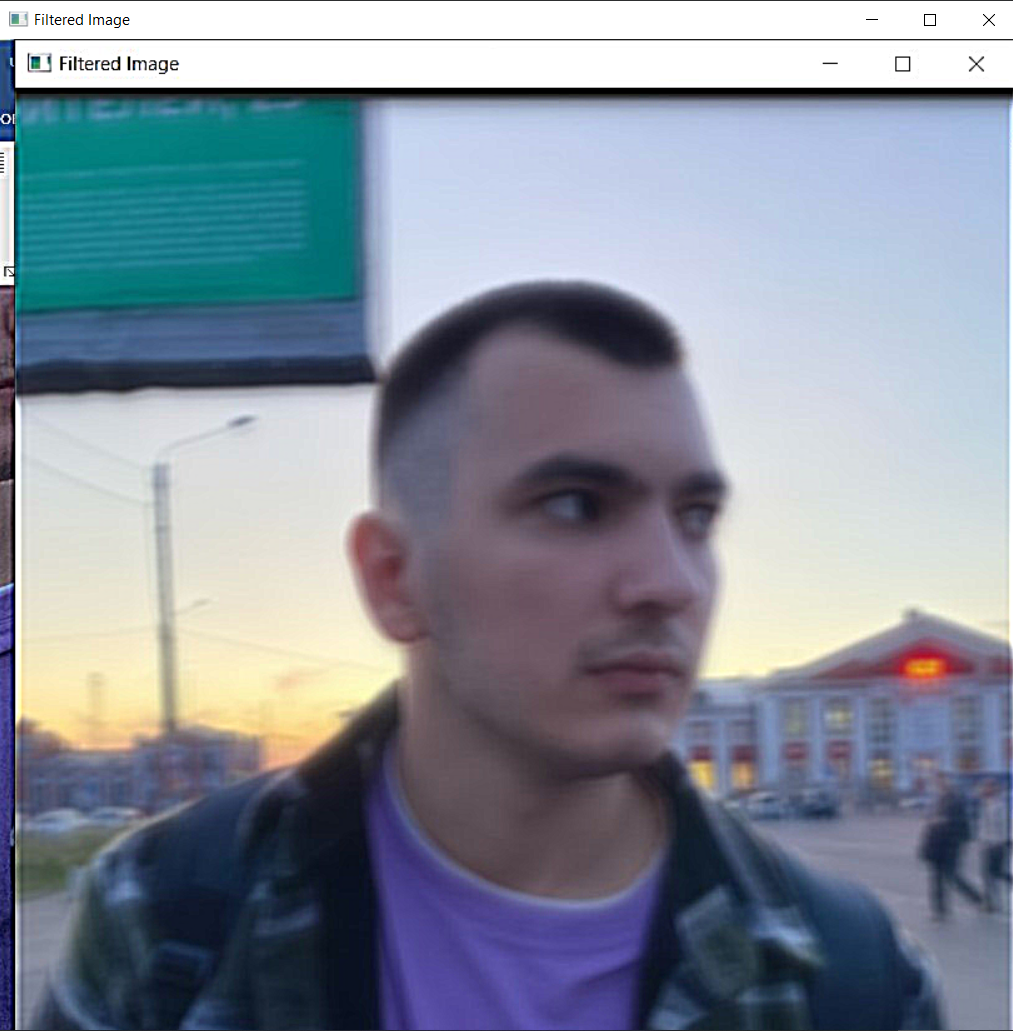
Увеличивается любой существующий контраст между цветом пикселя и цветами его соседей.

До



После  


И еще раз



Акварелизация:

Происходит сглаживание цветов и после этого обрабатывается резкостью.



Код:

Main.py

from Noise import Noise;  
from Core\_Uniform import CoreUniform  
from Core\_Gaus import CoreGaus  
from Core\_Sharpness import CoreSharpness  
from Filter import Filter  
  
import cv2  
import numpy as np  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
   
 image\_path = 'Image/roma.jpg'  
 image = cv2.imread(image\_path)  
 scale\_image = np.copy(image);  
 fil = Filter(scale\_image)  
   
 while True:  
 print(" Выбирите один из вариантов")  
 print(" 1: Фильтр шумоподавления")  
 print(" 2: Фильтр повышения резкости")  
 print(" 3: Акварелизация")  
 print(" 4: Наложить шум")  
 print(" 5: Сбросить изображение")  
 print(" 0: Выход")  
   
 FLAG = int(input(":: "))  
 # ФИЛЬТР ШУМОПОДАВЛЕНИЯ  
 if FLAG == 1:  
 print("\n [Фильтр шумоподавления]")  
 print(" Выбирите один из вариантов")   
 print(" 1: Равномерный фильтр")  
 print(" 2: Гаусовский фильтр")  
 print(" 3: Медианная фильтрация")  
 print(" 0: Веонуться")  
   
 FLAG = int(input(" :: "))  
 if FLAG == 1:  
 print("\n [Равномерный фильтр]")  
 seg = int(input(" Размер сегмента: "))  
 coreu = CoreUniform(seg);  
 scale\_image = fil.applyAperture(coreu)  
 elif FLAG == 2:  
 print("\n [Гаусовский фильтр]")  
 seg = int(input(" Размер сегмента: "))  
 O = float(input(" Среднеквадратичное отклонение: "))  
 coreg = CoreGaus(seg, O);  
 scale\_image = fil.applyAperture(coreg)  
 elif FLAG == 3:  
 print("\n [Медианная фильтрация]")  
 seg = int(input(" Размер сегмента: "))  
 scale\_image = fil.medianFilter(seg)  
  
 # ФИЛЬТР ПОВЫШЕНИЯ РЕЗКОСТИ  
 elif FLAG == 2:  
 print("\n [Фильтр повышения резкости]")  
 seg = int(input(" Размер сегмента: "))  
 K = float(input(" Степень резкости: "))  
 coresh = CoreSharpness(seg, K);  
 scale\_image = fil.applyAperture(coresh)  
   
 # АКВАРЕЛИЗАЦИЯ   
 elif FLAG == 3:  
 print("\n [Акварелизация]")  
 seg = int(input(" Размер сегмента: "))  
 coreu = CoreUniform(seg);  
 scale\_image = fil.medianFilter(seg);   
 coresh = CoreSharpness(seg, 2);  
 scale\_image = fil.applyAperture(coresh)  
   
 elif FLAG == 4:  
 print("\n [Наложить шум]")  
 per = float(input(" Частота шума: "))  
 scale\_image = Noise.salt\_and\_papper(scale\_image, per)  
   
 elif FLAG == 5:  
 print("\n [Сбросить изображение]")  
 scale\_image = np.copy(image)  
 fil = Filter(scale\_image)  
   
 elif FLAG == 0:  
 print("Завершение работы")  
 break;  
   
 print(" ")  
 print(" ==============\n")  
  
 cv2.imshow('Filtered Image', scale\_image)   
 cv2.waitKey(0)  
 cv2.destroyAllWindows()

Noise.py

import numpy as np  
  
class Noise:  
   
 @staticmethod  
 def salt\_and\_papper(image, rand):  
 row, col, ch = image.shape  
 salt\_and\_pepper = np.random.rand(row, col)  
 image[salt\_and\_pepper < rand] = 0  
 image[salt\_and\_pepper > 1 - rand] = 255  
 return image

Gaus.py

from Core import Core  
import numpy as np  
import math   
  
class CoreGaus(Core):  
 def \_\_init\_\_(self, size, O):  
 self.O = O;  
 super().\_\_init\_\_(size);  
   
   
   
 def createCore(self, size):  
 kernel = np.zeros((size, size, 3))  
 center = size//2  
   
 for i in range(size):  
 for j in range(size):  
 x = i - center  
 y = j - center  
 kernel[i, j] = np.exp(-(x\*\*2 + y\*\*2) / (2 \* self.O\*\*2))  
  
 kernel /= np.sum(kernel, axis=(0,1));  
   
 return kernel

Sharpness.py

from Core import Core  
import numpy as np  
import math   
  
class CoreSharpness(Core):  
 def \_\_init\_\_(self, size, K):  
 self.K = K;  
 super().\_\_init\_\_(size);  
   
 def createCore(self, size):  
 divider = size\*\*2 - 1  
 kernel = np.full((size, size, 3), -(self.K / divider))  
 kernel[size//2, size//2] = np.full((3), self.K + 1)   
 return kernel