«Методы и реализация фильтрации цифровых изображений»

Работа выполнена Новиковой Дианой,

студенткой ИВТ 42-БО

Описание работы.

**Цель работы**: Получить практические навыки программной реализации алгоритмов цифровой фильтрации изображений

Описание работы с программой.

1. В меню «Файл» выберите вариант «Открыть», затем выберите желаемое изображение. Оно отобразится в программе.
2. Далее выберите фильтр из списка. Он отобразится в таблице. При выборе опции «Свой фильтр» введите необходимые значения в таблицу.
3. Нажмите «Применить». Итоговое изображение будет выведено на экран.
4. Для сохранения изображения в меню «Файл» выберите «Сохранить» и укажите путь для сохранения файла.

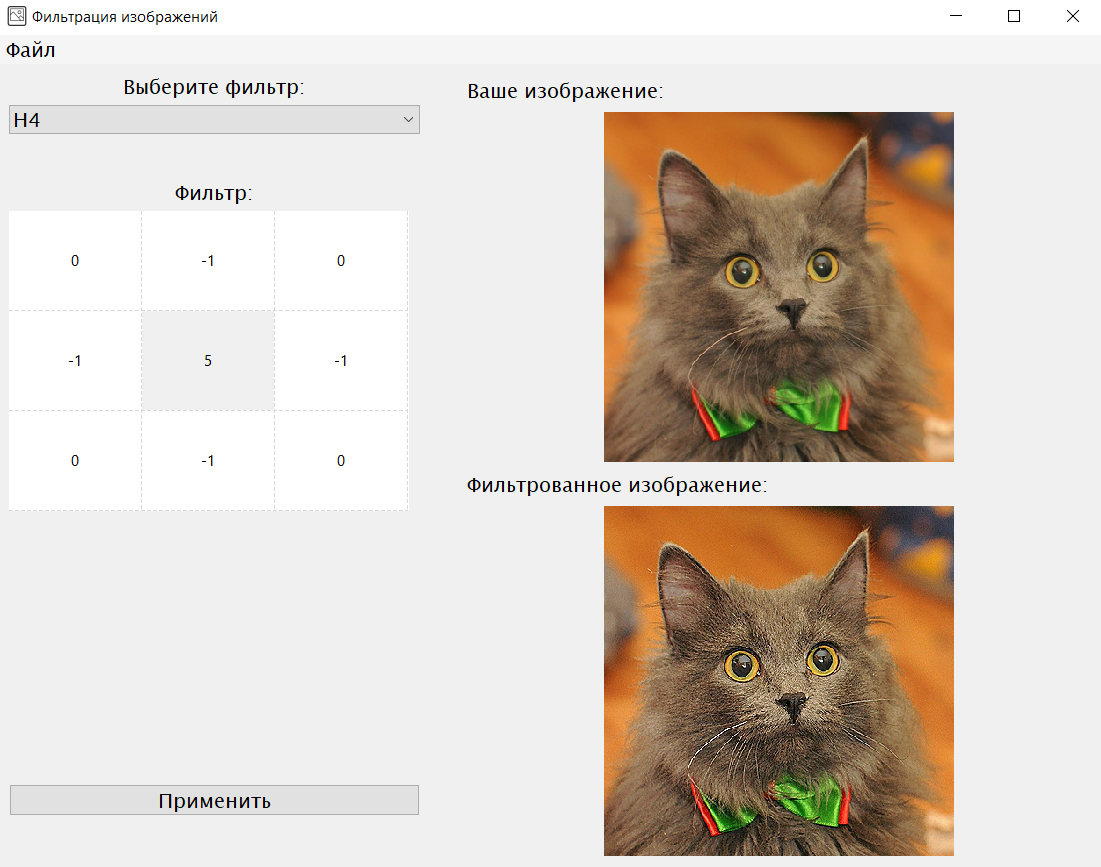


Рисунок 1. Скриншот работы программы

Описание работы алгоритма программы.

1. При открытии файла картинка переводится в массив пикселей и сохраняется в памяти программы.
2. После нажатия «Применить» массив разделяется на каналы. Для каждого канала производятся следующие операции:
   1. Формируется новый массив по следующему принципу: для каждого пикселя берется его окрестность, необходимая для фильтрации, и превращается в строку нового массива.
   2. Далее для каждой строки полученного массива выполняется операция поэлементного умножения на соответствующий элемент фильтра. Все элементы суммируются (в случае медианного фильтра за ядро берется массив единиц, суммирование заменяется на поиск медианы).
   3. Все каналы соединяются обратно, полученный результат выводится на экран с возможностью сохранения на компьютер.

Анализ полученных результатов.

Сравним работу программы на данном изображении. Картинка была уменьшена до размера 400 пикселей для упрощения работы.



Рисунок 2. Оригинал

Фильтры нижних частот увеличивают размытость изображения. Можно заметить, что фильтр H1 создает равномерное размытие, а фильтр Н3 действует так, что более дальние пиксели меньше влияют на результирующий, что помогает сохранить более четкие линии.



Рисунок 3.Применение НЧ фильтров. Слева – H1, справа – H3.

Фильтры верхних частот увеличивают резкость изображения. Причем, как видно на картинках, чем больше пикселей из определенной окрестности участвуют в фильтрации, тем резче получается результат. Так фильтр Н4 использует только ближайшие пиксели, а Н6 в свою очередь использует все.



Рисунок 4. Применение ВЧ фильтров. Слева – H4, справа – H6

Медианный фильтр широко используется для удаления шумов на изображениях. На исходное изображение был искусственно добавлен шум. Данный фильтр с размером 3х3 прекрасно справился со сглаживанием практически всех зашумленных областей. Фильтр с большими измерениями сгладил бы картинку еще больше, но при этом потерялась бы резкость краев.



Рисунок 5. Оригинал с искусственным шумом, справа - после медианного фильтра

Таким образом, цель работы выполнена – программа выполняет фильтрацию изображений с заданными заранее, а также введенными пользователем, ядрами и предоставляет функции для сохранения результата.

Приложение 1. Используемые фильтры

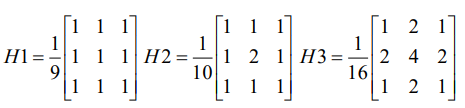


Рисунок 6. Фильтры нижних частот

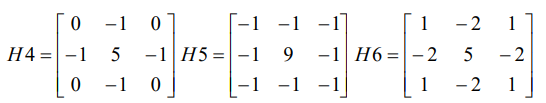


Рисунок 7. Фильтры верхних частот

Приложение 2. Результат работы программы на других изображениях.



Рисунок 8. Слева - оригинал ч-б, справа - фильтр H1.



Рисунок 9. Слева - оригинал в цвете, справа - фильтр Н2.



Рисунок 10. Слева - фильтр Н5, справа - фильтр Н6.



Рисунок 11. Слева - искусственный шум, справа - после медианного фильтра.

Приложение 3. Код основных частей программы

Программа состоит из 2 основных классов и двух дополнительных. Класс App отвечает за интерфейс программы, класс Filter за работу фильтрации изображения. Класс TableModel отвечает за вывод параметров фильтра на экран, AlignDelegate - класс для стилевого оформления. В качестве библиотеки для графического отображения использовалась PyQT5.

Логика фильтрации.

Картинка разбивается на отдельные каналы:

|  |
| --- |
| cv2.split(self.image) |

Для каждого канала выполняется функция *filter\_channel.*

|  |
| --- |
| **def** filter\_channel(self, channel, kernel, function=np.sum):  img = self.im2col(channel, kernel.shape[:2])  output = np.zeros(img.shape[:2])  kernel\_flat = kernel.flatten()  **for** x, y **in** itertools.product(range(img.shape[0]),  range(img.shape[1])):  output[x, y] = function(img[x, y] \* kernel\_flat)  **return** output |

Она вызывает описанный выше алгоритм im2col с помощью функции библиотеки numpy.

|  |
| --- |
| **def** im2col(img, kernel\_size):  **return** np.lib.stride\_tricks.sliding\_window\_view(img, kernel\_size).reshape(  img.shape[0] - int(kernel\_size[0] / 2) \* 2, img.shape[1]  - int(kernel\_size[1] / 2) \* 2, -1) |

Далее происходит поэлементное умножение и суммирование для каждого пикселя картинки. Результат возвращается и добавляется в массив каналов, который и является выходной картинкой.

Основные функции.

Открытие и сохранение картинок происходит с помощью файлового диалога из используемой библиотеки.

|  |
| --- |
| **def** open\_file(self):  file\_dialog = QFileDialog()  file\_dialog.setAcceptMode(QFileDialog.AcceptOpen)  file\_dialog.setFileMode(QFileDialog.ExistingFile)  filename = file\_dialog.getOpenFileName(self, 'Открыть', '', "Images (\*.png \*.bmp \*.pcx \*.jpg)") |

Для исправности работы программы используются подсказки пользователя и обработка ошибок в случае неверного ввода.

|  |
| --- |
| **if** size > len(self.data) **or** size > len(self.data[0]):  self.show\_message("Фильтр слишком большой. Выберите размер меньше, чем измерения картинки") |

Фильтры в памяти хранятся в отдельном файле в виде массивов numpy.

|  |
| --- |
| h1 = (np.array([  [1, 1, 1],  [1, 1, 1],  [1, 1, 1]  ]) / 9) |