Отчет по лабораторной работе 4

Студента группы №3331506/00401 Орехова Алексея Михайловича

Код работы выложен на [гитхаб](https://github.com/alexeyorehoff/cv-lessons)

Первым в рамках работы был создан алгоритм вычисления преобразования Фурье методом «в лоб». Для него использовалась классическая формула 2D преобразования Фурье, предполагающая, что для каждого пикселя исходного изображения будет выполнена операция вычисления коэффициентов Фурье, которая, в свою очередь, еще раз проходится по всем пикселям исходного изображения. Таким образом, временная сложность алгоритма получается O(n4). Для отображения результатов работы алгоритмов была создана функция display\_magnitude, выводящая амплитуду комплексного значения функции Фурье в каждом пикселе. Функции прямого, обратного преобразования Фурье «в лоб», а также созданные вспомогательные функции представлены на рисунке 1.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

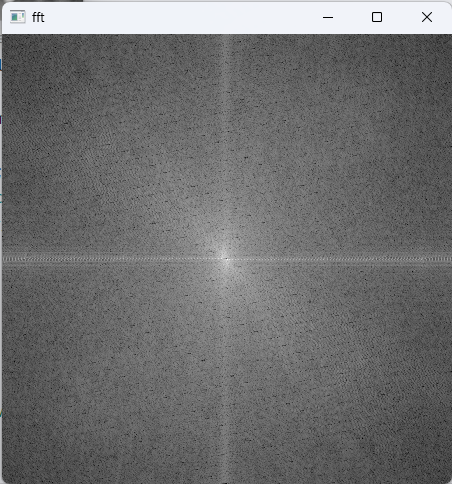
# Рисунок 1

Время работы алгоритма составило 35.4 секунды для размера исходного изображения 128 на 128, а поскольку алгоритм обладает алгоритмической сложностью O(n4), то можно оценить время обработка изображения в исходном размере 440 на 440 почти в 1 час, что не годится для большинства сценариев применения. Результаты работы алгоритма представлены на рисунке 2.

# Изображение выглядит как текст, одежда, шляпа, Человеческое лицо Автоматически созданное описаниеИзображение выглядит как снимок экрана, серый Автоматически созданное описаниеИзображение выглядит как одежда, текст, головной убор, шляпа Автоматически созданное описание

# Рисунок 2 – Исходное изображение, преобразование и восстановленное изображение

С целью увеличения скорости вычисления был использован алгоритм быстрого преобразования Фурье radix-2. С его помощью удалось достигнуть кратного ускорения работы, с временем преобразования 39мс для изображения 128 на 128, что соответствует приросту скорости более чем в 900 раз. Код и результат работы алгоритма fft представлены на рисунке 3.   
Также, была протестирована встроенная функция cv::dft. время выполнения преобразования встроенным методом opencv составило 5.76мс, что делает его оптимальным среди рассмотренных методов. Результат работы встроенного алгоритма также представлен на рисунке 3. Примененный алгоритм считается эквивалентным двумерному ДПФ

Изображение выглядит как снимок экрана, компьютер, текст

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

# Рисунок 3

Далее, основываясь на встроенной функции opencv была создана функция свертки с произвольным ядром. Результат ее работы возвращается в частотной области, выводится на экран, после чего производится обратное преобразование Фурье для получения исходного изображения с примененным фильтром.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

# Рисунок 4

На рисунке 5 показаны промежуточные результаты работы функции свертки и обработанные изображения. На рисунке показаны соответственно фильтр среднего, фильтр Собеля вдоль X и Y и фильтр Лапласа.

Изображение выглядит как искусство, черно-белый, снимок экрана, монохромный

Автоматически созданное описание

# Рисунок 5

Для фильтра низких и высоких частот была создана функция, вырезающая из изображения внешнюю либо внутреннюю часть круга, с центром в центре изображения и с размером, задаваемым как отношение радиуса круга к стороне изображения. Код и результат работы функции представлены на рисунке 6.

Изображение выглядит как текст, снимок экрана, Человеческое лицо, Редактирование

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

# Рисунок 6

Для детекции символов на номере была использована встроенная функция match\_template. Код и результат его работы представлены на рисунке 7. Можно отчетливо видеть черные пятна на месте символов «0» и «А» соответственно

Изображение выглядит как текст, Шрифт, число, лицензия

Автоматически созданное описание Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

# Рисунок 7

**Вывод:**

Получены навыки применения преобразования Фурье для обработки изображений.

**Дополнительное задание:**

Почему вы считаете, что так можно? И что это даст двумерное ДПФ

*std::vector&lt;std::complex&lt;double&gt;&gt; fft\_result;*

*fft\_radix2(image\_vector, fft\_result, false);*

*std::cout <<; “radix: “; << steady\_clock::now() - start << std::endl;*

*cv::imshow(“fft”;, display\_magnitude(vec2mat(fft\_result, image.size())))*

Двумерное преобразование Фурье

То есть, для каждого пикселя результирующей матрицы с изображением по Фурье вычисляется сумма значений функции Фурье по строкам и столбцам.   
В случае алгоритма radix-2, перед его работой требуется преобразовать исходную матрицу изображения в вектор, где данные хранятся строка за строкой. В ходе своей работы radix рекурсивно делит матрицу пополам, выполняя то же действие для подвекторов четных и нечетных элементов до момента, пока от вектора не останется один элемент (условие выхода из рекурсии).   
При работе алгоритма распределение четных и нечетных элементов может комбинироваться таким образом, чтобы формировать на оригинальном изображении столбцы пикселей, после в следующей итерации рекурсии суммирование идет не только по строкам, но и по столбцам. Таким образом удается добиться эквивалентности БПФ по алгоритму radix двумерному ДПФ.

Также, было получено дополнительное задание, суть которого заключалась в детектировании уха на изображении кота (Рисунок 8).

**Изображение выглядит как млекопитающее, кошка европейская, манул, на открытом воздухе

Автоматически созданное описание**

# Рисунок 8

Сложность задания дополнялась тем, что шерстка кота обладает схожей текстурой с объектами окружения, что не позволяло достоверно детектировать участок, соответствующий уху. Для решения этой проблемы были протестированы различные вариации предварительной обработки, включая заблюривания для сглаживания высокочастотных шумов, фильтр высоких частот, нормализация. В результате стабильного выделения уха удалось достичь, применив к изображению фильтр Лапласа. Результаты работы алгоритма представлены на рисунке 9.

Изображение выглядит как снимок экрана, черно-белый, рентгеновская пленка, монохромный

Автоматически созданное описаниеИзображение выглядит как млекопитающее, кошка европейская, кот, мех

Автоматически созданное описание

Рисунок 9