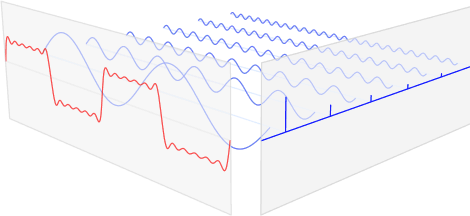
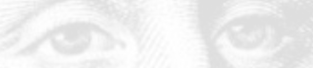
**ЛАБОРАТОРНАЯ РАБОТА №4**

Использование преобразования **Фурье**

в обработке изображений

Мороз Юрий Сергеевич

e-mail: [crazysiberianscientist@gmail.com](mailto:crazysiberianscientist@gmail.com)

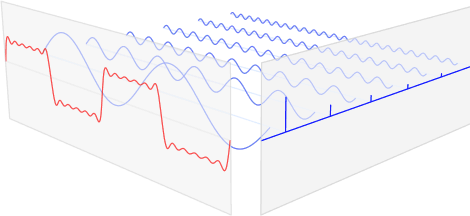
*Ключевые слова*: тут надо как можно больше картинок про Фурье, Фурье, самая мутная тема.

**ТЕОРИЯ**

1. На этот раз пойдёт разговор о преобразовании Фурье, которое уже поколениями выносит мозг студентам на математическом анализе. Оказывается, это вами любимое, преобразование Фурье позволяет ускорять обработку изображений. Применяется в обработке изображений оно в виде Дискретного Преобразования Фурье (ДПФ, не по-нашему: Discrete Fourier Transform (DFT)).

Как вы ~~не~~ помните из лекций, то свёртка двух массивов в пространственной области равнозначна их перемножению в частотной области. За счёт применения поэлементного произведения спектров вместо обычного метода скользящего окна происходит экономия вычислительных мощностей.

Немного вспомним, что такое ряды Фурье. Чтобы вас не пугать, то не буду давать формул (которые уже были в лекции). А скажу вам только, что ряд Фурье представляет собой массив комплексных чисел, каждый элемент которого содержит значение комплексного числа, соответствующего ***i***-ой гармоники, где ***i*** – индекс текущего элемента. Чем больше индекс, тем больше частота гармоники. Вот вам наглядная картиночка:



1. Функция для проведения ДПФ. С ней надо уметь обращаться, тут не всё так просто.

void dft(InputArray src, OutputArray dst, int flags = 0);

InputArray src – входное изображение (или одномерный массив). Если хотим осуществить прямое ДПФ, может быть одноканальным с вещественным числами, пусть оно будет CV\_32FC1. Если обратное - двухканальным с вещественным числами, где в одной хранятся вещественные числа, а в другой мнимые, пусть оно будет CV\_32FC2. Зависит от флага.

OutputArray dst – выходное изображение. Прямое – двухканальное. Обратное – одноканальное.

int flags – флаг режима работы.

DFT\_COMPLEX\_OUTPUT **–** для прямого ДПФ с выводом в виде полной комплексной формы

DFT\_INVERSE|DFT\_REAL\_OUTPUT **– такая совокупность флагов используется для обратного ДПФ**

1. **Ускорение работы функции ДПФ. Из-за особенностей алгоритма ДПФ, производительность ДПФ не имеет линейную зависимость от размера входного массива. Массивы, размер которых представляет собой двойку в степени (2,4,8,16…), обрабатываются быстрее всего. Так же обрабатываются достаточно эффективно массивы, имеющие размер равный произведению 2-ек, 3-ек, 5-ок (например,** 300 = 5\*5\*3\*2\*2)**.**

int getOptimalDFTSize(int vecsize)

int vecsize **– размер массива(вектора) , для которого требуется просчитать оптимальный размер.**

**Функция возвращает значение размера, которое больше исходного, но при этом является оптимальным. Если значение входящего массива слишком велико (близко к INT\_MAX), то возвращает отрицательное число.**

**Как правильно использовать для создания размера холста при перемножении спектров:**

Size dftSize;

dftSize.width = getOptimalDFTSize(img.cols + Gy.cols - 1);

dftSize.height = getOptimalDFTSize(img.rows + Gy.rows - 1);

1. **Функция для перемножения спектров.**

void mulSpectrums(InputArray a, InputArray b, OutputArray c, int flags=0, bool conjB = false);

**Для получения адекватного результата перемножения следует использовать изображения одинаковые по размеру.**

**Для этого, до начала проведения преобразований с ДПФ, у исходных изображений увеличивают холст. Увеличение холста – это когда создают пустое изображение, которое больше по размеру, чем исходное, и копируют туда исходное, совмещая его с холстом по левому верхнему углу. Кто не понял:** <http://photoshop.e-publish.ru/p15aa1.html> **(только там по центру совмещение).**

**Данную операцию проводят для получения одинакового числа гармоник, число которых равняется площади изображения.**

**Если два изображения разные по размеру, то размер холста равен их суммарному размеру (колонки\_а+колонки\_б, строки\_а+строки\_б). С классом Size можно использовать оператор +.**

bool conjB **– сопрягать ли массив b до умножения. *0* – если хотим провести свёртку, *1* – корреляцию (сравнение). Сопряжение это когда меняют знак на обратный у мнимой части членов ряда Фурье.**

1. **Как красиво нарисовать Фурье-образ. Красиво нарисовать Фурье образ нужно не только для того, чтобы красиво нарисовать, но и чтобы было удобно анализировать и обрабатывать изображение. Вот порядок использования нужных функций.**
2. void split(const Mat& src, Mat\* mvbegin)

**Разбивает исходное многоканальное изображение на массив одноканальных.**

Mat\* mvbegin **– выходной массив 1-канальных изображений. Должен быть заранее определён и иметь размер, равный количеству каналов входного изображения.**

1. **Теперь найдём магнитуду для каждого элемента образа Фурье:**

void magnitude(InputArray x, InputArray y, OutputArray magnitude)

1. **Дальше нам надо перейти к логарифмическому масштабу (пример):**

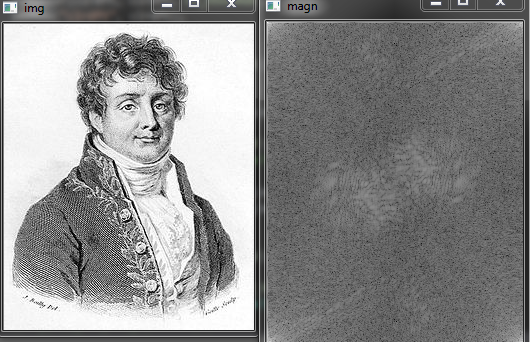
magn += Scalar::all(1);

log(magn, magn); //Натуральный логарифм для каждого элемента

1. **Нормализуем изображение, чтобы масштабировать значения его элементов до диапазона между 0 и 1 (для** CV\_32F**). В противном случае функция imshow(…) покажет нам подобие двоичного изображения(Это в случае изображений с вещественными числами).**

normalize(magn, magn, 0, 1, NormTypes::NORM\_MINMAX);

**Получим Фурье в 2-ух мерном образе самого себя:**



**Но картинка с магнитудой вышла не особо понятной и не удобной для анализа частотного (низкие частоты расположены по углам), зато красивой и похожа на космическую непонятную фигню. Чтобы привести её к принятой везде форме сделаем следующее. Данная функция перемещает квадранты (1\4 изображения) так, чтобы низкие частоты оказались в центре:**

void krasivSpektr(Mat &magI){

// rearrange the quadrants of Fourier image so that the origin is at the image center

int cx = magI.cols / 2;

int cy = magI.rows / 2;

Mat q0(magI, Rect(0, 0, cx, cy)); // Top-Left - Create a ROI per quadrant

Mat q1(magI, Rect(cx, 0, cx, cy)); // Top-Right

Mat q2(magI, Rect(0, cy, cx, cy)); // Bottom-Left

Mat q3(magI, Rect(cx, cy, cx, cy)); // Bottom-Right

Mat tmp; // swap quadrants (Top-Left with Bottom-Right)

q0.copyTo(tmp);

q3.copyTo(q0);

tmp.copyTo(q3);

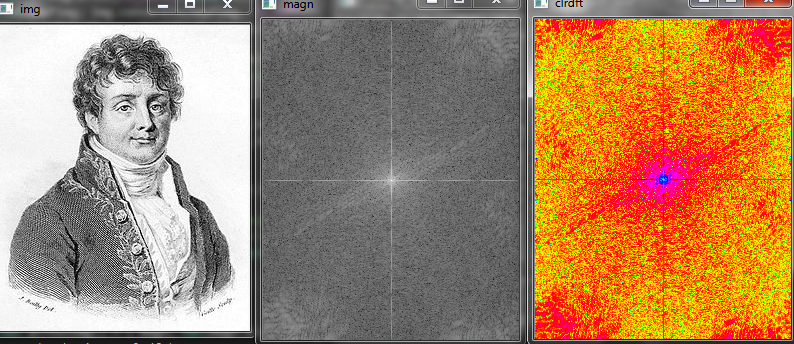
q1.copyTo(tmp); // swap quadrant (Top-Right with Bottom-Left)

q2.copyTo(q1);

tmp.copyTo(q2);

}

**И теперь можно получить красивую «понятную» картинку образа Фурье, особые молодцы могут сделать её цветной при помощи цветовой модели HSV.**



**Напоминает голубого гиганта посреди космической пыли (теперь вы можете рисовать космофотки). При желании можете построить фазовое изображение образа Фурье.**

**!!! Функции** mulSpectrums **и** dft **работают с образами Фурье только в их первоначальном виде, не приведённом к «красивому».**

1. **Функции, которые помогут вам выполнить задания.**

**Логические операции:**

1. void bitwise\_and(InputArray **src1**, InputArray **src2**, OutputArray **dst**, InputArray **mask**=noArray())
2. void bitwise\_not(InputArray **src**, OutputArray **dst**, InputArray **mask**=noArray())
3. void bitwise\_or(InputArray **src1**, InputArray **src2**, OutputArray **dst**, InputArray **mask**=noArray())
4. void bitwise\_xor(InputArray **src1**, InputArray **src2**, OutputArray **dst**, InputArray **mask**=noArray())
5. **Поэлементное** перемножение матриц

void multiply(InputArray **src1**, InputArray **src2**, OutputArray **dst);**

1. Как расширять холст:

Mat tempROI(dftimg, Rect(0, 0, img.cols, img.rows));

img.copyTo(tempROI); //копируем исходное изображение в ОИ от холста

dftimg **– ранее созданное изображение для холста, заполненное 0-ми, имеет размер больший размера оригинального.**

1. Поиск наименьшего и наибольшего значения на изображении:

void minMaxLoc(InputArray src, double\* minVal, double\* maxVal = 0, Point\* minLoc = 0, Point\* maxLoc = 0, InputArray mask = noArray());

**8) Пороговая фильтрация.**

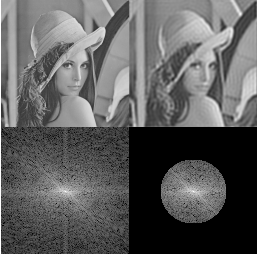
**ЗАДАНИЯ**

1. Загрузку изображений в программу (т.е. получение их имён и т.д.) крайне желательно осуществлять при помощи функций, указанных в 3-ей лаб.работе. Приучиваемся сразу грамотно и удобно загружать данные в программы. Операции свёртки и корреляции выполнять можно только при помощи представлений Фурье. Работаем с одноканальным изображением. Используем функцию повышения производительности.
2. Произвести по отдельности свёртку какого-либо изображения (шалуны могут использовать полное изображение Лены) с ядром фильтров: Собеля (по горизонтали и вертикали), усредняющего (BoxFilter), Лапласа . Необходимо «красиво» вывести магнитуду образа Фурье исходного изображения и ядра свёртки.

Полученные образы Фурье в результате выполнения свёртки следует обратно преобразовать в изображение. Сначала обрезаем полученное изображение (при помощи ОИ) до первоначального размера. Полученное изображение нормализуем (с теми же параметрами как и в примере, приведённом в теории) и выводим.

Функция **filter2D** [http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#filter2d] использует алгоритм, основанный на свёртке через ДПФ в случае ядер от 11х11 и больше.

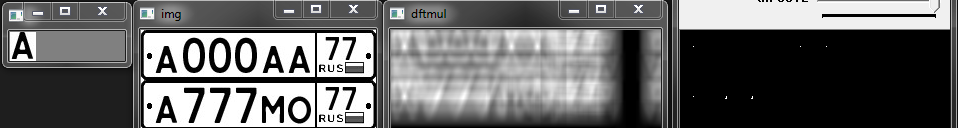
1. Взять какое-нибудь изображение и в его спектре обрезать в одном случае элементы спектра с высокими частотами, в другом – низкими. А потом выполнить обратное преобразование на основе полученных спектров:

Не забудьте после обрезания выполнить обратное перемещение квадрантов. Т.е. применить ещё раз krasivSpektr.

Такой подход позволит вам удобно создавать собственные ядра свёртки для фильтров. А отбрасывание гармоник (они же элементы образа Фурье) с малой магнитудой позволит вам хранить данные в сжатом виде. Наглядный тому пример, где Лене отрезали высокие частоты.

1. Провести корреляцию (сравнение) изображений автомобильных номеров по очереди с 3-мя символами. Полученный образ Фурье обратно преобразовать в обычное изображение. Найти на нём наибольшее значение, которое принимают элементы. Отнять от этого значения небольшое число (около 0.01). Использовать полученное число в качестве порога для пороговой фильтрации от полученного изображения.



Если честно, то я поленился обрезать 3-е изображение, но вам надо это сделать.

Данный метод позволяет примерно производить поиск по шаблону.

Подобный алгоритм (но более точный) использует функция [matchTemplate](http://docs.opencv.org/modules/imgproc/doc/object_detection.html?highlight=matchtemplate#matchtemplate)

[http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template\_matching/template\_matching.html]