Преобразование Фурье.

Определение. *Дискретным преобразованием Фурье* изображения $X[n,m] (0 \leqslant n < N, 0 \leqslant m < M)$ называется массив:

$$\hat{F}(X)[k,l] = \frac{1}{\sqrt{NM}} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} X[n,m] e^{-2\pi i \left(\frac{nk}{N} + \frac{ml}{M}\right)},\ (0 \leqslant k < N, 0 \leqslant l < M)$$

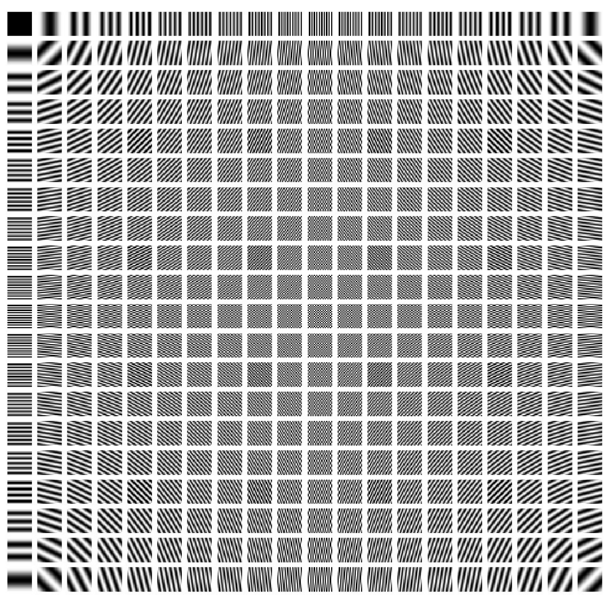
Обратным преобразованием Фурье называется массив:

$$\hat{f}(F)[n,m] = \frac{1}{\sqrt{NM}} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} F[k,l] e^{2\pi i \left(\frac{nk}{N} + \frac{ml}{M}\right)},\ (0 \leqslant n < N, 0 \leqslant m < M)$$

Несложно убедиться, что $\hat{f} \circ \hat{F} = \mathrm{Id}$

Что происходит

Рассмотрим семейство комплексных «изображений» $\{E_{kl}\}_{k=0,l=0}^{N-1,M-1} \in \mathbb{C}^{N,M} : E_{k,l}[n,m] = \frac{1}{\sqrt{NM}} e^{2\pi i \left(\frac{nk}{N} + \frac{ml}{M}\right)}$. Они образуют ортогональный базис. Соответственно, преобразование фурье – это всего лишь коэффициенты разложения данного изображения в этом базисе.
Для наглядности, нарисуем этот базис (точнее, его действительные компоненты) для $N = 20, M = 20$:



Слева изображен базис в нумерации, используемой выше. Базисные элементы с большими или маленькими k, l называются низкочастотными (потому, что каждый элемент базиса – это переодические полоски, и чем экстремальнее значения индексов, тем они реже), а со средними – высокочастотными. Низкочастотные компоненты изображения (то есть слагаемые в разложении на базисные вектора) отвечают за общий вид изображения, а высокочастотные – за мелкие детали. Теперь разберемся, зачем это может быть нужно.
Теперь разберемся, как можно применить преобразование Фурье.

Быстрые свертки.

Сверткой изображения $X \in \mathbb{C}^{N \times M}$ с ядром $Y \in \mathbb{C}^{K \times L}$ называется изображение $Z \in \mathbb{C}^{N+K-1 \times M+L-1}$:

$$Z[x, y] = \sum_{s=\max(0, x-N+1)}^{\min(K, x)} \sum_{t=\max(0, y-M+1)}^{\min(L, y)} Y[s, y] \cdot X[x-s, y-t]$$

Обозначение: $Z = (X * Y)$
Свертки часто используются в анализе и обработке изображений, однако наивное вычисление занимает $\mathcal{O}(NMKL)$ времени, что много для больших размеров ядер.
Однако на помощь приходит *Теорема о свертке*:

$$\hat{F}(X * Y) = \hat{F}(X')\hat{F}(Y')$$

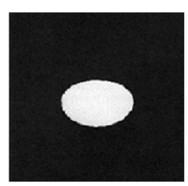
Где X', Y' – это X, Y добитые нулями справа и снизу до $(N + K - 1, M + L - 1)$.
То есть, чтобы посчитать свертку двух изображений, надо:

- Добить их нулями до нужного размера
- Посчитать преобразование Фурье от каждого из них
- Поэлементно перемножить
- Взять обратное преобразование

Осталось заметить, что прямое и обратное преобразование Фурье можно вычислить за $\mathcal{O}(NM\log(NM))$ времени. [Немного деталей: чтобы это сделать, достаточно уметь Быстро считать одномерные прямое и обратное преобразование фурье и осознать, что двумерное преобразование – это то же самое, что взять одномерное преобразование сначала независимо по столбцам, а потом по строкам]

Бесшовная склейка изображений

Задача. Даны два изображения X, Y (одинаковых размеров) и маска M (массив того же размера из нулей и единиц). Единица означать, что надо брать пиксель из первого изображения, а нолик – что из второго. Нужно создать коллаж.

Если мы напрямую будем брать пиксели согласно маске, то переход между изображениями будет слишком резкий, а хочется его сгладить.

Введем два понятия:

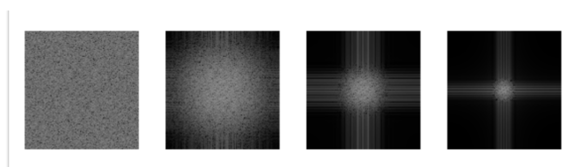
Определение. *Гауссовской пирамидой* будем называть последовательность изображений, $G_0, G_1, \ldots, G_k$ полученных из исходного изображения I_0 сверткой с гауссовским фильтром:

$$G_0 = I_0$$
$$G_1 = G_0 * G$$
$$\ldots$$
$$G_k = G_{k-1} * G$$

Лапласовской пирамидой будем называть последовательность изображений, $L_0, L_1, \ldots, L_k$ полученных из гауссовской пирамиды операцией попарной разности

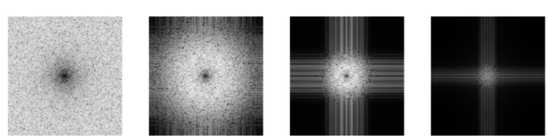
$$L_0 = G_0 - G_1$$
$$L_1 = G_1 - G_2$$
$$\ldots$$
$$L_{k-1} = G_{k-1} - G_k L_k = G_k$$

Замечание: обычно в определение гауссовской пирамиды добавляют еще и операцию уменьшения размера изображения в два раза после применения свертки (отсюда и название – пирамида), то есть $G_i = Subsample(G_{i-1} * G)$. В определении палдасовской пирамиды, соответственно, $L_i = G_i - Upsample(G_{i+1})$, но в нашем случае это нужно разве что для ускорения вычислений.
Итак, давайте посмотрим, что же происходит с точки зрения частот в этих пирамидах. Для этого надо заметить, что фурье-образ гауссовского фильтра – гауссовский фильтр.
Тогда в G_0 у нас есть все изображение, а дальше мы начинаем последовательно убирать высокие частоты, оставляя только низкие.



(изображены абсолютные значения преобразования Фурье, где ближе к центру изображены коэффициенты, отвечающие более низким частотам)

Что же происходит в лапласовской пирамиде? Для этого осознаем, что преобразование Фурье – линейная операция и вычитание в обычных координатах соответствует вычитанию в базисе Фурье. Если в G_0 у нас есть все частоты, а в G_1 – только низкие, то в $L_0 = G_0 - G_1$ у нас останутся только высокие частоты. Аналогично, в $L_1 = G_2 - G_1$ мы получим некоторую среднюю полосу частот (более низких, чем в L_0). Аналогично, в L_2 мы получаем еще более низкую полосу частот и так далее, пока в L_k не останутся только самые низкие частоты.



Заметим, что если сложить все элементы лапласовской пирамиды, то получится исходное изображение.
Идея алгоритма: поскольку мелкие детали изображений проявляются только в высоких частотах, их мы будем склеивать с помощью более резкой маски, а низкие частоты будем склеивать более плавной маской. Это и даст нам искомую плавность на границах.
Алгоритм:

1. Вычисляем лапласовские пирамиды исходных изображений L^X и L^Y, а так же гауссовскую пирамиду маски G^M

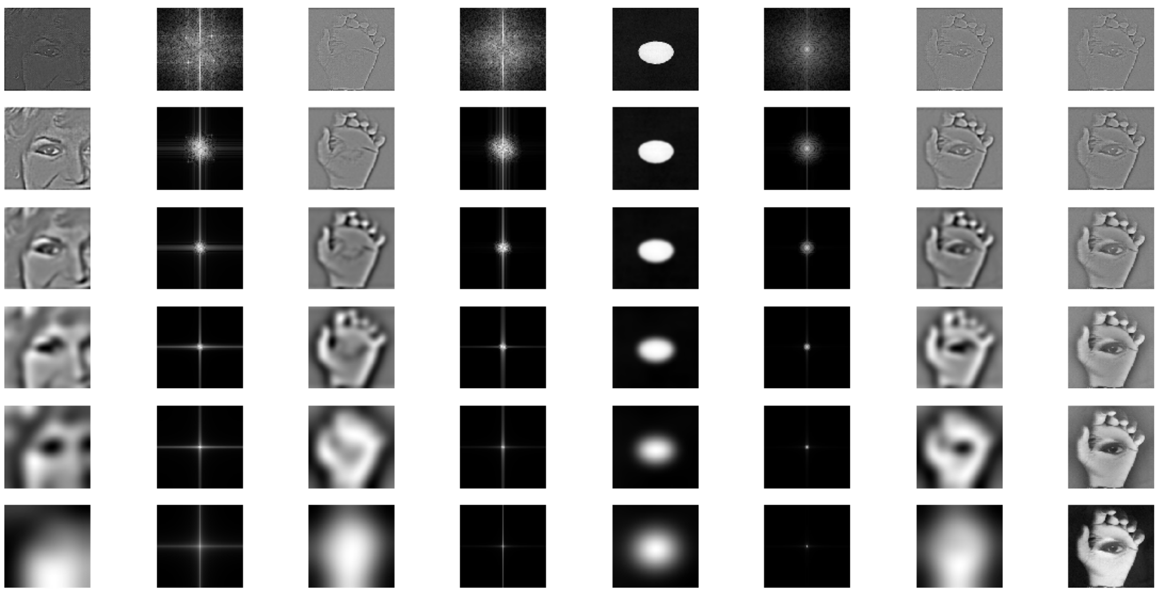
2. Комбинируем поуровнево лапласовские пирамиды с помощью пирамиды масок:

$$L_i = L_i^X \cdot G_i^M + (1 - G_i^M) \cdot L_i^Y$$

3. Получаем результат суммируя получившуюся пирамиду

$$C = \sum_i L_i$$

Пример:



(Изображены: $L^X, \hat{F}(L^X), L^Y, \hat{F}(L^Y), G^M, \hat{F}(G^M), L_i, \sum_{k=0}^{i} L_i$)

Заметим, что преобразованием Фурье мы пользовались лишь в теоретических рассуждениях, на практике мы его не вычисляли.

JPEG

Разберемся, как работает алгоритм сжатия JPEG.

1. Для начала изображение переводится в другое цветовое пространство так, чтобы одним из каналов стала яркость. В случае JPEG это YCbCr:

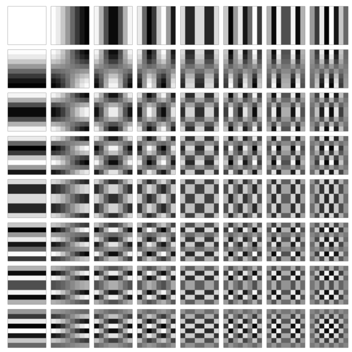
$$\begin{aligned}
Y' &= \quad 0 + (0.299 \cdot R'_D) + (0.587 \cdot G'_D) + (0.114 \cdot B'_D) \\
C_B &= 128 - (0.168736 \cdot R'_D) - (0.331264 \cdot G'_D) + (0.5 \cdot B'_D) \\
C_R &= 128 + (0.5 \cdot R'_D) - (0.418688 \cdot G'_D) - (0.081312 \cdot B'_D)
\end{aligned}$$

   Далее компоненты C_B и C_R уменьшаются в два раза, потому что, с одной стороны, это позволяет сэкономить метсто, а с другой – не сильно меняет визуальное восприятие (после разжатия), так как человеческий глаз более восприимчив к Y компоненте, нежели к цветным.

   После этого каждый из каналов разбивается на блоки 8×8 и каждый блок обрабатывается отдельно.

2. Из каждого блока вычитается 128 (если раньше значения были от 0 до 255, то теперь они от -128 до 128)

3. Далее к блоку применяется дискретное косинусное преобразование (DCT) – это как дискретное преобразование Фурье, только мы раскладываем изображение не по базису комплексных экспонент, а по базису $E_{k,l}[n,m] \propto \cos\frac{\pi(2n+1)k}{2N}\cos\frac{\pi(2m+1)l}{2M}$:



4. Далее происходит квантование амплитуд – мы нацело поэлементно делим результат DCT на специальную матрицу – матрицу квантования (зависящую от заранее выбраного параметра качества).

   Пример матрицы для качества в 50%:

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

   Смысл в том, что человеческий глаз получает большую часть информации из низких частот изображения, поэтому потерять информацию о высоких частотах не так страшно. Поэтому в матрице квантования более высоким частотам соответствуют более высокие значения – чтобы после деления нацело получить ноль с высокой вероятностью.

5. После этого у нас получается матрица с большим количеством нулей. Мы ее обходим (зигзагом), после чего применяем кодирование длин серий, а затем кодирование Хаффмана

   Чтобы раскодировать JPEG надо применить все в обратном порядке – для каждого блока раскодировать код Хаффмана и код серий, умножить на матрицу квантования, применить обратное дискретное косинусное преобразование, прибавить 128. Далее соединить все блоки в нужном порядке, увеличить размеры цветовых компонент и перевести обратно в RGB пространство.

Список источников

- Курс ШАДа "Анализ изображений и видео (часть 1)". В открытом доступе отсутствует, но эти темы освещены в курсе того же семинариста на stepik.org
- Wikipedia: JPEG
- Scipy documentation