

Обзор статей про восстановление размытых изображений

Процесс искажения изображений можно представить в виде формулы

$$G(u,v)=H(u,v)*F(u,v)+N(u,v)$$

Где:

$G(u,v)$ - искаженное изображение

$H(u,v)$ -искажающая функция

$F(u,v)$ -искажаемое изображение

$N(u,v)$ -адаптивный шум

*-операция свертки

Некоторые модели шумов

1) Гауссов шум

Функция плотности распределения гауссовой случайной величины z задается выражением

$$p(z)= \frac{1}{\sigma \sqrt{2\pi}} e^{-(z-\mu)^2/2\sigma^2}$$

Где: z – значение яркости

μ - среднее значение случайно величины z

σ -ее среднеквадратичное отклонение

2)Шум Релея

функция плотности распределения вероятностей шума Релея задается выражением

$$p(z)=\begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & \text{при } z \geq a \\ 0 & \text{при } z < a \end{cases}$$

где среднее и дисперсия имеют вид

$$\mu = a + \sqrt{\pi b/4}$$

$$\sigma^2 = \frac{b(4-\pi)}{4}$$

3)Шум Эрланга

$$p(z)=\begin{cases} \frac{a^b z^{(b-1)}}{(b-1)!} & \text{при } z \geq 0 \\ 0 & \text{при } z < 0 \end{cases}$$

где $a > 0$, b -положительное целое число

$$\mu = \frac{b}{a}$$

$$\sigma^2 = \frac{b}{a^2}$$

4)Экспоненциальный шум

$$p(z)=\begin{cases} a * e^{-az} & \text{при } z \geq 0 \\ 0 & \text{при } z < 0 \end{cases}$$

где $a > 0$

и среднее и дисперсия имеют вид

$$\mu = \frac{1}{a}$$

$$\sigma^2 = \frac{1}{a^2}$$

По сути, это распределение Эрланга с $b=1$

5)Равномерный шум

$$p(z)=\begin{cases} \frac{1}{b-a} \text{ при } a \leq z \leq b \\ 0, \text{ в остальных случаях} \end{cases}$$

среднее значение и дисперсия равны

$$\mu = \frac{a+b}{2}$$

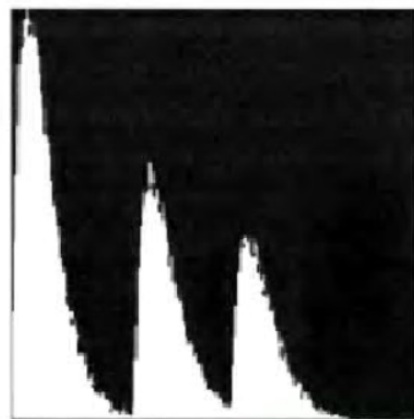
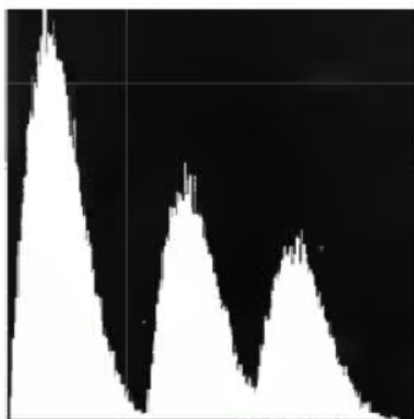
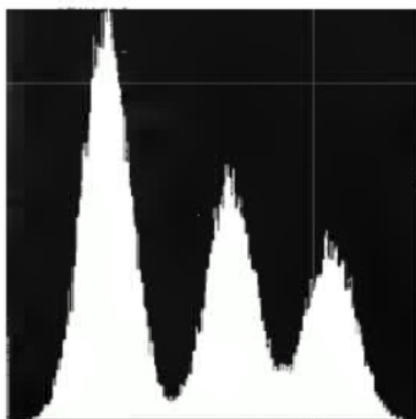
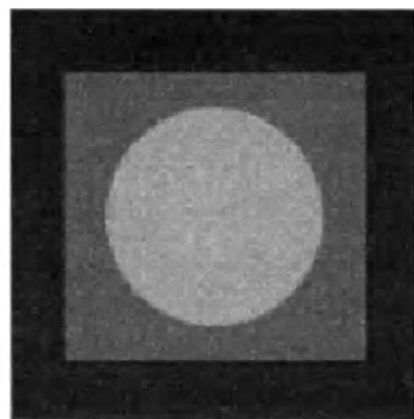
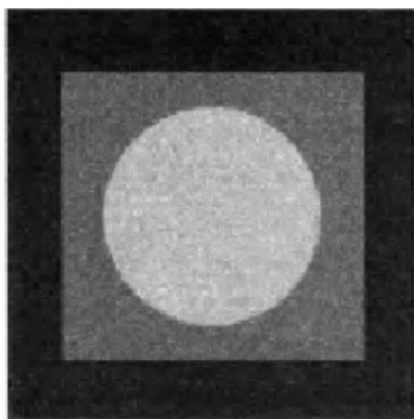
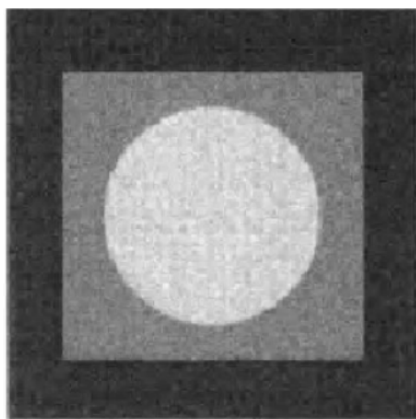
$$\sigma^2 = \frac{(b-a)^2}{12}$$

6)Импульсный шум

$$p(z)=\begin{cases} P_a, z=a \\ P_b, z=b \\ 0, \text{ иначе} \end{cases}$$

Эти распределения представляют собой средства для моделирования искажений

Исходное изображение:



Гауссов шум

Релеевский шум

Гамма шум

Экспоненциальный шум

Равномерный шум

Импульсный шум

Если изображение искажалось только шумами, то его вид следующий

$$g(x) = f(x, y) + n(x, y)$$
$$G(u, v) = F(u, v) + N(u, v)$$

Для подавления шумов существуют разные фильтры, основанные на порядковых статистиках, т. е. В этих фильтрах считаются среднее, дисперсия, медиана и еще необходимые данные, и в зависимости от них изменяются значения согласно тому или иному фильтру.

Адаптивные фильтры

Существуют так же адаптивные фильтры, они так называются, т. к. их поведение меняется в зависимости от статистических свойств изображения внутри прямоугольной области $m \times n$ в окрестности S_{xy} .

Рассмотрим следующий фильтр:

$$\hat{f} = g(x, y) - \frac{\sigma_n^2}{\sigma_L^2} [g(x, y) - m_L]$$

где

\hat{f} - восстановленное изображение
 $g(x, y)$ - искаженное изображение
 σ_n^2 - дисперсия по всему изображению
 σ_L^2 - дисперсия по окрестности S_{xy}
 m_L - среднее по окрестности S_{xy}

если дисперсия по окрестности больше дисперсии по всему изображению, то это отношение равно 1

Попытка реализовать этот фильтр на Python

```
def addapt_loc_filter():
    im=plb.imread("olen.jpg")
    bwi=make_black_white_im(im)
    n=7
    h=len(im[:])
    w=len(im[0,:])
    bw=np.zeros((h,w,3))
    bw[:, :, 0]=bwi
    bw[:, :, 1]=bwi
    bw[:, :, 2]=bwi
    plb.imsave("olen_wb.jpg", bw)
    d_gl=np.var(bwi)
    for i in range(n/2, h-n/2):
        for j in range(n/2, w-n/2):
            m=np.mean(bwi[i-n//2:i+n//2+1, j-n//2:j+n//2+1])
            d=np.var(bwi[i-n//2:i+n//2+1, j-n//2:j+n//2+1])
            k=0
            if (d_gl>d):
                k = 1
            else:
                k = float(d_gl)/d
            bw[i, j, 0]=bwi[i, j]-k*(bwi[i, j]-m)
            bw[i, j, 1] = bwi[i, j] - k * (bwi[i, j] - m)
            bw[i, j, 2] = bwi[i, j] - k * (bwi[i, j] - m)
    bw = np.uint8(bw)
    plb.imsave("addapt_loc_filter/res9.jpg", bw)
```

Оценка искажающей функции

Существует 3 основных методов оценки искажающей функции (ядра искажающего оператора)

- 1) визуальный анализ
- 2) эксперимент
- 3) математическое моделирование

1) Рассмотрим оценку на основе визуального анализа изображения

Предположим, что есть искаженное изображение, но информация об искажающей функции H отсутствует. Идея в том, чтобы оценить эту функцию непосредственно из изображения. Мы можем (наверное) рассмотреть ту область изображения, которая содержит полезный сигнал большой амплитуды. Используя яркости объекта и фона, мы приблизительно можем построить неразмытое изображение (наверное, но я такого не делал)

обозначим рассматриваемую область $g_{(x,y)}$ а построенное изображение как $\hat{f}_s(x,y)$

Тогда, предположив, что влияние шума пренебрежительно мало, с большим полезным сигналом, имеем

$$H_s(u,v) = \frac{G_s(u,v)}{\hat{F}_s(u,v)}$$

исходя из свойств функции $H_s(u,v)$ мы можем сделать вывод о полной искажающей функции $H(u,v)$, если использовать тот факт, что искажения предполагаются трансляционно-инвариантными

2) оценка на основе эксперимента

Если доступно такое же оборудование, которое использовалось при получении изображения, то, в принципе, можно получить точную оценку искажающей функции

Для начала надо подобрать параметры системы, чтобы искажения на получаемых с ее помощью изображениях как можно лучше соответствовали искажениям на изображении. Далее идея в том, чтобы сформировать импульсный отклик (ядро искажающего оператора), для чего надо получить изображение импульса, используя систему с подобранными операторами. Импульс симулируется яркой световой точкой. Чтобы уменьшить влияние шума, яркость должна как можно больше. Затем, учитывая, что фурье-преобразование импульса есть константа, получаем

$$H(u,v) = \frac{G(u,v)}{A}$$

Где $G(u,v)$ - фурье-преобразование полученного изображения

A - константа, описывающая величину яркости импульса

3) Оценка на основе моделирования

Модель искажения, возникающей при из-за турбулентности атмосферы

$$H(u,v) = \exp(-k(u^2 + v^2)^{5/6})$$

Где k - константа, описывающая турбулентные свойства атмосферы

Это выражение с точностью до коэффициента $5/6$ совпадает по форме с выражением гауссова низкочастотного фильтра

Модель размытия, возникающей в результате равномерного поступательного движения изображения фотоаппарата или другого устройства.

Предположим, что $f(x,y)$ участвует в плоском движении и что функции $x_0(t), y_0(t)$

определяют закон движения в направлениях x и y соответственно. Полная экспозиция в любой точке записывающего устройства - это интеграл по времени от величины мгновенной экспозиции

$$g(x,y) = \int_0^T f(x - x_0(t), y - y_0(t)) dt$$

где $g(x,y)$ - смазанное изображение

Фурье преобразование имеет вид

$$G(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(x, y) e^{-i2\pi(ux+vy)} dx dy = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \left[\int_0^T f(x-x_0(t), y-y_0(t)) dt \right] e^{-i2\pi(ux+vy)} dx dy$$

Изменив порядок интегрирования

$$G(u, v) = \int_0^T \left[\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x-x_0(t), y-y_0(t)) e^{-i2\pi(ux_0(t)+vy_0(t))} dx dy \right] dt = \int_0^T F(u, v) e^{-i2\pi(ux_0(t)+vy_0(t))} dt$$
$$G(u, v) = F(u, v) \int_0^T e^{-i2\pi(ux_0(t)+vy_0(t))} dt$$

Используя ранее известную формулу $G(u, v) = F(u, v) H(u, v)$

Получим

$$H(u, v) = \int_0^T e^{-i2\pi(ux_0(t)+vy_0(t))} dt$$

Если нам известны функции $x_0(t), y_0(t)$, то мы можем легко вычислить искажающую функцию

Инверсная фильтрация

Самый простой способ восстановления. Предполагает получение оценки $\hat{F}(u, v)$ - фурье-преобразования исходного изображения делением фурье-преобразования искаженного изображения на частотное представление искажающей функции

$$\hat{F}(u, v) = \frac{G(u, v)}{H(u, v)}$$

Деление поэлементное. Попытка реализации на языке Python

```
def inverse_filter(g, h):
    width_g=g.shape[0]
    height_g=g.shape[1]
    width_h=h.shape[0]
    height_h=h.shape[1]
    g1=np.zeros((2*width_g,2*height_g))
    h1=np.zeros((2*width_g, 2*height_g))
    g1[0:width_g,0:height_g]=g
    h1[0:width_h, 0:height_h] = h
    G=np.fft.fft2(g1)
    H=np.fft.fft2(h1)
    F=G/H
    f=np.fft.ifft2(F)
    f=np.real(f)
    f=f[0:width_g,0:height_g]
    return f
```



Фильтрация методом минимизация среднеквадратического отклонения (винеровская фильтрация)

Фильтр Винера является методом, соединяющий в себе учет свойств искажающей функции и статистических свойств шума в процессе восстановления. Метод основан на рассмотрении изображений и шума как случайных процессов, и задача ставится следующим образом: найти такую оценку \hat{f} для неискаженного изображения f чтобы среднеквадратичное отклонение этих величин было минимальным. Стандартное отклонение задается следующей формулой:

$$e^2 = E\{(f - \hat{f})^2\}$$

Предполагается, что выполнены условия:

- 1) шум и неискаженное изображение некоррелированы между собой
- 2) либо шум, либо неискаженное изображение имеют нулевое среднее значение
- 3) оценка линейно зависит от искаженного изображения. При выполнении этих условий минимум среднеквадратичного отклонения достигается на функции, которая задается в частотной области выражением

$$\hat{F}(u, v) = \left(\frac{H'(u, v) S_f(u, v)}{S_f(u, v) |H(u, v)|^2 + S_n(u, v)} \right) G(u, v) = \left(\frac{1}{H(u, v)} \frac{|H(u, v)|^2}{|H(u, v)|^2 + S_n(u, v) / S_f(u, v)} \right) G(u, v)$$

Где:

$H(u, v)$ - искажающая функция (ее частотное представление)

$H'(u, v)$ - комплексное сопряженное искажающей функции

$|H(u, v)|^2 = H'(u, v) H(u, v)$

$S_n = |N(u, v)|^2$ - энергетический спектр шума

$S_f(u, v) = |F(u, v)|^2$ - энергетический спектр неискаженного изображения

$G(u, v)$ - фурье-преобразование искаженного изображения

Последнее равенство имеет место, из-за того, что произведение комплексного числа на комплексно-сопряженное равно квадрату модуля

Этот результат был получен Винером, и этот результат известен как оптимальная фильтрация по Винеру. Фильтр внутри скобок часто называют фильтром среднеквадратического отклонения или винеровским фильтром.

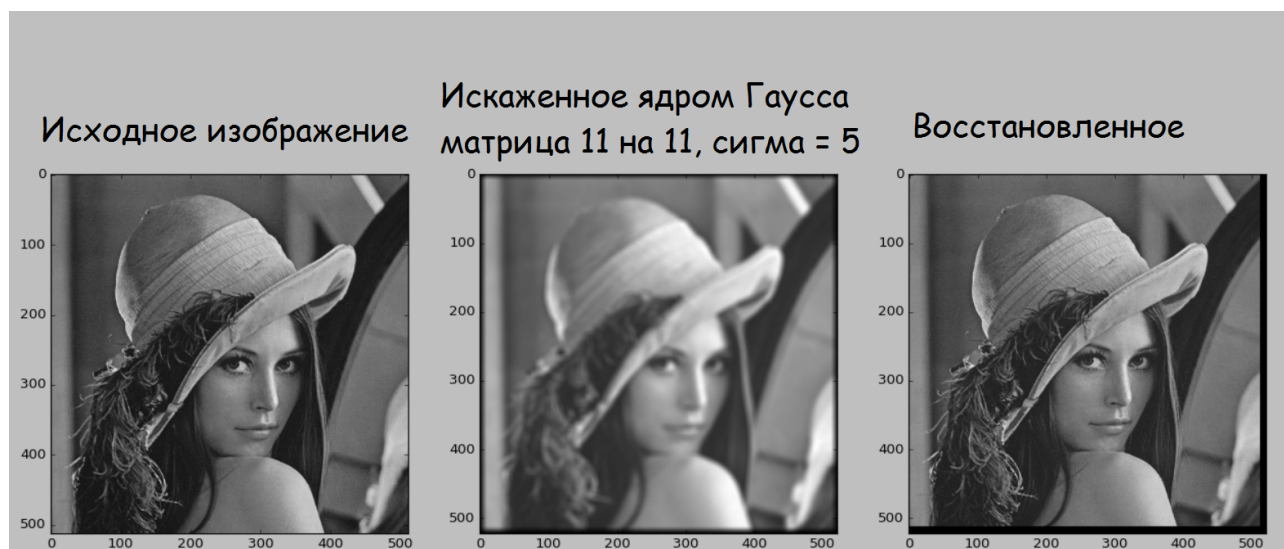
Когда спектр шума и неискаженного изображения неизвестно и не могут быть оценены, часто пользуются подходом, состоящий в предыдущего выражения следующим

$$\hat{F}(u,v) = \left(\frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + K} \right) G(u,v)$$

где K- определенная константа.

Попытка реализации фильтра на языке Python

```
def wiener_filter(g, h, k):
    width_g=g.shape[0]
    height_g=g.shape[1]
    width_h=h.shape[0]
    height_h=h.shape[1]
    g1=np.zeros((2*width_g,2*height_g))
    h1=np.zeros((2*width_g, 2*height_g))
    g1[0:width_g,0:height_g]=g
    h1[0:width_h, 0:height_h] = h
    G=np.fft.fft2(g1)
    H=np.fft.fft2(h1)
    k1=np.zeros((2*width_g, 2*height_g))
    k1[0:k.shape[0],0:k.shape[1]]=k
    K=np.fft.fft2(k1)
    #F=(np.abs(H)**2/(H*((np.abs(H)**2))+K))*G // с K все хуже
    пока
    F = (np.abs(H) ** 2 / (H * ((np.abs(H) ** 2)) )) * G
    f=np.fft.ifft2(F)
    f=np.real(f)
    f=f[0:width_g,0:height_g]
    return f
```

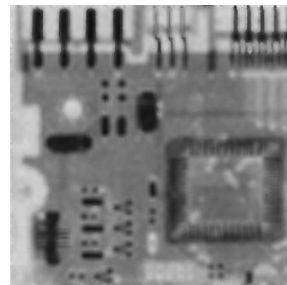
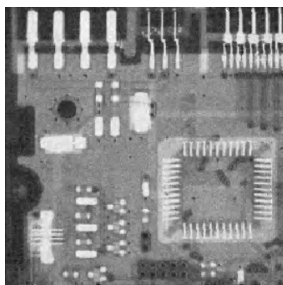


Регуляризация по Тихонову

Для этого метода надо знать среднее и дисперсию шума. Так же этот метод обладает тем свойством, что позволяет получить оптимальный результат для каждого конкретного изображения, к которому применяется.

$$\hat{F}(u,v) = \left(\frac{H'(u,v)}{|H(u,v)|^2 + \gamma |P(u,v)|^2} \right) G(u,v)$$

// Кода еще нет, т. к. я не очень понял как все-таки получить оценку искажающей функции. А локальный адаптивный алгоритм работает мягко говоря, не очень



слева исходные изображения, справа «восстановленные»

попытка с новыми знаниями реализовать это дело на питончике

```
def tickhonov_regularization(g,h):
    p=np.array([
        [0,-1,0],
        [-1, 4, -1],
        [0,-1,0.0]
    ])
    width_g = g.shape[0]
    height_g = g.shape[1]
    width_h = h.shape[0]
    height_h = h.shape[1]
    g1 = np.zeros((2 * width_g, 2 * height_g))
    h1 = np.zeros((2 * width_g, 2 * height_g))
    g1[0:width_g, 0:height_g] = g
    h1[0:width_h, 0:height_h] = h
    G = np.fft.fft2(g1)
    H = np.fft.fft2(h1)
    p1=np.zeros((2 * width_g, 2 * height_g))
    p1[0:3,0:3]=p
    P=np.fft.fft2(p1)
```



```

gamma=0. #это пока только так
F=(np.conjugate(H) / (np.abs(H)**2+gamma*np.abs(P)**2)) *G
f = np.fft.ifft2(F)
f = np.real(f)
f = f[0:width_g, 0:height_g]
return f

```



Метрика сравнения изображений.

Пока релизовал функцию в 1 строчку, которая вычисляет сумму квадратов разностей между 2 изображениями. В Интернетах говорят, что это нормально <https://habrahabr.ru/post/122372/>, по крайней мере для изображений одинакового размера. Для разных там возникают сложности

+ ко всему этому код

```

def comp_image(a,b):
    return np.sum((a-b)**2)

```

Частотные методы улучшения

Хоть, по логике, эта тема должна была быть до обзора фильтра Винера, регуляризации по Тихонову, но так получилось, это не в моих силах. И чтобы лучше попробовать понять преобразования Фурье, на эту тему я так же решил попробовать написать, обзор, чтобы пропустить всю информацию сначала через мозг, а потом через руки.

После этого лирического отступления, хотелось бы сказать пару слов, формул о преобразованиях Фурье.

1) Прямое фурье-преобразование (фурье-образ) $F(u)$ непрерывной функции одной переменной $f(x)$ определяется равенством

$$F(u) = \int_{-\infty}^{+\infty} f(x) e^{-i2\pi ux} dx$$

где i - мнимая единица, ($i^2 = -1$)

Также по заданному фурье-преобразованию $F(u)$ можно получить исходную функцию $f(x)$ при помощи обратного преобразования Фурье:

$$f(x) = \int_{-\infty}^{+\infty} F(u) e^{i2\pi ux} dx$$

Эти преобразования составляют **пару преобразований Фурье**, а входящие в них функции образуют **фурье-пару**.

Эти преобразования можно распространить и на функции от двух переменных

$$F(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) e^{-i2\pi(ux+vy)} dx dy$$

и аналогично для обратного преобразования

$$f(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(u, v) e^{i2\pi(ux+vy)} dx dy$$

Но т. к. изображения — это все-таки дискретные функции, нас интересуют именно они
Фурье-преобразование дискретной функции одной переменной

$f(x)$, $x=0, 1, 2, \dots, M-1$ задаются равенством

$$F(u, v) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-i2\pi ux/M}, u=0, 1, 2, \dots, M-1$$

Это (**прямое**) **дискретное преобразование Фурье (ДПФ)**

Также можно восстановить исходную функцию по заданному фурье-преобразованию при помощи **обратного ДПФ**

$$f(x) = \sum_{u=0}^{M-1} F(u) e^{i2\pi ux/M}, x=0, 1, \dots, M-1$$

Множитель $\frac{1}{M}$ часто ставится в формуле, определяющей обратное, а не прямое

преобразование Фурье. Реже оба равенства содержат $\frac{1}{\sqrt{M}}$

Местоположение множителя не имеет значения. Единственное требование при

использовании двух множителей состоит в том, чтобы их произведение было равно $\frac{1}{M}$

Эти формулы, хоть и просты, но очень важны

Понятие частотной области следует прямо из формулы Эйлера

$$e^{i\theta} = \cos \theta + i \sin \theta$$

подставляя эту формулу в формулу дискретного Фурье-преобразования, получим

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) [\cos 2\pi ux/M - i \sin 2\pi ux/M] \quad u=0, 1, \dots, (M-1)$$

Таким образом, мы видим, что каждый элемент Фурье-преобразования состоит из суммы по всем значениям функции $f(x)$, а значения функции $f(x)$, в свою очередь, умножаются на синусы и косинусы разных частот. Область значений переменной u , на которой принимает свои значения функция $F(u)$ называется **частотной областью**.

Каждый из M элементов функции $F(u)$ называется **частотной компонентой** преобразования. Использование терминов *частотная область* и *частотные компоненты* по существу не отличается от использования терминов *временная область* и *временные компоненты*, которыми будут обозначаться область определения и значения функции $f(x)$ в случае, когда x - временная переменная (или не будут, если не войдут в обзор)

Как и в случае комплексных чисел, значения $F(u)$ иногда удобно выражать в полярных координатах

$$F(u) = |F(u)| e^{-i\phi(u)},$$

где величины

называются **модулем** или **спектром** фурье-преобразования, а величины

$$\phi(u) = \arctg \left[\frac{I(u)}{R(u)} \right]$$

называются **фазой** или **фазовым спектром** преобразования Фурье.

и $I(u)$ обозначают действительную и мнимую части величины $F(u)$

Другая величина, так же используемая в этой главе книги ЦОИ является **энергетический спектр**, который определяется как квадрат фурье-спектра

$$P(u) = |F(u)|^2 = R^2(u) + I^2(u)$$

Наряду с термином **энергетический спектр** используется также термин **спектральная плотность**

Для двумерных дискретных фурье-преобразований фурье-спектр, фаза и энергетический спектр определяются таким же образом как и для одномерных

Обычно так же умножают исходную функцию на $(-1)^{x+y}$, используя свойства экспонент, говорят, что нетрудно доказать

$$\text{Фурье преобразование } (f(x, y)(-1)^{x+y}) = F(u - M/2, v - N/2)$$

я просто букву не нашел

Если функция $f(x, y)$ - вещественная, то ее фурье-преобразование обладает симметрией по отношению к операциям комплексного сопряжения

$$F(u, v) = F^*(u, v)$$

* означает обычное комплексное сопряженное

отсюда следует

$$|F(u, v)| = |F(-u, -v)|$$

которое говорит о том, что спектр фурье-преобразования симметричен

Процедура фильтрации в частотной области

процедура состоит из следующих шагов:

1. Исходное изображение умножается на $(-1)^{x+y}$, чтобы его фурье-преобразование оказалось центрированным //обычно не делают
2. Вычисляется прямое ДПФ изображения, полученное после шага 1

3. Функция $F(u, v)$ умножается на функцию фильтра

4. Вычисляется обратное ДПФ от результата предыдущего шага

5. Выделяется вещественная часть

6. Результат умножается на $(-1)^{x+y}$

Причина по которой множитель $H(u, v)$ называется фильтром состоит в том, что он подавляет некоторые частотные преобразования, оставляя другие при этом без изменения.

Слепая деконволюция (blind deconvolution)

Попробуем про слепую деконволюцию

Методы Винера, регуляризация по Тихонову предполагают, что $h(x, y)$ у нас известно до проведения деконволюции, но на практике в большинстве случаев это не так, к сожалению.

И для того, чтобы пробовать восстанавливать искаженные изображения, не зная при этом ядро искажающего фильтра, был разработан метод слепой деконволюции

Есть несколько важных составляющих проблемы слепой деконволюции изображений:

1. Для проведения деконволюции изображение и функция импульсного отклика должны обладать свойством нескоратимости. **Несократимый сигнал** — это сигнал, который не может быть точно выражен с помощью конволюции двух или более компонентов сигнала, при учете, что двумерная дельта-функция не является компонентой сигнала. Это важное свойство системы, т. к. если изображение или функция импульсного отклика будут сократимыми, тогда полученное решение будет приближенным.

2. В классическом подходе к восстановлению линейными методами целью является получение точного приближения к реальному изображению. В идеале $\hat{f}(x, y) = f(x, y)$, где $\hat{f}(x, y)$ - приближенное изображение (оценка) полученная с помощью процедуры восстановления. Цель методов слепого восстановления — получить масштабированную и сдвинутую копию оригинального изображения. После проведения слепой деконволюции, если это возможно, масштабирование и сдвиг могут быть восстановлены с помощью дополнительных действий.

3. На практике изображение представляют в виде свертки, я 100 раз писал эту формулу, мне лень повторять

Существующие методы восстановления изображений

существует 2 основных подхода к задаче слепой деконволюции:

1. Определение функции импульсного отклика $h(x, y)$ отдельно от восстанавливаемого изображения, чтобы использовать полученную информацию позже, применив один из классических методов восстановления. Оценка импульсного отклика и восстановление изображения — это отдельные процедуры в данном подходе. Алгоритмы, используемые для данного метода являются вычислительно простыми

2. Включение процедуры определения функции импульсного отклика в восстанавливающий алгоритм. Этот подход подразумевает одновременную оценку функции импульсного отклика и восстанавливаемого изображения, что приводит к более сложным алгоритмам

Слепая деконволюция одиночного объекта — некорректная задача, т. е. неизвестных слишком много. И раньше на ядро смаза накладывали ограничения и использовались параметризованные формы для ядра. Недавно были предложены методы для обработки более общих случаев смаза, которые используют изображение одиночного объекта.

Большинство методов работают итеративно, т. е. Попеременно оптимизируют ядро смаза и скрытое изображение.

В последнее время многие авторы добились успехов в решении задачи слепой деконволюции, используя поочередную оптимизацию f и h в итеративном процессе восстановления. Для того, чтобы ускорить вычисление изображения f был введен «предсказывающий» шаг. Резкие (сильно выделяющиеся) края предсказываются по вычисленному искомому изображению f на шаге предсказываний и потом самостоятельно используются для получения ядра.

На шагах вычисления скрытого изображения и определения ядра в указанном порядке решаются уравнения вида:

$$\begin{aligned} f' &= \arg \min_f \|g - h * f\| + p_f(f) \\ h' &= \arg \min_h \|g - h * f\| + p_h(h) \end{aligned}$$

в этих уравнениях компонента $\|g - h * f\|$ отвечает за подгонку данных, для которых использовалась форма L_2 , а компоненты p_f, p_h - компоненты регуляризации.

Попытка написать алгоритм быстрой слепой деконволюции

3 основных шага в каждой итерации:

- предсказание

- нахождение ядра
- деконволюция

1) Предсказание

Входные данные: смазанное изображение g

Вычисление карты градиентов $\{P_x, P_y\}$ (фильтром Собеля, на сколько я помню) от f по направлениям x и y , для предсказания области с резкими краями L с подавлением шума в гладких областях

f находится на предыдущей итерации, т. е. Это f_i на i -том шаге итерации (а что тогда использовать на первом шаге, полагаю g – вопрос!!!)

2) Нахождение ядра

минимизировать функцию

$$\text{fun } c_h(h) = \sum_{\delta} \omega \|h * P - g\| + \beta \|h\|^2$$

там все кроме h со звездочкой

омега со звездочкой показывают вес для каждой частной производной и там веселуха с P и g

каждый член $(h * P - g)$ формирует карту

определяем $\|I\|^2 = \sum_{x,y} I(x,y)^2$ для карты I , (x,y) – координаты пикселей в I . Бетта- вес для

регуляризации Тихонова

3) Деконволюция

при помощи известных h и g вычисляем f , которое будет использоваться в следующей итерации (Деконволюция происходит любым известным способом??? фильтр Винера и т. п., или раз упоминают о регуляризации по Тихонову, надо его использовать???)

Метод Люси-Ричардсона

относится к итерационным методам восстановления, как и БСД

как я понял, этот метод используется для восстановления изображений, искаженных линейным смазом (вроде так, камера или объект движутся)

что такое кепстр???

Используемая литература

Вудс Р., Гонсалес Р. -Цифровая обработка изображений

<https://habrahabr.ru/post/136853/>

<https://habrahabr.ru/post/147828/>

<https://habrahabr.ru/post/152885/>

<https://habrahabr.ru/post/175717/>

http://courses.graphicon.ru/files/courses/vision/2010/cv_2010_02.pdf

Переславцева Е.Е., Филиппов М. В. Метод ускоренного восстановления изображений

К.В. Панфилова Компенсация линейного смаза цифровых изображений с помощью метода Люси-Ричардсона