Восстановление искаженных изображений

Никита Лисин, Арсений Широков, Влад Шахуро

Данное задание посвящено восстановлению искаженных изображений. Мы будем рассматривать восстановление размытых изображений, но рассматриваемые методы могут быть применены и к другим видам искажений.

Формализуем процесс искажения. Для простоты будем рассматривать одноканальные изображения в градациях серого. Для восстановления цветных изображений достаточно применить процедуру восстановления к каждому цветовому каналу. Будем использовать следующую модель искаженного изображения:

$$g(x, y) = f(x, y) * h(x, y) + \eta(x, y), \tag{1}$$

$f(x, y)$ — исходное неискаженное изображение,

$h(x, y)$ — искажающая функция,

$*$ — операция свертки в пространственной области,

$\eta(x, y)$ — аддитивный шум,

$g(x, y)$ — результат искажения (смазанное изображение).

1. Гауссовская функция (1 балл)

В данном задании мы будем восстанавливать только размытые изображения. В роли искажающей функции h выступает фильтр Гаусса:

$$h(r) = \frac{1}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}}.$$

Здесь $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$, x_0, y_0 – центр фильтра.

Реализуйте функцию `gaussian_kernel`, возвращающую ядро фильтра Гаусса заданного размера и с заданным значением параметра σ. Сумма всех элементов ядра должна равняться 1. Проверьте реализацию с помощью юнит-теста:

```
$ ./run.py unittest gaussian
```

2. Инверсная фильтрация (2 балла)

Свертка в пространственной области — достаточно сложная операция. Чтобы обратить свертку, перейдем в частотную область и воспользуемся теоремой о свертке. Теорема гласит, что свертка в пространственной области эквивалентна умножению в частотной области:

$$f * h = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{h\}\}.$$

Модель искаженного изображения (1) с помощью теоремы о свертке может быть записана следующим образом:

$$G(u, v) = F(u, v) \cdot H(u, v) + N(u, v). \tag{2}$$

Заглавными буквами обозначены Фурье-образы соответствующих функций в (1). При этом свертка функций заменяется на умножение функций.

Простейший способ восстановления исходного изображения — инверсная фильтрация. В этом методе оценка $\tilde{F}(u, v)$ Фурье-образа исходного изображения получается по формуле

$$\tilde{F}(u, v) = \frac{G(u, v)}{H(u, v)} \tag{3}$$

Деление здесь поэлементное. Подставив правую часть выражения (2) в (3), получим оценку

$$\tilde{F}(u, v) = F(u, v) + \frac{N(u, v)}{H(u, v)} \tag{4}$$

Видно, что при наличии шума точно восстановить исходное изображение невозможно, поскольку функция $N(u, v)$ неизвестна. Поэтому инверсную фильтрацию обычно используют при отсутствии шума, в таком случае удается точно восстановить изображение.

Имеется и другая проблема. Если функция $H(u, v)$ принимает нулевые или близкие к нулевым значения, то вклад второго слагаемого в правой части (4) может стать преобладающим, что часто бывает на практике. Проблему можно решить, убрав близкие к нулю значения H. Формулу (3) можно переписать в следующем виде:

$$\tilde{F}(u, v) = G(u, v)H_{inv}(u, v), \tag{5}$$

$$H_{inv}(u, v) = \begin{cases} 0, & \text{если } |H(u, v)| \leqslant \text{threshold} \\ \frac{1}{H(u,v)}, & \text{иначе} \end{cases}$$

Но стоит понимать, что расматриваемый метод никак не учитывает наличие и характеристики шума, поэтому на хорошее приближение исходного изображения можно надеяться только при условии отсутствия шума.

Фурье-образ искажающей функции (0.5 балла)

Реализуйте функцию `fourier_transform`, возвращающую Фурье-образ искажающей функции заданного размера. Обратите внимание, что искажающая матрица зачастую имеет размерность меньше размерности изображения, но размерности их Фурье-образов должны совпадать. Чтобы этого добиться, достаточно дополнить матрицу h нулями до нужного размера. Реализацию преобразования Фурье возьмите из библиотеки scipy. Проверьте реализацию с помощью юнит-теста:

```
$ ./run.py unittest fourier_transform
```

Вычисление H_{inv} (0.5 балла)

Реализуйте функцию `inverse_kernel`, возвращающую H_{inv} по известному H. Проверьте реализацию с помощью юнит-теста:

```
$ ./run.py unittest inverse_kernel
```

Оценка исходного изображения (1 балл)

Реализуйте функцию `inverse_filtering` инверсной фильтрации по формуле (5):

1. Получите оценку $\tilde{F}$.
2. Получите оценку $\tilde{f}$, воспользовавшись обратным преобразованием Фурье.
3. Матрица $\tilde{f}$ получится комплексной. Возьмите каждый её элемент по модулю, чтобы получить восстановленное изображение.

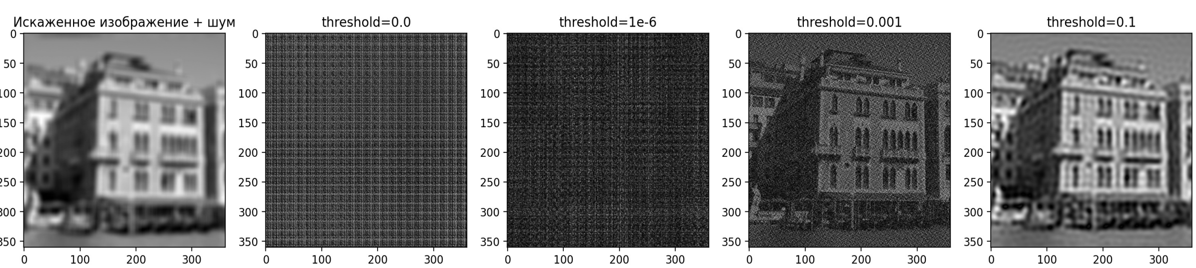
Проверьте реализацию с помощью юнит-теста:

```
$ ./run.py unittest inverse_filtering
```

Результаты работы функции можно отобразить с помощью команды

```
$ ./visualization.py inverse
```

Скрипт создаст изображения `inverse_filtering_blurred.jpg` и `inverse_filtering_noisy.jpg` с визуализацией:



Винеровская фильтрация (2 балла)

Следующий метод работает с зашумленными изображениями. Винеровский метод основан на рассмотрении изображений и шума как случайных переменных. Задача ставится следующим образом: найти такую оценку $\tilde{f}$ для неискаженного изображения f, чтобы средний квадрат отклонения этих величин друг от друга (ошибка) был минимальным:

$$e^2 = \mathsf{E}[(f - \tilde{f})^2]. \tag{6}$$

Будем считать, что выполнены следующие условия:

1. Шум и неискаженное изображение не коррелированы между собой.
2. Либо шум, либо неискаженное изображение имеют нулевое среднее значение.
3. Оценка линейно зависит от искаженного изображения.

Тогда минимум среднего квадрата отклонения (6) достигается на функции, которая задается в частотной области выражением

$$\tilde{F}(u,v) = \frac{\overline{H(u,v)}}{|H(u,v)|^2 + \frac{S_\eta(u,v)}{S_f(u,v)}} G(u,v),$$

$H(u,v)$ — Фурье-образ искажающей функции,

$\overline{H(u,v)}$ — комплексное сопряжение $H(u,v)$,

$|H(u,v)|^2 = \overline{H(u,v)}H(u,v)$,

$S_\eta(u,v)$ — энергетический спектр шума,

$S_f(u,v)$ — энергетический спектр неискаженного изображения.

Данный результат был получен Норберт Винером и известен как оптимальная фильтрация по Винеру. В тех случаях, когда спектры шума и неискаженного изображения неизвестны и не могут быть оценены, используется подход, состоящий в аппроксимации их отношения некоторой константой K:

$$\tilde{F}(u,v) = \frac{\overline{H(u,v)}}{|H(u,v)|^2 + K} G(u,v), \tag{7}$$

Далее будем использовать именно эту формулу.

Оценка исходного изображения (1 балл)

Реализуйте функцию `wiener_filtering` оптимальной фильтрации по Винеру:

1. Получите оценку $\tilde{F}$, используя формулу (7).
2. Получите оценку $\tilde{f}$ с помощью обратного преобразования Фурье.
3. Матрица $\tilde{f}$ получится комплексной. Возьмите каждый её элемент по модулю, чтобы получить восстановленное изображение.

Проверьте реализацию с помощью юнит-теста:

```
$ ./run.py unittest wiener_filtering
```

Метрика PSNR (0.5 балла)

Что оценить качество восстановления изображения, будем использовать метрику PSNR (Peak Signal-to-Noise Ratio). Чем ближе восстановленное изображение к исходному, тем выше значение метрики:

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right).$$

Здесь MAX_I — это максимально возможное значение пикселя изображения (255 для 8 бит), а MSE (Mean Squared Error) — среднеквадратичное отклонение двух изображений.

Реализуйте функцию `compute_psnr`. Проверьте реализацию с помощью юнит-теста:

```
$ ./run.py unittest psnr
```

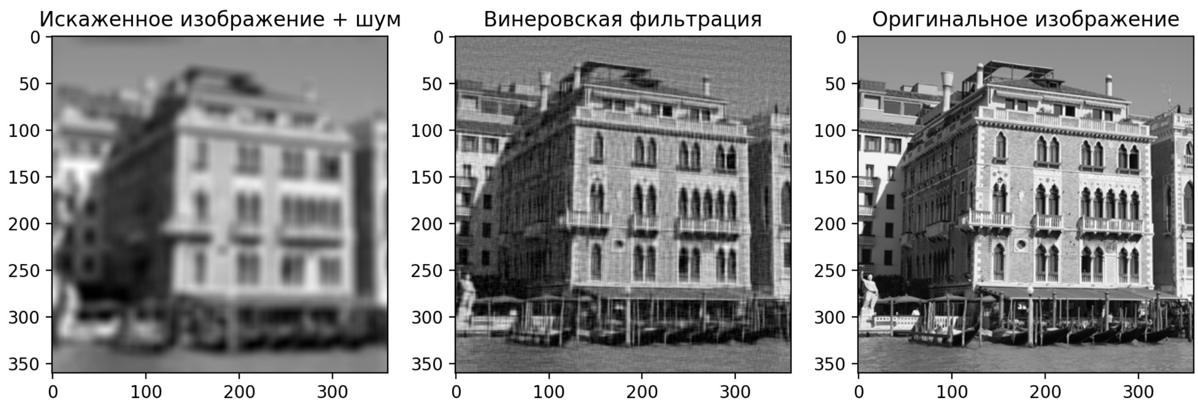
Подбор константы K (0.5 балла)

Требуется подобрать константу K сигнал/шум в выражении (7) и, применив фильтр Винера, восстановить изображение, размытое фильтром Гаусса размера 15 и с σ равным 5. В процессе размытия к изображению был добавлен Гауссовский шум.

Задание считается выполненным, если значение метрики PSNR увеличится хотя бы на 7 пунктов по сравнению с неотфильтрованным изображением. Для подбора значения константы K воспользуйтесь функцией `visualization.vis_wiener` и командой

```
$ ./visualization.py wiener
```

Результат восстановления изображения будет содержаться в `wiener_filtering_noisy.jpg`:



Найденную константу требуется указать в функции `wiener_filtering` в качестве значения по умолчанию для K. Проверьте реализацию с помощью юнит-теста:

```
$ ./run.py unittest filtering_constant
```

Список литературы

1. Гонсалес Р., Вудс Р. Цифровая обработка изображений

2. Восстановление расфокусированных и смазанных изображений
   https://habr.com/ru/post/136853/