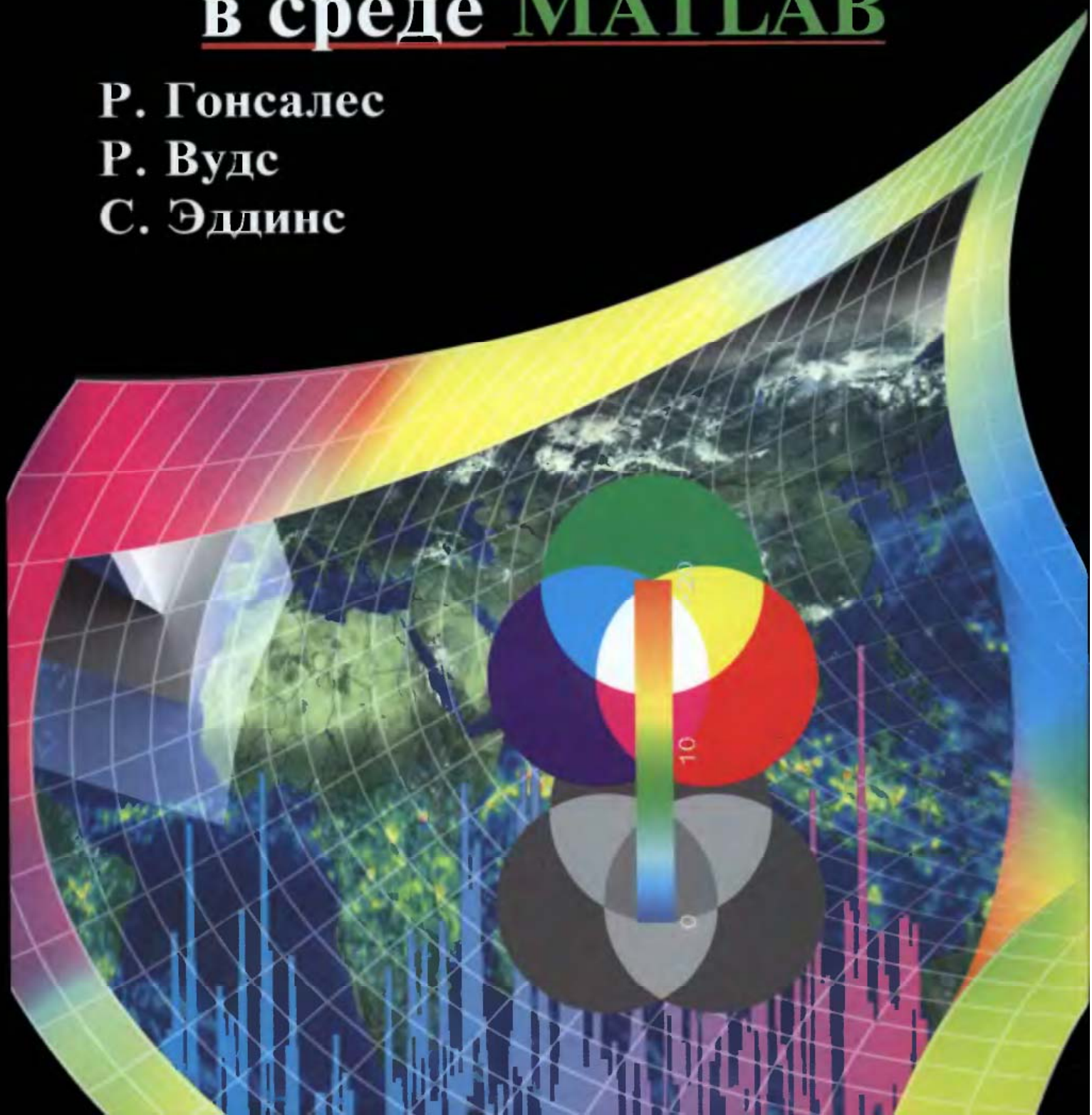
Цифровая обработка изображений в среде MATLAB

Р. Гонсалес
Р. Вудс
С. Эддинс



МИР
цифровой обработки

Р. ГОНСАЛЕС, Р. ВУДС
С. ЭДДИНС

Цифровая обработка изображений в среде MATLAB

Перевод с английского
В.В. Чепыжова

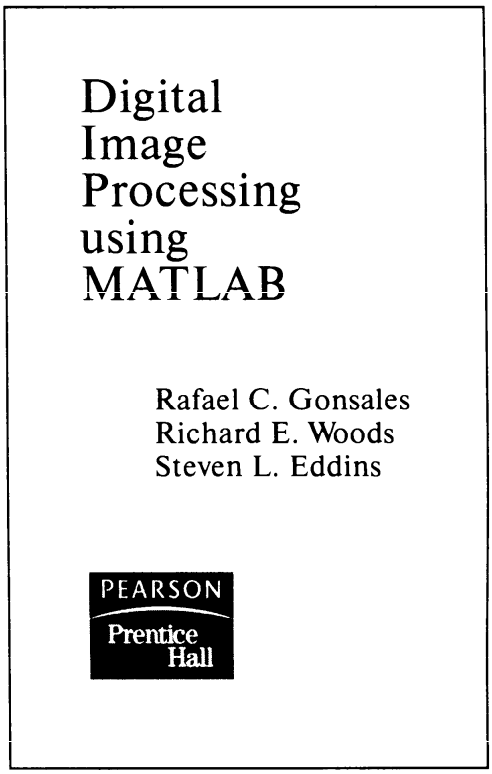
ТЕХНОСФЕРА
Москва
2006

Гонсалес Р., Вудс Р., Эддинс С.
Цифровая обработка изображений в среде MATLAB
Москва:
Техносфера, 2006. – 616с. ISBN 5-94836-092-X

Монография предназначена для тех, кто хочет в короткие сроки освоить методы обработки изображений с использованием пакета MATLAB.

Книга разбита на 12 глав, охватывающих самые важные области обработки изображений: градационные преобразования, линейную и нелинейную пространственную фильтрацию, вейвлеты, фильтрацию в частотной области, восстановление, регистрацию, сжатие, морфологическую обработку, сегментацию, представление и описание областей и границ изображений, а также распознавание объектов и обработку цветных изображений.

Книга будет полезна всем, кто хочет овладеть практическими навыками работы с изображениями, особенно специалистам по дистанционному зондированию, цифровому телевидению, компьютерной микроскопии, системам безопасности, программистам и дизайнерам.



ISBN 5-94836092-X
ISBN 0-13-008519-7 (англ.)

к выбранной форме. Заметим, однако, что этот сдвиг вправо был не столь радикальным, как на гистограмме, показанной на рис. 3.10, *г)*, которая соответствует очень слабому улучшению изображения, приведенного на рис. 3.10, *в)*. □

3.4. Пространственная фильтрация

Как уже было отмечено в § 3.1 и проиллюстрировано на рис. 3.1, окрестностная обработка изображений состоит из следующих действий:

(1) определение центральной точки (x, y);
(2) совершение операции, которая использует лишь значения пикселов в заранее оговоренной окрестности вокруг центральной точки;
(3) назначение результата этой операции «откликом» совершаемого процесса в *этой* точке;
(4) повторение всего процесса для каждой точки изображения.

В результате перемещения центральной точки образуются новые окрестности, отвечающие каждому пикселу изображения. Для описанной процедуры принято использовать термины *окрестностная обработка* и *пространственная фильтрация*, причем последний является более употребимым. Как будет объяснено в следующем параграфе, если операции, совершаемые над пикселами окрестности, являются линейными, то вся процедура называется *линейной пространственной фильтрацией* (также иногда используется термин *пространственная свертка*), в противном случае она называется *нелинейной пространственной фильтрацией.*

3.4.1. Линейная пространственная фильтрация

Понятие линейной фильтрации тесно связано с преобразованием Фурье при обработке сигналов в частотной области. Эта тематика будет рассматриваться в гл. 4. В настоящей главе рассматриваются операции фильтрации, непосредственно применяемые к пикселам изображения. Использование термина *линейная пространственная фильтрация* подчеркивает отличие этого процесса от *фильтрации в частотной области.*

Линейные операции, рассматриваемые в настоящей главе, состоят из умножения каждого пиксела окрестности на соответствующий коэффициент и суммирование этих произведений для получения результирующего *отклика* процесса в каждой точке (x, y). Если окрестность имеет размер $m \times n$, то потребуется mn коэффициентов. Эти коэффициенты сгруппированы в виде матрицы, которая называется *фильтром, маской, фильтрующей маской, ядром, шаблоном* или *окном*, причем первые три термина являются наиболее распространенными. По причинам, которые скоро станут известны, мы также будем использовать термины *сверточный фильтр, маска* или *ядро.*

Механизм линейной пространственной фильтрации проиллюстрирован на рис. 3.12. Процесс заключается в перемещении центра фильтрующей маски w от точки к точке изображения f. В каждой точке (x, y) откликом фильтра является сумма произведений коэффициентов фильтра и соответствующих пикселов окрестности, которые накрываются фильтрующей маской. Для маски размера $m \times n$

обычно предполагается, что $m = 2a+1$ и $n = 2b+1$, где a и b — неотрицательные целые числа, т. е. основное внимание уделяется маскам, имеющим нечетные размеры, причем наименьшим содержательным размером маски считается размер 3×3 (маска 1×1 исключается как тривиальная). Преимущественное обращение с масками нечетных размеров является вполне обоснованным, поскольку в этом случае у маски имеется выраженная центральная точка.

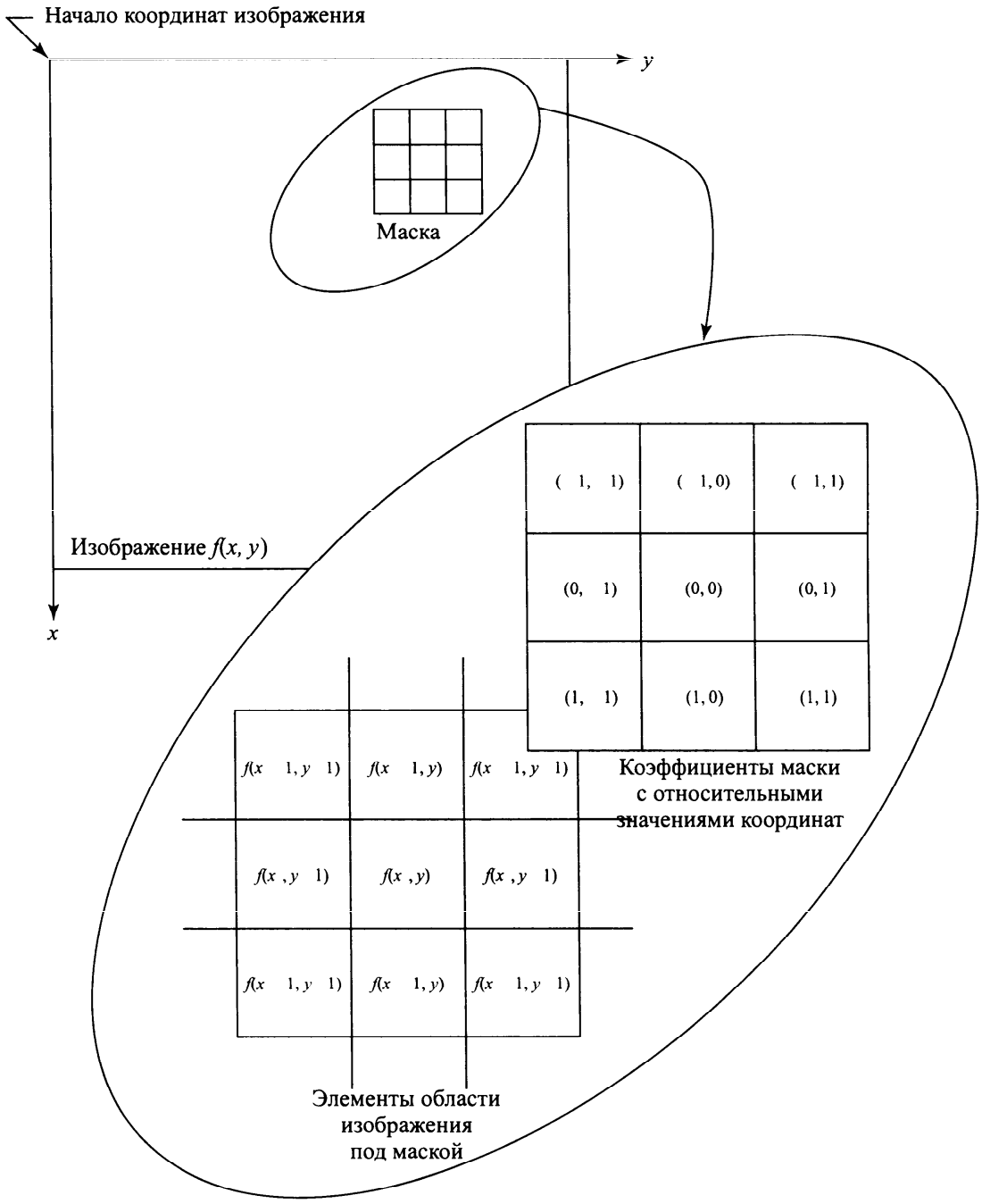


Рис. 3.12. Механизм линейной пространственной фильтрации. Увеличенный рисунок показывает маску 3×3 и соответствующий фрагмент изображения под ней, который несколько смещен для удобства прочтения формул

Имеется две тесно связанные концепции, которые необходимо хорошо понимать при совершении линейной пространственной фильтрации. Первая — это

корреляция, а вторая — *свертка*. Корреляция состоит в прохождении маски w по изображению f, показанному на рис. 3.12. С точки зрения механики процесса, свертка делается так же, но маску w надо повернуть на 180° перед прохождением по изображению f. Две эти концепции лучше всего объяснить на простых примерах.

На рис. 3.13, *а)* изображена одномерная функция f и маска w. Началом функции f считается ее самая левая точка. Чтобы совершить корреляцию этих двух функций, перемещаем w так, чтобы ее самая правая точка совпала с началом f, как показано на рис. 3.13, *б)*. Заметим, что имеются точки на обеих функциях, которые не перекрываются. Для решения этой проблемы предполагается, что функция f равна нулю везде, где это необходимо, чтобы гарантировать наличие соответствующих точек на f при прохождении над ней маски w. Такая ситуация проиллюстрирована на рис. 3.13, *в)*.

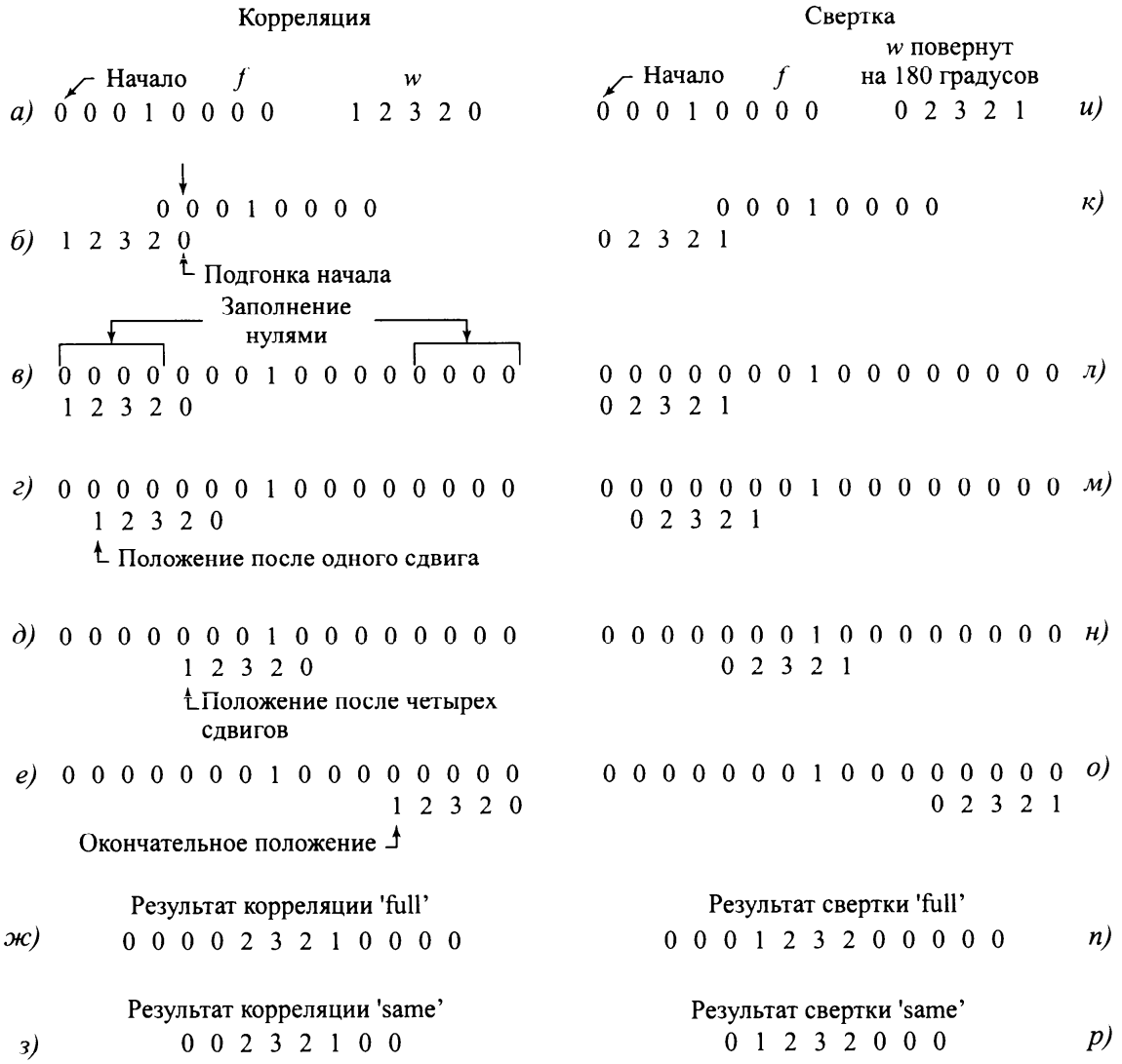


Рис. 3.13. Иллюстрация одномерной корреляции и свертки

Теперь все готово для совершения корреляции. Первым значением корреляции является сумма поэлементных произведений двух функций в положении на

рис. 3.13, *в)*. В этом случае сумма произведений равна 0. Затем маска w перемещается на один шаг вправо и процесс вычисления повторяется [рис. 3.13, *г)*]. Сумма произведений опять равна 0. После четырех сдвигов [рис. 3.13, *д)*] нам встретится первое ненулевое значение $2\times1 = 2$. Продолжая тем же манером до тех пор, пока w полностью пройдет f [последняя конфигурация показана на рис. 3.13, *е)*], мы получим результат, показанный на рис. 3.13, *ж)*. Эта строка чисел и является корреляцией w и f. Отметим, что если бы мы зафиксировали w, а перемещали бы f, то результат был бы иным, так что порядок операции здесь имеет значение.

Метка `'full'` над результатом корреляции на рис. 3.13, *ж)* является флагом (он будет обсуждаться позже), который предписывает применение корреляции с расширением изображения продемонстрированным выше методом. Пакет имеет и другую опцию, `'same'` [рис. 3.13, *з)*], когда вычисляется корреляция, размер которой совпадает с размером исходного изображения f. Эти вычисления также используют расширение нулями, но начальная позиция центральной точки маски (это точка с отметкой 3) совмещается с началом f. Последнее вычисление производится, когда центральная точка маски совмещается с концом f.

Для совершения свертки необходимо повернуть маску w на 180° и совместить ее самый правый конец с началом f [см. рис. 3.13, *к)*]. Затем процесс скольжения/вычисления совершается как при получении корреляции. Он проиллюстрирован на рис. 3.13 от *л)* до *о)*. Результаты свертки с флагами `'full'` и `'same'` показаны, соответственно, на рис. 3.13, *п)* и *р)*.

Функция f на рис. 3.13 является дискретным единичным импульсом, т. е. она равна 1 только в одной координате, а во всех остальных она равна 0. Из результатов на рис. 3.13, *п)* или *р)* видно, что свертка просто «копирует» w в то место, где был единичный импульс. Это простое свойство копирования (называемое *сдвигом*) является фундаментальной концепцией теории линейных систем, чем объясняется необходимость поворота одной из функций на 180° при выполнении операции свертки. Заметим, что перестановка порядка функций при свертке даст тот же самый результат, чего не происходит при корреляции. Если сдвигаемая функция является симметричной, то, очевидно, корреляция и свертка дают одинаковые результаты.

Представленные концепции легко обобщаются на двумерные функции, т. е. на изображения, что иллюстрируется на рис. 3.14. Начало находится в верхнем левом углу изображения $f(x, y)$ [см. рис. 2.1]. Для совершения корреляции надо поместить нижнюю правую точку $w(x, y)$ так, чтобы она совпала с началом $f(x, y)$, см. рис. 3.14, *в)*. Обратите внимание на использование расширения нулями, причина которого обсуждалась при рассмотрении рис. 3.13. Для совершения корреляции надо перемещать $w(x, y)$ по всем возможным положениям, так чтобы хоть один пиксел маски перекрывался с пикселами изображения $f(x, y)$. Результат корреляции `'full'` показан на рис. 3.14, *г)*. Чтобы получить корреляцию `'same'`, приведенную на рис. 3.14, *д)*, необходимо перемещать маску так, чтобы ее центр перекрывал исходное изображение $f(x, y)$.

Для совершения свертки необходимо сначала повернуть $w(x, y)$ на 180° и совершить ту же процедуру, что и при вычислении корреляции [см. рис. 3.14 от *е)* до *з)*]. Как и в обсуждавшемся выше одномерном примере, свертка дает одинако-

вый результат независимо от того, какая из двух функций подвергается перемещению. При нахождении корреляции порядок функций имеет значение. В пакете IPT при реализации этих процедур всегда перемещается фильтрующая маска. Заметим также, что результаты корреляции и свертки получаются друг из друга поворотом на 180°. В этом нет ничего удивительного, так как свертка — это не что иное, как корреляция с повернутой фильтрующей маской.

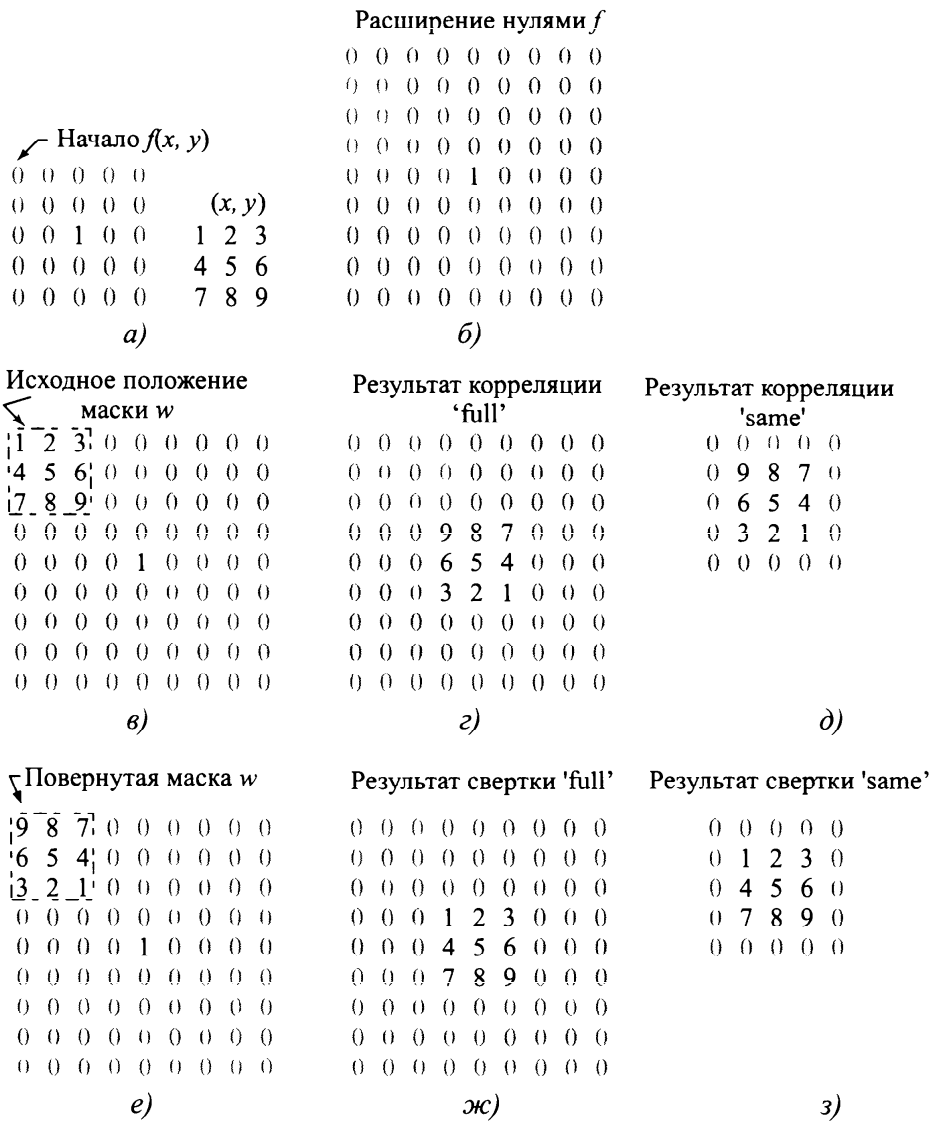


Рис. 3.14. Иллюстрация двумерной корреляции и свертки. Нули показаны серым цветом для лучшей наглядности

В пакете IPT *линейная* пространственная фильтрация реализована функцией `imfilter`, которая имеет следующий синтаксис:

```
>> g=imfilter(f, w, filtering_mode, boundary_options, size_options);
```

где `f` — это входное изображение, `w` — фильтрующая маска, `g` — результат фильтрации, а остальные параметры сведены в табл. 3.2. Параметр `filtering_mode` определяет, что совершает фильтр, корреляцию (`'corr'`) или свертку (`'conv'`). Опция `boundary_options` отвечает за расширение границ, причем размеры расширения определяются размерами фильтра. Подробнее этот параметр будет раз-

биратьcя в примере 3.7. Опция `size_options` — это либо `'full'`, либо `'same'`, смысл которых уже объяснялся на рис. 3.13 и 3.14.

Чаще всего функция `imfilter` применяется в виде команды

```
g = imfilter(f, w, 'replicate') .
```

Эта форма команды используется при реализации в IPT стандартных линейных пространственных фильтров, которые будут обсуждаться в § 3.5.1. Эти фильтры уже повернуты на 180°, поэтому можно делать процедуру корреляции, которая задана в функции `imfilter` по умолчанию. А как уже обсуждалось при изучении рис. 3.14, совершение корреляции с перевернутым фильтром эквивалентно свертке с исходным фильтром. Если фильтр симметричен относительно своего центра, то обе опции дают одинаковые результаты.

Таблица 3.2. Опции функции `imfilter`

Опции	Описание
Мода фильтрации	
`'corr'`	Фильтрация делается методом корреляции (см. рис. 3.13 и 3.14). Мода по умолчанию.
`'conv'`	Фильтрация делается методом свертки (см. рис. 3.13 и 3.14)
Граничные опции	
`P`	Границы изображения расширяются значением `P` (без апострофов). По умолчанию `P=0`.
`'replicate'`	Размер изображения увеличивается повторением величин на его боковых границах.
`'symmetric'`	Размер изображения увеличивается путем зеркального отражения через границы.
`'circular'`	Размер изображения увеличивается периодическим повторением двумерной функции.
Опции размера	
`'full'`	Выход имеет те же размеры, что и расширенное входное изображение.
`'same'`	Выход имеет те же размеры, что и вход. Это достигается с помощью ограничения перемещения центра фильтрующей маски точками, принадлежащими исходному изображению. (см. рис. 3.13 и 3.14). Опция по умолчанию.

При работе с фильтрами, которые не были предварительно перевернуты или были несимметричными, когда требуется построить свертку, можно поступать двумя способами. Один из них использует синтаксис

```
g = imfilter(f, w, 'conv', 'replicate').
```

Другой подход заключается в предварительной обработке маски w с помощью функции `rot90(w, 2)`[1], которая поворачивает w на 180°. После этого применяется команда `imfilter(f, w, 'replicate')`. Конечно, эти два шага можно записать в одной формуле. В результате получится изображение, размер которого совпадает с размером исходного (т. е. по умолчанию принята опция `'same'`, которая обсуждалась ранее).

[1]Функция `rot90(w, k)` поворачивает `w` на угол $k \times 90°$, где k — целое число.

Каждый элемент фильтрованного изображения вычисляется с использованием арифметики двойной точности с плавающей запятой, но в конце работы функция `imfilter` конвертирует выходное изображение в класс исходного. Значит, если `f` — это целый массив, то элементы обработанного массива, которые выходят за область целых чисел, будут обрезаться, а дробные величины — округляться. Если требуется результат с повышенной точностью, то необходимо предварительно перевести изображение в класс `duble` с помощью функций `im2double` или `double` перед применения `imfilter`.

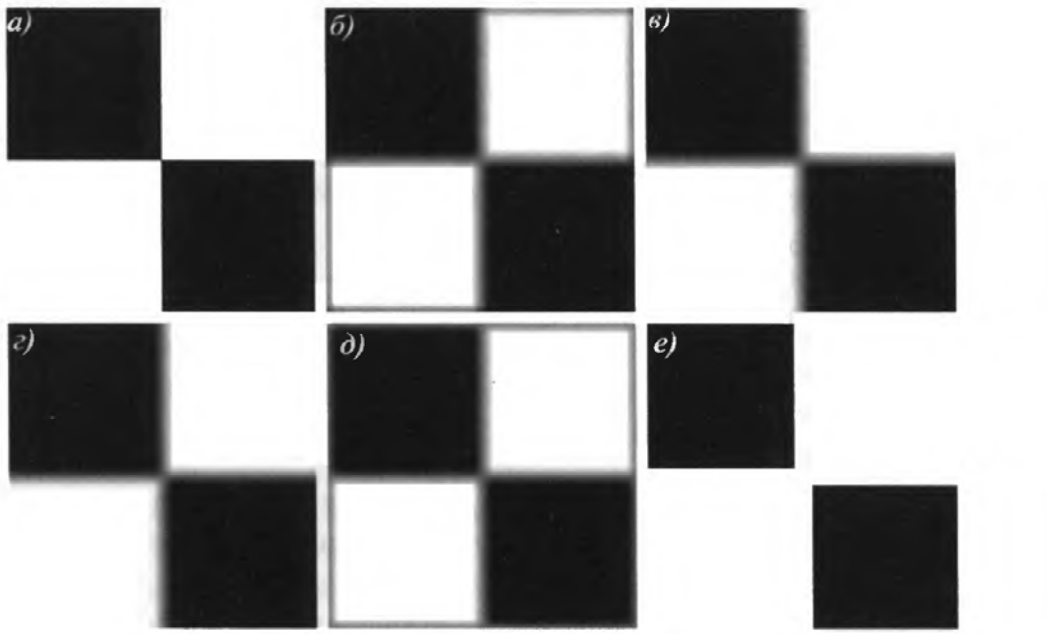


Рис. 3.15. *а)* Исходное изображение. *б)* Результат применения `imfilter` с нулевым продолжением. *в)* Результат с опцией `'replicate'`. *г)* Результат с опцией `'symmetric'`. *д)* Результат с опцией `'circular'`. *е)* Результат конвертации исходного изображения в класс `uint8` с последующей фильтрацией с опцией `'replicate'`. Везде использован фильтр размера 31×31 из одних единиц

Пример 3.7. *Применение функции* `imfilter`.
На рис. 3.15, *а)* дано изображение класса `double`, размера 512×512 пикселов. Рассмотрим простой фильтр размера 31×31

```
>>  w = ones(31);
```

который пропорционален фильтру усреднения. Мы не делим коэффициенты на 31×31, чтобы продемонстрировать в конце этого примера действие функции `imfilter` на изображениях класса `uint8`.

Свертка с фильтром w дает эффект размытия исходного изображения. Поскольку фильтр симметричен, можно использовать моду корреляции, заданную в `imfilter` по умолчанию. На рис. 3.15, *б)* приведен результат выполнения следующих действий:

```
>>  gd = imfilter(f, w);
>>  imshow(gd, [ ])
```

где использовалась граничная опция, принятая по умолчанию, т. е. расширение изображения нулями (черным цветом). Как и ожидалось, границы между белыми и черными областями стали размытыми. Тот же эффект имел место на границах изображения, к которым примыкают белые области, что вполне понятно в силу расширения границ черным цветом. С этой трудностью можно справиться, если использовать опцию `'replicate'`

```
>> gr = imfilter(f, w, 'replicate');
>> figure, imshow(gr, [ ])
```

Из рис. 3.15, *в)* видно, что теперь границы фильтрованного изображения выглядят так, как и ожидалось. Эквивалентный результат получается с опцией `'symmetric'`

```
>> gs = imfilter(f, w, 'symmetric');
>> figure, imshow(gs, [ ])
```

Результат показан на рис. 3.15, *г)*. Однако, если применить опцию `'cirular'`

```
>> gc = imfilter(f, w, 'cirular');
>> figure, imshow(gc, [ ])
```

(см. рис. 3.15, *д)*), то обнаружится та же проблема, что и при расширении нулями. Этому не стоит удивляться, ведь при периодическом повторении исходного изображения черные квадраты примыкают к белым.

Наконец, проиллюстрируем эффекты, происходящие из-за сохранения класса изображения функцией `imfilter`, который может привести к определенным проблемам, если не обращать на это внимание:

```
>> f8 = im2uint8(f);
>> g8r = imfilter(f8, w, 'replicate');
>> figure, imshow(g8r, [ ])
```

На рис. 3.15, *е)* изображен результат этих команд. Здесь при конвертации выходного изображения в класс `uint8` функцией `imfilter` произошло усечение части изображения. Причина этого состоит в том, что сумма коэффициентов фильтрации не принадлежит интервалу $[0, 1]$, в результате чего фильтрованные коэффициенты вышли за допустимые пределы области определения $[0, 255]$ данного класса. Чтобы избежать такой неприятности, следует делать нормировку коэффициентов фильтра, чтобы их сумма принадлежала интервалу $[0, 1]$ (в рассмотренном примере для этого необходимо разделить коэффициенты на 31^2, тогда их сумма будет равна 1), или переводить изображение в класс `double`. Отметим, однако, что даже во втором случае данные придется где-то перенормировать для приведения их к нужному формату (например, при сохранении изображения на диске). В любом случае, следует всегда помнить об области значений данных, чтобы избегать неожиданных результатов. □

3.4.2. Нелинейная пространственная фильтрация

Нелинейная пространственная фильтрация также основана на окрестностных операциях, причем механизм определения окрестности размера $m \times n$ и скольжения ее центра по изображению является таким же, как и в линейной фильтрации, описанной в предыдущем параграфе. Однако там, где линейная фильтрация использует сумму произведений (т. е. линейную операцию), нелинейная фильтрация основана (что следует из ее названия) на нелинейных операциях, совершаемых над пикселами текущей окрестности. Например, если положить, что отклик фильтра в каждой центральной точке равен максимальному значению в ее окрестности, то это определит нелинейную операцию фильтрации. Другое фундаментальное отличие состоит в том, что концепция маски не превалирует в нелинейных процессах. Идея фильтрации остается, но сам «фильтр» следует представлять себе в виде нелинейной функции, которая применяется к пикселам окрестности, и ее отклик состоит из отклика операции, примененной к центральному пикселу окрестности.

В пакете предусмотрены две функции для совершения общей нелинейной фильтрации: `nlfilter` и `colfilt`. Первая из них совершает операции непосредственно в матричной форме, а функция `colfilt` организует данные в форме столбцов. Хотя `colfilt` требует большего объема памяти, она выполняется существенно быстрее, чем `nlfilter`. В большинстве приложений обработки изображений скорость вычисления является первостепенным фактором, поэтому функция `colfilt` является более предпочтительной при реализации общей нелинейной пространственной фильтрации.

Для заданного входного изображения `f` размера $M \times N$ и для заданной окрестности размера $m \times n$ функция `colfilt` строит матрицу, назовем ее `A`, максимального размера $mn \times MN$[2], в которой каждый столбец соответствует пикселам, окруженным окрестностью с центром в некоторой точке изображения. Например, первый столбец соответствует пикселам, окруженным окрестностью, центр которой расположен в самом левом верхнем углу изображения `f`. Все необходимые расширения выполняются функцией `colfilt` (с помощью нулевых добавок).

Синтаксис функции `colfilt` имеет вид

```
g = colfilt(f, [m, n], 'sliding', @fun, parameters) ,
```

где, как и раньше, `m` и `n` — это размеры области фильтра, `'sliding'` обозначает, что процесс производится скольжением области $m \times n$ от пиксела к пикселу по входному изображению `f`, `@fun` обозначает ссылку на функцию `fun`, а `parameters` — это параметры (разделенные запятыми), которые могут потребоваться функции `fun`. Символ `@` называется *дескриптором функции*, который является особым типом данных MATLAB, содержащим информацию, которая используется при вызове функции. Как скоро будет видно, эта конструкция является весьма плодотворной.

В силу организации матрицы `A`, функция `fun` должна обращаться к каждому столбцу этой матрицы индивидуально, возвращая вектор-строку `v`, в которой

[2]Матрица `A` всегда имеет mn строк, но число столбцов может меняться в зависимости от размера входного изображения. Выбор размера производится функцией `colfilt` автоматически.

записан результат для всех столбцов. В k-ом элементе вектора v стоит результат применения операции fun к k-ому столбцу матрицы A. Поскольку матрица A может иметь до MN столбцов, максимальный размер v равен $1 \times MN$.

Обсуждавшаяся в предыдущих параграфах линейная фильтрация имеет средства для расширения изображений, чтобы разрешать граничные проблемы, присущие пространственной фильтрации. Однако при использовании colfilt необходимо заранее расширить входное изображение до его фильтрации. Для этого служит функция padarray, которая в двумерном случае имеет синтаксис

```
fp = padarray(f, [r c], method, direction) ,
```

где f — входное изображение, fp — расширенное изображение, [r c] дает число добавляемых к f строк и столбцов, а параметры method и direction объясняются в табл. 3.3. Например, если изображение f = [1 2; 3 4], то команда

```
>>  fp = padarray(f, [3 2], 'replicate', 'post')
```

дает результат

```
fp =
   1   2   2   2
   3   4   4   4
   3   4   4   4
   3   4   4   4
   3   4   4   4 .
```

Если среди аргументов отсутствует direction, то значение по умолчанию есть 'both', а если отсутствует аргумент method, то расширение по умолчанию делается нулями. Если оба параметра опущены, то делается расширение нулями со всех сторон изображения. В конце вычислений изображение обрезается до исходного размера.

Таблица 3.3. Опции функции padarray

Опции	Описание
Method	
'symmetric'	Размер изображения увеличивается с помощью его зеркального отображения через границы.
'replicate'	Размер изображения увеличивается с помощью продолжения приграничными значениями.
'circular'	Расширение производится путем периодического повторения исходного изображения.
Direction	
'pre'	Расширять перед первыми элементами по каждой размерности.
'post'	Расширять после последних элементов по каждой размерности.
'both'	Расширить в обе стороны по каждой размерности.

Пример 3.8. *Использование функции* `colfilt` *при реализации нелинейной пространственной фильтрации.*
В качестве иллюстрации применения функции `colfilt` мы реализуем нелинейный фильтр, отклик которого в любой точке совпадает со средним геометрическим яркости пикселов в окрестности с центром в данной точке. Геометрическим средним окрестности размера $m \times n$ называется произведение величин яркости всех пикселов окрестности, возведенное в степень $1/mn$. Сначала построим нелинейную функцию, которую мы назовем `gmean`[3]:

```
function v = gmean(A)
mn = size(A, 1); % The length of the columns of A is always mn.
v = prod(A, 1).^(1/mn);
```

Для подавления граничных эффектов мы расширим изображение, например, с помощью опции `'replicate'` в функции `padarray`:

```
>>  f = padarray(f, [m n], 'replicate');
```

Наконец, вызываем функцию `colfilt`

```
>>  g = colfilt(f, [m n], 'sliding', @gmean);
```

В этих действиях имеется несколько важных моментов. Прежде всего отметим, что несмотря на то, что матрица `A` является частью аргументов функции `gmean`, сама она не включается в список аргументов функции `colfilt`. Эта матрица автоматически передается в `gmean` из `colfilt` посредством дескриптора функции. Кроме того, поскольку матрица `A` обрабатывается в `colfilt` автоматически, число столбцов в `A` является переменным (но, как уже подчеркивалось, число строк, т. е. длина столбца в `A`, всегда равна `mn`). Следовательно, размер `A` должен вычисляться каждый раз при вызове функции аргумента функцией `colfilt`. Процесс фильтрации в нашем случае заключается в вычислении произведения всех пикселов окрестности и возведения результата в степень $1/mn$. Для каждой пары координат (x, y) результат фильтрации будет записан в соответствующий элемент вектора `v`. Функция, обозначенная дескриптором `@`, может быть любой функцией, которую можно вызывать из того места, где этот дескриптор был создан. Ключевое требование заключается в том, что нелинейная функция оперирует над столбцами матрицы `A` и возвращает вектор-строку, содержащую результат для каждого индивидуального столбца. Затем функция `colfilt` принимает эти результаты и группирует их в виде соответствующего выходного изображения `g`. □

Некоторые часто встречающиеся нелинейные фильтры можно также построить на основе других функций MATLAB и IPT, например, `imfilter` и `ordfilt2` (см. § 3.5.2). В частности, функция `spfilt` из § 3.5 реализует фильтр геометрического среднего из примера 3.5 с помощью функции `imfilter` и стандартных функций `log` и `exp` системы MATLAB. При этом производительность вычислений обычно повышается, а объем требуемой памяти становится меньше, чем

[3]Функция `prod(A)` возвращает произведение элементов `A`, а функция `prod(A, dim)` вычисляет произведение элементов `A` *вдоль* размерности `dim`.

у colfilt. Тем не менее, функция colfilt остается хорошим выбором при выполнении операций нелинейной фильтрации, для которых нет альтернативных реализаций.

3.5. Стандартные пространственные фильтры из пакета IPT

В этом параграфе обсуждаются линейные и нелинейные пространственные фильтры, которые поддерживаются в пакете IPT. Дополнительные нелинейные фильтры будут реализованы в конце § 5.3.

3.5.1. Линейные пространственные фильтры

В пакете IPT имеются некоторые стандартные двумерные линейные пространственные фильтры, которые можно получить из функции `fspecial`, которая генерирует маску фильтра `w` при выполнении команды

```
w = fspecial('type', parameters),
```

где `'type'` обозначает тип фильтра, а в аргументах `parameters` задаются параметры выбранного фильтра. Пространственные фильтры, получаемые этой командой, приведены в табл. 3.4 вместе с соответствующими параметрами каждого фильтра.

Пример 3.9. *Использование функции* `imfilter`.
Проиллюстрируем использование `fspecial` и `imfilter` при улучшении изображения фильтром Лапласа. Оператор Лапласа изображения $f(x, y)$ обозначается $\nabla^2 f(x, y)$ и задается формулой

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}.$$

В качестве численных приближений вторых производных часто используются выражения

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

и

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y),$$

поэтому

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y).$$

Это выражение можно применить в любой точке (x, y) изображения, сделав свертку со следующей пространственной маской:

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}.$$

Таблица 3.4. Пространственные фильтры функции `fspecial`

Тип	Синтаксис и параметры
`'average'`	`fspecial('average', [r c])`. Прямоугольный усредняющий фильтр размера `r`×`c`. По умолчанию 3×3. Одно число на месте `[r c]` означает квадратный фильтр.
`'disk'`	`fspecial('disk', r)`. Круговой усредняющий фильтр (внутри квадрата со стороной `2r+1`) радиуса `r`. По умолчанию `r = 5`.
`'gaussian'`	`fspecial('gaussian', [r c], sig)`. Низкочастотный гауссов фильтр размера `r`×`c` со стандартным (положительным) отклонением `sig`. Значения по умолчанию 3×3 и `0.5`. Одно число на месте `[r c]` означает квадратный фильтр.
`'laplacian'`	`fspecial('laplacian', alpha)`. Фильтр Лапласа 3×3, форма которого задается параметром `alpha` из интервала $[0, 1]$. По умолчанию `alpha = 0.5`.
`'log'`	`fspecial('log', [r c], sig)`. Лаплас от гауссова фильтра (LoG) размера `r`×`c` со стандартным (положительным) отклонением `sig`. Значения по умолчанию 5×5 и `0.5`. Одно число на месте `[r c]` означает квадратный фильтр.
`'motion'`	`fspecial('motion', len, theta)`. Выдает фильтр, который, будучи свернутым с изображением, приближает линейное перемещение (видеокамеры по отношению к изображению) на `len` пикселов. Направление перемещения задается углом `theta`, который измеряется в градусах от горизонтали против часовой стрелки. Значения по умолчанию `9` и `0`, что соответствует перемещению на 9 пикселов в горизонтальном направлении.
`'prewitt'`	`fspecial('prewitt')`. Выдает 3×3 маску Превитта `wv`, которая аппроксимирует вертикальный градиент. Маску горизонтального градиента можно получить, транспонировав результат: `wh = wv'`.
`'sobel'`	`fspecial('sobel')`. Выдает 3×3 маску Собела `sv`, которая аппроксимирует вертикальный градиент. Маску горизонтального градиента можно получить, транспонировав результат: `sh = sv'`.
`'unsharp'`	`fspecial('unsharp', alpha)`. Выдает 3×3 маску нечеткого фильтра. Параметр `alpha` контролирует форму, он должен быть не меньше 0 и не больше `1.0`. По умолчанию `alpha = 0.2`.

Альтернативное приближение вторых производных учитывает значения диагональных элементов и дает маску

$$\begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}.$$

Обе производные иногда определяются с противоположным знаком, что приводит к смене знаков в приведенных выше формулах для масок.

Улучшение изображений с помощью оператора Лапласа производится по формуле

$$g(x, y) = f(x, y) + c\nabla^2 f(x, y),$$

где $f(x, y)$ — это исходное изображение, $g(x, y)$ — улучшенное изображение, а параметр c равен 1, если центральный коэффициент маски положителен, и $c = -1$ в противном случае (см. [Gonzalez, Woods, 2002]). Поскольку оператор Лапласа является дифференциальным, он повышает резкость изображения, но переводит области с постоянными значениями яркости в 0. Добавление исходного изображения восстанавливает тональность уровней таких областей.

Функция `fspecial('laplacian', alpha)` реализует более сложную маску Лапласа:

$$\begin{matrix} \frac{\alpha}{1+\alpha} & \frac{1-\alpha}{1+\alpha} & \frac{\alpha}{1+\alpha} \\ \frac{1-\alpha}{1+\alpha} & \frac{-4}{1+\alpha} & \frac{1-\alpha}{1+\alpha} \\ \frac{\alpha}{1+\alpha} & \frac{1-\alpha}{1+\alpha} & \frac{\alpha}{1+\alpha} \end{matrix},$$

которая позволяет тоньше подстраивать результат. Однако преобладающее использование оператора Лапласа основано на приведенных выше двух масках.

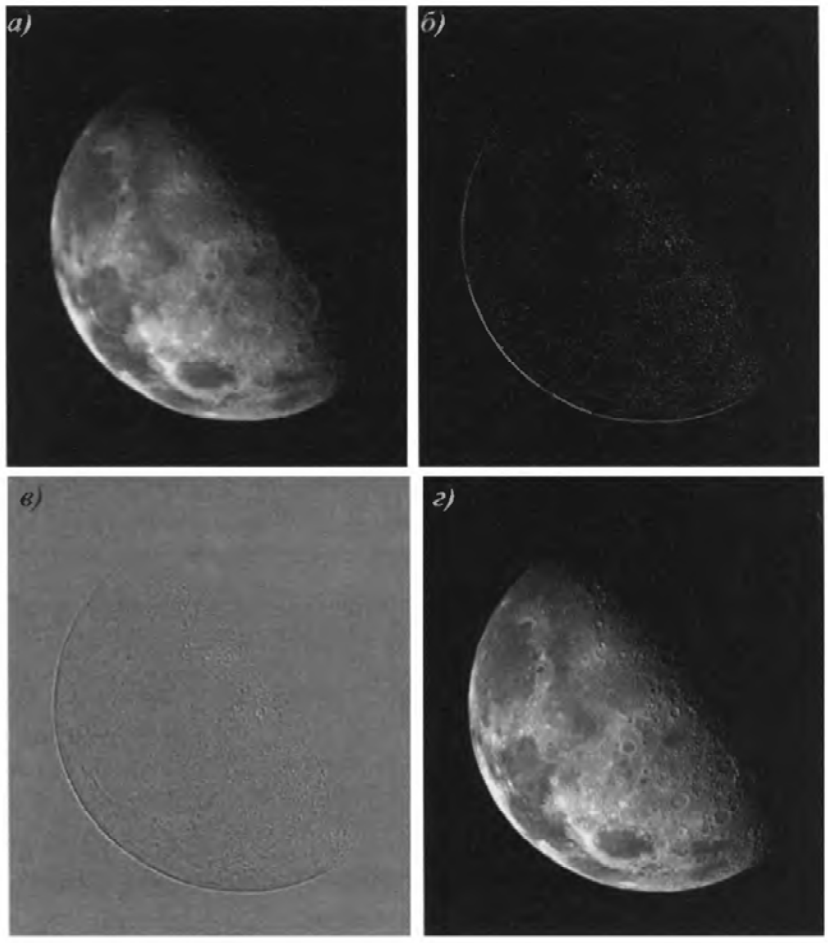


Рис. 3.16. *а)* Снимок Северного полюса Луны. *б)* Изображение, отфильтрованное лапласианом в формате `uint8`. *в)* Изображение, отфильтрованное лапласианом в формате `double`. *г)* Улучшение изображения вычитанием *б)* из *а)*. (Исходное изображение предоставлено Агентством NASA.)

Теперь рассмотрим улучшение изображения на рис. 3.16, *а)* с помощью фильтра Лапласа, который часто называется лапласианом. На этой фотографии дан несколько расплывчатый снимок Северного полюса Луны. Улучшение в этом случае заключается в подчеркивании перепадов уровней на изображении при сохранении, насколько это возможно, областей постоянной тональности. Сначала мы генерируем и отображаем фильтр Лапласа:

```
>>  w = fspecial('laplacian', 0);
w =
```

```
0.0000   1.0000  0.0000
1.0000  -4.0000  1.0000
0.0000   1.0000  0.0000
```

Заметим, что фильтр принадлежит классу `duoble`, и параметр формы `alpha = 1` приводит к описанному выше фильтру. Такую форму можно легко задать вручную

```
>>  w = [0 1 0; 1 -4 1; 0 1 0];
```

Теперь применим фильтр `w` к изображению `f`, которое имеет класс `uint8`:

```
>>  g1 = imfilter(f, w, 'replicate');
>>  imshow(g1, [ ])
```

На рис. 3.16, *б)* приведен результат этих команд. Этот результат представляется правдоподобным, но имеется одна проблема: все его пикселы положительны. В силу отрицательности центрального коэффициента фильтра можно ожидать появления фильтрованного изображения с отрицательными значениями пикселов. Однако в нашем случае исходное изображение было класса `uint8`, а, как уже говорилось в предыдущем параграфе, фильтрация функцией `imfilter` производит на выходе изображение того же класса, что было на входе, поэтому отрицательные величины будут обрезаны. Чтобы обойти эту проблему, следует преобразовать изображение `f` в класс `double` перед фильтрацией:

```
>>  f2 = im2double(f);
>>  g2 = imfilter(f2, w, 'replicate');
>>  imshow(g2, [ ])
```

Результат, показанный на рис. 3.16, *в)*, является правильно обработанным изображением при фильтрации с помощью лапласиана.

Наконец, для восстановления тонов областей, потерянных при выполнения фильтрации лапласианом, следует вычесть (напомним, что центральный коэффициент фильтра отрицателен) отфильтрованное изображение из исходного:

```
>>  g = f2 - g2;
>>  imshow(g)
```

Окончательный результат приведен на рис. 3.16, *г)*. Видно, насколько улучшенное изображение является более резким по сравнению с исходным. □

Пример 3.10. *Подбор параметров фильтров и сравнение разных техник улучшения изображений.*

Задача улучшения изображений часто требует подбора параметров фильтров из имеющегося набора. Лапласиан является хорошим примером. В пакете имеется фильтр-лапласиан 3×3 с числом −4 в центре. Как правило, еще большую резкость изображения можно получить с лапласианом, в центре которого стоит число −8, окруженное со всех сторон 1 (см. обсуждение выше). Целью следующего примера является ручная реализация этого фильтра для сравнения результатов, которые получаются с помощью двух форм фильтра лапласиана. Имеем следующую цепочку команд:

```
>> f = imread('moon.tif');
>> w4 = fspecial('laplacian', 0); % Same as Example 3.9.
>> w8 = [1 1 1; 1 -8 1; 1 1 1];
>> f = im2double(f);
>> g4 = f - imfilter(f, w4, 'replicate');
>> g8 = f - imfilter(f, w8, 'replicate');
>> imshow(f)
>> figure; imshow(g4)
>> figure; imshow(g8)
```

На рис. 3.17, *а)* еще раз для удобства приведено исходное изображение `f`. На рис. 3.17, *б)* дано `g4`, которое совпадает с рис. 3.16, *г)*, а на рис. 3.17, *б)* изображено `g8`. Как и ожидалось, оно еще более четкое, чем 3.17, *б)*. □

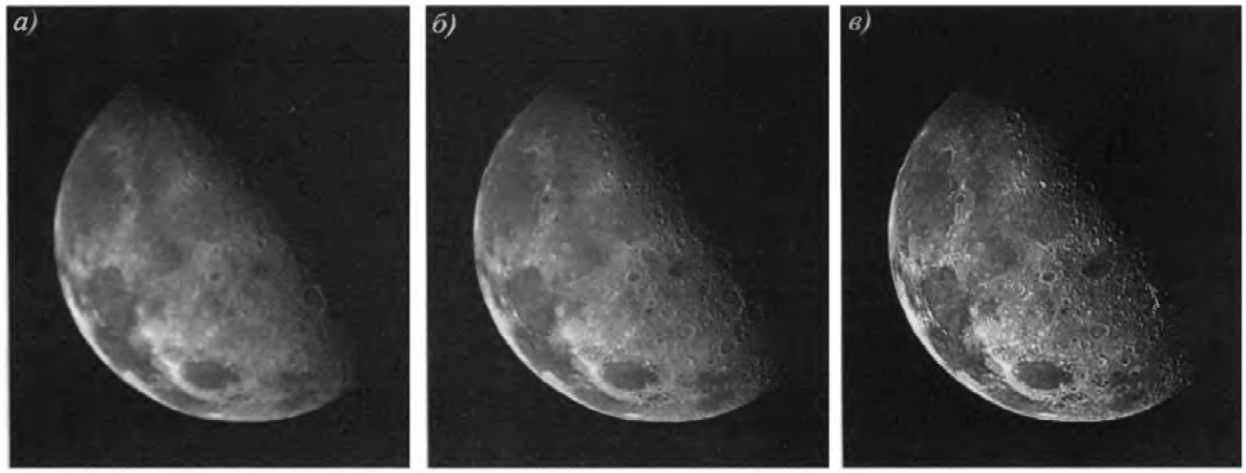


Рис. 3.17. *а)* Снимок Северного полюса Луны. *б)* Изображение, улучшенное лапласианом с -4 в центре. *в)* Изображение, улучшенное лапласианом с -8 в центре

3.5.2. Нелинейные пространственные фильтры

Наиболее употребительные нелинейные фильтры порождаются в пакете IPT функцией `ordfilt2`, которая строит *фильтры порядковых статистик* (их также называют *ранговыми фильтрами*). Отклики этих нелинейных пространственных фильтров основаны на предварительном упорядочении (ранжировании) пикселов изображения из текущей окрестности, после чего центральному пикселу присваивается значение, определенное в результате данного упорядочения. В этом параграфе все внимание будет сосредоточено на фильтрах, получающихся при помощи функции `ordfilt2`. Несколько дополнительных нелинейных фильтров будут рассмотрены в § 5.3.

Синтаксис функции `ordfilt2` имеет следующий вид:

```
g = ordfilt2(f, order, domain).
```

Эта функция создает выходное изображение `g`, заменяя каждый элемент `f` на элемент номер `order` в упорядоченной последовательности ненулевых элементов из окрестности, заданной параметром `domain`. Здесь `domain` — это матрица $m \times n$,

состоящая из нулей и единиц, которые обозначают позиции пикселов, участвующие в вычислениях. В этом смысле матрица `domain` действует как маска. Пикселы окрестности, которым соответствуют нули в `domain`, не участвуют в вычислениях. Например, чтобы реализовать *фильтр минимума* (порядка 1) размера $m \times n$, применяется команда

```
g = ordfilt2(f, 1, ones(m, n)).
```

В такой записи `order = 1` означает первый элемент в упорядоченной последовательности из mn пикселов, а `ones(m, n)` — это матрица $m \times n$ из одних единиц, указывающая на то, что все элементы окрестности участвуют в вычислении отклика фильтра.

Используя терминологию статистик, фильтр минимума (первый элемент упорядоченной по возрастанию последовательности) — это нулевой процентиль. Аналогично, сотый процентиль — это последний элемент упорядоченной последовательности, имеющий номер mn. Он соответствует *фильтру максимума*, который реализуется командой

```
g = ordfilt2(f, m*n, ones(m, n)).
```

Самым известным фильтром порядковых статистик в цифровой обработке изображений является *медианный*[1] *фильтр*, который соответствует 50-ому процентилю. Можно использовать функцию `median` в `ordfilt2` для построения этого фильтра

```
g = ordfilt2(f, median(1:m*n), ones(m, n)).
```

Здесь `median(1:m*n)` — это медиана упорядоченной последовательности чисел $1, 2, \ldots, mn$. Функция `median` имеет следующий общий синтаксис:

```
v = median(A, dim),
```

где `v` — это вектор, элементы которого образуют медиану `A` вдоль размерности `dim`. Например, если `dim = 1`, то каждый элемент `v` — это медиана элементов вдоль соответствующих столбцов матрицы `A`.

В силу практической важности медианного фильтра в пакете IPT предусмотрена специальная реализация этого фильтра:

```
g = medfilt2(f, [m, n], padopt),
```

где пара `[m, n]` задает окрестность размера $m \times n$, по которой вычисляется медиана, а параметр `padopt` специфицирует три возможные опции расширения границ изображения: опция по умолчанию `'zeros'` с нулевым расширением, `'symmetric'`, при которой изображение `f` расширяется путем его зеркального отражения через границы, и `'indexed'`, при которой `f` расширяется значением 1, если `f` имеет класс `double` и значением 0 в противном случае. По умолчанию эта функция имеет вид

```
g = medfilt2(f),
```

[1] Напомним, что медиана некоторого множества величин — это такое число ξ, что половина этих величин не больше ξ, а другая половина — не меньше ξ .

при этом используется окрестность 3×3 для вычисления медианы, и входное изображение расширяется нулевыми значениями пикселов.

Пример 3.11. *Медианная фильтрация с функцией* `medfilt`.
Медианная фильтрация весьма эффективна при удалении с изображений импульсных шумов, которые иногда называют шумами «соль и перец». Борьбу с различными шумами мы будем обсуждать подробно в гл. 5, однако здесь уместно проиллюстрировать эти действия с помощью медианного фильтра.

На рис. 3.18, *а)* представлен рентгеновский снимок `f` печатной платы, полученный на стадии автоматического контроля продукции производства. На рисунке 3.18, *б)* дано то же изображение, сильно подпорченное шумом «соль и перец», при котором белые и черные точки имеют вероятность появления 0.2. Этот шум был искусственно сгенерирован функцией `imnoise`, которая будет подробно обсуждаться в § 5.2.1:

```
>>  fn = imnoise(f, 'salt & paper', 0.2);
```

На рис. 3.18, *в)* дан результат медианной фильтрации этого искаженного изображения при выполнении команды

```
>>  gm = medfilt2(fn);
```

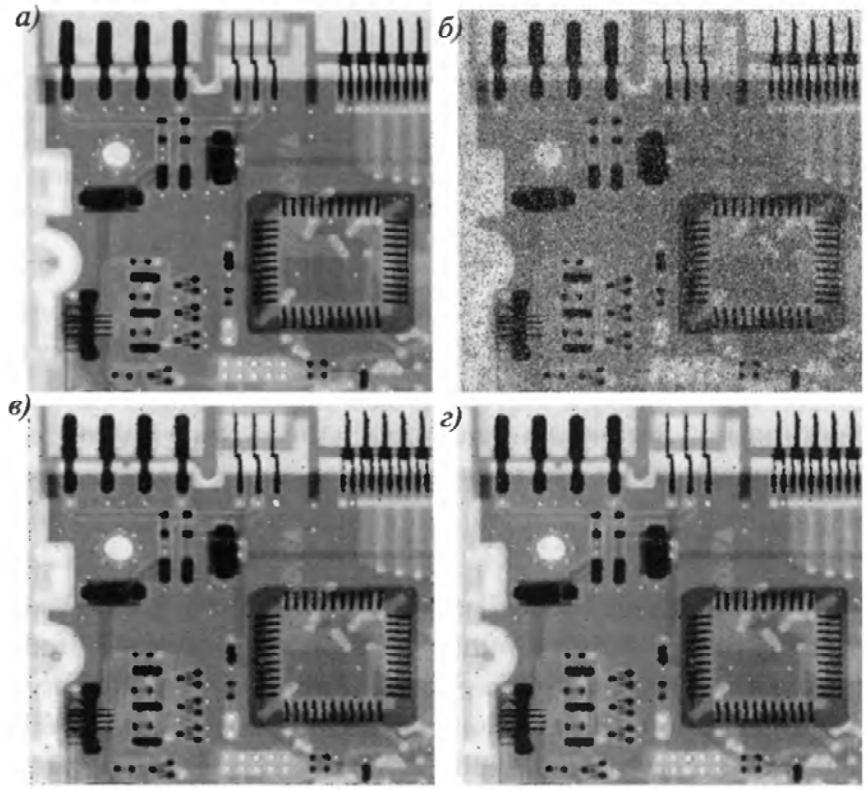


Рис. 3.18. Медианная фильтрация. *а)* Рентгеновский снимок. *б)* Изображение, испорченное шумом «соль и перец». *в)* Результат медианной фильтрации функцией `medfilt2` с опциями, принятыми по умолчанию. *г)* Результат медианной фильтрации с опцией расширения границ `'symmetric'`. Обратите внимание на различие поведения границ изображений *в)* и *г)*

При рассмотренном уровне шумов на рис. 3.18, *б)* медианный фильтр дает хорошие результаты при использовании его настроек по умолчанию. Обратите, однако, внимание на черные пятнышки на границах отфильтрованного изображения. Это произошло из-за расширения границ изображения черными пикселами (напомним, что по умолчанию изображение продолжается нулевыми, т. е. черными пикселами). Такого рода артефакты часто можно избежать, используя опцию `'symmetric'`:

```
>> gms = medfilt2(fn, 'symmetric');
```

Результат, изображенный на рис. 3.18, *г)*, похож на рис. 3.18, *в)*, но эффект почернения границ на нем не проявляется. □

Выводы

В этой главе, помимо обсуждения методов улучшения изображения, были заложены основы многих концепций, которые будут изучаться в следующих главах. Например, пространственная фильтрация снова встретится в гл. 5 в связи с поворотами изображений, где мы еще раз подробно рассмотрим задачу удаления шумов и приведем функции MATLAB, моделирующие различные шумы. Некоторые пространственные маски лишь вскользь упоминались в этой главе, и они еще будут подробным образом изучаться в гл. 10 при обнаружении границ в задачах сегментации изображений. Концепции свертки и корреляции будут рассматриваться вновь в гл. 4 применительно к частотной области. Базовые идеи обработки маской при использовании различных методов пространственной фильтрации будут часто встречаться в дальнейшем. По ходу изложения нового материала будет продолжено обсуждение вопросов эффективной реализации пространственной фильтрации в среде MATLAB.