CHAPTER 24

Linear Image Processing
Линейная Обработка Изображения

Linear image processing is based on the same two techniques as conventional DSP: *convolution* and *Fourier analysis*. Convolution is the more important of these two, since images have their information encoded in the spatial domain rather than the frequency domain. Linear filtering can improve images in many ways: sharpening the edges of objects, reducing random noise, correcting for unequal illumination, deconvolution to correct for blur and motion, etc. These procedures are carried out by convolving the original image with an appropriate filter kernel, producing the filtered image. A serious problem with image convolution is the enormous number of calculations that need to be performed, often resulting in unacceptably long execution times. This chapter presents strategies for designing filter kernels for various image processing tasks. Two important techniques for reducing the execution time are also described: *convolution by separability* and *FFT convolution*.

Линейная обработка изображения основана на тех же самых двух методах как обычный ЦОС: *свертка* и *анализ Фурье*. Свертка более важна из эти двух, так как информация изображения кодируются в пространственной области(пространственном домене) скорее чем частотном домене. Линейная фильтрация может улучшать изображения многими способами: заостряя грани объектов, приводя случайный шум, исправляя неравномерное освещения, деконволюцию, чтобы исправить пятна и движение, и т.д. Эти процедуры выполнены, свертывая первоначальное изображение с ядром соответствующего фильтра, производя фильтрованное изображение. Серьезная проблема со сверткой изображения - огромное число вычислений, которые должны быть выполнены, часто приводя к неприемлемо длинным временам выполнения. Эта глава представляет стратегии для проектирования ядер фильтров для различных задач обработки изображения. Два важных метода для сокращения времени выполнения также описаны: *свертка отделимостью* и *свертка БПФ*.

Convolution
Конволюция(Свертка)

Image convolution works in the same way as one-dimensional convolution. For instance, images can be viewed as a summation of *impulses*, i.e., scaled and shifted delta functions. Likewise, *linear systems* are characterized by how they respond to impulses; that is, by their *impulse responses*. As you should expect, the output image from a system is equal to the input image *convolved* with the system's impulse response.

Свертка изображения работает таким же образом как одномерная свертка. Например, изображения могут быть просмотрены как суммирование *импульсов*, то есть, масштабирование и сдвинутые дельта функции. Аналогично, *линейные системы* характеризованы тем, как они отвечают на импульсы; то есть их *импульсными передаточными функциями*. Поскольку Вы должны ожидать, изображение выхода от системы равно входному изображению, свернутому с импульсной передаточной функцией системы.

The two-dimensional delta function is an image composed of all zeros, except for a single pixel at: *row* = 0, *column* = 0, which has a value of *one*. For now, assume that the row and column indexes can have both positive and negative values, such that the *one* is centered in a vast sea of

zeros. When the delta function is passed through a linear system, the single nonzero point will be changed into some other two-dimensional pattern. Since the only thing that can happen to a point is that it *spreads out*, the impulse response is often called the **point spread function (PSF)** in image processing jargon.

Двумерная дельта функция - изображение, составленное из всех нулей, если бы не единственный(отдельный) пиксел в: *строке* = 0, *столбце* = 0, который имеет значение *единица*. Пока, предположите, что индексы строки и столбца могут иметь, и положительные и отрицательные значения, так, что единица центрирована в обширном море нулей. Когда дельта функцию пропускают через линейную систему, единственная точка отличная от нуля будет изменена на некоторый другой двумерный образец. Так как единственная вещь, которая может случаться с точкой, состоит в том, что это распространение(рассеивание), импульсная передаточная функция часто **функция рассеяния точки** (**ФРТ**) на жаргоне обработки изображений.

The human eye provides an excellent example of these concepts. As described in the last chapter, the first layer of the retina transforms an image represented as a pattern of light into an image represented as a pattern of nerve impulses. The second layer of the retina *processes* this neural image and passes it to the third layer, the fibers forming the optic nerve. Imagine that the image being projected onto the retina is a very small spot of light in the center of a dark background. That is, an *impulse* is fed into the eye. Assuming that the system is linear, the image processing taking place in the retina can be determined by inspecting the image appearing at the optic nerve. In other words, we want to find the *point spread function* of the processing. We will revisit the assumption about linearity of the eye later in this chapter.

Человеческий глаз обеспечивает превосходный пример этих концепций. Как описано в прошлой главе, первый уровень сетчатки преобразовывает изображение, представленное как образец индикатора в изображение, представленное как образец импульсов нерва. Второй уровень сетчатки обрабатывает это невральное изображение и передает это к третьему уровню, стекловолокна, формирующие зрительный нерв. Вообразите, что изображение, проектируемое на сетчатку - очень маленькое пятно(ячейка) индикатора в центре темного фона. То есть в глаз подан импульс. Принимая, что система является линейной, обработка изображения, имеющая место в сетчатке, может быть определена, осматривая изображение, появляющееся в зрительном нерве. Другими словами, мы хотим найти функцию рассеяния точки обработки. Мы повторно посетим предположение относительно линейности глаза позже в этой главе.

Figure 24-1 outlines this experiment. Figure (a) illustrates the impulse striking the retina while (b) shows the image appearing at the optic nerve. The middle layer of the eye passes the bright spike, but produces a circular region of increased *darkness*. The eye accomplishes this by a process known as *lateral inhibition*. If a nerve cell in the middle layer is activated, it decreases the ability of its nearby neighbors to become active. When a complete image is viewed by the eye, each point in the image contributes a scaled and shifted version of this impulse response to the image appearing at the optic nerve. In other words, the visual image is *convolved* with this PSF to produce the neural image transmitted to the brain. The obvious question is: how does convolving a viewed image with this PSF improve the ability of the eye to understand the world?

Рисунок 24-1 основывает этот эксперимент. Рисунок (a) иллюстрирует импульс, нажимающий сетчатку, в то время как (b) показывает изображение, появляющееся в зрительном нерве. Средний уровень глаза передает яркий выброс, но производит круговую область увеличенной темноты. Глаз выполняет это процессом известный как боковое подавление. Если нервная клетка в среднем уровне активизирована, это уменьшает способность

его близлежащих соседей, чтобы стать активным. Когда полное изображение просмотрено глазом, каждая точка в изображении жертвует(вносит) масштабируемую и сдвинутую версию этой импульсной передаточной функции к изображению, появляющемуся в зрительном нерве. Другими словами, визуальное изображение свернуто с этим ФРТ, чтобы произвести невральное изображение, переданное к мозгу. Очевидный вопрос: как свертка просмотренного изображения с этим ФРТ улучшает способность глаза, чтобы понять мир?

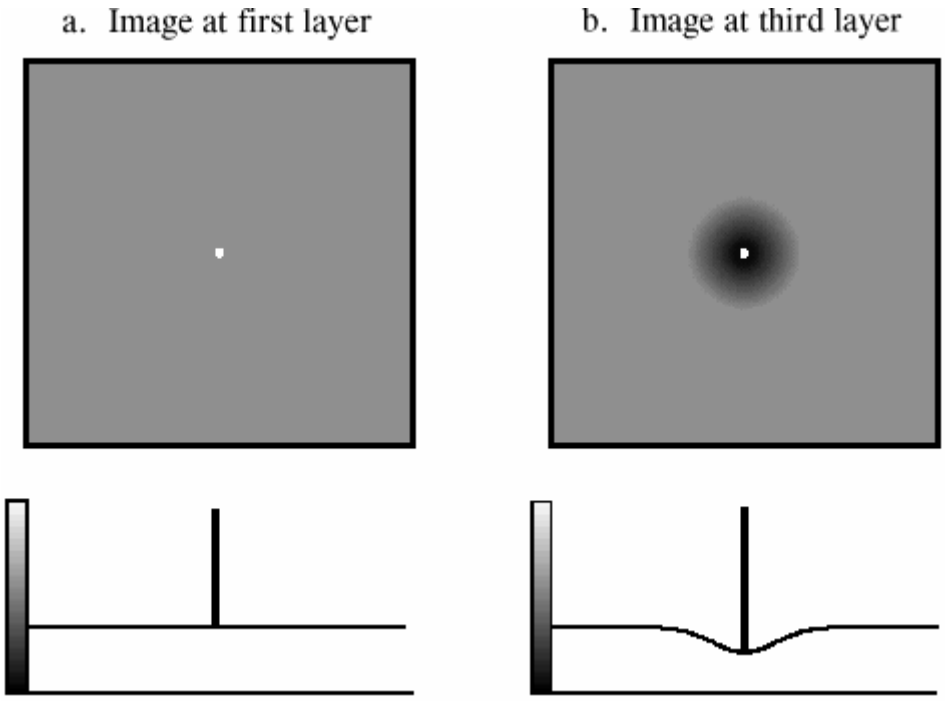
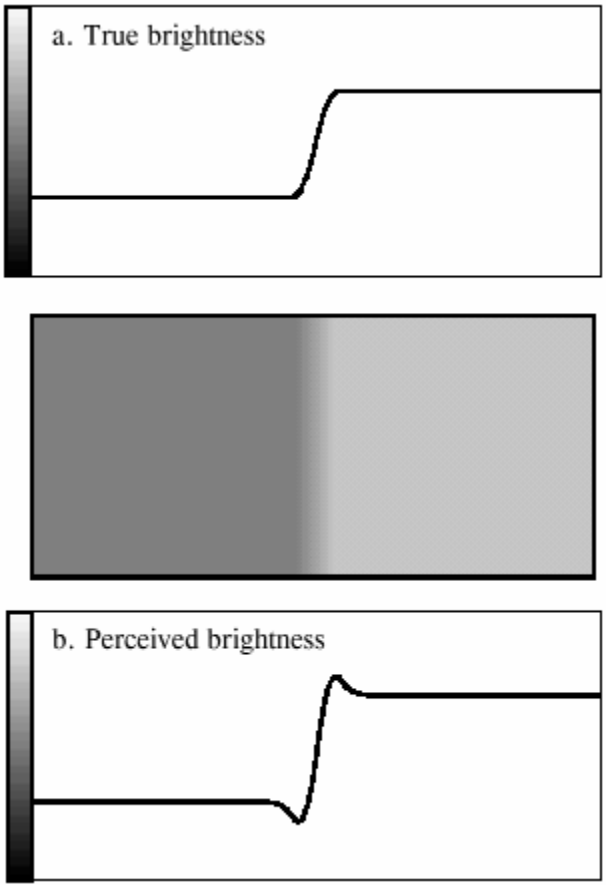


FIGURE 24-1
The PSF of the eye. The middle layer of the retina changes an impulse, shown in (a), into an impulse surrounded by a dark area, shown in (b). This point spread function enhances the edges of objects.
РИСУНОК 24-1
ФРТ глаза. Средний уровень сетчатки изменяет импульс, показанный в (a), в импульс, окруженный темной областью, показанный в (b). Эта функция рассеяния точки расширяет грани объектов.

FIGURE 24-2
Mach bands. Image processing in the retina results in a slowly changing edge, as in (a), being sharpened, as in (b). This makes it easier to separate objects in the image, but produces an optical illusion called *Mach bands*. Near the edge, the overshoot makes the dark region look darker, and the light region look lighter. This produces dark and light bands that run parallel to the edge.
РИСУНОК 24-2
Полосы Маха. Обработка Изображения в сетчатке приводит к медленно изменяющемуся краю(фронту), как в (a), заостряемый, как в (b). Это облегчает возможность отделять объекты в изображении, но производит оптический обман по имени полосы Маха. Около края(фронта), перерегулирование делает темный просмотр области более темным, и светлый просмотр области светлее. Это производит темные и светлые полосы, которые проходят параллельно краю(фронту).



Humans and other animals use vision to identify nearby objects, such as enemies, food, and mates. This is done by distinguishing one region in the image from another, based on differences in brightness and color. In other words, the first step in recognizing an object is to identify its *edges*, the discontinuity that separates an object from its background. The middle layer of the retina helps this task by sharpening the edges in the viewed image. As an illustration of how this works, Fig. 24-2 shows an image that slowly changes from dark to light, producing a blurry and poorly defined edge. Figure (a) shows the intensity profile of this image, the pattern of brightness entering the eye. Figure (b) shows the brightness profile appearing on the optic nerve, the image transmitted to the brain. The processing in the retina makes the edge between the light and dark areas appear more abrupt, reinforcing that the two regions are different.

Люди и другие животные используют зрение, чтобы идентифицировать поблизости объекты, типа врагов, продовольствия, и помощников. Это сделано, отличая одну область в изображении от другой, основано на различиях в яркости и цвете. Другими словами, первый шаг в признание объекта должен идентифицировать его грани, разрыв, который отделяет объект от его фона. Средний уровень сетчатки помогает этой задаче, заостряя грани в просмотренном изображении. Как иллюстрация того, как это работает, рис. 24-2 показывает изображение которое медленно изменяется от темного, до светлого, производя расплывчатый и плохо определенный край(фронт). Рисунок (a) показывает конфигурацию интенсивности этого изображения, образец яркости, вводящей глаз. Рисунок (b) показывает конфигурацию яркости, появляющуюся на зрительном нерве, изображение, переданное к мозгу. Обработка в сетчатке делает край(фронт) между световыми и темными областями, кажутся(появляются) более резкими, укрепляя, что эти две области различны.

The overshoot in the edge response creates an interesting optical illusion. Next to the edge, the dark region appears to be unusually dark, and the light region appears to be unusually light. The resulting light and dark strips are called **Mach bands**, after Ernst Mach (1838-1916), an Austrian physicist who first described them.

Перерегулирование в ответе края(фронта) создает интересный оптический обман. Рядом с краем(фронтом), темная область, кажется, необычно темной, и светлая область, кажется, необычно светлой. Заканчивающийся индикатор и темные ленты называются **полосами Маха**, по имени Ernst Mach (1838-1916), австрийского физика который первым описал их.

As with one-dimensional signals, image convolution can be viewed in two ways: from the input, and from the output. From the input side, each pixel in the input image contributes a scaled and shifted version of the point spread function to the output image. As viewed from the output side, each pixel in the output image is influenced by a group of pixels from the input signal. For one-dimensional signals, this region of influence is the impulse response flipped *left-for-right*. For image signals, it is the PSF flipped *left-for-right* and *top-for- bottom*. Since most of the PSFs used in DSP are symmetrical around the vertical and horizonal axes, these flips do nothing and can be ignored. Later in this chapter we will look at nonsymmetrical PSFs that must have the flips taken into account.

Как с одномерными сигналами, свертка изображения может быть просмотрена двумя способами: от ввода, и от выхода. От входной стороны, каждый пиксел во входном изображении жертвует(вносит; способствует)) масштабируемую и сдвинутую версию функции рассеяния точки к изображению выхода. Как просмотрено от стороны выхода, каждый пиксел в изображении выхода - под влиянием группы пикселов от входного сигнала. Для одномерных сигналов, эта область влияния - зеркально отраженная импульсная передаточная функция, лево - право. Для сигналов изображения, это - зеркально отраженный ФРТ, *лево - право* и *низ - верх*. Так как большинство ФРТ, используемых в ЦОС симметричны вокруг вертикальной и горизонтальной осей, они зеркально отражаясь, они не делают ничего и можгут игнорироваться. Позже в этой главе мы будем смотреть на несимметрические ФРТ, которые должны принять во внимание, зеркальное отражение.

Figure 24-3 shows several common PSFs. In (a), the **pillbox** has a circular top and straight sides. For example, if the lens of a camera is not properly focused, each point in the image will be projected to a circular spot on the image sensor (look back at Fig. 23-2 and consider the effect of moving the projection screen toward or away from the lens). In other words, the pillbox is the point spread function of an out-of-focus lens.

Рисунок 24-3 показывает несколько обычных ФРТ. В (a), **pillbox** имеет круговые(циклические) высшие и прямые стороны. Например, если хрусталик камеры должным образом не сосредоточен, каждая точка в изображении будет проектироваться к круговому(циклическому) пятну(ячейке) на датчике изображения (взгляд назад в рис. 23-2 и рассматривать эффект перемещения проекционного экрана к или далеко от хрусталика). Другими словами, pillbox - функция рассеяния точки вне фокуса(нерезкости) хрусталика(линзы).

The **Gaussian**, shown in (b), is the PSF of imaging systems limited by *random* imperfections. For instance, the image from a telescope is blurred by atmospheric turbulence, causing each point of light to become a Gaussian in the final image. Image sensors, such as the CCD and retina, are often limited by the scattering of light and/or electrons. The Central Limit Theorem dictates that a Gaussian blur results from these types of random processes.

Гауссиан, показанный в (b), является ФРТ отображения систем, ограниченных случайными недостатками. Например, изображение от телескопа размыто атмосферной турбулентностью, вызывая каждую точку индикатора стать Гауссиан в конечном(заключительном) изображении. Датчики изображения, типа ПЗС и сетчатки, часто ограничиваются рассеиванием индикатора и-или электронов. Центральная предельная теорема диктует, что Гауссиан пятно следует из этих типов вероятностных процессов.

The pillbox and Gaussian are used in image processing the same as the *moving average filter* is used with one-dimensional signals. An image convolved with these PSFs will appear blurry and have less defined edges, but will be lower in random noise. These are called **smoothing filters**, for their action in the time domain, or **low-pass filters**, for how they treat the frequency domain. The **square** PSF, shown in (c), can also be used as a smoothing filter, but it is not circularly symmetric. This results in the blurring being different in the diagonal directions compared to the vertical and horizontal. This may or may not be important, depending on the use.

Pillbox и Гауссиан используются в изображении, обрабатывающем тот же самый, поскольку фильтр скользящего среднего значения используется с одномерными сигналами. Изображение, свернутое с этими ФРТ будет казаться расплывчатым и меньше определило грани, но будет более низкое в случайном шуме. Они называются, **фильтры сглаживания**, по их действию в домене времени, или фильтрами нижних частот, по тому, как они обрабатывают частотный домен. **Квадратный** ФРТ, показанный в (c), может также использоваться как фильтр сглаживания, но это не циркулярно симметрично. Это приводит к размыванию, являющемуся отличным в диагональных направлениях, сравненных с вертикальной и горизонтальной линиями. Это может или не может быть важным, в зависимости от использования.

The opposite of a smoothing filter is an **edge enhancement** or **high-pass filter**. The spectral inversion technique, discussed in Chapter 14, is used to change between the two. As illustrated in (d), an edge enhancement filter kernel is formed by taking the *negative* of a smoothing filter, and adding a delta function in the center. The image processing which occurs in the retina is an example of this type of filter.

Противоположность фильтра сглаживания - **подчеркивание контуров** или **фильтр верхних** частот. Спектральная методика обращения, обсужденная в главе 14, используется, чтобы измениться между двумя. Как иллюстрировано в (d), ядро фильтра подчеркивания контуров, сформировано, беря негатив фильтра сглаживания, и добавления дельта функции в центре. Обработка изображения, которая происходит в сетчатке - пример этого типа фильтра.

Figure (e) shows the two-dimensional sinc function. One-dimensional signal processing uses the windowed-sinc to separate frequency bands. Since images do not have their information encoded in the frequency domain, the sinc function is seldom used as an imaging filter kernel, although it does find use in some theoretical problems. The sinc function can be hard to use because its tails decrease very slowly in amplitude ($1/x$), meaning it must be treated as infinitely wide. In comparison, the Gaussian's tails decrease very rapidly (e^{-x^2}) and can eventually be truncated with no ill effect.

Рисунок (e) показывает двумерную sinc(синхронизирующую?) функцию. Одномерная обработка сигналов использует windowed-sinc, чтобы отделить полосы частот. Так как изображения не кодируют их информацию в частотном домене, sinc функция редко используется, как ядро фильтра отображения, хотя это находит использование в некоторых тео-

ретических проблемах. Sinc функция может быть интенсивно, чтобы использовать потому что его Остатки(срезы импульса(хвосты) уменьшаются очень медленно в амплитуде ($1/x$), означая это должно быть обработано как бесконечно широкий. Для сравнения, уменьшение остатков(срезов импульса) гауссиана очень быстро (e^{-x^2}) и может в конечном счете быть усечено без вредного воздействия.

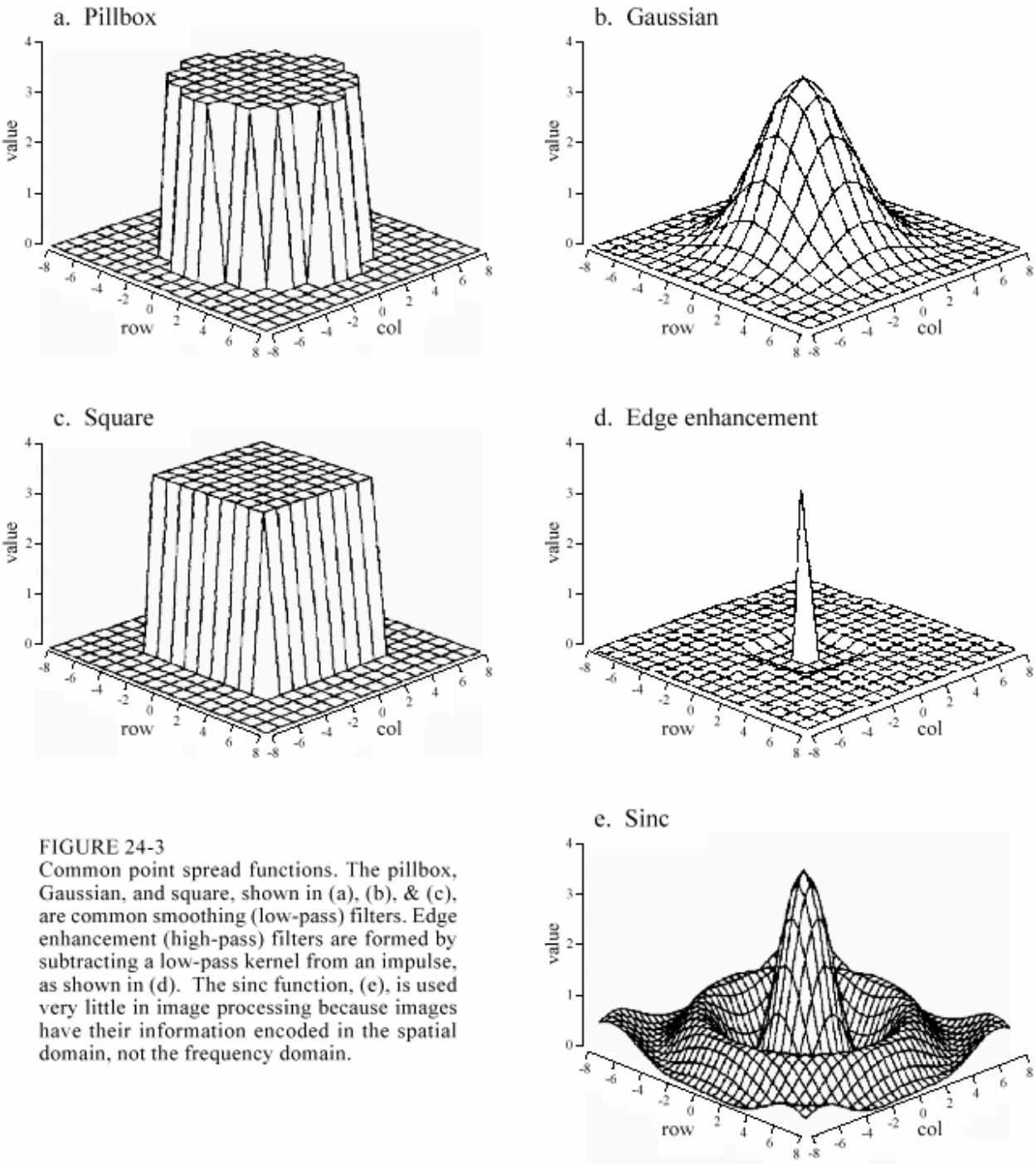


FIGURE 24-3
Common point spread functions. The pillbox, Gaussian, and square, shown in (a), (b), & (c), are common smoothing (low-pass) filters. Edge enhancement (high-pass) filters are formed by subtracting a low-pass kernel from an impulse, as shown in (d). The sinc function, (e), is used very little in image processing because images have their information encoded in the spatial domain, not the frequency domain.

FIGURE 24-3
Common point spread functions. The pillbox, Gaussian, and square, shown in (a), (b), & (c), are common smoothing (low-pass) filters. Edge enhancement (high-pass) filters are formed by subtracting a low-pass kernel from an impulse, as shown in (d). The sinc function, (e), is used very little in image processing because images have their information encoded in the spatial domain, not the frequency domain.

РИСУНОК 24-3
Обычные функции рассеяния точки. Pillbox, Гауссиан, и квадрат, показанный в (a), (b), и (c), являются обычными сглаживающими фильтрами (фильтры никих частот). Фильтры подчеркивания контуров (фильт-

ры верхних частот) сформировано, вычитая ядро с низким проходом от импульса, как показано в (d). Sinc функция, (e), используется очень немного в обработке изображения, потому что изображения кодируют их информацию в пространственном домене, не частотном домене.

All these filter kernels use *negative* indexes in the rows and columns, allowing the PSF to be centered at *row* = 0 and *column* = 0. Negative indexes are often eliminated in one-dimensional DSP by shifting the filter kernel to the right until all the nonzero samples have a positive index. This shift moves the output signal by an equal amount, which is usually of no concern. In comparison, a shift between the input and output images is generally not acceptable. Correspondingly, negative indexes are the norm for filter kernels in image processing.

Все эти ядра фильтра используют отрицательные индексы в строках и столбцах, позволяя ФРТ быть центрированными в *строке* = 0 и *столбце* = 0. Отрицательные индексы часто устраняются в одномерном ЦОС, сдвигая ядро фильтра направо, пока все выборки отличные от нуля не имеют положительный индекс. Этот сдвиг перемещает сигнал выхода равным количеством, которое не вызывает обычно никакого беспокойства. Для сравнения, сдвиг между изображениями ввода и вывода вообще не приемлем. Соответственно, отрицательные индексы - норма для ядер фильтра в обработке изображения.

A problem with image convolution is that a large number of calculations are involved. For instance, when a 512 by 512 pixel image is convolved with a 64 by 64 pixel PSF, more than a *billion* multiplications and additions are needed (i.e., 64 x 64 x 512 x 512). The long execution times can make the techniques impractical. Three approaches are used to speed things up.

Проблема со сверткой изображения состоит в том, что большое количество вычислений вовлечено. Для образца, когда изображение 512 by 512 пикселов свернуто с ФРТ 64 by 64 пикселов, больше чем миллиард умножений и добавлений необходим (то есть, 64 x 64 x 512 x 512). Длинные времена выполнения могут делать методы непрактичными. Три подхода используются, чтобы ускорить эти вещи.

The first strategy is to use a very small PSF, often only 3Ч3 pixels. This is carried out by looping through each sample in the output image, using optimized code to multiply and accumulate the corresponding nine pixels from the input image. A surprising amount of processing can be achieved with a mere 3Ч3 PSF, because it is large enough to affect the *edges* in an image.

Первая стратегия состоит в том, чтобы использовать очень маленький ФРТ, часто только 3x3 пиксела. Это выполнено выполнением цикла через каждую выборку в изображении выхода, используя оптимизированную программу, чтобы умножать и накопить передачу девять пикселов от входного изображения. Удивительное количество обработки может быть достигнуто с простым ФРТ 3x3, потому что достаточно эффективно затронуть *грани* в изображении.

The second strategy is used when a large PSF is needed, but its shape isn't critical. This calls for a filter kernel that is *separable*, a property that allows the image convolution to be carried out as a series of one-dimensional operations. This can improve the execution speed by *hundreds* of times.

Вторая стратегия используется, когда большой ФРТ необходим, но его форма не критическая. Это запрашивает ядра фильтра, которое является *разделяемым*, свойство, которое позволяет свертке изображения быть выполненной как ряд одномерных операций. Это может улучшать *быстродействие* выполнения в сотни раз.

The third strategy is FFT convolution, used when the filter kernel is large and has a specific shape. Even with the speed improvements provided by the highly efficient FFT, the execution time will be hideous. Let's take a closer look at the details of these three strategies, and examples of how they are used in image processing.

Третья стратегия - свертка БПФ, используемая, когда ядро фильтра большое и имеет специфическую(точно определенную) форму. Даже с уточнениями быстродействия, обеспеченными высоко эффективным БПФ, время выполнения будет отвратительно. Давайте ближе рассмотрим подробности этих трех стратегий, и примеров того, как они используются в обработке изображения.

3x3 Edge Modification
Модификация края 3x3

Figure 24-4 shows several 3Ч3 operations. Figure (a) is an image acquired by an airport x-ray baggage scanner. When this image is convolved with a 3Ч3 delta function (a *one* surrounded by 8 zeros), the image remains unchanged. While this is not interesting by itself, it forms the baseline for the other filter kernels.

Рисунок 24-4 показывает несколько операций 3x3. Рисунок (a) - изображение, приобретенное рентгеновским сканером авиа-багажа. Когда это изображение свернуто с дельта функцией 3x3 (*единица*, окруженная 8 нулями), изображение остается неизменяемым. В то время как это не интересно отдельно, это формирует опорную линию для других ядер фильтра.

Figure (b) shows the image convolved with a 3Ч3 kernel consisting of a one, a negative one, and 7 zeros. This is called the **shift and subtract** operation, because a *shifted* version of the image (corresponding to the -1) is *subtracted* from the original image (corresponding to the 1). This processing produces the optical illusion that some objects are closer or farther away than the background, making a 3D or embossed effect. The brain interprets images as if the lighting is from *above*, the normal way the world presents itself. If the edges of an object are bright on the top and dark on the bottom, the object is perceived to be poking out from the background. To see another interesting effect, turn the picture upside down, and the objects will be pushed *into* the background.

Рисунок (b) показывает изображение, свернутое с ядром 3x3, состоящее из *минус единицы*, и 7 нулей. Это называется операцией **сдвига и вычитания**, потому что *сдвинутая* версия изображения (соответствующего -1) *вычитается* от первоначального изображения (соответствующего 1). Эта обработка производит оптическую иллюзию, что некоторые объекты ближе или дальше чем фон, создавая трехмерный эффект или эффект штампа. Мозг интерпретирует изображения, как будто освещение - *сверху*, нормальный способ представления реального мира. Если грани объекта светлые вверху и темные внизу, объект воспринят, чтобы выделяться от фона. Чтобы видеть другой интересный эффект, поверните изображение, инвертированное вниз(вверх-дном), и объекты будут помещены *в* фон.

Figure (c) shows an **edge detection** PSF, and the resulting image. Every edge in the original image is transformed into narrow dark and light bands that run parallel to the original edge. Thresholding this image can isolate either the dark or light band, providing a simple algorithm for detecting the edges in an image.

Рисунок (c) показывает **обнаружение края** ФРТ, и заканчивающееся изображение. Каждый край в первоначальном изображении преобразован в узкие темные и светлые полосы, расположенные параллельно оригинальному (первоначальному) краю. Это пороговое изображение может изолировать или темную или светлую полосу, обеспечивая простой алгоритм для обнаружения граней в изображении.

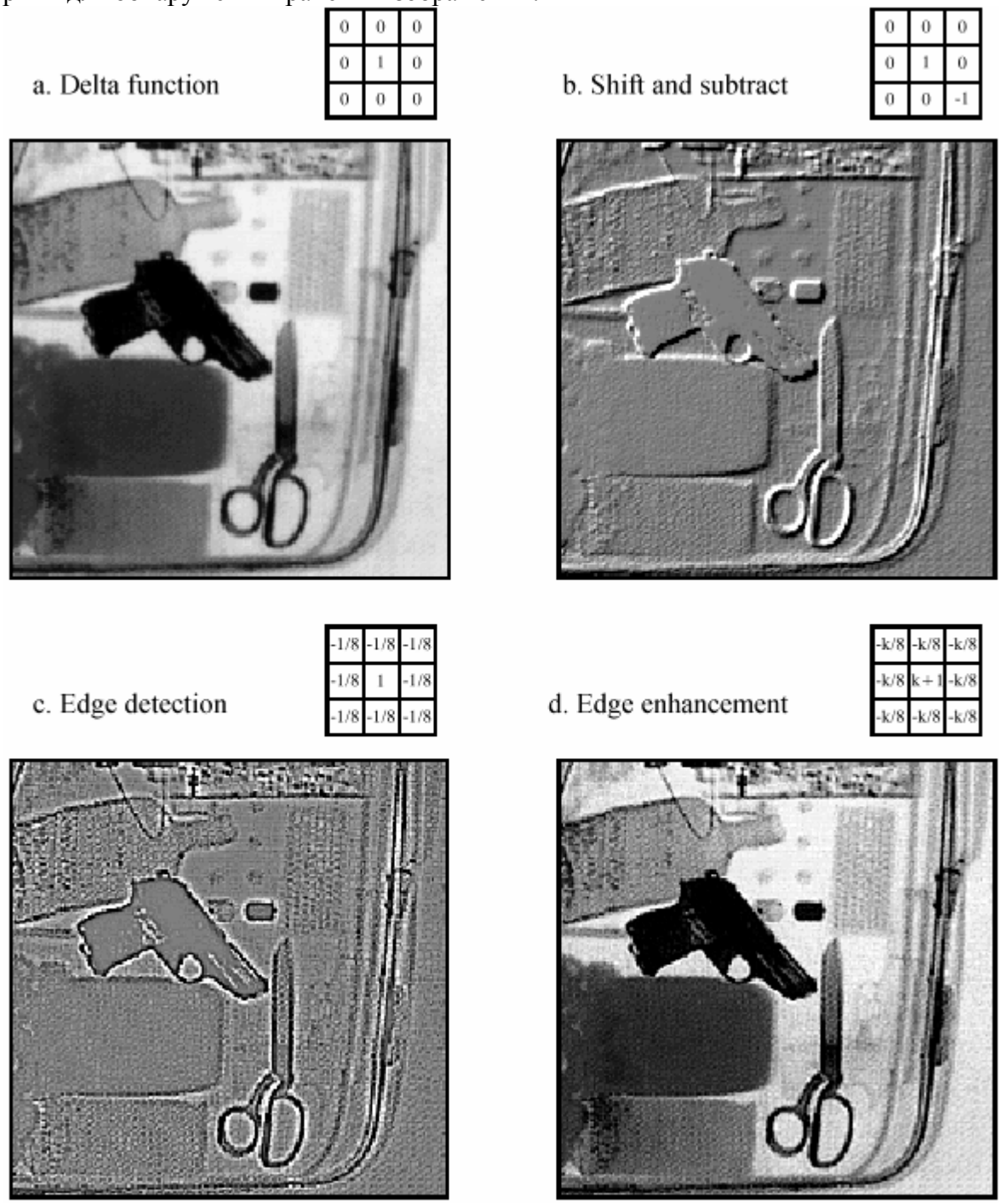


FIGURE 24-4
3Ч3 edge modification. The original image, (a), was acquired on an airport x-ray baggage scanner. The shift and subtract operation, shown in (b), results in a pseudo three-dimensional effect. The edge detection operator in (c) removes all contrast, leaving only the edge information. The edge enhancement filter, (d), adds various ratios of images (a) and (c), determined by the parameter, *k*. A value of $k = 2$ was used to create this image.

РИСУНОК 24-4
Модификация края(обрамления; контура?) 3x3. Первоначальное изображение, (a), было приобретено на рентгеновском сканере авиа-багажа. Операция сдвига и вычитания, показанная в (b), приводит к псевдо трехмерному эффекту. Оператор обнаружения края в (c) удаляет весь контраст, оставляя только информа-

цию края(контуров?). Фильтр подчеркивания контуров, (d), прибавляет различные отношения(коэффициенты) изображений (a) и (c), определенные параметром, *k*. Чтобы создать это изображение использовалось значение, $k = 2$.

A common image processing technique is shown in (d): **edge enhancement**. This is sometimes called a *sharpening* operation. In (a), the objects have good contrast (an appropriate level of arkness and lightness) but very blurry edges. In (c), the objects have absolutely no contrast, but very sharp edges. The strategy is to multiply the image with *good edges* by a constant, *k*, and add it to the image with *good contrast*. This is equivalent to convolving the original image with the 3Ч3 PSF shown in (d). If *k* is set to 0, the PSF becomes a delta function, and the image is left unchanged. As *k* is made larger, the image shows better edge definition. For the image in (d), a value of $k = 2$ was used: two parts of image (c) to one part of image (a). This operation mimics the eye's ability to sharpen edges, allowing objects to be more easily separated from the background.

Обычная методика обработки изображения, показывается в (d): **подчеркивание контуров**. Это иногда называется операцией увеличения *резкости*. В (a), объекты имеют хороший контраст (соответствующий уровень тени и света) но очень расплывчатые грани. В (c), объекты не имеют абсолютно никакого контраста, но очень резкие края. Стратегия состоит в том, чтобы умножить(мультиплицировать) изображение с *хорошими гранями* константой, *k*, и прибавлять это к изображению с *хорошим контрастом*. Это эквивалентно скручиванию первоначального изображения с ФРТ 3x3, показанный в (d). Если *k* установлен в 0, ФРТ становится дельта функцией, и изображение оставлено неизменяемым. Как *k* сделан большим, изображение показывает лучше определение края. Для изображения в (d), использовалось значение $k = 2$: две части изображения (c) к одной части изображения (a). Эта операция подражает способности глаза заострить грани, позволяя объектам быть более легко отделенными от фона.

Convolution with any of these PSFs can result in negative pixel values appearing in the final image. Even if the program can handle negative values for pixels, the image display cannot. The most common way around this is to add an offset to each of the calculated pixels, as is done in these images. An alternative is to truncate out-of-range values.

Свертка с любым из этих ФРТ может приводить к отрицательным значениям пиксела, появляющимся в конечном изображении. Даже если программа может обрабатывать отрицательные значения для пикселов, дисплей изображения не может. Наиболее обычный путь вокруг этого состоит в том, чтобы прибавить смещение к каждому из расчетных пикселов, как сделано в этих изображениях. Альтернатива должна усечь значения из диапазона.

Convolution by Separability
Свертка Отделимостью

This is a technique for fast convolution, as long as the PSF is **separable**. A PSF is said to be *separable* if it can be broken into two one-dimensional signals: a vertical and a horizontal projection. Figure 24-5 shows an example of a separable image, the square PSF. Specifically, the value of each pixel in the image is equal to the corresponding point in the horizontal projection multiplied by the corresponding point in the vertical projection. In mathematical form:

Это - методика для быстрой свертки, пока ФРТ **отделим**. ФРТ, как считают, отделимым, если это может быть разбито на два одномерных сигнала: вертикальная и горизонтальная проекция. Рисунок 24-5 показывает пример отделимого изображения, квадратного ФРТ. Определенно, значение каждого пиксела в изображении равно соответствующей точке на горизонтальной проекции, мультиплицированной(умноженной) соответствующей точкой на вертикальной проекции. В математической форме:

УРАВНЕНИЕ 24-1. Разделение Изображения. Изображение упомянуто как *отделимое*, если оно может быть расчленено на горизонтальную и вертикальную проекции.

$$x[r,c] = vert[r] \times horz[c]$$

where $x[r, c]$ is the two-dimensional image, where $vert[r]$ и $horz[c]$ are the one-dimensional projections. Obviously, most images do not satisfy this requirement. For example, the pillbox is not separable. There are, however, an *infinite* number of separable images. This can be understood by generating arbitrary horizontal and vertical projections, and finding the image that corresponds to them. For example, Fig. 24-6 illustrates this with profiles that are double-sided exponentials. The image that corresponds to these profiles is then found from Eq. 24-1. When displayed, the image appears as a diamond shape that exponentially decays to zero as the distance from the origin increases.

где $x[r, c]$ - двумерное изображение, и где $vert[r]$ и $horz[c]$ - одиноразмерные проекции. Очевидно, большинство изображений не удовлетворяет этому требованию. Например, pillbox не отделим. Имеется, однако, *бесконечное* число отделимых изображений. Это может быть понято, генерируя произвольные горизонтальные и вертикальные проекции, и находя изображение, которое соответствует им. Например, рис. 24-6 иллюстрирует это с конфигурациями, которые являются двусторонними показательными функциями. Изображение, которое соответствует этим конфигурациям, тогда найдено из уравнения 24-1. Когда отображено, изображение появляется как ромбовидная форма, которая распадается по экспоненте, распадаясь до нуля по мере того, как расстояние начала координат увеличивается.

In most image processing tasks, the ideal PSF is *circularly symmetric*, such as the pillbox. Even though digitized images are usually stored and processed in the rectangular format of rows and columns, it is desired to modify the image the same in all directions. This raises the question: is there a PSF that is circularly symmetric *and* separable? The answer is, yes, but there is only one, the *Gaussian*. As is shown in Fig. 24-7, a two-dimensional Gaussian image has projections that are also Gaussians. The image and projection Gaussians have the *same* standard deviation.

В большинстве задач обработки изображений, идеальный ФРТ *циркулярный симметричен*, типа pillbox. Даже притом, что цифровые изображения обычно сохраняются и обрабатываются в прямоугольном формате строк и столбцов, это желательно, чтобы изменить изображение одинаково во всех направлениях. Это поднимает вопрос: имеется ФРТ, который циркулярный симметричный *и* отделим? Ответ, да, но имеется только один, *Гауссиан*. Как показывается в рис. 24-7, двумерное изображение Гауссиана имеет проекции, которые являются также гауссианами. Изображение и проекции гауссиана имеют то *же самое* среднеквадратичное отклонение.

To convolve an image with a separable filter kernel, convolve each *row* in the image with the *horizontal projection*, resulting in an intermediate image. Next, convolve each *column* of this intermediate image with the *vertical projection* of the PSF. The resulting image is identical to the

direct convolution of the original image and the filter kernel. If you like, convolve the columns first and then the rows; the result is the same.

Чтобы свертывать изображение с отделимым ядром фильтра, сверните каждую *строку* в изображении с *горизонтальной проекцией*, приводя к промежуточному изображению. Затем, сверните каждый столбец этого промежуточного изображения с *вертикальной проекцией* ФРТ. Заканчивающееся изображение идентично прямой свертке первоначального изображения и ядра фильтра. Если Вам нравится, свертывайте сначала столбцы и затем строки; результат - тот же самый.

FIGURE 24-5
Separation of the rectangular PSF. A PSF is said to be *separable* if it can be decomposed into horizontal and vertical profiles. Separable PSFs are important because they can be rapidly convolved.

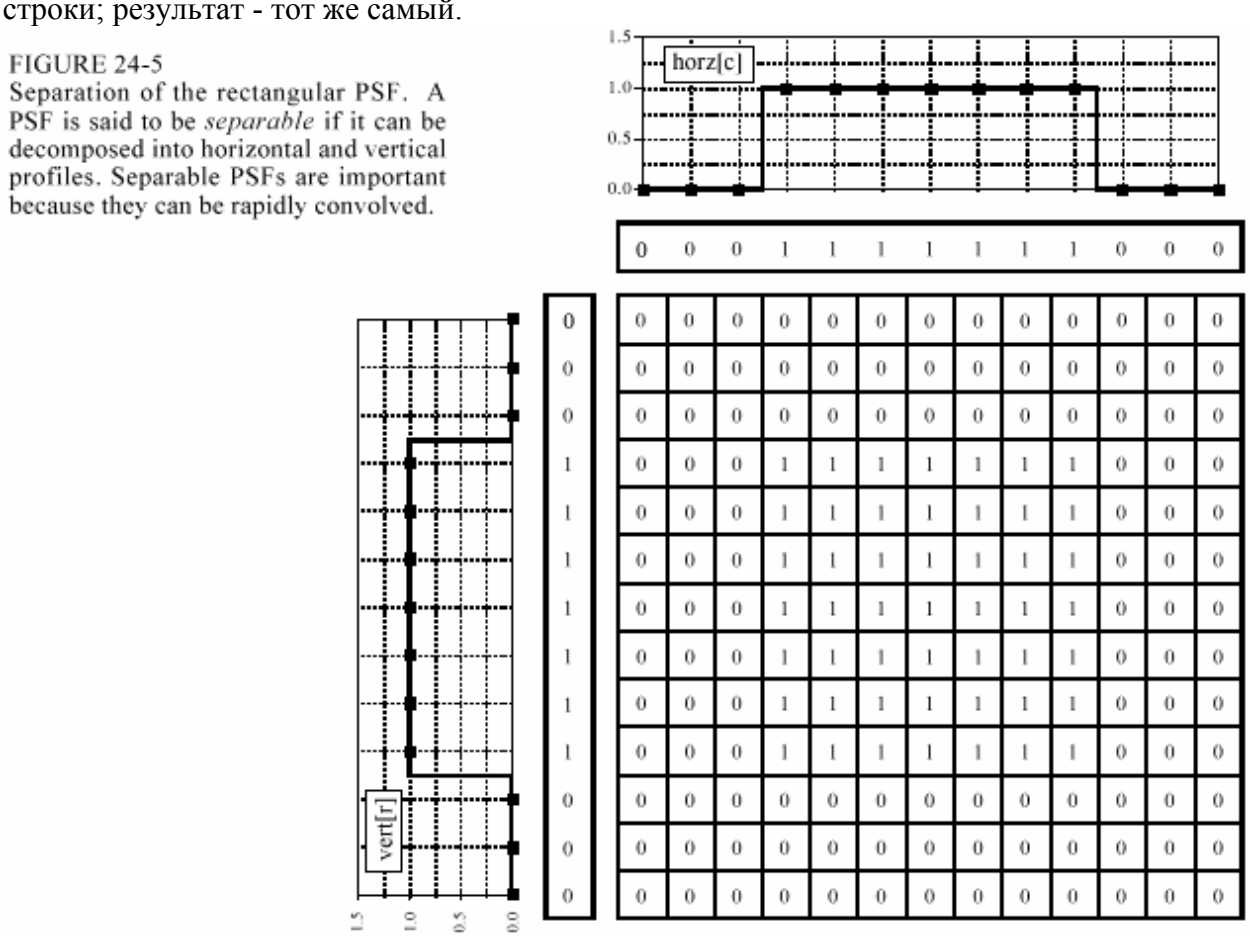


FIGURE 24-5
Separation of the rectangular PSF. A PSF is said to be *separable* if it can be decomposed into horizontal and vertical profiles. Separable PSFs are important because they can be rapidly convolved.
РИСУНОК 24-5
Разделение прямоугольного ФРТ. ФРТ, как считают, *отделимый*, если может быть расчленен на горизонтальные и вертикальные конфигурации. Отделимые ФРТ важны, потому что они могут быть быстро свернуты.

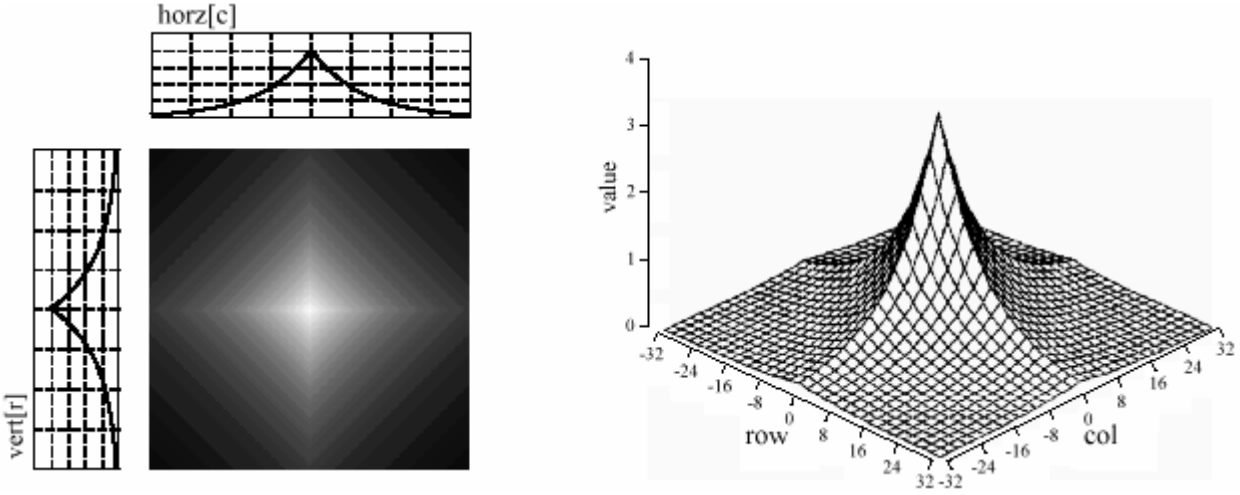


FIGURE 24-6
Creation of a separable PSF. An infinite number of separable PSFs can be generated by defining arbitrary projections, and then calculating the two-dimensional function that corresponds to them. In this example, the profiles are chosen to be double-sided exponentials, resulting in a diamond shaped PSF.
РИСУНОК 24-6
Создание отделимого ФРТ. Бесконечное число отделимых ФРТ может быть сгенерировано, определяя произвольные проекции, и затем, вычисляя двумерную функцию, которая им соответствует. В этом примере, конфигурации выбраны, чтобы быть двусторонними показательными функциями, приводя к ФРТ имеющему ромбовидному форму.

FIGURE 24-7
Separation of the Gaussian. The Gaussian is the only PSF that is circularly symmetric *and* separable. This makes it a common filter kernel in image processing.

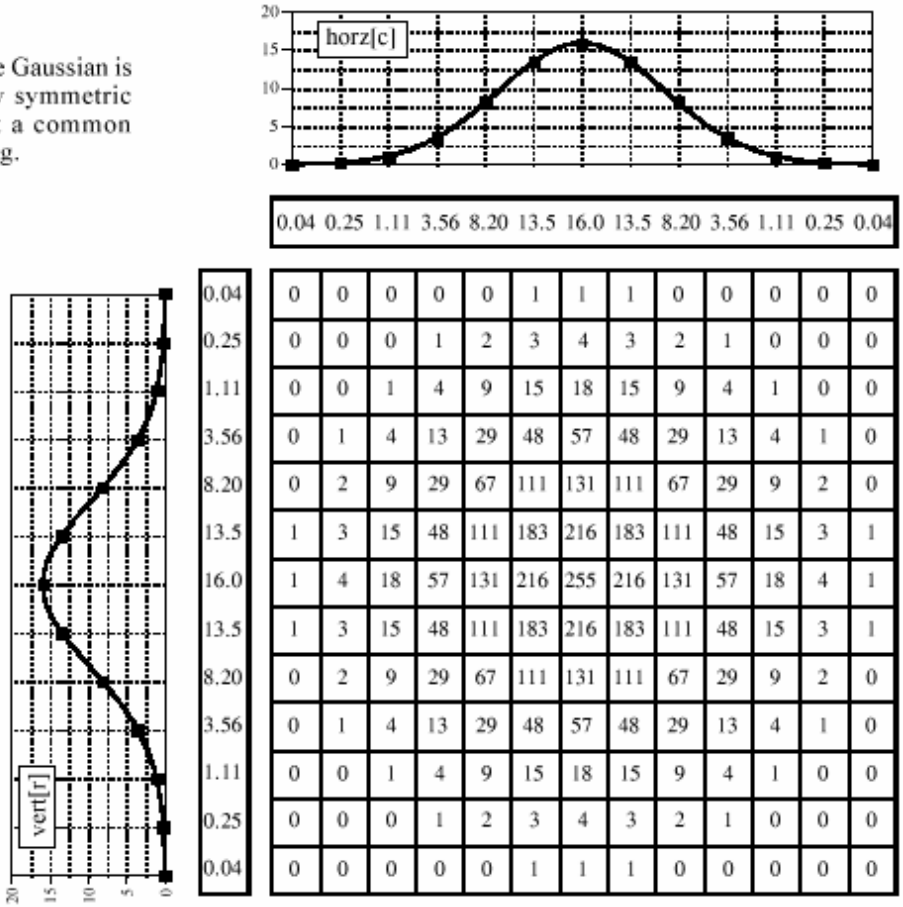


FIGURE 24-7
Separation of the Gaussian. The Gaussian the only PSF that is circularly symmetric *and* separable. This makes it a common filter kernel in image processing.
РИСУНОК 24-7
Разделение Гауссиана. Гауссиан единственный ФРТ, который является циркулярным симметричным *и* отделимым. Это делает это обычным ядром фильтра в обработке изображения.

The convolution of an *N*x*N* image with an *M*x*M* filter kernel requires a time proportional to N^2M^2. In other words, each pixel in the output image depends on *all* the pixels in the filter kernel. In comparison, convolution by separability only requires a time proportional N^2M to For filter kernels that are hundreds N^2M. of pixels wide, this technique will reduce the execution time by a factor of *hundreds*.

Свертка изображения *N*x*N* с ядром фильтра *M*x*M* требует времени пропорционально к N^2M^2. Другими словами, каждый пиксел в изображении выхода зависит от всех пикселов в ядре фильтра. Для сравнения, свертка отделимостью требует только пропорции времени к N^2M. Для ядер фильтра, которые являются сотнями широких пикселов, эта методика приведет время выполнения в сотни раз.

Things can get even better. If you are willing to use a *rectangular* PSF (Fig. 24-5) or a *double-sided exponential* PSF (Fig. 24-6), the calculations are even more efficient. This is because the one-dimensional convolutions are the *moving average* filter (Chapter 15) and the *bidirectional single pole* filter(Chapter 19), respectively. Both of these one-dimensional filters can be rapidly carried out by recursion. This results in an image convolution time proportional to only N^2, completely independent of the size of the PSF. In other words, an image can be convolved with as large a PSF as needed, with only a few integer operations per pixel. For example, the convolution of a 512x512 image requires only a few hundred milliseconds on a personal computer. That's fast! Don't like the shape of these two filter kernels? Convolve the image with one of them *several times* to approximate a Gaussian PSF (guaranteed by the Central Limit Theorem, Chapter 7). These are great algorithms, capable of snatching success from the jaws of failure. They are well worth remembering.

Вещи могут становиться четными лучше. Если Вы желаете использовать *прямоугольный* ФРТ (рис. 24-5) или *двусторонний показательный* ФРТ (рис. 24-6), четные вычисления более эффективны. Это - то, потому что одномерные свертки - фильтр *скользящего среднего* значения (глава 15) и *двунаправленный однополюсный* фильтр (глава 19), соответственно. Оба из этих одномерных фильтров могут быть быстро выполнены рекурсией. Это приводит к свертке изображения пропорционально только к времени *N 2*, полностью независимо от размера ФРТ. В других слова, изображение могут быть свернуты со столь же большим ФРТ как необходимо, с только несколько целочисленных операций в пиксел. Например, свертка изображения 512x512 требует только нескольких сотен миллисекунд на персональном компьютере. Это быстро! Не любите форму из этих двух ядер фильтра? Сверните изображение с одним из них *несколько раз*, чтобы аппроксимировать ФРТ Гауссиана (гарантируемый Центральной Предельной Теоремой, Глава 7). Они - большие алгоритмы, способные к схватившему успеху от захватов неудачи. Их стоит помнить.

Example of a Large PSF: Illumination Flattening
Пример Большого ФРТ: Выравнивание Освещения

A common application requiring a large PSF is the enhancement of images with unequal illumination. Convolution by separability is an ideal algorithm to carry out this processing. With only a few exceptions, the images seen by the eye are formed from *reflected* light. This means that a viewed image is equal to the reflectance of the objects multiplied by the ambient illumination. Figure 24-8 shows how this works. Figure (a) represents the *reflectance* of a scene being viewed, in this case, a series of light and dark bands. Figure (b) illustrates an example illumination signal, the pattern of light falling on (a). As in the real world, the illumination slowly varies over the imaging area. Figure (c) is the image seen by the eye, equal to the reflectance image, (a), multiplied by the illumination image, (b). The regions of poor illumination are difficult to view in (c)

for two reasons: they are too dark and their contrast is too low (the difference between the peaks and the valleys).

Обычное приложение, требующее большого ФРТ – расширение(увеличение) изображений с неравным освещением. Свертка отделимостью - идеальный алгоритм, чтобы выполнить эту обработку. С только несколько исключений, изображения, замеченные глазом сформированы из *отраженного* индикатора. Это означает, что просмотренное изображение равно коэффициенту отражения объектов, умноженных освещением фона. Рисунок 24-8 показывает, как это работает. Рисунок (a) представляет коэффициент отражения просматриваемой сцены, в этом случае, ряд световых и темных полос. Рисунок (b) иллюстрирует примера сигнал освещения, образец индикатора, падающего на (a). Как в реальном мире, освещение медленно изменяется по области отображения. Рисунок (c) - изображение, замеченное глазом, равным изображению коэффициента отражения, (a), умноженный изображением освещения, (b). Области плохого освещения трудны рассмотреть в (c) по двум причинам: Они слишком темны, и их контраст слишком низок (разность между пиками и минимумами).

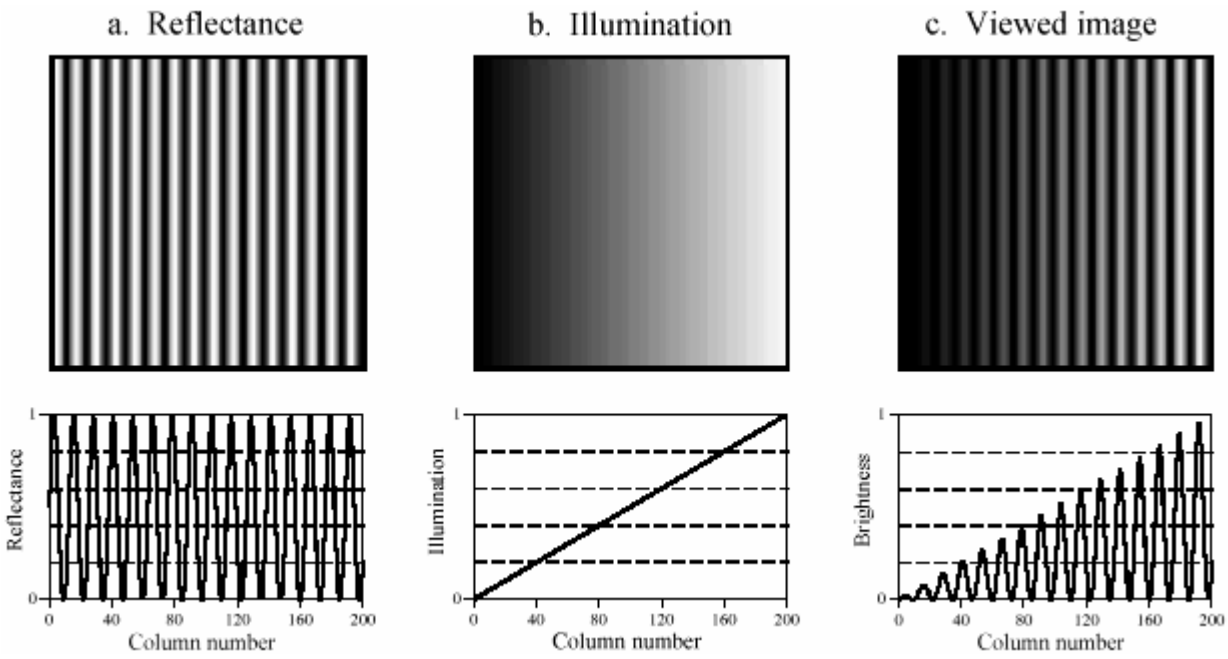


FIGURE 24-8
Model of image formation. A viewed image, (c), results from the multiplication of an illumination pattern, (b), by a reflectance pattern, (a). The goal of the image processing is to modify (c) to make it look more like (a). This is performed in Figs. (d), (e) and (f) on the opposite page.
РИСУНОК 24-8. Модель формирования изображения.
Просмотренное изображение, (c), следует из умножения образца освещения, (b), образцом коэффициента отражения, (a). Цель обработки изображения состоит в том, чтобы изменить (c), чтобы заставить это смотреть скорее (a). Это выполнено в рис. (d), (e) и (f) на противоположной странице(продолжении рисунка 24-8).

To understand how this relates to the problem of every day vision, imagine you are looking at two identically dressed men. One of them is standing in the bright sunlight, while the other is standing in the shade of a nearby tree. The percent of the incident light reflected from both men is the same. For instance, their faces might reflect 80% of the incident light, their gray shirts 40% and their dark pants 5%. The problem is, the illumination of the two might be, say, ten times different. This makes the image of the man in the shade ten times darker than the person in the sunlight, and the contrast (between the face, shirt, and pants) ten times less.

Чтобы понимать, как это касается, проблема зрения каждый день, вообразите, что Вы смотрите на двух тождественно перевязанных людей. Один из них стоит в ярком солнечном свете, в то время как другой стоит в оттенке близлежащего дерева. Процент от инцидентного индикатора, отраженного от обоих людей - тот же самый. Например, их лица могли бы отражать 80 % инцидентного индикатора, их серые рубашки 40 % и их темные штаны 5 %. Проблема, освещение этих двух могло бы быть, скажем, десять раз различно. Это делает изображение из человека в тени в десять раз темнее чем человек в солнечном свете, и контрасте (между лицом, рубашкой, и штанами) в десять раз меньше.

The goal of the image processing is to *flatten* the illumination component in the acquired image. In other words, we want the final image to be representative of the objects' reflectance, not the lighting conditions. In terms of Fig. 24-8, given (c), find (a). This is a nonlinear filtering problem, since the component images were combined by multiplication, not addition. While this separation cannot be performed perfectly, the improvement can be dramatic.

Цель обработки изображения(образа) состоит в том, чтобы *сгладить* компонент освещения в приобретенном изображении. Другими словами, мы хотим, чтобы конечное изображение было представительским из коэффициента отражения объектов, не условия(состояния) освещения. В терминах рис. 24-8, учитывая (c), находят (a). Это - нелинейная проблема фильтрации, так как составляющие изображения были объединены умножением, не сложением. В то время как это разделение не может быть выполнено совершенно, уточнение может быть впечатляющее.

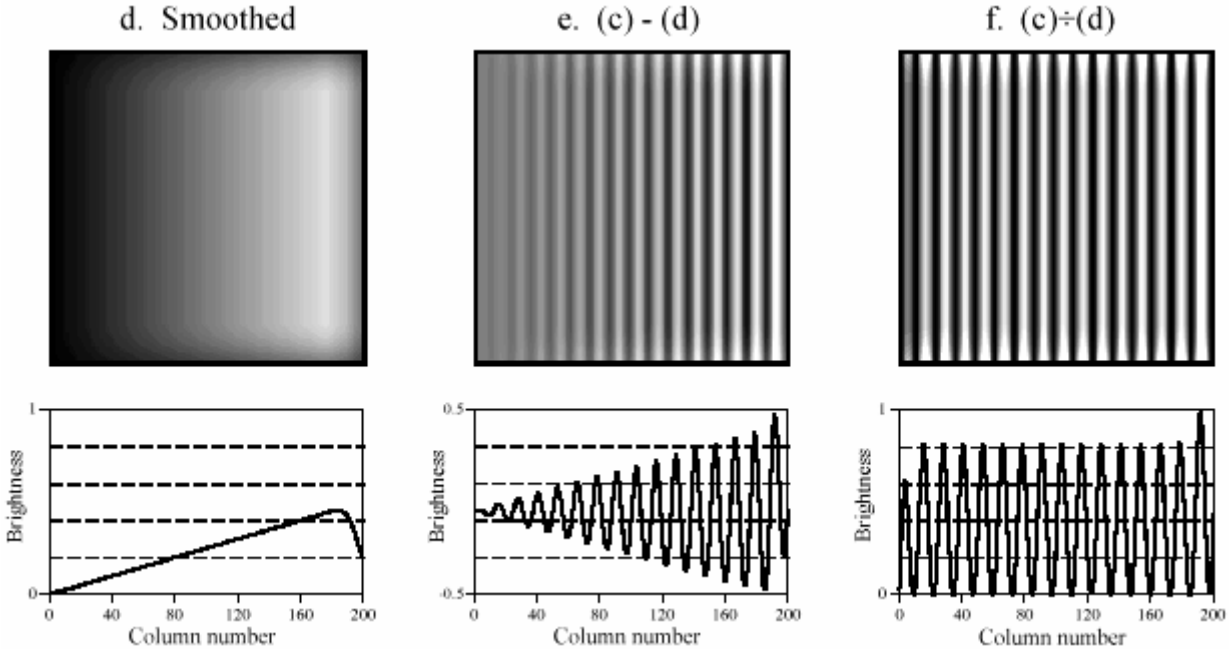


FIGURE 24-8 (continued)
Figure (d) is a smoothed version of (c), used as an approximation to the illumination signal. Figure (e) shows an approximation to the reflectance image, created by *subtracting* the smoothed image from the viewed image. A better approximation is shown in (f), obtained by the nonlinear process of *dividing* the two images.
Рисунок 24-8 (продолжение)
Рисунок (d) - приглаженная версия (c), используемого как аппроксимация к сигналу освещения. Рисунок (e) показывает аппроксимацию изображению коэффициента отражения, созданному, *вычитая* приглаженное изображение от просмотренного изображения. Лучшая аппроксимация показывается в (f), получена нелинейным процессом *разделения* двух изображений.

To start, we will convolve image (c) with a large PSF, one-fifth the size of the entire image. The goal is to eliminate the sharp features in (c), resulting in an approximation to the original illumination signal, (b). This is where convolution by separability is used. The exact shape of the PSF

is not important, only that it is much wider than the features in the reflectance image. Figure (d) is the result, using a Gaussian filter kernel.

Чтобы начинаться, мы свернем изображение (c) с большим ФРТ, пятая часть размера полного изображения. Цель состоит в том, чтобы устранить крутые(резкие) особенности в (c), приводя к аппроксимации к первоначальному сигналу освещения, (b). Это - то, где свертка отделимостью используется. Точная форма ФРТ не важна, только, что это является намного более широким чем особенности в изображении коэффициента отражения. Рисунок (d) - результат, используя ядро фильтра Гауссиана.

Since a smoothing filter provides an estimate of the illumination image, we will use an edge enhancement filter to find the reflectance image. That is, image (c) will be convolved with a filter kernel consisting of a delta function minus a Gaussian. To reduce execution time, this is done by subtracting the smoothed image in (d) from the original image in (c). Figure (e) shows the result. It doesn't work! While the dark areas have been properly lightened, the contrast in these areas is still terrible.

Так как фильтр сглаживания обеспечивает оценку освещения изображения, мы будем использовать фильтр расширения края, чтобы найти изображение коэффициента отражения. То есть изображение (c) будет свернуто с ядром фильтра, состоящим из отрицательной дельта функции Гауссиана. Чтобы приводить время выполнения, это сделано, вычитая приглаженное изображение в (d) от первоначального изображения в (c). Рисунок (e) показывает результат. Это не работает! В то время как темные области были должным образом осветлены), контраст в этих областях все еще ужасен.

Linear filtering performs poorly in this application because the reflectance and illumination signals were original combined by multiplication, not addition. Linear filtering cannot correctly separate signals combined by a nonlinear operation. To separate these signals, they must be *unmultiplied*. In other words, the original image should be *divided* by the smoothed image, as is shown in (f). This corrects the brightness and restores the contrast to the proper level.

Линейная фильтрация исполняет плохо в этом приложении, потому что коэффициент отражения и сигналы освещения были первоначальные объединены умножением, не слжением. Линейная фильтрация не может правильно отделить сигналы, объединенные нелинейной операцией. Чтобы отделить эти сигналы, они должны быть *не-умножены*. Другими словами, первоначальное изображение должно быть *разделено* приглаженным изображением, как показывается в (f). Это исправляет яркость и восстанавливает контраст к надлежащему уровню.

This procedure of dividing the images is closely related to **homomorphic processing**, previously described in Chapter 22. Homomorphic processing is a way of handling signals combined through a nonlinear operation. The strategy is to change the nonlinear problem into a linear one, through an appropriate mathematical operation. When two signals are combined by multiplication, homomorphic processing starts by taking the *logarithm* of the acquired signal. With the identity: $\log(a \times b) = \log(a) + \log(b)$, the problem of separating *multiplied* signals is converted into the problem of separating *added* signals. For example, after taking the logarithm of the image in (c), a linear high-pass filter can be used to isolate the logarithm of the reflectance image. As before, the quickest way to carry out the high-pass filter is to subtract a smoothed version of the image. The antilogarithm (exponent) is then used to undo the logarithm, resulting in the desired approximation to the reflectance image.

Эта процедура деления изображений близко связана с гомоморфной обработкой, предварительно описана в главе 22. Гомоморфная обработка - путь обработки сигналов, объединенных через нелинейную операцию. Стратегия состоит в том, чтобы заменить нелинейную проблему на линейную, через соответствующую математическую операцию. Когда два сигнала объединены умножением, гомоморфные запуском обработки, беря *логарифм* приобретенного сигнала. Тождественно: $\log(a \times b) = \log(a) + \log(b)$, проблема отделения *умноженных* сигналов преобразована в проблему отделения *сложенных* сигналов. Например, после взятия логарифма изображения в (c), линейный фильтр верхних частот может использоваться, чтобы изолировать логарифм изображения коэффициента отражения. Как прежде, самый быстрый способ выполнять фильтр верхних частот состоит в том, чтобы вычесть приглаженную версию изображения. Антилогарифм (экспонента) тогда используется, чтобы отменить логарифм, приводя к желательной аппроксимации к изображению коэффициента отражения.

Which is better, dividing or going along the homomorphic path? They are nearly the same, since taking the logarithm and subtracting is *equal* to dividing. The only difference is the approximation used for the illumination image. One method uses a smoothed version of the acquired image, while the other uses a smoothed version of the logarithm of the acquired image.

Который является лучше, разделяя или идя по гомоморфному пути? Они - почти, тот же самый, начиная с взятия логарифма и вычитания *равен* делению. Единственная разность - аппроксимация, используемая для изображения освещения. Один метод использует приглаженную версию приобретенного изображения, в то время как другие используют приглаженную версию логарифма приобретенного изображения.

This technique of flattening the illumination signal is so useful it has been incorporated into the neural structure of the eye. The processing in the middle layer of the retina was previously described as an edge enhancement or high-pass filter. While this is true, it doesn't tell the whole story. The first layer of the eye is nonlinear, approximately taking the *logarithm* of the incoming image. This makes the eye a homomorphic processor. Just as described above, the logarithm followed by a linear edge enhancement filter flattens the illumination component, allowing the eye to see under poor lighting conditions. Another interesting use of homomorphic processing occurs in photography. The density (darkness) of a negative is equal to the logarithm of the brightness in the final photograph. This means that any manipulation of the negative during the development stage is a type of homomorphic processing.

Эта методика выравнивания сигнала освещения настолько полезна, это было включено в невральную структуру глаза. Обработка в среднем уровне сетчатки была предварительно описана как подчеркивание контуров или фильтр верхних частот. В то время как это истинно, это не сообщает всю историю. Первый уровень глаза не линеен, приблизительно беря логарифм входящего изображения. Это делает глаз гомоморфным процессором. Также, как описано выше, логарифм, сопровождаемый линейным фильтром подчеркивания контуров сглаживает компонент освещения, позволяя глазу видеть при плохих условиях освещения. Другое интересное использование гомоморфной обработки происходит в фотографии. Плотность (темнота) негатива равна логарифму яркости на конечной фотографии. Это означает, что любая манипуляция негатива в течение стадии развития - тип гомоморфной обработки.

Before leaving this example, there is a nuisance that needs to be mentioned. As discussed in Chapter 6, when an N point signal is convolved with an M point filter kernel, the resulting signal is $N + M - 1$ points long. Likewise, when an MxM image is convolved with an NxN filter kernel, the result is an image. The problem is, it is often difficult to manage a changing image size. For

instance, the allocated memory must change, the video display must be adjusted, the array indexing may need altering, etc. The common way around this is to *ignore* it; if we start with a 512x512 image, we want to end up with a512x512 image. The pixels that do not fit within the original boundaries are discarded.

Перед оставлением этого примера, имеется помеха, которая должна быть упомянута. Как обсуждено в главе 6, когда сигнал точек N свернут с M точками ядра фильтра, заканчивающийся сигнал – N+M-1 точек длинной. Аналогично, когда изображение MxM свернуто(скручено) с ядром фильтра NxN, результат - изображение(M + N - 1) x (M + N - 1). Проблема, часто трудно управлять изменением размера изображения. Например, отведенная память должна измениться, дисплей видео должен быть откорректирован, индексация массива может нуждаться в изменении, и т.д. Обычный путь вокруг этого состоит в том, чтобы *игнорировать* это; если мы начинаем с изображения 512x512, мы хотим закончиться с изображением 512x512. Пикселы, которые не соответствуют пределам первоначальной границе изображения, отвергнуты.

While this keeps the image size the same, it doesn't solve the whole problem; these is still the *boundary condition* for convolution. For example, imagine trying to calculate the pixel at the upper-right corner of (d). This is done by centering the Gaussian PSF on the upper-right corner of (c). Each pixel in (c) is then multiplied by the corresponding pixel in the overlaying PSF, and the products are added. The problem is, three-quarters of the PSF lies outside the defined image. The easiest approach is to assign the undefined pixels a value of zero. This is how (d) was created, accounting for the dark band around the perimeter of the image. That is, the brightness smoothly decreases to the pixel value of zero, exterior to the defined image.

В то время как это сохраняет размер изображения тем же самым, это не решает всю проблему; они - все еще *ограничивающее условие* для свертки. Например, вообразите пробовать вычислять пиксел в правом верхнем углу (d). Это сделано, выравнивая по центру ФРТ Гауссиана в правом верхнем углу (c). Каждый пиксел в (c) тогда умножен на соответствующий пиксел в наложенном ФРТ, и продукт сложения. Проблема, три четверти ФРТ, лежит вне определенного изображения. Самый простой подход состоит в том, чтобы назначить неопределенным пикселам значение нуля. Это - то, как (d) был создан, составляя(объясняя) темную полосу по периметру изображения. То есть яркость гладко уменьшается к значению пиксела нуля, внешняя область к определенному изображению.

Fortunately, this dark region around the boarder can be corrected (although it hasn't been in this example). This is done by dividing each pixel in (d) by a correction factor. The correction factor is the fraction of the PSF that was immersed in the input image when the pixel was calculated. That is, to correct an individual pixel in (d), imagine that the PSF is centered on the corresponding pixel in (c). For example, the upper-right pixel in (c) results from only 25% of the PSF overlapping the input image. Therefore, correct this pixel in (d) by dividing it by a factor of 0.25. This means that the pixels in the center of (d) will not be changed, but the dark pixels around the perimeter will be brightened. To find the correction factors, imagine convolving the filter kernel with an image having all the pixel values equal to *one*. The pixels in the resulting image are the correction factors needed to eliminate the edge effect.

К счастью, эта темная область вокруг бордюра может быть исправлена (хотя это не было в этом примере). Это сделано, разделяя каждый пиксел в (d) на поправочный коэффициент. Поправочный коэффициент - дробь(доля) ФРТ, который был поглощен во входном изображении, когда пиксел был рассчитан. То есть чтобы исправлять индивидуальный пиксел в (d), вообразите, что ФРТ центрирован на соответствующем пикселе в (c). Например, правый верхний пиксел в (c) следует только из 25 % ФРТ, накладывающегося на входное

изображение. Поэтому, исправьте этот пиксел в (d), разделяя это коэффициентом 0.25. Это означает, что пикселы в центре (d) не будут изменены, но темные пикселы вокруг периметра будут украшены. Чтобы находить поправочные коэффициенты, вообразите свертывание ядра фильтра с изображением, имеющим все значения пиксела, равные единице. Пикселы в заканчивающемся изображении - поправочные коэффициенты, необходимые, чтобы устранить краевой эффект.

Fourier Image Analysis
Анализ Изображения Фурье

Fourier analysis is used in image processing in much the same way as with one-dimensional signals. However, images do not have their information encoded in the frequency domain, making the techniques much less useful. For example, when the Fourier transform is taken of an *audio* signal, the confusing time domain waveform is converted into an easy to understand frequency spectrum. In comparison, taking the Fourier transform of an image converts the straightforward information in the spatial domain into a scrambled form in the frequency domain. In short, don't expect the Fourier transform to help you understand the information encoded in images.

Анализ Фурье используется в обработке изображения аналогичным способом как с одномерными сигналами. Однако, изображения не кодируют их информацию в частотном домене, делая методы, намного менее полезными. Например, когда преобразование Фурье(трансформанта Фурье) принятого аудио-сигнала, запутывающая форма волны домена времени преобразована в простой понять спектр частот. Для сравнения, беря преобразование Фурье(трансформанту Фурье) изображения, преобразовывает прямую информацию в пространственной области(пространственном домене) в борющуюся(зашифрованную; скремблированную; рандомизированную) форму в частотном домене. Короче говоря, не ожидайте, что преобразование Фурье поможет Вам понять информацию, закодированную в изображениях.

Likewise, don't look to the frequency domain for filter design. The basic feature in images is the edge, the line separating one *object* or *region* from another *object* or *region*. Since an edge is composed of a wide range of frequency components, trying to modify an image by manipulating the frequency spectrum is generally not productive. Image filters are normally designed in the spatial domain, where the information is encoded in its simplest form. Think in terms of *smoothing* and *edge enhancement* operations (the spatial domain) rather than *high-pass* and *low-pass* filters (the frequency domain).

Аналогично, не обратитесь к частотному домену для проекта фильтра. Основная особенность в изображениях - край, строка(линия), отделяющая один *объект* или *область* от другого *объекта* или *области*. Так как край составлен из широкого диапазона частотных компонентов, пробуя изменять изображение, управляя спектром частот вообще не производительно. Фильтры изображения обычно разрабатываются в пространственном домене, где информация закодирована в ее самой простой форме. Думайте в терминах операций *сглаживания* и *подчеркивания* контуров (пространственный домен) скорее чем *фильтр верхних частот* и *фильтр нижних частот* (частотный домен).

In spite of this, Fourier image analysis does have several useful properties. For instance, *convolution* in the spatial domain corresponds to *multiplication* in the frequency domain. This is important because multiplication is a simpler mathematical operation than convolution. As with one-dimensional signals, this property enables FFT convolution and various deconvolution tech-

niques. Another useful property of the frequency domain is the *Fourier Slice Theorem*, the relationship between an image and its projections (the image viewed from its sides). This is the basis of *computed tomography*, an x-ray imaging technique widely used medicine and industry.

Несмотря на это, анализ изображения Фурье имеет несколько полезных свойств. Например, *свертка* в пространственном домене соответствует *умножению* в частотном домене. Это важно, потому что умножение - более простая математическая операция, чем свертка. Как и с одномерными сигналами, это свойство допускает свертке БПФ и различным методам деконволюции. Другое полезное свойство частотного домена - *Теорема Сектора Фурье*, отношения между изображением и его проекциями (изображение, просмотренное от его сторон). Это - основание *копьютерной томографии*, рентген, отображающий методику широко используемую в медицине и промышленности.

The frequency spectrum of an image can be calculated in several ways, but the FFT method presented here is the only one that is practical. The original image must be composed of *N* rows by *N* columns, where *N* is a power of two, i.e., 256, 512, 1024, etc. If the size of the original image is not a power of two, pixels with a value of zero are added to make it the correct size. We will call the two-dimensional array that holds the image the **real array**. In addition, another array of the same size is needed, which we will call the **imaginary array**.

Спектр частот изображения может быть рассчитан несколькими способами, но методом БПФ, представленным имеется единственный, который является практическим. Первоначальное изображение должно быть составлено из *N* строк на *N* столбцов, где *N* - мощь два, то есть, 256, 512, 1024, и т.д. Если размер первоначального изображения - не, мощь два, пикселы со значением нуля добавлены, чтобы делать это правильным размером. Мы назовем двумерный массив, который проводит(держит) изображение реальным(вещественным) массивом. Кроме того, другой массив того же самого размера необходим, которым мы назовем мнимый(несобственный) массив.

The recipe for calculating the Fourier transform of an image is quite simple: take the one-dimensional FFT of each of the rows, followed by the one-dimensional FFT of each of the columns. Specifically, start by taking the FFT of the *N* pixel values in row 0 of the real array. The real part of the FFT's output is placed back into row 0 of the real array, while the imaginary part of the FFT's output is placed into row 0 of the imaginary array. After repeating this procedure on rows 1 through *N* - 1, both the real and imaginary arrays contain an intermediate image. Next, the procedure is repeated on each of the *columns* of the intermediate data. Take the *N* pixel values from column 0 of the real array, *and* the *N* pixel values from column 0 of the imaginary array, and calculate the FFT. The real part of the FFT's output is placed back into column 0 of the real array, while the imaginary part of the FFT's output is placed back into column 0 of the imaginary array. After this is repeated on columns 1 through *N* - 1, both arrays have been overwritten with the image's frequency spectrum.

Рецепт для вычисления трансформанты Фурье изображения весьма прост: берите одномерное БПФ каждой из строк, сопровождаемое одномерным БПФ каждого из столбцов. Определенно, начало, возьмем БПФ *N* значений пиксела в строке 0 из реального массива. Реальная(вещественная) часть выхода БПФ помещена назад в строку 0 реального(вещественного) массива, в то время как мнимая часть выхода БПФ помещена в строку 0 мнимого массива. После повторения этой процедуры на строках 1 через *N* - 1, и реальные(вещественные) и мнимые массивы содержат промежуточное изображение. Затем, процедура повторена на каждом из *столбцов* промежуточных данных. Возьмем *N* значений пиксела от столбца 0 вещественного массива, *и N* значений пиксела из столбца 0 мнимого массива, и вычислим БПФ. Вещественная часть выхода БПФ помещена назад в

столбец 0 вещественного массива, в то время как мнимая часть выхода БПФ помещена назад в столбец 0 мнимого массива. После того, как это повторено на столбцах 1 через *N*-1, оба массива были записаны поверх и спектра частот изображения.

Since the vertical and horizontal directions are equivalent in an image, this algorithm can also be carried out by transforming the columns first and then transforming the rows. Regardless of the order used, the result is the same. From the way that the FFT keeps track of the data, the amplitudes of the low frequency components end up being at the corners of the two-dimensional spectrum, while the high frequencies are at the center. The inverse Fourier transform of an image is calculated by taking the inverse FFT of each row, followed by the inverse FFT of each column (or vice versa).

Так как вертикальные и горизонтальные направления эквивалентны в изображении, этот алгоритм может также быть выполнен, преобразовывая столбцы сначала и затем преобразовывая строки. Независимо от используемого порядка, результат - тот же самый. От пути, за которым БПФ следит, данные, амплитуды низкочастотных компонентов заканчивают находиться в углах двумерного спектра, в то время как высокие частоты - в центре. Обратное Преобразование Фурье изображения рассчитана, беря обратное БПФ каждой строки, сопровождается обратным БПФ каждого столбца (или наоборот).
Figure 24-9 shows an example Fourier transform of an image. Figure (a) is the original image, a microscopic view of the input stage of a 741 op amp integrated circuit. Figure (b) shows the real and imaginary parts of the frequency spectrum of this image. Since the frequency domain can contain negative pixel values, the grayscale values of these images are offset such that negative values are dark, zero is gray, and positive values are light. The low-frequency components in an image are normally much larger in amplitude than the high-frequency components. This accounts for the very bright and dark pixels at the four corners of (b). Other than this, the spectra of *typical* images have no discernable order, appearing random. Of course, images can be *contrived* to have any spectrum you desire.

Рисунок 24-9 показвает пример преобразования изображения преобразованием Фурье(трансформантой Фурье). Рисунок (a) - первоначальное изображение, микроскопическое представление(вид) входного каскада интегральной схемы операционного усилителя 741. Рисунок (b) показывает реальные(действительные) и мнимые части спектра частот этого изображения. Так как частотный домен может содержать отрицательные значения пиксела, полутоновые значения этих изображений смещены так, что отрицательные значения являются темными, нулевые - серыми, и положительные значения, светлыми. Низкочастотные компоненты в изображении обычно намного большие в амплитуде чем высокочастотные компоненты. Это объясняет самые яркие и темные пикселы в четырех углах (b). Другие чем эти, спектры *типичных* изображений не имеют никакого видимого(различимого; заметного) порядка(заказа), появляясь случайно. Конечно, изображения могут быть *изобретены*, чтобы иметь любой спектр, которого Вы желаете.

As shown in (c), the polar form of an image spectrum is only slightly easier to understand. The low-frequencies in the magnitude have large positive values (the white corners), while the high-frequencies have small positive values (the black center). The phase looks the same at low and high-frequencies, appearing to run randomly between $-\pi$ and $-\pi$ radians.

Как показано в (c), полярная форма спектра изображения только слегка проще, чтобы понять. Низкие частоты в величине имеют большие положительные значения (белые углы), в то время как высокие частоты имеют маленькие положительные значения (черный центр). Фаза выглядит одинаково на низких и высоких частотах, появляющихся беспорядочно между $-\pi$ и $-\pi$ радианами.

Figure (d) shows an alternative way of displaying an image spectrum. Since the spatial domain contains a *discrete* signal, the frequency domain is *periodic*. In other words, the frequency domain arrays are duplicated an infinite number of times to the left, right, top and bottom. For instance, imagine a tile wall, with each tile being the N x N magnitude shown in (c). Figure (d) is also an N x N section of this tile wall, but it straddles four tiles; the center of the image being where the four tiles touch. In other words, (c) is the same image as (d), except it has been shifted $N/2$ pixels horizontally (either left or right) and $N/2$ pixels vertically (either up or down) in the periodic frequency spectrum. This brings the bright pixels at the four corners of (c) together in the center of (d).

Рисунок (d) показывает альтернативный путь отображения спектра изображения. Так как пространственный домен содержит *дискретный* сигнал, частотный домен *периодический*. Другими словами, массивы частотных доменов дублированы бесконечное число раз налево, право, верхняя и нижняя граница. Для образца, вообразите границу неперекрывающего расположения, с каждым неперекрывающим расположением, являющимся величиной N x N, показанной в (c). Рисунок (d) - также N x N раздел этой стенки(границы) неперекрывающего расположения, но это колеблется между четырьмя неперекрывающими расположениями; центр изображения, являющегося, где эти четыре касание неперекрывающих расположений. Другими словами, (c) - то же самое изображение, как (d), кроме этого был сдвинутые $N/2$ пикселы горизонтально (или левый или правый) и $N/2$ пикселы вертикально (любой вверх или вниз) в периодическом спектре частот. Это приводит к ярким пикселам в четырех углах (c) вместо в центре (d).

FIGURE 24-9
Frequency spectrum of an image. The example image, shown in (a), is a microscopic photograph of the silicon surface of an integrated circuit. The frequency spectrum can be displayed as the real and imaginary parts, shown in (b), or as the magnitude and phase, shown in (c). Figures (b) & (c) are displayed with the low-frequencies at the corners and the high-frequencies at the center. Since the frequency domain is periodic, the display can be rearranged to reverse these positions. This is shown in (d), where the magnitude and phase are displayed with the low-frequencies located at the center and the high-frequencies at the corners.

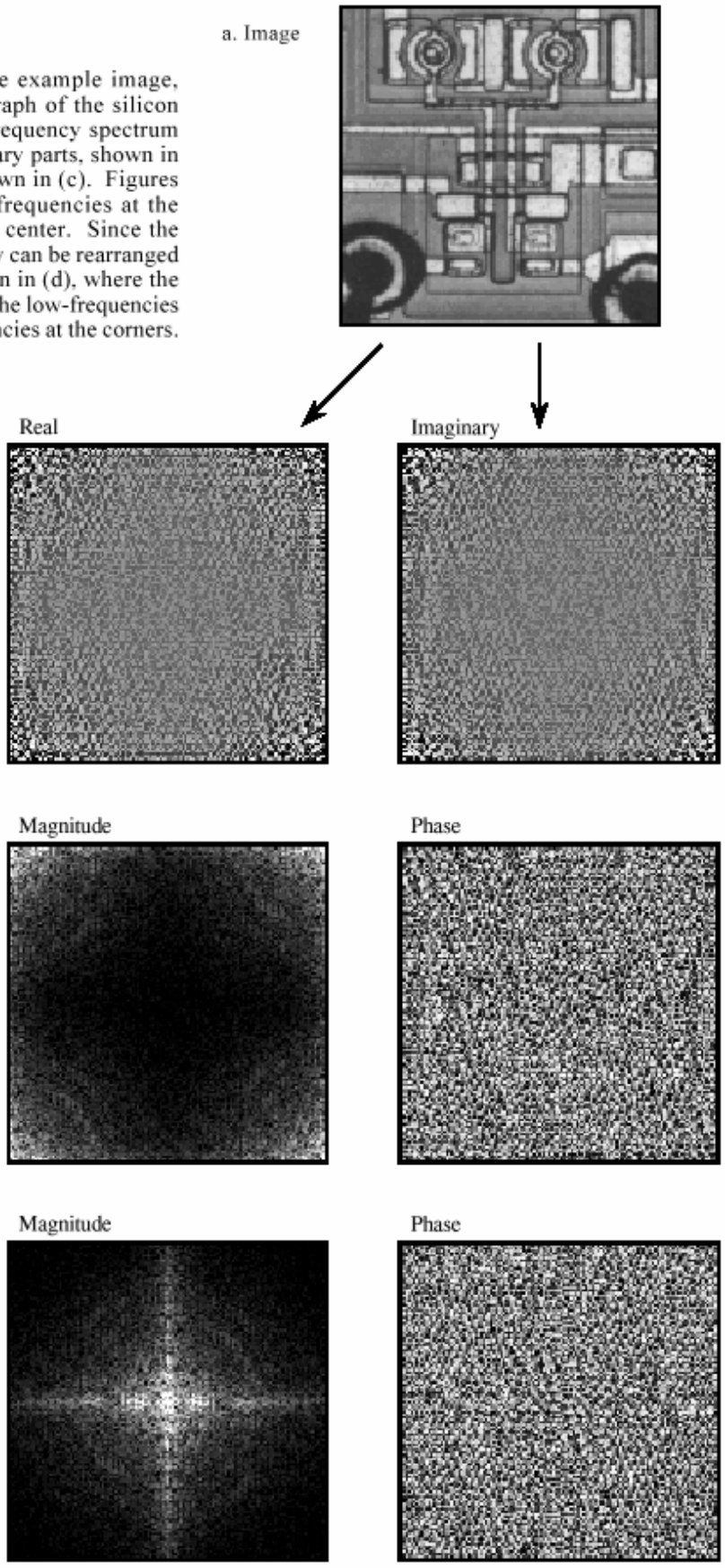


FIGURE 24-9
Frequency spectrum of an image. The example image, shown in (a), is a microscopic photograph of the silicon surface of an integrated circuit. The frequency spectrum can be displayed as the real and imaginary parts, shown in (b),

or as the magnitude and phase, shown in (c). Figures (b) & (c) are displayed with the low-frequencies at the corners and the high-frequencies at the center. Since the frequency domain is periodic, the display can be rearranged to reverse these positions. This is shown in (d), where the magnitude and phase are displayed with the low-frequencies located at the center and the high-frequencies at the corners.

c. Frequency spectrum displayed in polar form (as the magnitude and phase).

d. Frequency spectrum displayed in polar form, with the spectrum shifted to place zero frequency at the center.

b. Frequency spectrum displayed in rectangular form (as the real and imaginary parts).

РИСУНОК 24-9
Спектр частот изображения. Пример изображения, показанного в (a), является микроскопической фотографией поверхности кремниевой интегральной схемы. Спектр частот может быть отображен как реальные и мнимые части, показан в (b), или как величина и фаза, показан в (c). Рисунки (b) и (c) отображены с низкими частотами в углах и высоких частотах в центре. Так как частотный домен периодический, дисплей может быть перестроен, чтобы полностью изменить эти позиции. Это показано в (d), где величина и фаза отображены низкими частотами, расположенными в центре и высокими частотами в углах.

c. Спектр частот, отображенный в полярной форме (как величина и фаза).

d. Спектр частот, отображенный в полярной форме, со спектром, сдвинутым, чтобы разместить нулевую частоту в центр.

b. Спектр частот, отображенный в прямоугольной форме (как реальные и мнимые части).

Figure 24-10 illustrates how the two-dimensional frequency domain is organized (with the low-frequencies placed at the corners). Row $N/2$ and column $N/2$ break the frequency spectrum into four quadrants. For the real part and the magnitude, the upper-right quadrant is a mirror image of the lower-left, while the upper-left is a mirror image of the lower-right. This symmetry also holds for the imaginary part and the phase, except that the values of the mirrored pixels are opposite in sign. In other words, every point in the frequency spectrum has a matching point placed symmetrically on the other side of the center of the image (row $N/2$ and column $N/2$). One of the points is the *positive* frequency, while the other is the matching frequency, as discussed in Chapter 10 for one-dimensional signals. In equation form, this symmetry is expressed as:

Рисунок 24-10 иллюстрирует, как двумерный частотный домен организован (с низкими частотами, помещенными в углы). Строка $N/2$ и Столбец $N/2$ разбивают спектр частот в четыре квадранта. Для реальной части и величины, правый верхний квадрант - зеркальное изображение левых нижних, в то время как левый верхний - зеркальное изображение нижних правых. Эта симметрия также держится для мнимой части и фазы, за исключением того, что значения отраженных пикселов - противоположны по знаку. Другими словами, каждая точка в спектре частот помещает точку соответствия симметрично с другой стороны центра изображения (строка $N/2$ и столбец $N/2$). Одна из точек - положительная частота, в то время как другая - частота соответствия, как обсуждено в главе 10 для одномерных сигналов. В форме уравнения, эта симметрия выражена как:

УРАВНЕНИЕ 24-2. Симметрия двумерного частотного домена. Эти уравнения могут использоваться в обоих форматах, когда низкие частоты отображены в углах, или при смещение размещает их в центр. В полярной форме, величина имеет ту же самую симметрию как вещественная часть, в то время как фаза имеет ту же самую симметрию как мнимая часть

$$Re\,X[r,c] = Re\,X[N-r,N-c]$$

$$Im\,X[r,c] = -Im\,X[N-r,N-c]$$

These equations take into account that the frequency spectrum is periodic, repeating itself every N samples with indexes running from 0 to N - 1. In other words $X[r, N]$, should be interpreted as $X[r, 0]$, $X[N, c]$ as $X[0, c]$, and $X[N, N]$ as $X[0, 0]$. This symmetry makes four points in the

spectrum match with *themselves*. These points are located at: [0, 0], [0, *N*/2], [*N*/2, 0] and [*N*/2, *N*/2].

Эти уравнения принимают во внимание, что спектр частот периодический, повторяя себя каждые *N* выборок с индексами, выполняющимися от 0 до *N*-1. Другими словами *X* [*r*, *N*], должен интерпретироваться как *X* [*r*, 0], *X* [*N*, *c*] как *X* [0, *c*], и *X* [*N*, *N*] как *X* [0, 0]. Эта симметрия делает четыре точки в соответствии спектра с собой. Эти точки расположены в: [0, 0], [0, *N*/2], [*N*/2, 0] и [*N*/2, *N*/2].

Each pair of points in the frequency domain corresponds to a sinusoid in the spatial domain. As shown in (a), the value [0, 0] at corresponds to the *zero frequency* sinusoid in the spatial domain, i.e., the DC component of the image. There is only one point shown in this figure, because this is one of the points that is its own match. As shown in (b), (c), and (d), other pairs of points correspond to two-dimensional sinusoids that look like waves on the ocean. One-dimensional sinusoids have a *frequency*, *phase*, and *amplitude*. Two dimensional sinusoids also have a *direction*.

Каждая пара точек в частотном домене соответствует синусоиде в пространственном домене. Как показано в (a), значение в [0, 0] переписывается к синусоиде *нулевой частоты* в пространственном домене, то есть, компонент постоянного тока изображения. Имеется только одна точка, показанная в этом рисунке, потому что это - одна из точек, которая является ее собственным соответствием. Как показано в (b), (c), и (d), другие пары точек соответствуют двумерным синусоидам, которые напоминают волны на океане. Одномерные синусоиды имеют *частоту*, *фазу*, и *амплитуду*. Две размерных синусоиды также имеют направление.

The frequency and direction of each sinusoid is determined by the location of the pair of points in the frequency domain. As shown, draw a line from each point to the *zero frequency* location at the outside corner of the quadrant that the point is in, i.e., [0, 0], [0, *N*/2], [*N*/2, 0], or [*N*/2, *N*/2] (as indicated by the circles in this figure). The direction of this line determines the direction of the spatial sinusoid, while its length is proportional to the frequency of the wave. This results in the low frequencies being positioned near the corners, and the high frequencies near the center.

Частота и направление каждой синусоиды определены расположением пары точек в частотном домене. Как показано, проведите линию от каждой точки до расположения *нулевой частоты* во внешнем углу квадранта, что точка находится в, то есть, [0, 0], [0, *N*/2], [*N*/2, 0], или [*N*/2, *N*/2] (как обозначено кругами в этом рисунке). Направление этой строки определяет направление пространственной синусоиды, в то время как ее длина пропорциональна к частоте волны. Это приводит к низким частотам, позиционируемым около углов, и высоких частот около центра.

When the spectrum is displayed with zero frequency at the center (Fig. 24-9d), the line from each pair of points is drawn to the DC value at the *center* of the image, i.e., [*N*/2, *N*/2]. This organization is simpler to understand and work with, since all the lines are drawn to the same

point. Another advantage of placing zero at the center is that it matches the frequency spectra of *continuous* images. When the spatial domain is continuous, the frequency domain is *aperiodic*. This places zero frequency at the center, with the frequency becoming higher in all directions out to infinity. In general, the frequency spectra of discrete images are displayed with zero frequency at the center whenever people will view the data, in textbooks, journal articles, and algorithm documentation. However, most calculations are carried out with the computer arrays storing data in the other format (low-frequencies at the corners). This is because the FFT has this format.

Когда спектр отображен с нулевой частотой в центре (рис. 24-9d), строка от каждой пары точек выведена к значению постоянного тока в *центре* изображения, то есть, [*N*/2, *N*/2]. Эта организация более простая понимать и работать с, так как все строки выведены к той же самой точке. Другое преимущество размещения нуля в центре состоит в том, что это соответствует частотным спектрам *непрерывных* изображений. Когда пространственный домен непрерывен, частотный домен апериодический. Это размещает нулевую частоту в центр, с частотой, становящейся выше во всех направлениях от и к бесконечности. Вообще, частотные спектры дискретных изображений отображены с нулевой частотой в центре всякий раз, когда люди рассмотрят данные, в учебниках, статьях журнала, и документации алгоритма. Однако, большинство вычислений выполнено с компьютерными массивами, сохраняющими данные в другом формате (низкие частоты в углах). Это - то, потому что БПФ имеет этот формат.

Even with the FFT, the time required to calculate the Fourier transform is a tremendous bottleneck in image processing. For example, the Fourier transform of a 512x512 image requires several minutes on a personal computer. This is roughly 10,000 times slower than needed for real time image processing, 30 frames per second. This long execution time results from the massive amount of information contained in images. For comparison, there are about the same number of *pixels* in a typical image, as there are *words* in this book. Image processing via the frequency domain will become more popular as computers become faster. This is a twenty-first century technology; watch it emerge!

Даже с БПФ, время, требуемое, чтобы вычислить Преобразование Фурье(трансформанту Фурье) - огромное узкое место в обработке изображения(образа). Например, преобразование Фурье изображения 512x512 требует нескольких минут на персональном компьютере. Это - грубо в 10000 раз медленнее, чем необходимо для обработки изображения реального времени, 30 кадров в секунду. Это длинное время выполнения следует из массивного количества информации, содержащейся в изображениях. Для сравнения, имеется относительно того же самого число *пикселов* в типичном изображении, как имеются слова в этой книге. Обработка изображения через частотный домен станет более популярной, поскольку компьютеры станут быстрее. Это – технология двадцать первого столетия; час этого пробил!

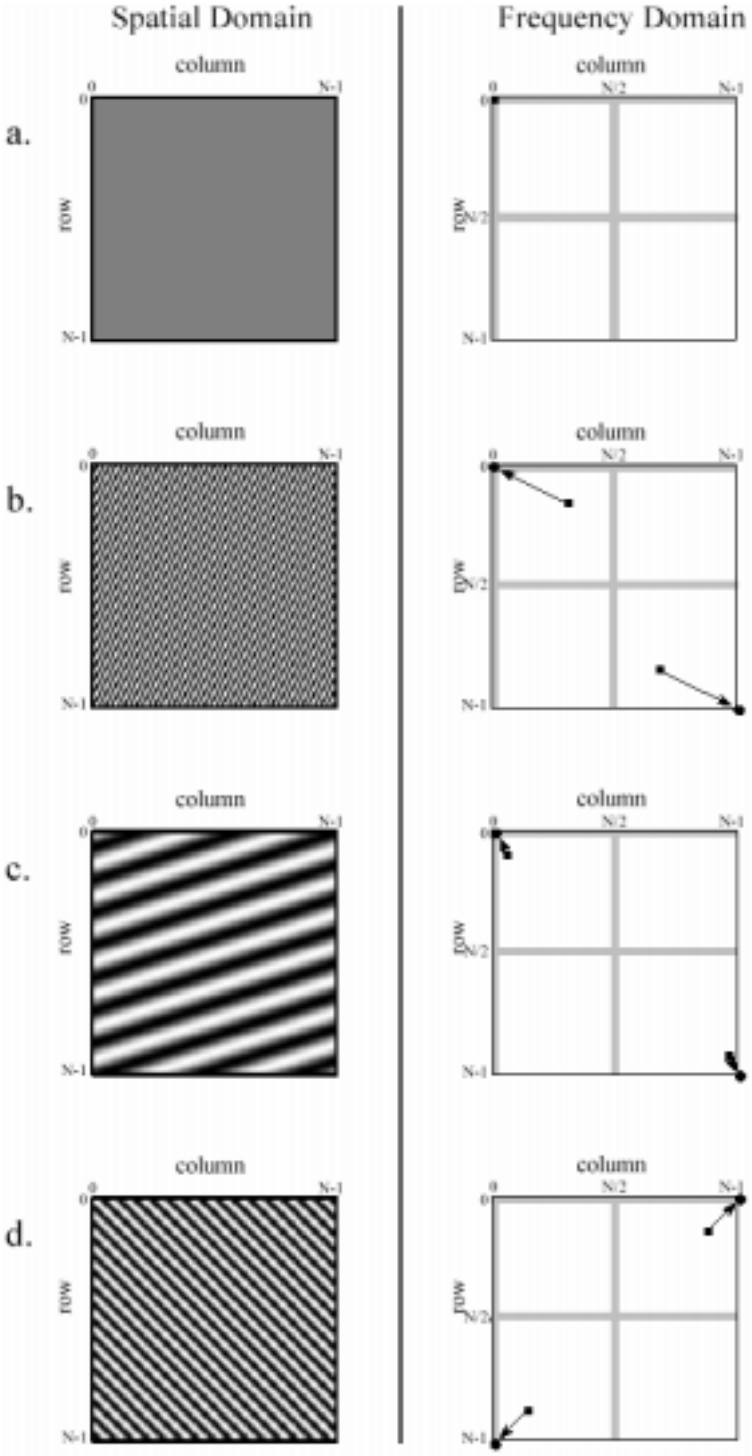


FIGURE 24-10
Two-dimensional sinusoids. Image sine and cosine waves have both a *frequency* and a *direction*. Four examples are shown here. These spectra are displayed with the low-frequencies at the corners. The circles in these spectra show the location of zero frequency.

РИСУНОК 24-10.
Двумерные синусоиды. Изображения волн синуса и косинуса имеют, и *частоту* и *направление*. Здесь показано четыре примера спектров. Эти спектры отображены с низкими частотами в углах. Кругами в этих спектрах показано расположение нулевой частоты.

FFT Convolution
Конволюция(Свертка) БПФ

Even though the Fourier transform is slow, it is still the fastest way to convolve an image with a large filter kernel. For example, convolving a 512x512 image with a 50x50 PSF is about 20 times faster using the FFT compared with conventional convolution. Chapter 18 discusses how FFT convolution works for one-dimensional signals. The two-dimensional version is a simple extension.

Даже при том, что преобразование Фурье медленно, это все же самый быстрый способ свернуть изображение с большим ядром фильтра. Например, свертывание изображения 512x512 с ФРТ 50x50 – приблизительно в 20 раз, быстрее по сравнению с использованием обычной свертки. Глава 18 обсуждает, как свертка БПФ работает для одномерных сигналов. Двумерная версия - простое вытяжение(расширение).

We will demonstrate FFT convolution with an example, an algorithm to locate a predetermined pattern in an image. Suppose we build a system for inspecting one-dollar bills, such as might be used for printing quality control, counterfeiting detection, or payment verification in a vending machine. As shown in Fig. 24-11, a 100x100 pixel image is acquired of the bill, centered on the portrait of George Washington. The goal is to search this image for a known pattern, in this example, the 29x29 pixel image of the face. The problem is this: given an acquired image and a known pattern, what is the most effective way to locate where (or if) the pattern appears in the image? If you paid attention in Chapter 6, you know that the solution to this problem is *correlation* (a matched filter) and that it can be implemented by using *convolution*.

Мы демонстрируем свертку БПФ на примере, алгоритма, чтобы расположить предопределенный образец в изображении. Предположим, что мы формируем систему для осмотра одно-долларовых счетов(векселей), типа, которая могла бы использоваться для контроля качества печати, обнаружения подделки, или верификацию (проверку; контроль) оплаты в торговом автомате. Как показано в рис. 24-11, приобретенное изображение счета(векселя) 100 x 100 пикселов, с портретом Джорджа Вашингтона в центре. Цель состоит в том, чтобы искать изображение известного образца, в этом примере, изображение лица 29x29 пикселов. Проблема - это: учитывая приобретенное изображение и известный образец, что наиболее эффективный путь состоит в том чтобы где расположить (или если расположен) где образец появляется в изображении? Если Вы обратили внимание в главе 6, Вы знаете, что решение этой проблемы - *корреляция* (согласованный фильтр?) и что это может быть осуществлено, используя *свертку*.

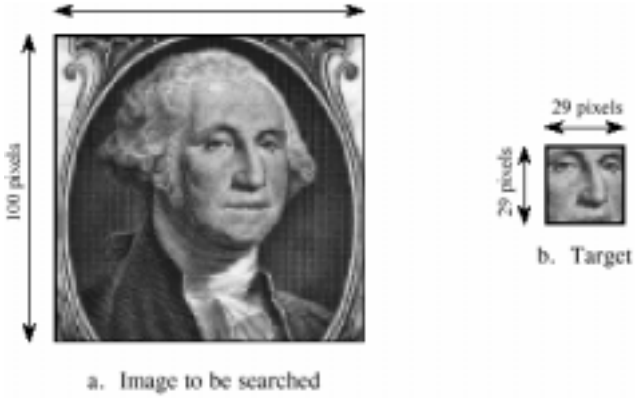
Before performing the actual convolution, there are two modifications that need to be made to turn the target image into a PSF. These are illustrated in Fig. 24-12. Figure (a) shows the target signal, the pattern we are trying to detect. In (b), the image has been rotated by 180º, the same as being flipped left-for-right and then flipped top-for-bottom. As discussed in Chapter 7, when performing *correlation* by using *convolution*, the target signal needs to be reversed to counteract the reversal that occurs during convolution. We will return to this issue shortly.

Перед выполнением фактической свертки, необходимо выполнить две модификации, которые должны быть сделаны, чтобы повернуть целевое изображение в ФРТ. Они иллюстрированы в рис. 24-12. Рисунок (a) показывает целевой сигнал, образец который мы пробуем обнаруживать. В (b), изображение повернуто на 180º, тот же самое как зеркально отразилить слева направо и затем зеркально отразить " голова - ноги ". Как обсуждено в главе 7, при выполнении *корреляции*, используя *свертку*, целевой сигнал должен быть ревер-

сирован, чтобы противодействовать инверсии, которая происходит в течение свертки. Мы возвратимся этой проблеме вскоре.

FIGURE 24-11
Target detection example. The problem is to search the 100100 pixel image of George Washington, (a), for the target pattern, (b), the 29x29 pixel face. The optimal solution is *correlation*, which can be carried out by *convolution*.

РИСУНОК 24-11
Пример обнаружения цели. Проблема состоит в том, чтобы искать изображение Джорджа Вашингтона 100x100 пикселов, (a), по целевому образцу лица 29 x 29 пикселов, (b). Оптимальное решение - корреляция, которая может быть выполнена сверткой.



The second modification is a trick for improving the effectiveness of the algorithm. Rather than trying to detect the face in the original image, it is more effective to detect the *edges of the face* in the *edges of the original image*. This is because the edges are sharper than the original features, making the correlation have a sharper peak. This step isn't required, but it makes the results significantly better. In the simplest form, a 3x3 edge detection filter is applied to both the original image and the target signal before the correlation is performed. From the associative property of convolution, this is the same as applying the edge detection filter to the target signal *twice*, and leaving the original image alone. In actual practice, applying the edge detection 3Ч3 kernel only once is generally sufficient. This is how (b) is changed into (c) in Fig. 24-12. This makes (c) the PSF to be used in the convolution

Вторая модификация - уловка для улучшения эффективность алгоритма. Скорее чем попытка обнаруживать лицо в первоначальном изображении, это более эффективно обнаружить *грани лица* в *гранях первоначального изображения*. Это - то, потому что грани более острые, чем первоначальные особенности, делая корреляцию иметь более острый пик. Этот шаг не требуется, но это делает результаты значительно лучше. В самой простой форме, обнаружения края применяется фильтр 3x3, и к первоначальному изображению и целевому сигналу прежде, чем корреляция выполнена. Из ассоциативного свойства свертки, это - тот же самое как применение фильтра обнаружения края к целевому сигналу *дважды*, и оставления одного первоначального изображения. В фактической практике, применяя обнаружение края 3x3 ядро только однажды вообще достаточно. Это - то, как (b) изменен в (c) в рис. 24-12. Это делает ФРТ (c), который нужно использовать в свертке

Figure 24-13 illustrates the details of FFT convolution. In this example, we will convolve image (a) with image (b) to produce image (c). The fact that these images have been chosen and pre-processed to implement *correlation* is irrelevant; this is a flow diagram of *convolution.* The first step is to pad both signals being convolved with enough zeros to make them a power of two in size, and big enough to hold the final image. For instance, when images of 100x100 and 29x29 pixels are convolved, the resulting image will be 128x128 pixels. Therefore, enough zeros must be added to (a) and (b) to make them each 128x128 pixels in size. If this isn't done, circular convolution takes place and the final image will be distorted. If you are having trouble understanding these concepts, go back and review Chapter 18, where the one-dimensional case is discussed in more detail.

Рисунок 24-13 иллюстрирует подробности свертки БПФ. В этом примере, мы свернем изображение (a) с изображением (b), чтобы произвести изображение (c). Факт, что эти изображения были выбраны и предварительно обработаны, чтобы осуществить *корреляцию*, нерелевантной(несоответствующей; Не имеющей значения); это - блок-схема свертки. Первый шаг должен дополнить оба сигнала, свертываемые с достаточными нулями, чтобы делать их мощью два в размере, и достаточно большой, чтобы провести(держать) конечное изображение. Например, когда изображения 100x100 и 29x29 пикселов свернуты, заканчивающееся изображение будет 128x128 пикселов. Поэтому, достаточно нулей должны быть добавлены к (a) и (b), чтобы делать их каждым 128x128 пикселов в размере. Если это не сделано, круговая(циклическая) свертка имеет место, и конечное изображение будет искажено. Если Вы имеете неприятность, понимая эти концепции, возвратитесь, и делайте обзор главы 18, где одномерный случай обсужден более подробно.

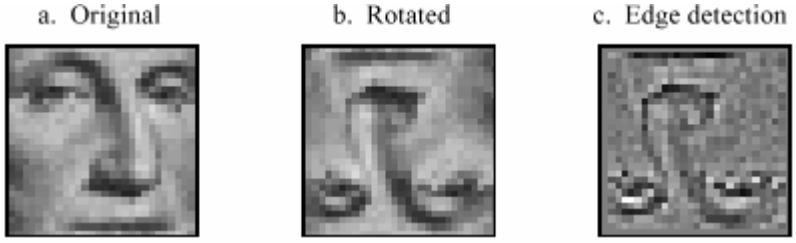


FIGURE 24-12
Development of a correlation filter kernel. The target signal is shown in (a). In (b) it is rotated by 180º to undo the rotation inherent in convolution, allowing correlation to be performed. Applying an edge detection filter results in (c), the filter kernel used for this example.

РИСУНОК 24-12
Развитие ядра фильтра корреляции. Целевой сигнал показывается в (a). В (b) это вращается 180º Отменить вращение(поворот?), свойственное свертке, позволяя корреляции быть выполненным. Применение фильтра обнаружения края приводит к (c), ядро фильтра, используемое для этого примера.

The FFT algorithm is used to transform (a) and (b) into the frequency domain. This results in *four* 128x128 arrays, the real and imaginary parts of the two images being convolved. Multiplying the real and imaginary parts of (a) with the real and imaginary parts of (b), generates the real and imaginary parts of (c). (If you need to be reminded how this is done, see Eq. 9-1). Taking the Inverse FFT completes the algorithm by producing the final convolved image.

Алгоритм БПФ используется, чтобы преобразовать (a) и (b) в частотный домен. Это приводит к четырем массивам 128x128, реальные и мнимые части из двух свертываемых изображений. Умножение реальных и мнимых частей (a) с реальными и мнимыми частями (b), генерирует реальные и мнимые части (c). (Если Вы должны напоминать, как это сделано, см. уравнение 9-1). Взятие Обратного БПФ заканчивает алгоритм, производя конечное свернутое изображение.

The value of each pixel in a correlation image is a measure of how well the target image matches the searched image *at that point*. In this particular example, the correlation image in (c) is composed of noise plus a single bright peak, indicating a good match to the target signal. Simply locating the brightest pixel in this image would specify the detected coordinates of the face. If we had not used the edge detection modification on the target signal, the peak would still be present, but much less distinct.

Значение каждого пиксела в изображении корреляции - мера того, насколько хорошо целевое изображение соответствует обысканному изображению в той точке. В этом специфическом примере, изображение корреляции в (c) составлено из шумового плюса единственный яркий пик, указывая хорошее соответствие к целевому сигналу. Просто расположение самого яркого пиксела в этом изображении определил бы обнаруженные координаты лица. Если бы мы не использовали модификацию обнаружения края на целевом сигнале, пик все еще присутствовал бы, но намного менее отличный.

While correlation is a powerful tool in image processing, it suffers from a significant limitation: the target image must be exactly the same *size* and *rotational orientation* as the corresponding area in the searched image. Noise and other variations in the amplitude of each pixel are relatively unimportant, but an exact spatial match is critical. For example, this makes the method almost useless for finding enemy tanks in military reconnaissance photos, tumors in medical images, and handguns in airport baggage scans. One approach is to correlate the image *multiple times* with a variety of shapes and rotations of the target image. This works in principle, but the execution time will make you loose interest in a hurry.

В то время как корреляция - мощный инструмент в обработке изображения, это страдает от существенного ограничения: целевое изображение должно быть точно того же *размера* и *ориентации вращения* как соответствующая область в обысканном изображении. Шум и другие вариации в амплитуде каждого пиксела относительно незначительны, но точное пространственное соответствие критическое. Например, это делает метод, почти бесполезным для обнаружения вражеских танков в военных фотографиях разведки, опухоли в медицинских изображениях, и пистолетов в просмотрах авиа-багажа. Один подход состоит в том, чтобы коррелировать изображения *множителем времени* с рядом форм и вращений целевого изображения. Это работает в принципе, но время выполнения будет делать Вас свободным интересом(процентом), гонки(поспешности; нетерпеливости).

A Closer Look at Image Convolution
Взгляд на Свертку Изображения Ближе

Let's use this last example to explore two-dimensional convolution in more detail. Just as with one dimensional signals, image convolution can be viewed from either the *input side* or the *output side*. As you recall from Chapter 6, the input viewpoint is the best description of how convolution works, while the output viewpoint is how most of the mathematics and algorithms are written. You need to become comfortable with both these ways of looking at the operation.

Давайте использовать этот последний пример, чтобы исследовать двумерную свертку более подробно. Так же, как с одноразмерными сигналами, свертка изображения может быть просмотрена или от *входной стороны* или *стороны выхода*. Как Вы помните из главы 6, входная точка зрения - лучшее описание того, как работы свертки, в то время как точка

зрения выхода состоит в том, как большинство математики и алгоритмов написано. Вам должны стать удобными оба эти взгляда на операцию.

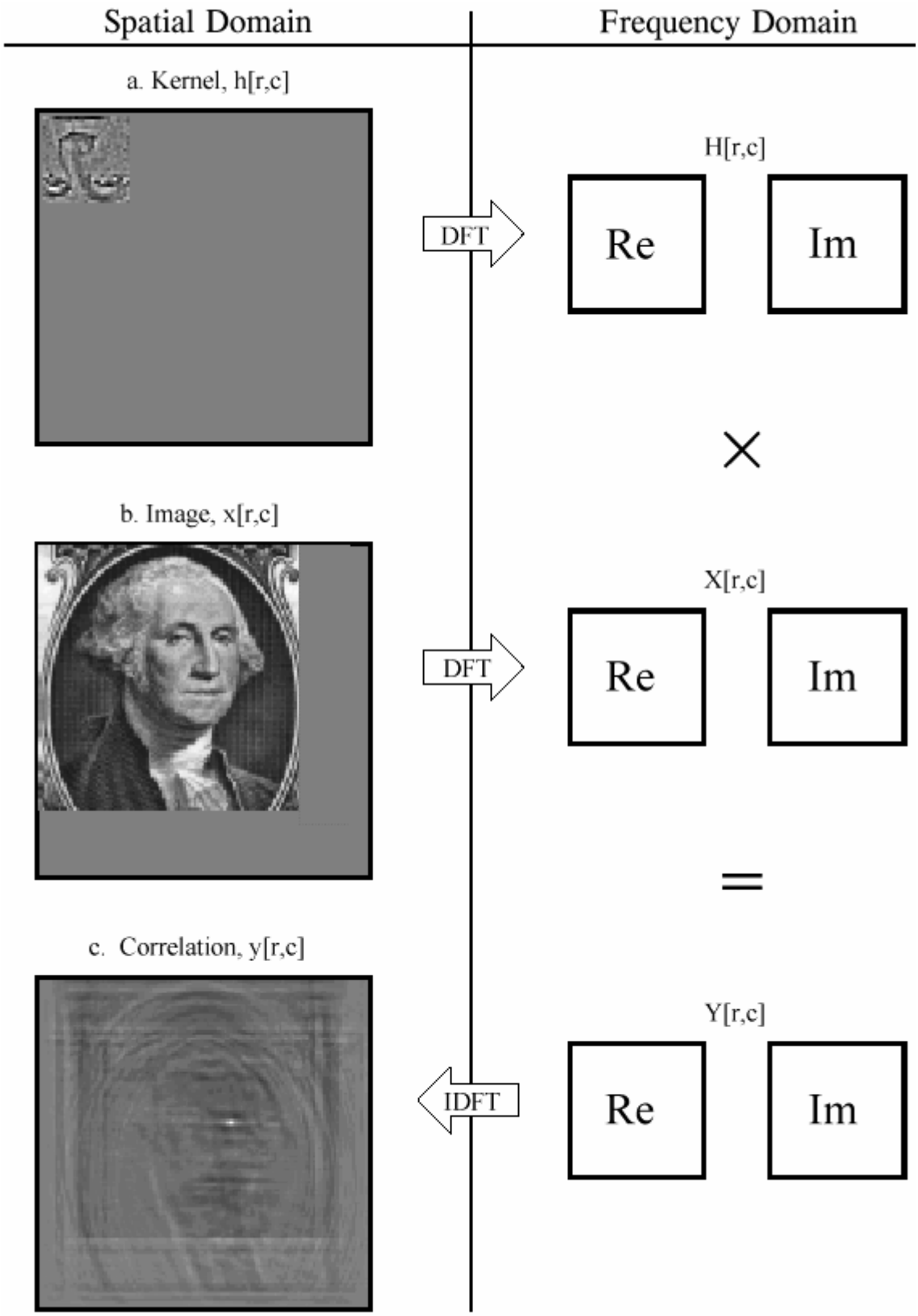


FIGURE 24-13. Flow diagram of FFT image convolution. The images in (a) and (b) are transformed into the frequency domain by using the FFT. These two frequency spectra are multiplied, and the Inverse FFT is used to move back into the spatial domain. In this example, the original images have been chosen and preprocessed to implement *correlation* through the action of convolution.
РИСУНОК 24-13

Блок-схема БПФ отображает свертку. Изображения в (a) и (b) преобразованы в частотный домен, используя БПФ. Эти два частотных спектра умножены, и Обратное БПФ используется, чтобы двигаться обратно в пространственный домен. В этом примере, первоначальные изображения были выбраны и предварительно обработаны, чтобы осуществить корреляцию через действие свертки.

Figure 24-14 shows the input side description of image convolution. Every pixel in the input image results in a scaled and shifted PSF being added to the output image. The output image is then calculated as the sum of all the contributing PSFs. This illustration show the contribution to the output image from the point at location [r,c] in the input image. The PSF is shifted such that pixel [0,0] in the PSF aligns with pixel [r,c] in the output image. If the PSF is defined with only positive indexes, such as in this example, the shifted PSF will be entirely to the lower-right of [r,c]. Don't be confused by the face appearing upside down in this figure; this upside down face *is* the PSF we are using in this example (Fig. 24-13a). In the input side view, there is no rotation of the PSF, it is simply shifted.

Рисунок 24-14 показывает входное побочное описание свертки изображения. Каждый пиксел во входном изображении приводит к масштабируемому и сдвинутому ФРТ, добавляемому к изображению выхода. Изображение выхода тогда рассчитано как сумма вкладов всех ФРТ. Этот показ иллюстрации содействия(вклада) изображению выхода от точки в расположении [r, c] во входном изображении. ФРТ сдвинут так, что пиксел [0,0] в ФРТ выравнивается с пикселом [r, c] в изображении выхода. Если ФРТ определен только положительными индексами, как в этом примере, сдвинутый ФРТ будет полностью к нижнему правому из [r, c]. Не будьте перепутаны лицом, появляющимся перевернутым вниз в этом рисунке; это перевернутое вниз лицо - *ФРТ*, который мы используем в этом примере из (рис. 24-13a). Во входном виде сбоку, не имеется никакого вращения ФРТ, это просто сдвинуто.

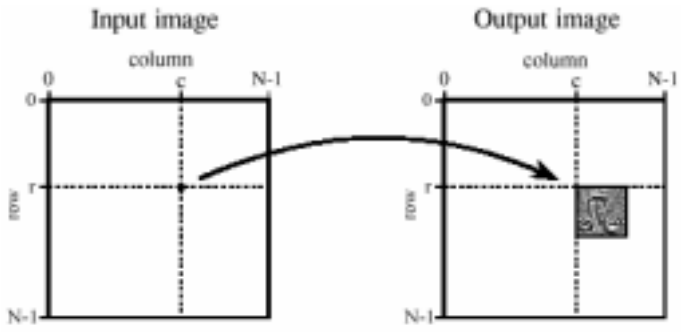


FIGURE 24-14
Image convolution viewed from the input side. Each pixel in the input image contributes a scaled and shifted PSF to the output image. The output image is the sum of these contributions. The face is inverted in this illustration because this is the PSF we are using.

РИСУНОК 24-14
Свертка изображения, просмотренная от входной стороны. Каждый пиксел во входном изображении вносит вклад в масштабируемое и сдвинутое изображение выхода ФРТ. Изображение выхода - сумма этих содействий(вкладов). Лицо инвертировано в этой иллюстрации, потому что это - ФРТ, который мы используем.

Image convolution viewed from the output is illustrated in Fig. 24-15. Each pixel in the output image, such as shown by the sample at [r,c], receives a contribution from many pixels in the input image. The PSF is rotated by 180º around pixel [0,0], and then shifted such that pixel [0,0] in the PSF is aligned with pixel [r,c] in the input image. If the PSF only uses positive indexes, it will be to the upper-left of pixel [r,c] in the input image. The value of the pixel at [r,c] in the output image is found by multiplying the pixels in the rotated PSF with the corresponding pixels in the input image, and summing the products. This procedure is given by Eq. 24-3, and in the program of Table 24-1.

Свертка изображения, просмотренная от выхода иллюстрирована в рис. 24-15. Каждый пиксел в изображении выхода, типа показанный выборкой в [r, c], получает содействие(вклад) от многих пикселов во входном изображении. ФРТ вращается 180º вокруг пиксела [0,0], и затем сдвинут так, что пиксел [0,0] в ФРТ выровнен с пикселом [r, c] во входном изображении. Если ФРТ использует только положительные индексы, это будет к левому верхнему из пиксела [r, c] во входном изображении. Значение пиксела в [r, c] в изображении выхода найдено, умножая пикселы во вращаемом ФРТ с соответствующими пикселами во входном изображении, и суммирования продуктов. Эта процедура дается уравнением 24-3, и в программе таблицы 24-1.

EQUATION 24-3

Image convolution. The images $x[,]$ and $h[,]$are convolved to produce image, $y[,]$. The size of $h[,]$ is MxM pixels, with the indexes running from 0 to M - 1. In this equation, an individual pixel in the output image, $y[r,c]$, is calculated according to the output side view. The indexes j and k are used to loop through the rows and columns of $h[\ ,\ ]$ to calculate the sum-of-products.

$$y[r,c] = \sum_{k=0}^{M-1} \sum_{j=0}^{M-1} h[k,j]\, x[r-k, c-j]$$

УРАВНЕНИЕ 24-3

Свертка изображения. Изображения $x[,]$ и $h[,]$ свернуты, чтобы произвести изображение, $y[,]$. Размер $h[,]$ – M x M пикселов, с индексами, выполняющиеся от 0 до M - 1. В этом уравнение, индивидуальный пиксел в изображении выхода, $y[r,c]$, рассчитан согласно виду сбоку выхода. Индексы j и k используются к циклу через строки и столбцы из $h[,]$ к вычислять суммы продуктов.

Notice that the PSF rotation resulting from the convolution has undone the rotation made in the design of the PSF. This makes the face appear upright in Fig. 24-15, allowing it to be in the same orientation as the pattern being detected in the input image. That is, we have successfully used *convolution* to implement *correlation*. Compare Fig. 24-13c with Fig. 24-15 to see how the bright spot in the correlation image signifies that the target has been detected.

Обратите внимание, что вращение ФРТ, следующее из свертки отменило вращение, сделанное в проекте ФРТ. Это заставит лицо казаться вертикальным в рис. 24-15, позволяя этому быть в той же самой ориентации как образец, обнаруживаемый во входном изображении. То есть мы успешно использовали *свертку*, чтобы осуществить *корреляцию*. Сравните рис. 24-13c с рис. 24-15, чтобы видеть, как яркое пятно(ячейка) в изображении корреляции показывает это, адресат был обнаружен.

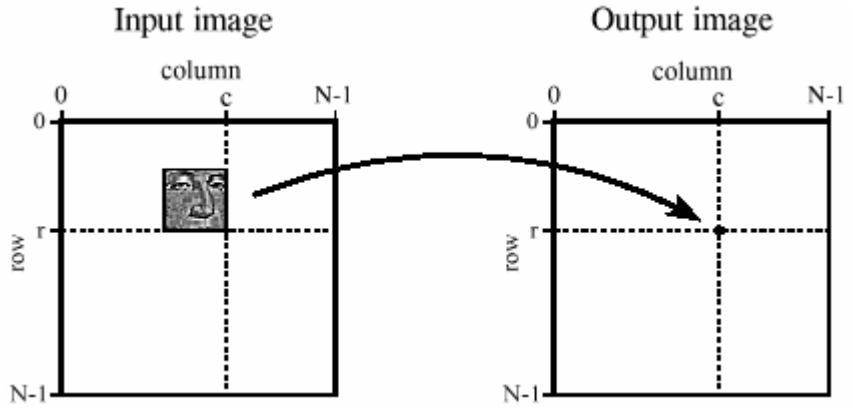


FIGURE 24-15

Image convolution viewed from the output side. Each pixel in the output signal is equal to the sum of the pixels in the rotated PSF multiplied by the corresponding pixels in the input image.

FFT convolution provides the same output image as the conventional convolution program of Table 24-1. Is the reduced execution time provided by FFT convolution really worth the additional program complexity? Let's take a closer look. Figure 24-16 shows an execution time com-

parison between conventional convolution using *floating point* (labeled FP), conventional convolution using *integers* (labeled INT), and FFT convolution using floating point (labeled FFT). Data for two different image sizes are presented, 512x512 and 128x128.

Свертка БПФ обеспечивает то же самое изображение выхода как обычная программа свертки Таблицы 24-1. Приведенное время выполнения обеспечивается сверткой БПФ, действительно стоящей дополнительной сложности программы? Давайте брать более близкий просмотр. Рисунок 24-16 показывает сравнение времени выполнения между обычной сверткой, использующей *плавающую запятую* (маркированная FP - ПЗ), обычная свертка, использующая *целые числа* (маркированная INT - ?), и свертку БПФ, используя плавающую запятая (меченое FFT - БПФ). Данные для двух различных размеров изображения представлены, 512x512 и 128x128.

```
100 CONVENTIONAL IMAGE CONVOLUTION
110 '
120 DIM X[99,99] 'holds the input image, 100Ч100 pixels
130 DIM H[28,28] 'holds the filter kernel, 29Ч29 pixels
140 DIM Y[127,127] 'holds the output image, 128Ч128 pixels
150 '
160 FOR R% = 0 TO 127 'loop through each row and column in the output
170 FOR C% = 0 TO 127 'image calculating the pixel value via Eq. 24-3
180 '
190 Y[R%,C%] = 0 'zero the pixel so it can be used as an accumulator
200 '
210 FOR J% = 0 TO 28 'multiply each pixel in the kernel by the corresponding
220 FOR K% = 0 TO 28 'pixel in the input image, and add to the accumulator
230 Y[R%,C%] = Y[R%,C%] + H[J%,K%] * X[R%-J%,C%-J%]
240 NEXT K%
250 NEXT J%
260 '
270 NEXT C%
280 NEXT R%
290 '
300 END
```

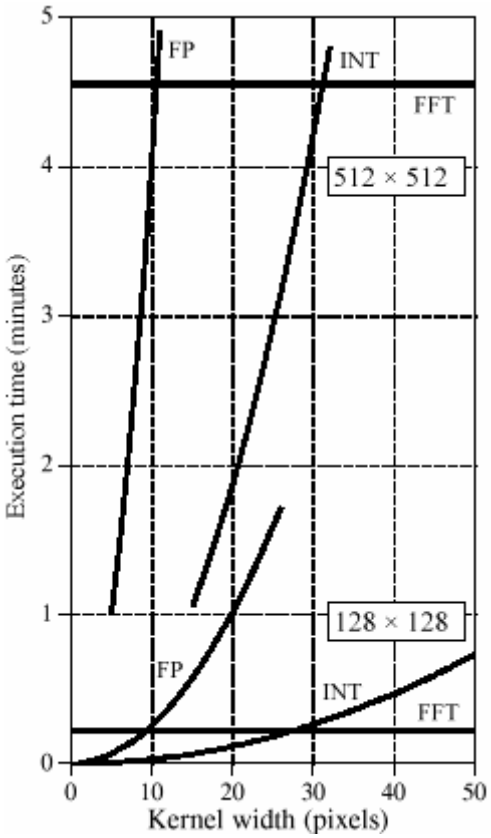
TABLE 24-1

First, notice that the execution time required for FFT convolution does not depend on the size of the kernel, resulting in flat lines in this graph. On a 100 MHz Pentium personal computer, a 128 x 128 image can be convolved in about 15 seconds using FFT convolution, while a 512Ч512 image requires more than 4 minutes. Adding up the number of calculations shows that the execution time for FFT convolution is proportional to N^2 Log_2 (N), for an NxN image. That is, a 512x512 image requires about 20 times as long as a 128x128 image.

Во первых, обратите внимание, что время выполнения, требуемое для свертки БПФ не зависит от размера ядра, приводя к единообразным строкам в этой диаграмме(графике). На персональный компьютер Pentium 100 MHz, изображение 128 x 128 может быть свернуто в приблизительно за 15 секунд, используя свертку БПФ, в то время как изображение 512x512 требует более 4 минут. Сложение числа вычислений показывает, что время выполнения для свертки БПФ пропорционально к N^2 Log_2 (N), для изображени NxN. То есть изображение 512x512 требует приблизительно 20 раз больше, чем изображение 128x128.

FIGURE 24-16
Execution time for image convolution. This graph shows the execution time on a 100 MHz Pentium processor for three image convolution methods: conventional convolution carried out with floating point math (FP), conventional convolution using integers (INT), and FFT convolution using floating point (FFT). The two sets of curves are for input image sizes of 512Ч512 and 128Ч128 pixels. Using FFT convolution, the time depends only on the image size, and not the size of the kernel. In contrast, conventional convolution depends on both the image and the kernel size.

РИСУНОК 24-16
Время выполнения для свертки изображения. Эта диаграмма(график) показывает время выполнения на процессоре Pentium 100 MHz для трех методов свертки изображения: обычная свертка, выполненная с математической с плавающей запятой (FP – ПЗ?), обычная свертка, использующая целые числа (INT), и свертку БПФ, использующую плавающую запятую (FFT – БПФ?). Два набора кривых - для входных размеров изображения 512x512 и 128x128 пикселов. Используя свертку БПФ, время зависит только от размера изображения, а не размера ядра. Напротив, обычная свертка зависит, и от изображения и от размера ядра.



Conventional convolution has an execution time proportional to N^2M^2 for an NxN image convolved with an MxM kernel. This can be understood by examining the program in Table 24-1. In other words, the execution time for conventional convolution depends *very strongly* on the size of the kernel used. As shown in the graph, FFT convolution is faster than conventional convolution using floating point if the kernel is larger than about 10x10 pixels. In most cases, integers can be used for conventional convolution, increasing the break-even point to about 30x30 pixels. These break-even points depend slightly on the size of the image being convolved, as shown in the graph. The concept to remember is that FFT convolution is only useful for *large* filter kernels.

Обычная свертка имеет член пропорции времени выполнения к N^2M^2 для изображения NxN, свернутое с ядром MxM. Это может быть понято, исследуя программу в таблице 24-1. Другими словами, время выполнения для обычной свертки зависит очень строго от размера используемого ядра. Как показано в диаграмме(графике), свертка БПФ быстрее чем обычная свертка, используя плавающую запятая, если ядро больше чем около 10x10 пикселов. В большинстве случаев, целые числа могут использоваться для обычной свертки, увеличивая точку безубыточности к относительно 30x30 пикселов. Эти точки безубыточности зависят слегка от размера свертываемого изображения, как показано в диаграмме(графике). Концепция, чтобы помнить - то, что свертка БПФ является полезной только для больших ядер фильтра.