MATLAB 8.0 (R2012b): обработка изображений в пакете Image Processing Toolbox

Владимир ДЬЯКОНОВ,
д. т. н., профессор
vpdyak@yandex.ru

Новая матричная система компьютерной математики MATLAB 8.0 (R2012b) имеет удобные средства для быстрой и эффективной обработки изображений. Они сосредоточены в пакете расширения Image Processing Toolbox. В этой статье впервые в отечественной литературе описаны возможности работы с изображениями как в среде самой системы MATLAB 8.0, так и с пакетом расширения Image Processing Toolbox R2012b. Автор благодарит корпорацию The MathWorks, Inc. [1] за предоставленную систему MATLAB 8.0 + Simulink 8.0, использованную для подготовки этой серии статей.

Состав и назначение пакета расширения Image Processing Toolbox

Окно справки по пакету расширения Image Processing Toolbox системы MATLAB 8.0 представлено на рис. 1. Полиграфическая база журнала «Компоненты и технологии» позволила представить многие примеры применения в цвете — как на экране персонального компьютера. Число примеров увеличено за счет представления нескольких из них в одном окне.

Судя по окну справки, пакет Imagel Processing Toolbox состоит из следующих разделов:

- Import, Export and Conversions — импорт, экспорт и преобразования изображения;
- Display and Exploration — отображение и исследование изображения;
- Geometric Transformation and Image Registration — геометрическое преобразование и регистрация изображения;
- Image Enhancement — улучшение изображения;
- Image Analysis — анализ изображения;
- Color — работа с цветом;
- Code Generation — генерация кодов.

Пакет Image Processing Toolbox не предназначен для оперативной обработки изображений. С этим прекрасно справляются специализированные редакторы фотографических изображений, например популярные Photoshop, Photo-Paint или Photoimpact. Основное назначение пакета — исследование, разработка и реализация математических алгоритмов и программ для обработки изображений, применяющихся в системах и устройствах сигнализации и опознания целей, компонентах и технологиях обработки изображений и в новых видеоустройствах. Image Processing Toolbox добавляет в матричную систему MATLAB около двухсот новых функций обработки изображений. Они естественно интегрируются с функциями матричной системы MATLAB и других пакетов расширения.

Считывание и запись файлов изображений

Для загрузки изображений из графических файлов служит функция:

```
RGB = imread('football.jpg');
[X,map] = imread('trees.tif');
```

Имя файла указывается в апострофах. Поддерживаются многие графические форматы файлов, например Microsoft Windows Bitmap (BMP), Graphics Interchange Format (GIF), Joint Photographic ExpertsGroup (JPEG), Portable Network Graphics (PNG) и Tagged Image File Format (TIFF). Обеспечена поддержка и файлов научных форматов DICOM, NITF, HDR, формата 7.5 и др.

Рис. 1. Окно справки по пакету расширения Image Processing Toolbox

Рис. 2. Просмотр файла формата DICOM

Приведем пример просмотра файла формата DICOM (рис. 2):

```
>> I = dicomread('CT-MONO2-16-ankle.dcm');
>> imshow(I,'DisplayRange',[])
```

Пример просмотра файла формата HDR (рис. 3) реализуется командами:

```
>> hdr_image = hdrread('office.hdr');%Считывание файла
>> rgb = tonemap(hdr_image); imshow(rgb);%Показ изображения
```

После знака % задаются текстовые неисполняемые комментарии. В программах пакета они применяются широко, но в этой статье они не приводятся ради экономии места и устранения повторов в описании программ. Предполагается, что читатель статьи знаком с основами программирования системы MATLAB [2–5].

С помощью функции **imfinfo** можно вывести информацию о файле, например:

```
>> imfinfo('moon.tif')
ans =
   Filename: 'C:\Program Files\MATLAB\R2012b\toolbox\images\
imdemos\moon.tif'
   FileModDate: '04-дек-2000 13:57:58'
      FileSize: 183950
         Format: 'tif'
   FormatVersion: []
      Width: 358
         Height: 537
            BitDepth: 8
               ColorType: 'grayscale'...
```

В этих примерах используется командный режим работы, на что указывает двойной символ ввода >>. Таким образом можно вводить как отдельные команды и строки, так и целые небольшие программы (в том числе копируемые из текста). Но после запуска можно повторно вводить из программного стека и редактировать только отдельные законченные команды. Этого неудобства лишен программный режим работы, при котором вся программа набирается, редактируется, отлаживается (если это нужно) и запускается во встроенном редакторе программных кодов MATLAB. Символ ввода >> в программах и их фрагментах не используется.

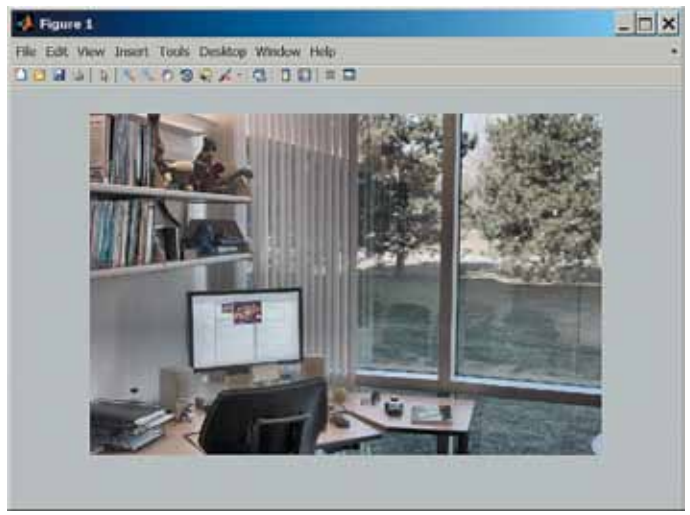

Рис. 3. Просмотр файла формата HDR

Структура файлов изображений

Из основных (широко распространенных) типов файлов можно отметить следующие:

- Binary Images — бинарные (черно-белые) изображения без полутонов;
- Indexed Images — индексированные изображения;
- Grayscale Images — монохромные изображения с полутонами;
- True Color Images — цветные изображения.

В бинарных файлах изображение хранится в одном массиве, все элементы которого — целые числа 0 (черный цвет) или 1 (белый цвет). Аналогичную структуру имеют полутоновые монохромные файлы grayscale, но элементы их массивов — числа с плавающей точкой различного, допустимого в MATLAB типа.

Структура цветного индексированного изображения представлена на рис. 4а. Один массив несет информацию о яркости точек пиксельными координатами, другой — коды цвета (десятичные).

Наиболее полноценным (True Color) цветным изображением характеризуются файлы в формате RGB с 24-битовым кодированием цвета. Они представлены тремя массивами красного (Red), зеленого (Green) и синего (Blue) цветов (рис. 4б). Возможно 8-, 16- и 32-битовое кодирование цветов.

Следующая программа поясняет разбивку RGB-изображения на его R, G и B составляющие (рис. 5):

```
RGB=reshape(ones(64,1)*reshape(jet(64),1,192),[64,64,3]);
R=RGB(:,:,1); G=RGB(:,:,2); B=RGB(:,:,3);
subplot(221);imshow(R); title('R')
subplot(222);imshow(G); title('G')
subplot(223);imshow(B); title('B')
subplot(224);imshow(RGB); title('RGB')
```

Здесь и далее команды ***subplot(m,n,№)***, где цифры — это номера строк, столбцов и изображений, разбивают программу и окно вывода на номера частей и позволяют представить их в едином виде.

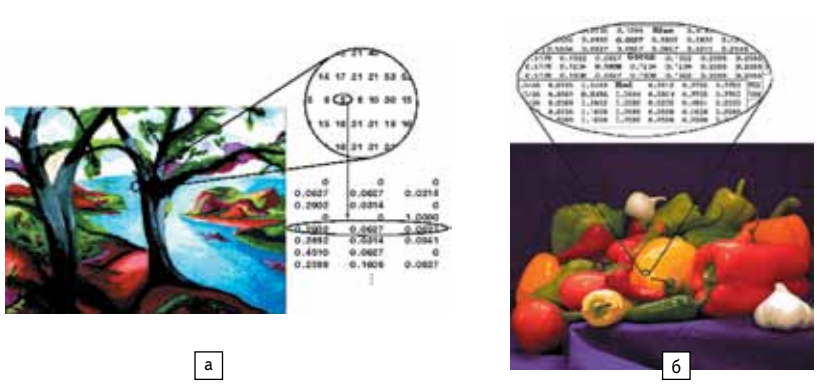

Рис. 4. Структура цветного: а) индексированного изображения; б) RGB-изображения

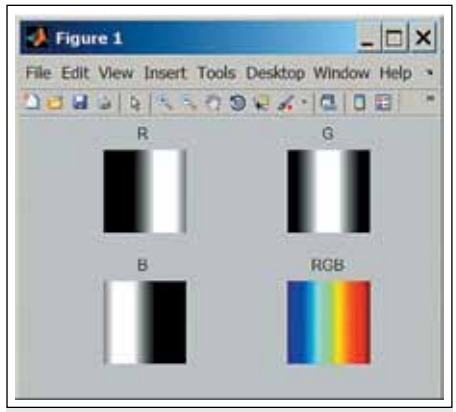

Рис. 5. Выделение составляющих R, G и B цветного RGB-изображения

Просмотр изображений

Просмотр изображений возможен с помощью ряда функций, например **imshow** и **subimage**:

```
RGB = imread('peppers.png');subplot(221);imshow(RGB)
[X1,map1]=imread('forest.tif');
[X2,map2]=imread('trees.tif');
subplot(222), imshow(X2,map2);subplot(223),
subimage(X1,map1);subplot(224), subimage(X2,map2)
```

Для просмотра изображений в специальном GUI-окне **Image Tool** служит функция **imtool**. Ее можно использовать для просмотра внешних графических файлов, не входящих в систему MATLAB. Для вывода этого окна достаточно активировать пиктограмму в меню каталога пакетов расширения APPS или вызвать функцию в командном режиме. На рис. 6 показан просмотр цветного изображения пингвинов из внешнего файла фотографического формата *.jpg*.

Рис. 6. Пример просмотра в imtool изображения из внешнего файла

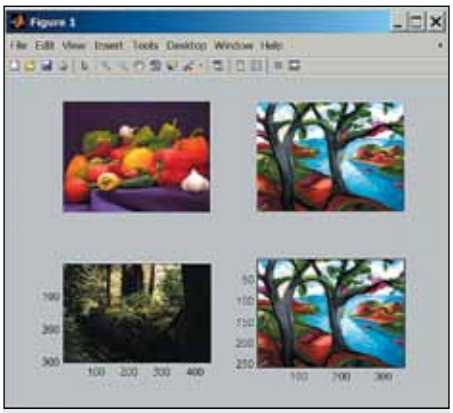

Рис. 7. Применение функций просмотра изображений

Окно **Image Tool** имеет позицию меню **Tool** с командами изменения размера просматриваемого изображения, выделения его участка, измерения расстояния между точками изображения и вывода окна с информацией об изображении. Контроль расстояния и окно с информацией об изображении показаны на рис. 7. Эти команды есть также в панели инструментов окна наряду с командами копирования изображения в буфер и записи в файл.

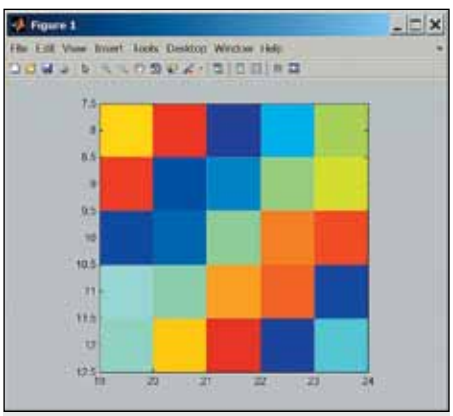

Рис. 8. Визуализация матрицы с цветной окраской ее элементов

Графику можно использовать и для визуализации матриц. Приведем пример визуализации матрицы (рис. 8):

```
A = magic(5); x = [19.5 23.5]; y = [8.0 12.0];
image(A,'XData',x,'YData',y), axis image, colormap(jet(25))
```

Некоторые изображения состоят из множества кадров (фреймов), каждый из которых представлен своим массивом или массивами (рис. 9):

```
load mri; montage(D,map)
```

Здесь функция **load** загружает изображение mri, а функция **montage** монтирует и показывает все фреймы в одном окне.

Другой фрагмент программы позволяет показать только первые девять фреймов:

```
figure; montage(D, map, 'Indices', 1:9);
```

Для просмотра с анимацией изображений с многими фреймами служит проигрыватель **implay** с кнопками управления и своей панелью инструментов (рис. 10). Проигрыватель можно вызвать из каталога APPS или командами:

```
mov = immovie(D,map); implay(mov)
```

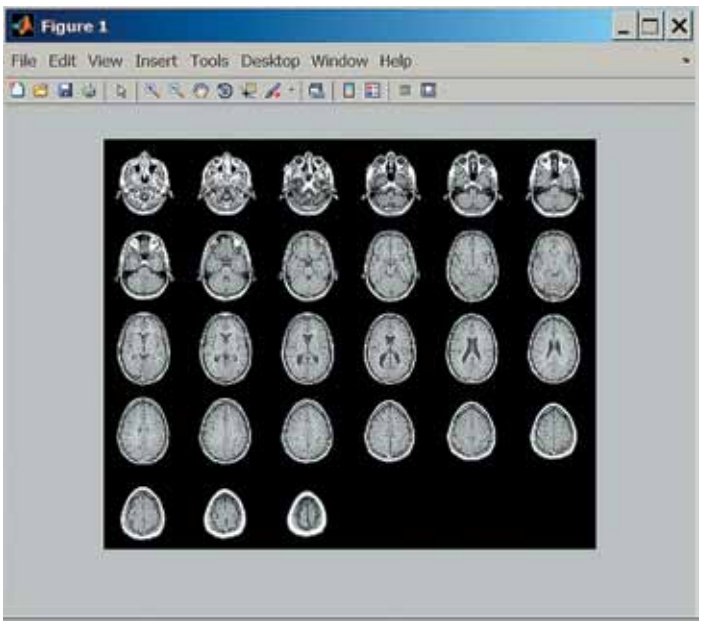

Рис. 9. Фреймы срезов черепной коробки

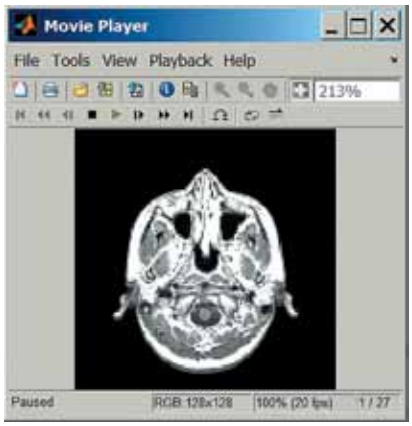

Рис. 10. Проигрыватель файлов с анимацией изображений

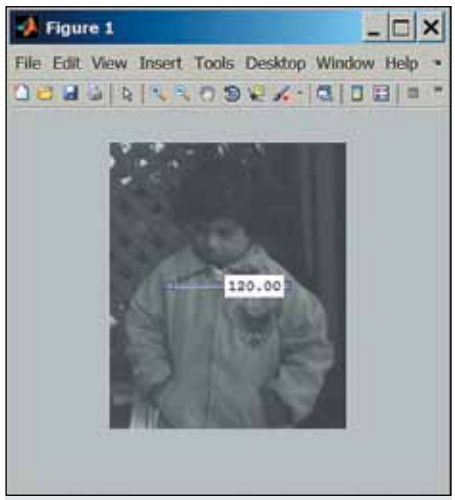

Рис. 11. Изображение файла pout.tif с контролем расстояния между двумя точками

В Image Processing Toolbox входит множество функций, объединяющих просмотр изображений с вычислением и визуализацией параметров изображения. Ограничимся примером показа файла *pout.tif* с вычислением расстояния между двумя точками изображения, указываемыми мышкой (рис. 11):

```
figure, imshow('pout.tif');
h = imdistline(gca); api = iptgetapi(h);
fcn = makeConstrainToRectFcn('imline',... get(gca,'XLim'),
get(gca,'YLim'));
```

Далее в статье будет приведено множество других примеров применения функций просмотра изображений.

Геометрические преобразования

Учет законов и формул геометрии лежит в основе многих геометрических преобразований изображений [4, 5]. К ним сводятся действия с изображениями, как с матрицами. Например, изменение размеров изображений (рис. 12а):

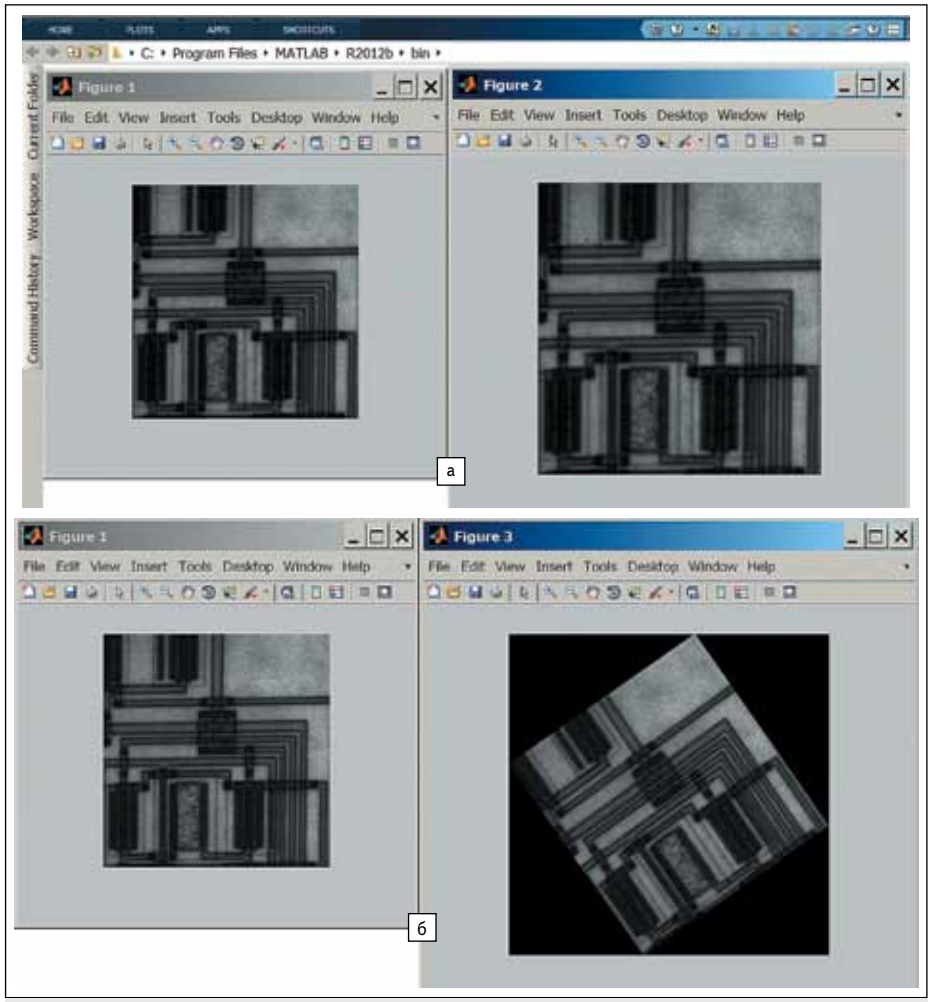

Рис. 12. Преобразование изображения: а) изменение размеров; б) поворот

```
I = imread('circuit.tif'); J = imresize(I,1.25);
imshow(I); figure, imshow(J)
```

Функция **imrotate** обеспечивает умножение матрицы на матрицу вращения, то есть обеспечивает поворот изображения на заданный угол (рис. 12б):

```
I = imread('circuit.tif'); J = imrotate(I,35,'bilinear');
imshow(I); figure, imshow(J)
```

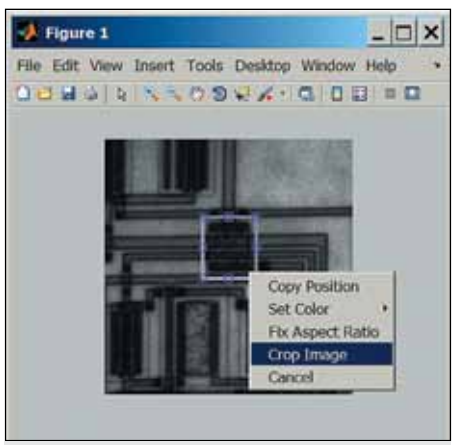

Рис. 13. Выделение фрагмента изображения мышкой

Для интерактивного ввода фрагмента изображения служит функция **imcrop**:

```
I = imread('circuit.tif'); J = imcrop(I);
```

Она выводит изображение **I**, на котором мышкой сначала можно выделить прямоугольник с нужным фрагментом (рис. 13), а затем присвоить ему имя **J**. Функция **imshow(J)** позволяет просмотреть выделенный фрагмент.

Функции показа изображения можно использовать для просмотра фотографий, сделанных при аэрофотосъемке и космической фотосъемке, а также изображений от устройств наведения на цель ракет, управляемых снарядов и авиационных бомб.

Рис. 14. Снимок, полученный при аэрофотосъемке

Приведенная ниже программа демонстрирует показ снимка участка города при аэрофотосъемке (рис. 14):

```
unregistered = imread('westconcordaerial.png');
subplot(121); imshow(unregistered)
subplot(122);imshow('westconcordorthophoto.png')
```

Работа с выделенными областями

Часто возникает необходимость работы с выделенными областями рисунка (регионами или областями интереса). Программа, представленная ниже, иллюстрирует такую работу с крайним правым объектом (монетой, рис. 15):

```
I = imread('eight.tif');
c = [222 272 300 270 221 194];
r = [21 21 75 121 121 75]; J = roifill(I,c,r);
subplot(221); imshow(I); subplot(222); imshow(BW)
subplot(223); imshow(J); BW = roipoly(I,c,r);
BW = roipoly(I,c,r); H = fspecial('unsharp');
J = roifilt2(H,I,BW); subplot(224); imshow(J)
```

Монета выделена подходящей фигурой (шестиугольником), которая закрашена затем цветом фона: это приводит к удалению изображения монеты. Можно, напротив, применить операцию усиления контрастности и таким образом выделить монету. Эти операции можно также выполнять с блоками изображения (рис. 16):

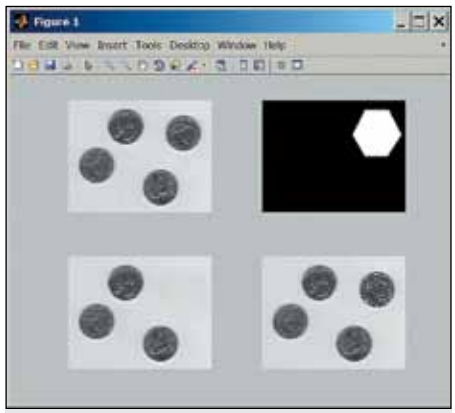

Рис. 15. Работа с выделенными областями

```
@(block_struct)imresize(block_struct.data,0.15);
I = imread('pears.png'); I2 = blockproc(I,[100 100],fun);
figure; imshow(I); figure; imshow(I2);
```

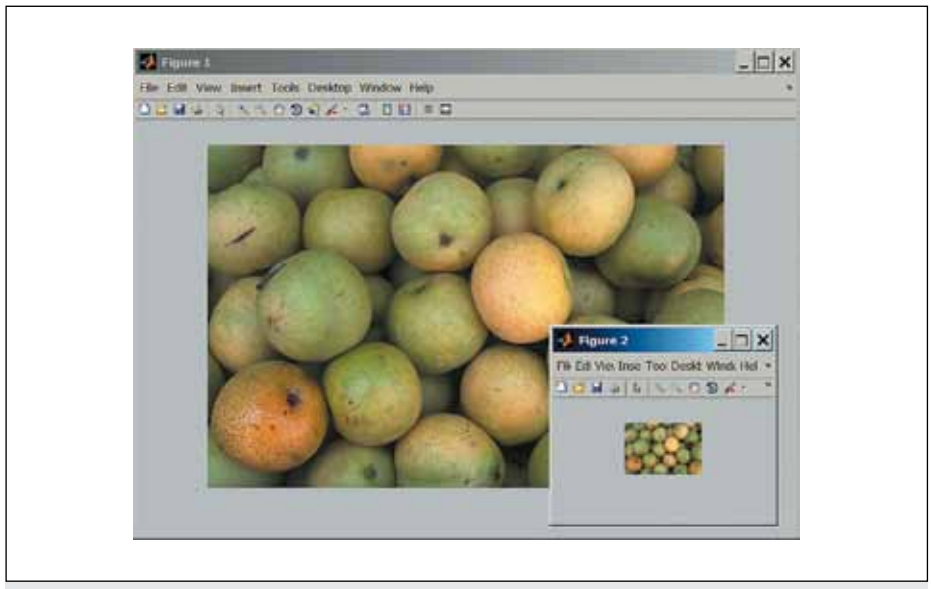

Рис. 16. Изображения яблок разного размера

Функция **roicolor(X,low,high)** выделяет участок из цветного изображения, а приведенная ниже программа строит черно-белое изображение этого участка (рис. 17):

```
load clown; BW = roicolor(X,10,20);
imshow(X,map); figure,imshow(BW)
```

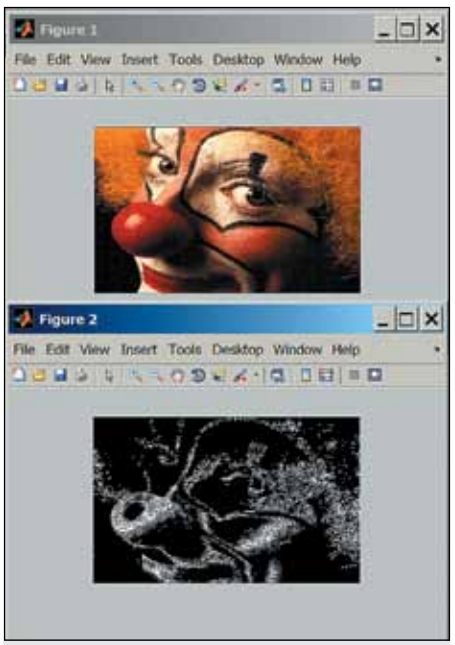

Рис. 17. Выделение участка цветного изображения и построение его в черно-белом виде

Улучшение изображения

Улучшение качества изображения — одна из важнейших задач при его редактировании. В приведенной ниже программе показано изменение контрастности монохромного и цветного изображений (рис. 18):

```
I = imread('pout.tif');J = imadjust(I);
subplot(221); imshow(I); subplot(222); imshow(J)
RGB1 = imread('football.jpg');
RGB2 = imadjust(RGB1,[.2 .3 0; .6 .7 1],[]);
subplot(223); imshow(RGB1),
subplot(224); imshow(RGB2)
```

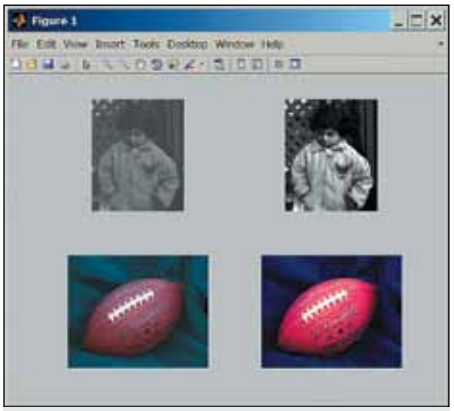

Рис. 18. Изменение контрастности монохромного и цветного изображений

Для получения негативного изображения черно-белого объекта BW достаточно заменить его объектом ~BW (рис. 19):

```
BW = imread('circles.png');
subplot(121); imshow(BW)
subplot(122); imshow(~BW)
```

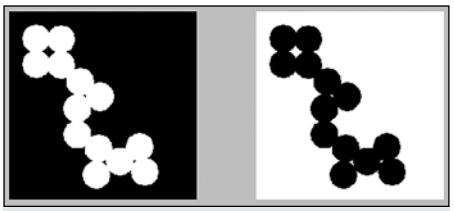

Рис. 19. Смена цвета — белого на черный и черного на белый

Расфокусировка изображений с технической точки зрения обычно ухудшает изображение. Но она же является важным художественным приемом, например при подготовке портретов. Следующая программа обеспечивает расфокусировку изображения с помощью фильтра **filteredRGB** (рис. 20):

```
originalRGB = imread('peppers.png');
subplot(121); imshow(originalRGB);
h = fspecial('motion', 50, 45);
filteredRGB = imfilter(originalRGB, h);
subplot(122); imshow(filteredRGB)
```

Для понижения и повышения четкости изображения служит ряд функций. Функции,

Рис. 20. Расфокусировка изображения — действие фильтра filteredRGB

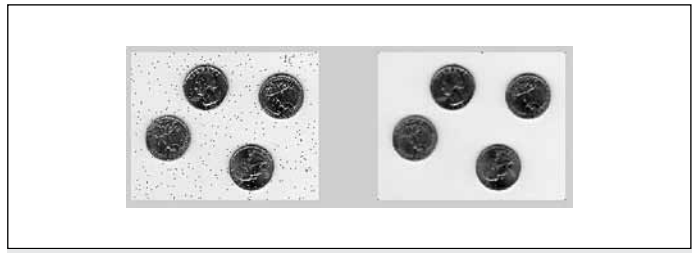

Рис. 23. Очистка изображения от шума (черные точки) медианным фильтром medfilter2

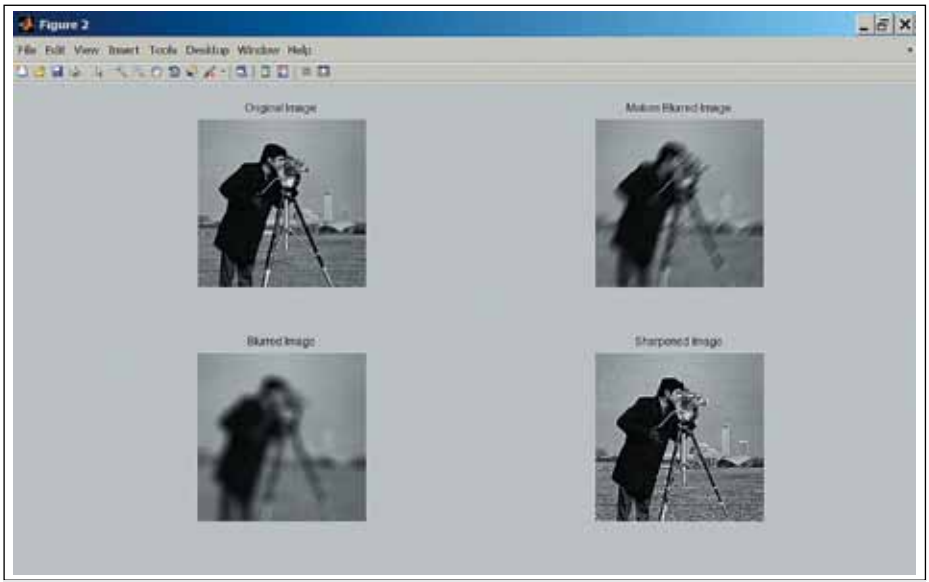

Рис. 21. Понижение и повышение четкости изображения

имеющие в названии слово **blur**, смягчают изображение, а функция **sharpened** делает изображение более резким. Последнее обеспечивает и специальный фильтр, подчеркивающий высокие частоты и не искажающий низкие и средние частоты. Следующая программа иллюстрирует эти действия на примере монохромного изображения фотографа с камерой (рис. 21):

```
I = imread('cameraman.tif');
subplot(2,2,1); imshow(I); title('Original Image');
H = fspecial('motion',20,45);
MotionBlur = imfilter(I,H,'replicate');
subplot(2,2,2);imshow(MotionBlur);title('Motion Blurred Image');
H = fspecial('disk',10); blurred = imfilter(I,H,'replicate');
subplot(2,2,3); imshow(blurred); title('Blurred Image');
H = fspecial('unsharp'); sharpened = imfilter(I,H,'replicate');
subplot(2,2,4); imshow(sharpened); title('Sharpened Image');
```

Еще один полезный прием — помещение ограниченного по размеру цветного изображения на черный прямоугольник большего размера (рис. 22):

```
load mandrill; figure('color','k')
image(X); colormap(map); axis off; axis image
```

Фильтрация изображений и фильтры

Фильтры играют важную роль в обработке изображений. Например, фильтры,

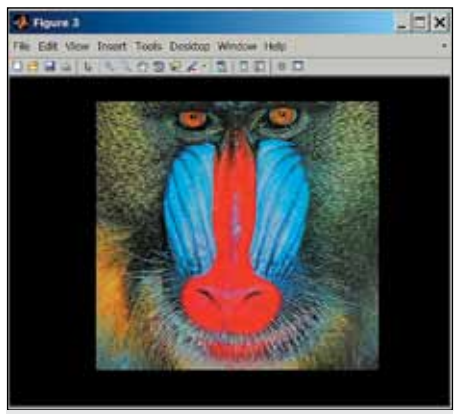

Рис. 22. Цветное изображение, помещенное на черный фон

использующие усреднение точек изображения, обычно применяются для сглаживания и очистки изображения от шума. В следующей программе для очистки изображения от черных точек применяется медианный фильтр **medfilter2** (рис. 23):

```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.02);
K = medfilt2(J);
subplot(121); imshow(J); subplot(122); imshow(K)
```

Фильтры обычно характеризуются своей амплитудно-частотной характеристикой (АЧХ). Следующая программа строит трехмерные АЧХ ряда фильтров, позволяющие судить об их назначении (рис. 24):

```
Hd = zeros(16,16); Hd(5:12,5:12) = 1;
Hd(7:10,7:10) = 0; h = fwind1(Hd,bartlett(16));
colormap(jet(64)); subplot(221);
freqz2(h,[32 32]); axis ([-1 1 -1 1 0 1])
[f1,f2] = freqspace(21,'meshgrid'); Hd = ones(21);
r = sqrt(f1.^2 + f2.^2); Hd((r<0.1)|(r>0.5)) = 0;
subplot(222); mesh(f1,f2,Hd)
b = remez(10,[0 0.05 0.15 0.55 0.65 1],[0 0 1 1 0 0]);
[H,w] = freqz(b,1,128,'whole');
subplot(223);plot(w/pi-1,fftshift(abs(H)))
h = ftrans2(b);subplot(224); freqz2(h)
```

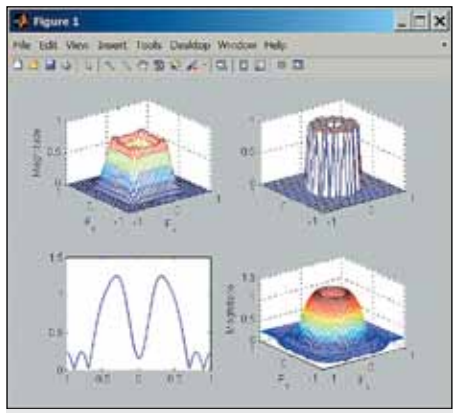

Рис. 24. АЧХ фильтров fwind1, remez и ftrans2

Программа (рис. 25) демонстрирует основные приемы обработки изображения — зашумление изображения и очистку от шума с выделением шумовых составляющих:

```
I=im2double(imread('cameraman.tif')); subplot(221);
imshow(I); title('Original Image (courtesy of MIT)');
LEN = 21; THETA = 11;
PSF = fspecial('motion', LEN, THETA);
blurred = imfilter(I, PSF, 'conv', 'circular');
noise_mean = 0; noise_var = 0.0001;
blurred_noisy=imnoise(blurred,'gaussian',noise_mean, noise_var);
subplot(222); imshow(blurred_noisy);
title('Simulate Blur and Noise')
estimated_nsr = noise_var / var(I(:));
wnr3 = deconvwnr(blurred_noisy, PSF, estimated_nsr);
subplot(223); imshow(wnr3)
title('Restoration of Blurred, Noisy Image Using Estimated NSR');
estimated_nsr = 0;
wnr2 = deconvwnr(blurred_noisy, PSF, estimated_nsr);
subplot(224), imshow(wnr2)
title('Restoration of Blurred, Noisy Image Using NSR = 0')
```

Морфологические преобразования

Морфологические преобразования выполняются над формой объектов изображений и составляют основу средств распознавания образов. Это нелинейные операции, базиру-

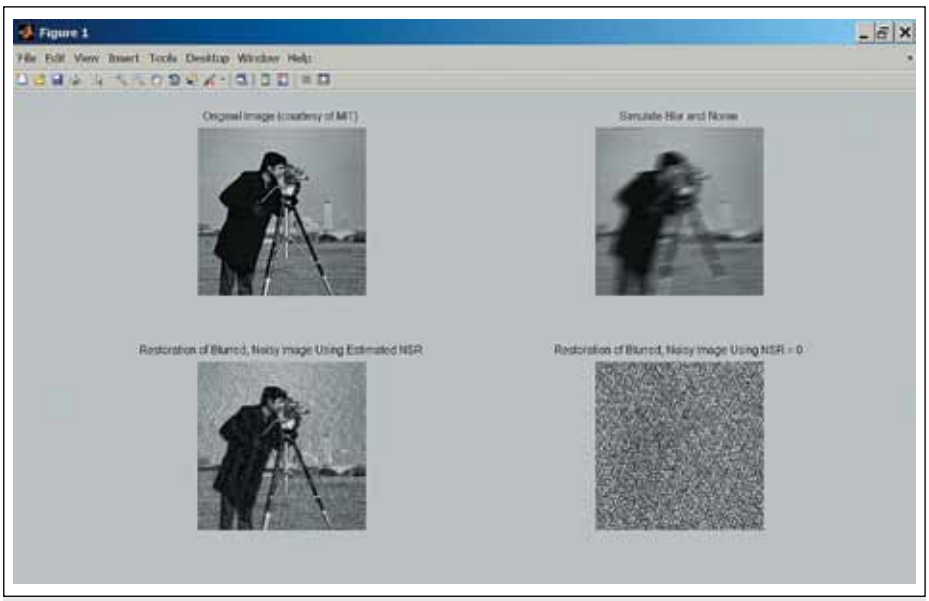

Рис. 25. Оригинальное изображение, изображение с Blur и шумом и выделение его составляющих

ющиеся на связанности пикселей. В MATLAB эти операции выполняются с бинарными (черно-белыми) изображениями, причем принято считать, что пиксели со значением 1 относятся к объекту, а со значением 0 — к фону. Из большого числа таких функций мы рассмотрим лишь несколько наиболее распространенных.

Например, функция бинарного закрытия изображения **imclose** приводит к удалению внутри изображения небольших фрагментов типа дыр. Функция эрозии **imerode** заменяет на 0 граничные пиксели объекта, то есть удаляет граничную область объекта толщиною в заданное число пикселей. Это иллюстрирует следующая программа (рис. 26):

```
originalBW = imread('circles.png');
subplot(221); imshow(originalBW); title('Original Image')
se = strel('disk',10); closeBW = imclose(originalBW,se);
subplot(222); imshow(closeBW); title('Close');
erodedBW = imerode(originalBW,se);
subplot(223); imshow(erodedBW); title('Eroded 10')
se = strel('disk',15); erodedBW = imerode(originalBW,se);
subplot(224); imshow(erodedBW); title('Eroded 15')
```

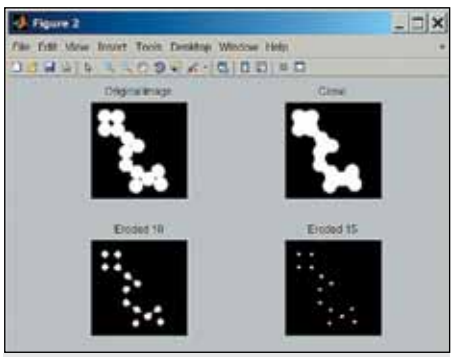

Рис. 26. Операции закрытия и эрозии изображения

Еще одна программа строит контуры черно-белого изображения с удалением закраски объектов и заменой их скелетоном (рис. 27):

```
BW = imread('circles.png'); subplot(131); imshow(BW);
BW2 = bwmorph(BW,'remove'); subplot(132); imshow(BW2)
BW3 = bwmorph(BW,'skel',Inf); subplot(133); imshow(BW3)
```

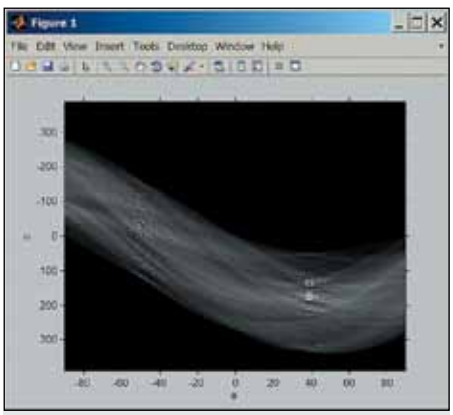

Рис. 28. Выделение двух точек изображения

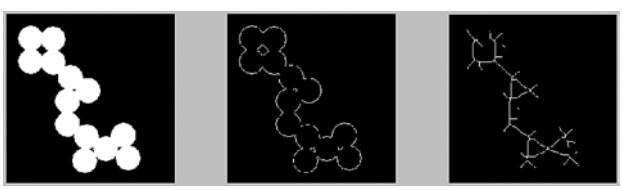

Рис. 27. Построение контуров объектов удалением их закраски и замена их скелетоном

А следующая программа обеспечивает поиск и обнаружение двух точек изображения (рис. 28):

```
I = imread('circuit.tif'); BW = edge(imrotate(I,50,'crop'),'canny');
[H,T,R] = hough(BW); P = houghpeaks(H,2);
imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit');
xlabel('\theta'), ylabel('\rho'); axis on, axis normal, hold on;
plot(T(P(:,2)),R(P(:,1)),'s','color','white');
```

Эта задача часто встречается при обработке астрономических изображений поверхности планет и астероидов. Актуальность подобных задач в последнее время быстро растет, особенно после падения метеорита около Челябинска.

Операции анализа изображения

При анализе обычно сопоставляются и изменяются свойства объектов изображения. Например, следующая программа выделяет с помощью функции boundaries контуры объектов черно-белого изображения и закрашивает внутреннюю область объектов тремя цветами (рис. 29):

```
I = imread('rice.png'); BW = im2bw(I, graythresh(I));
[B,L] = bwboundaries(BW,'noholes');
imshow(label2rgb(L, @jet, [.5 .5 .5])); hold on
for k = 1:length(B)
    boundary = B{k};
    plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end
```

Рис. 29. Выделение и окраска объектов (бактерий)

Следующая программа напоминает предыдущую, но не строит границу объектов, а окрашивает их целиком и позволяет менять цвет фона (рис. 30):

```
I = imread('rice.png'); subplot(131); imshow(I)
BW = im2bw(I, graythresh(I)); CC = bwconncomp(BW);
L = labelmatrix(CC); RGB = label2rgb(L);
subplot(132); imshow(RGB)
RGB2 = label2rgb(L, 'spring', 'c', 'shuffle');
subplot(133); imshow(RGB2)
```

Обработка изображений печатных плат

Изображения печатных плат легко преобразуются в бинарные, что позволяет исполь-

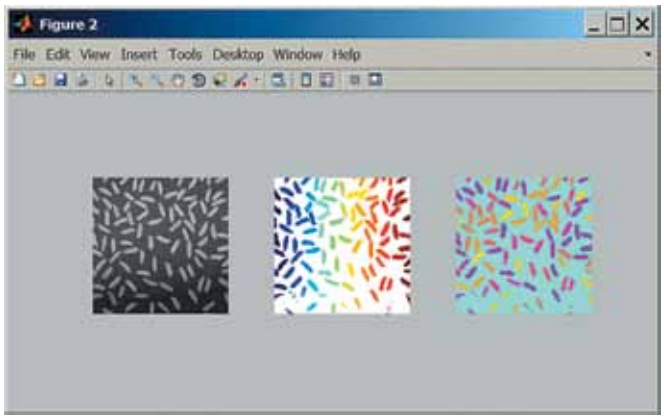

Рис. 30. Замена цвета объектов и фона

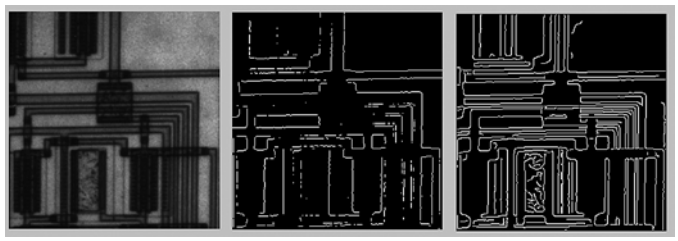

Рис. 31. Оригинал изображения печатной платы и действие функции edge при методе Превита и Канни

зовать обширный аппарат морфологических преобразований для совершенствования проектирования таких плат. Приведенная ниже программа иллюстрирует технику получения границ проводников печатной платы специальными методами Превита и Канни и соответствующими функциями (рис. 31):

```
I = imread('circuit.tif');subplot(131);imshow(I);
BW1 = edge(I,'prewitt');BW2 = edge(I,'canny');
subplot(132); imshow(BW1); subplot(1,3,3); imshow(BW2)
```

Другая программа обеспечивает это методом Канни с поворотом печатной платы (рис. 32):

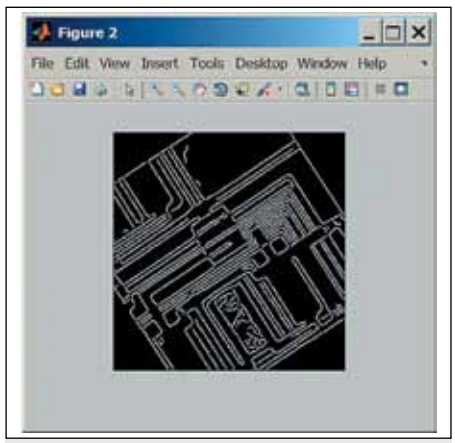

Рис. 32. Поворот изображения печатной платы

```
I = imread('circuit.tif');
rotI = imrotate(I,33,'crop'); fig1 = imshow(rotI);
BW = edge(rotI,'canny'); figure, imshow(BW);
```

Интересно, что при наклонных границах проводников их выделение происходит более четко. Часто важную информацию несет зависимость яркости заданной строки изображения от расстояния, то есть диаграмма профиля (рис. 33):

```
circuitBoard = rot90(rgb2gray(imread('board.tif')));
subplot(211); imshow(circuitBoard); title('Original Image')
offsets0 = [zeros(40,1) (1:40)'];
glcms = graycomatrix(circuitBoard,'Offset',offsets0);
stats = graycoprops(glcms,'Contrast Correlation');
subplot(212); plot([stats.Correlation]);
title('Texture Correlation as a function of offset');
xlabel('Horizontal Offset'); ylabel('Correlation')
```

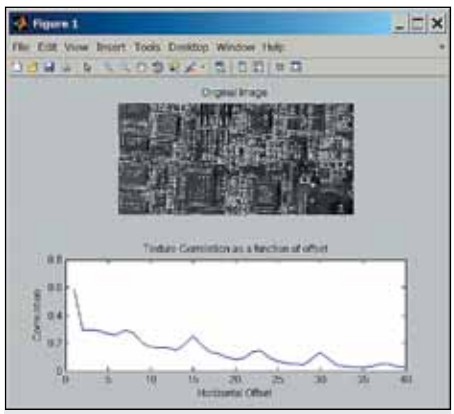

Рис. 33. Зависимость яркости строки от положения в ней пикселя

Построение цветного изображения печатной платы также возможно. Это иллюстрирует следующая программа (рис. 34):

```
BW = imread('blobs.png'); [B,L,N] = bwboundaries(BW);
figure; imshow(BW); hold on;
for k=1:length(B),
  boundary = B{k};
  if(k > N)
    plot(boundary(:,2),...
      boundary(:,1),'g','LineWidth',2);
  else
    plot(boundary(:,2), boundary(:,1),'r','LineWidth',2);
  end
end
```

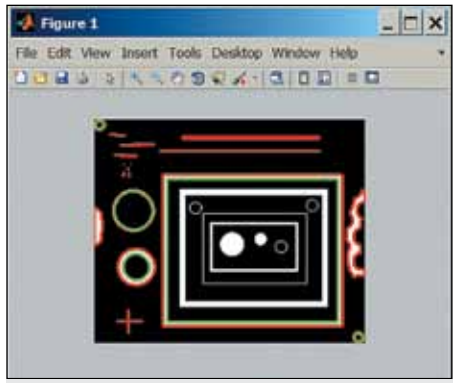

Рис. 34. Выделение и окраска элементов печатной платы

Специальные преобразования изображения

В технике компрессии (сжатия) и декомпрессии изображений часто используются специальные преобразования изображения. Для этого в пакете Image Processing Toolbox имеется ряд функций. Одна из них — **dct2** — осуществляет прямое косинусное преобразование изображения (рис. 35):

```
RGB = imread('autumn.tif'); I = rgb2gray(RGB); J = dct2(I);
subplot(121);imshow(RGB);title('Оригинал');
subplot(122); imshow(log(abs(J)),[]),
colormap(jet(64)), colorbar; title('dct2')
```

Рис. 35. Прямое косинусное преобразование dct

При прямом косинусном преобразовании получается изображение, которое почти не содержит мелких деталей, свойственных исходному изображению. Его спектр намного более узкий, что говорит о существенном сжатии изображения — практически в десятки раз. Такое сжатие используется в формате изображений *.jpeg*. С 2000 года для сжатия стали использовать и вейвлеты [4, 5], но пока

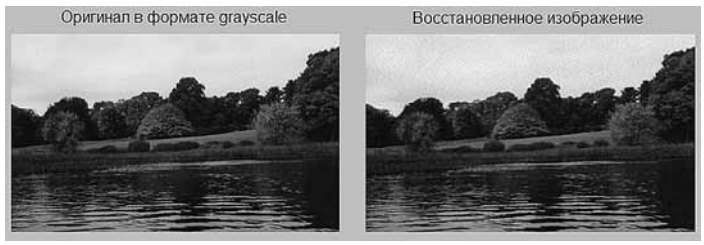

Рис. 36. Оригинальное изображение в формате grayscale и после прямого и обратного косинусного преобразования

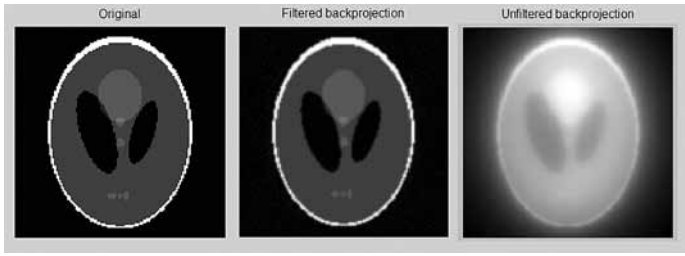

Рис. 37. Прямое и обратное преобразование Радона с промежуточной фильтрацией

в Image Processing Toolbox нет средств для работы с ними.

Косинусное преобразование является обратимым, так что возможно восстановление изображения. В ходе этой операции можно использовать более узкий спектр и отказаться от составляющих сигнала с очень малым уровнем. Особенно эффектно это при преобразовании изображения в формат *grayscale* (рис. 36):

```
J(abs(J) < 10) = 0; K = idct2(J);
subplot(211); imshow(I); title('Оригинал в формате grayscale');
subplot(212); imshow(K,[0 255]);
title('Восстановленное изображение');
```

При обработке фотографий (например, рентгеновских снимков) применяется специальное прямое и обратное преобразование Радона (рис. 37):

```
P = phantom(128); R = radon(P,0:179);
I1 = iradon(R,0:179); I2 = iradon(R,0:179,'linear','none');
subplot(1,3,1), imshow(P), title('Original')
subplot(1,3,2), imshow(I1), title('Filtered backprojection')
subplot(1,3,3), imshow(I2,[]), title('Unfiltered backprojection')
```

Большая группа преобразований связана с изменением форматов данных и чисел и арифметическими операциями с матрицами изображений и цветов. Ввиду их очевидности функции подобных преобразований в [4, 5] не описаны.

Операции с цветом

Цвета каждого пикселя изображения в MATLAB и пакете Image Processing Toolbox задаются числами (кодами). Поэтому операции с цветом сводятся к матричным операциям и реализуются соответствующими функциями цвета. Работа с цветом изображения или его объектов происходит на уровне этих функций (рис. 38):

```
RGB = imread('peppers.png');
M = [0.30, 0.59, 0.11];
gray = imapplymatrix(M, RGB);
subplot(2,2,1), imshow(RGB),
title('Original RGB')
subplot(2,2,2), imshow(gray),
title('Grayscale Conversion')
```

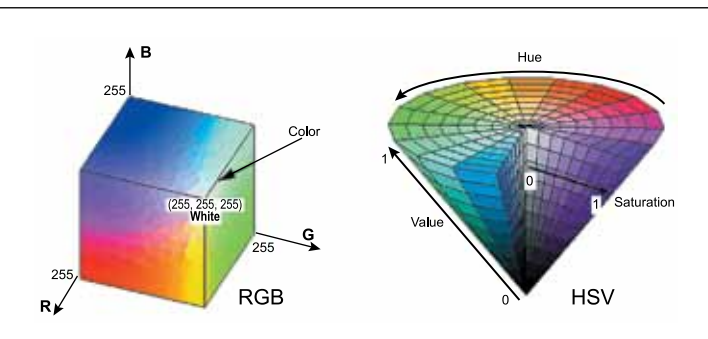

Рис. 39. Куб цветов RGB и конус цветов HSV

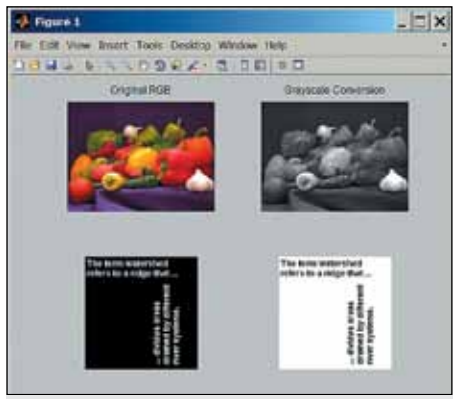

Рис. 38. Преобразование RGB-изображения в grayscale формата PNG и изображения текстового формата в бинарный

```
bw = imread('text.png');
bw2 = imcomplement(bw);
subplot(2,2,3),imshow(bw);
subplot(2,2,4),imshow(bw2)
```

Определенную специфику имеют операции и функции изменения форматов изображений с разными цветами. Мы уже неоднократно применяли преобразования RGB- и grayscale-изображений. В формате RGB для задания кодов цветов R, G и B каждого пикселя используется «куб цветов», показанный на рис. 39 (слева) при 8-битовом кодировании ($2^8 = 256$ цветов).

Отметим еще один популярный формат — HSV. При нем используется «конус цветов» (рис. 39, справа). Для задания цвета нужны три параметра: **Hue** (цвет); **Saturation** (насыщенность) и **Value** (сила цвета). Следующая программа вычисляет и представляет матрицы компонентов и исходное изображение формата HSV (рис. 40):

```
RGB=reshape(ones(64,1)*reshape(jet(64),1,192),[64,64,3]);
HSV=rgb2hsv(RGB); H=HSV(:,:,1); S=HSV(:,:,2); V=HSV(:,:,3);
subplot(2,2,1), imshow(H); subplot(2,2,2), imshow(S)
subplot(2,2,3), imshow(V); subplot(2,2,4), imshow(RGB)
```

Есть ряд функций, поддерживающих и другие системы изображения и цвета — менее распространенные. Их описание можно найти в справке и книгах [2–5].

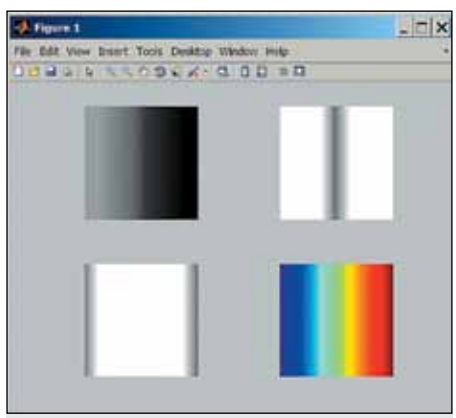

Рис. 40. Составляющие HSV цветного изображения

Взаимодействие с другими пакетами расширения

Пакет расширения Image Processing Toolbox R2012b функционально закончен, для его работы необходима только базовая матричная система MATLAB R1212b. Блочное имитационное моделирование систем и устройств обработки изображений и управления видеопотоками в этом пакете не предусмотрено. Для этой цели служит другой пакет расширения — Image Acquisition Toolbox. Кроме того, в области обработки и фильтрации изображений как сигналов можно использовать пакеты расширения Signal Processing Toolbox, DSP System, Wavelet Toolbox и др. Возможности работы с картографическими изображениями имеются в специальном пакете расширения Mapping Toolbox. Это расширяет возможности базовой системы MATLAB R2012b и позволяет при необходимости применять средства блочного имитационного моделирования Simulink 8.0 (R2012b).

Заключение

Image Processing Toolbox R2012b вобрал в себя обширные возможности предшествующих реализаций этого мощного пакета расширения. Введен ряд новых возможностей и функций. Доступ к возможностям пакета стал более открытым и удобным, особенно к его браузерам изображений. Пакет имеет множество функций, написанных на языке программирования системы MATLAB 8.0 — лучшем для научно-технических расчетов. Порою всего две-три строки вполне понятного машинного кода MATLAB достаточно для решения сложной технической задачи. При этом программные коды открыты для разбора и модификации пользователем. Их можно целиком вводить в командном или программном режимах работы. Дополнительные функции, вводимые пакетом расширения Image Processing Toolbox в систему MATLAB, обеспечивают решение основных задач импорта и экспорта файлов изображений, их обработки, анализа, улучшения качества, ряда преобразований и т. д.

Литература

1. www.mathworks.com
2. Дьяконов В. П. MATLAB R2006/2007/2008 + Simulink 5/6.7. Основы применения. М.: Солон-Пресс, 2008.
3. Дьяконов В. П. MATLAB. Полный самоучитель. М.: ДМК-Пресс, 2012.
4. Дьяконов В. П. MATLAB 6.5 SP1/7//7 P1+Simulink 5.6. Работа с изображениями и видеопотоками. М.: Солон-Пресс, 2010.
5. Гонсалес Р., Вудс Р., Эддинс Э. Цифровая обработка изображений в среде MATLAB. М.: Техносфера, 2006.