

The background of the cover is a dark, textured image of a cathedral tower. Overlaid on this are several smaller images: a high-contrast, black and white image of a cathedral facade in the top right; a pixelated, low-resolution version of the same facade in the middle right; a blue-tinted, slightly blurred version of the facade in the bottom right; and a red line-drawing or edge-detection result of the facade in the center.

Image Processing, Analysis, and Machine Vision

A MATLAB Companion

Tomas Svoboda

Jan Kybic

Vaclav Hlavac

Image Processing, Analysis, and Machine Vision

A MATLAB COMPANION

Tomas Svoboda

Czech Technical University, Prague

Jan Kybic

Czech Technical University, Prague

Vaclav Hlavac

Czech Technical University, Prague





Image Processing, Analysis, and Machine Vision: A MATLAB Companion

by Tomas Svoboda, Jan Kybic, Vaclav Hlavac

General Manager:

Chris Carson

Developmental Editor:

Hilda Gowans

Permissions Coordinator:

Vicki Gould

Production Manager:

Renate McCloy

Interior Design:

Vit Zyka

Cover Design:

Andrew Adams

Compositor:

Vit Zyka

Printer:

Thomson/West

COPYRIGHT © 2008 by Thomson Learning, part of the Thomson Corporation.

Printed and bound in the United States of America

1 2 3 4 10 09 08 07

For more information contact Thomson Learning, 1120 Birchmount Road, Toronto, Ontario, M1K 5G4. Or you can visit our Internet site at <http://www.thomsonlearning.com>

Library of Congress Control Number 2007904151

ISBN: 10: 0-495-29595-7

ISBN: 13: 978-0-495-29595-2

ALL RIGHTS RESERVED. No part of this work covered by the copyright herein may be reproduced, transcribed, or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, Web distribution, or information storage and retrieval systems—without the written permission of the publisher.

For permission to use material from this text or product, submit a request online at www.thomsonrights.com

Every effort has been made to trace ownership of all copyrighted material and to secure permission from copyright holders. In the event of any question arising as to the use of any material, we will be pleased to make the necessary corrections in future printings.

North America

Thomson Learning
1120 Birchmount Road
Toronto, Ontario M1K 5G4
Canada

Asia

Thomson Learning
5 Shenton Way #01-01
UIC Building
Singapore 068808

Australia/New Zealand

Thomson Learning
102 Dodds Street
Southbank, Victoria
Australia 3006

Europe/Middle East/Africa

Thomson Learning
High Holborn House
50/51 Bedford Row
London WC1R 4LR
United Kingdom


Latin America

Thomson Learning
Seneca, 53
Colonia Polanco
11560 Mexico D.F.
Mexico



Spain

Paraninfo
Calle/Magallanes, 25
28015 Madrid, Spain

Introduction to the companion book

The book you are reading is a companion to the textbook *Image Processing, Analysis, and Machine Vision* by M. Sonka, V. Hlavac, and R. Boyle [Sonka et al., 2007]. As the references to the textbook occur often, we will use an icon .


Structure of the companion book

This book provides additional material for readers of . It should assist students, teachers and practitioners to acquire practical understanding in a ‘hands on’ fashion. This book offers the reader problems of varying difficulty and selected algorithms from  (and some additional ones) in **Matlab**.

Matlab has been selected as an implementation language because:

- It is widely used by developers of image analysis algorithms and researchers in the field.
- It allows quick prototyping.
- Our experience is that using **Matlab** in exercises and assignments allows students to concentrate more on algorithms than on programming.
- Most image analysis algorithms coded in **Matlab** can be easily rewritten in other procedural languages such as C, C++ or Java.
- There are many third party implementations of algorithms available in **Matlab**.
- Among other libraries, **Matlab** is accompanied by the Image Processing Toolbox which covers basic image processing and a few image analysis capabilities well. It is easy to use. This book exploits the Image Processing Toolbox instead of rewriting simple algorithms anew. The image processing toolbox is accompanied by a freely downloadable booklet¹.


Problems are classified into three classes according to their level of difficulty; we follow the analogy of classifying difficulty of downhill ski slopes and mark problems by the colors blue, red and black. The blue problems are the easiest.

Blue problems are often questions that may be answered by students in their self-study as a reassurance that the material provided in  was well understood. Most blue problems can be solved immediately or in at most a few minutes with just pen and paper. Teachers may use blue problems in written tests or other assignments.


¹<http://www.mathworks.com/products/image>

Red problems are also mostly of a pen and pencil nature or very short computer exercises. However, they might need more thinking and usually up to half an hour to be solved.

Black problems require more thorough analysis and/or require practical use of computer tools and/or development of short application programs. They can be used as computer exercises or homework assignments for students.

The rest of each chapter contains the **Matlab** implementations of selected algorithms provided in  and also some additional algorithms. This book emphasizes pedagogical aspects—algorithms are presented to demonstrate the principles. While writing the code, the preference was given to *clarity* over broad functionality and maximal processing speed. For instance, we did not consider images of all possible gray levels or color models. Whenever possible the code is fully functional and self-contained. However, we often skip uninteresting parts related, for example, to error detection and default parameter handling. A full version of the codes can be obtained on the book homepage, except for a few rare cases where we did not include it for space or legal reasons.

Each function code starts with a list of input and output parameters in a gray background box. Optional parameters are typeset in a slanted font, see `dft_edu` (p. 20) for an example. The second column contains the parameter sizes for regular matrix parameters, the parameter type (such as structure or function) for non-matrix parameters, and the default value in curly braces for optional parameters.

Chapters of this book have the same general structure as  to allow the reader to work with both books easily. On the other hand, this book is self-contained and can be used on its own by teachers or students who learned the subject by other means.

The WWW page for the companion book

The webpage <http://visionbook.felk.cvut.cz> (book homepage) accompanying this book was created with the intention that the authors will maintain and extend it in the future. The fully commented **Matlab** implementations of methods provided in this book are accessible here for educational purposes.

We anticipate use of the books will generate feedback from readers and the webpage will accumulate it as time goes by.

Useful learning material by others

There are several other commonly used textbooks introducing image processing and analysis which provide experimental material and sample code. Two principal ones are [Gonzalez and Woods, 2002], which is widely used in image processing courses and has a companion book [Gonzalez et al., 2004] providing **Matlab** examples; secondly, [Burger and Burge, 2006] (written in German, with an English version expected in fall 2007) is excellent in its educational aspects. The Java programming environment ImageJ² from the National Institute of Health is taken as its basis, and the authors maintain a webpage with very good examples³.

²<http://rsb.info.nih.gov/ij>

³<http://www.imagingbook.com>

Tomáš Svoboda

*Czech Technical University
Prague, Czech Republic*

svoboda@cmp.felk.cvut.cz
<http://cmp.felk.cvut.cz/~svoboda>



Jan Kybic

*Czech Technical University
Prague, Czech Republic*

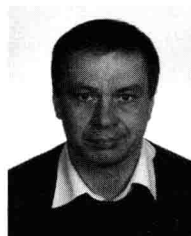
kybic@cmp.felk.cvut.cz
<http://cmp.felk.cvut.cz/~kybic>



Václav Hlaváč

*Czech Technical University
Prague, Czech Republic*

hlavac@cmp.felk.cvut.cz
<http://cmp.felk.cvut.cz/~hlavac>



Contents

Introduction to the companion book	ix
Structure of the companion book	ix
The WWW page for the companion book	x
Useful learning material by others	x
1 Introduction	1
1.1 Problems	1
1.2 Viewing an image: <code>image_view_demo</code>	2
2 The image, its representations and properties	7
2.1 Problems	7
2.2 Displaying a coarse binary image: <code>coarse_pixels_draw</code>	9
2.3 Distance transform: <code>dist_trans_demo</code>	10
2.4 Border of a region: <code>region_border_demo</code>	12
3 The image, its mathematical and physical background	15
3.1 Problems	15
3.2 Convolution, shift-multiply-add approach: <code>conv_demo</code>	17
3.3 Discrete Fourier Transform: <code>dft_edu</code>	19
3.4 Inverse DFT: <code>idft_edu</code>	20
3.5 1D Discrete Fourier Transform: <code>dft1d_demo</code>	21
3.6 2D Discrete Fourier Transform: <code>dft2d_demo</code>	23
3.7 Basis functions for the 2D Discrete Cosine Transform: <code>dct2base</code>	25
3.8 Principal Component Analysis: <code>pca</code>	28
4 Data structures for image analysis	33
4.1 Problems	34
4.2 Matlab data structures: <code>structures</code>	35
4.3 Displaying image values: <code>showim_values</code>	35
4.4 Co-occurrence matrix: <code>cooc</code>	36
4.5 Integral image construction: <code>integralim</code>	39
5 Image pre-processing	42
5.1 Problems	43
5.2 Grayscale transformation, histogram equalization: <code>hist_equal</code>	47
5.3 Geometric transformation: <code>imgeomt</code>	49


vi Contents

5.4	Smoothing using a rotating mask: <code>rotmask</code>	53
5.5	Image sharpening by Laplacian: <code>imsharpen</code>	57
5.6	Harris corner detector: <code>harris</code>	60
5.7	Frequency filtering: <code>buttfilt</code>	62
6	Segmentation I	66
6.1	Problems	66
6.2	Iterative threshold selection: <code>imthresh</code>	69
6.3	Line detection using Hough transform: <code>hough_lines</code>	71
6.4	Dynamic programming boundary tracing: <code>dpboundary</code>	73
6.5	Region merging via boundary melting: <code>regmerge</code>	75
6.6	Removal of small regions: <code>remsmall</code>	78
7	Segmentation II	81
7.1	Problems	81
7.2	Mean shift segmentation: <code>meanshsegm</code>	83
7.3	Active contours (snakes): <code>snake</code>	85
7.4	Gradient vector flow snakes: <code>mgvf</code>	91
7.5	Level sets: <code>levelset</code>	93
7.6	Graph cut segmentation: <code>graphcut</code>	97
8	Shape representation and description	102
8.1	Problems	102
8.2	B-spline interpolation: <code>bsplineinterp</code>	104
8.3	Convex hull construction: <code>convexhull</code>	107
8.4	Region descriptors: <code>regiondescr</code>	110
8.5	Boundary descriptors: <code>boundarydescr</code>	115
9	Object recognition	119
9.1	Problems	119
9.2	Maximum probability classification for normal data: <code>maxnormalclass</code>	122
9.3	Linear separability and basic classifiers: <code>linsep_demo</code>	123
9.4	Recognition of hand-written numerals: <code>ocr_demo</code>	126
9.5	Adaptive boosting: <code>adaboost</code>	127
10	Image understanding	134
10.1	Problems	134
10.2	Random sample consensus: <code>ransac</code>	136
10.3	Gaussian mixture model estimation: <code>gaussianmixture</code>	139
10.4	Point distribution models: <code>pointdistrmodel</code>	144
10.5	Active shape model fit: <code>asmfit</code>	150
11	3D vision, geometry	154
11.1	Problems	154
11.2	Homography estimation from point correspondences—DLT method: <code>u2Hdlt</code>	156
11.3	Mathematical description of the camera: <code>cameragen</code>	159
11.4	Visualize a camera in a 3D plot: <code>showcams</code>	161



11.5	Decomposition of the projection matrix P: <code>P2KRtC</code>	162
11.6	Isotropic point normalization: <code>pointnorm</code>	163
11.7	Fundamental matrix—8-point algorithm: <code>u2Fd1t</code>	164
11.8	Geometrical Explanation of Epipolar Geometry: <code>u2Fd1t_demo</code>	166
11.9	3D point reconstruction—linear method: <code>uP2Xd1t</code>	168
12	Use of 3D vision	171
12.1	Problems	171
12.2	Iterative closest point matching: <code>vtxicrp</code>	171
13	Mathematical morphology	174
13.1	Problems	175
13.2	Top hat transformation: <code>tophat</code>	176
13.3	Object detection using opening: <code>objdetect</code>	178
13.4	Sequential thinning: <code>thinning</code>	180
13.5	Ultimate erosion: <code>ulterosion</code>	183
13.6	Binary granulometry: <code>granulometry</code>	185
13.7	Watershed segmentation: <code>wshed</code>	188
14	Image data compression	190
14.1	Problems	190
14.2	Huffman code: <code>huffman</code>	192
14.3	Predictive compression: <code>dpcm</code>	197
14.4	JPEG compression pictorially, step by step: <code>jpegcomp_demo</code>	203
15	Texture	207
15.1	Problems	207
15.2	Haralick texture descriptors: <code>haralick</code>	209
15.3	Wavelet texture descriptors: <code>waveletdescr</code>	213
15.4	Texture based segmentation: <code>texturesegm</code>	215
15.5	L-system interpreter: <code>lsystem</code>	219
16	Motion analysis	224
16.1	Problems	224
16.2	Adaptive background modeling by using a mixture of Gaussians: <code>bckggm</code>	225
16.3	Particle filtering: <code>particle_filtering</code>	232
16.4	Importance sampling: <code>importance_sampling</code>	241
16.5	Kernel-based tracking: <code>kernel_based_tracking</code>	242
	Acknowledgments	247
	References	248
	Index	252

Chapter 1

Introduction

Chapter 1 of  discusses the image and its formation informally in a broad context. It provides neither formal definitions of image related concepts, nor algorithms to be implemented. The problems formulated below address these general image related considerations. In the experimental part of the chapter, we demonstrate pragmatically how to display an image in **Matlab**.

1.1 Problems

- 1.1. What is the difference between image analysis (or computer vision) on the one side and computer graphics on the other?
- 1.2. Enumerate the principal reasons why computer vision is hard. Compare with [Section :1.2].
- 1.3. Digital signal processing and low-level image processing typically do not interpret image data. Explain what is meant by *interpretation*, trying to use mathematical formalisms for the explanation. What is the benefit of using interpretation in image analysis? On the other hand, what is the main constraint of using interpretation?
- 1.4. (i) Briefly discuss the difference between local and global analysis of images; (ii) Give some advantages and disadvantages of both; (iii) Give two examples of local analysis; (iv) Give two examples of global analysis.
- 1.5. Assuming that an image captured by your camera is digital and in color, split it into red, green, and blue components and display them.
- 1.6. Experiment with the **Matlab** function :**improfile** in the interactive mode. Find three interesting intensity profiles in your image. Discuss why you found them interesting.

1.2 Viewing an image: `image_view_demo`

In the experimental section of this chapter, the intention is to assist the reader in displaying an image using **Matlab** commands. Guidance will be very pragmatic, trusting in the reader's common sense. The example should allow the **Matlab** novice to get some feeling of the software environment and see how an image can be displayed. However, it is assumed that **Matlab** and its Image Processing toolbox are already installed.

The guidance provided at the beginning will be quite detailed and wordy. In later chapters, the accompanying text is going to be much more concise and the **Matlab** code will often speak for itself; this will be necessary to present many advanced algorithms in reasonable space.

The reader probably has a digital camera; assume that we wish to capture an image, display it in **Matlab**, and demonstrate some of the issues discussed in [Chapter 1:1]. Most cameras provide JPEG-compressed color images. Such an image can be copied from the camera to the computer.

We will provide one such image depicting a boy kneeling on a bench and turning his back to the viewer. The colors appearing in the image and the striped sweatshirt will be of use in this example.

Start **Matlab** and wait for the prompt; commands can now be issued. If some command is unclear then consult the extensive **Matlab** help or user documentation: Just type `help` for the former or `doc` for the latter.

The image was captured by an ordinary digital camera and copied to the computer disk, from where it has to be read. It is named `Ondra_sampling.jpg` and is stored in the subdirectory `images`. The suffix `.jpg` suggests that the image is compressed using JPEG. We ignore image format issues at this point because the **Matlab** function `imread` copes with many different formats: the filename suffix suggests to **Matlab** in which format the data are stored.

We will read the image using function `imread` and store it to the **Matlab** variable `boy`, then display it on the screen using function `image`. There are other functions in **Matlab**, which can be used for image visualization, e.g., `imshow` or `imagesc`. This is not important for us at the moment. We will also give the image a title; axes in both coordinate directions (rows, columns) are displayed too. The function `image` does not display square pixels in its default setting. The command `axis image` will achieve this.

```
% read the input color image from a disk
boy = imread('images/Ondra_sampling.jpg');
% display the image in the MATLAB figure
image(boy);
axis image
% add the title to the figure
title('Input color image of a boy');
```

The color image is shown in Figure 1.1, consisting of three color components, red, green and blue. The **Matlab** function `imread` creates a data structure of a **Matlab** internal data type (a multidimensional array) and assigns it to the variable `boy`. This particular multidimensional array is three-dimensional; it can be imagined as a collection of three matrices. The first matrix stores a red component image, the second matrix contains a green component image, and the third represents the blue component image.

Pixel values in each of the three component images correspond to the intensity of the appropriate color. For instance, in the matrix corresponding to the red channel the

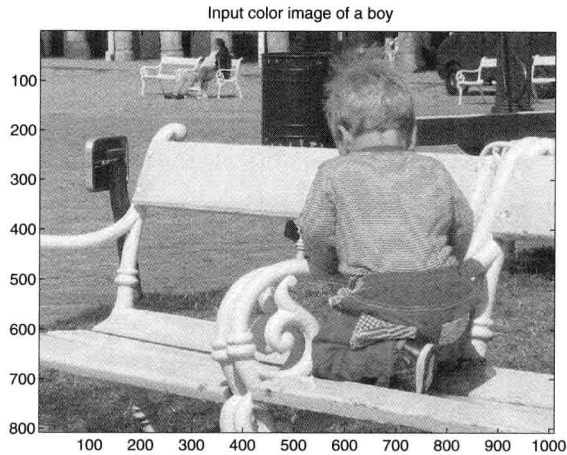


Figure 1.1: A color image of a boy on a bench.

value 0 means that there is no red color and the maximal value (in this particular case value 255 because pixel values in each color channel are encoded in 8 bits) indicates that a saturated red color is present.

There is a very useful **Matlab** command `whos` which lists in alphabetical order all variables in the currently active workspace along with their sizes, types, class and attributes. This text information is written to the **Matlab** command window. If the command `whos` is followed by the name of the variable then only the information about the particular variable is displayed. If the command `whos boy` is issued then the following information appears:

Name	Size	Bytes	Class
boy	808x1010x3	2448240	uint8

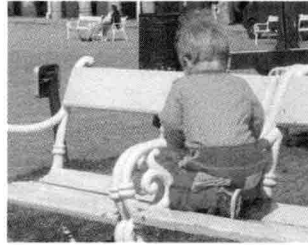
The size information shows that each component image (matrix in **Matlab**) has 808 rows, 1010 columns and there are three such matrices in the multidimensional array. E.g., the **Matlab** expression `boy(157, 205, 3)` provides the pixel value with the row coordinate 157, the column coordinate 205 in the third matrix (in our case the blue channel).

We are now ready to separate the image stored in `boy` into three separate matrices representing individual color components. To understand the expressions involved, we have to use the **Matlab** colon operator, which allows the expression of iterations in (multidimensional) array subscripts concisely. If `A` is a matrix then the expression `A(:,j)` points to the j^{th} column of matrix `A`. Alternatively, the expression `A(i,:)` points to the i^{th} row. In our case, the expression `boyR = boy(:,:,1)` selects the first matrix of our three dimensional array and creates an appropriate matrix with the red channel component image.

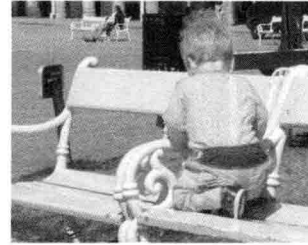
```
% extract individual color components
boyR = boy(:,:,1); % image with the red channel content
boyG = boy(:,:,2); % image with the green channel content
boyB = boy(:,:,3); % image with the blue channel content
```

Having separated individual color channels we are ready to display them; the following commands do the job. Four images are displayed in Figure 1.2.

```
figure; % create a new MATLAB figure
% draw four images into the figure
subplot(2,2,1), subimage(boy), axis off, title('boy—color image');
subplot(2,2,2), subimage(boyR), axis off, title('boyR—red channel');
subplot(2,2,3), subimage(boyG), axis off, title('boyG—green channel');
subplot(2,2,4), subimage(boyB), axis off, title('boyB—blue channel');
```



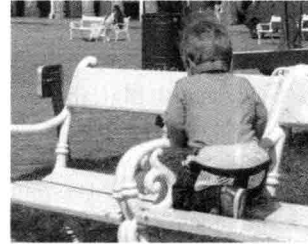
(a) boy—color image



(b) boyR—red channel



(c) boyG—green channel



(d) boyB—blue channel

Figure 1.2: (a) A color image is separated into three color component images; (b) the red channel image; (c) the green channel image, and (d) the blue channel image.

Have a look at the color image and the three color component images, and try to understand how particular colors manifest in components. The red trousers and a blue stripe around his waist are good candidates for inspection.

A color image can be converted to a grayscale image; the values of the three constituent color channels are taken as input. Matlab has a function `rgb2gray` which performs the conversion. The other expressions involved are used to display the image in a proper form. The grayscale image can be seen in Figure 1.3. Ignore for a while the red line segment in Figure 1.3 and the Matlab code in the bottom line which draws this line segment into the image. The role of the red line segment will be explained soon.

```
boyGray = rgb2gray(boy); % convert the color image into a grey-level one
figure; % create a new MATLAB figure
image(boyGray); % display the grayscale image
colormap(gray(256)); % use the appropriate color map
axis off % switch off the axes with scales
% create a line segment between pixels (460,140) and (872,457),
% values given in (row,column) coordinates
r1 = 460; c1 = 140; r2 = 872; c2 = 457;
% draw the line to the picture
line([r1, r2], [c1, c2], 'Color','r', 'LineWidth',3);
```

Humans are very good at understanding images because they have the ability to grasp the global situation. The human eye is also an excellent and precise sensor for relative measurements, i.e., for comparison with other reference values. However, the eye performs badly as a measurement device of absolute values. In image analysis, such quantitative measurements are often needed if a human wishes to understand properties of image data. For instance, we humans do not see noise properties in images well.

There is a very useful visualization tool permitting measurement of images quantitatively by image profiling. The idea is to cut the image along some curve (profile)—a line segment is the simplest instance of such a curve. The image values of the image can be plotted as a one-dimensional function on which the details are much more clearly visible.

Matlab contains a function `improfile` which implements an image profiling tool. If it is run without parameters then it allows the user to specify a line with a mouse along which the profile is going to be displayed. This mode is often used. We use the other option allowing us to enter the line endpoints off-line. In our demonstration, we use Figure 1.3 as input. The line segment is selected in such a way that it shows the changing intensity values across the stripes on the sweatshirt: it is shown in Figure 1.3 in red.

Now we are ready to display the intensity profile using the Matlab expression `improfile`. The code for adding the title to the figure and the label of the y -axis is self-explanatory.

```
% create a new MATLAB figure
figure;
% calculate and display the intensity profile
% along the line segment created earlier
improfile( boyGray, [r1, r2], [c1, c2] );
ylabel('Pixel value');
title('Intensity profile along the line segment');
```



Figure 1.3: A grayscale image of a boy on a bench obtained from the color image in Figure 1.1. A line in red is added to the image along which to visualize an intensity profile, see Figure 1.4.

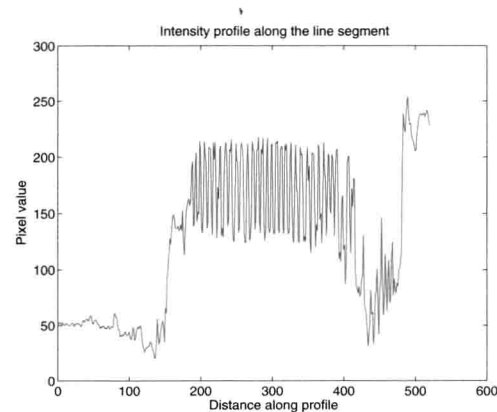



Figure 1.4: The intensity along the line segment profile.

Look at the intensity profile in Figure 1.4. The intensities are ordered from left to right on the x -axis. This ordering matches the orientation of the chosen line segment. Before reading further, consider how the line segment is oriented in the image to provide the shown profile. Is it from the top-left to bottom-right or vice versa?

Try to understand to which pixels in the image the values of the profile correspond. The stripes on the boy's sweatshirt are close to the image resolution limit given by the Shannon sampling theorem, see [Chapter :2].

Chapter 2

The image, its representations and properties

This chapter introduces basic concepts needed to pursue image processing and analysis. A 2D grayscale image is represented by a scalar function f of two variables which give coordinates in a plane: the value of the image function $f(x, y)$ is called the intensity. In many cases, a 2D image is formed as the result of a projection of a 3D scene into 2D.

2.1 Problems

- 2.1. Explain the notion of the *image* (continuous image function) $f(x, y)$ or $f(x, y, t)$. What do parameters x, y, t correspond to? Give several examples of ‘real life’ images captured using different physical principles.
- 2.2. In signal/image discretization, the distance between (equidistant) samples is governed by Shannon’s sampling theorem. Explain informally what the theorem says. Consider both the domain of frequencies and distances between samples. (NB: The mathematics of the sampling theorem is explained in detail in [Chapter 4:3].)
- 2.3. Consider image sampling. As in 1D signal analysis, the distance between equidistant samples is determined by Shannon’s theorem. In the 2D case, there is one more issue to be solved: mutual arrangement of samples in the plane. Discuss what spatial arrangements of samples are commonly used. What are their advantages and disadvantages? (Note: This question does not concern image quantization.)
- 2.4. What is the advantage of hexagonal grid sampling (similar to a honeycomb)? Why is the hexagonal spatial arrangement not used in most digitizers?
- 2.5. What is image quantization? How many quantization levels are needed if a human is observing the image? What artifact appears in the image when there are fewer quantization levels than necessary?
- 2.6. Why do analog television norms such as NTSC, PAL (which are several decades old) use interlaced image lines?

- 2.7. Explain what is meant by (i) *spatial* resolution; (ii) *spectral* resolution; (iii) *radio-metric* resolution; and (iv) *time* (also temporal) resolution.
- 2.8. Define (i) *Additive* noise; (ii) *multiplicative* noise; (iii) *Gaussian* noise; (iv) *impulsive* noise; and (v) *salt-and-pepper* noise.
- 2.9. Define a *region* in a two-dimensional binary image.
- 2.10. The relation ‘being contiguous’ between two pixels of a binary digital image (i.e., those pixels between them there is a path) induces a decomposition of the binary image (of a set) into classes of equivalence (regions). Which three properties must the relation ‘being contiguous’ fulfill to be equivalence? Verify these three properties in the specific case of the relation ‘being contiguous’.
- 2.11. (i) Define a convex region in a 2D binary image; (ii) draw an example of a convex and a non-convex region; (iii) define a convex hull; (iv) draw a convex hull of an example non-convex region from question (ii).
- 2.12. Explain the notion of *palette* in a color image.
- 2.13. Find a grayscale image. Create a palette that makes the image negative. For a Matlab implementation, see `colormap`.
- 2.14. The camera captures a 3D scene and projects it into 2D. The simplest mathematical approximation of such image capture is the pin-hole model which mathematically corresponds to perspective projection. A point in 3D (x, y, z) is projected as a point (x', y') in the image plane. Draw a schematic figure which depicts such a projection—you can simplify it by one dimension by depicting the situation in the plane $x = 0$. Assume that the 3D coordinates (x, y, z) and focal length (i.e., the distance of the image plane from the center of projection) are known. Derive the relation for y' . Is this relation linear? Why?
- 2.15. An interlaced television signal of 50 half-frames per second is sampled into the discrete image (matrix) of 500×500 pixels in 256 gray levels. Calculate the minimal sampling frequency in kHz (kilohertz) which has to be used in the frame-grabber performing analog to digital conversion.
- 2.16. For each uppercase printed letter of the 26 letter English alphabet, determine the number of lakes and bays it has. Derive a look-up table that lists the candidate letters, given the number of lakes and bays. Comment on this quality of this *feature* as an identifier of letters.
- 2.17. Write a program that computes an image histogram; plot the histogram of a range of images.
- 2.18. Develop a program that reads an input image and manipulates its resolution in the spatial and gray domains. Use only subsampling, do not apply any interpolation. For a range of images (synthetic, of man-made objects, of natural scenes, etc.) conduct experiments on the minimum resolution that leaves the image recognizable. Conduct such experiments on a range of subjects.
- 2.19. Implement chamfering on a rectangular grid, and test it on a synthetic image consisting of a (black) subset of specified shape on a (white) background. Display the results for a range of shapes, basing the chamfering on the: (i) Euclidean metric D_E ; (ii) city block metric D_4 ; and (iii) chessboard metric D_8 .
- 2.20. Implement chamfering on a hexagonal grid and display the results.