Dietrich Paulus
Joachim Hornegger

Applied Pattern Recognition
A Practical Introduction
to Image and Speech
Processing
in C++

DRAFT for Edition 4

Version of 15th February 2001
To Dorothea, Belinda, and Dominik

1st edition as
Pattern recognition of images and speech in C++

2nd edition

Reprint with corrections

In the text we use the following names which are protected, trademarks owned by a company or should be emphasized otherwise: GNU, HP, AT&T, Unix, PostScript \LaTeX, HTK, Entropic, Corel Draw.
# CONTENTS

## Part I Introductions

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Images and Sound</td>
<td>5</td>
</tr>
<tr>
<td>1.2</td>
<td>Applications of Pattern Recognition</td>
<td>6</td>
</tr>
<tr>
<td>1.3</td>
<td>Environment, Problem Domain, and Patterns</td>
<td>7</td>
</tr>
<tr>
<td>1.4</td>
<td>Characterization of Pattern Recognition</td>
<td>8</td>
</tr>
<tr>
<td>1.5</td>
<td>Speech Recording</td>
<td>9</td>
</tr>
<tr>
<td>1.6</td>
<td>Video Cameras and Projections</td>
<td>11</td>
</tr>
<tr>
<td>1.7</td>
<td>From Continuous to Digital Signals</td>
<td>13</td>
</tr>
<tr>
<td>1.8</td>
<td>Sampling Theorem in Practice</td>
<td>14</td>
</tr>
<tr>
<td>1.9</td>
<td>Visualization and Sound Generation</td>
<td>15</td>
</tr>
</tbody>
</table>

## 2 From C to C++

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Syntax Notation</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>Principle of C++ Compilation</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Function Calls and Arguments</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>Declaration and Definition of Variables</td>
<td>20</td>
</tr>
<tr>
<td>2.5</td>
<td>Unix-File Access via Standard Functions</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>Numeric Expressions</td>
<td>23</td>
</tr>
<tr>
<td>2.7</td>
<td>Main Program</td>
<td>24</td>
</tr>
<tr>
<td>2.8</td>
<td>Function Definition</td>
<td>25</td>
</tr>
<tr>
<td>2.9</td>
<td>Scope and Lifetime</td>
<td>26</td>
</tr>
</tbody>
</table>

## 3 Software Development

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Software for Pattern Recognition</td>
<td>29</td>
</tr>
<tr>
<td>3.2</td>
<td>Software Development and Testing</td>
<td>30</td>
</tr>
<tr>
<td>3.3</td>
<td>Modular and Structured Programming</td>
<td>32</td>
</tr>
<tr>
<td>3.4</td>
<td>Comments and Program Layout</td>
<td>32</td>
</tr>
<tr>
<td>3.5</td>
<td>Documentation</td>
<td>33</td>
</tr>
<tr>
<td>3.6</td>
<td>Teamwork</td>
<td>34</td>
</tr>
<tr>
<td>3.7</td>
<td>Tools for Software Development</td>
<td>34</td>
</tr>
<tr>
<td>3.8</td>
<td>Efficiency</td>
<td>35</td>
</tr>
<tr>
<td>3.9</td>
<td>Approximation</td>
<td>36</td>
</tr>
</tbody>
</table>

## 4 Control and Data Structures

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Structures</td>
<td>39</td>
</tr>
<tr>
<td>4.2</td>
<td>Enumerations</td>
<td>40</td>
</tr>
<tr>
<td>4.3</td>
<td>Scope Resolution</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>Unions</td>
<td>41</td>
</tr>
<tr>
<td>4.5</td>
<td>Bit- and Shift Operations and Bit-Fields</td>
<td>41</td>
</tr>
<tr>
<td>4.6</td>
<td>Logical Values and Conditionals</td>
<td>42</td>
</tr>
<tr>
<td>4.7</td>
<td>Loops</td>
<td>44</td>
</tr>
<tr>
<td>4.8</td>
<td>Exception Handling</td>
<td>45</td>
</tr>
<tr>
<td>4.9</td>
<td>Switches</td>
<td>46</td>
</tr>
</tbody>
</table>

## 5 Arrays and Pointers

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Vectors and Matrices</td>
<td>49</td>
</tr>
<tr>
<td>5.2</td>
<td>Pointers</td>
<td>51</td>
</tr>
<tr>
<td>5.3</td>
<td>Vectors vs. Pointer</td>
<td>51</td>
</tr>
<tr>
<td>5.4</td>
<td>Pointer Operations and Allocation</td>
<td>52</td>
</tr>
<tr>
<td>5.5</td>
<td>Pointer to Structures</td>
<td>54</td>
</tr>
</tbody>
</table>
Part II Object–Oriented Programming

9 Object–Oriented Programming 97
  9.1 Object–Oriented Software Techniques 97
  9.2 Basic Vocabulary 98
  9.3 Data Abstraction and Modules 99
  9.4 Inheritance 100
  9.5 Templates 101
  9.6 Abstract Classes 102
  9.7 Object–Oriented Classification 103
  9.8 Polymorphism 103
  9.9 Object–Oriented Programming Languages 104

10 Classes in C++ 107
  10.1 Methods and ADT’s 107
  10.2 Class Declarations 109
  10.3 Object Construction 109
  10.4 Destruction of Objects 111
  10.5 Operators 112
  10.6 User-Defined Conversion 113
  10.7 Advanced Methods and Constructors 114
  10.8 Vector Class 114
  10.9 Class Design 117

11 Representation of Signals 119
  11.1 Array Class 119
  11.2 Templates in C++ 121
  11.3 Images 122
  11.4 External Data Formats 123
  11.5 Binary Images 125
  11.6 Color Images 125
  11.7 Subimages 127
  11.8 Matrix Operations 128
  11.9 Speech Signal Class 129

12 Signals and Images 131
  12.1 Synthetic Sound 131
  12.2 Geometric Patterns 132
  12.3 Noise Signals 132
  12.4 Combination of Signals 134
  12.5 Simple Image Manipulation 134
  12.6 2–D Views of 3–D Polyhedral Objects 135
  12.7 Single Stereo Images 137
  12.8 Textures 138
  12.9 Audio and Video Devices 138

13 Fourier Transform 141
  13.1 Introductory Considerations 141
  13.2 Fourier Series 142
  13.3 Fourier Transform 144
  13.4 Discrete Fourier Transform 148
  13.5 Complex Number Class 149
13.6 Inverse Discrete Fourier Transform ............................................. 150
13.7 Fourier Transforms of Speech Signals ......................................... 151
13.8 Fast Fourier Transform .................................................................. 152
13.9 2-D Fourier Transform ................................................................... 153

14 Inheritance for Classes ................................................................. 157
14.1 Motivation and Syntax .................................................................. 157
14.2 Access to Members of Base Classes ............................................. 158
14.3 Construction and Destruction ......................................................... 159
14.4 Pointers to Objects ...................................................................... 160
14.5 Virtual Functions ......................................................................... 162
14.6 Abstract Classes ........................................................................... 163
14.7 Image Class Hierarchy ................................................................... 164
14.8 Multiple Inheritance ..................................................................... 165
14.9 Implementation Issues .................................................................. 167

15 Edge Images ................................................................................... 169
15.1 Strategies ...................................................................................... 169
15.2 Discrete Derivatives of Intensity Functions .................................... 170
15.3 Mask Operators ............................................................................ 172
15.4 Discrete Directions ....................................................................... 173
15.5 Edge Class ..................................................................................... 175
15.6 Edge Images .................................................................................. 176
15.7 Robert’s Cross .............................................................................. 178
15.8 Second Derivative ......................................................................... 178
15.9 Color Edge Operators .................................................................... 179

16 Class Libraries .................................................................................. 183
16.1 Stream Input and Output ............................................................... 183
16.2 National Institutes of Health Class Library ..................................... 185
16.3 Input and Output for Objects ......................................................... 188
16.4 Frequently Used Classes ............................................................... 190
16.5 Collection Classes ......................................................................... 191
16.6 Memory Allocation ........................................................................ 193
16.7 Standard Template Library ........................................................... 195
16.8 Advanced C++ Features ............................................................... 198
16.9 Templates vs. Inheritance ............................................................. 200
21.6 Mel Spectral and Cepstral Features ........................................ 281
21.7 Linear Predictive Coding .................................................. 283
21.8 Model Spectrum and Cepstrum ............................................... 286
21.9 Implementation Issues ..................................................... 286

22 Numerical Pattern Classification ............................................. 289
  22.1 General Notes on Classifiers ............................................. 290
  22.2 Design of Classifiers .................................................... 291
  22.3 Linear Discriminants .................................................... 292
  22.4 Polynomial Classifiers .................................................. 296
  22.5 Bayesian Classifiers ..................................................... 296
  22.6 Properties of Bayesian Classifiers .................................... 298
  22.7 From Bayesian to Geometric Classifiers ............................... 300
  22.8 Nearest Neighbor Classifiers .......................................... 301
  22.9 Implementation of Classifiers ......................................... 303

23 Speech Recognition .......................................................... 307
  23.1 Classification of Speech Signals ....................................... 307
  23.2 Dynamic Time Warping .................................................. 308
  23.3 Mixture Densities ........................................................ 314
  23.4 Hidden Markov Models ................................................... 318
  23.5 Topological and Statistical Variations ................................. 322
  23.6 Incomplete Data Estimation ............................................. 323
  23.7 Learning from Multiple Observations ................................ 325
  23.8 Hidden Markov Model Classes ......................................... 328
  23.9 Statistical and Neural Speech Understanding ........................ 330

24 An Image Analysis System ................................................... 333
  24.1 Design of PUMA and ANIMALS ........................................ 333
  24.2 Hierarchy of Picture Processing Objects ............................... 335
  24.3 Segmentation Objects .................................................... 340
  24.4 External Representation ................................................ 342
  24.5 Graphical User Interfaces .............................................. 343
  24.6 Display ........................................................................... 345
  24.7 Computer Vision ............................................................ 345
  24.8 Object Recognition ........................................................ 347
  24.9 Model-Based Image Analysis ............................................ 348
Chapter 1

Pattern Recognition

In this chapter we will briefly introduce the basic ideas and the models used in pattern recognition. We exclude biological aspects and treat only the mathematical and technical aspects of perception. This is done in a very informal way, since it is not within the scope of this book to present a rigorous discussion of pattern recognition theory. We put our main emphasis on explaining image and speech processing concepts. The research problems treated are motivated by practical examples. After a brief introduction to the applications of pattern recognition, a sketched mathematical description of patterns, problem domain, and environment is given. Due to the fact that modern computer systems need digital data, we will also discuss the central problem of how continuous, observable signals can be transformed into digital signals. More technical description can be found in the literature (e.g., in [Bis95, Nie83, Rip96]).

1.1 Images and Sound

The basic input data to any pattern recognition system is recorded in the form of digitized signals. These digitized signals are then processed by the system. Images as well as speech are typical examples of input data and represent the most important areas in the research and application of pattern recognition.

Digital images and speech signals are very common in today’s computer and audio-visual equipment. Digital high-definition video is becoming a huge market. Almost all personal computers have video and audio capabilities and publishing programs now enable the mixing of digital images with text, thereby creating new, so called, hypertext documents. PC users are familiar with the JPEG and MPEG standards, which are often used for image transmission. Image data formats (like TIFF e.g., [Poy92]) are compatible across hardware borders. Special hardware for video conferences using personal computers and standard computer communication networks are also now available. Several types of media are commonly used in conjunction with each other: text, speech, pictures, movies, etc. The combination of these many media sources and their uses is called multimedia.

1 see the reference website for MPEG http://www.bok.net/~tristan/MPEG/ for further information
Digital signals can be synthesized by a computer based upon a description given to it; sound can be generated by a synthesizer or a voice generator and images can be created by computer graphics. Natural signals are recorded by special devices; sound, for instance, is recorded by microphones and images are captured by cameras. The quality of sensor data is dependent on the sensor used.

The treatment of sensor data is called signal processing. If a computer tries to “understand” what a natural signal “means”, then we call this process pattern recognition and analysis. The terms “image processing” and “speech processing” are used as general terms for signal processing as well as the analysis of images and speech. The relation between graphics and image analysis is shown in Figure 1.1. In this book, we cover image and speech processing and the recognition of visual and audio signals: this is different from algorithms that treat visualization or sound generation.

1.2 Applications of Pattern Recognition

Applications of pattern recognition can be found in several areas. For instance, industry, medicine, and the military make extensive use of pattern recognition techniques. Image processing of satellite images [Jäh93], automatic and computer-aided medical diagnosis based on X-ray or MR-images [Udu91, Weil95], robot control using visual information [Rim91], and autonomous vehicles [Tal93] serve as common examples. Other applications are automatic postal address reading systems [Sch78] or the development of an electronic appointment diary [Bub96], where the interface is a system for handwritten character recognition. Bar code readers are commonly used in banks and shopping centers.

Acoustic communications with computers, dialogue systems, and speaker recognition are potential applications of speech processing. Already car telephones are available with which you can dial using just your voice.

Other applications may be found, for example, in seismic processing where the input signal comes from a seismic sensor. Other signals are processed in medicine such as sounds of the heart or signals from the brain (which have more similarity to speech processing than to image processing).
• if necessary, { } pairs are used to indicate what is to be repeated, or where the alternatives refer to
• a + indicates at least one repetition,
• terminal strings (i.e. those strings which will literally appear in the source code) are typed in teletype and are underlined.

An example including several of these features is shown in the syntax of floating point numbers. Verbally, it reads as “an integer number is a sequence of at least one digit, a real number can have either a sequence of digits before an optional dot followed by the fraction which may be omitted, or the integer part may be omitted, in which case the fraction must be present.”

Syntax:

digit := 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
int_number := digit+
real_number := {int_number+ int_number*} | {int_number* [.] int_number+}
signed_real_number := [{+-} int_number*]

When an intuitive description is simpler than a formal definition, we either mix the style or use a verbal description only. The following is an example of a syntax definition for comments in C++.

Syntax: comment := // any text until end of line

The syntax description do often not cover all variants but serve as a guideline for the most common cases. A complete formal definition of the C++ language can be found in the appendix of [Str97].

2.2 Principle of C++ Compilation

The source code of C++ programs and, similarly, of C programs is translated by a compiler. By convention, C source files have the extension .c and the C++ files end in .C. Initially, sources are pre-processed by a program called cpp. In this step all lines beginning with the symbol # are evaluated. Except for comments, no information other than pre-processor directives may be present on these lines.

First, we consider the lines starting with #include followed by a filename in angle brackets as <system-file> or in quotes as in "personal-file". In both cases, a temporary file is generated by the compiler, where the corresponding system and personal files are explicitly inserted (gray box in Figure 2.1). By convention, included files usually have the extension

2 Operating systems which do not distinguish upper and lower case letters in file names use .cxx and .cpp to denote C++ programs.
3 Some non-Unix systems may call it differently or work without it. The principle of compilation is, however, the same. Usually, C code is pre-processed by a program called cpp; C++ compilers use the pre-processor as well.
4 Some systems also allow compiler directives such as hints for optimization or parallel implementation here.
.h”; they are called header-files. The path entries of the compiler are searched for the files included by \#include <file>. When \#include "file" is used, the compiler searches for the file in the current directory first, before looking at the default compiler path.

Most of the files searched for by <file.h> are part of the environment for the compiler or operating system. They may be found at a common place for all users of the system. Private files "file.h" will often be used by only one user.

The temporary file produced is then compiled successively by one or more programs contained in the compiler. Usually, two compiler passes produce an object module, which has the extension .o or .obj. Executables are then created by a linker which resolves external symbols from the system libraries and adds the interface to the operating system. Alternatively, the object module may be added to create or modify a library (usually with extension .a or .lib).

Figure 2.1 shows the data flow of the compile process annotated by the required programs.

2.3 Function Calls and Arguments

We now present a brief introduction to C and C++ functions and their arguments. In doing so, we concern ourselves solely with how to deal with constant arguments and function call syntax; the remaining details about functions are given later (Sect 2.8). Here is a very simple example of a program (see Listing 1). It consists of a main function (always called main) and a pre-processor directive (#include). The code is syntactically correct both in C and C++.

\[5\] If dynamic linkage is used (so called “shared libraries”), the resolution of external symbols from the libraries happens at program runtime or when starting the program. Building such libraries requires special treatment which is described in compiler manuals.

\[6\] The examples for programs are numbered consecutively. An index can be found on the page.
### Index

- abstract data type, 120
- access
  - control, 440
  - to member, 185
- accumulator, 325–328
- active vision, 82, 288, 426, 430
- Ada, 126, 230
- allocation, 65
- ANIMALS, 46, 251
- ANSI–C, 21, 91
- artificial intelligence, 73, 81, 428
- ASCII, 236
- atan2, 203, 212
- atof, 35, 95, 97
- atoi, 32, 35, 62, 95, 97
- autocorrelation, 354
- autostereogram, 273, 274
- base class, 188
  - access, 185
  - private, 184
  - protected, 184
  - public, 184
- Bayes Classifier, 383, 405
  - decision rule, 384, 405
  - error, 375
- Bellman principle, 400
- binarization, 293
- bipartition, 295, 296
- bispectrum, 360
- bit field, 53
- block, 352
- bool, 54
- break, 56, 57
- byte, 67
- C–beautifier, 42
- CASE, 38
- cast, 140, 188
- catch, 58
- CCD camera, 9, 13, 150, 269, 287
- CD, 114
- chain code, 240, 318, 335
  - conversion, 343
  - length, 340
  - smoothing, 335
- char, 29
- chi–square–test, 337
- class, 119, 124
  - abstract, 124
  - AffineDist, 258
  - Bag, 224, 376, 422
  - Chain, 241, 242, 335
  - ChainSeq, 240
  - Class, 217
  - Collection, 224
  - ColorImage, 154
  - Date, 222
  - declaration, 131
  - design, 141
  - Dictionary, 225
  - Edge, 312, 315
  - EdgeImage, 206, 312
  - GeoObj, 245
  - GrayLevelImage, 119
  - hierarchy, 183
  - HipposObj, 235, 236
  - Histogram, 303
  - Image, 206, 237, 239
  - image hierarchy, 192
  - instance, 117, 119
  - library, 126, 213
  - LineRep2D, 335
  - Stack, 224
  - Matrix, 119, 149, 239
  - Number, 124
  - Object, 216, 225, 235
  - OIOxdrin, 248
  - OIOxdrout, 248
  - OrderedCltn, 224
  - PointXY, 49
  - polymorphic, 189
  - RelObj, 237
  - Represent, 237
  - SegObj, 246, 247, 318
  - SegCltn, 224, 422
  - Set, 224, 318
  - Stack, 224
  - String, 222, 225
  - StrLineSeg, 333
  - template, 148
  - Time, 219, 222
  - vector, 139
  - XDR, 236
  - classification, 74, 101, 373, 395
  - architecture, 74
  - numerical, 74
  - syntactical, 74
  - classifier, 346, 375
  - consistent, 375
  - speech, 395
  - clients, 122
  - color, 10, 16, 37, 153, 154, 192, 209, 212, 239, 252, 255, 293, 304, 350, 426, 427
  - space, 154
  - command line, 31, 32, 35, 42
  - compilation, 23
  - conditional, 131
  - computer algebra, 303
  - consistency, 375
  - const, 27, 139
  - constructor, 132, 146, 186
  - default, 133
  - reference, 133
  - continue, 56
  - contrast, 300
  - conversion, 137
  - operator, 137