subject which relies on precise logical formuation. For example, the distinction between may and can seems to have eluded the author just as it seems to elude many undergraduates, when at times the distinction is most important. Another omission which is rife is the hyphen between compound words; but since this endemic perhaps it too should be forgiven. Fortunately the style improves and by Chapter 2 the text becomes more interesting to read, even though many sentences remain irritatingly terse. Another change of style occurs at the start of Chapter 3 where the prose becomes more decisive, indicating that the author is more confident with the technical details of the design process than general introductory comments.

Chapters 4–9 represent the body of the text and here important ideas are introduced in a methodical manner. Considerable time is spent explaining the nature and variety of faults that can exist in real circuit implementations and how difficult it is to model all possible fault conditions. The complexity of the arguments and ideas increases rapidly and students of computer science may have difficulty in keeping up with some aspects of electronic engineering. Similarly programming concepts such as 'a table based event driven simulator' may be difficult to grasp by students of engineering. This is an intrinsic difficulty of the subject and the author does well to maintain a sensible balance for readers new to the problems of digital simulation. Of key importance is completeness of formulation and the astute reader will realise that there are many subtle unanswered questions. For the experienced reader much that is already taken for granted is placed on a more formal footing and some *ad hoc* design rules may be seen in a new light. In particular the chapter on Models and Model Design show just how difficult it is to model even a simple flip–flop!

As a definitive text on the design and use of simulators the style in the early chapters is too informal, and as an introductory text for undergraduates it lacks any tutorial examples. This is not to say that it is not worth reading, since the author is correct when he points out that there are very few texts in this area. Of considerable potential help is the extensive list of over 150 references to related technical papers.

DAVID C. DYER
University of Warwick

J. R. PARKER
Practical Computer Vision Using C. John Wiley, 1993, 476 pp., softbound, £24.50, ISBN 0 471 59262 5 (book/disk), 0 471 59259 5 (paper), 0 471 59411 3 (disk)

This volume from Wiley is intended to provide the non-mathematical reader with a basic, wide-ranging and extremely practical cookbook of standard low-level computer vision procedures. It is optionally accompanied by a diskette (my review copy wasn't) which contains the entire suite of algorithms described, written in Borland C. Most of the core routines are also printed in the book, about half of which is given over to code listings which appear at the end of each chapter. Dotted throughout the text are stripped-down code and pseudocode listings intended to illustrate the basic structure of each procedure.

The book is organized by increasing complexity. The early chapters review image types and describe simple operations on two-level images such as the measurement of geometric properties of regions, before moving on to greyscale images and basic operations such as thresholding. Chapters on simple feature extraction, object counting and classification follow, and the final chapters cover computer-readable codes, optical character recognition and the analysis of scientific imagery with examples from astronomy (stellar photometry) and biology (DNA gel electrophoresis). The three appendices contain source code descriptions, a partial review of available imaging software and an excellent bibliography which could perhaps also have included the near-standard reference to Geman and Geman in the simulated annealing section, and some reference to the neural network literature under character recognition and classification.

What the book is not, and does not attempt to be, is a complete and rigorous treatment of the field of computer vision. As the author writes in the preface, the book is pitched at 'the general computing public and students of subjects in which computer vision is a useful tool'. The mathematical content is minimal and instruction is by copious examples which are intended to generate intuition in the reader. Inherently mathematical topics are omitted altogether; there is no coverage, for example, of restoration save for a very heuristic treatment of small linear masks ('Giving a higher weight to the center pixel seems like a good idea') and median filters. There is no general treatment of filtering; simple filter masks are introduced on an *ad hoc* basis as, for example, edge or line detectors, without reference to the optimality criteria which characterize the more academic signal processing approach to such problems.

While a desire for currency is commendable, the publisher really should employ a proofreader even if it delays publication by a week. I found quite a number of word-substitution errors in the text which had presumably survived a spelling check. There were also some typographical errors in formulae, and figures which did not match their captions. The author boldly states that the DNA double helix is a string of *amino acids* (emphasis his) rather than nucleotide bases, which howler should also have been spotted before causing public embarrassment.

These minor grumbles apart, though, the book is a fairly clear and easy read, and provides a wide ranging cook's tour around basic computer vision problems, complete with a large assortment of ready-made routines with which to tackle them. It is likely to be of utility to those involved in programming low-level industrial

vision applications and to those who suspect that computer vision might offer solutions to their problems but who have no experience in the field and require a fairly comprehensive and very comprehensible introduction.

SIMON CLIPPINGDALE
University of Warwick

JOSEPH BERGIN
Data Abstraction, the Object-Oriented Approach Using C++. McGraw Hill, 1994, £41.95, 666 pp., hardbound, ISBN 0 07 911691 4

This book aims to teach data abstraction, object-oriented programming and computer science in that order of priority. In these aims I believe the book largely succeeds, with two qualifications described below.

The first part of the book ('Fundamentals') introduces C++ with enough detail so that examples in the book can be understood. (The author makes clear that the book is not intended as a comprehensive primer for C++ and a basic knowledge of C is assumed.) Concepts of data abstraction, and object-oriented (OO) programming are then introduced followed by an introduction to 'high level' classes like 'object', 'magnitude' (characters, strings, fractions etc), 'collection' (sets, bags, lists, etc), and so on. The ideas are then applied to the design and implementation of a deterministic finite state automaton and to the translation of arithmetic expressions.

The second part of the book ('Implementations') gives a relatively detailed account of representational structures including lists, sets, stacks, queues, trees, characters, strings, heaps, graphs, with related concepts such as sorting and recursion.

The last part ('Applications') gives examples in areas such as program translation, recursive descent parsing, program verification and database normalization.

One reservation I have about the book is tht it puts what I feel is an unreasonable emphasis on data and data abstraction at the expense of procedures and the abstraction of procedural code. It is misleading to define software objects as 'data elements' since, in OO languages from Simula onwards, an object comprises either or both of data structures and procedures (methods or functions). Inheritance, which is the key abstraction mechanism in OO languages, applies to procedural code just as much as it does to data structures. And structures like lists and trees can be, and often are, used to represent procedures as well as data (witness Lisp and OO extensions of Lisp). 'Software Abstraction' would provide a more even-handed focus for the book.

I also have a reservation about the way the book concentrates almost exclusively on computer-oriented concepts (lists, arrays, finite-state automata, etc) and neglects what I believe is the great strength of the the OO paradigm: to facilitate the modelling of objects and classes in the world *outside* the computer. The author recognises that 'object-oriented programming ... offers the benefit of making it possible, and easy, to make the running program more nearly model the real situation in which the original problem arose ... every program is a simulation or model of some phenomenon' (p. 46). It is true, as he says, that computer-oriented concepts may be modelled in software. But most applications of computers are about 'real world' things like people, cars and aeroplanes, and about procedures, processes and rules in the world outside the computer.

OO design allows a 1:1 mapping from these kinds of objects (and classes of objects) to corresponding 'objects' (and 'classes') in software. Despite the attractive simplicity of this idea, subtle problems of analysis can arise. In my experience, students need plenty of examples to show how OO modelling can be applied in the design of software to assist the management of things like libraries, shops, clubs, warehouses, elections and so on. It would be good if the 'Applications' section of any future edition of this book could plug this gap.

Leaving these points aside, the book is clearly written and well presented. It is good to have the C++ examples supplied on a disk. For any course which aims to teach representational devices (lists, stacks, arrays, etc) and associated concepts within the object-oriented paradigm, this book will serve very well.

GERRY WOLFF
University of Wales, Bangor

KEVIN LANO and HOWARD HAUGHTON (Eds)
Object-oriented Specification Case Studies. Prentice Hall, 1994, £22.95, 236 pp., softbound, ISBN 0 13 097015 8

This book splits naturally into two main parts. These are discussed here in reverse order. The second part comprises of Chapters 4–10 inclusive and this contains the specification case studies which give the book its name. These are drawn from a number of areas. Included are specifications of the Unix filing system written in MooZ, a mobile phone system in Object-Z, a concept-recognition system—a kind of machine-learning program—in Z^{++}, a variety of bank accounts and a block-structured symbol table in OOZE, various quadrilaterals in VDM++ and a partial specification of a process scheduler in SmallVDM. Also included in this part of the book is an 18-page chapter entitled 'Refinement in Fresco'. Although this contains some small specifications of quadrilaterals, it is predominantly devoted to the topic of the refinement of specifications into Smalltalk and the proof system needed for this. In my opinion, Lano and Haughton should not have included this chapter in their book as it has a very different emphasis and focus from everything else that the book contains. There is a very pronounced emphasis on object-oriented specification languages based on Z in this part of the book. In fact, 70% of it is devoted to discussing case studies written in languages that are extensions of Z, whereas only 17% is devoted to languages derived from