CHAPTER

Epilogue—Perspectives in vision

24

24.1 INTRODUCTION

The preceding chapters have covered many topics relating to vision—how images may be processed to remove noise, how features may be detected, and how objects may be located from their features; they have also given insights into how to set up lighting and how to select rapid hardware systems, e.g., for automated visual inspection. The subject is one that has developed over a period of more than 40 years and has come a long way. However, it has developed piecemeal rather than systematically. Often, development is motivated by the particular interests of small groups of workers and is relatively ad hoc. Coupled with this is the fact that algorithms, processes, and techniques are all limited by the creativity of the various researchers: the process of design tends to be intuitive rather than systematic, and so again some arbitrariness tends to creep in from time to time. As a result, sometimes no means has yet been devised for achieving particular aims, but more usually a number of imperfect methods are available, and there is limited scientific basis for choosing between them.

All this poses the problem of how the subject may be placed on a firmer foundation. Time may help but time can also have the effect of making things more difficult as more methods and results arise which have to be considered; in any case, there is no shortcut to intellectual analysis of the state of the art. This book has aimed to carry out a degree of analysis at every stage; but in this last chapter, it is worth trying to tie it all together, to make some general statements on methodology and to indicate the direction that might be taken in the future.

Computer vision is an engineering discipline and, like all such disciplines, it has to be based on science and understanding of fundamental processes. However, as an engineering discipline, it should involve specification-based design. Once the specifications for a vision system have been laid down, it can be seen how they match up against the constraints provided by nature and

Computer Vision. DOI: http://dx.doi.org/10.1016/B978-0-12-809284-2.00024-1

technology. In what follows, we consider first the parameters of relevance for the specification of vision systems; then we consider constraints and their origins. This leads to some clues as to how the subject could be developed further.

However, there is an important caveat to the way of thinking expressed above: by the end of the chapter, it will become clear that a major disruption has actually been introduced by the advent of a series of impressive, powerful deep learning architectures—thereby warranting a radical rethink of the way computer vision should go in the future.

24.2 PARAMETERS OF IMPORTANCE IN MACHINE VISION

The first thing that can be demanded of any engineering design is that it should work! This applies as much to vision systems as to other parts of engineering. Clearly, there is no use in devising edge detectors that do not find edges, corner detectors that do not find corners, thinning algorithms that do not thin, 3-D object detection schemes that do not find objects, and so on. But in what way could such schemes fail? Even if we ignore the possibility of noise or artifacts preventing algorithms from operating properly, there remains the possibility that at any stage important fundamental factors have not been taken into account.

For example, a boundary tracking algorithm can go wrong because it encounters a part of the boundary that is one pixel wide and crosses over instead of continuing. A thinning algorithm can go wrong because every possible local pattern has not been taken into account in the design and hence it disconnects a skeleton. A 3-D object detection scheme can go wrong because proper checks have not been made to confirm that a set of observed features is not coplanar. Of course, these types of problem may arise very rarely (i.e., only with highly specific types of input data), which is why the design error may not be noticed for a time. Often, mathematics or enumeration of possibilities can help to eliminate such errors, so problems can be removed systematically. However, being absolutely sure no error has been made is difficult—and it must not be forgotten that transcription errors in computer programs can contribute to the problems. These factors mean that algorithms should be put to extensive tests with large data sets in order to ensure that they are correct or at least contain adequate levels of robustness. There is no substitute for subjecting algorithms to variegated tests of this type to check out ideas that are "evidently" correct. This obvious fact is still worth stating, as silly errors continually arise in practice.

At this stage, imagine that we have a range of algorithms that all achieve the same results on ideal data, and that they really work. The next problem is to compare them critically and, in particular, to find how they react to *real* data and the nasty realities such as noise that accompany it. These nasty realities may be summed up as follows:

1. noise
2. background clutter
3. occlusions

4. object defects and breakages
5. optical and perspective distortions
6. nonuniform lighting and its consequences
7. effects of stray light, shadows, and glints.

In general, algorithms need to be sufficiently robust to overcome these problems. However, things are not so simple in practice. For example, HT and many other algorithms are capable of operating properly and detecting objects or features despite considerable degrees of occlusion. But how much occlusion is permissible? Or how much distortion, noise, or how much of any other of the nasty realities can be tolerated? In each specific case we could state some figures that would cover the possibilities. For example, we may be able to state that a line detection algorithm must be able to tolerate 50% occlusion, and so, a particular HT implementation is (or is not) able to achieve this. However, at this stage, we end with a lot of numbers that may mean very little on their own: in particular, they seem different and incompatible. In fact, this latter problem can largely be eliminated: each of the defects can be imagined to obliterate a definite proportion of the object (in the case of impulse noise, this is obvious; with Gaussian noise, the equivalence is not so clear but we suppose here that an equivalence can at least in principle be computed). Hence, we end up by establishing that artifacts in a particular dataset eliminate a certain proportion of the area and perimeter of all objects, or a certain proportion of all small objects. Clearly, certain of the nasty realities (such as optical distortions) tend to act in such a way as to cut down accuracy, but we concentrate here on robustness of object detection. Taking account of all these remarks, we are now in a position to proceed to the next stage of analysis.

To go further, it is necessary to set up a complete specification for the design of a particular vision algorithm. The specification can be listed as follows (but generality is maintained by not stating any particular algorithmic function):

1. the algorithm must work on ideal data
2. the algorithm must work on data that is $x\%$ corrupted by artifacts
3. the algorithm must work to p pixels accuracy
4. the algorithm must operate within s seconds
5. the algorithm must be trainable
6. the algorithm must be implemented with failure rate less than 1 per d days
7. the hardware needed to implement the algorithm must cost less than £L.

(The failure rate referred to in specification 6 can often be taken as arising mainly from hardware problems and will be ignored in what follows.)

The set of specifications listed above may at any stage of technological (especially hardware) development be unachievable; this is because they are phrased in a particular way, so they are not compromisable. However, if a given specification is getting near to its limit of achievability, a switch to an alternative

algorithm might be possible—but note that several, or all, relevant algorithms may be subject to almost identical limitations, because of underlying technological or natural constraints; alternatively, an internal parameter might be adjusted which keeps that specification within range, whereas pushing another specification closer to the limits of its range. In general, there will be some hard (nonnegotiable) specifications and others for which a degree of compromise is acceptable. As has been seen in various chapters of the book, this leads to the possibility of tradeoffs—a topic which is reviewed in the next section.

24.3 TRADEOFFS

Tradeoffs form one of the most important features of algorithms, as they permit a degree of flexibility subject only to what is possible in the nature of things. Ideally, the tradeoffs that are enunciated by theory provide absolute statements about what is possible, so that if an algorithm approaches these limits it is then probably as "good" as it can possibly be.

Next, there is the problem about where on a tradeoff curve, an algorithm should be made to operate. In many cases, the tradeoff curve (or surface) is bounded by hard limits. However, once it has been established that the optimum working point is somewhere within these limits, in a continuum, then it is appropriate to select a criterion function, whereby an optimum can be located uniquely. Details will vary from case to case but the crucial point is that an optimum must exist on a tradeoff curve, and that it can be found systematically once the curve is known. Clearly, all this implies that the science of the situation has been studied sufficiently so that relevant tradeoffs have been determined. We further illustrate this in the following subsections, which may be bypassed on a first reading.

24.3.1 SOME IMPORTANT TRADEOFFS

Earlier chapters of this book have revealed some quite important tradeoffs that are more than just arbitrary relations between relevant parameters. Here, a few examples will have to suffice by way of summary.

First, in Chapter 5, Edge Detection, the DG edge operators were found to have only one underlying design parameter—that of operator radius r. Ignoring here, the important matter of the effect of a discrete lattice in giving preferred values of r, it was found that

1. signal-to-noise ratio varies linearly with r, because of underlying signal and noise averaging effects;
2. resolution varies inversely with r, as relevant linear features in the image are averaged over the active area of the neighborhood: The scale at which edge positions are measured is given by the resolution;

3. the accuracy with which edge position (at the current scale) may be measured depends on the square root of the number of pixels in the neighborhood and hence varies as r;
4. computational load, and associated hardware cost, is typically proportional to the number of pixels in the neighborhood and hence varies as r^2.

Thus, operator radius carries with it four properties which are intimately related—signal-to-noise ratio, resolution (or scale), accuracy, and hardware/computational cost.

Another important problem was that of fast location of circle centers (Chapter 10: Line, Circle, and Ellipse Detection); in this case, robustness was seen to be measurable as the amount of noise or signal distortion that can be tolerated. For HT-based schemes, noise, occlusions, distortions, etc. all reduce the peak height in parameter space, thereby reducing the signal-to-noise ratio and impairing accuracy. Furthermore, if a fraction β of the original signal is removed, leaving a fraction $\gamma = 1 - \beta$, either by such distortions or occlusions or else by deliberate sampling procedures, then the number of independent measurements of the center location drops to a fraction γ of the optimum. This means that the accuracy of estimation of the center location drops to a fraction around $\sqrt{\gamma}$ of the optimum.

What is important is that the effect of sampling is substantially the same as that of signal distortion, so that the more distortion that must be tolerated, the higher α, the fraction of the total signal sampled, has to be. This means that as the level of distortion increases, the capability for withstanding sampling decreases, and therefore the gains in speed achievable from sampling are reduced—that is, for fixed signal-to-noise ratio and accuracy, a definite robustness−speed tradeoff exists. Alternatively, the situation can be viewed as a three-way relation between accuracy, robustness, and speed of processing. This provides an interesting insight into how the edge operator tradeoff considered earlier might be generalized.

To underline the value of studying such tradeoffs, note that any given algorithm will have a particular set of adjustable parameters which are found to control—and hence lead to tradeoffs between—the important quantities such as speed of processing, signal-to-noise ratio, and attainable accuracy already mentioned. Ultimately, such *practically* realizable tradeoffs (i.e., arising from the given algorithm) should be considered against those that may be deduced on purely theoretical grounds. Such considerations would then indicate whether a better algorithm might exist than the one currently being examined.

24.3.2 TRADEOFFS FOR TWO-STAGE TEMPLATE MATCHING

Two-stage template matching has been mentioned a number of times in this book as a means whereby the normally slow and computationally intensive process of template matching may be speeded up. In general, it involves looking for easily

distinguishable subfeatures, so that locating the features that are ultimately being sought involves only the minor problem of eliminating false alarms. The reason this strategy is useful is that the first stage eliminates the bulk of the raw image data, so that only a relatively trivial testing process remains. This latter process can then be made as rigorous as necessary. In contrast, the first "skimming" stage can be relatively crude, the main criterion being that it must not eliminate any of the desired features: false positives are permitted but not false negatives. However, the efficiency of the overall two-stage process is naturally limited by the number of false alarms thrown up by the first stage. (Note that similar principles arise with the boosting techniques described in Section 21.4: see also Fig. 21.5.)

Suppose that the first stage is subject to a threshold h_1 and the second stage to a threshold h_2. If h_1 is set very low, then the process reverts to the normal template matching situation, as the first stage does not eliminate any part of the image. In fact, setting $h_1 = 0$ initially is useful so that h_2 may be adjusted to its normal working value. Then, h_1 can be increased to improve efficiency (reduce overall computation); a natural limit arises when false negatives start to occur—that is, some of the desired features are not being located. Further increases in h_1 now have the effect of cutting down available signal, although speed continues to increase. This clearly gives a tradeoff between signal-to-noise ratio and hence accuracy of location and speed.

In a particular application in which objects were being located by the HT, the numbers of edge points located were reduced as h_1 increased, so accuracy of object location was reduced (Davies, 1988f). A criterion function approach was then used to determine an optimum working condition. A suitable criterion function turned out to be $C = T/A$, where T is the total execution time and A the achievable accuracy. Although this approach gave a useful optimum, the optimum can be improved further if a mix of normal two-stage template matching and random sampling is used. This turns the problem into a 2-D optimization problem with adjustable parameters h_1 and u (the random sampling coefficient, equal to $1/\alpha$). However, in reality these types of problem are even more complex than indicated so far: in general, this is a 3-D optimization problem, the relevant parameters being h_1, h_2, and u, although in fact a good approximation to the global optimum may be obtained by the procedure of adjusting h_2 first, and then optimizing h_1 and u together—or even of adjusting h_1 first, then h_1 and then u (Davies, 1988f). Further details are beyond the scope of the present discussion.

24.4 MOORE'S LAW IN ACTION

It has been indicated once or twice that the constraints and tradeoffs limiting algorithms are sometimes not accidental but rather the result of underlying technological or natural constraints. If so, it is important to determine this in as many

cases as possible; otherwise, workers may spend much time on algorithm development only to find their efforts repeatedly being thwarted. Usually, this is more easily said than done, but it underlines the necessity for scientific analysis of fundamentals.

The well-known law due to Moore (Noyce, 1977) relating to computer hardware states that the number of components that can be incorporated onto a single-integrated circuit increases by a factor of about two per year. Certainly, this was so for the 20 years following 1959, although the rate subsequently decreased somewhat (not enough, however, to prevent the growth from remaining approximately exponential). It is not the purpose of this chapter to speculate on the accuracy of Moore's law. However, it is useful to suppose that computer memory and power will grow by a factor approaching two per year in the foreseeable future. Similarly, computer speeds may also grow at roughly this rate in the foreseeable future. When then of vision?

Unfortunately, many vision processes such as search are inherently NP-complete and hence demand computation that grows exponentially with some internal parameter such as the number of nodes in a match graph. This means that the advance of technology is able to give only a roughly linear improvement in this internal parameter (e.g., something like one extra node in a match graph every 2 years): It is therefore not solving the major search and other problems but only easing them.

NP-completeness apart, we can often take an optimistic view that the relentless advance of computer power described by Moore's Law is leading to an era when conventional PCs will be able to cope with a fair proportion of vision tasks. Certainly, when combined with specially designed algorithms, it should prove possible to implement many of the simpler tasks in this way, leading to a much less strenuous life for the vision systems designer.

24.5 HARDWARE, ALGORITHMS, AND PROCESSES

The previous section raised the hope that improvements in hardware systems will provide the key to the development of impressive vision capabilities. However, it seems likely that breakthroughs in vision algorithms will also be required before this can come about. My belief is that until robots can play with objects and materials in the way that tiny children do they will not be able to build up sufficient information and the necessary databases for handling the complexities of real vision. The real world is too complex for all the rules to be written down overtly: these rules have to be internalized by training each brain individually. In some ways this approach is better, as it is more flexible and adaptable and at the same time more likely to be able to correct for the errors that would arise in direct transference of huge databases or programs. Nor should it be forgotten that it is the underlying processes of vision and intelligence that are important: Hardware

merely provides a means of implementation. If an idea is devised for a hardware solution to a visual problem, it reflects an underlying algorithmic process that either is or is not effective. Once it is known to be effective, then the hardware implementation can be analyzed to confirm its utility. However, we must not segregate algorithms too much from hardware design: In the end, it is necessary to optimize the whole system, which means considering both together. Ideally at least, the underlying processes should be considered first, before a hardware solution is frozen in. Hardware should not be the driving force as there is a danger that some type of hardware implementation (especially one that is temporarily new and promising) will take over and make workers blind to underlying processes. And many readily designed hardware architectures (from serial pipelines to SIMD (single instruction stream, multiple data stream), VLSI (very large scale integration), and ASIC (application specific integrated circuit) to FPGA (field programmable gate array)—and nowadays very frequently, GPU (graphics processing unit) are restricted and embody low-level vision capability rather than high-level functionality. Hardware should not be the tail that wags the vision dog.

24.6 THE IMPORTANCE OF CHOICE OF REPRESENTATION

This book has progressed steadily from low-level ideas, through intermediate-level methods to high-level processing, covering 3-D image analysis, the necessary technology, etc.—admittedly with its own type of detailed examples and emphasis. Many ideas have been covered and many strategies described. But where have we got to, and to what extent have we solved the problems of vision referred to in Chapter 1, Vision, The Challenge?

Among the worst of all the problems of vision is that of minimizing the amount of processing required to achieve particular image recognition and measurement tasks. Not only do images contain huge amounts of data but often they also need to be interpreted in frighteningly small amounts of time, and the underlying search and other tasks tend to be subject to combinatorial explosions. Yet, in retrospect, we seem to have remarkably few *general* tools for coping with these problems. Indeed, the truly general tools available—ignoring high-level processing methods such as AI (artificial intelligence) tree-search—appear to be

1. reducing high-dimensional problems to lower dimensional problems that can be solved in turn;
2. the Hough transform and other indexing techniques;
3. location of features that are in some sense sparse, and which can hence help to reduce redundancy quickly (obvious examples of such features are edges and corners);
4. two-stage and multistage template matching; and
5. random sampling.

These are said to be general tools as they appear in one guise or another in a number of situations, with totally different data. However, it is pertinent to ask to what extent these are genuine tools rather than almost accidental means (or tricks) by which computation may be reduced. Further analysis yields interesting answers to this question, as will now be seen.

First, consider the Hough transform, which takes a variety of forms—the normal parametrization of a line in an abstract parameter space, the GHT which is parametrized in a space congruent to image space, the adaptive thresholding transform (Chapter 4: The Role of Thresholding) which is parametrized in an abstract 2-D parameter space, and so on. What is common about these forms is *the choice of a representation in which the data peak naturally at various points*, so that analysis can proceed with improved efficiency. The relation with item 3 above now becomes clear, making it less likely that either of these procedures is purely accidental in nature.

Next, item 1 appears in many guises—see e.g., the approaches used to locate ellipses (Chapter 10: Line, Circle, and Ellipse Detection). Thus, item 1 has much in common with item 4. Note also that item 5 can be considered a special case of item 4 (random sampling is a form of two-stage template matching with a "null" first stage, capable of eliminating large numbers of input patterns with particularly high efficiency: see Davies, 1988f). Finally, note that the example of so-called two-stage template matching covered in Section 24.3.2 was actually part of a larger problem which was really multistage: the edge detector was two-stage, but this was incorporated in an HT which was itself two-stage, making the whole problem at least four-stage. It can now be seen that items 1−5 are all forms of multistage matching (or sequential pattern recognition) which are potentially more powerful and efficient than a single-stage approach. Similar conclusions are arrived at in Appendix A, which deals with robust statistics and their application to machine vision.

The above discussion clearly raises the question of how complex tasks are to be broken down into the most appropriate multistage processes, and equivalently what the most suitable representation has to be for sparse feature location. At the same time, when looking at representations for vision algorithms, we need to be aware that *all representations impose their own order on a system*: for a time, this may be a good imposition, but in the end, it may turn into a dire restriction that is past its sell-by date. (This is what happened to the old chain code representation for boundary coding and also what happened to the centroidal profile approach to shape analysis.)

24.7 PAST, PRESENT, AND FUTURE

In some sense, the contents of a book such as this have to be concentrated on subject matter that is definite: Indeed, it is the duty of an author to provide

information on the definite rather than the ephemeral, so there has to be some concentration on the past. Yet, a book must also concentrate on fundamental principles, and these necessarily continue from the past to the present and the future. The difference is that principles that will only become known in the future cannot possibly be included, and here, a sound framework together with the current difficulties and unsolved problems can at least provide readers with a readiness for any principles that are to come. In fact, this book has solved some of the problems it set itself—starting with low-level processing, concentrating on strategies, limitations, and optimizations of intermediate-level processing, going some way with higher level tasks, and attempting to create an awareness of the underlying processes of vision. At the same time, there are many interesting current developments that will prove even more interesting in the future. For the subject has passed the stage of overconcentration on hardware and absolute efficiency and has focused on the important need to extend effectiveness and capability. In addition, the developments of the past decade or so have taken the subject out of the era of the ad hoc into that of mathematical precision and probabilistic formulation, so that whatever vision is expected to achieve is written down in terms of estimators that are mathematically defined and turned into rigorous implementations. Nowhere is this clearer that for the new invariant feature detectors with their massive descriptors that arguably make 3-D interpretation and motion tracking almost trivial to implement. All this means that exotic yet direly needed applications such as vision-based driver assistance systems are able to come into being—and it is possible to predict that they will be with us in the cars of the immediate future, if only we and the legal system will allow this.

Only a fool would make rash predictions (and many predictions within AI have remained elusive for more than 40 years), but it is different if the principles are clear: and they are evident to many vision workers nowadays; in fact, there is an air of euphoria over the rapidly growing maturity of the latest vision algorithms and the capability of the newest computers to implement them, so the very momentum is starting to make it straightforward to estimate when various developments will happen—a situation that is advancing all types of video analytics, in areas ranging from transport to crime detection and prevention, not to mention face recognition, biometrics, and robotics. It is hoped that the present volume will be able to communicate some of the excitement underlying these present and future developments and also some means for understanding their basis.

24.8 THE DEEP LEARNING EXPLOSION

Earlier sections of this chapter have dwelt strongly on a conventional view of the development path of computer vision, in which creativity, design, and scientific optimization need to go hand in hand. However, in 2011−2012, deep learning exploded onto the computer vision scene with staggering performance levels,

showing just how much further the subject could be pressed to go if only we gave up thinking of pure scientific analysis for a while. Effectively, this stimulus brought with it a cogent existence theorem showing things that could not be ignored—however, much we as scientists might have misgivings about them (because we don't know what is *really* going on in a neural system that is learning for itself rather than being guided by a "proper" algorithm). Nevertheless, it is well known that science advances in phases—first a practical advance, then a theoretical one, then a practical one, then a phenomenological model, and so on. Just because we have not reached an ideal theoretical stage at this point in time is not necessarily a bad thing: The required theory will emerge in very few years when its time has come. In fact, what is needed is a substantial amount more experimental data from a variety of application areas, so that we will be in a position to generalize over what is possible and arrive at sound scientific conclusions about the real capabilities and proper roles of deep networks in computer vision.

Although some of the concentration of this chapter has been on tradeoffs and optimization, deeper issues are involved, such as finding out how to make valid specifications for image data, what representations are needed within vision algorithms and how the latter break down the overall process into viable subprocesses. There are also questions about the way in which vision algorithms are set up to rigorously estimate key parameters—a factor that relates directly to reliability, robustness, and fitness for purpose. Added to this are the exciting new applications of this rapidly maturing subject.

However, the new deep learning networks seem to change all this. They are now extremely impressive at the performance level. The ultimate question is whether they can be made to adequately embody the scientific approach that is necessary to allow us to be confident that their internal workings are completely reliable, and indeed that these hidden workings do not prevent us from getting a rigorous enough view of how any ***overall*** *vision algorithm should ideally be constructed.*

24.9 BIBLIOGRAPHICAL AND HISTORICAL NOTES

Much of this chapter has summarized the work of earlier chapters and attempted to give it some perspective. In particular, two-stage template matching has been highlighted in the current chapter: the earliest work on this topic was carried out by Rosenfeld and VanderBrug (1977) and VanderBrug and Rosenfeld (1977), whereas the ideas of Section 24.3.2 were developed by Davies (1988f). Two-stage template matching harks back to the spatial matched filtering concept discussed in Chapter 11, The Generalized Hough Transform and elsewhere. Ultimately, this concept is limited by the variability of the objects to be detected. However, it has been shown that some account can be taken of this problem, e.g., in the design of filter masks (see Davies, 1992d). It ought also to be stated that this topic is highly formative, and though it is here developed in the context of

template matching, it is possible to see shadows and reflections of it right through the whole subject: one has only to ask how any new algorithm breaks down visual analysis into an efficient set of subprocesses and what representations they are operating in, in order to see the ramifications of this concept.