The other piece of work was Horn's (1975) analysis of shape from shading, which was the first in what was to become a distinguished series of articles on the formation of images. By carefully analyzing the way in which the illumination, surface geometry, surface reflectance, and viewpoint conspired to create the measured intensity values in an image, Horn formulated a differential equation that related the image intensity values to the surface geometry. If the surface reflectance and illumination are known, one can solve for the surface geometry (see also Horn, 1977). Thus from shading one can derive shape.

The message was plain. There must exist an additional level of understanding at which the character of the information-processing tasks carried out during perception are analyzed and understood in a way that is independent of the particular mechanisms and structures that implement them in our heads. This was what was missing—the analysis of the problem as an information-processing task. Such analysis does not usurp an understanding at the other levels—of neurons or of computer programs—but it is a necessary complement to them, since without it there can be no real understanding of the function of all those neurons.

This realization was arrived at independently and formulated together by Tomaso Poggio in Tübingen and myself (Marr and Poggio, 1977; Marr, 1977b). It was not even quite new—Leon D. Harmon was saying something similar at about the same time, and others had paid lip service to a similar distinction. But the important point is that if the notion of different types of understanding is taken very seriously, it allows the study of the information-processing basis of perception to be made *rigorous*. It becomes possible, by separating explanations into different levels, to make explicit statements about what is being computed and why and to construct theories stating that what is being computed is optimal in some sense or is guaranteed to function correctly. The ad hoc element is removed, and heuristic computer programs are replaced by solid foundations on which a real subject can be built. This realization—the formulation of what was missing, together with a clear idea of how to supply it—formed the basic foundation for a new integrated approach, which it is the purpose of this book to describe.

1.2 UNDERSTANDING COMPLEX INFORMATION-PROCESSING SYSTEMS

Almost never can a complex system of any kind be understood as a simple extrapolation from the properties of its elementary components. Consider, for example, some gas in a bottle. A description of thermodynamic effects—

temperature, pressure, density, and the relationships among these factors—is not formulated by using a large set of equations, one for each of the particles involved. Such effects are described at their own level, that of an enormous collection of particles; the effort is to show that in principle the microscopic and macroscopic descriptions are consistent with one another. If one hopes to achieve a full understanding of a system as complicated as a nervous system, a developing embryo, a set of metabolic pathways, a bottle of gas, or even a large computer program, then one must be prepared to contemplate different kinds of explanation at different levels of description that are linked, at least in principle, into a cohesive whole, even if linking the levels in complete detail is impractical. For the specific case of a system that solves an information-processing problem, there are in addition the twin strands of process and representation, and both these ideas need some discussion.

Representation and Description

A *representation* is a formal system for making explicit certain entities or types of information, together with a specification of how the system does this. And I shall call the result of using a representation to describe a given entity a *description* of the entity in that representation (Marr and Nishihara, 1978).

For example, the Arabic, Roman, and binary numeral systems are all formal systems for representing numbers. The Arabic representation consists of a string of symbols drawn from the set (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), and the rule for constructing the description of a particular integer n is that one decomposes n into a sum of multiples of powers of 10 and unites these multiples into a string with the largest powers on the left and the smallest on the right. Thus, thirty-seven equals $3 \times 10^1 + 7 \times 10^0$, which becomes 37, the Arabic numeral system's description of the number. What this description makes explicit is the number's decomposition into powers of 10. The binary numeral system's description of the number thirty-seven is 100101, and this description makes explicit the number's decomposition into powers of 2. In the Roman numeral system, thirty-seven is represented as XXXVII.

This definition of a representation is quite general. For example, a representation for shape would be a formal scheme for describing some aspects of shape, together with rules that specify how the scheme is applied to any particular shape. A musical score provides a way of representing a symphony; the alphabet allows the construction of a written representation

of words; and so forth. The phrase "formal scheme" is critical to the definition, but the reader should not be frightened by it. The reason is simply that we are dealing with information-processing machines, and the way such machines work is by using symbols to stand for things—to represent things, in our terminology. To say that something is a formal scheme means only that it is a set of symbols with rules for putting them together—no more and no less.

A representation, therefore, is not a foreign idea at all—we all use representations all the time. However, the notion that one can capture some aspect of reality by making a description of it using a symbol and that to do so can be useful seems to me a fascinating and powerful idea. But even the simple examples we have discussed introduce some rather general and important issues that arise whenever one chooses to use one particular representation. For example, if one chooses the Arabic numeral representation, it is easy to discover whether a number is a power of 10 but difficult to discover whether it is a power of 2. If one chooses the binary representation, the situation is reversed. Thus, there is a trade-off; any particular representation makes certain information explicit at the expense of information that is pushed into the background and may be quite hard to recover.

This issue is important, because how information is represented can greatly affect how easy it is to do different things with it. This is evident even from our numbers example: It is easy to add, to subtract, and even to multiply if the Arabic or binary representations are used, but it is not at all easy to do these things—especially multiplication—with Roman numerals. This is a key reason why the Roman culture failed to develop mathematics in the way the earlier Arabic cultures had.

An analogous problem faces computer engineers today. Electronic technology is much more suited to a binary number system than to the conventional base 10 system, yet humans supply their data and require the results in base 10. The design decision facing the engineer, therefore, is, Should one pay the cost of conversion into base 2, carry out the arithmetic in a binary representation, and then convert back into decimal numbers on output; or should one sacrifice efficiency of circuitry to carry out operations directly in a decimal representation? On the whole, business computers and pocket calculators take the second approach, and general purpose computers take the first. But even though one is not restricted to using just one representation system for a given type of information, the choice of which to use is important and cannot be taken lightly. It determines what information is made explicit and hence what is pushed further into the background, and it has a far-reaching effect on the ease and

difficulty with which operations may subsequently be carried out on that information.

Process

The term *process* is very broad. For example, addition is a process, and so is taking a Fourier transform. But so is making a cup of tea, or going shopping. For the purposes of this book, I want to restrict our attention to the meanings associated with machines that are carrying out information-processing tasks. So let us examine in depth the notions behind one simple such device, a cash register at the checkout counter of a supermarket.

There are several levels at which one needs to understand such a device, and it is perhaps most useful to think in terms of three of them. The most abstract is the level of *what* the device does and *why*. What it does is arithmetic, so our first task is to master the theory of addition. Addition is a mapping, usually denoted by $+$, from pairs of numbers into single numbers; for example, $+$ maps the pair $(3, 4)$ to 7, and I shall write this in the form $(3 + 4) \rightarrow 7$. Addition has a number of abstract properties, however. It is commutative: both $(3 + 4)$ and $(4 + 3)$ are equal to 7; and associative: the sum of $3 + (4 + 5)$ is the same as the sum of $(3 + 4) + 5$. Then there is the unique distinguished element, zero, the adding of which has no effect: $(4 + 0) \rightarrow 4$. Also, for every number there is a unique "inverse," written (-4) in the case of 4, which when added to the number gives zero: $[4 + (-4)] \rightarrow 0$.

Notice that these properties are part of the fundamental *theory* of addition. They are true no matter how the numbers are written—whether in binary, Arabic, or Roman representation—and no matter how the addition is executed. Thus part of this first level is something that might be characterized as *what* is being computed.

The other half of this level of explanation has to do with the question of *why* the cash register performs addition and not, for instance, multiplication when combining the prices of the purchased items to arrive at a final bill. The reason is that the rules we intuitively feel to be appropriate for combining the individual prices in fact define the mathematical operation of addition. These can be formulated as *constraints* in the following way:

1. If you buy nothing, it should cost you nothing; and buying nothing and something should cost the same as buying just the something. (The rules for zero.)

2. The order in which goods are presented to the cashier should not affect the total. (Commutativity.)

3. Arranging the goods into two piles and paying for each pile separately should not affect the total amount you pay. (Associativity; the basic operation for combining prices.)

4. If you buy an item and then return it for a refund, your total expenditure should be zero. (Inverses.)

It is a mathematical theorem that these conditions define the operation of addition, which is therefore the appropriate computation to use.

This whole argument is what I call the *computational theory* of the cash register. Its important features are (1) that it contains separate arguments about what is computed and why and (2) that the resulting operation is defined uniquely by the constraints it has to satisfy. In the theory of visual processes, the underlying task is to reliably derive properties of the world from images of it; the business of isolating constraints that are both powerful enough to allow a process to be defined and generally true of the world is a central theme of our inquiry.

In order that a process shall actually run, however, one has to realize it in some way and therefore choose a representation for the entities that the process manipulates. The second level of the analysis of a process, therefore, involves choosing two things: (1) a *representation* for the input and for the output of the process and (2) an *algorithm* by which the transformation may actually be accomplished. For addition, of course, the input and output representations can both be the same, because they both consist of numbers. However this is not true in general. In the case of a Fourier transform, for example, the input representation may be the time domain, and the output, the frequency domain. If the first of our levels specifies what and why, this second level specifies *how*. For addition, we might choose Arabic numerals for the representations, and for the algorithm we could follow the usual rules about adding the least significant digits first and "carrying" if the sum exceeds 9. Cash registers, whether mechanical or electronic, usually use this type of representation and algorithm.

There are three important points here. First, there is usually a wide choice of representation. Second, the choice of algorithm often depends rather critically on the particular representation that is employed. And third, even for a given fixed representation, there are often several possible algorithms for carrying out the same process. Which one is chosen will usually depend on any particularly desirable or undesirable characteristics that the algorithms may have; for example, one algorithm may be much

more efficient than another, or another may be slightly less efficient but more robust (that is, less sensitive to slight inaccuracies in the data on which it must run). Or again, one algorithm may be parallel, and another, serial. The choice, then, may depend on the type of hardware or machinery in which the algorithm is to be embodied physically.

This brings us to the third level, that of the device in which the process is to be realized physically. The important point here is that, once again, the same algorithm may be implemented in quite different technologies. The child who methodically adds two numbers from right to left, carrying a digit when necessary, may be using the same algorithm that is implemented by the wires and transistors of the cash register in the neighborhood supermarket, but the physical realization of the algorithm is quite different in these two cases. Another example: Many people have written computer programs to play tic-tac-toe, and there is a more or less standard algorithm that cannot lose. This algorithm has in fact been implemented by W. D. Hillis and B. Silverman in a quite different technology, in a computer made out of Tinkertoys, a children's wooden building set. The whole monstrously ungainly engine, which actually works, currently resides in a museum at the University of Missouri in St. Louis.

Some styles of algorithm will suit some physical substrates better than others. For example, in conventional digital computers, the number of connections is comparable to the number of gates, while in a brain, the number of connections is much larger ($\times 10^4$) than the number of nerve cells. The underlying reason is that wires are rather cheap in biological architecture, because they can grow individually and in three dimensions. In conventional technology, wire laying is more or less restricted to two dimensions, which quite severely restricts the scope for using parallel techniques and algorithms; the same operations are often better carried out serially.

The Three Levels

We can summarize our discussion in something like the manner shown in Figure 1–4, which illustrates the different levels at which an information-processing device must be understood before one can be said to have understood it completely. At one extreme, the top level, is the abstract computational theory of the device, in which the performance of the device is characterized as a mapping from one kind of information to another, the abstract properties of this mapping are defined precisely, and its appropriateness and adequacy for the task at hand are demonstrated. In the center is the choice of representation for the input and output and the

Computational theory	Representation and algorithm	Hardware implementation
What is the goal of the computation, why is it appropriate, and what is the logic of the strategy by which it can be carried out?	How can this computational theory be implemented? In particular, what is the representation for the input and output, and what is the algorithm for the transformation?	How can the representation and algorithm be realized physically?

Figure 1–4. The three levels at which any machine carrying out an information-processing task must be understood.

algorithm to be used to transform one into the other. And at the other extreme are the details of how the algorithm and representation are realized physically—the detailed computer architecture, so to speak. These three levels are coupled, but only loosely. The choice of an algorithm is influenced for example, by what it has to do and by the hardware in which it must run. But there is a wide choice available at each level, and the explication of each level involves issues that are rather independent of the other two.

Each of the three levels of description will have its place in the eventual understanding of perceptual information processing, and of course they are logically and causally related. But an important point to note is that since the three levels are only rather loosely related, some phenomena may be explained at only one or two of them. This means, for example, that a correct explanation of some psychophysical observation must be formulated at the appropriate level. In attempts to relate psychophysical problems to physiology, too often there is confusion about the level at which problems should be addressed. For instance, some are related mainly to the physical mechanisms of vision—such as afterimages (for example, the one you see after staring at a light bulb) or such as the fact that any color can be matched by a suitable mixture of the three primaries (a consequence principally of the fact that we humans have three types of cones). On the other hand, the ambiguity of the Necker cube (Figure 1–5) seems to demand a different kind of explanation. To be sure, part of the explanation of its perceptual reversal must have to do with a bistable neural network (that is, one with two distinct stable states) somewhere inside the

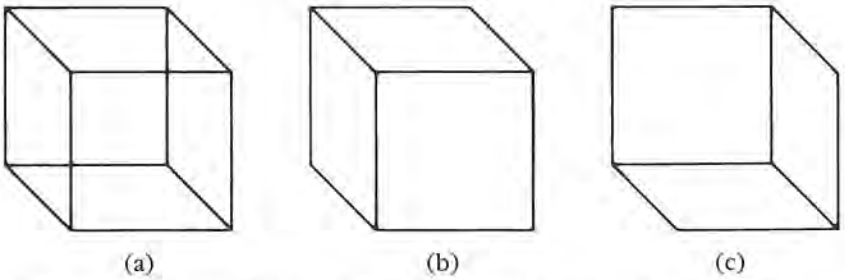

Figure 1–5. The so-called Necker illusion, named after L. A. Necker, the Swiss naturalist who developed it in 1832. The essence of the matter is that the two-dimensional representation (a) has collapsed the depth out of a cube and that a certain aspect of human vision is to recover this missing third dimension. The depth of the cube can indeed be perceived, but two interpretations are possible, (b) and (c). A person's perception characteristically flips from one to the other.

brain, but few would feel satisfied by an account that failed to mention the existence of two different but perfectly plausible three-dimensional interpretations of this two-dimensional image.

For some phenomena, the type of explanation required is fairly obvious. Neuroanatomy, for example, is clearly tied principally to the third level, the physical realization of the computation. The same holds for synaptic mechanisms, action potentials, inhibitory interactions, and so forth. Neurophysiology, too, is related mostly to this level, but it can also help us to understand the type of representations being used, particularly if one accepts something along the lines of Barlow's views that I quoted earlier. But one has to exercise extreme caution in making inferences from neurophysiological findings about the algorithms and representations being used, particularly until one has a clear idea about what information needs to be represented and what processes need to be implemented.

Psychophysics, on the other hand, is related more directly to the level of algorithm and representation. Different algorithms tend to fail in radically different ways as they are pushed to the limits of their performance or are deprived of critical information. As we shall see, primarily psychophysical evidence proved to Poggio and myself that our first stereo-matching algorithm (Marr and Poggio, 1976) was not the one that is used by the brain, and the best evidence that our second algorithm (Marr and Poggio, 1979) *is* roughly the one that is used also comes from psychophysics. Of course, the underlying computational theory remained the same in both cases, only the algorithms were different.