Reverse engineering the world

A commentary on Hoffman, Singh and Prakash, "The interface theory of perception"

Chris Fields

528 Zinnia Court

Sonoma, CA 95476 USA

fieldsres@gmail.com

707-980-5091

Abstract

Does perception hide the truth? Information theory, computer science and quantum theory all suggest that the answer is "yes." They suggest, indeed, that useful perception is only feasible because the truth can be hidden.

Introduction

Hoffman, Singh and Prakash (hereafter "HSK") show in their target article that multi-generational competition for environmental resources will inevitably favor perceptual strategies that preferentially detect features of such resources that correlate positively with reproductive fitness. They show, moreover, that such competition will drive perceptual strategies that preferentially detect features of

environmental resources that do not correlate positively with reproductive fitness to extinction. They conclude from these results that successful perceptual strategies – indeed, all strategies likely to be implemented by extant organisms – will be "interface strategies" that present the organisms deploying them with species- or even individual-specific fitness-relevant affordances of their environments, not with "objective" observer-independent "facts" about their environments. They insist, indeed, that such interfaces will in general *hide* the "objective" truth (Abstract, italics in original) in the same way that the user interface of a laptop computer hides the truth about the laptop's inner structure and operations.

I will argue in this commentary that, despite its *prima facie* conflict with common thinking about the relationship between perception and truth, we should not be surprised by HSK's conclusion. We should instead regard HSK's careful computational experiments as driving one more nail into the coffin of a view of reality as perceptually self-evident that has been challenged by philosophers at least since Heraclitus in the West and Siddhārtha Gautama in the East (both ca. 500 BCE) and that was, in the 20th century, progressively deconstructed by special and general relativity, quantum theory, classical information theory and computer science. By formulating both their experiments and their conclusion in evolutionary terms, HSK provide us with an opportunity to re-examine evolution, and in particular the evolution of perceptual mechanisms and their coupling to action-generating mechanisms, in light of these other sciences. They motivate us to ask what *kinds* of information perceptual systems hide, both in our own case and in the cases of other organisms. They raise the possibility of a comparative psychology of interfaces that may reveal deep insights into the dense web of relationships between thermodynamics and information theory, organismal and evolutionary biology, multi-scale ecology and the cognitive sciences.

The next two sections briefly review some relevant results in information theory and computer science,

respectively. The fourth section proposes a general hypothesis suggested by these results: that the perceptual interfaces of "higher" organisms can be expected to hide *more* information about the world than those of "lower" organisms. A final coda comments briefly on the consonance between the interface theory as proposed by HSK and contemporary quantum theory.

"System identification" problems

World War II gave practical urgency to two previously-obscure "reverse engineering" problems: 1) given a signal of unknown provenance, how does one determine its source? and 2) given a device of unknown manufacture, how does one determine its intended function or behavior? An important wartime constraint on solutions to these problems was that they must be non-destructive; in the case of the second problem, this constraint sometimes took the form of probing the system without causing it to blow up in one's face. As any engineer could perform only a finite number of manipulations and measurements and could record their outcomes only at finite resolution, either of these problems can be represented by the abstract problem of determining, by a finite number of manipulations and observations, both the complete set of states and the complete set of allowed state transitions of an abstract finite-state machine (FSM). The abstract problem of fully characterizing an FSM with a finite sequence of non-destructive operations is the classical "system identification" problem. Ashby (1956), Moore (1956) and others proved that the classical system identification problem is unsolvable: while finite sequences of operations can establish *lower* limits on the number of degrees of freedom and hence the potential behavioral complexity of a device, no finite sequence of operations can establish an *upper* limit on the number of degrees of freedom or the potential behavioral complexity of a device. The very next manipulation of any device may result in completely unexpected behavior, regardless of one's previous experience of manipulating it.

All organisms are, at all times, clearly in the position of a reverse engineer: all organisms face a local environment of unknown complexity, about which they can store in memory at most a finite amount of information. The very next interaction of any organism with its environment may result in an arbitrarily-large surprise. This predicament is rendered more severe by a time-varying environment, and more severe still by competing organisms. Any organism's limited memory therefore contains, at best, only an approximate model of its environment, one that may be proven grossly inadequate at any moment.

The question of interest for HSK is whether such models are not merely incomplete in principle but rather actually wrong, in a particular, precise sense, across the board. As HSK suggest in the discussion of their “Invention of symmetry” theorem, answering this question requires looking at how organisms *encode* their models of their environments. Clearly such encodings must be finite. They can, moreover, only comprise tokens, whether procedural or declarative, that correspond either to features of the environment that the organism can detect, or to combinations of other tokens that correspond to such detectable features. The question then becomes: given such a token, what, in the environment, does it refer to? This is a slightly generalized form of the “symbol grounding” problem introduced by Harnad (1990). As shown in Fields (2014), the symbol grounding problem is equivalent to the system identification problem. An organism cannot, therefore, determine by finite observations what the tokens comprising its own model of the environment refer to, and cannot determine what the tokens comprising any other organism's model of the environment refer to either. The best an organism can do is to construct an abstracted, meta-level model of its own or another's model of the environment; expectations about another organism's behavior provide an example. One might expect models at either level to be at least approximately optimized to represent affordances encountered in the recent past. HSK's experiments confirm this expectation brilliantly.

Virtual machines

HSK employ the relationship between a computer's user interface and what is “really” going on inside the computer as a continuing and apt metaphor. It is useful, however, to examine this metaphor closely, as it rests on what might be regarded as the founding insight of computer science: that any algorithm can be instantiated by any of an arbitrarily large set of distinct physical systems. It is this insight that allows us to regard laptops manufactured by different suppliers, or laptops and mainframes, or even laptops and abstract Turing machines as instantiating the same algorithms. This insight has, moreover, a converse that is also an insight: any sequence of observed physical state transitions executed by any physical device can be interpreted as a sample of an execution trace of an arbitrarily large set of algorithms. These two insights come together in the notion of a *virtual machine* (e.g. Goldberg, 1974; Tanenbaum, 1976). A virtual machine is a well-defined semantic interpretation of the behavior of a device as an execution trace of an algorithm. The “device” in question can either be a physical, “hardware” device or another virtual machine. Computers are useful because their physical dynamics as “hardware” support hierarchies of such semantic interpretations: assembly languages, compilers, operating systems, application programming languages, programs written in such languages, and the user interfaces of such programs are all virtual machines. When a user clicks a mouse over a colored box on a computer's screen, the hardware tumbles through a specific sequence of hundreds to hundreds of thousands of state transitions. No one, not even the hardware designers, needs to understand which sequences of state transitions follow which user actions. Indeed, no one has to understand any more than the behavior of the virtual machine that they or the program they are writing manipulates. This is the benefit of a *hierarchy* of virtual machines: each virtual machine only needs to explicitly represent the structure and behavior of the virtual machine immediately below it in the hierarchy. A computer's end users only have to understand the user interface; the virtual machine hierarchy below it handles all

of the rest.

The “user interface metaphor” is, therefore, much more than a metaphor. If we regard our perceived “world” as a virtual machine, computer science tells us that arbitrarily many distinct physical implementations of that world are possible. The causal structures of these implementations are not, in general, homologous to that of the virtual-machine world; the “virtual causation” that generates state transitions in the virtual-machine world is just a *semantic interpretation* of the causal structure of the implementation, one of arbitrarily many that are possible. Computer science also reminds us that user interfaces handle input *and* output; our actions on the world are every bit as virtual as our perceptions. Whatever we are doing is having *some* effect on the underlying “hardware” of the world. Indeed we can only assume that “we” and all of our actions are implemented by that same hardware. But we cannot discover anything more than a lower limit on the complexity of the hardware by manipulating the virtual machine.

It is worth emphasizing that these kinds of statements are utterly non-controversial in a computer science context. Read through any program written in java, C++, prolog or even FORTRAN; there are no statements referring to registers, logic gates, transistors, voltage levels, leakage currents or silicon. It is, moreover, *because* programming languages do not specify what the hardware is doing that they are useful; if programming required directly manipulating logic gates, for example, none of the application software with which we are all familiar would be even remotely feasible. In HSK's somewhat unfelicitous terminology, “hiding the truth” is what programming languages are all about. A programming language that hides the truth – not just the truth about the hardware, but also the truth about all but the immediately underlying virtual machine – much better than its competitors is honored with the label “next generation.”

What is evolution hiding?

HSK's paper, on the other hand, *is* controversial. It claims that biological evolution hides the truth from living organisms, and it presents impressive computational experiments to back up that claim. This just *feels wrong*. Surely we know something about the way the world works. We found the Higgs boson, after all.

This is a case where, it seems to me, a computer science perspective really is useful. A “high-level” programming language like java hides the truth very effectively – a java programmer can be completely ignorant of the hardware, the display and memory managers, the device drivers, the operating system, and every other component or property of the end-user's computer system except its java compiler and still design and build an extraordinarily useful piece of software. High-level languages like java are not *less* useful because they hide all of this information, they are *more* useful. How can they hide the truth and still be more useful? They can do this because they abstract out useful *and complex* behaviors of the underlying hardware. Providing `sin(x)' as a predefined function in a programming language, much less providing something like `print' that has to deal with dozens of manufacturer-specific device drivers, requires not just a huge abstraction of what the hardware is doing, but a huge abstraction that is also *general*: it works not just for one physical system, but for many physical systems that could, in principle, share no structural components or causal processes whatsoever. A successful programming language does not just hide the truth, it abstracts, generalizes, summarizes and then relabels the truth in a way that increases functionality and minimizes effort.

And that, according to HSK, is precisely what evolution does. Evolution optimizes fitness, and fitness is just another word for efficient functionality. This is a bold statement, and it suggests a bold

hypothesis: we should expect “higher” organisms, like “high-level” programming languages, to encode *less* of the truth about the “hardware” of the world, and to do so in a way that is *more useful* than the ways that “lower” organisms do it. This sounds paradoxical, but it is not: we are surrounded by, and our culture and economy are increasingly driven by, devices that implement exactly this principle. We should, moreover, expect organisms to be organized hierarchically as information processors, with virtual machines that “know more” about the hardware of the world closer to the bottom of the hierarchy and virtual machines that “know less” about the hardware of the world closer to the top. We should expect cellular metabolism, for example, to encode lower-level information about nutritional chemistry than organismal metabolism. We should expect metabolism to encode lower-level information about the physical structure of the world than cognition. And we should expect the limbic system to encode lower-level information about the affordances of the world than the cortex. These are all testable predictions, and they are all at least *prima facie* plausible.

Coda: Quantum theory

Psychology is generally viewed as distant from physics. It is, however, interesting to note the rumblings of a sea change in physics that parallels the sea change that HSK propose for psychology. Nearly everyone has heard of Schrödinger's famous cat and the idea that quantum states “collapse” when measured. This idea is driven by a particular view of what is happening when an observer interacts with a physical system: before the interaction, the system is in a “superposed” state, and after the interaction it is in a different, “collapsed” state. The observer, moreover, *knows* that this state change has objectively occurred. A growing movement within physics, however, rejects this idea of collapse, claiming instead that the *only* thing the observer knows is a number, the outcome value that was obtained by doing the experiment (see, for example, Rovelli, 1996; Clifton, Bub & Halvorson, 2003; Schlosshauer, 2006; Fuchs, 2010; Fields, 2012). The observer, in other words, does something or

another to the world, and receives a number. What the world does in response to the observer's actions is unknown, and unknowable. The numbers that result for these unknowable responses of the world to our actions are, however, extraordinarily useful.

References

Ashby, W.R. (1956). *An Introduction to Cybernetics*. London: Chapman & Hall.

Clifton, R., Bub, J. & Halvorson, H. (2003). Characterizing quantum theory in terms of information-theoretic constraints. *Foundations of Physics* 33, 1561-1591.

Fields, C. (2012). If physics is an information theory, what is an observer? *Information* 3, 92-123.

Fields, C. (2014). Equivalence of the symbol grounding and quantum system identification problems. *Information* 5, 172-189.

Fuchs, C. (2010). QBism: The perimeter of quantum Bayesianism. Preprint arxiv:1003.5209v1 [quant-ph].

Goldberg, R. P. (1974). A survey of virtual machine research. *IEEE Computer* 7(6), 34-45.

Harnad, S. (1990). The symbol grounding problem. *Physica D* 42, 335-346.

Moore, E.F. (1956). Gedanken-experiments on sequential machines. In: Shannon, C.W. and McCarthy, J. (Eds.) *Automata Studies*. Princeton, NJ: Princeton University Press, pp. 129-155.

Rovelli, C. (1996). Relational quantum mechanics. *International Journal of Theoretical Physics* 35, 1637-1678.

Schlosshauer, M. (2006). Experimental motivation and empirical consistency of minimal no- collapse quantum mechanics. *Annals of Physics* 321, 112-149 .

Tanenbaum, A. S. (1976). *Structured Computer Organization*. Upper Saddle River, NJ: Prentice Hall.