The robotics path to AGI using SERVO STACKS

J. Storrs Hall
Institute for Molecular Manufacturing
Laporte, PA, USA

Abstract

The case is made that the path to AGI through cognitive and developmental robotics is compelling. Beyond the familiar argument that it keeps researchers honest by forcing their systems to cope with the real world, it encourages them to recapitulate the evolutionary developmental path which gave rise to intelligence in humans. Insights from this perspective are embodied in the SERVO STACKS cognitive architecture with several salient features. The brain evolved as a body controller and thus is based largely on computational structures appropriate to physical process control. Evolution typically copies existing structure and modifies it minimally to meet a new demand. We should therefore expect the higher cognitive functions to be performed by body controllers pressed into service as brain controllers.

Introduction and Motivation

The brain evolved as a body controller; except for size, the human brain is structurally similar to that of other primates. Neocortex in surprisingly homogeneous, suggesting a general computing fabric instead of hard-wired functional modules. This uniformity has been cited as a strong argument for a common computational function (Mountcastle 1978). An argument can thus be made that evolution took a computing substrate evolved for body control and simply extended it to support the complex cognitive functions of symbolic ratiocination characteristic of human intelligence.

The SERVO STACKS architecture is an attempt to use this insight as a guide in developing a general artificial intelligence. For example, there is some evidence that the neural structures which manipulate grammar and language developed from, or indeed overlap, the ones which produce finely nuanced and sequenced manipulations by the hands. Thus we feel that insight into the former may be gained by investigating the latter, particularly to the extent of identifying mechanisms that might be responsible for compliant dexterity and seamless fluidity in both domains.

It is common in higher-level cognitive architectures for there to be a chain something like "sensory input to interpretation to cogitation to action sequencing to motor output." In evolution, however, the structure of a simpler animal is usually augmented by adding a controller to the top the entire lower-form structure, giving rise to a laminar structure with crosstalk between input and output at each level. This form is clearly reflected in the control architectures of modern robotics as pioneered in (Brooks 1986).

To extend this laminar control architecture to be a more general-purpose cognitive system, we reinterpret the afferent and efferent signals at each level, together with their crosstalk in both directions, as a feedback-loop servo controller. The resulting stack of servos then represents a classic abstraction hierarchy, with each successive servo "thinking" about what is happening in terms of a more general "language" as we ascend the stack.

For example, we might have lower-level servos concerned with pressure patterns on the skin and muscle activation strengths; a higher level with joint angles and speeds; higher yet with step-taking and foot placement; then room-to-room navigation; then the task of serving coffee; then the entertainment of guests, and so forth.

Given this general form for a cognitive architecture, the key open questions are

- Can a single general model of servo be developed that is appropriate to a broad range of these levels? If so, the notion that the brain grows by copying and modifying existing servos, both evolutionarily and in individual mental development, would gain increased cogency.
- The servos in a working model of mind will not form a simple linear stack but a complex network, which must support varying patterns of communication for different overall cognitive tasks and states. How is this done?
- For mental growth and learning, new servos, with new internal "languages", must be formed to implement the ability to think in new abstractions about new concepts. How are they created, programmed, and connected?

After examining these we will return to the question of a specific architecture.

A Symbolic Servo

Upon seeing the algorithm of Newell and Simon's General Problem Solver, John McCarthy quipped, "It's a symbolic servo."[1]

A standard process-control servo receives a control input signal s (the "setpoint") and a feedback signal f from the process (called the "plant") to be controlled. These two signals must be commensurate, e.g. both a measure of engine speed. The servo then adjusts some controllable parameter p of the plant based on a function of the control and feedback signals.

If the servo has a memory of enough cases, arranged as triples (f_0, f_1, p) (e.g. the value of the feedback signal before and after setting the output to p), "case-based" control can obtained by using p from the tuple where f_0 is closest to current feedback f, and f_1 is nearest the current control input s. For well-behaved plants, even a sparsely populated memory can be interpolated with standard numerical regression techniques.

The Servo Stacks model specifies a controller whose memory is (s, p, d, f), where s is the setpoint, $f = f_0$ is the original value of the feedback signal, $d = f_1 - f_0$ the differential of the trajectory, and p is the process control signal sent to the plant. The use of a differential as a trajectory link (instead of explicitly storing the successive value) is prompted by the observed difficulty of following well-practiced behaviors backwards, and because matching differentials affords a cheap generalization over translation, and with a nod to the "difference operators" in GPS (Newell and Simon 1963). It also brings us within shouting distance of the standard transfer function formulation of linear dynamical systems in control theory, although we have yet to take practical advantage of this.

We refer to this as a "sigma" (*si*tuation / *g*oal / *m*emory / *a*ction) to distinguish it from standard formulations of servo or state machine. In our model, each signal has a strength as well as a value; at zero strength, it acts as a "don't-care", and at partial strength it acts as a bias in case the other inputs incompletely determine a memory record, but can be overriden by stronger signals on other inputs. More precisely, each signal is a vector whose components individually have strengths.

If sufficiently populated, the stored trajectories form a manifold in the memory space, which is equivalent to an equation (or system of them) which forms a model of the observed behaviors. Typically the manifold will be a p-valued function of (f, s)-space, and generally interpreted as a function selected by s in f-space.

If some of the dimensions of the space are time derivatives of others, it forms a dynamical systems model (equivalent to a system of differential equations). In the absence of such derivatives, the sigma can either be clocked to obtain state-machine behavior or allowed to run free, in which case it will find fixed points of its function.

[1]As related by Marvin Minsky at AAAI FSS 2001, North Falmouth, MA

A sigma can be used in a variety of modes, depending on its inputs:

1. Homeostatic servo mode: as above, s is driven by a control signal from above, f is driven by the feedback signal from below, d is sent back up as a feedback. p is output to drive the plant, and may optionally be added into d as part of the feedback.
2. Sequencing control mode: s is driven by control from above, $d + f$ is output to drive f, and $p + d$ is sent back up as feedback.
3. Simulate mode: s is driven by control from above, $d + f$ is output to drive f, and $p + d$ is sent back up as feedback.
4. Recognize mode: f and d are driven by the signal from below and its derivative; s and $p + d$ are output to be sent back up as feedback. s may also be fed back weakly to itself, and/or driven weakly from above for a priming effect.

Signals and Representation

Analogy (Hofstadter 1995) and blending (Fauconnier and Turner 2003) have both been suggested as key basic operations that a cognitive architecture must implement. It is instructive to note that vectors of real numbers support both these operations as simple geometric combinations, while representing concepts as complex as a recognizable sketch of a human face (Churchland 1986; 2005). Although at higher levels of abstraction it is surely the case that more complex structures and operations will be necessary to support analogy and blending, it seems reasonable to begin with a primitive representation that provides a head start.

The physical signals in Servo Stacks are fixed-length numeric vectors of length n ($n = 1024$ in current experiments), but support an arbitrary number of notional signals. The components of the notional signals are given a superimposed coding as sums of randomly-chosen base vectors in the physical signal space. Any two such vectors have a high probability of being nearly orthogonal (Kanerva 1988). Any port on any sigma can be connected to any other port, with a high probability that notional signals unknown to the receiver will simply be ignored (by virtue of being orthogonal to the active manifold). Notional signals can be generated locally without the need of an overall registry, and can be combined simply by adding physical signals. Hereinafter, we shall ignore the encoding and refer only to the notional signal vectors of arbitrary length.

Such vectors can obviously record the position in configuration space of an arm, or be interpreted as a raster (including the areal functions of dynamic neural field theory (Erlhagen and Bicho 2006)), or specify force values for muscles. However, throughout most of the architecture, they will typically record the activation of, and signals to be sent to, sets of other sigmas. In the sense that such a vector can be thought of as representing a situation as recognized or understood by the other sigmas, it functions as a frame (Minsky 1975). In cases where that is an appropriate interpretation, the sigma itself acts as a frame-system, predicting new situations as the result of actions or suggesting actions given desired situations.

It is perhaps important to emphasize that there is not one big vector space in which everything is represented, as in some connectionist theories. Each sigma has its own language, in the sense of an interpretation of the vectors. More precisely, each pair of sigmas which communicate have a common interpretation for the signals that pass between them, but these interpretations vary completely from one area to another.

Stacks and Networks

Servos are commonly arranged in stacks in complex mechanisms. Upper-level servos may send control signals directly to lower ones (the attitude-control system in a fly-by-wire aircraft commanding aileron servos) or indirectly through a process parameter (room thermostats in a home hydronic heating system run hot-water pumps, relying on a homeostasis provided by a boiler thermostat). Leading roboticists Albus (Albus 1992) and Brooks (Brooks 1986) have based their flagship architectures on hierarchies of FSAs.

Thus the Servo Stacks model, as a layered network of elements that can act as sequencing or homeostatic controllers, is straightforwardly adapted into these roles. A key difference between this and a conventional view of a control hierarchy, however, is that sigmas by their nature provide a sensing and translation function. The setpoint and feedback signals s and f are necessarily in a different "language" than p, and thus, in a directly connected stack, from the s and f of the sigma below.

A servo stack is as reasonable a model for a sensory pathway as for a motor one. It is becoming increasingly recognized that sensing is neurophysiologically an active, not a passive process (Wolpert, Ghahramani, and Jordan 1995). In vision, for example, there are examples of homeostasis, such as iris dilation for retina illumination levels, and sequencing control, as of saccading in intermediate object recognition to trace boundaries and salient features.

Holonic recognition, including key features such as priming and context sensitivity, is readily provided for in Servo Stacks by weakly driving the s input from above. A key feature of the superimposed coding of signals is that the signal representing a percept can have components corresponding to several possibilities, and the interpretation process may provide (efferent!) feedback from semantic constraints.

At the lower sensorimotor levels, the reconfigurability of the sigmas is not important, and indeed there may be a level below which they are hard-wired and cognitively impenetrable. However, at higher levels where sigmas represent concepts such as words and goal-directed actions, reconfigurability is crucial to the use of the network as a fabric of representation. The same set of sigmas can be used to recognize an action, imagine doing it, predict its effects, and actually to perform it.

Recursion and Concepts

We posit a mechanism similar to Minsky's *k-line* (Minsky 1980), which can record all the network elements active at a given point, but also able to record their connections: which port on each is driving which other ports at what strength. These *active subnet configurations* (hereinafter ASCs) can be named and passed as values between higher-level sigmas, which can perform operations on them such as substitution: the ASCs for "pick up the red block", "red block", and "blue ball" can be combined in such a way as to form an ASC for "pick up the blue ball".

The formation of ASCs is at least neurally plausible. A generally broadcast signal could cause all active elements to associate a code (one component of the notional vector) with their current activity or connectivity, and a similar broadcast could cause that activity and connectivity to be reestablished. A number of associative datastructure manipulation techniques are similarly plausible for operations between ASCs (Foster 1976; Potter 1991; Hall 1994).

Since ASCs are capable of recursion (in the linguistics sense, i.e. capable of being combined to form complex structures from simple ones) and other typically symbolic manipulations, they form a "language of thought" in the sense of Fodor (Fodor 1975; 1978).

Perhaps the most important question one can ask of a cognitive architecture is how it represents concepts. In the common quip, something is a duck if it walks like a duck and quacks like a duck. Rather than representing a duck as some static datastructure essentially equivalent to a dictionary entry, an ASC represents a duck with an active *working machine* that is capable of recognizing a duck, simulating a duck, and even imitating a duck. This is the active subnet which activates and connects all the sigmas which store duck-relevant trajectories.

Note again that ASCs are manipulated, not by some exogenous program, but by sigmas whose memories p are records of ASC-manipulation control signals. The s and f signals to such sigmas are typically the output of more complex processing and recognition. The higher-level sigmas which manipulate ASCs are no different in principle from any others – manipulating one's own thoughts must be learned and practiced.

Play, Practice, and Imitation

Evidence ranging from visually deprived kittens (Wiesel and Hubel 1963) to language-deprived children (Sacks 1989) indicates that appropriate sensory experience is necessary for the development of cognitive ability across a wide range of levels of organization. The same phenomenon would affect our sigma, which would be incompetent at its task in the absence of a populated memory.

For a sigma to become competent at recognition or control, its memory must be populated with enough trajectories to form a reasonable model of the dynamics of the space implied by its signals. Underpopulated sigmas can improve their models by indulging in play: driving outputs so that the resulting state falls into vacant regions of the space. The plant to be driven in such play can be either the actual robot body, through the entire sensorimotor stack below the playing sigma, or simulations, in which case the stack is disconnected by putting some cutset of intermediate sigmas into simulate mode.

By far the major mode of human learning is imitation. After a sigma records the experience of someone else doing an action, the action may be imitated substituting oneself for the original actor in the ASC. This will rarely be perfect, but it gives the student mind a scaffolding upon which to improve by practice.

An Architecture

The testbed for Servo Stacks is an upper-torso anthropoid robot in the genre of Cog (Brooks et al. 1999). The robot is intended to be able to learn enough skills and concepts to be roughly the equivalent of a SHRDLU (Winograd 1972), but with physical cameras, manipulators, wooden blocks and table, and hearing and responding in spoken language.

Given the demanding computational and memory requirements of the Servo Stacks model, particularly the associative memory of the sigmas, it seems likely that processing power will form a substantial bottleneck for the near future. We consider this appropriate, however: any biologically inspired theory of the mind must take into account the substantial apparent processing power of the brain (Merkle 1987). Any such theory whose computational requirements fit available computer hardware too neatly seems suspiciously *ad hoc*.

Autogenous Kernel

We are primarily concerned with the ability of a mind to learn and grow. We adopt the basic architecture of a self-extending system from (Hall 1999), which specifies an "autogenous kernel" with irreducibly basic self-construction capabilities, which builds, in successive layers, a series of extensions that have both more general capabilities and more sophisticated self-construction abilities.

In a cognitive architecture, the kernel consists of some basic sensorimotor sigmas, pre-programmed with records that allow for infant-like activities like waving arms and learning to track them with eyes. Perhaps more importantly, it contains higher-level sigmas pre-programmed with records that allow them to do basic ASC manipulation.

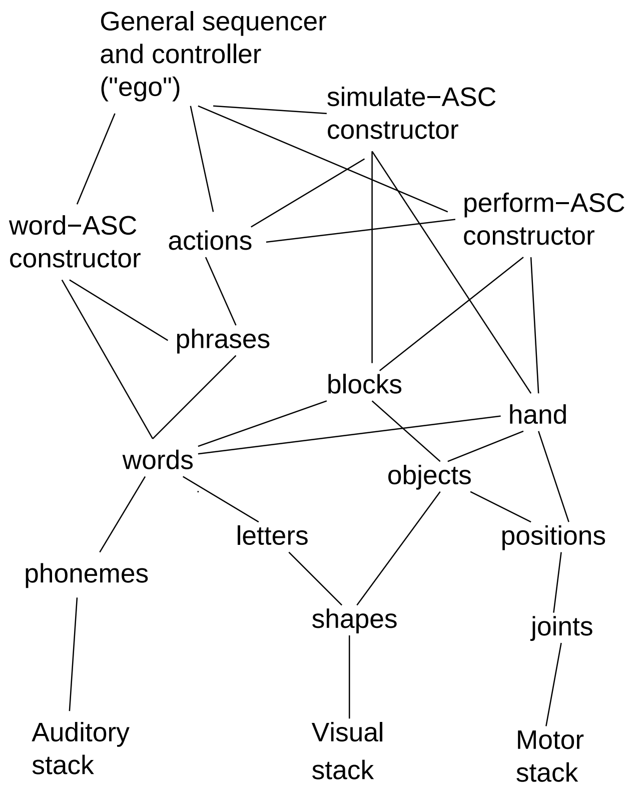

Figure 1: Kernel robot cognitive architecture.

The key issue in learning is the provenance of the mappings of signals and connectivity of the sigmas. For example, consider the sigma that maps from 3-D space to the configuration space of a multi-jointed arm, constituting the forward or inverse kinematics function depending on how it is used. This works after it is populated – where did the 3-D space come from originally, however?

The process starts with a handful of data-mining heuristics that are applied more or less at random to all the signals in the system. In early experiments these are Kohonen map formation, other standard dimensionality reduction techniques such as PCA, and hierarchical clustering using affinity propagation (Frey and Dueck 2007).

Sigmas generated this way are run in prediction mode and compete in an agoric/genetic ecosystem (Hall, Steinberg, and Davison 1998) on the basis of how well their predictions match experience. Those that succeed are linked together by the supply-chain dynamics of the agoric phase of the ecosystem and continue to compete in the marketplace. (The top-level inflow of funds in this system corresponds to motivations, and is currently a very simplistic stub that provides for answering questions and performing requested actions.)

Sensorimotor stack confluence

A key feature of the Servo Stacks model is that the stacks implementing any given sensorimotor modality (such as vision) are not separate but merge into other stacks at relatively low levels. In the testbed robot architecture, the main stacks are vision, hearing, and manipulation.

- Vision and manipulation converge at a low level to allow a fairly tight hand-eye coordination control loop. The upper end of this confluence feeds into an object-model stack.
- At a somewhat higher level, vision and hearing converge at the level of letters/words, feeding into a language stack.
- The language stack and object model converge at a level such that model semantics are active in sentence parsing, as in SHRDLU.

When the robot hears the sentence, "Dutch the blue blog," the sigmas which store word and syntax-related trajectories – words are trajectories through phonemes, sentences are trajectories through words – are configured to perform a spreading-activation parse similar to a Jones APN (Jones 1983; Jones and Driscoll 1985). There is enough feedback in this process to force the recognition of "Touch the blue block." The recognition itself will involve setting up the performance sigmas in simulation mode (at a high level only) to verify that the meaning is within the robot's experience. This will amplify the salience of "touch" and diminish that of "dutch", etc.

Memory

There is no separate module labelled "memory" in the architecture. Long-term memories are represented by the trajectory memories in all the sigmas and the mappings from signals to ASCs. Short-term or working memory is represented by the content of the active signals (including which ASCs are active).

Memories within a given sigma are managed, in our early experiments, by clustering and weighting heuristics (and by having each sigma have a fixed, limited capacity). Segmentation of trajectories into distinct traces is frankly *ad hoc* and is an area of active investigation.

Related Work

Servo Stacks falls squarely in the field of cognitive robotics (Clark and Grush 1999; Sloman et al. 2006). It shares in particular a strong concern for ontogenetic development with such projects as iCub (Tsagarakis et al. 2007). It is distinguished from many of the specific efforts in developmental robotics (Lungarella et al. 2003), however, by focussing on function, representation, and organization at a higher level than a neural network model (Shanahan 2006) or even a dynamic neural fields model (Erlhagen and Bicho 2006).

Minsky and Papert's original "Society of the Minds" cognitive architecture (Minsky 1977) was considerably more neurally inspired (and more oriented toward mental growth) than the majority of the "agent-based" architectures which followed – although the latter typically had the advantage of being more well-specified and indeed actually being implemented. The present work is strongly inspired by SoM but differs from it in several significant ways, particularly in the notion that ASCs can be handed from agent to agent as values. (Note that the first published mention of a rule-based controller reconfigurable as a simulator is in Marvin Minsky's Princeton dissertation (Minsky 1954)!)

Our model follows Grush's theory (Grush 2004) of representation as emulation based on forward modelling in Kalman filters and presumed equivalent functionality in the brain in several ways.

The notion of simply using the full record of experience as a model is generally referred to in AI as "case-based reasoning" (Kolodner 1992). Some of the neural-network approaches that allow one-shot learning are the original Hopfield network (with its Hebbian programming algorithm) (Hopfield 1982) and Aleksander's G-RAM neural unit model (Aleksander 1990). Servo Stacks might be implemented in either of these; for example, similar in spirit to our use of sigmas as associative-memory based sequencers is Orponen's programming language for Hopfield nets (Orponen and Prost 1996). However, for any digital simulation, an exhaustive search of a vector list is as fast as one single iteration of a Hopfield relaxation, so we choose instead to concentrate on the abstract properties of proximity in n-spaces and assume that conventional techniques can be developed to implement them efficiently (Shakhnarovich, Darrell, and Indyk 2006; Garcia, Debreuve, and Barlaud 2008).

References

Albus, J. S. 1992. Rcs: A reference model architecture for intelligent control. *IEEE Computer* 25(5):56–59.

Aleksander, I. 1990. Neural systems engineering: towards a unified design discipline? *Computing and Control* 1:259–265.

Brooks, R. A.; Breazeal, C.; Marjanovic, M.; Scassellati, B.; and Williamson, M. M. 1999. The cog project: Building a humanoid robot. *Lecture Notes in Computer Science* 1562:52–87.

Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE J. of Robotics and Automation* RA-2:14–23.

Churchland, P. M. 1986. Some reductive strategies in cognitive neurobiology. *Mind* 95(379):279–309.

Churchland, P. M. 2005. Vector completion, relevant abduction, and the capacity for 'globally sensitive' inference. In Raftopoulos, A., ed., *Cognitive Penetrability of Perception: Attention, Action, Strategies, and*

Bottom-Up Constraints. Nova Science Publishers. 1–12.

Clark, A., and Grush, R. 1999. Towards a cognitive robotics. *Adaptive Behavior* 1(7):5–16.

Erlhagen, W., and Bicho, E. 2006. The dynamic neural field approach to cognitive robotics. *Journal of Neural Engineering* 3:36–54.

Fauconnier, G., and Turner, M. 2003. Conceptual blending, form and meaning. *Recherches en communication* 19(19):57–86.

Fodor, J. 1975. *The Language of Thought*. Thomas Y. Crowell Company.

Fodor, J. 1978. Tom Swift and his procedural grandmother. *Cognition* 6:204–224.

Foster, C. C. 1976. *Content Addressable Parallel Processors*. Wiley.

Frey, B. J., and Dueck, D. 2007. Clustering by passing messages between data points. *Science* 315:972–976.

Garcia, V.; Debreuve, E.; and Barlaud, M. 2008. Fast k nearest neighbor search using gpu.

Grush, R. 2004. The emulation theory of representation: Motor control, imagery and perception. *Behavioral and Brain Sciences* 27(3):377–442.

Hall, J. S.; Steinberg, L.; and Davison, B. D. 1998. Combining agoric and genetic methods in stochastic design. *Nanotechnology* 9(3):274–284.

Hall, J. S. 1994. *Associative Processing: Architectures, Algorithms, Applications*. Ph.D. Dissertation, Rutgers University.

Hall, J. S. 1999. Architectural considerations for self-replicating manufacturing systems. *Nanotechnology* 10(3):323–330.

Hofstadter, D. 1995. *Fluid Concepts and Creative Analogies*. New York: Basic Books.

Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, volume 79. Washington, DC: National Academy Press.

Jones, M. A., and Driscoll, A. S. 1985. Movement in active production networks. In *Meeting of the Association for Computational Linguistics*, 161–166.

Jones, M. A. 1983. Activation-based parsing. In *IJCAI-VIII*, 678–682.

Kanerva, P. 1988. *Sparse Distributed Memory*. Cambridge, MA: MIT Press.

Kolodner, J. L. 1992. An introduction to case-based reasoning. *AI Review* 6:3–34.

Lungarella, M.; Metta, G.; Pfeifer, R.; and Sandini, G. 2003. Developmental robotics: a survey. *Connection Science* 15(4):151–190.

Merkle, R. 1987. Reverse engineering the brain. In *Proc. AIAA Computers in Aerospace VI*, 375.

Minsky, M. 1954. *Neural-Analog Networks and the Brain-Model Problem*. Ph.D. Dissertation, Princeton University.

Minsky, M. 1975. A framework for representing knowledge. In Winston, P. H., ed., *The Psychology of Computer Vision*. McGraw-Hill.

Minsky, M. 1977. Plain talk about neurodevelopmental epistemology. In *IJCAI-V*, 1083–1092.

Minsky, M. 1980. K-lines: A theory of memory. *Cognitive Science* 4(2):117–133.

Mountcastle, V. B. 1978. An organizing principle for cerebral function: The unit model and the distributed system. In Edelman, G. M., and Mountcastle, V. B., eds., *The Mindful Brain*. MIT Press.

Newell, A., and Simon, H. 1963. GPS: A program that simulates human thought. In Feigenbaum, E., and Feldman, J., eds., *Computers and Thought*. New York, NY: McGraw-Hill. 279–296.

Orponen, P., and Prost, F. 1996. Parallel programming on hopfield nets. In *Proc. Finnish AI Conference*, 5–12.

Potter, J. L. 1991. *Associative Computing: A Programming Paradigm for Massively Parallel Computers*. Perseus Publishing.

Sacks, O. 1989. *Seeing Voices, A Journey into the World of the Deaf*. Berkeley, CA: University of California Press.

Shakhnarovich, G.; Darrell, T.; and Indyk, P. 2006. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press.

Shanahan, M. 2006. A cognitive architecture that combines internal simulation with a global workspace. *Consciousness and Cognition* 15:433–449.

Sloman, A.; Wyatt, J.; Hawes, N.; Chappell, J.; and Kruijff, G.-J. M. 2006. Long term requirements for cognitive robotics. In *Proceedings CogRob2006, The Fifth International Cognitive Robotics Workshop. The AAAI-06 Workshop on Cognitive Robotics*.

Tsagarakis, N. G.; Metta, G.; Sandini, G.; Vernon, D.; Beira, R.; Becchi, F.; Righetti, L.; Santos-Victor, J.; Ijspeert, A. J.; Carrozza, M. C.; and Caldwell, D. G. 2007. icub: the design and realization of an open humanoid platform for cognitive and neuroscience research. *Advanced Robotics* 21(10):1151–1175.

Wiesel, T. N., and Hubel, D. H. 1963. Single-cell responses in striate cortex of kittens deprived of vision in one eye. *J Neurophysiol* 26:1003–17.

Winograd, T. 1972. *Understanding Natural Language*. Academic Press.

Wolpert, D. M.; Ghahramani, Z.; and Jordan, M. I. 1995. An internal model for sensorimotor integration. *Science* 269(5232):1880–1882.