To appear in *Proceedings of the First International Conference on Autonomous Agents* (1997). Marina del Rey, CA: ACM.

Learning to Sense Selectively in Physical Domains

Pat Langley*(LANGLEY@CS.STANFORD.EDU)
Robotics Laboratory, Computer Science Dept.
Stanford University, Stanford, CA 94305

Abstract

In this paper we describe an approach to representing, using, and improving sensory skills for physical domains. We present ICARUS, an architecture that represents control knowledge in terms of durative states and sequences of such states. The system operates in cycles, activating a state that matches the environmental situation and letting that state control behavior until its conditions fail or until finding another matching state with higher priority. Information about the probability that conditions will remain satisfied minimizes demands on sensing, as does knowledge about the durations of states and their likely successors. Three statistical learning methods let the system gradually reduce sensory load as it gains experience in a domain. We report experimental evaluations of this ability on three simulated physical tasks: flying an aircraft, steering a truck, and balancing a pole. Our experiments include lesion studies that identify the reduction in sensing due to each of the learning mechanisms and others that examine the effect of domain characteristics.

Introduction

Autonomous physical agents interact with the environment through sensors and effectors, and AI research on physical control has focused on taking the proper actions under the right perceptual conditions. Most work on execution and sensing employs a closed-loop approach that samples all available sensors on every time step. However, such work typically assumes that there is no cost to sensing and that the agent has unlimited perceptual resources.

Clearly this assumption does not hold for humans in many domains. Information overload is a common difficulty in tasks like flying an aircraft, and we believe AI systems will encounter similar problems on tasks of equivalent complexity. We hold that the standard closed-loop scheme makes sense only in domains where the results of sensor tests are unpredictable and where the sensing process is cheap. Given an environment where test results change infrequently or where sensing is expensive, a strategy that senses only occasionally is both possible and desirable.

In this paper, we describe ICARUS, an architecture for physical agents that addresses the issue of selective sensing. Rather than sampling sensors on each time step, the system checks them only if this seems likely to provide useful information. As in other areas of endeavor, intelligent sensing must rely on knowledge of the domain. This in turn suggests a role for learning, since we would prefer to avoid the task of encoding such knowledge manually. Instead, we would like ICARUS agents to acquire their strategies for selective sensing from experience with domains in which they operate.

In the following sections, we present the architecture and illustrate its behavior in a simple sensori-motor domain. First we consider ICARUS' representation of control knowledge, then turn to the ways that it uses that knowledge to operate in the environment. After this, we examine the architecture's learning mechanisms and the manner in which they alter performance with experience. Next we present experimental evidence of the system's learning ability in three simulated domains, and examine the components and domain characteristics responsible for the improvement. In closing, we review ICARUS' relation to other work on agent architectures and suggest some directions for future research.

Representation of Control Knowledge

Before we can describe the mechanisms that underlie ICARUS' performance and learning, we must first explain the knowledge structures on which these processes operate. The basic structure in long-term memory is the *state*, which describes a duration of time during which certain characteristics of the environment hold. This idea corresponds closely to the notion of a quali-

*Author's new address: Intelligent Systems Laboratory, Daimler-Benz Research and Technology Center, 1510 Page Mill Road, Palo Alto, CA 94304.

tative state in research on qualitative physics (Forbus, 1985; Kuipers, 1985), though here the characteristics may be either qualitative or quantitative in nature. Each state includes a name and a set of arguments that denote objects in the state, a set of activation conditions that must hold for the state to apply, and the expected duration of the state. Optionally, the state may also specify a set of actions, a set of effects these actions have on the state, and a set of successor states that are likely to follow the current one. The last feature also gives the flavor of a finite-state machine.

Consider a simple control task that involves balancing a pole hinged to a cart that moves only horizontally. Our formulation assumes three sensory variables: the pole's angle and angular velocity, in the activation conditions, and the pole's angular acceleration, in the effects field. This particular system does not sense the cart's position or velocity, though naturally these variables influence the simulation. Two actions are possible – pushing the cart to the left and pushing it to the right with the same force setting.

Figure 1 depicts graphically six states from this domain. One of these states, named `push-right-when-falling`, describes situations in which the pole is leaning to the right and falling downward in that direction. The single action for this state is `push-right`, which applies a force to the right, and the expected result is an increase in the pole's angular velocity. Another state, `push-right-when-rising`, has the same action and expected effects, but the angular velocity is negative, indicating that the pole is leaning to the right but rising rather than falling.

Icarus states also associate numbers with each descriptive literal. For activation conditions, these indicate the probability that the literal will continue to be satisfied on any given time step once the state is active. For the effects field, the numbers likewise show the probability that, if the actions are carried out while the state is active, they will have the specified effects. The numbers associated with each successor indicate the probability that the specified state will be active following the current one. A final field specifies the mean and standard deviation for the state duration.[1]

The figure also shows some successor relations for this domain. The state `push-right-when-falling` is typically followed by either `push-right-when-rising` or `no-push-when-fallen-right`, which has no successors. Similarly, `push-right-when-rising` is succeeded by the state `push-left-when-falling`, which in turn comes before either `push-left-when-rising` or `no-push-when-fallen-left`, another terminating state. Finally, state `push-left-when-rising` precedes `push-right-when-falling`, closing the control loop.

[1]The architecture's inclusion of state durations constitutes an important difference from the framework of Markov decision processes, which models duration by letting states be their own successors. Icarus neither allows self transitions nor requires them, since it models state durations directly.

Icarus' state representation has much in common with Strips operators (Fikes, Hart, & Nilsson, 1971) in that they specify application conditions, actions, and effects; they also bear a close kinship to production rules (e.g., Anderson, 1983; Langley, 1987). However, recall that, in our framework, both actions and effects are optional. When the effects are absent, Icarus states are more similar to the entries in the state-action tables used in reinforcement learning. When actions are absent, they have more in common with the states used in qualitative physics. The inclusion of information about successors is consistent with the latter's notion of envisionment, but it diverges from standard operator, production-rule, or state-action representations, none of which store sequential knowledge.

In addition to its long-term state memory, Icarus also includes a perceptual short-term memory that contains information about the attributes of objects in the perceptual field. For the pole-balancing domain, this is where the system retains information about angle, velocity, and acceleration. We assume that this memory is updated only when the architecture invokes a sensor for a particular object; otherwise the contents remain unchanged from time step to time step.

Finally, a goal memory stores information about the relative desirability of certain states. This memory contains elements that specify state names and their instantiated arguments, along with their desirability on the current time step, which corresponds roughly to the state-action tables used in reinforcement learning to specify expected rewards. We store this information apart from states themselves because different orders are appropriate for different tasks (e.g., balancing the pole vs. making it fall over to the left), thus giving the architecture more flexibility. Moreover, different instantiations of a state may have different priorities. The current architecture requires the programmer to provide these priorities, though future versions should be able to learn this ordering.

Sensing and Execution in Icarus

Now that we have examined Icarus' representation of knowledge, we can turn to its use of that knowledge. The basic process operates in cycles, as in architectures for both production systems and reactive agents. On each cycle, the system senses the environment, determines which states match the current situation, selects one of these matched states, and executes its associated actions. However, once Icarus has activated a state S, it continues to keep S active until its activation conditions or expected effects become false, or until it finds a successor state that matches and that has a higher priority than S. This constitutes an important difference from most other architectures, which reconsider all possibilities on each cycle.

Another essential difference is that sensing, although closely intertwined with execution, does not occur by default. On most cycles, Icarus checks at most only those sensors that relate to conditions in the currently

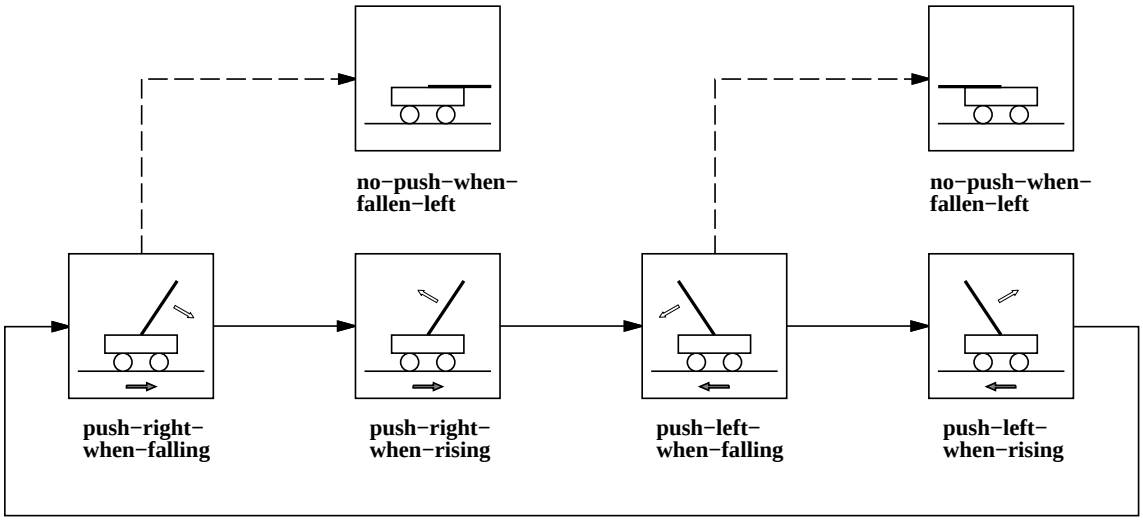

Figure 1: Six states for the pole-balancing domain that differ in the direction in which the pole leans, in the direction the pole is moving (light arrow), and in the direction of the applied force (dark arrow).

active state or those in its immediate successors. Nor does the system check the activation conditions of successor states on every time step. Instead, it carries out the sensing necessary for this process only when knowledge about the state's duration suggests that successors are likely to match. In particular, it assumes that the state duration obeys a normal distribution (an unlikely but convenient model) and uses the stored mean and standard deviation to compute the probability that successors are satisfied. When this quantity exceeds a system parameter (set to 0.84 in our studies), ICARUS invokes the sensors needed to test the conditions on the current state's successors.

Another strategy for reducing sensor load goes into effect once the system has activated a new state. Recall that each activation condition C has an associated probability S_C of being satisfied once the state has become active. ICARUS assumes that each time step constitutes an independent coin toss, letting it compute the probability $F_{C,T}$ that the condition will still be satisfied after T steps as ${S_C}^T$. The architecture carries out the sensing needed to test a condition C only when $F_{C,T}$ drops below another global parameter (set to 0.5 in our runs). The system uses the same strategy for checking expected effects. For cycles on which a sensor is not called, ICARUS assumes the value in short-term memory, which comes from the most recent invocation, is sufficiently accurate.

For example, the state `push-right-when-falling` contains four activation conditions. Two literals simply posit that the pole has an associated angle and angular velocity; thus, they have an associated probability of one. The two other literals state that the angle and velocity are greater than zero. These have an estimated probability of 0.4, so that if the parameter is 0.5, they would produce sensing on every cycle.[2] In contrast, if a condition had an estimated probability of 0.8, it would produce sensing only on every fourth cycle, since $0.8^4 < 0.5 < 0.8^3$. Thus, ICARUS senses no more than necessary to be reasonably sure that the conditions and effects of the current state are still satisfied.

In some cases, the architecture finds that either the activation conditions or the expected effects of the state S are not matched on the current time step, and that no known successor states are satisfied. In this situation, the system does apply its sensors exhaustively to all objects in the environment in an attempt to find an appropriate state. (ICARUS uses the same scheme on the first cycle, since it has no expectations to guide it.) If one or more states match, the system selects the state with the highest goal priority score and makes it active; these scores also resolve conflicts among different instantiations of the same state. However, in our experience the system seldom encounters such situations, provided it has states that accurately describe the domain. To date, we have tested the system only using noise-free sensors, but we anticipate that averaging sensor values across a number of cycles will produce similar behavior even when noise is present.

To summarize, ICARUS' behavior on each cycle involves carrying out the actions associated with the currently active state and, if the probability is low that the state's activation conditions or effects are still satisfied, invoking the associated sensors. If these literals no longer hold, or if the state's expected duration has been exceeded, the architecture examines the likely

[2] Because a literal like `(> ?a 0)` requires that the variable `?a` be bound, a decision to sense it can lead instead to sensing another literal like `(angle ?pole ?a)`, even when the latter is guaranteed to be satisfied.

successors for another state to activate. These strategies should reduce the system's sensing costs in domains with predictable tests and states of long duration, but they rely on accurate information about conditions, durations, and successors. Let us now consider the origin of these statistics.

Sensory Learning in Icarus

There exist many places within the Icarus architecture where learning might occur. However, our current work assumes that the basic control states and priorities are already in place, and that learning occurs within this context. This approach follows a long tradition, widespread within the machine learning community, that relies on background knowledge to constrain the learning process. Rather than attempting to acquire control knowledge, we have focused on three forms of statistical learning aimed at reducing the demand for sensing. These processes, which we describe below, learn about which states follow others, durations of states, and reliability of state conditions.

Learning about Successor States

Earlier we saw that states can include information about which states may succeed them, along with the probability of each option. This knowledge determines the states that Icarus checks to see if their activation conditions hold, and thus indirectly influences decisions about sensing. Although the programmer can provide the system with information about likely successors, it can also revise this information with experience.

This learning process operates in a straightforward manner. Each state S retains a count C for every successor that has occurred immediately following S, along with the total number of times T that S has been active. Icarus updates these counts each time a state stops being active and its successor has been identified. From these counts, the system estimates the probability for each state's successor as

$$\frac{C + C_I}{T + T_I} ,$$

where C_I and T_I are initial counts that correspond to a prior probability for each state-successor pair given at the outset.

By convention, we typically provide Icarus with no information about the successors of each state. Thus, for each state S it assumes initially that every state (except S itself, since self transitions cannot occur) may follow with equal probability, leading it to check the activation conditions for each one. As it gains experience in a given domain, the system obtains reliable statistics about the sequence of states that tend to occur in practice. Icarus ignores successors for which the estimated probability is below a certain threshold (set to 0.05 in the runs described later). As a result, for domains in which some successors are unlikely, such learning can reduce sensing costs, in that fewer successor states (and thus fewer activation conditions) must be checked. However, for domains in which successors are less predictable, the system will continue to check all candidates.

Learning about State Durations

We have also noted that states include information about their duration, and that the architecture uses this in determining when to consider possible successors. Icarus uses another simple updating scheme for this state parameter, retaining the total number of times the state has been active, the sum of its durations, and the sum of squares. Together, these let the system determine the mean and standard deviation of the state duration from experience.

Again, the effect of this learning depends on the domain. By default, we set each state's prior mean to 1.0 and its standard deviation to 0.1, which causes Icarus to check likely successor states on every time step. If the state's actual mean duration is low or its standard deviation is high, the system will continue this strategy regardless of the amount of experience with that state. In contrast, if the mean duration is high and the deviation low, then learning will gradually lead the agent to make ever longer delays before checking the successor states, thus reducing the number of activation conditions checked and decreasing the sensing load; this process will continue until the estimated mean approaches the actual one.

Learning about State Conditions

We have also mentioned that, during execution of an active state, Icarus uses the probability of success associated with each activation condition to determine how often to sense that condition, and that it uses a similar scheme for expected effects. Although the programmer specifies the initial probabilities for each condition, a third learning process revises these estimates in the light of experience.

In support of this activity, Icarus retains two counts for each condition and effect in a state: the number of times K it has been sensed during execution and the number of times H it has held when sensed. Every time the system carries out the sensing needed to test a literal, it increments K, but it increments H only if the literal is satisfied. As with successors, Icarus is also given a prior probability in terms of initial counts K_I and H_I. From these numbers it computes

$$\frac{H + H_I}{K + K_I} ,$$

the estimated probability that a condition will remain satisfied on a time step if that test was satisfied on the previous one.

By convention, we set the sensing threshold to 0.5 and initialize the probabilities for each condition and effect to 0.4, which guarantees Icarus will sense them on each time step, as this should produce the most rapid learning. For a literal that is seldom satisfied, the associated probability will remain low and sensing will

continue to be frequent. However, for literals that tend to remain satisfied for many time steps, the probability will increase and the system will gradually sense it less and less often.

Experiments with Sensory Learning

Although the intuitions behind our approach to sensory learning seem reasonable, we would prefer stronger evidence that Icarus' mechanisms lead to improved performance in practice. To this end, we constructed knowledge bases for three simulated control problems and carried out a number of experiments to evaluate the system's behavior. In this section we describe these domains, our experimental design, and our findings.

Three Simulated Control Domains

We selected three domains for our study that varied in their characteristics, in an attempt to ensure generality in our results. We have already described the pole-balancing task, which has been widely used in studies of reinforcement learning (e.g., Anderson, 1989; Selfridge, Sutton, & Barto, 1985). Our formulation of this domain uses only three sensory variables and one control variable, which exerts force to the left or right. The knowledge base we encoded for the task incorporates the six states shown in Figure 1, which are sufficient to balance the pole indefinitely before any sensory learning occurs. On each run, the pole starts with zero angular velocity and a randomly selected angle between -0.1 and 0.1 radians.

Our second task involves backing up a truck, composed of a cab and trailer, to a dock while moving at constant speed. Anderson and Miller (1991) describe the differential equations for this domain, whereas Nguyen and Widrow (1989) report an approach to learning control knowledge in this context. Anderson and Miller's formulation assumes four sensory variables – two coordinates for the back of the trailer and the angles for the trailer and cab. We have altered this slightly, using the cab's angle with respect to the trailer and adding a sensor variable for the wheel angle. The single control variable lets the agent increment or decrement the angle of the cab's wheels. Our control program for this domain incorporates 11 distinct states. On each run, the truck starts with the trailer between 10 and 30 degrees, with the cab angle between 5 and -5 degrees, and with the x coordinate between 50 and 70 meters from the dock, all sampled randomly from the uniform distribution.

The third control problem involves flying a simulated airplane through a three-dimensional slalom course; the plane must traverse a sequence of rectangular gates that occur at different angles and elevations from the plane's starting position in mid-air. The simulator supports sensors for the roll and pitch of the aircraft, its altitude, and a variety of other variables, including first derivatives of many measures. The agent also has information about the position of each gate, G, with respect to the plane, in terms of the apparent horizontal angles to the left and right sides of G, the vertical angles to the top and bottom of G, and the distance to G. For the two-gate courses used in our experiments, this gives a total of 14 sensors. Controls include the ability to reposition the control stick forward and sideways.[3] This slalom task may appear straightforward to the reader, but Goettl (1993) has found that most human subjects take hours to become proficient enough to go through all the gates without error. Our knowledge base for this task contains some nine durative states.

Basic Learning Results

Our main hypothesis was that the learning mechanisms described above can lead to a reduction in sensory load with little increase in error, and our first experiment was designed to test this prediction. Our main dependent measures were the probability of invoking each sensor on a given time step, augmented by the domain-specific measures of error discussed below. The primary independent variable, apart from the domain, was the number of training or practice runs the agent had carried out. Our aim was to generate learning curves that plotted sensory load as a function of experience. Here we assume that matching contributes little cost once the sensing process is complete, so that reduction in sensor load corresponds to potential increase in speed; this view contrasts with most work on 'speedup' learning (e.g., Minton, 1990), which emphasizes reduction in matching and search costs.

For each domain, we provided Icarus with the control states mentioned earlier and ran the system a number of times. To maximize sensing at the outset, we initialized each knowledge base with 0.4 probabilities for sensor conditions, with mean durations of 1.0 and deviations of 0.1, and with uniform probabilities for successor states. We turned all learning off for the first run to establish a baseline, then alternated between practice runs with learning and test runs without, the latter to obtain measures of performance for a given experience level.[4] For each domain, we repeated this process 20 times and combined the results to obtain an average learning curve.

Figure 2 (a) shows the average results, with 95% confidence intervals, for three sensors on the pole-balancing task, using 100 cycles for both practice and test runs. Icarus initially calls the sensors for pole angle and angular velocity on nearly every time step, but after ten practice runs, this has dropped to less than a two-thirds probability. The reduction for the angular acceleration is even more significant; its initial probability of being

[3] This task is distinct from the flight-control domain reported by Sammut, Hurst, Kedzier, and Michie (1992), which includes takeoff, traversal of a specified path, and landing on the runway.

[4] Because the tasks themselves can require alternation among states and sensors, this approach seemed more appropriate than tracking online learning, as typically done with reinforcement methods.

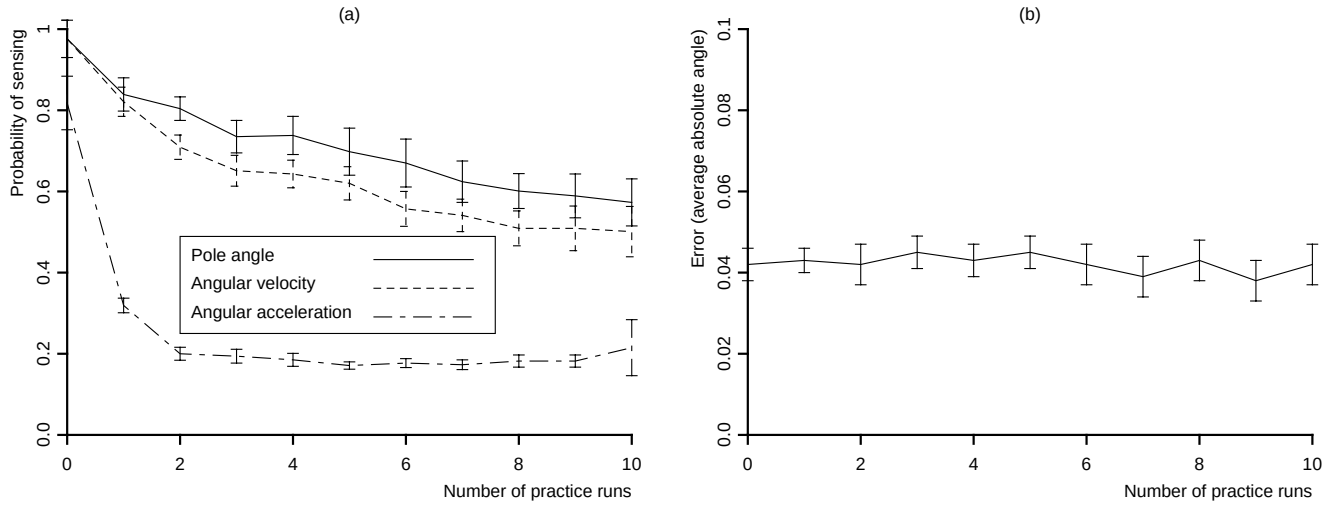

Figure 2: Learning curves for the pole-balancing domain: (a) the probability of invoking three sensors; and (b) error measured as the absolute angle of the pole from vertical. The results are averaged over 20 runs; the error bars represent 95% confidence intervals.

called is 0.8, but after two runs it has dropped to 0.2, having been reduced by a factor of four. Our measure of error here was the pole's average angular distance, in radians, from the upright position, which ranges from zero to ± 1.571 (when the pole has fallen); Figure 2 (b) shows that this error appears unaffected by the reduction in sensing. Moreover, additional test runs of 1000 cycles after the tenth practice trial showed that the system could balance the pole reliably for this period.

We observed very similar behavior (not shown here) in the truck-steering domain, with both practice and test runs lasting 90 cycles, for the wheel-angle, trailer-angle, and cab-angle sensors. On this task, each sensor was initially invoked on nearly every time step, but the probabilities for the trailer angle and cab angle dropped more rapidly than the wheel angle, which appeared to be less predictable. For this domain, we defined error as the final angle of the trailer, after the truck has reached the dock. As before, sensory learning did not seem to increase this performance measure; behavior was more erratic than for pole balancing but the angular error remained small (around three degrees). The important point is that, qualitatively, the system had no trouble aligning the trailer with either full or selective sensing.

Again, the learning curves for the flight-control domain (based on 70-cycle practice and test runs) reveal a similar picture. Here the sensors associated with the plane typically have a higher probability of invocation than those concerning gates, because the system focuses on the nearest gate, but in all cases the sensing probability generally decreases with experience. Figure 3 (a) shows the results for the plane's pitch and roll, as well as the left angle of the first gate. Other sensors have different slopes and intercepts but follow the same pattern. The measure of error here is the distance from the gate's center as the plane passes through the gate's x coordinate. Figure 3 (b) indicates this quantity remains unaffected with increased practice and reduced sensing; more important, the system continues to successfully thread gates throughout the learning process.

Sources of Power

The above experiments gave evidence that Icarus' learning processes can reduce sensory load without serious increase in errors, but it did not identify which mechanisms were responsible for the shift. Reasoning suggests that revising condition probabilities will have no effect without increases in expected duration, and that altering durations will not change behavior unless successor checking becomes selective. To test this hypothesis, we ran a 'lesion' study in which we excised some processes but retained others. We examined three experimental conditions: one in which all learning mechanisms were active, one in which only successor and duration learning operated, and one in which only successor learning occurred. Thus, our main independent variable besides experience level was the type of learning that took place, while our dependent measure was again the probability of sensing.

Figure 4 (a) gives the learning curves for angular velocity in the pole-balancing domain under these three situations. They indicate that all three learning processes must operate for reduction in sensing to occur on this task. We found similar results for the truck-steering domain, though here the combination of successor and duration learning gives minor improvement (not shown) on some sensor variables. However, the situation was very different for the flight-control task. As Figure 4 (b) shows for the roll sensor, learning only about likely successors still provides some improvement, revising the expected duration leads to additional reduction, and altering condition probabilities gives fur-

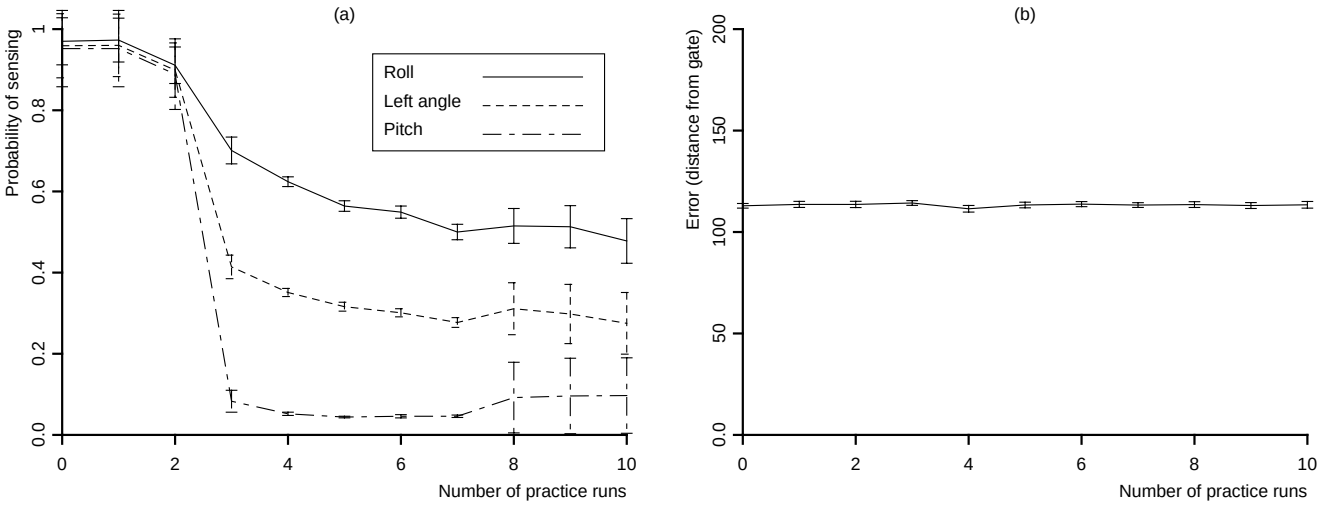

Figure 3: Learning curves for the flight-control domain: (a) the probability of invoking three sensors; and (b) error measured as distance from the gate center as the plane passes that gate's x coordinate.

ther benefit. We can explain this behavior in terms of this domain's states, which tend to employ different sensors; in the absence of condition learning, delayed and selective checking of successors should reduce sensing under such circumstances. In contrast, they should not help in domains for which states typically contain the same sensors, such as pole balancing and truck steering.

Effect of Domain Characteristics

As Kinny, Georgeff, and Hendler (1992) have shown, the optimal sensing rate varies with a domain's characteristics. For tasks in which the environment changes slowly and the effects of actions are predictable, infrequent sensing is sufficient to produce effective behavior; but for rapidly changing or unpredictable domains, more frequent sensing is needed. Our final experiments were designed to show that Icarus' learning mechanisms responded appropriately to such characteristics, so that they reduce sensing only to the level proper for the domain.

To this end, we systematically varied two aspects of the pole-balancing task, the simplest of our three domains. For one comparison, we altered the simulator parameter (*tau*) that specifies the length of each time step, giving conditions under which the simulator ran at different rates than in our previous studies. In another comparison, we modified the simulator to insert randomly a force twice the amount exerted by the agent, in a randomly selected direction. We intended this to mimic a malicious observer attempting to upset the balancing act. Here we varied the probability of force insertion to influence domain uncertainty.

The results of the first study appear in Figure 5 (a), which shows that when *tau* is higher than the default (0.005), the system continues to sense frequently (with nearly 0.9 probability), as it realizes the need for more updates to keep the pole balanced in the rapidly changing world. In contrast, when *tau* is low, Icarus senses less often (with only 0.4 probability) than in the default situation. The experimental results for unpredictable environments also agree with our expectations. Figure 5 (b) shows that increasing the probability of an external force raises the asymptotic sensing rate, which is appropriate given that the presence of outside influence requires more monitoring to avoid disaster.

In both the more rapid and the less predictable variants of the task, Icarus retained the ability to balance the pole, though naturally the average error was higher for the more difficult tasks, even with full sensing. In summary, our statistical learning mechanisms tend to acquire sensing rates that are appropriate for the task at hand, effectively taking into account the speed and uncertainty of the domain.

Related Work on Learning and Control

Icarus holds many features in common with previous approaches to building intelligent agents, but it also differs from them in important ways. One common framework for modeling intelligent behavior is known as a *production system*. Like Icarus, such architectures contain both short-term and long-term memories, with the latter consisting of condition-action rules. A production system also operates in cycles, on each round matching rules' conditions against the literals in short-term memory, selecting a rule for application, and using that rule's action side to determine behavior. Anderson's (1983) ACT and Laird and Rosenbloom's (1990) Soar are two well-known production-system architectures that include learning components, and much of the work on learning search heuristics (e.g., Langley, 1987; Minton, 1990) assumes a similar framework.

However, most research in this paradigm has focused on planning and problem solving rather than sensing and execution. The contents of short-term memory,

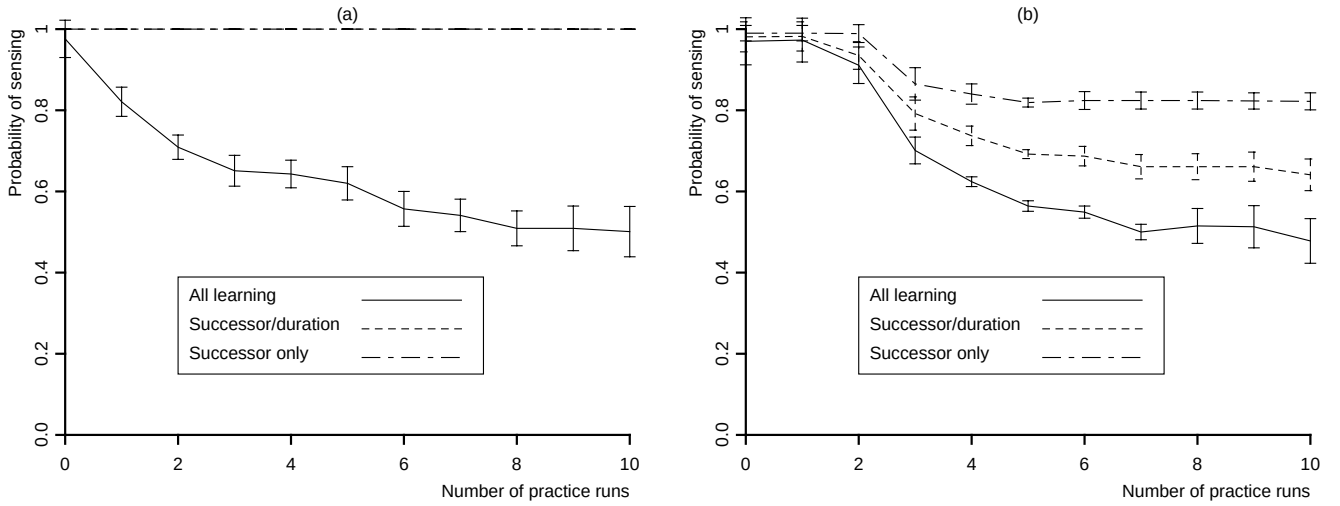

Figure 4: The effect of lesioning learning mechanisms for condition satisfaction and state duration on (a) the angular-velocity sensor in the pole-balancing domain and (b) the roll sensor in the flight-control domain. (The curves for successor only and successor/duration in (a) are identical.)

against which rules match, contain inferences rather than sensory information, and rules' actions generate such inferences rather than alter the environment. Learning typically involves changing the conditions on rules, composing multiple rules, or changing their relative priorities. Although this work emphasizes speeding up processing, it focuses on reducing search and matching costs during problem solving rather than reducing sensing costs during execution.

In contrast, research on reactive behavior directly addresses issues of sensing and execution. The standard reactive system operates in cycles, much like a production system. However, rather than matching against literals in short-term memory, it matches against perceptions of the environment, and rather than altering the contents of short-term memory, its actions affect the environment. Work on reinforcement learning (e.g., Watkins, 1989; Kaelbling, 1993) incorporates the same basic control scheme, but also stores priorities with state-action pairs for use in selecting actions. Such systems typically invoke all sensors on each cycle, making decisions locally with no notion of planning or projection. Within this paradigm, our work comes nearest to that by Grefenstette, Ramsey, and Schultz (1990), in that our states closely resemble their control rules.

Icarus approximates a reactive system when its state descriptions contain probabilities that force sensing on every cycle. However, it differs from them in that, having entered a state S, it keeps S active until its conditions no longer hold, or until a successor link directs its attention to an active successor state. Moreover, the system invokes only those sensors associated with the state during this period. The architecture's use of state priorities to select among successor states shares some features with strategies used in reinforcement learning, but its selective sensing strategies are quite different from the exhaustive approach to sensing commonly assumed in that framework.

Nilsson's (1994) approach to reactive behavior, which he calls *teleoreactive systems*, has the greatest similarity to our own. His scheme incorporates rules that match against sensory information, that execute actions which alter the environment, and that are durative in nature. The main differences are that Nilsson's framework assumes sensors should be checked on each cycle, and that it includes no statistical information about conditions, durations, or successor states to support selective sensing. Nevertheless, the two architectures share many basic assumptions and techniques, and Nilsson's work has influenced our stance on the integration of perception, action, and cognition. Benson (1995) describes an extension that learns models of states' conditions and effects, but that does not alter the sensing strategy.

Our approach to reactive control bears a strong resemblance as well to DeJong's (1994) work, which also uses qualitative states to represent knowledge about domains with continuous variables. DeJong's main concern lies with improving the accuracy of state descriptions over time, rather than with learning to sense selectively, and he stores no explicit information about succession among states. However, the qualitative representation of control knowledge and the general spirit of the work are much the same.

A few AI researchers have explored the idea of selective sensing, motivated by the realization that monitoring often incurs some cost. Chrisman and Simmons (1991) describe an approach to static sensing policies that, although selective, must be determined at development time. Another line of work is represented by Abramson (1991), Kinny, Georgeff, and Hendler (1992), and Langley, Iba, and Shrager (1994); each presents a formal model of plan execution that predicts the opti-

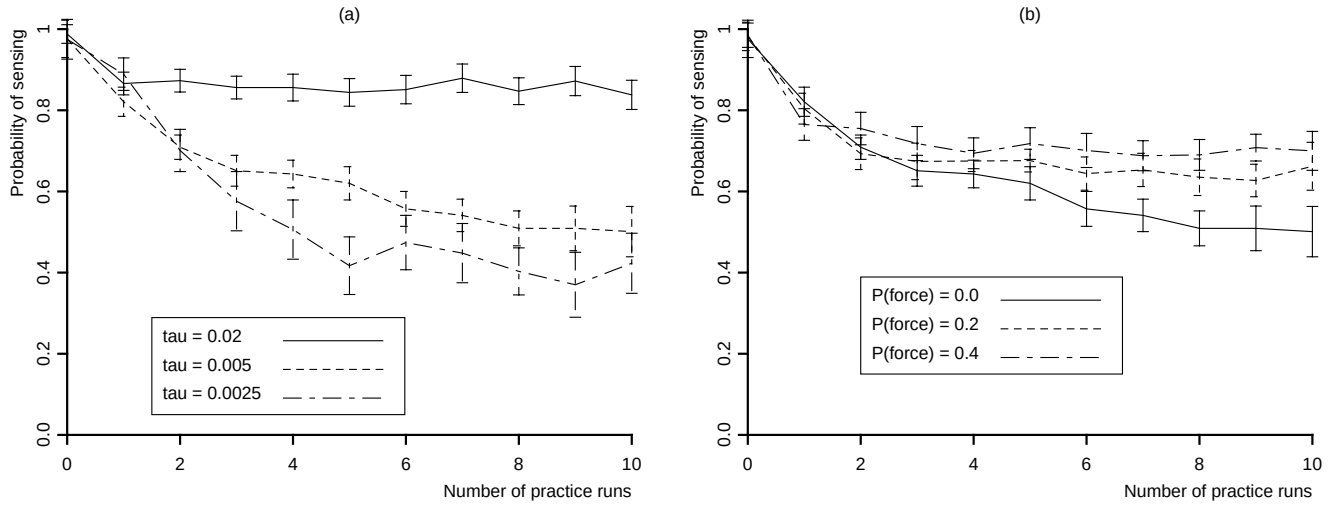

Figure 5: The effect on sensing probability for the angular-velocity sensor in the pole-balancing domain (a) when the rate of change is altered from the default ($tau = 0.005$) and (b) when there exists an outside force that affects the cart with some probability.

mal sensing rate under different levels of sensor cost and domain uncertainty. Their various models differ somewhat but share the assumption that a single sensing rate holds across the entire execution process. Tan (1991) reports work on the induction of sensing strategies, but his goal was to determine an efficient sequence of sensor calls, rather than to learn when sensing is necessary.

Icarus' association of separate statistics with each state has more in common with the framework of Hansen (1994), which represents the environment as a Markov model in which different sensing rates are optimal for different states. However, our method for determining when to sense relies on local state information, whereas his more sophisticated dynamic-programming scheme involves some search. Related work by Hansen and Cohen (1993) adapts ideas from reinforcement learning to determine sensing rates for each state from experience; again, their motivation is similar to our own but the details differ.

Concluding Remarks

In the previous sections we described Icarus, an architecture designed to control physical agents in domains where sensing resources are limited. We found that the system represents control knowledge in terms of durative states that borrow features from research in both planning and qualitative physics. We saw that Icarus can behave in purely reactive mode, but that it can also take advantage of knowledge about the probability of sensory conditions being unchanged, the duration of states, and likely successors. We also described three statistical learning mechanisms that can acquire this knowledge from experience, and we reported experimental evidence that they reduce sensing load without seriously increasing error.

Despite the progress we have made in developing and testing the architecture, clearly more work remains. In future research, we plan to further evaluate Icarus' performance and learning abilities in new control domains, as additional evidence of the system's generality. Obvious candidates include the problems reported by Grefenstette et al. (1990) and by Anderson and Miller (1991), for which simulators and results with other systems are available.

We also need to better motivate the architecture's bias toward sensing as seldom as possible. Humans appear to have severe limits on the number of environmental features that can occupy their attention, which presumably drives their need to sense selectively. Future versions of Icarus should include a limit on the number of features that can be inspected on each cycle, so that there is a clear function for the selective sensing made possible by learning. Under these conditions, frequent sensing could actually prohibit the agent from taking actions needed to handle the task, so that reduction in sensory load may reduce errors. Attaching distinct costs to sensors also opens the way to decision-theoretic methods for selecting the best sensors to sample on each time step.

Another research direction involves introduction of additional learning mechanisms, which are currently limited to altering the probability estimates on conditions, durations, and successors. Given our architecture's similarity to the teleoreactive systems of Nilsson (1994) and Benson (1995), it seems natural to borrow their approach to inducing action models for use in learning state descriptions from experience. Moreover, the system's use of priorities on states suggests that we should take advantage of techniques from reinforcement learning to alter those weights dynamically. Finally, we

should incorporate methods from Nordhausen and Langley (1993), who also represent knowledge as durative states, to induce numeric relations between sensor variables and state duration, which could further inform the sensing process. Taken together, these extensions should make Icarus a more robust and flexible architecture for controlling physical agents.

Acknowledgements

Thanks to Nils Nilsson, Dan Shapiro, Claude Sammut, Scott Benson, and Jeff Shrager for discussions that improved our designs of the architecture. This research was funded by AFOSR Grant No. F49620-94-1-0118.

References

Abramson, B. (1991). An analysis of error recovery and sensory integration for dynamic planners. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 744–749). Anaheim, CA: AAAI Press.

Anderson, C. W. (1989). Learning to control an inverted pendulum. *IEEE Control Systems Magazine*, *it 9*, 31–37.

Anderson, C. W., & Miller, W. T. (1991). Challenging control problems. In W. T. Miller, R. S. Sutton, & P. J. Werbos (Eds.), *Neural networks for control*. Cambridge, MA: MIT Press.

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge: Harvard University Press.

Benson, S. (1995). Inductive learning of reactive action models. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 47–54). Lake Tahoe, CA: Morgan Kaufmann.

Chrisman, L., & Simmons, R. (1991). Sensible planning: Focusing perceptual attention. *Proceeding of the Ninth National Conference on Artificial Intelligence* (pp. 756–761). Anaheim, CA: AAAI Press.

DeJong, G. F. (1994). Learning to plan in continuous domains. *Artificial Intelligence*, *64*, 71–141.

Fikes, R. E., Hart, P. E., & Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*, 189–208.

Forbus, K. D. (1985). Qualitative process theory. In D. G. Bobrow (Ed.), *Qualitative reasoning about physical systems*. Cambridge, MA: MIT Press.

Goettl, B. P. (1993). *Analysis of skill on a flight simulator: Implications for training*. Unpublished manuscript, Armstrong Laboratory, Brooks Air Force Base, Texas.

Grefenstette, J. J., Ramsey, C. L., & Schultz, A. C. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning*, *5*, 355–381.

Hansen, E. A. (1994). Cost-effective sensing during plan execution. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 1029–1035). Seattle, WA: AAAI Press.

Hansen, E. A., & Cohen, P. R. (1993). Learning monitoring strategies to compensate for model uncertainty. *Working Notes of the AAAI-93 Workshop on Learning Action Models* (pp. 33–35). Washington, D.C.: AAAI Press.

Kaelbling, L. P. (1993). Hierarchical learning in stochastic domains: Preliminary results. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 167–173). Amherst, MA.

Kinny, D., Georgeff, M., & Hendler, J. (1992). Experiments in optimal sensing for situated agents. *Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence*. Seoul,Korea.

Kuipers, B. (1985). Commonsense reasoning about causality: Deriving behavior from structure. In D. G. Bobrow (Ed.), *Qualitative reasoning about physical systems*. Cambridge, MA: MIT Press.

Laird, J. E., & Rosenbloom, P. S. (1990). Integrating execution, planning, and learning in Soar for external environments. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 1022–1029). Boston: AAAI Press.

Langley, P. (1987). A general theory of discrimination learning. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press.

Langley, P., Iba, W., & Shrager, J. (1994). Reactive and automatic behavior in plan execution. *Proceedings of the Second International Conference on AI Planning Systems* (pp. 299–304). Chicago: AAAI Press.

Minton, S. N. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, *42*, 363–391.

Nilsson, N. J. (1994). Teleoreactive programs for agent control. *Journal of Artificial Intelligence Research*, *1*, 139–158.

Nordhausen, B., & Langley, P. (1993). An integrated framework for empirical discovery. *Machine Learning*, *12*, 17–47.

Nguyen, D., & Widrow, B. (1989). The truck backer-upper: An example of self-learning in neural networks. *Proceedings of the International Joint Conference on Neural Networks*. Washington.

Sammut, C., Hurst, S., Kedzier, D., & Michie, D. (1992). Learning to fly. *Proceedings of the Ninth International Conference on Machine Learning* (pp. 385–393). Aberdeen, Scotland: Morgan Kaufmann.

Selfridge, O. G., Sutton, R. S., & Barto, A. G. (1985). *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 670–672). Los Angeles: Morgan Kaufmann.

Tan, M. (1991). *Cost-sensitive robot learning*. Doctoral dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Doctoral dissertation, Department of Psychology, Cambridge University, Cambridge, UK.