In *Connection Science (Special Issue on Developmental Robotics)*,
18(2): 97-119, 2006.

Towards Autonomous Sensor and Actuator Model Induction on a Mobile Robot

Daniel Stronger
Department of Computer Sciences
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712-1188
stronger@cs.utexas.edu
http://www.cs.utexas.edu/~stronger

Peter Stone
Department of Computer Sciences
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712-1188
phone: +1 512 471-9796
fax: +1 512 471-8885
pstone@cs.utexas.edu
http://www.cs.utexas.edu/~pstone

April 13, 2006

Abstract

This article presents a novel methodology for a robot to autonomously induce models of its actions and sensors called ASAMI (Autonomous Sensor and Actuator Model Induction). While previous approaches to model learning rely on an independent source of training data, we show how a robot can induce action and sensor models without any well-calibrated feedback. Specifically, the only inputs to the ASAMI learning process are the data the robot would naturally have access to: its raw sensations and knowledge of its own action selections. From the perspective of developmental robotics, our robot's goal is to obtain self-consistent internal models, rather than to perform any externally defined tasks. Furthermore, the target function of each model-learning process comes from within the system, namely the most current version of another internal system model. Concretely realizing this model-learning methodology presents a number of challenges, and we introduce a broad class of settings in which solutions to these challenges are presented. ASAMI is fully implemented and tested, and empirical results validate our approach in a robotic testbed domain using a Sony Aibo ERS-7 robot.

Keywords: autonomous agents, mobile robots, automated modeling

1 Introduction

A broad goal of intelligent robotics is to enable robots to behave autonomously in a wide variety of situations. One popular approach towards achieving this goal is to imbue the robot with a general reasoning capability that relies on 1) a *model* of the current state of the world; and 2) a model of the effects of the robot's actions on the world. Given these models, the robot can then plan its actions so as to best achieve its goals given the current state of the world.[1]

For such a model-based approach to be effective, the sensor and actuator models must be accurate and well-calibrated, at least in relation to one another. For example, if a trash-collection robot "sees" with its camera a discarded bottle that extends across one tenth of the camera's image plane, the robot's sensor model may lead it to conclude that the bottle is 3 meters away. Its actuator model may then provide information regarding how long it will take the robot to traverse the 3 meters, from which it can decide whether or not to collect the bottle. Similarly, state-of-the-art robot localization algorithms such as particle filtering [2, 3] rely on calibrated sensor and actuator (odometry) models to fuse sensory and action history information into a unified probabilistic estimate of the robot's location.

Currently, these sensor and actuator models are typically calibrated manually: sensor readings are correlated with actual measured distances to objects, and robot actuator commands are measured with a stopwatch and a tape measure. However this type of approach has three significant drawbacks. First, it is labor intensive, requiring a human operator to take the necessary measurements. Second, for sensors and actuators with many (perhaps infinitely many) possible readings or parameter settings, the measured model can only be made to coarsely approximate the complete model. Third, and perhaps most importantly from our perspective, the model is necessarily tuned to a specific environment and may not apply more generally.

This *brittleness* is a major motivation for the automatic model building advocated in this article. There are many different circumstances that can cause a robot's sensors and actuators to change in performance over time. First, if the robot's physical parts wear down over time, this can change how its actions or sensations correspond to the state of the world. Furthermore, if a robot moves into new environments, the different settings can correspond to different sensor and actuator behaviors.

[1] An alternate approach is to treat the robot control problem as a direct mapping from sensors to actuators without any intermediate model [1].

Some examples of this change include a locomotion module that goes at different speeds on different terrains or a vision module that sees colors differently in different lighting conditions. The ability to automatically learn its sensor and action models would dramatically increase the applicability of an autonomous robot.

When considered in isolation, there is no choice but to build each individual sensor and action model manually. In practice, however, each of the robot's sensors as well as its action selection mechanism can be related through their reflection on the world state, as illustrated in Figure 1. The arrow from a sensor or action to information about the state of the world represents the robot's model of that sensor or action. These models enable the robot to convert its raw sensory inputs and action selections into useful information about the world state. These different sources of information can easily be quite redundant, affording the robot a valuable opportunity to learn about the meanings of each of its sensors and actions from the values of others.

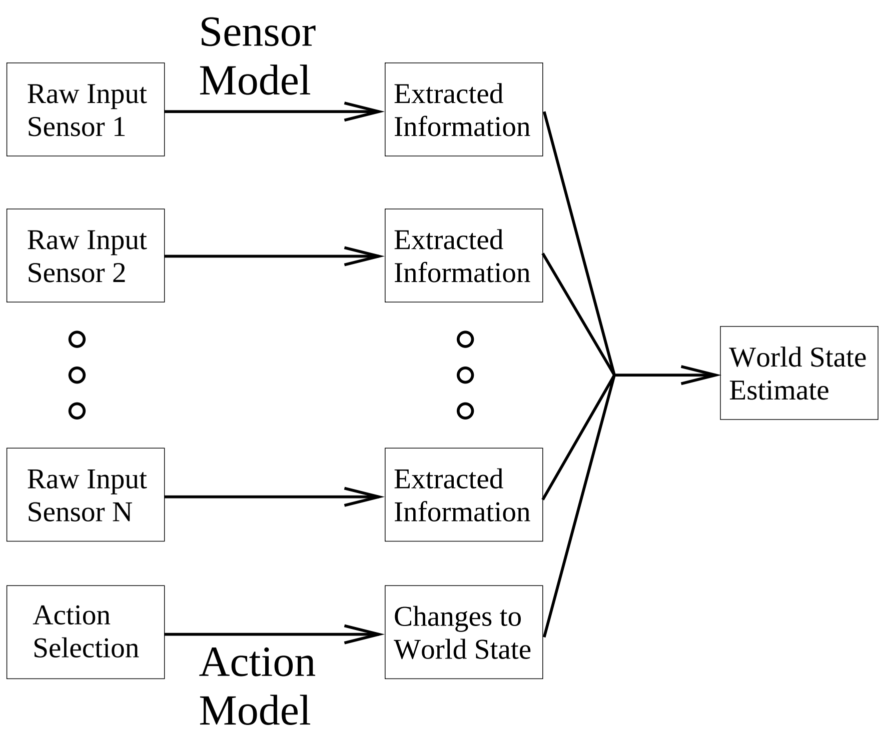


Figure 1: The flow of information on an autonomous robot. The data from each sensor and the action selections are interpreted based on the robot's action and sensor models, represented by arrows here. The resulting *Extracted Information* can then be used to inform the robot's estimate of the state of the world.

Because the world state can be estimated from independent, redundant sources of information, the robot can use this information to learn a model of any given individual source. It learns each model by comparing the input to that model to an estimate of the world state based on all of the other information sources. Figure 2 shows (in a simplified setting) how the robot can use redundant information to learn its action and sensor models. For the sensor model, the world state estimate is first relayed back through arrow A to the “information about world state” from the sensor model. This tells us what the output of the sensor model should have been assuming the world state estimate is perfectly accurate. When this data is combined with the raw sensory input via arrow B, the result is training data for learning the sensor model. This data can be processed by supervised learning method based on the structure of the sensor model. Similarly, the world state estimate can be relayed back to the “changes to world state” from the action model (arrow C). In this case, the world state, if accurate, indicates how the world actually changed as a result of previous actions. This information can be combined with the action selections (arrow D) to train the action model.

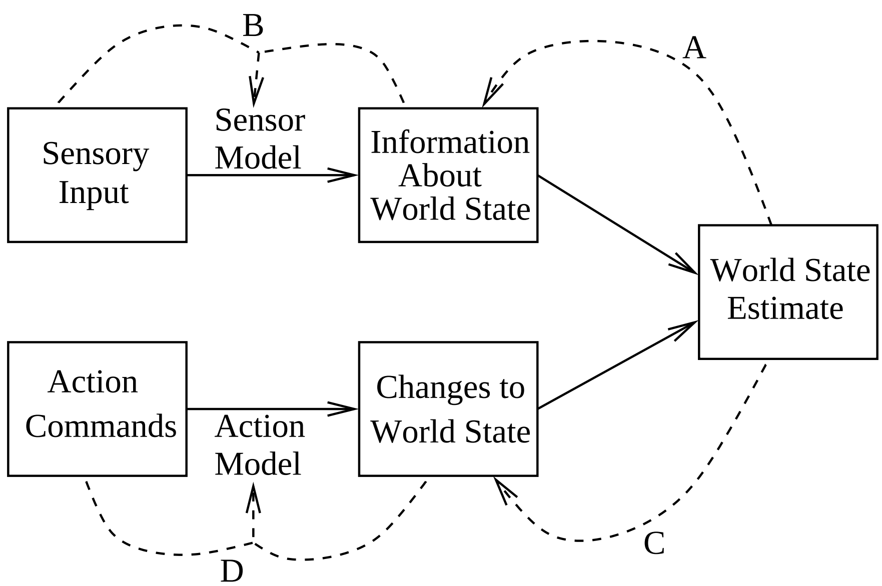


Figure 2: Dashed arrows A through D show how information can be propagated back from a redundantly informed world state estimate to calibrate the action and sensor models.

Another sense in which our approach combats brittleness is that, if successful, redundant sensors may be correlated enough to take over functionality from one another in the case that one sensor fails. For example, a robot with a visual distance sensor and a laser range finder could learn the relationship between the two sensors. Then, if the robot moves into a setting where one of the sensors is inapplicable, for example a dark room, the other sensor can provide the missing information.

Motivated by the observation of redundant world state information, this article proposes a general methodology by which a robot can learn its action and sensor models *from each other*. The benefit of this approach is that it enables a robot to induce models of its sensors and actions without any manually labeled training data. That is, the only inputs to the learning process are the data the robot would naturally have access to: its raw sensations and its knowledge of its own action selections. This general methodology promises to increase the robustness and general applicability of autonomous robots. If a fully featured robot could automatically calibrate all of its processing modules, it would be able to navigate throughout a variety of environmental conditions without any human supervision. However, achieving this goal is not straightforward. Standard techniques for model induction require accurately labeled training data; in the absence of manually labeled data, the robot must combine information from its actuators and sensors to bootstrap accurate estimates of both models. The resulting challenges are addressed throughout the remainder of this article.

In addition to introducing this general methodology, this article contributes a fully-implemented realization of it in a setting defined by one sensor and one actuator. Without any human intervention, the robot is able to build its own sensor and actuator models based on just a few minutes of experience in the world. Though there is not enough information to ground out these models to human units (e.g. meters and meters/second), the learned models are internally consistent, which arguably is all that is necessary for full autonomous behavior on the part of the robot.

From the perspective of *developmental robotics* [4, 5], our approach moves away from focusing on any individual task, instead concentrating on the robot's ability to build and maintain self-consistent internal models that can eventually be useful for a wide variety of autonomous behaviors. Furthermore, the target function of each model-learning process comes from within the system, namely from the most current version of another internal system model.

The remainder of this article is organized as follows. Section 2 introduces the setting in which we apply our methodology. Section 3 shows how an action or sensor model can be learned in the presence of a different, accurate model. Section 4 shows how to use use this ability to bootstrap accurate estimates of both models starting with accurate estimates of neither. Section 5 instantiates the technique on a robotic platform and evaluates the method's effectiveness with empirical tests. Finally, Section 6 discusses related work and Section 7 concludes.

2 Overview

There are many settings in which the methodology motivated in Section 1 can apply. In this article, we focus on one representative class of such settings that enables us to address the challenges set forth in Section 1. In this context, we call our approach ASAMI, for *Autonomous Sensor and Actuator Model Induction*[2]. The class of settings is defined by the following properties:

- The set of possible states of the world can be characterized as a one-dimensional, continuous state space.
- The agent has one sensor that converts the state of the world into numerical input data.
- The agent has at its disposal a continuum of actions that it can take. Each action maps onto a single rate of change in the state of the world over time.
- The effects of the actions and the sensor readings are perturbed by zero-mean, random noise.

Even in this somewhat restricted setting, there are many interesting domains that satisfy these conditions. One straightforward example is a robot on a one-dimensional track that has a global positioning sensor and takes actions that control its velocity along the track. Another example is a temperature regulator that can sense the temperature in a room and set the rate at which the temperature changes. Yet another example is where the velocity of the vehicle is the state and the agent has access to a velocity sensor and a throttle whose settings correspond to accelerations. In this article, we use an approximation of the first setting (a robot on a track) as a running example and for our empirical results.

ASAMI's goal is to learn a model of the agent's sensor and actions. In this context, a model of the sensor consists of a function from sensor readings to the actual state of the world. The action model is a function from the selected action to the rate of change of the state of the world.

[2] ASAMI is a more general formulation of SCASM, introduced in previous work [6].

ASAMI learns both models at once by having the agent simultaneously perform the following three operations.

1. Exploring the state space of the world, covering the entire range of relevant states and state velocities.[3]

2. Learning a function from action commands to actual state velocities, assuming the sensor model is accurately calibrated.

3. Learning a function from sensor readings to the actual state of the world, assuming the action model is accurately calibrated.

For operation 1, any action policy that allows the agent to experience the full range of possible combinations of states and velocities will suffice. The following sections describe in detail how to perform operations 2 and 3, first individually, and then simultaneously via a bootstrapping process.

3 Learning the Action and Sensor Models

This section defines the action and sensor models precisely and provides a framework for learning these models from one another. It then shows how each model can be learned by assuming the other one is already accurate. The remaining question of how the agent can learn both models simultaneously is addressed in Section 4.

3.1 Notation and Setup

As the agent interacts with its domain, we denote the true state of the world at time t as $W(t)$. During this process, the agent has two sources of information about its location along its axis of movement. For one, the agent receives a sequence of sensor observations, the kth one denoted by obs_k and occurring at time t_k. In general, a sensor model can be represented as $obs_k = F(W(t_k))$, where the function F specifies the sensation corresponding to any given world state. There is additionally a zero-mean random offset due to noise inherent in the sensor. In this article, we restrict our attention to invertible sensors, so that each value reported by the sensor corresponds

[3]We use the term *velocity* to mean the rate of change of the state, including such things as acceleration and rate of temperature change in the previous examples.

to a specific state of the domain. The inverse function, F^{-1}, is then well-defined. We denote it by S and refer to it as the sensor model, so that on average $W(t_k) = S(obs_k)$. This function S is one of the two functions that the agent is trying to learn. This relationship is depicted in Figure 3.

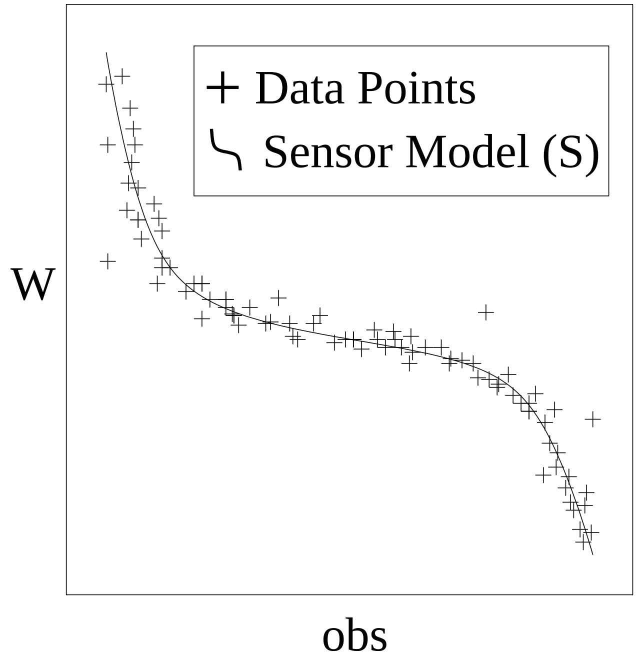


Figure 3: The sensor model S relates the sensory observations obs_k to the world state $W(t_k)$ through the equation $W(t_k) = S(obs_k)$.

At the same time, the agent continuously executes an action command, $C(t)$, that varies with time. Each action command changes the state of the domain at a specific velocity, and we denote the function from command to velocity by A. This function is the action model that the agent learns along with the sensor model S. The action model also provides information about the state of the world: $W(t) = W(0) + \int_0^t A(C(s))\,ds$ (again offset by random zero-mean noise). For example, if the agent executes a piecewise constant series of actions, the world state will vary in a continuous, piecewise linear manner with respect to time. This scenario is depicted in Figure 4. ASAMI works by implicitly performing a continual comparison between the information from the action and sensor models. Specifically, since they are used to calculate the same location function $W(t)$, we can set them equal, yielding

$$S(obs_k) = W(0) + \int_0^t A(C(s))\,ds$$

Although both sides of the equation are perturbed by random noise, because the noise is unbiased, the equation is true *in expectation*, and can therefore be used to learn the action and sensor models. The agent knows the values of obs_k, t_k, and $C(t)$, and its task is to learn the functions A and S.

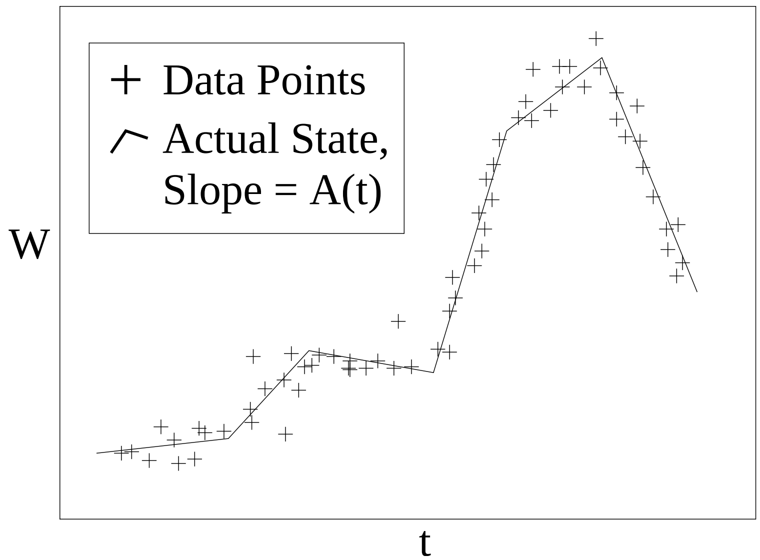


Figure 4: The rate of change of the state of the world is depicted by the slopes of the lines here. It is equal to $A(C(t))$, where $C(t)$ is the action command being executed at time t and A is the action model. The data points here represent the agent's estimates of the world state over time.

Because the agent is trying to learn two arbitrary continuous functions, it must represent them with a function approximator. We use polynomial regression for both functions. From among the many function approximator systems that exist, we chose to use polynomial regression for both functions due to its robustness, versatility, and simplicity. Specifically, it can filter out random noise, closely approximate any continuous function, and incorporate new data points efficiently. Initially, the degrees of the polynomials for the sensor and action models are chosen manually, although Section 5.4 discusses the possibility of further automating the process by choosing the degree itself on-line.

For learning the sensor model, the agent's goal is to identify coefficients s_0 through s_d such that the polynomial $\sum_{i=0}^{d} s_i(obs)^i$ approximates the actual $S(obs)$ as closely as possible, where d is the degree of the polynomial being fit to the data. Similarly, the agent learns coefficients a_0 through a_d for the action model, with the goal of $\sum_{i=0}^{d} a_i c^i \approx A(c)$ over the range of commands c.

ASAMI learns the action and sensor models *from each other* in that it does not take as input any ground truth as to the actual world state or its rate of change. Therefore, it cannot learn the two models in any particular units. The sensor model maps observations onto points on a linear axis, but it makes no claims as to what particular state corresponds to the number zero, or what magnitude in the domain's single dimension corresponds to the model's unit. For instance, if the domain is a temperature, the model's unit could correspond to any fixed number of degrees. Similarly, the action model is learned in arbitrarily units, although here the number zero is constrained to correspond to a zero rate of change.

Despite this lack of grounding to actual units, ASAMI ensures that the learned action and sensor models are consistent with one another. For any given time unit, the agent can represent a learned velocity as an amount of state change per unit time. Then whatever turns out to be the unit for the learned distances, that amount per unit time is the unit for the learned rates of change. Note that this property is sufficient for it to perform domain-specific planning, for instance by predicting the amount of time a specific action command will take to yield a certain visual sensor reading. In other words, we enable the agent to learn models in its own natural frame of reference, rather than imposing upon it human units such as meters or meters per second.

3.2 Learning the Sensor Model

First we demonstrate that it is possible to learn a relative sensor model given any *constant* action, even in the absence of any knowledge about that action. We will then generalize to the case of varying actions, assuming access to an accurate action model. Section 4 shows how this ability can be incorporated into a process that can learn both models from scratch.

Consider the situation in which the agent executes a constant action. Recall that the sensor model is a function from the observations, obs_k, to the corresponding world states, $W(t_k)$. Furthermore, while the agent executes a constant action command, c, the state changes at a constant rate, $A(c)$. Thus if this command is executed continuously starting at time 0, the state of the world at time t_k will be given by $W(t_k) = W(0) + t_k A(c)$. As long as the constant $A(c)$ is not zero, solving for t_k yields $t_k = (W(t_k) - W(0))/A(c)$. This expression represents a shifted and scaled version of the world state at time t_k. Furthermore, since ASAMI is only trying to learn a sensor model up to shifting and scaling, it suffices to learn a function directly from obs_k to t_k. Such a function will represent a relative sensor model. Thus ASAMI can learn a satisfactory sensor model even without knowing the constant rate of change $A(c)$. The agent learns the function by performing polynomial regression on the pairs (obs_k, t_k).

In general, polynomial regression is performed on n data points (x_i, y_i) (in this case $x_i = obs_i$ and $y_i = t_i$). ASAMI computes the coefficients of the best fit d-degree polynomial, $P(x) = \alpha + \sum_{j=1}^{d} \beta_j x^j$, which minimizes the total squared error between $P(x_i)$ and the corresponding y_i over all i from 1 to n. To compute α and $\beta = [\beta_1, \cdots, \beta_d]^\top$, we reformulate the problem as a multivariable linear regression by representing each of the powers of x from x^1 to x^d with its own variable $V_j = x^j$. The input data is then an $n \times d$ matrix V given by $V_{i,j} = x_i^j$ and the n-dimensional output vector is y. To perform this regression, first we define M and Y to be versions of V and y where the variables are centered around zero. That is, their means are subtracted from their values: $M_{i,j} = V_{i,j} - \overline{V_j}$ and $Y_i = y_i - \overline{y}$. Then α and β are given by [7], [8]

$$\beta = (M^\top M)^{-1}(M^\top Y) \qquad \text{and} \qquad \alpha = \frac{1}{n}\sum_{i=1}^{n}(y - V\beta)_i \tag{1}$$

Fortunately, it is not necessary to store V and y explicitly and compute these quantities from them each time. To save space and time as arbitrarily many data points come in, ASAMI incrementally maintains a number of sums that require constant storage space in the number of data points. For example, $(M^\top M)_{i,j}$ evaluates to

$$\sum_k V_{k,i} V_{k,j} - \frac{1}{n}(\sum_k V_{k,i})(\sum_k V_{k,j}) \tag{2}$$

The agent maintains these sums (for all i and j from 1 to d) incrementally, along with $\sum_k V_{k,i} y_k$ and $\sum_k y_k$. They enable it to compute $M^\top M$ as above, $M^\top Y$, and then β and α. This computation, when applied to the data (obs_k, t_k), identifies a suitable sensor model under the restrictive assumption that the agent is executing a constant action command.

With access to an accurate action model, it is possible to use a very similar process to learn a sensor model while the agent performs an arbitrary series of actions. Specifically, since the sensor model is a function from the observations to the world state, if the agent can compute a world state estimate from its actions, it can perform polynomial regression on its observations and those state estimates to learn a sensor model. Given an action model A, the agent can use its knowledge of the state velocities to compute the world state as a function of time. As presented in Section 3.1, the world state $W(t)$ is given by $W(0) + \int_0^t A(C(s))\, ds$, which we denote by $W_a(t)$. Since it is learning relative distances, it suffices to assume that $W(0) = 0$. Thus the agent can accumulate an estimate for $W(t)$ by initializing W to be 0 at time 0 and continually incrementing it by $A(C(t))\Delta t$, where Δt is the amount of time between increments. Then, by performing polynomial regression on the pairs $(obs_k, W_a(t_k))$, the agent effectively learns a sensor model from the action model. This regression is done as described above, except that now $x_i = obs_i$ and $y_i = W_a(t_i)$.

3.3 Learning the Action Model

In the previous section, we showed how ASAMI can learn a sensor model when given an accurate action model. In this section, we show that the reverse is also possible: we assume that the agent has an accurate sensor model and show how the agent can use it to learn an action model. This learning uses the sensor model to provide an estimate of the state of the world from each observation. We denote this estimate by $W_s(t_k)$, and it is given by $S(obs_k)$. For learning the action model, the training data consists of the state estimates $W_s(t_k)$, combined with the knowledge of the action selections $C(t)$. Since the action model maps the action selections to the *rate of change* of the world state, rather than directly to the state itself, the training data does not allow for a direct polynomial regression as in the previous section. Intuitively, the task is to learn a function from the action command to the slope of the world state function, or dW/dt (see Figure 4).

More precisely, the agent's goal is to learn the function $A(c) = \sum_{i=0}^{d} a_i c^i$ that causes the values of $W(t_k)$ based on A to match those based on S as closely as possible. That is, the agent computes the coefficients a_i that minimize the error defined by

$$\begin{aligned} E &= \sum_{k=1}^{n} \left[W_s(t_k) - \left(W(0) + \int_0^{t_k} \sum_{i=0}^{d} a_i C(s)^i \, ds \right) \right]^2 \\ &= \sum_{k=1}^{n} \left[W_s(t_k) - \left(W(0) + \sum_{i=0}^{d} a_i \int_0^{t_k} C(s)^i \, ds \right) \right]^2, \end{aligned} \tag{3}$$

where the agent knows the values obs_k, $W_s(t_k)$, and the values of $C(s)$. This problem is an instance of a multivariable linear regression, with $d+1$ variables V_1 through V_{d+1} defined as $V_j = \int_0^t C(s)^{j-1} \, ds$ and output $y_k = W_s(t_k)$. The regression computes the coefficients a_i (and a value for $W(0)$) that minimize the error. This regression has the effect of identifying the curve that fits the data $(t_k, W_s(t_k))$ as closely as possible, provided that the slope of the line at any time t is a constant d-degree function of $(C(t))$. Figure 4 shows what this curve might look like in the case where the action selection policy is piecewise constant.

4 Learning Both Models Simultaneously

We have so far demonstrated the ability for the agent to learn the sensor model from the action model and vice versa. Making use of both of these capabilities, this section shows how the agent can simultaneously learn both models, even when it is given very little useful starting information. This learning is possible because, even though the action (sensor) model learned from an inaccurate sensor (action) model will be inaccurate, it will be an improvement. As each model grows more accurate, its ability to help the other model improve grows. As this bootstrapping process continues, the two models converge to functions that accurately reflect what they are trying to model.

Because both models grow in accuracy as time goes on, the regressions should give more weight to the more recent data points. Thus a weighted regression is used, where each data point has a weight that decreases over time. Note that for both learning directions, there is one regression data point for each visual sensor observation. We define the weight of each data point to start at one and decrease by a constant factor $\gamma < 1$ every time a new observation is taken. Thus if there have been n observations so far, the weight of the data points corresponding to the ith one is γ^{n-i}.

To compute the solution to the weighted regression, we define the following variables. The diagonal $n \times n$ weight matrix D is given by $D_{i,i} = \gamma^{n-i}$, and N is defined as the sum of the weights $\sum_{i=1}^{n} \gamma^{n-i}$. The matrices V, y, M and Y are defined just as before (see Section 3.2), except that now M and Y are recentered with respect to the *weighted* averages of the variables. Then we can use a weighted version of Equation 1 [8]:

$$\beta = (M^\top DM)^{-1}(M^\top DY) \quad \text{and} \quad \alpha = \frac{1}{N}\sum_{i=1}^{n} \gamma^{n-i}(y - V\beta)_i \tag{4}$$

As in Section 3.2, these quantities are represented in terms of sums. For example $(M^\top DM)_{i,j}$ is given by

$$\sum_k \gamma^{n-k} V_{k,i} V_{k,j} - \frac{1}{N}(\sum_k \gamma^{n-k} V_{k,i})(\sum_k \gamma^{n-k} V_{k,j}) \tag{5}$$

These sums can also be maintained incrementally, because $\sum_{k=1}^{n+1} \gamma^{(n+1)-k} z_k = \gamma(\sum_{k=1}^{n} \gamma^{n-k} z_k) + z_{n+1}$ for any sequence z_k.

At time t, the agent makes use of its best estimates thus far of the action and sensor models, A_t and S_t. Throughout the learning, the agent maintains two estimates of the world state, one based on its current sensor model, $W_s(t)$, and the other based primarily on its action model, $W_a(t)$. After any observation obs_k at time t_k, $W_s(t_k)$ is given by $S_t(obs_k)$. At the same time, $W_a(t)$ is maintained by continually incrementing it by $A_t(C(t))\Delta t$, where Δt is the amount of time between increments and $A_t(C(t))$ is ASAMI's current estimate of the state velocity.

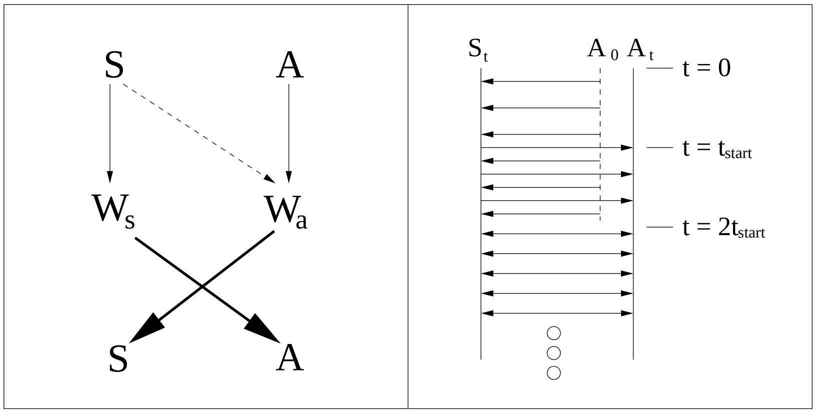
The update of $W_a(t)$ ensures that the derivative of $W_a(t)$ is the agent's best estimation of the state velocity. However, by itself this constraint allows for the possibility that there is a large, persistent difference between the estimates $W_a(t)$ and $W_s(t)$. It is necessary to avoid such a difference because it would have the following adverse effect. Because the sensor model S is trained based on the values of W_a, the estimates $W_s(t)$ would gradually drift towards the $W_a(t)$. Then, when the action model A was learned from the estimates $W_s(t)$, the drift would be interpreted as state velocity, which would in turn cause the values of $W_a(t)$ to drift in the same direction. This would cause the difference between the estimates to persist while both drifted in the same direction continually. With both state estimates drifting, ASAMI would be unable to converge on accurate estimates of the action and sensor models.[4]

To avoid this problem, a mechanism is included that constrains W_a and W_s to come into closer agreement with each other over the course of the learning. The mechanism adjusts $W_a(t)$ towards $W_s(t)$ every time an observation is taken. The adjustment is implemented by the assignment $W_a(t_k) \leftarrow (1-\lambda)W_a(t_k) + \lambda W_s(t_k)$, where λ is a constant that determines the strength with which $W_a(t)$ is pulled towards $W_s(t)$.

Figure 5a) depicts the resulting flow of information in ASAMI. The model estimates S_t and A_t are continually updated in accordance with the location estimates $W_a(t)$ and $W_s(t)$, with each model being updated by the location estimate based on the other model. These incremental updates comprise the weighted polynomial regressions that give the best fit estimates of S and A, as described above.

At the start of the training, there is no data to ground either the action model or the sensor

[4] In initial tests in the experimental environment to be described in Section 5, we indeed observed such a divergence of $W_s(t)$ and $W_a(t)$.



a) b)

Figure 5: a) The flow of information. The thick arrows represent incorporating a data point into the weighted regression for a model. The thin arrows indicate that each model is used to construct the corresponding estimate of the world state. The dashed arrow signifies S's influence on the estimate W_a as described in the text. b) The ramping up process. The arrows indicate one model being learned based on another. Note that aside from A_0, a model is not learned from until it has been learned for a sufficient amount of time.

model, raising the challenge of how to get the learning process started. If the learning process starts with uninitialized models, it is likely to lead to meaningless, possibly deceptive models. To address this challenge, for a period of time at the beginning, the agent uses a fixed, pre-set action model, A_0, instead of A_t. The function used for A_0 can be a very rough approximation of the true action model; it does not have to be particularly accurate, as Section 5.4 will show. During this time, the sensor model is learned based on A_0, but the action model is not learned yet, because the sensor model is based on too few data points. The amount of time taken to initialize a model is denoted by t_{start}. After this time has passed, the sensor model can be used to start learning an action model. However, until another period of time has passed, this new action model is not based on enough data points for it to be used for learning. We set this second period of time to be the same length as the first for the sake of parsimony. After these two periods of length t_{start}, the action and sensor models can learn from each other. This process is depicted in Figure 5b).

Pseudocode for the entire algorithm is given in Algorithm 1. The routine UPDATE incorporates one new data point into the weighted regression for the model being updated.

Algorithm 1 Pseudocode for ASAMI.

$W_a(t) \leftarrow 0$
for each time step **do**
  if $t < 2t_{start}$ **then**
    $W_a(t) \leftarrow W_a(t) + A_0(C(t))\Delta t$
  else
    $W_a(t) \leftarrow W_a(t) + A_t(C(t))\Delta t$
  end if
  if an observation obs_k is made **then**
    if $t > t_{start}$ **then**
      $W_s(t) \leftarrow S_t(obs_k)$
      UPDATE A_t with $(t, W_s(t))$
      $W_a(t) \leftarrow (1 - \lambda)W_a(t) + \lambda W_s(t)$
    end if
    UPDATE S_t with $(t, W_a(t))$
  end if
end for

The goal of this process is for the models S and A to grow in accuracy over time, so that their corresponding estimates of the state of the world eventually come into agreement with one another, and with the actual state of the world. The following section presents empirical evidence that these goals are met in our test domain.

5 Empirical Validation

This section presents an implementation of ASAMI on a robotic platform. The robotic domain is designed to meet the four characteristics described in Section 2. In particular, the agent navigates through a one-dimensional state space, its sensor readings map onto possible states, its actions map onto rates of change in that same state space, and its sensations and actions are corrupted by zero-mean random noise. To realize these characteristics, we task a robot with estimating its distance from a visible landmark while moving towards and away from it. Experimental results bear out that, in this domain, ASAMI successfully learns accurate models of the robot's sensor and actions.

The remainder of this section introduces the robotic testbed domain used to validate ASAMI and presents empirical results demonstrating the success of ASAMI in that domain. Section 5.1 describes the testbed domain in detail. Section 5.2 demonstrates ASAMI's ability to learn one model from the other, as developed in Sections 3.2 and 3.3, in the robotic domain. Section 5.3 shows how ASAMI enables the robot to learn both models simultaneously, as described in Section 4. Finally, Section 5.4 presents some additional experimental results that illustrate ASAMI's robustness and flexibility.

5.1 Testbed Domain

ASAMI is implemented and tested on a commercially available robot platform, namely the Sony Aibo ERS-7.[5] In our domain, the state of the world is defined to be the robot's distance to a fixed landmark. The sensor model is based on the robot's visual sensor and maps the size of the fixed landmark in the robot's image plane to the robot's distance from that landmark. The Aibo and the landmark, a colored cylindrical beacon, are shown in Figure 6, along with a view of the beacon taken through the Aibo's camera. The actuator model maps an energy parameter (like "throttle" on a car) to the actual speed of the robot. The remainder of this section describes the testbed domain used to validate our methods.

The results reported in this article make use of walking and vision processing modules that we created earlier as part of a larger project [9]. The walk is defined by a number of parameters that specify the attempted trajectories of the Aibo's feet. To move forwards and backwards at different speeds, the robot interpolates between parameters for an idle walk $\mathbf{p_i}$ that steps in place, a fast forwards walk $\mathbf{p_f}$ that goes at a speed of v_{max} (335 mm/s), and a fast backwards walk $\mathbf{p_b}$ that goes at a speed of v_{min} (−280 mm/s). The action commands are labeled by a desired velocity r and have parameters given by:

$$\mathbf{p_r} = \begin{cases} \mathbf{p_i} + \frac{r}{v_{max}}(\mathbf{p_f} - \mathbf{p_i}) & \text{if } r \geq 0 \\ \mathbf{p_i} + \frac{r}{v_{min}}(\mathbf{p_b} - \mathbf{p_i}) & \text{if } r < 0 \end{cases} \tag{6}$$

Note that Equation 6 is based on the assumption that the Aibo's velocity is linear in its walking parameters. However, it turns out not to be linear, as the experimental results will bear out. We take the robot's actual velocity to be an unknown, arbitrary function of the desired velocity. ASAMI

[5] http://www.aibo.com

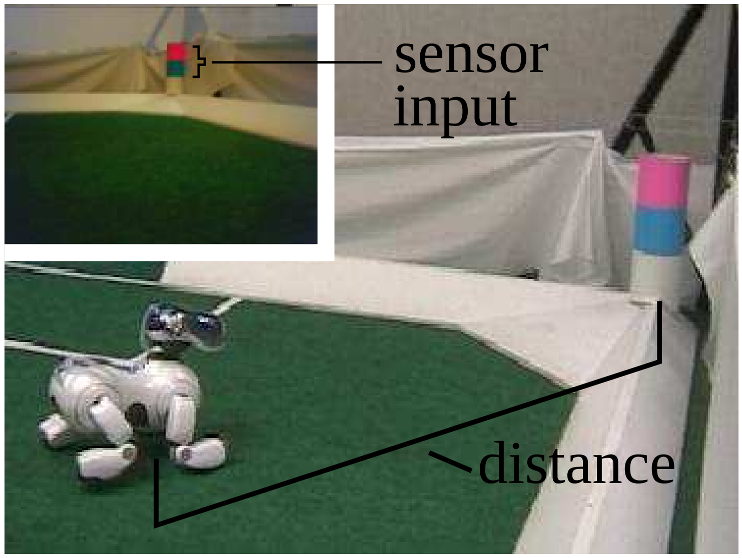


Figure 6: The Aibo and the beacon. The inset is a picture of the beacon taken through the Aibo's camera.

learns this function from the robot's action commands to their corresponding velocities. For the rough approximation of the action model, A_0, which is used to start off the learning process (see Section 4), we use the identity function, so that $A_0(c) = c$. The sensitivity of ASAMI to this choice is explored in Section 5.4.

The Aibo's visual sensor is based on its camera, which we have previously trained to recognize objects in the robot's environment. One of these objects is a colored cylindrical beacon, shown in Figure 6, that the robot can use to help it localize while on a playing field. The height of the beacon in the robot's image plane decreases with the robot's distance from the beacon; this observed height (in pixels) is the visual sensor reading used for the experiments reported in this paper. Detailed descriptions of the walking and vision modules used here are given in [9].

As discussed in Section 2, ASAMI works by simultaneously performing three operations. Operations 2 and 3 consist of learning the action and sensor models. Operation 1 is for the agent to act so that it experiences the full range of relevant action commands and observations. To achieve this goal, the Aibo walks alternatingly forwards and backwards across a pre-set range of distances from the beacon. For the experiments reported in this paper, the robot's goal is to learn about the action commands in the range $[-300, 300]$. Hence, the robot chooses a random action command in the range $[0, 300]$ while going forwards and from $[-300, 0]$ during the backwards phase. It continues to execute each action for three seconds before choosing a new one. It switches between walking forwards and backwards when the beacon height in the image gets too big or too small. These size thresholds are chosen manually so as to keep the robot in its field of operation. This behavior covers the full range of relevant distances and velocities, as desired.

Although the action commands being executed only attempt to move forwards and backwards, random drift would cause the Aibo to slowly get off course. To counteract this effect, the walking controller is set to constantly turn towards the beacon with an angular velocity proportional to the beacon's horizontal angular distance from straight ahead. This small angular velocity has a negligible impact on the robot's forwards or backwards velocity. A video of the Aibo performing the training behavior described here is available online.[6]

5.2 Learning One Model at a Time

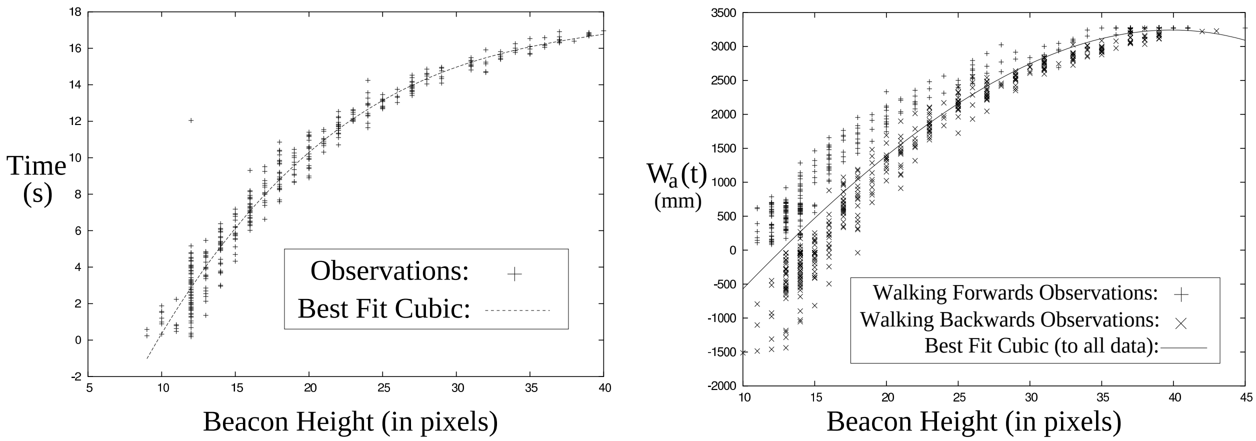
This section demonstrates that ASAMI is able to learn the sensor and action models, as described in Sections 3.2 and 3.3, in our robotic test domain. The experiments in this section show qualitatively how the different parts of ASAMI work, in isolation, on the robotic platform. The following section demonstrates ASAMI's ability to learn both models simultaneously and provides quantitative experimental results.

The sensor and action models are learned as polynomials of degree three and four respectively, based on the estimation (without detailed experimentation) that these are roughly the polynomial degrees necessary to capture the complexity of the functions being modeled. The possibility of having the agent autonomously select the polynomial degrees is discussed in section 5.4.

First we show that the robot can learn a sensor model while executing a constant action. This learning entails applying polynomial regression to the pairs (obs_k, t_k) with $d = 3$ while a constant action command is being executed. When this process is performed, the cubic learned is typically quite an accurate fit to the data, as shown in Figure 7a).

Second, the robot must also be able to learn a sensor model while executing any arbitrary series of actions, such as the random one described in Section 5.1. In this case, we assume that the robot already has access to an accurate action model. Then it can use the action model to

[6] http://www.cs.utexas.edu/~AustinVilla/?p=research/simultaneous_calibration



a) b)

Figure 7: a) After walking forwards via a constant action, these are the observed data points (+), mapped against time. The dashed curve is the best fit cubic to these points. The variation in beacon height at any given time is due to inherent noise in vision. b) The plotted points are $(obs_k, W_a(t_k))$ as the robot performs one full cycle of walking towards the beacon and backing away from it. The +'s are the observations while walking forwards and the $\times$'s are while walking backwards. The polynomial is fit to all the points.

compute its velocity at any time, and use its velocity to accumulate an estimate of its location: $W_a(t) = W(0) + \int_0^t A(C(s))\, ds$. To test this hypothesis, we provide the robot with a starting action model estimate, and train the sensor model based on the resultant world state estimates, using a third-degree polynomial regression. The result of such a regression is shown in Figure 7b). In that graph, the vertical axis represents the location estimates based on the action model and the selected action commands. Note that because the action model used here is actually not perfectly accurate, the estimates taken while walking forwards and backwards are not well aligned with one another. Nonetheless, the learned sensor model is still a qualitatively reasonable one, in that as the beacon height increases, the rate of change of the corresponding location decreases, as would be expected.

Recall from Section 3.3 that ASAMI can also learn an action model if given an accurate sensor model. To verify this ability in our test domain, the robot first learns a rough sensor model using the constant action method described above. It uses that sensor model to produce a running estimate of the robot's location, $W_s(t)$. As discussed in Section 3.3, this data can be combined with the knowledge of the executed actions to identify the action model that minimizes the disagreement between the location estimates based on the sensor and action models. Equation 3 is used to convert this minimization problem into a multivariable linear regression. Performing the regression yields ASAMI's action model estimate. For simplicity, the robot's unit of time is set equal to one second. The result of this process is shown in Figure 8.

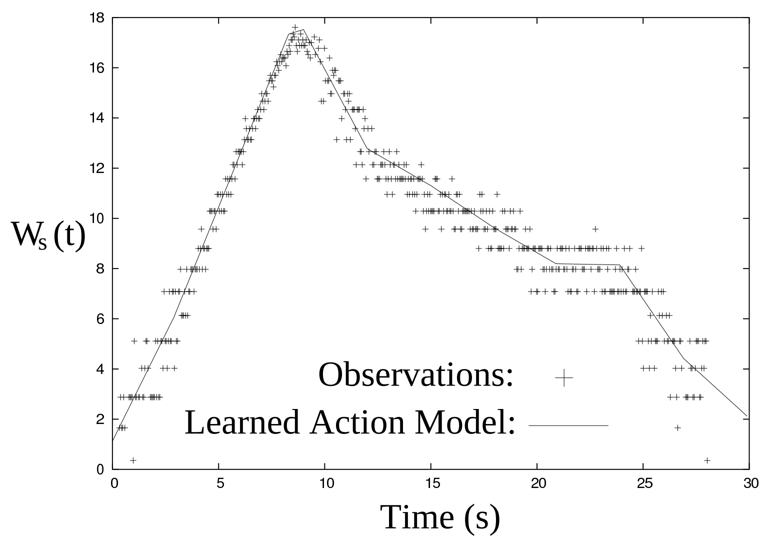


Figure 8: The plotted points are $(t_k, W_s(t_k))$ as the robot performs one full cycle of walking towards the beacon and backing away from it. The learned action model is applied to the executed action commands to yield the piecewise linear location estimate shown here. Note that the units in the vertical axis of this graph are arbitrary, since it is based on the learned relative sensor model, W_s.

5.3 Learning Both Models Simultaneously

This section demonstrates the robot's ability to learn the sensor and action models simultaneously using the technique described in Section 4. The learned sensor and action models are evaluated by comparison to models measured manually. The experimental results show that ASAMI's learned models closely match the measured models.

Implementing ASAMI on a specific platform requires finding suitable values for a few constants.

Finding these values did not require any extensive tuning. The discount factor we used for the regression weights, γ, is 0.999. The strength of the pull of W_a towards W_s, λ, is 1/30. These values were the first ones that were tried for γ and λ. The starting phase time, t_{start}, is 20 seconds. We tried 10 seconds first but that was too short.

When the models S and A are learned simultaneously, Figure 9 depicts how $W_s(t)$ and $W_a(t)$ vary over time. Note that both oscillate with the robot's walking towards and away from the beacon. As A and S grow more accurate, their corresponding estimates of the location come into stronger agreement.

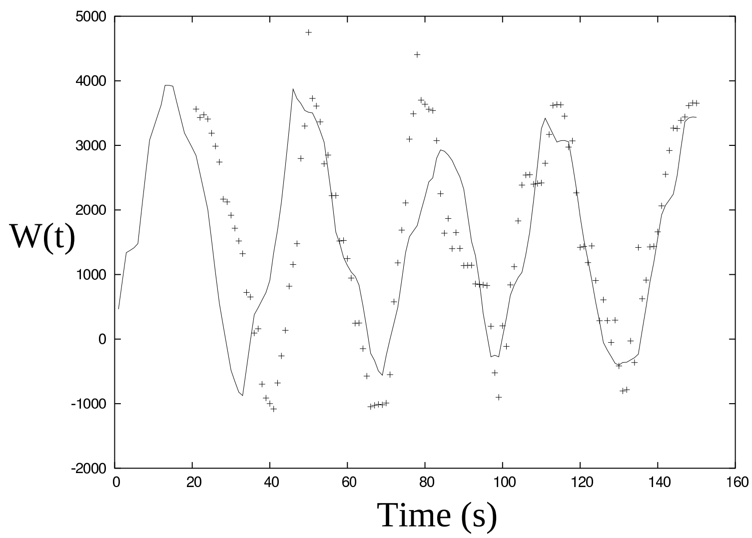


Figure 9: This figure shows how $W_a(t)$, and $W_s(t)$ vary over time. In this example run, the +'s are values of $W_s(t)$, and the curve depicts $W_a(t)$. Over time, each model learns how to keep its estimate of the location close to the other model's estimate.

The processing required to execute ASAMI consists of two regression computations every time the robot processes an image, corresponding to about 20 Hertz. This process happens concurrently with all of the robot's other real-time computation, including vision and motion processing, all on-board on a single 576 MHz processor.

After ASAMI has run for a pre-set amount of time (two and a half minutes), we consider its best estimates for A and S to be the models that it has learned at that point. ASAMI learns the action and sensor models starting without an accurate model of either, so in order to evaluate the learned models, we need an accurate account of the actual models to compare to the learned ones. For comparison purposes, we performed manual measurement of the robot's actual action and sensor models. These measurements were performed with a stopwatch and a tape measure. The measured action model is obtained by measuring the velocity of each action command that is a multiple of 20 from -300 to 300. We measure the velocity of an action command by timing it across a distance of 4.2 meters[7] five times. The standard deviation of the velocity measurement for a given action command across the five timings never exceeded 7 mm/s. The measured action model is shown in Figure 10a).

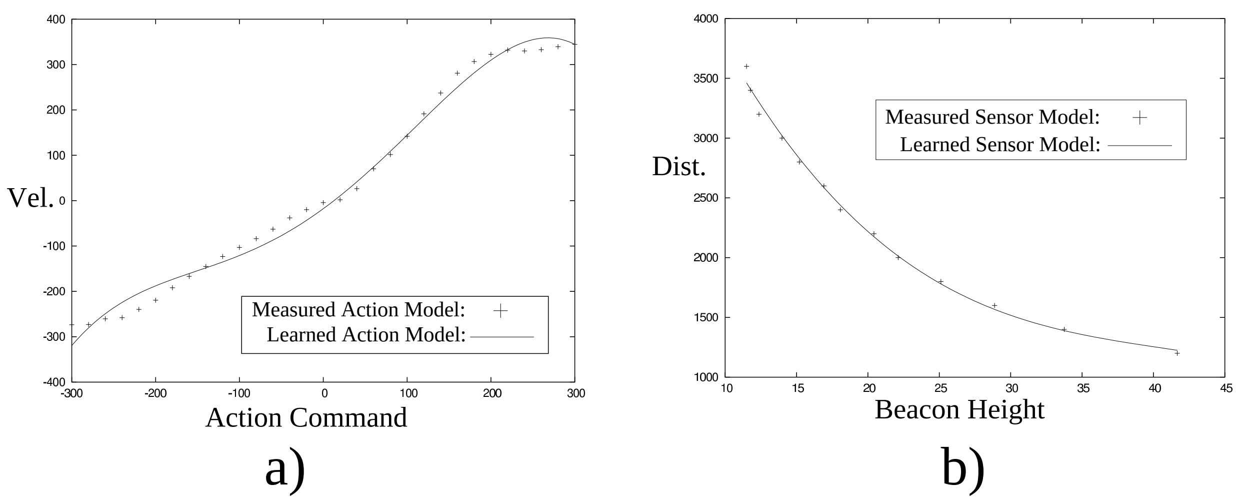


Figure 10: A learned action and sensor model

Similarly, the accuracy of the learned sensor model is gauged by comparing it to a measured sensor model. The sensor model is measured by having the Aibo stand at measured distances from the beacon. The distances used were the multiples of 20 cm from 120 cm to 360 cm. At each distance, the robot looked at the beacon until it had collected 100 beacon height measurements. The average of these measurements was used as a data point for the sensor model, and their standard deviation did not exceed 1.1 pixels at any distance. The measured sensor model is shown in Figure 10b).

The learning process was executed 15 times, with each trial lasting for two and a half minutes. Figure 10a) shows a typical learned action model, compared to the measured action model data.

[7] For a few very slow action commands, shorter distances were used.

Note that since the action model is not learned in any specific units, in order to compare the learned model to the measured one, we must first determine the appropriate (linear) y-axis scaling factor. This evaluation is done by calculating the scaling factor that minimizes the mean squared error. On average, the root mean square error between the scaled learned action model and the measured action model was 29.6 ± 12.4 mm/s. Compared to the velocity range of 600 mm/s, the error is 4.9 percent. The best fit possible by a fourth degree polynomial to the measured action model has an error of 17.2 mm/s. By contrast, when the the initial action model, A_0, is evaluated in the same manner, the error is 43.0 mm/s.

Figure 10b) shows a typical learned sensor model with the measured sensor model. The learned model S maps observations to relative distances, $S(obs)$, which are intended to model the actual distances from the beacon. These actual distances are given by $a + bS(obs)$, where a and b are two constants that are not learned. Thus in order to evaluate a learned sensor model, we compute the values of a and b that minimize the mean squared error between $a + bS(obs)$ and the measured sensor model. This minimization is done with a linear regression on the points $(S(obs_i), S_m(obs_i))$, where the obs_i are the sensor readings corresponding to the measured distances $S_m(obs_i)$. Our evaluation of a learned sensor model is the root mean square error between it and the measured model, once this process has been applied. This value was, on average, 70.4 ± 13.9 mm. Compared to the distance range of 2400 mm, the error is 2.9 percent. The best fit possible by a cubic to the measured sensor model has an error 48.8 mm.

Over the course of a trial, both models get progressively more accurate. The learning curves are depicted in Figure 11. Both models' errors are shown, compared to the best possible error for the measured model and the degree of the polynomial being learned. The data is averaged over all 15 trials.

Although the action and sensor models are not learned to any particular scale, since they are learned from each other they should be to the same scale. This property is tested by comparing the scaling constants used to give the best fits to the measured models, the scaling constant for the action model and b for the sensor model. These two values should be equal to each other in absolute value. We evaluate the degree of equality by computing the average distance between the absolute value of the ratio between the two scaling constants and 1. The average distance is 0.08 ± 0.06. This result shows that the two learned models are generally consistent with each other.

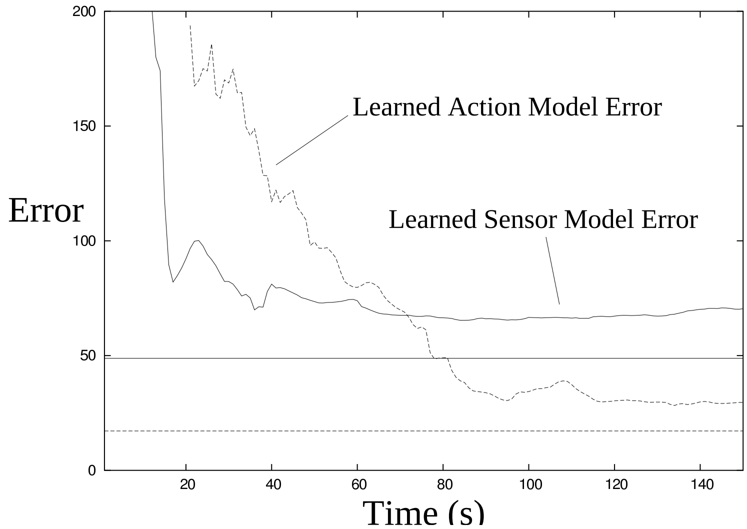


Figure 11: This figure depicts the average error in the learned models as a function of time. The error for the action model is in mm/s, and for the sensor model in mm. The horizontal lines are at the minimum possible error to the measured models for a polynomial of the appropriate degree.

The amount of time taken by ASAMI to accurately learn its action and sensor models in this domain is two and a half minutes. Note that for a fixed environment, ASAMI only needs to be executed once. Considering that each time the robot is booted up, it takes about 27 seconds to initialize, we consider ASAMI's one-time execution time to be qualitatively quite short. Certainly, it is within the bounds of what can be reasonably executed on-line upon insertion into a new environment.

5.4 Additional Results

In this section, we examine the impacts of two assumptions used to this point: that there is a reasonable initial action model, and that the degrees of the polynomials used for regression are chosen manually. For the initial action model, we find that it can convey very little information and still be sufficient to get the learning started. In fact, even when the initial action model conveys no useful information, ASAMI can frequently learn accurate action and sensor models. Furthermore, we discuss the challenge of choosing the polynomial degree autonomously and present some preliminary results in this regard.

Initial sensor model. Recall from Section 4 that an initial action model A_0 is used for an amount of time t_{start} to seed the learning. This is the only information that ASAMI starts with about either the action or sensor model. The results described above use a linear initial action model, $A(c) = c$, that is somewhat similar to the measured action model (shown in Figure 10). To examine the reliance of ASAMI on this initial action model, we performed two tests with more impoverished starting points. First, we used a piecewise constant model equal to $\mathrm{Sgn}(c)$: 1 for positive action commands and -1 for negative ones. This model conveys only the direction of the action but no information about its speed. In 15 runs, the robot was able to achieve an average error of 85.3 ± 24.5 mm in its learned sensor model and 31.3 ± 9.2 mm/s in the action model after two and a half minutes. These errors are comparable to those attained with the linear model (70.4 ± 13.9 mm and 29.6 ± 12.4 mm/s for the sensor and action models respectively).

Even with a starting model of $A(c) = 1$, which imparts no information about the action model, on 10 out of 15 trials the robot was able to achieve an average performance of 88.6 ± 11.5 mm error in the sensor model and 27.3 ± 6.2 mm/s in the action model after five minutes. The remaining trials diverged, presumably due to initially learning a pair of models that were so inaccurate that no useful information could be recovered from them. The results from the three different starting conditions are presented in Table 1. Note that the errors achieved with these more impoverished models are similar to those achieved with the linear model, indicating that our results are not particularly sensitive to the choice of the starting action model.

$\mathbf{A_0}$	**Sensor Model (mm)**	**Action Model (mm/s)**	**Success Rate**
$A_0(c) = c$	70.4 ± 13.9	29.6 ± 12.4	15/15
$A_0(c) = \mathrm{Sgn}(c)$	85.3 ± 24.5	31.3 ± 9.2	15/15
$A_0(c) = 1$	88.6 ± 11.5	27.3 ± 6.2	10/15

Table 1: For each of three initial action models tried, this table shows the average fidelity of the learned sensor and action models. The last column shows on how many of the 15 trials ASAMI successfully learned sensor and action models.

Polynomial degree selection. A potential enhancement to ASAMI would be to enable it to choose the degrees for the polynomial regressions automatically. The remainder of this section presents some preliminary results towards that goal.

Choosing the degree of the polynomial is a type of model selection, the problem of identifying the best parameters for a function approximator. There are many popular model selection techniques, such as the Akaike Information Criterion [10], the Bayes Information Criterion [11], and cross-validation [12]. For choosing the degree of a polynomial for regression in the context of ASAMI, we tried a method that takes advantage of the increasing expressive power of the successive possible polynomial degrees and the wealth of data typically available in a robotic setting.

To choose the degree, we start by fitting a first degree (linear) polynomial and continually monitoring the fit to see if the degree needs to be increased. If so, the regression is restarted with the degree incremented by one. The degree continues to be incremented until a satisfactory fit is found. In order to determine whether or not a particular degree is satisfactory, the robot compares a *global prediction error* and a *local noise estimate*, two values that are continually maintained. A high global prediction error indicates a poor fit, suggesting that the polynomial degree should perhaps be increased. However, such an error might also be accounted for by a large amount of random noise in the observed data, in which case increasing the degree will not help. This comparison finds the lowest degree polynomial that achieves a satisfactory fit, which has the effect of implicitly balancing the higher computational costs of higher degrees against their improved accuracy.

Because of the added complications in learning an action model (see Section 3.3), we tested this method only on learning a sensor model. To obtain training data for the sensor model while the robot executed the random behavior described in Section 5.1, we used a fixed action model that is the best fit fourth degree polynomial to the measured data. The estimates of the robot's location based on this action model were used as training data for the sensor model learning with automated degree finding.

Fifteen trials were run, each lasting five minutes. In each trial, the degree stabilized within two and a half minutes and did not increase after that time. The fact that the robot was able to settle on a degree every time demonstrates the method's stability. Nevertheless, randomness in the training data caused some variation in the final polynomial degree. The average degree chosen was 3.33, with a standard deviation of 1.29. This degree corroborates our earlier estimate that a third degree polynomial was roughly the amount of complexity needed to learn the sensor model.

The learned sensor models are evaluated as described in Section 5.3. The average errors for the learned sensor models was 101 ± 34 mm. Compared to the distance range of 2400 mm, the average error is 4.2 percent. For comparison, with the manually chosen fixed degree, the average sensor model in Section 5.3 was 70.4 ± 13.9 mm. This experiment demonstrates the feasibility of enabling ASAMI to choose the degrees for its polynomial regressions fully autonomously.

6 Discussion and Related Work

In addition to the related work described throughout the earlier portions of this article, this section discusses ASAMI in the context of developmental robotics and provides an overview of previous work related to the novel contributions of this article.

ASAMI is motivated by the ideals of developmental robotics [4, 5], an approach to artificial intelligence in which a robot autonomously learns about itself and its environment via a general exploration process, as opposed to a task-specific learning algorithm. Specifically, the robot should learn concepts *on its own terms*, without being restricted to seeing the world exactly the same way that humans do. Much recent work has been done in developmental robotics, including a system for language acquisition [13], intrinsically motivated learning systems [14, 15, 16], and developmental approaches to interacting with objects [17, 18, 19].

ASAMI accomplishes the goals of developmental robotics in two major ways. First, the only inputs to the algorithm are the robot's sensations and action selections; there is no externally generated training data. Second, the robot learns mappings between sensations and distances, and between actions and velocities, but the distances and velocities are not constrained to be in human units (such as meters or meters per second). The only constraint is that the two models are consistent *with each other*.

One previous example of a robot learning about its sensors and effectors from scratch is given by Pierce and Kuipers [20]. They present a technique by which a robot can learn a hierarchical model of its world. At the lowest level, this process identifies useful relationships between the robot's various sensors and effectors. Olsson et al. [21] extend that work by enabling a Sony Aibo to learn the spatial relationships between its camera elements, and subsequently to perform motion flow computations on the resulting sensor structure. It is then able to associate different action commands with corresponding motion flow observations, so that the robot can track a moving image. ASAMI differs from these approaches in that it uses a function approximator to learn the relationships between relevant variables. This property allows ASAMI to accurately represent a diverse range of possible action and sensor models in a one-dimensional domain.

There is a wide range of previous work in modeling robotic sensors. One common type of sensor modeling is camera calibration. Cameras have both intrinsic parameters, such as focal length and distortion factors, and extrinsic parameters, such as the camera's location and orientation. Tsai [22] presents a general method for calibrating all of these parameters by a combination of geometric analysis, linear regression, and nonlinear optimization. This method relies on labeled training data in the form of coordinates of points in the world and their corresponding coordinates in the image plane. By contrast, ASAMI learns action and sensor models without any labeled training data.

Another common type of sensor modeling involves calibrating networks of sensors. This work typically focuses on networks with large numbers of sensors and determining their respective locations and orientations. For example, Ihler et al. [23] present a method for a network of sensors to localize themselves with respect to each other efficiently via a belief propagation method and Savvides et al. [24] use a distributed technique to identify the relative locations of the sensors. We know of no previous work calibrating a sensor model based on an action model.

Action modeling has been done in a number of different domains. In terms of modeling the actuators themselves, the authors [25] have previously developed a mathematical model of a robot joint to enable more precise control. Regarding odometry calibration, some previous work has focused on mobile robots calibrating their odometry models automatically based on their sensors. On wheeled robots, Roy and Thrun [26] calibrate the odometry using an incremental maximum likelihood method, while Martinelli et al. [27] and Larsen et al. [28] use an augmented Kalman filter to estimate odometry errors. On a legged robot, Quinlan et al. [29] discuss a method for calibrating the odometry based on the robot's vision-based localization. These methods all rely on already calibrated sensor models to make the action models more accurate.

ASAMI differs from all of this previous work in the following significant way. ASAMI learns models of its actions and sensors starting without an accurate model of either. To the best of our knowledge, all previous approaches to calibration rely either on accurate training data or on sensors that are already well calibrated.

Looking forward, we consider ASAMI to be a first step towards enabling a robot to autonomously navigate through a very high-dimensional space while learning many functions with various numbers of input and output variables. Extending ASAMI in this direction raises two main challenges. First, ASAMI currently allows a robot to learn two functions: one sensor model and one action model. With more than two models to learn, the robot will have *multiple* sources of information that can be used to train each model. This situation raises the question of how these sources can best be integrated with each other to provide effective training data for each model. Second, ASAMI currently assumes that the robot operates in a one-dimensional state space, thus restricting its learned models to being from one variable to one other variable. For many applications, it will be necessary or useful to learn functions with multiple input and output variables. In this case, straightforward polynomial regression is insufficient. In principle, the ASAMI methodology should extend to other general-purpose function approximators such as neural networks [30] and tile coding (also known as CMACs) [31]. However, additional research is needed to determine how these methods may need to be adapted for automated model learning. For example, incorporating a new function approximator into ASAMI's framework would require adapting it to be able to learn velocities from a series of state estimates, as is necessary for learning the action model (see Section 3.3).

As another direction of future work, ASAMI currently maintains two estimates of the agent's world state: one based on the action model and one based on the sensor model. Instead of keeping these estimates separate, one possibility would be to continually combine the two sources of information via a state estimation technique such as Kalman filtering [32, 33] or Monte Carlo particle filtering [2, 3]. These state estimation techniques can be seen as complementary to ASAMI; the possibility of combining such a state estimation method with ASAMI is a promising avenue for future research.

7 Conclusion

This article introduces a methodology by which a mobile robot can simultaneously learn an action model and a sensor model, from each other. The methodology is instantiated in a broad class of settings, namely those in which an agent navigates through a one-dimensional state space with a sensor that provides information about the state of the world and actions that determine the world state's rate of change. Examples of such settings include a robot on a track with a global positioning sensor and a velocity control; a temperature regulator; and a vehicle with a throttle whose settings correspond to accelerations. The resulting technique, ASAMI, enables the agent to autonomously induce models of its sensor and actions. ASAMI works by simultaneously learning an action model and a sensor model, each one based on the current estimate of the other one. This bootstrapping process enables the agent to learn accurate approximations to its true action and sensor models, starting with only a very simplistic action model estimate. Furthermore, the learning process is completely autonomous and unsupervised, so that no human oversight or feedback is necessary.

ASAMI is implemented and validated in a robotic test-bed domain based on the Sony Aibo ERS-7. In experimental tests, the robot learns models from its action commands to the resultant velocities and from its visual sensor readings to the corresponding distances. The learning process takes only two and a half minutes of completely autonomous behavior. Overall, the work presented here represents an exciting start towards the long-term challenge of enabling fully autonomous calibration of complex, multi-modal, and multi-dimensional sensor and action models on mobile robots.

Acknowledgments

The authors thank Ben Kuipers for helpful discussions. Thanks also to the members of the UT Austin Villa team for their efforts in developing the software used as a basis for the work reported in this paper. This research was supported in part by NSF CAREER award IIS-0237699, ONR YIP award N00014-04-1-0545, and DARPA grant HR0011-04-1-0035.

References

[1] R. A. Brooks, "New approaches to robotics," *Science*, vol. 253, pp. 1227–1232, 1991.

[2] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[3] C. Kwok, D. Fox, and M. Meila, "Adaptive real-time particle filters for robot localization," in *Proc. of the IEEE International Conference on Robotics & Automation*, 2003.

[4] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen, "Autonomous mental development by robots and animals," *Science*, vol. 291, pp. 599–600, 2001.

[5] D. Blank, D. Kumar, and L. Meeden, "A developmental approach to intelligence," in *Proceedings of the Thirteenth Annual Midwest Artificial Intelligence and Cognitive Society Conference*, S. J. Conlon, Ed., 2002.

[6] D. Stronger and P. Stone, "Simultaneous calibration of action and sensor models on a mobile robot," in *IEEE International Conference on Robotics and Automation*, April 2005.

[7] R. F. Gunst and R. L. Mason, *Regression Analysis and its Application.* New York: Marcel Dekker, Inc., 1980.

[8] S. Weisberg, *Applied Linear Regression.* New York: John Wiley & Sons, Inc., 1980.

[9] P. Stone, K. Dresner, P. Fidelman, N. K. Jong, N. Kohl, G. Kuhlmann, M. Sridharan, and D. Stronger, "The UT Austin Villa 2004 RoboCup four-legged team: Coming of age," The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, Tech. Rep. UT-AI-TR-04-313, October 2004.

[10] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Proc. Second International Symposium on Information Theory*, Budapest, 1973.

[11] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[12] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning.* New York: Springer-Verlag, 2001.

[13] S. Levinson, K. Squire, R. Lin, and M. McClain, "Automatic language acquisition by an autonomous robot," in *The AAAI Spring Symposium on Developmental Robotics*, March 2005.

[14] P. Dangauthier, P. Bessiere, and A. Spalanzani, "Auto-supervised learning in the Bayesian programming framework," in *The AAAI Spring Symposium on Developmental Robotics*, March 2005.

[15] P. Oudeyer, F. Kaplan, V. Hafner, and A. Whyte, "The playground environment: task-independent development of a curious robot," in *The AAAI Spring Symposium on Developmental Robotics*, March 2005.

[16] A. Stout, G. Konidaris, and A. Barto, "Intrinsically motivated reinforcement learning: A promising framework for developmental robot learning," in *The AAAI Spring Symposium on Developmental Robotics*, March 2005.

[17] J. Modayil and B. Kuipers, "Bootstrap learning for object discovery," in *IEEE International Conference on Intelligent Robots and Systems (IROS-04)*, 2004, pp. 742–747.

[18] L. Natale, G. Metta, and G. Sandini, "A developmental approach to grasping," in *The AAAI Spring Symposium on Developmental Robotics*, March 2005.

[19] A. Stoytchev, "Toward learning the binding affordances of objects: A behavior-grounded approach," in *The AAAI Spring Symposium on Developmental Robotics*, March 2005.

[20] D. Pierce and B. Kuipers, "Map learning with uninterpreted sensors and effectors," *Artificial Intelligence*, vol. 92, pp. 169–229, 1997.

[21] L. Olsson, C. Nehaniv, and D. Polani, "From unknown sensors and actuators to actions grounded in sensorimotor perceptions," *Connection Science*, vol. 18, no. 2, 2006.

[22] R. Tsai, "An efficient and accurate camera calibration technique for 3d machine vision," in *IEEE CVPR 1986*, 1986.

[23] A. T. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-calibration in sensor networks," in *Proceedings of the third international symposium on Information processing in sensor networks*, Berkeley, CA, April 2004.

[24] A. Savvides, C. C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc wireless sensor networks," in *Proceedings of the International Conference on Mobile Computing and Networking*, July 2001.

[25] D. Stronger and P. Stone, "A model-based approach to robot joint control," in *RoboCup-2004: Robot Soccer World Cup VIII*, D. Nardi, M. Riedmiller, and C. Sammut, Eds. Berlin: Springer Verlag, 2005, pp. 297–309.

[26] N. Roy and S. Thrun, "Online self-calibration for mobile robots," in *Proceeding of the IEEE International Conference on Robotics and Automation*, vol. 3. Detroit, MI: IEEE Computer Society Press, May 1999, pp. 2292–2297.

[27] A. Martinelli, N. Tomatis, A. Tapus, and R. Siegwart, "Simultaneous localization and odometry calibration for mobile robot," in *Proceedings of the 2003 International Confrerence on Intelligent Robots and Systems*, Las Vegas, NV, October 2003.

[28] T. D. Larsen, M. Bak, N. Andersen, and O. Ravn, "Location estimation for an autonomously guided vehicle using an augmented Kalman filter to autocalibrate the odometry," in *FUSION98 Spie Conference*, Las Vegas, NV, July 1998.

[29] M. Quinlan, C. Murch, T. Moore, R. Middleton, L. Li, R. King, and S. Chalup, "The 2004 NUbots team report," 2004, http://robots.newcastle.edu.au/publications/NUbotFinalReport2004.pdf.

[30] S. Haykin, *Neural Networks*. Upper Saddle River, New Jersey: Prentice-Hall, Inc., 1999.

[31] J. S. Albus, *Brains, Behavior, and Robotics*. Peterborough, NH: Byte Books, 1981.

[32] E. Kalman, Rudolph, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[33] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina at Chapel Hill, Department of Computer Science, Tech. Rep. 95-041, 2004.