Data-Driven Identification of Group Dynamics for Motion Prediction and Control

Mac Schwager, Carrick Detweiler, and Iuliu Vasilescu
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139, USA
e-mail: schwager@mit.edu, carrick@mit.edu, iuliuv@mit.edu

Dean M. Anderson
USDA–ARS, Jornada Experimental Range
Las Cruces, New Mexico 88003, USA
e-mail: deanders@nmsu.edu

Daniela Rus
Computer Science and
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139, USA
e-mail: rus@csail.mit.edu

Received 19 December 2007; accepted 2 April 2008

A distributed model structure for representing groups of coupled dynamic agents is proposed, and the least-squares method is used for fitting model parameters based on measured position data. The difference equation model embodies a minimalist approach, incorporating only factors essential to the movement and interaction of physical bodies. The model combines effects from an agent's inertia, interactions between agents, and interactions between each agent and its environment. Global positioning system tracking data were collected in field experiments from a group of 3 cows and a group of 10 cows over the course of several days using custom-designed, head-mounted sensor boxes. These data are used with the least-squares method to fit the model to the cow groups. The modeling technique is shown to capture overall characteristics of the group as well as attributes of individual group members. Applications to livestock management are described, and the potential for surveillance, prediction, and control of various kinds of groups of dynamic agents are suggested. © 2008 Wiley Periodicals, Inc.

1. INTRODUCTION

We wish to model groups of interacting dynamic agents, such as flocks, swarms, and herds, using measured data from those agents. For example, we would like to use the trajectories of people in a crowd to develop dynamic models that capture the behaviors of the crowd as a whole. This is a prohibitively complicated problem in general; however, we provide a practical solution by restricting our attention to a special model structure. We embrace a minimalist approach in that we use only position measurements, with a minimum of prior environmental information incorporated into the model. We propose a difference equation model that is decentralized and nonlinear, though it is designed to be linear in parameters. The least-squares method is then used to fit model parameters to position data from a group of agents. Such a model may then be used, for example, to predict future states of the group, to determine individual roles of agents within the group (e.g., leaders vs. followers), or, ultimately, to control the group.

The most immediate application of these ideas is for virtual fencing of livestock (Anderson, 2007; Butler, Corke, Peterson, & Rus, 2006; Wark et al., 2007), in which physical fences are replaced with sensor/actuator devices mounted on the animals. The animals' positions are monitored, and if they stray beyond a virtual fence line, the animals are given cues to return to the desired area. Our modeling techniques will be useful for virtual fencing in several ways. First, our models lead to verified behavioral simulations that can be used to test virtual fencing algorithms in a simulation environment before they are implemented in a costly and time-consuming field test. Second, our dynamic models can be used to enhance the animal control algorithm itself, so that it works in conjunction with the animals' natural tendencies. Finally, because our model is inherently distributed and because our minimalist approach requires few computational resources, we envision that the model can run online over the same network of animal-mounted sensor devices that carry out the virtual fencing algorithm. The distributed model can then be used to predict where the group is headed and inform the controller in real time. Simultaneously, the model can be updated to fit the most recent position data collected from the animals. This simultaneous model learning and model-based control is in the spirit of adaptive control.

In addition to livestock management applications, there are many other uses for learned models of distributed dynamic systems. In the case of people, the ability to model group behavior has numerous applications in surveillance, urban planning, and crowd control. Also, the models can be used to drive groups of robots to mimic the behavior of observed groups. This may be useful in reproducing collaborative behaviors exhibited in natural systems, or in producing decoy robots to participate with natural or engineered groups, and even to influence group behavior (Halloy et al., 2007).

The problem of learning models for groups of interacting dynamic agents lies at the intersection of two fields of research: modeling of distributed dynamical systems and system identification. A vigorous body of work is emerging from the controls and robotics communities focused on analyzing models of flocks, swarms, and similar distributed dynamic systems. This work, however, has not considered using learning techniques to generate these models from data. Instead, it concentrates on the dynamic properties of models, such as stability of formations (Gazi & Passino, 2003, 2004; Tanner, Jadbabaie, & Pappas, 2007; Tanner, Pappas, & Kumar, 2004; Zavlanos & Pappas, 2007), asymptotic consensus of agent positions or velocities (Cucker & Smale, 2007; Jadbabaie, Lin, & Morse, 2003; Olfati-Saber & Murray, 2004; Wang & Slotine, 2004), or designing local controllers from global specifications (Belta & Kumar, 2004; Ferraru-Trecate, Buffa, & Gati, 2006). These considerations are elemental in describing more complex social phenomena, but they are quite different from the question of learning models from data, which we address in this work.

Conversely, the rich literature on learning dynamic systems from data, often called system identification, has not yet addressed models of distributed dynamic systems, such as the ones we consider in this work. Some related problems have been considered, however. For example, in Correll and Martinoli (2006) a system identification technique is used to model global properties of a swarm of robots over time using observed data from the robots. These properties include collision likelihoods of robots and transition probabilities among robot behaviors. There also has been considerable activity in learning behavioral models of individual natural agents. In Oh, Rehg, Balch, and Dellaert (2005) and Pavlovic, Rehg, and MacCormick (2001), system identification is carried out on switching linear systems to learn models of the honey bee waggle dance and human hand motion, respectively, and in Delmotte, Egerstedt, and Austin (2004) a technique is used to find motion description language codes from observed ants. These works, however, do not consider group interactions but investigate the action of individuals isolated from their group roles.

It is our intention in this paper to bridge the gap between these two research communities by applying system identification techniques to distributed model structures. In addition to this cross-pollination of ideas, we also contribute a new technique for modeling general vector fields (i.e., nongradient vector fields) in a way that is amenable to system identification. We also pursue our ideas from theory through implementation by testing our method with data from natural agents.

For this purpose, we developed a hardware platform to record position and orientation information of groups of free-ranging cows. The hardware platform is capable of recording global positioning system (GPS) position information and head orientation and is able to provide sound and electrical stimuli, though no stimuli were administered during the data collection for this study. We demonstrate our model learning technique by fitting it to GPS data collected from a group of 3 and a group of 10 free-ranging cows and validate the resulting models by testing the whiteness of the residual error and by comparing global statistics of simulations with the actual data. Previous works have considered animal-mounted sensor network devices, such as the ZebraNet platform (Juang et al., 2002) and sensor/actuator devices (Anderson, 2007; Butler et al., 2006; Rutter, Champion, & Penning, 1997; Wark et al., 2007) for automatic livestock management. Our device has several innovations for applying animal control stimuli and for using communication between devices over a network; however, we do not describe these innovations in detail in this work. In the context of this work, the devices were used as a means to collect GPS data for learning and validating dynamic models.

The remainder of this paper is organized as follows. The model structure is described in Section 2. The application of system identification to identify model parameters is described in Section 3, along with a review of basic system identification techniques in Section 3.1. Our data collection device and experimental method are described in Section 4. Results of the system identification technique are presented in Section 5 with GPS tracking data from a group of 3 cows and a group of 10 cows, and the quality of the learned models is evaluated in Section 5.3. Finally, in Section 6 we use a learned model to control a group of mobile robots to behave like the group of three cows. Simulation results of the group of robots are presented. Concluding remarks and directions for future work are given in Section 7.

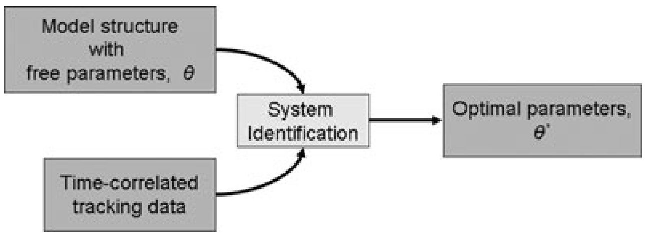
Model structure with free parameters, θ
System Identification
Optimal parameters, θ^*
Time-correlated tracking data

Figure 1. Schematic of the method of system identification. The time-correlated (not independent identically distributed) data and the model structure are combined in an optimization procedure to get model parameters tuned to fit the data.

2. MODEL DESCRIPTION

We consider a linear-in-parameters model structure with three naturally distinct parts to describe the motion of coupled physical agents moving over a plane surface. First, each agent is given internal dynamics to enforce the constrains of Newton's laws. Second, a force[1] is applied to each agent from its interaction with each of the other agents in the group. Third, a force is applied to each agent as a function of its position in the environment. All remaining effects are modeled as a white noise process.

Throughout this section, we refer to free parameters as θ, and features, or regressors, are denoted by ϕ. It should be understood that the parameters θ are left unknown for now. In Section 3 we describe how position data are used to tune these parameters to fit the data. A schematic showing the different parts of the model learning process is shown in Figure 1.

2.1. Individual Agent Dynamics

Given a group of m agents, the proposed model structure for an individual agent $i \in \{1, \ldots, m\}$ can be

[1]In this work the term "force" is used in a metaphoric sense. When we talk of a "force" we are referring to the intention of the agent to accelerate in a particular way using its own motive mechanisms.

written in state-space, difference equation form as

$$x_i^{\tau+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & a_i & 0 \\ 0 & 0 & 0 & a_i \end{bmatrix} x_i^\tau + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \times \left[\sum_{j=1, j\neq i}^{m} f_{ij}(p_i^\tau, p_j^\tau) + g_i(p_i^\tau) + w_i^\tau \right]. \quad (1)$$

Agent i's state $x_i^\tau = [e_i^\tau \; n_i^\tau \; u_i^\tau \; v_i^\tau]^T$ consists of its east position, north position, eastern component of velocity, and northern component of velocity after the τth iteration, and its position is given by $p_i^\tau = [e_i^\tau \; n_i^\tau]^T$. The time step Δt is given by $t^{\tau+1} - t^\tau$, and we assume that it is constant for all τ. The term a_i represents damping, $a_i = 1$ for zero damping and $|a_i| < 1$ for stable systems. The function $f_{ij}(p_i^\tau, p_j^\tau)$ determines the coupling force applied by agent j to agent i. The function $g_i(p_i^\tau)$ represents the force applied by the environment to the agent at point p_i^τ. Finally, w_i^τ is a zero-mean, stationary, Gaussian white noise process uncorrelated with $p_j \; \forall j$ used to model the unpredictable decision-motive processes of agent i. Nonholonomic constraints that are often present in mobile agents, such as people, cattle, and automobiles, are neglected in this treatment, though they could be incorporated with an increase in the complexity of the model structure. Note that the force terms are applied only to affect changes in velocity in accordance with Newton's second law.

2.2. Agent-to-Agent Interaction Force

Dropping the τ superscripts for clarity, the form of the agent coupling force $f_{ij}(p_i, p_j)$ is given by

$$f_{ij}(p_i, p_j) = \left(\theta_{1_{ij}} - \frac{\theta_{2ij}}{\|p_j - p_i\|}\right) \frac{n_{ij}}{(m-1)}, \quad (2)$$

where $n_{ij} = (p_j - p_i)/\|p_j - p_i\|$ is the unit vector along the line from p_i to p_j (henceforth, $\|\cdot\|$ will denote the ℓ^2 norm). The factor $(m-1)$ is included to normalize the force exerted by one neighbor by the total number of neighbors.

This is the simplest of a family of force laws commonly used in computational models of physical, multibody systems. The important feature of this family is that an agent is repulsed from its neighbor at small distances and attracted to its neighbor at large distances. To see this property clearly, examine the magnitude of force exerted by one neighbor ($m - 1 = 1$) given by $\|f_{ij}\| = \theta_{1_{ij}} - \theta_{2_{ij}}/\|p_j - p_i\|$ and shown in Figure 2(a). Notice that with $\theta_{1_{ij}} > 0$ and $\theta_{2_{ij}} > 0$ the desired characteristic is achieved. Indeed, as $\|p_j - p_i\| \to 0$, $\|f_{ij}\| \to -\infty$, which is repulsive, while as $\|p_j - p_i\| \to \infty$, $\|f_{ij}\| \to \theta_{1_{ij}} > 0$, which is attractive. Other, similar force laws can be created

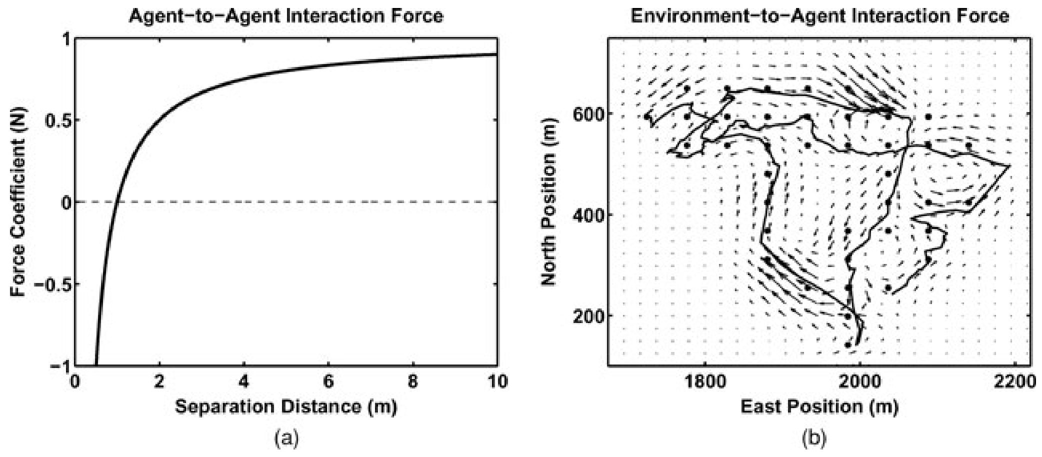

Figure 2. (a) Magnitude of the agent-to-agent interaction force for $\theta_1 = \theta_2 = 1$. (b) The vector field representing the force felt by an agent at each point on the plane shown for an example agent trajectory. The swirling patterns evident in the field are made possible by a novel parameterization.

to produce unbounded attraction as $\|p_j - p_i\| \to \infty$ and zero attraction as $\|p_j - p_i\| \to \infty$. We chose this law for its simplicity. The function can equivalently be expressed as the gradient of a potential function.

After some manipulation, the sum of f_{ij} over all neighbors j can be expressed as

$$\sum_{j \neq i} f_{ij} = \begin{bmatrix} \phi_{fu_i} \\ \phi_{fv_i} \end{bmatrix} \theta_{f_i}, \tag{3}$$

where

$$\phi_{fu_i} = \left[\frac{(e_1 - e_i)}{\|p_1 - p_i\|} \cdots \frac{(e_m - e_i)}{\|p_m - p_i\|} \frac{-(e_1 - e_i)}{\|p_1 - p_i\|^2} \cdots \right.$$
$$\left. \times \frac{-(e_m - e_i)}{\|p_m - p_i\|^2} \right] \frac{1}{(m-1)},$$
$$\phi_{fv_i} = \left[\frac{(n_1 - n_i)}{\|p_1 - p_i\|} \cdots \frac{(n_m - n_i)}{\|p_m - p_i\|} \frac{-(n_1 - n_i)}{\|p_1 - p_i\|^2} \cdots \right.$$
$$\left. \times \frac{-(n_m - n_i)}{\|p_m - p_i\|^2} \right] \frac{1}{(m-1)},$$
$$\theta_{f_i} = \left[\theta_{1_{i1}} \cdots \theta_{1_{im}} \; \theta_{2_{i1}} \cdots \theta_{2_{im}} \right]^T,$$

and where the indices $j = i$ are excluded from the above vectors (because we do not want an agent to feel a force from itself). This notation will be useful in what follows.

The agent-to-agent force law with the dynamics described above gives a so-called potential field–based flocking model, the analytical properties of which have been treated extensively in the controls and robotics literature (Gazi & Passino, 2003, 2004; Tanner et al., 2004, 2007; Zavlanos & Pappas, 2007). The environment-to-agent force described below makes our model rather different, however, and the inclusion of the noise term w_i^τ makes the model a random process, which is fundamentally different from the deterministic systems treated in those works.

2.3. Environment-to-Agent Interaction Force

The agent's preference for certain paths in the environment is modeled as a nonlinear mapping from each point on the plane to a force vector felt by the agent. To this end, two networks of Gaussian basis functions are used, one for each of two perpendicular force components.

In particular, the function $g_i(p_i)$ can be written

$$g_i(p_i) = \begin{bmatrix} \theta_{u_{i1}} & \cdots & \theta_{u_{in}} \\ \theta_{v_{i1}} & \cdots & \theta_{v_{in}} \end{bmatrix} \begin{bmatrix} \phi_{g_1}(p_i) \\ \vdots \\ \phi_{g_n}(p_i) \end{bmatrix}, \tag{4}$$

where

$$\phi_{g_{ik}}(p_i) = \frac{1}{2\pi\sigma_{ik}^2} \exp\left(-\frac{\|p_i - \gamma_{ik}\|^2}{2\sigma_{ik}^2} \right)$$

is the bivariate Gaussian function and $k \in \{1, \ldots, n\}$. Each Gaussian is centered at γ_{ik}, with standard deviation σ_{ik}, and its strength is represented by the unknown parameters $\theta_{u_{ik}}$ for the eastern component and $\theta_{v_{ik}}$ for the northern component. Gaussian basis functions were chosen for their familiarity, the objective being to demonstrate the modeling approach with a minimum of complications. A number of other basis function types could be used, including wavelets, sigmoidal functions, and splines.

It is important to note that a vector field parameterized in this way is *not* a potential gradient. A potential gradient field cannot admit circulation around closed paths.[2] We introduce a nongradient parameterization to enable circulation, as one can imagine agents intending to traverse closed orbits on the plane. For example, a cow may have a routine of passing between a water source, a shaded tree, and a grassy patch in a periodic fashion.

Figure 2(b) shows a plot of an example force field parameterized in the above way. The arrows show the forces induced by the field, the heavy dots show the centers of the Gaussian functions γ_{ik}, and the curve shows the path of an agent over the vector field. The swirling patterns evident in the vector field would be impossible if it were a gradient field.

The expression in Eq. (4) can be put into a different form to match that of Eq. (3). In particular,

$$g_i(p_i) = \begin{bmatrix} \phi_{gu_i} \\ \phi_{gv_i} \end{bmatrix} \theta_{g_i}, \tag{5}$$

[2]Proof: Let $\Psi(p)$ be a potential function and $V(p) = -\mathrm{grad}(\Psi)$ its gradient field. Then $\mathrm{curl}(V) = \mathrm{curl}(-\mathrm{grad}(\Psi)) = 0$; thus by Green's theorem, $\oint_s V ds = \int_{A_s} \mathrm{curl}(V) dA = 0$, where s is any closed curve on the plane and A_s is the area enclosed by s.

where

$$\phi_{gu_i} = [\,\phi_{g_1} \cdots \phi_{g_n} \cdots 0 \cdots\,],$$
$$\phi_{gv_i} = [\,\cdots 0 \cdots \phi_{g_1} \,\cdots\, \phi_{g_n}\,],$$
$$\theta_{g_i} = [\,\theta_{u_{i1}} \cdots \theta_{u_{in}} \theta_{v_{i1}} \cdots \theta_{v_{in}}\,]^T.$$

This form will become useful in what follows.

To consider the computational complexity of this model, consider that the number of agent-to-agent interaction terms grows as the square of the number of agents $O(m^2)$ and, in the worst case, the number of environment-to-agent interaction terms grows as the product of the time duration and the number of agents $O(Tm)$. Therefore computing successive iterations of the model, not to mention learning the model parameters, will become intractable as the size of the group approaches hundreds or thousands of members. However, we can alleviate these difficulties in a natural way. First, if the area in which the agents move is bounded, the environment-to-agent interaction terms will approach $O(m)$ as the entire area is explored by all the agents. Also, for large groups we could simply add a finite communication radius around each agent, so that neighbor agents outside that radius do not produce a force. This would limit the complexity of agent-to-agent parameters to $O(m)$. Thus we can modify the model to have an overall complexity linear in the number of agents. Also, the model is naturally decentralized, and thus it could easily be implemented on a network of, say, m processors, reducing the computation time to a constant independent of the size of the group. In this paper we do not consider such implementation issues, and the groups of agents we deal with are small enough that computation speed is not a concern.

3. SYSTEM IDENTIFICATION WITH LEAST-SQUARES FITTING

We will provide a brief introduction to the field of system identification. Then we will use a least-squares method to identify optimal parameters for our model. We will also discuss recursive methods for least-squares fitting that can be used to tune parameters for our model online as data are collected.

3.1. Method Overview

In this section we employ the tools of system identification (Ljung, 1999), the basics of which are briefly reviewed here as they may be unfamiliar to the reader. If a stochastic dynamic system is such that its state at the next time step is determined by its state at the current time step and the inputs at the current time step, a state-space model of its dynamics can be formed as a difference equation:

$$x^{\tau+1} = F(x^\tau, u^\tau, w^\tau, \tau),$$

where x is the state, u is the input, w is a zero mean, stationary, Gaussian white noise process, and τ is the discrete time index. Furthermore, we may formulate a model structure in which several of the parameters θ of the model are unknown. If these parameters are time invariant and occur linearly in the function F, and if the noise is additive, we can write the model as

$$x^{\tau+1} = \phi(x^\tau, u^\tau, \tau)\theta + w^\tau, \tag{6}$$

where ϕ is a row vector of functions of the state, input, and time (these are called statistics, regressors, or features depending on the research field in which they are used) and θ is a column vector of the unknown parameters. Suppose that we have some arbitrary value of the parameters of the system θ. Then we can interpret Eq. (6) as a means of predicting the expected output at the next time step given the state x^τ, inputs u^τ, and time τ measured at the current time:

$$\hat{x}^{\tau+1} = \mathbb{E}[x^{\tau+1} \mid x^\tau, u^\tau, \tau, \theta] = \phi(x^\tau, u^\tau, \tau)\theta, \tag{7}$$

where the $\hat{\cdot}$ denotes a predicted value (w^τ drops out in the expectation because it is zero mean). Notice that the predicted output is a function of the parameter values $\hat{x}^{\tau+1}(\theta)$. If we then compare the predicted value $\hat{x}^{\tau+1}$ with the actual value $x^{\tau+1}$, we have an error that gives an indication of how different our model is from the actual system. We form a cost function using this error. One common cost function is constructed from summing over all the squared errors that we have collected from measuring the output of the actual system and comparing it to the predicted output, $J = \sum_\tau \|\hat{x}^\tau(\theta) - x^\tau\|^2$. We can then use an *analytical* optimization method, the least-squares method, to find the parameters $\theta = \theta^*$ that minimize the cost function $J(\theta)$. This can be interpreted as "fitting" the model parameters to the data, and it results in a model that we would expect to give the best prediction of outputs given the inputs. This process is described graphically in Figure 1.

System identification shares many similarities with machine learning; however, because it deals with dynamic systems, the training data are time correlated and are presented in a specific order—they are not independent identically distributed (IID). This is a crucial difference between system identification and most other computational learning problems. Because of this fact, much of the machine learning intuition does not apply to system identification, especially with regard to model validation, as described in more detail in Section 5.3.

3.2. Manipulating the Linear Model

The model structure discussed in Section 2 has the convenient property that it is linear in its unknown parameters. For this reason, it can be manipulated into a form so that its parameters can be fitted using the system identification technique described above. In keeping with our minimalist approach, we assume that only position measurements, p_i^τ, $\tau = 1, \ldots, N$, are available to perform the fitting. We can eliminate u_i and v_i from the dynamics in Eq. (1) to provide a second-order equation in the position only. Notice that from Eq. (1) we can write

$$p_i^{\tau+1} = p_i^\tau + \Delta t \begin{bmatrix} u_i^\tau \\ v_i^\tau \end{bmatrix}, \tag{8}$$

$$\begin{bmatrix} u_i^{\tau+1} \\ v_i^{\tau+1} \end{bmatrix} = a_i \begin{bmatrix} u_i^\tau \\ v_i^\tau \end{bmatrix} + \sum_{j=1, j\neq i}^{m} f_{ij}^\tau + g_i^\tau + w_i^\tau. \tag{9}$$

We can solve Eq. (8) for $[u_i^\tau \; v_i^\tau]^T$ and substitute into the right-hand side of Eq. (9). We then substitute the result back into the right-hand side of Eq. (8), shifting time indices appropriately, to obtain the desired expression

$$p_i^{\tau+2} = p_i^{\tau+1} + (p_i^{\tau+1} - p_i^\tau) a_i + \Delta t \left(\sum_{j=1, j\neq i}^{m} f_{ij}^\tau + g_i^\tau + w_i^\tau \right).$$

We can use the above expression to formulate a one-step-ahead predictor in the form of Eq. (7). First, define the combined regressor vectors $\phi_{u_i}^\tau = [(e_i^{\tau+1} - e_i^\tau)/\Delta t \phi_{fu_i}^\tau \phi_{gu_i}^\tau]$ and $\phi_{v_i}^\tau = [(n_i^{\tau+1} - n_i^\tau)\Delta t \phi_{fv_i}^\tau \; \phi_{gv_i}^\tau]$ and a combined parameter vector $\theta_i = [a_i \, \theta_{f_i}^T \, \theta_{g_i}^T]^T$. By taking the expectation conditioned on the positions, substituting Eqs. (3) and (5) for $\sum_{j\neq i} f_{ij}$ and g_i, respectively, and then making use of the combined regressor and parameter vectors, we get

$$\hat{p}_i^{\tau+2} = p_i^{\tau+1} + \Delta t \begin{bmatrix} \phi_{u_i}^\tau \\ \phi_{v_i}^\tau \end{bmatrix} \theta_i, \tag{10}$$

where $\hat{p}_i^{\tau+2}$ is the expected value of p_i after $\tau + 2$ time steps, given positions up to $\tau + 1$, and w_i^τ drops out in the conditional expectation.

3.3. Batch Method

The so-called least-squares batch method is now implemented to find the optimal model parameters. Specifically, we wish to find the parameters θ_i to minimize the mean squared prediction error over all available time steps. The mean squared prediction error can be written $J_i = 1/(N-2)\sum_{\tau=1}^{N-2}(p_i^{\tau+2} - \hat{p}_i^{\tau+2})^T(p_i^{\tau+2} - \hat{p}_i^{\tau+2})$. Substituting into J_i with Eqs. (10) and (8) yields

$$J_i = \frac{\Delta t^2}{N-2}(Y_i - \Phi_i\theta_i)^T(Y_i - \Phi_i\theta_i), \tag{11}$$

where

$$Y_i = \left[u_i^2 \ldots u_i^{N-1} \; v_i^2 \ldots v_i^{N-1}\right]^T,$$

$$\Phi_i = \left[\phi_{u_i}^{1\,T} \ldots \phi_{u_i}^{N-2\,T} \; \phi_{v_i}^{1\,T} \ldots \phi_{v_i}^{N-2\,T}\right]^T,$$

and u_i^τ and v_i^τ are obtained from Eq. (8). The least-squares problem is then formulated as $\theta_i^* = \arg\min_{\theta_i} J_i(\theta_i)$. Following the typical procedure for solving the least-squares problem, we find that

$$\theta_i^* = \left[\Phi_i^T \Phi_i\right]^{-1} \Phi_i^T Y_i. \tag{12}$$

The right-hand side of Eq. (12) consists entirely of measured data, and the left-hand side is the vector that represents the optimal parameters of the model. We assume that the data are rich enough that the matrix inversion in Eq. (12) is possible. The deep implications of this invertibility are discussed in Ljung (1999). The myriad merits and deficiencies of least-squares fitting compared with other learning methods will not be discussed in this work.

The white noise signal w_i^τ can now be estimated using the resulting residual error in the fitting process, so that

$$\hat{w}_i^\tau = \begin{bmatrix} u_i^{\tau+1} \\ v_i^{\tau+1} \end{bmatrix} - \begin{bmatrix} \phi_{u_i}^\tau \theta_i^* \\ \phi_{v_i}^\tau \theta_i^* \end{bmatrix},$$

where $\hat{w}_i^\tau$ is our estimate of w_i^τ. If the "true" system dynamics are represented by the fitted model, we expect to find that $\hat{w}_i^\tau$ is zero-mean, stationary, Gaussian white noise, as this would confirm our initial assumption on the properties of w_i^τ. Specifically, for perfect fitting, $\mathbb{E}[\hat{w}_i(t)\hat{w}_i^T(t+\tau)] = \delta(\tau)Q_i$, where $\delta(\tau)$ is the Kronecker delta function. Therefore, the "whiteness" of $\hat{w}_i$ can be used as an indicator of the goodness of fit that has been achieved. We use this fact in Section 5.3 to validate our learned models. For simulation purposes, as in Section 6, we would assume that w_i^τ is a white noise process with covariance Q_i equal to the empirical covariance of $\hat{w}_i^\tau$.

In such a way, we learn a cow model for each cow in a herd using measured tracking data. The optimal parameters are found and the characteristics of the random vector $\hat{w}_i^\tau$ are determined for each cow $i = 1, \ldots, m$ to yield parameters for the entire herd. To make the entire process more clear, we have codified it as Algorithm 1.

Algorithm 1 Batch Identification of Group Dynamics

```
**for** All agents in the group **do**
    Apply the measured data to (12)
    Use $\theta_i^*$ in (10)
    This defines the model for agent $i$
**end for**
```

3.4. Recursive Method

The least-squares method can also be formulated recursively, so that each new available measurement becomes integrated into the parameter estimates, tuning them as time progresses. This method would be particularly useful for the parameter identification step in an adaptive control loop.

First, let $\phi_i^\tau = [\phi_{u_i}^{\tau\,T} \phi_{v_i}^{\tau\,T}]^T$ and $y_i^\tau = [u_i^\tau v_i^\tau]^T$. We wish to tune parameters dynamically according to

$$\theta_i^\tau = \theta_i^{\tau-1} + K_i^\tau\left(y_i^\tau - \phi_i^\tau \theta_i^{\tau-1}\right), \tag{13}$$

where

$$K_i^\tau = P_i^{\tau-1}\phi_i^{\tau\,T}\left[\lambda_i I_2 + \phi_i^\tau P_i^{\tau-1}\phi_i^{\tau\,T}\right]^{-1}, \tag{14}$$

$$P_i^\tau = P_i^{\tau-1} - \left(P_i^{\tau-1}\phi_i^{\tau\,T}\left[\lambda_i I_2 + \phi_i^\tau P_i^{\tau-1}\phi_i^{\tau\,T}\right]^{-1}\phi_i^\tau P_i^{\tau-1}\right)/\lambda, \tag{15}$$

where K_i^τ is the parameter gain, P_i^τ is the parameter covariance matrix, λ_i is a forgetting factor ($0 < \lambda_i \leq 1$), and I_2 is the 2×2 identity matrix. This standard algorithm is stated here without derivation. The interested reader can find a thorough discussion in Ljung (1999). The algorithm for this method is given in Algorithm 2.

Algorithm 2 Recursive Identification of Group Dynamics

```
**for** All agents in the group **do**
    Initialize parameters $\theta_i$ and $P_i$ to an arbitrary value
    Use $P_i$ to calculate $K_i$
**end for**
**loop**
    **for** Each agent in the group **do**
        Apply one position to (13) and (15), using $K_i$
        Use resulting $P_i$ to calculate $K_i$ for the next iteration
        Use $\theta_i^*$ in (10)
        This defines the model for agent $i$ for one time step
    **end for**
**end loop**
```

Note that the kinds of systems under consideration are likely to have time-varying parameters. For instance, cows are likely to change their behavior throughout the day in accordance with sunlight, temperature, their hunger and thirst, etc. For this reason, we would expect the parameter-following properties of the recursive algorithm with a forgetting factor to be advantageous. The recursive least-squares algorithm can be used to learn the model while it is simultaneously being used for prediction in a control algorithm. This would result in an adaptive control algorithm for distributed groups. The results presented in the following sections use the batch method. We save a detailed study of online and distributed learning algorithms for future work.

4. DATA COLLECTION EXPERIMENTS

4.1. Animal Monitoring Hardware

We have developed a small lightweight box (see Figure 3) for data collection and animal control for use during our field experiments. The box contains electronics for recording the GPS location of the animal as well as other sensor data that we do not use in this work (a three-axis accelerometer, a three-axis magnetometer, and a temperature sensor). The box also contains electronics for networking with other boxes and for applying sound and electrical stimuli to the animal, though the stimuli were not applied during the data collection experiments described here. Building on the pioneering work of Butler et al. (2006)

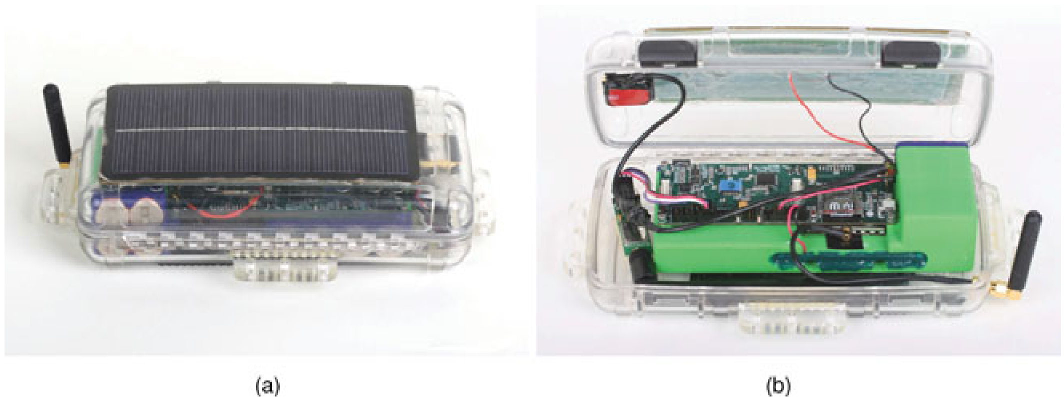

Figure 3. The sensor box is shown here with lid closed (a) and lid open (b). The box is roughly 21.5 × 12.0 × 5.5 cm and weighs approximately 1 kg. It is equipped with a GPS receiver, wireless networking features, and a suite of sensing and actuation capabilities. The lithium-ion batteries and solar panel allow for indefinite operation under normal conditions. It can also modularly accommodate expansion boards for various other applications.

and Wark et al. (2007) on animal monitoring hardware, we improved the performance of the device by mounting it on top of the animal's head, as shown in Figure 4, instead of packaging it as a collar. We found that the head-mounted device improved several aspects of the device's performance compared to the previous collar mounting: (1) the GPS satellites were more likely to be visible from the top of the head, (2) solar panels on the box were more likely to receive direct sun exposure, (3) networking radio communication was less obstructed by the animal's body, (4) the animal was less able to deliberately rotate the box, and (5) the box was prevented from being dipped in water or mud and was generally better protected.

Our sensor box is approximately 21.5 × 12.0 × 5.5 cm and weighs approximately 1 kg. The processor is a 32-bit ARM7TDMI CPU (NXP model LPC2148) with 512 kB of program memory, 40 kB of RAM, USB, and a 10-bit A/D converter. The device also has 256 kB of FRAM (external nonvolatile memory with no rewrite limit) and a removable secure digital (SD) card with 2 GB of storage capacity. Data can be easily and quickly downloaded to a computer by physically transferring the SD card or by downloading remotely via the radios. Two hardware serials are multiplexed

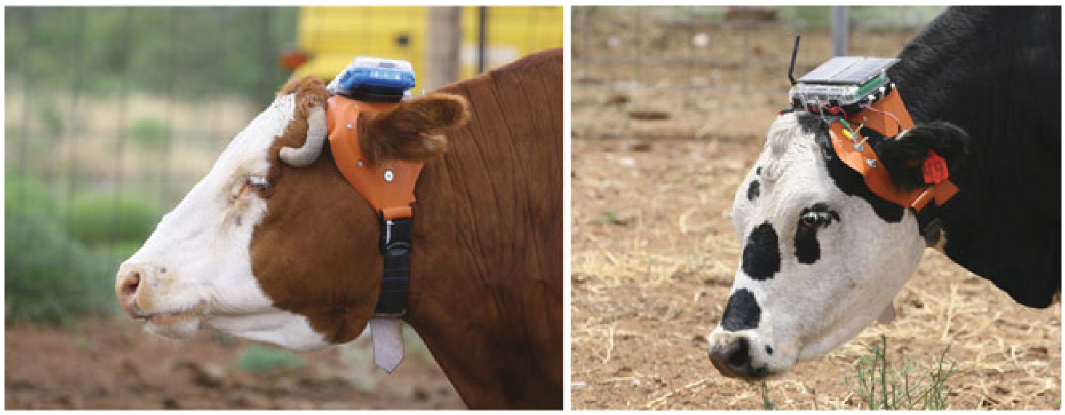

Figure 4. The sensor box is mounted to the head of the cow with a custom-fitted apparatus made of fabric and plastic. The apparatus is designed to use the cow's ears to keep the box in an upright position, as shown in this figure.

for a total of five. The sensors in the box include a GPS engine, three-axis accelerometer, three-axis magnetic compass, and ambient air temperature sensor. There are many general purpose analogue and digital input/output (I/O) lines, so additional sensors can be included.

The communication system consists of two radios. First, a 900-MHz radio (Aerocomm AC4790) with 1-W transmit power is used for long-range, low-bandwidth communication. This radio has a claimed 32-km range and a claimed 57,600-bits (b)/s transfer rate. However, we observed a maximum of only 2-km range and a data transfer rate of only 1,000 b/s. This is particularly odd as the flat, remote environment in which the radios were tested should have been ideal for radio transmission. The cause of the poor performance of this radio is still unknown. Second, the box uses a Bluetooth radio with 100-m range and 100-kb/s data rate for short-range, high-bandwidth communication.

Power is provided by a bank of eight lithium-ion batteries with a total capacity of 16 W-h. The batteries are continuously recharged by a solar panel mounted on the top of the box, allowing the box to run indefinitely under normal conditions. The batteries have enough capacity for several days of operation without the solar panels.

Finally, we have a two-tier animal control system consisting of a set of speakers for applying arbitrary, differential sound stimuli and a set of electrodes that enable the application of differential electrical stimuli. The animal control system was not used during the collection of the data described in this paper.

The box's operating system is a custom-designed collaborative multitasking architecture. Processes run as scheduled events that can be scheduled to run at millisecond intervals with no preemption or real-time constraints. The software supports arbitrary network topologies for communication. Users interact with the system via a serial console or a Java user interface. These can be accessed directly through the serial port or remotely over either of the radios. This allows remote reconfiguration of the monitoring devices in the field. The operating system can be completely reprogrammed using an attached serial cable, remotely over the radio, or by placing a file on the SD card.

4.2. Experimental Methodology

Data were collected during two trials, the first taking place from February 2 to 5, 2007, and the second from July 9 to 11, 2007, during which time 3 head and 10 head of cows were monitored, respectively, using the sensor boxes described above. During both trials cows were allowed access to a 466-ha, or 4.66-km^2, paddock (named 10B) located on the U.S. Department of Agriculture–Agricultural Research Service's (USDA–ARS) Jornada Experimental Range (JER) in southern New Mexico (32° 37′ N, 106° 45′W), which is approximately 37 km northeast of the city of Las Cruces at an elevation of approximately 1,260 m above sea level. The climate of this arid area has ambient air temperatures that range from a high of 36°C in June to below 13° C in January, with 52% of the mean annual precipitation (230 mm) falling as rain between July and September (Paulsen & Ares, 1962; Wainright, 2006). Grasses (39%–46%) and forbs (36%–49%) comprise the predominant vegetation, and woody shrubs compose 14%–19% of the remaining standing crop (Anderson, Smith, & Hulet, 1985; Hulet, Anderson, Nakamatsu, Murray, & Pieper, 1982) that grows in a mosaic pattern across this relatively flat landscape composed of three major landforms (Monger, 2006).

In the first trial, three free-ranging mature beef cattle of Hereford and Hereford × Brangus genetics, labeled Cows 1–3, were fitted with the sensor boxes described above. Data were collected over four days from February 2 to 5, 2007, at a data collection rate of 1 Hz. In the second trial, 10 free-ranging mature beef cattle of similar genetics, labeled Cows 1–10, were fitted with the sensor boxes. Data were collected at 1 Hz over three days from July 9 to 11, 2007. Cows 1, 2, and 3 correspond to the same three cows in the first and second trials. The paddock for the experiments was fenced with the geometry shown in Figure 5. During these two trials, the animals received no audio or electric cues from the sensor boxes.

When they are introduced to a new paddock, cows commonly trace out the perimeter to familiarize themselves with the extent of their new environment (Anderson & Urquhart, 1986). They then concentrate their activities on certain areas, depending on vegetation and other factors. During the first trial, shown in Figure 5(a), the cows had been recently introduced to paddock 10B from another neighboring paddock (though they had previous experience in paddock 10B), and their perimeter tracing behavior is evident in the plot. In the second trial [in Figure 5(b)], the cows had already been in the paddock for some time before data were collected.

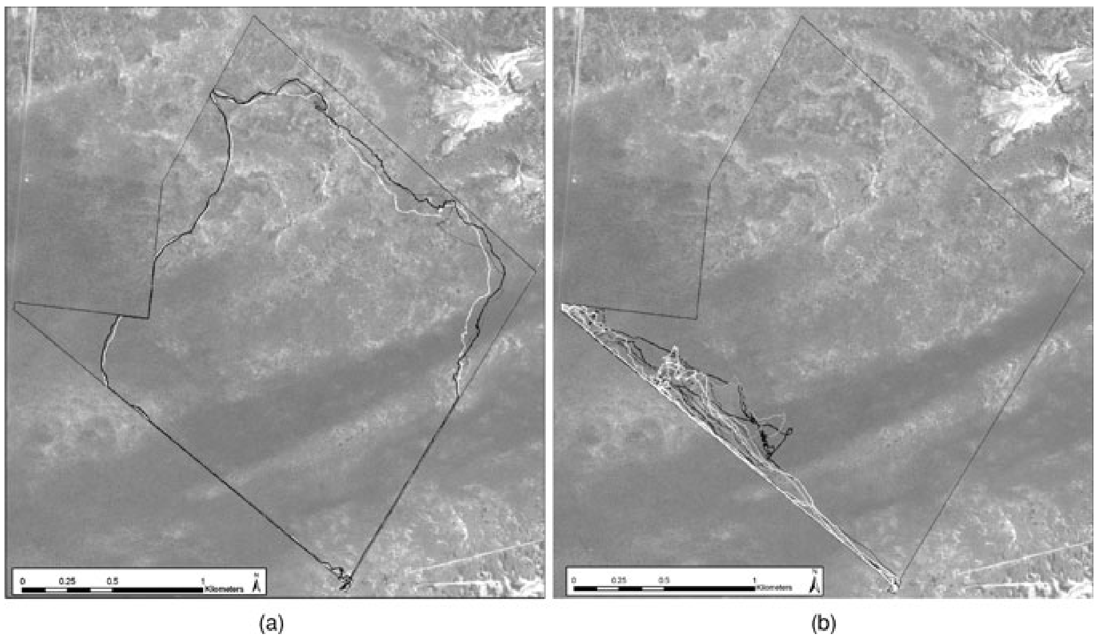

Figure 5. The GPS positions of the cows are shown superimposed on satellite images of the paddock in which the data were collected. (a) Data collected from three cows in the first trial between February 2 and 5, 2007. (b) Data collected from 10 cows in the second trial between July 9 and 11, 2007.

5. MODELING A GROUP OF COWS

The method presented in Section 3 was used to model the dynamics of a group of 3 cows, as well as a group of 10 cows. Data collected as described in Section 4 were used for fitting the model parameters and for evaluating the resulting model. We will first present modeling results for the 3 cows as it is less complicated to interpret data for a smaller group, and then we will show results for the 10 cows. The total number of agent-to-agent interaction forces grows like the square of the number of agents; hence the difficulty in efficiently displaying results for large groups. Finally we discuss the problem of validating the learned models and propose a statistically justified method for validation. Results of the validation method are shown for both the 3- and 10-cow models.

5.1. Three Cows

The dynamics of a cow group are known to be modal (Schwager, Anderson, Butler, & Rus, 2007), in the sense that model parameters are approximately constant over contiguous intervals but can change rapidly when switching between such intervals, for example, when the group transitions from resting to foraging. We intentionally selected a 52-min interval of data (from approximately 18:02 hrs to 18:54 hrs on February 2, 2007) for learning model parameters that corresponded to a stretch of time when the herd was apparently in a constant foraging/walking mode. For each cow, the data used for the least-squares fitting consisted of 3,100 GPS position entries collected at 1 Hz. The data for all animals were artificially synchronized to a common clock using a standard linear interpolation. The characteristic time scale of cow dynamics is considerably longer than 1 s (that is to say, cows move little in the span of 1 s); thus such an interpolation is expected to have a negligible effect on modeling results.

The data were used to find model parameters as described in Section 3. The panels in Figure 6 show the agent-to-agent force magnitudes $\|f_{ij}(p_i, p_j)\|$ for the three cows. For each cow, the two curves show the force imposed by each of the two other cows in the group. Note that the forces are not necessarily pairwise symmetric, that is, $\|f_{ij}\| \neq \|f_{ji}\|$ in general. The force curves are useful for analyzing behavioral traits of the cows. It is well known that groups of

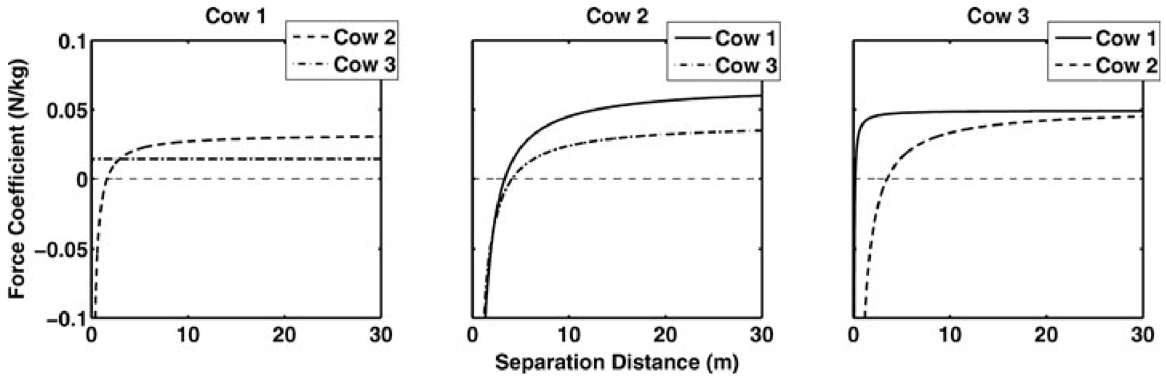

Figure 6. The agent-to-agent interaction forces for the three cows. Each curve represents the size of the force imposed by one cow on another as a function of the distance between the cows. A positive value is attractive, and a negative value is repulsive.

cows have complicated social subgroupings and hierarchies (Lindberg, 2001). The plots indicate that Cows 1 and 3 had an affinity for one another, while Cow 2 was comparatively not very attractive to, or attracted by, Cows 1 and 3. We will reexamine the behavior of Cow 2 below in the context of the 10-cow group.

The environment-to-agent vector fields are shown in Figure 7 for the three cows. The heavy dots show the centers of the Gaussian basis functions γ_{ki}, the arrows show the direction and magnitude of the force felt by a cow at each point, and the curve indicates the position data used for learning. The Gaussian centers were spaced over an even grid containing the trajectory of the cow. If the trajectory did not come within one standard deviation σ_{ki} of a Gaussian function, the Gaussian was dropped from the network. This primitive pruning algorithm was used for simplicity; more complex algorithms could be employed. The Gaussian widths were chosen to be 2/3 the length of the grid space occupied by the Gaussian. This width was found to give good performance with our data. One could imagine including the widths as free parameters in the least-squares cost function (11), but the cost function becomes nonconvex in this case and is therefore very difficult to optimize.

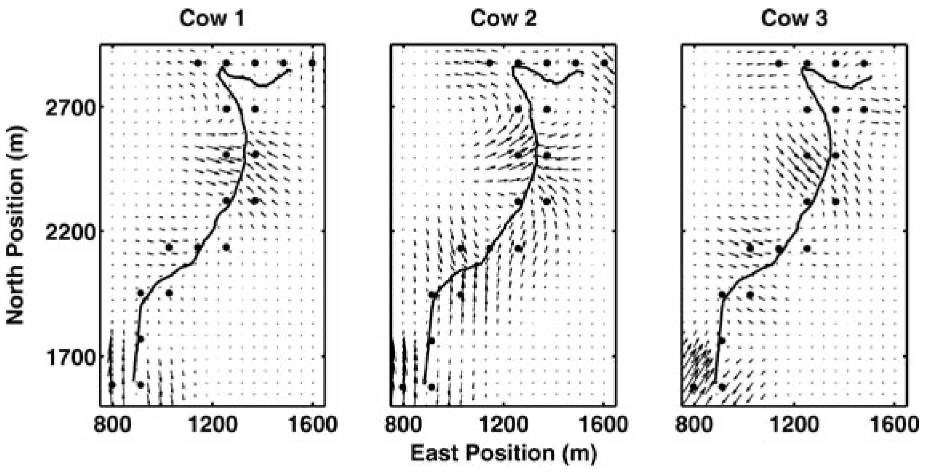

Figure 7. The environment-to-agent force fields for the three cows. The heavy dots indicate the centers of the Gaussian functions, and the arrows show the forces produced by the learned vector field. The continuous curve marks the actual cow's path over the region.

5.2. Ten Cows

For each cow, data consisted of 4,000 GPS position entries collected at 1 Hz during the second trial described in Section 4.2. As before, care was taken to use a contiguous stretch of data (from approximately 11:33 hrs to 12:40 hrs on July 9, 2007) during which the cow group appeared to be in a foraging/walking mode. Cows 1, 2, and 3 were the same animals as in the first trial. The data for all animals were artificially synchronized to a common clock using a standard linear interpolation as was done for the three-cow data.

The data were used to find model parameters as described in Section 3. The panels in Figure 8 show the magnitude of the agent-to-agent force for the 10 cows. The number of agent-to-agent interaction forces is much higher than for three cows (10×9 as opposed to 3×2), so the plots are correspondingly more complicated. In particular, the force plot for each animal shows 10 curves. Each of the nine thin curves represents the magnitude of force caused by each of the nine other animals as a function of separation distance. The thick curve shows the mean over all nine force curves. Despite considerable variation over animals (including some inverted force curves), the mean force felt by any one animal as a result of its proximity to all of the others is relatively similar, as indicated by the mean force curve.

The environment-to-agent vector fields are shown in Figure 9 for the 10 cows. The heavy dots show the centers of the Gaussian basis functions γ_{ki}, the arrows show the direction and magnitude of the force felt by a cow at each point, and the curve shows the position data used for regression. The Gaussian centers were spaced and pruned as described for the three-cow trial.

To demonstrate the potential usefulness of the learned model to study animal behavior, consider again the behavior of Cow 2 in the context of the 10-cow group. By comparing the mean force curves in Figure 8 with the curves in Figure 6, we see that Cow 2 does not tend to stay as far from the other cows in the larger group as in the smaller group. It seems, for example, that Cow 3 stays farther from the other cows than does Cow 2 in the larger group. The apparent dependence of animal behavior on group size is a property of interest to the animal behavioral sciences. Of course, a number of other factors could be responsible for this behavior, including time of year, the animals' physiological state, weather conditions, and the quality and quantity of standing crop. However, by analyzing the learned model we have generated an interesting hypothesis about cow behavior, which can be used to guide the design of further experiments.

5.3. Model Validation

In terms of signal processing, our learning algorithm can be seen as taking a time-correlated velocity signal and producing model parameters and a residual error signal. If our velocity data are rich in temporal correlation, it is good for modeling. Also, if our learned model is successful in capturing the relevant correlation of the velocity signal, the residual error signal will have little temporal correlation. More plainly, we want our velocity signal not to be white and our residual error signal to be white. Therefore, we are interested in testing for "whiteness" in each of these signals by comparing them against a 90% whiteness confidence interval.

To be specific, consider some random signal $x(t)$ generated by a stationary Gaussian white noise process $X(t)$. Each point on the empirical autocovariance function,

$$K_x(\tau) = \frac{1}{T-\tau}\sum_{t=\tau}^{T} x(t-\tau)x(t),$$

is asymptotically normally distributed, with zero mean and variance equal to $K_x(0)^2/(T-\tau)$ (see Ljung, 1999; Lemma 9.A1, or Orey, 1958). The 90% confidence interval is then found from the inverse cumulative normal distribution to have boundaries defined by the curves

$$C_5(\tau) = \sqrt{\frac{2}{T-\tau}}K_x(0)\mathrm{erf}^{-1}(2 \times .05 - 1),$$

$$C_{95}(\tau) = \sqrt{\frac{2}{T-\tau}}K_x(0)\mathrm{erf}^{-1}(2 \times .95 - 1),$$

meaning the process $X(t)$ would produce a value $K_x(\tau)$ below $C_5(\tau)$ with probability .05 and below $C_{95}(\tau)$ with probability .95 for each point τ.

Applying this reasoning to our velocity $y_i(t) = [v_i(t)u_i(t)]$ and residual error $\hat{w}_i(t)$ signals, we validate the learned model by examining the empirical autocovariance functions,

$$K_{y_i}(\tau) = \frac{1}{T-\tau}\sum_{t=\tau}^{T} y_i(t-\tau)y_i(t)^T,$$

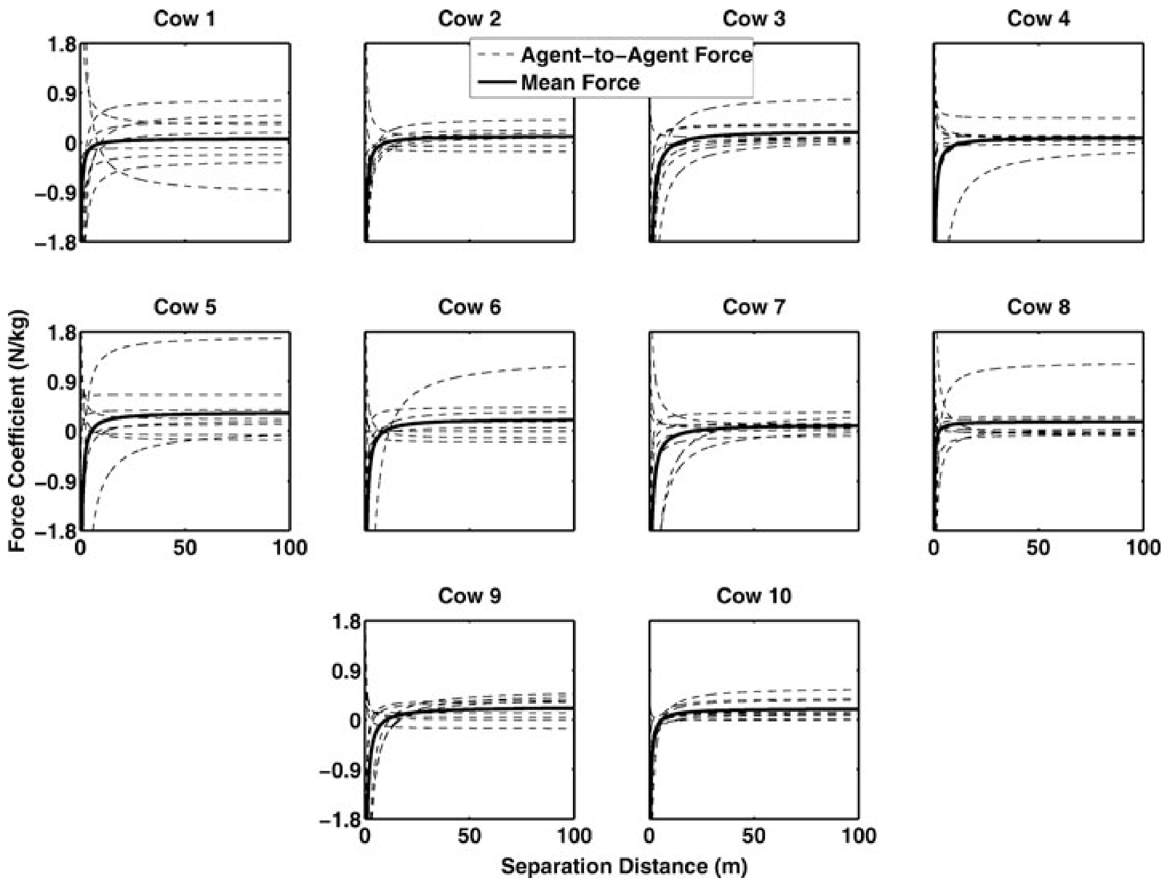

Figure 8. The agent-to-agent interaction forces for the group of 10 cows. Each thin, dashed curve represents the size of the force imposed by one cow on another as a function of the distance between the cows. The thick, solid curve shows a mean over all of the individual force curves. A positive value is attractive, whereas a negative value is repulsive.

$$K_{\hat{w}_i}(\tau) = \frac{1}{T-\tau}\sum_{t=\tau}^{T} \hat{w}_i(t-\tau)\hat{w}_i(t)^T,$$

respectively, where the time of the sample is now explicitly written as an argument, for example $\hat{w}_i(t) = \hat{w}_i^t$. If the velocity $y_i(t)$ and the residual error $\hat{w}_i(t)$ were generated by a white noise process, we would expect $K_{y_i}(\tau)$ and $K_{\hat{w}_i}(\tau)$ to fall within their respective whiteness confidence intervals with probability .9 at each τ. Again, we want the velocity signal to fail this test and the residual error signal to pass it.

There are other tests for whiteness, but this is the simplest one with a rigorous statistical interpretation (Ljung, 1999). This whiteness test takes the place of leave-one-out validation, or other similar validation methods common in machine learning applications. We cannot use such methods because our data are not IID, a key assumption in most machine learning algorithms. Indeed, our model is specifically trying to capture correlation between data points, so to leave one data point out would obscure precisely the relationship we want to learn.

Figure 10(a) shows the autocovariance from the three-cow trial of the eastern component of the velocity for Cow 1, and Figure 10(b) shows the autocovariance of the corresponding residual error. Notice that there is strong temporal correlation in the velocity and all points in the plot lie outside the confidence interval; therefore it fails the whiteness test, as desired. For the residual error autocovariance, there is apparently little temporal correlation and a large

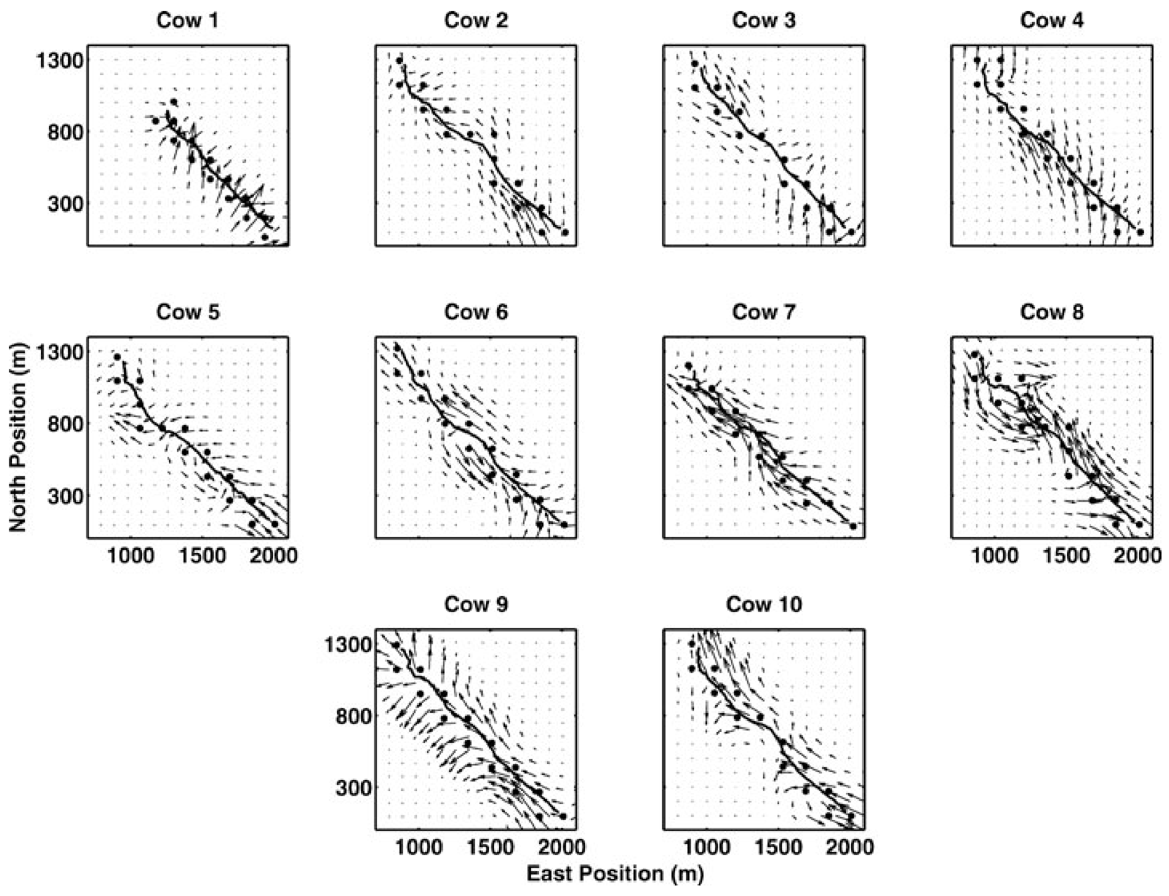

Figure 9. The environment-to-agent force fields for the group of 10 cows. Heavy dots indicate the centers of the Gaussian functions, and the arrows show the force produced by the learned vector field. The continuous curve marks the cow's actual path over the region.

majority of the points lie inside the whiteness confidence interval; therefore it passes the whiteness test. Thus, by this measure, the algorithm has done a good job of producing a model to describe the cow's dynamics. The plots for the other components of the autocovariance functions and for the other cows in the three-cow trial are excluded in the interests of space. Instead, we summarize the results in Table I, which shows for each cow, and for each of the four components of the autocovariance functions $K_{\hat{w}_i}(\tau)$ and $K_{y_i}(\tau)$, the percentage of points within the 90% whiteness interval. The results show that the velocity signals for all cows fail the whiteness test (as desired), whereas the residual error signals can all be considered nearly white in that nearly 90% of their values were within the confidence interval.

The whiteness test was also carried out for 10 cows with similar results as summarized in Table II. The results in the table show that all of the residual errors for the 10-cow model are nearly white. As for K_{y_i} in this case, all of the points for all of the

Table I. Percentage of points lying within the 90% whiteness confidence interval for each of the three cows in the first trial and for each of the four components of the autocovariance function. According to this test, the velocity signal is not white, and the residual error is approximately white, and so the model fits the data well.

	Velocity			Residual		
Cow Number	1	2	3	1	2	3
East–east	0	0	0	76	81	87
East–north	0	0	5	73	78	83
North–east	21	4	17	81	84	91
North–north	0	0	0	66	68	87

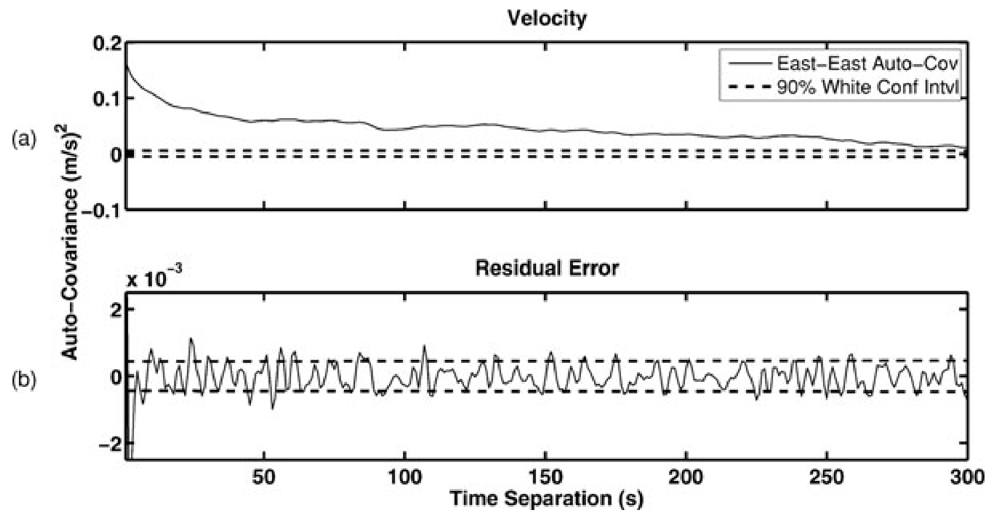

Figure 10. The empirical autocovariance function for the eastern component of the velocity (a) and for the error residual (b). The dotted lines indicate a 90% whiteness confidence interval, meaning that a stationary, Gaussian, white noise process would have generated an empirical autocovariance inside the interval with probability .9 at each point. By this metric, the velocity signal is not white and the residual error signal is "nearly white," indicating that a good model has been learned for the data.

components and all of the cows lie outside of the whiteness confidence interval; therefore the velocity is very likely not white for any cow.

6. SYNTHETIC CONTROL

Simulation experiments were carried out with the model fitted in Section 5.1. We simulated a group of

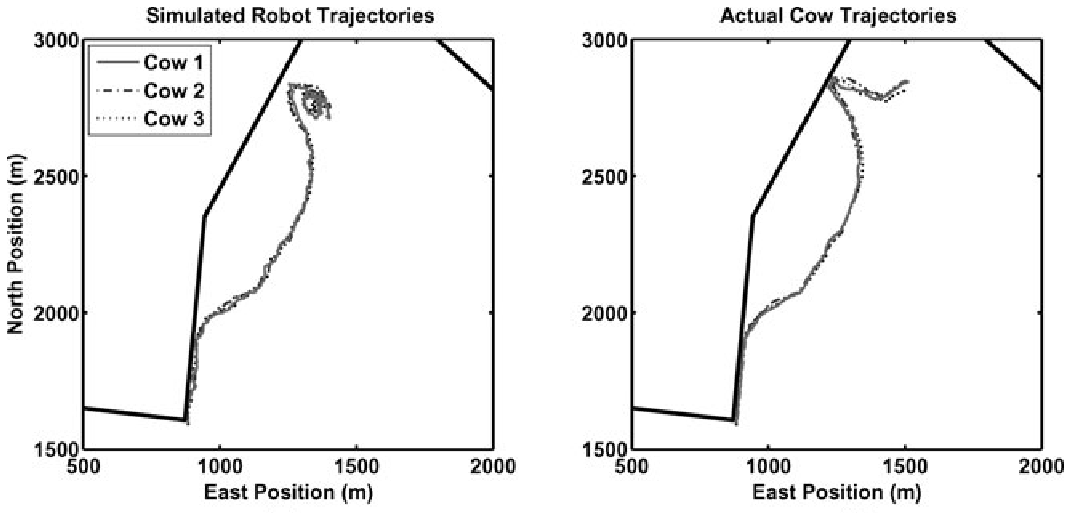

Figure 11. Trajectories of a team of simulated robots controlled to behave like a group of cows. The robots use dynamic laws generated from the procedure described in this work. Their trajectories are superimposed over the fence lines of the paddock where the original cow data were collected, though they have no direct knowledge of fence positions. (b) The actual cow data over the same time window.

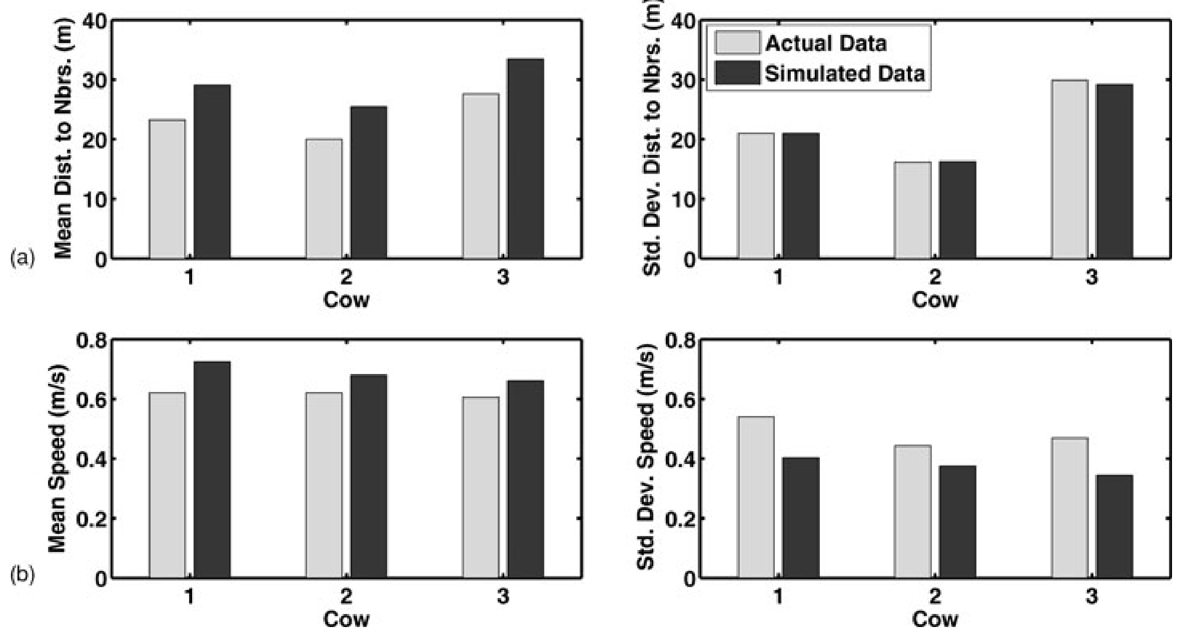

Figure 12. Bar charts comparing statistics for the actual three-cow data from trial 1 and the simulated robots. (a) The mean and standard deviations of the distance from one cow to the other two cows in the group. (b) The mean and standard deviations of the speed of each cow.

Table II. Percentage of points lying within the 90% whiteness confidence interval for each of the 10 cows and for each of the four components of the residual error autocovariance function. By this metric, the residual errors for all cows are approximately white. For the velocity autocovariance function (not shown in the table), no point is within the interval for any cow, and thus the velocity is very likely not white. By this test, the 10-cow model successfully fits the data.

	Residual									
Cow Number	1	2	3	4	5	6	7	8	9	10
East–east	75	85	87	81	69	85	87	79	71	84
East–north	69	82	76	83	60	75	78	77	74	83
North–east	79	80	82	84	61	72	80	80	75	80
North–north	68	75	75	84	62	61	74	73	69	83

three simple mobile robots controlled to have the dynamics in (1) with the parameters found in Section 5.1 These equations were iterated forward in time in a Matlab environment with the robots started from the same initial positions as the cows. The simulation procedure is summarized in Algorithm 3

Algorithm 3 Synthetic Control Algorithm

```
Execute Algorithm 1 to obtain a set of optimal parameters
for each agent
Set initial conditions for simulated group of robots
loop
    for Each robot in the group do
        Use the current state of all the robots and $\theta_i$ obtained
        from Algorithm 1.
        Apply these to the dynamical equations for agent $i$ (1)
        to produce the next robot state
    end for
end loop
```

The trajectories of the robots from a typical simulation are shown in Figure 11(a) laid over a schematic showing the fences of the paddock where the actual cow data were recorded. The trajectories of the simulation are similar to those of the real cows. Most importantly, the simulated robots track the fence lines, as did the real cows. This tendency is captured solely through the agent-to-environment force field (described in Section 2.3), as the model has no direct knowledge of where fence lines may lie.

Furthermore, statistics were gathered for the simulated robots and compared with those from the cow data. Figure 12 shows a comparison of the two sets of statistics. Specifically, the distance between cows over time and the speed of the cows over time have similar mean and standard deviations for the real and simulated data. Thus the model preserves global properties of the group, as measured by these statistics.

One should expect the trajectories of the simulation to be *qualitatively* similar to the actual training data, but the question of *how* similar is not a simple one. The model we have constructed is a random process, and two different sets of data generated by the same random process will almost certainly be different. It is also not informative to look at, for example, the mean distance between points of the actual and simulated data, because, again, two signals from the same random process can generate trajectories arbitrarily far from one another. The appropriate test for model validation is the whiteness test described in Section 5.3. We show Figures 11 and 12 only to indicate that the properties verified with the whiteness test lead, in practice, to a *qualitative* match in performance.

It is also important to point out that comparing these simulation results to the cow data is not the same as testing a learned model on training data, a common pitfall in machine learning applications. Indeed, the only training data given to the simulation are the initial positions of the robots. The model recursively generates its own data points, which then become inputs for successive time steps. This is a manifestation of the fact that system identification takes place in a non-IID setting, and so much of the intuition that applies in typical machine learning problems is not applicable.

This simulation study suggests that our model equations can be used to control a group of robots to exhibit the behavior of the modeled group. In this way controllers can be automatically synthesized for robots to mimic groups that have some desirable collective behavior, such as flocking or herding. One can also imagine introducing artificial members of a group without changing the group dynamics (i.e., without "being noticed") or for the purpose of modifying the group dynamics in a nondisruptive way, for example, to influence collective decision making in natural groups, as was done in Halloy et al. (2007).

7. CONCLUSIONS

In this paper, we presented a method to generate behavior models of groups of dynamic agents, such as cow herds, using observations of the agents' positions over time. We formulated a physically motivated difference equation model and used least-squares system identification to fit the model to data. We demonstrated the method by learning models for a group of 3 cows and a group of 10 cows by using GPS position data. The position data were collected with specially designed sensor boxes fitted to the heads of free-ranging cows. An important and surprising contribution of this work is the demonstration that a minimalist approach to modeling group interactions using only position data leads to meaningful group dynamic models.

Our approach is minimalist in that no information is included in the model about the geometry and configuration of the environment or about any attractive (e.g., vegetation) or repulsive (e.g., fences) features in the environment. It was shown in Section 6, however, that our method can be used to infer the locations of such features, because the robots avoided a fence obstacle even though they were given no prior indication of the fence's existence. An interesting research direction is to investigate the trade-offs between including additional information about features in the environment and the quality of the resulting model. More specifically, we can explicitly model obstacles in the space as a force field with some free parameters that are learned from the position data. We can also include dependencies on weather and other ambient environmental conditions for which measurements are available. The question is, Does the performance improvement of the learned model justify the extra complexity and prior information required for such a model? Our preliminary studies with explicit fence models show that this additional information leads to models that give similar behavior to those without the explicit obstacle features, but the explicit inclusion of the obstacle gives the ability to enforce hard position constraints on the agents. We generally prefer the minimalist approach described in this paper in that it is amenable to situations in which no detailed environmental information is available.

An important problem that we hope to address in the future is capturing the modal changes in the dynamics of groups of agents over long time scales. We collected data at 1 Hz continuously over several days, but as discussed previously, we expect our model to describe the cow group dynamics only over an interval of approximately 1 h, during which time the group was in a single behavioral mode. In the future, we intend to broaden the model class to include switching state-space models. That is, we will model both the motion of the group while it is in one mode and the transitions among modes. With such a model structure, we expect to be able to capture the behavior of the cow group over extended periods of time and to be able to model other natural and artificial groups that exhibit modal properties (e.g., traffic motion, which is congested during rush hour and less so at other times). Unfortunately, exact system identification is known to be intractable for switching state-space models (Chang & Athens, 1978). A topic of current research in the system identification and learning communities is to find approximately optimal parameters using, e.g., variational approaches (Ghahramani & Hinton, 2000) or Markov-chain Monte Carlo methods (Oh et al., 2005). We will investigate these ideas in the future, as well as other approximate methods that may be better suited to our problem. We are also currently exploring other model classes and learning methods. For example, we expect that the environment-to-agent interaction force could be well represented by a kernel-based model with a regularization term included in the least-squares cost function. Other loss functions, such as the hinge loss (for support vector machines), may also be of interest, though the computational difficulties introduced by the hinge loss seem contrary to our minimalist vision.

Also, our method was evaluated on only two sizes of animal groups. It will be interesting to validate the model on much larger groups, though this presents serious technological and logistical challenges in instrumenting and maintaining a large number of cows. We are currently preparing for large-scale field experiments with more animals and incorporating control actuation from the sensor boxes. Eventually we wish to use the stimulus capability of the sensor box along with the prediction capabilities of the model to control the location of the cow herd. Specifically, we plan to investigate how the model will inform the control system to choose actions consistent with the animals' natural inclination for action.

Our work has provided some insights into developing a minimalist approach to modeling group behavior; however, many questions remain to be resolved. Learning models of complex natural and artificial groups is an exercise in balancing trade-offs between model fidelity and model complexity. The systems we are interested in modeling are too sophisticated to characterize their motion in its entirety, but we have shown in this work that a simple model structure with a simple learning algorithm can give enough prediction power to be practically useful for controlling, simulating, and interacting with groups of dynamic agents.

ACKNOWLEDGMENTS

We are grateful for financial support provided in part by National Science Foundation grant II-0513628 and the ARO (Army Research Office) MURI (Multidisciplinary University Research Initiative) SWARMS (Scalable Swarms of Autonomous Robots and Mobile Sensors) project. We thank the staff at the USDA–ARS JER for their support in the use of their facilities and animals, and we thank Barbara Nolen for providing Global Information System (GIS) software support. The cattle data were collected under the New Mexico State University Institutional Animal Care and Use Committee (IACUC), Number 2007-001.

REFERENCES

Anderson, D. M. (2007). Virtual fencing—Past, present, and future. The Rangelands Journal, 29, 65–78.

Anderson, D. M., Smith, J. N., & Hulet, C. V. (1985). Livestock behavior—The neglected link in understanding the plant/animal interface (pp. 116–148). In F. Baker & D. Childs (Eds.), Proceedings of the Conference on Multispecies Grazing. Morrilton, AR: International Institute for Agricultural Development.

Anderson, D. M., & Urquhart, S. (1986). Using digital pedometers to monitor travel of cows grazing arid rangeland. Applied Animal Behaviour Science, 16, 11–23.

Belta, C., & Kumar, V. (2004). Abstraction and control for groups of robots. IEEE Transactions on Robotics and Automation, 20(5), 865–875.

Butler, Z., Corke, P., Peterson, R., & Rus, D. (2006). From robots to animals: Virtual fences for controlling cows. International Journal of Robotics Research, 25, 485–508.

Chang, C. B., & Athens, M. (1978). State estimation for discrete systems with switching parameters. IEEE Transactions on Aerospace and Electronic Systems, 14(3), 418–424.

Correll, N., & Martinoli, A. (2006). System identification of self-organizing robotic swarms (pp. 31–40). In Proceedings of 8th Int. Symp. on Distributed Autonomous Robotic Systems, Minneapolis/St. Paul, MN. Springer Japan.

Cucker, F., & Smale, S. (2007). Emergent behavior in flocks. IEEE Transactions on Automatic Control, 52(5), 852–862.

Delmotte, F., Egerstedt, M., & Austin, A. (2004). Data-driven generation of low-complexity control programs. International Journal of Hybrid Systems, 4(1 & 2), 53–72.

Ferraru-Trecate, G., Buffa, A., & Gati, M. (2006). Analysis of coordination in multi-agent systems through partial difference equations. IEEE Transactions on Automatic Control, 51(6), 1058–1063.

Gazi, V., & Passino, K. M. (2003). Stability analysis of swarms. IEEE Transaction on Automatic Control, 48(4), 692–697.

Gazi, V., & Passino, K. M. (2004). A class of repulsion/attraction forces for stable swarm aggregations. International Journal of Control, 77(18), 1567–1579.

Ghahramani, Z., & Hinton, G. (2000). Variational learning for switching state-space models. Neural Computation, 12, 831–864.

Halloy, J., Sempo, G., Caprari, G., Rivault, C., Asadpour, M., Tache, F., Said, I., Durier, V., Canonge, S., Ame, J. M., Detrain, C., Correll, N., Martinoli, A., Mondada, F., Siegwart, R., & Deneubourg, J. (2007). Social integration of robots into groups of cockroaches to control self-organized choices. Science, 318(5853), 1155–1158.

Hulet, C. V., Anderson, D. M., Nakamatsu, V. B., Murray, L. W., & Pieper, R. D. (1982). Diet selection of cattle and bonded small ruminants grazing arid rangeland. Sheep Research Journal, 8, 11–18.

Jadbabaie, A., Lin, J., & Morse, A. S. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Transactions on Automatic Control, 48(6), 988–1001.

Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L.-S., & Rubenstein, D. (2002). Energy efficient computing for wildlife tracking: Design and early experiences with ZebraNet (pp. 96–107). In Proceedings of Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA. Association for Computing Machinery.

Lindberg, A. C. (2001). Group life (pp. 37–58). In L. J. Keeling & H. W. Gonyou (Eds.), Social behaviour in farm animals. New York: CABI Publishing.

Ljung, L. (1999). System identification: Theory for the user. Upper Saddle River, NJ: Prentice–Hall.

Monger, H. C. (2006). Soil development in the Jornada Basin (pp. 81–106). In K. M. Havstad, L. F. Huenneke, & W. H. Schlesinger (Eds.), Structure and function of a Chihuahuan desert ecosystem. New York: Oxford University Press.

Oh, S. M., Rehg, J. M., Balch, T., & Dellaert, F. (2005). Data-driven MCMC for learning and inference in switching linear dynamic systems (pp. 944–949). In Proceedings of 20th National Conference on Artificial Intelligence, Pittsburgh, PA. Association for the Advancement of Artificial Intelligence.

Olfati-Saber, R., & Murray, R. R. (2004). Consensus problems in networks of agents with switching topology and time-delays. IEEE Transactions on Automatic Control, 49(9), 1520–1533.

Orey, S. (1958). A central limit theorem for m-independent random variables. Duke Mathematics Journal, 25, 543–546.

Paulsen, H. A., & Ares, F. N. (1962). Grazing values and management of black grama and tobosa grasslands and associated shrub ranges of the Southwest (Tech. Rep. 1270, Forrest Service Technical Bulletin). Washington, DC: U.S. Government Printing Office.

Pavlovic, V., Rehg, J. M., & MacCormick, J. (2001). Learning switching linear models of human motion. In Advances in Neural Information Processing Systems 13 (NIPS*2000). Cambridge, MA: MIT Press.

Rutter, S. M., Champion, R. A., & Penning, P. D. (1997). An automatic system to record foraging behaviour in free-ranging ruminants. Applied Animal Behaviour Science, 54, 185–195.

Schwager, M., Anderson, D. M., Butler, Z., & Rus, D. (2007). Robust classification of animal tracking data. Computers and Electronics in Agriculture, 56, 46–59.

Tanner, H. G., Jadbabaie, A., & Pappas, G. J. (2007). Flocking in fixed and switching networks. IEEE Transactions on Automatic Control, 52(5), 863–868.

Tanner, H. G., Pappas, G. J., & Kumar, R. V. (2004). Leader-to-formation stability. IEEE Transactions on Robotics and Automation, 20(3), 443–455.

Wainright, J. (2006). Climate and climatological variations in the Jornada Basin (pp. 44–80). In K. M. Havstad, L. F. Huenneke, & W. H. Schlesinger (Eds.), Structure and function of a Chihuahuan desert ecosystem. New York: Oxford University Press.

Wang, W., & Slotine, J. J. E. (2004). On partial contraction analysis for coupled nonlinear oscillators. Biological Cybernetics, 23(1), 38–53.

Wark, T., Crossman, C., Hu, W., Guo, Y., Valencia, P., Sikka, P., Corke, P., Lee, C., Henshall, J., Prayaga, K., O'Grady, J., Reed, M., & Fisher, A. (2007). The design and evaluation of a mobile sensor/actuator network for autonomous animal control (pp. 206–215). In IPSN '07: Proceedings of the 6th International Conference on Information Processing in Sensor Networks. New York: Association for Computing Machinery.

Zavlanos, M. M., & Pappas, G. J. (2007). Potential fields for maintaining connectivity of mobile networks. IEEE Transactions on Robotics, 23(4), 812–816.