Chapter 1

Algorithmic Materials: Embedding Computation within Material Properties for Autonomy

Ana Pervan* **and Todd Murphey***

*, *Mechanical Engineering, Northwestern University, Evanston IL 60208*

ABSTRACT

The constraints of energy and power density, as well as the speed of internal communications, will drive the next generation of robotic and autonomous systems to computation that is distributed away from a central processor towards embedded computation, perhaps within the compositional materials themselves. One of the most critical drivers for computation within classical systems is control—the ability to impact a system's state evolution in time through decisions—and estimation—the ability to estimate the state based on sensory measurements. Typically both control and estimation techniques are implemented using a centralized, digital computer. Widely used theories of control and estimation unnecessarily push systems towards this centralization by relying on a foundation of mathematical assumptions often employing continuous signals in time and a continuous dependence of the dynamics on state variables. This reliance manifests itself as a limiting but non-essential constraint for next generation robotics. As with recent advances in nanotechnology, the ability to design materials at a variety of scales provides opportunities to enable autonomous systems that operate while *bypassing the need for explicit computation*. Such "algorithmic materials" can assume some aspects of decision making and control embedded into their material properties.

Realizing this vision, however, prompts a revisiting of some of the most basic assumptions of classical approaches. Materials as sensors and actuators will often display discrete states (e.g. low or high conductivity, swelled or un-swelled) rather than continuous ones. In this chapter a route to such "algorthmic materials" is developed. We show that, although the two settings may appear rather dissonant with each other, the classical theory of control and estimation can map in a natural way to materials-based implementations of controllers and estimators. We provide a brief description of classical control and estimation techniques and then use a simple example of a robot with a one-bit sensor (analogous to chemical comparators) and control consisting of the ability to move toward a landmark (analogous to controlled attraction to hydrogels in solution). We additionally provide simulated examples of synthesizing designs based on data-driven models. Altogether, this chapter provides a potential template of a pipeline of mathematical and computational elements that combine to map a control task into an implementable physical design.

KEYWORDS

1.1 INTRODUCTION

The benefits of next generation robotic and autonomous systems manifesting computation that is distributed away from a central processor towards embedded computation are manifold (McEvoy and Correll [2015]). This work aims to show that traditional notions of control theory and system/controller design can be reinterpreted for use in analysis and design of materials-based systems. The chapter is split into two main sections. The first part is an introduction to classical, continuous control and how its formulation creates the appearance of conflict with the discrete capabilities a materials-based systems would often have. The second part uses an example system to demonstrate how these challenges and apparent conflicts can be addressed.

In Section 1.2, we introduce the classical controls perspective—that is, one that assumes the availability of nearly unlimited computation to process signals. This method generally takes the form of a system with an existing model (e.g., the cruise control system in a car) which is composed of a set of differential equations based on physical laws, environmental constraints, and stochasticity (e.g., Newton's laws of motion, the maximum acceleration of a car engine, and unmodeled disturbances from wind, respectively). Control is then synthesized (Section 1.2.1) using an objective function that represents the desired behavior of a system (e.g., error with respect to the desired speed). This control is based on signals that the system measures from the environment (e.g., the speedometer) that have to be processed and interpreted (Section 1.2.2) which requires computational effort. The question we are left with is how to map these capabilities to a discrete setting where the control may be the ability to switch from one physical state to another (e.g., using a chemical inhibitor or not) based on sensors which themselves have a discrete number of states (e.g., chemical comparators that measure threshold concentrations of a chemical). Moreover, given that computation may not be available at the micro-scale, can systems controlled using discrete controllers in a computation-free setting have the same characteristics as systems in a classical controls setting? That is, can we synthesize *cyber-free* autonomy using new capabilities to design and create sophisticated devices using new materials?

To address the above question, Section 1.3 proposes solutions to these challenges. First, in Section 1.3.1, we discuss a simple robotic example that uses a one-bit sensor and six discrete actuator states to control itself in an austere environment, using a nominal model of the relationship between control decisions and the robot's time evolution. Although nominal models, developed from first-principles, are not always good, the model is good enough for the control system to make decisions. Next, in Section 1.3.2, we use the example to motivate the importance of automated exploration, both for useful environmental data that enable decision-making and for the purpose of evaluating and updating the sys-

tem model. Importantly, autonomous systems have the ability to expend energy in order to maximize the quality of information that is sensed. If models are insufficient to predict time evolution well enough for decision-making, data-driven models can be used, as discussed in Section 1.3.3. We end Section 1.3 using the one-bit robot and micro-state machines (which are similar to "smarticles" introduced in (Cannon et al. [2017])) in Section 1.3.4 as motivation for model-based synthesis of cyber-free systems. That is, we argue that one can in some cases use control principles to *generate the organization of sensor components, actuator components, and their interconnections to create desired autonomous behavior.* The techniques outlined in this chapter demonstrate that materials-based implementations of controllers and estimators can be formulated using classical methods of control as a foundation of analysis. They can improve traditional autonomy by taking advantage of the relationships of form and function to omit explicit computation in the cases of systems that are cyber-free either by choice (e.g., to reduce cost) or necessity (e.g., physical scale).

1.2 THE BASICS OF AUTONOMOUS CONTROL

Control has a long history in electronics and macroscale mechanical systems. Generally, there are two classes of systems—linear and nonlinear—addressed using a combination of analysis and computation. Both of these typically assume that there is a *state* $x \in \mathbb{R}^n$, a *control* $u \in \mathbb{R}^m$, and *outputs/measurements* $y \in \mathbb{R}^l$, all measured in time t. An illustration of a basic discrete-time control system is shown in Fig. 1.1. The archetypal distinction between linear and nonlinear systems is in the synthesis techniques available, discussed momentarily in Section 1.2.1. A key thing to note is that the fundamental data associated with the mathematical specification of a control problem is a collection of finite dimensional vectors x and u, consisting of real-valued variables, changing over time.

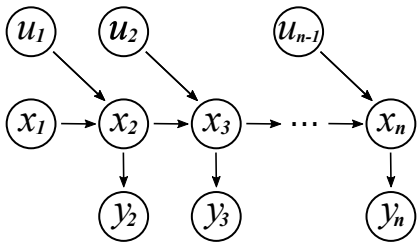


FIGURE 1.1 Basic control environment

A basic control loop begins with an observed state y_k, which the controller uses to compute an action u_k. After u_k is applied, the new state of the system is x_{k+1} and the new observed state is y_{k+1}. This new observed state is then used to calculate the new control output u_{k+1}, and the cycle continues. The

fundamental goal of control theory is to adjust u so that a desired set of x values is achieved. For those interested in a more thorough review of basic control principles, (Franklin et al. [1994]) is an excellent resource.

To illustrate the use of these variables, we consider a kinematic car example (further described in Section 1.3.1). A kinematic car evolves in a plane and is able to observe its x and y position in meters and its orientation angle θ in radians (with respect to a fixed frame). In terms of control, it is able to move forward (u_1) and rotate about an axis (u_2). For this example, the state vector is $\mathbf{x} = (x, y, \theta)$ (variables upon which the controller depends) and the control vector is $\mathbf{u} = (u_1, u_2)$ (outputs from the controller). The goal of the control is typically to move the car from an initial state $x(t_0)$ to a final $x(t_f)$ in time $t_f - t_0$ in an efficient manner, often while avoiding obstacles. Other potential goals are discussed shortly.

The relationship between state and control variables is typically described by a *model* in the form of an ordinary differential equation (ODE)

$$\dot{x}(t) = f(x(t), u(t)) \tag{1.1}$$

where special cases include

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{1.2}$$

and

$$\dot{x} = A(t)x(t) + B(t)u(t), \tag{1.3}$$

the last of which is a *linear* ordinary differential equation. (In general, we will drop the dependence on t for notational brevity.) In the context of materials, a note is helpful here. The ODEs above are almost always finite dimensional—it is comparatively rare to see control of partial differential equations (PDEs). As a result, control typically focuses on regulating how a finite set of variables evolve over time, treating Eq. (1.1) as a constraint on allowable evolutions in time. The list $(x(t), u(t))$ is a *trajectory* if it satisfies Eq. (1.1), where $x(t)$ is the series of measurements one would expect if each state was measured by a sensor.

The right hand side of the ODEs above must be obtained in some manner. First-principle modeling (e.g., based on Newton's Laws in mechanics, Kirchhoff's laws in circuits, or other physical laws) often play a large role. Examples of this include equations of motion describing an automobile suspension or laws of thermodynamics describing fluid in a thermometer. These equations rarely fully specify a correct predictive model of a system. Instead, there are typically parameters that must be empirically determined (e.g., coefficients of friction) through *system identification* (Åström and Eykhoff [1971]); there are cases where nonparametric models must be generated. Models can be based purely or partially on empirical data—something discussed more in Section 1.3.3. Controllers based off of first principle models or empirical models are both considered *model*

predictive controllers, but there exist other controllers (e.g., PID controllers) that do not use models.

Constraints often play a role in a model. For instance, it may be known that a motor saturates (and that as a result $a \leq u \leq b$, where a and b may depend on x). Or it may be that a state x_i is restricted to a subset of $\mathbb{R}^n$ because of obstacles in the work space. These restrictions are called *constraints*, and—although common—create significant procedural and mathematical obstacles when designing control approaches. Another common part of a model includes stochasticity (the addition of a noisy signal) (Chirikjian [2009]). One of the challenges in modeling uncertainty using stochastic models is that stochastic signals model uncertainty in time (i.e., the random variable is a forcing function in time, such as in Brownian motion), whereas in robotics a great deal of uncertainty—perhaps the majority of it—is spatially distributed (e.g., an unexpected visual occlusion does not necessarily vary in time, so a robot with a camera has to *move* in order to overcome the resulting uncertainty or ambiguity, a point we return to in Section 1.3.2).

Only rarely can models be used to analytically predict a system's state response to control. Usually numerical methods (Hairer and Wanner [1996]) must be used to make predictions, and the choice of numerical methods influences both precision of the prediction and complexity of the computation involved. This challenge will be explored further in Section 1.3.3.

To sum up, if materials science is going to help with the practical control aspects of autonomy, it will do so at least partially by impacting $f(x, u)$ in (1.1) and, potentially, impacting the *processing* associated with controlling the system described by $f(x, u)$. For instance, using materials to make f more "well behaved"—e.g., lowering the frequency response or imposing linear interactions between otherwise nonlinearly interacting components—would make control easier and reduce the requirements on the autonomy. Or, materials that simply make the system more predictable in its evolution would radically improve control effectiveness. However, this impact of materials on f is generally not what we mean by "Algorithmic Materials" in this chapter. Instead, we intend algorithmic materials to refer to how materials properties can enable control, encode properties of f or even encode the control design u itself.

It is important to note several abstractions of control are possible, and these provide more flexibility in interpreting what an algorithmic material could be. For instance, discretization of (1.1) in time yields, with some choices of interpolation, a dynamical system of the form $x(i + 1) = f(x(i), u(i))$, which is an example of digital control. Moreover, $u(t) \in \mathbb{R}^m$ can often be discretized so that $u(t) \in \{u_1, ..., u_M\}$, which occurs in the cases of stepper motors, switches, and gears. Lastly, the state—or, more likely, the measurement—can be rewritten as a finite set $x(t) \in \{x_1, ...x_N\}$. Such a system is a *hybrid* system (Liberzon [2012b]) or a finite state machine, depending on whether any of the variables are left in the continuous regime. We will argue that synthesis of these hybrid models from continuous representations will form part of the basis for designing

algorithmic materials.

1.2.1 Synthesis Approaches to Control

System synthesis involving control systems typically focuses on *control synthesis*—finding u based on f's dependence on x and a mathematical statement of the *objective* of the control. Conventionally, this objective is taken to be a function of the trajectory $x(t)$ and control $u(t)$.[1] This allows for a reasonably broad set of specifications of the control system's purpose; for the kinematic car example, the objective could be approaching a specific location in the state space in minimum time, or tracking a desired trajectory while minimizing energy, et cetera. However, not all goals are neatly represented this way, particularly those that focus on linguistic or information-oriented specifications. (Later, we will discuss some alternative choices of objective that could enable/facilitate the use of algorithmic materials.) Most objectives can be written as a functional of the following form:

$$J(x(t), u(t)) = \int_{t_0}^{t_f} \ell(x(t), u(t))dt + m(x(t_f)) \tag{1.4}$$

and, sometimes, the final time t_f under consideration is taken to be ∞. (In this expression, $\ell()$ is the *Lagrangian* of the objective, similar to the Lagrangian from the extreme action principle of mechanics (Marsden and Ratiu [2013]).) In the case of the car example, choosing (x_d, y_d, θ_d) as the desired state could lead to an objective of the form: $J(x(t), u(t)) = \int_{t_0}^{t_f} \frac{1}{2}(u_1(t)^2 + u_2(t)^2)dt + \frac{1}{2}(x(t_f) - x_d)^2 + \frac{1}{2}(y(t_f) - y_d)^2 + \frac{1}{2}(\theta(t_f) - \theta_d)^2$. Parametric changes can be made in terms of weighting terms—there are substantial choices to be made by a designer. If J is differentiable with respect to $(x(t), u(t))$, one extremizes J subject to the constraint $\dot{x} = f(x, u)$ with $x(t_0) = x_0$—that is, the constraint from (1.1)—in order to obtain $u(t)$. Alternatively, one can find $u(x)$ (which we refer to as a control policy, since it is dependent on the state x), which forms the control as a feedback law. Both approaches typically require nontrivial computation, often in the form of optimizations, including: a) direct methods (Kelley [1999]) that discretize the control in time and solve a finite dimensional constrained optimization, b) trajectory optimization (Liberzon [2012a]) optimizing the functional directly, using numerical methods at every iterate, c) dynamic programming (Bertsekas [1995]) where discretization of the state (rather than time) allows one to approximate $u(x)$. All of these approaches are mathematically and practically formidable, and would appear to be a prerequisite to designing materials that implicitly encode the control response, making computational overhead seemingly prohibitive.

There are special cases where control synthesis simplifies dramatically. When one has a system of the form of (1.3), and a quadratic function $J(x, u)$,

1. This function is often called a *functional* to indicate that it takes functions as arguments. It is perfectly fine to view it as a standard function like those encountered in vector calculus.

then $u(t)$ can be expressed in closed form by solving a *Riccati* equation (Bryson and Ho [1975]), yielding u as a linear policy based on state $u(x) = K(t)x$. The Riccati equation itself satisfies a nonlinear, matrix-valued ordinary differential equation, even though the dynamics are themselves linear; as a result, solutions to the differential equation can exhibit finite escape time and other types of numerical instability. Moreover, synthesis of this type can lead to arbitrarily nonrobust control response. Nevertheless, control of linear systems is one of the most successful areas of both theoretical and practical advances. Even in the nonlinear setting, recent progress in synthesis techniques has made many control computation issues less challenging. Specifically, techniques for efficiently approximating the optimal $u(t)$ and the optimal policy $u(x)$ have become more common (Horowitz et al. [2014], Ansari and Murphey [2016], Gong et al. [2008]). (As the engineer faced with control synthesis for nonlinear, high dimensional systems, things have gotten much easier in the last two decades.)

However, as previously mentioned, control synthesis is not the only synthesis one may wish to do. Classical framing of control assumes f is given, but one may wish to design f to make control synthesis easier (e.g., designing a mechanism so that its input-output dynamics are linear). Most challenging is the co-design problem—simultaneously designing f and u so that together they regulate x (Censi [2015]).

Lastly, given x, $f(x, u)$, and u (either expressed as $u(t)$ or $u(x)$), how do we determine if u should be applied to a system? Generally, *stability* is the key requirement used to indicate whether a system will be well-behaved. Roughly speaking, a system is stable if there exists a positive definite function $V(x)$—called a Lyapunov function—that is monotonically decreasing in time for all trajectories (that is, $\dot{V} < 0$) (Khalil [1996]). There are variants of this idea, but for our purposes they all have roughly the same implications—stable control policies are better than unstable ones.

1.2.2 Autonomous Estimation and Perception

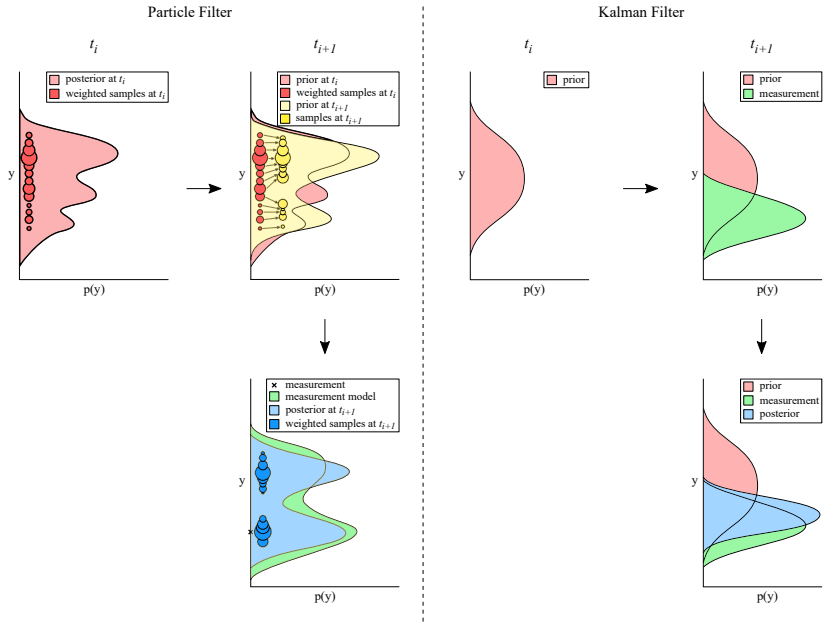
If control theory is the mathematical basis of action based on state, then estimation theory forms the basis of processing data in order to generate a state update. Although estimation is important, a related—and in some respects more general—idea in autonomy is that of perception. For our purposes here, the distinction between estimation and perception is the difference between identifying a signal (e.g., a state (x, y, θ) in a vector space) and an abstraction (e.g., a square or a cup or a door). Despite the dominance of state estimation in classical control, perception algorithms play an extremely important role in robotics, partially because perception of a *feature* (say, of a door) enables a robot to use the feature as a landmark for localization. Here we will argue that perception-based decisions generalize usefully to the setting where sensors may only be sensitive to a discrete set of changes (e.g., a sensor that is purely a *door sensor*, returning a 0 whenever a door is absent and a 1 whenever it is present). Such sensors are

ridiculous sounding in the context of macro-scale robotic systems—a door will always be detected with a multi-purpose sensor such as a camera—but in the case of micro-scale systems chemical comparator sensors can be interpreted in exactly this way—they detect structural features in the environment.

Next, we briefly describe estimators, specifically particle filters and Kalman filters. These two classes of estimation techniques dominate practical estimation, particularly in robotics.

Particle Filters: Particle filters (Thrun et al. [2005]) sample a distribution with a collection of *particles*, generate a prediction of the distribution by forward predicting each particle using Eq. (1.1), and then compare and update that prediction using a measurement and its uncertainty characteristics. There are many variants on how to implement this relatively simple idea, but the key aspects of most implementations are as follows. First, there exists a distribution $\phi(x(t_i))$ that describes the probability that the state $x \in \mathbb{R}^n$ is any particular value at time t_i. This distribution is called the *belief* at time t_i. However, we know that whatever the state is, it satisfies Eq. (1.1) in time, so we have a model of how each state would evolve. As a result, we can generate a prediction called a *prior*, at time t_{i+1} for each state using Eq. (1.1) (possibly including noise as an additive term, if the system has a stochastic model). Naturally we cannot generate a prediction for every possible state, since there are an infinite number of them if the belief has a nonzero value over a volume, but we can *sample* the belief with N states and then use (1.1) for each one of the N samples, called particles. Then we compare the prior distribution for time t_{i+1} with the measurement at time t_{i+1}, combining the two in an optimal way that takes into account the uncertainty of the prediction and the uncertainty of the measurement. The resulting prediction is a reweighting of each sample at time t_{i+1}, and is called a *posterior*. This posterior then becomes the belief for the next time increment. Fig. 1.2 illustrates the operation of a particle filter for a one-dimensional state space. This particle filtering process is able to handle nonlinearities, impacts, and other challenging mechanical phenomena without modification (though sometimes the mathematical analysis of the filter can become challenging in those settings), at the expense of extremely high computation requirements. This description of particle filters is just one of several approaches to implementation, and these sample-based approaches are just one of a broad class of nonparametric filters.

Kalman Filters: Kalman filters (Kalman [1960]) are representative of the other most standard approach to filtering. Rather than represent the entire distribution, Kalman filters minimize the *variance* of the prediction, as illustrated in Fig. 1.2. The formulation of the Kalman filter is perhaps one of the most famous in signals and systems theory, because Kalman used the fact that the optimal update to the estimation problem must be in the form of a projection, and used this to generate an explicit formula for how to incorporate a measurement into an estimate, as a weighted average between the predicted state and the measured state (again balancing the uncertainty of the prediction and the uncertainty of



[th]

FIGURE 1.2 Left: Particle Filter. The leftmost plot shows the posterior distribution at time t_i, and $N = 13$ samples taken from the posterior. The weights of each sample are indicated by the size of the circle. The top right plot shows the prior distribution at time t_i, which is computed by stepping the samples forward in time using Eq. (1.1). The bottom right plot shows the single measurement taken at time t_{i+1} and the distribution of states that may have generated the measurement. The posterior at time t_{i+1} is computed using this measurement model and the prior at time t_{i+1}. New weights for the samples are calculated based on this posterior distribution. Practical implementations of particle filters often have $N > 10^3$. Right: Kalman Filter. The leftmost plot shows the variance in the prior distribution (the prediction at time t_i of the state at time t_{i+1}) and the top right plot shows the measurement taken at time t_{i+1} and the distribution of states that may have generated the measurement. The bottom right plot illustrates the linear combination of the predicted state and measured state, weighted based on the uncertainty of each, resulting in the posteriot at time t_{i+1}. The Kalman filtering process is notably simpler than the particle filter, requiring only one algebraic calculation involving the mean and variance.

the measurement). In fact, a Kalman filter is an implementation of a particle filter if we were to assume a normal distribution of particles and a mapping from t_i to t_{i+1} that preserves the normality of the distribution. As a result, Kalman filters are extremely simple to implement, and require much less computation than particle filters. Moreover, they are globally optimal in the case of linear systems with normally distributed beliefs, and are in fact optimal in an even broader set of nonlinear situations. The upshot is that when one is looking for a minimal implementation, Kalman filters and their variants are almost invariably the preferred filter. Like the particle filter, Kalman filters as described here are representative of a broad class of parametric filtering approaches.

1.3 SYNTHESIS AND DESIGN OF ALGORITHMIC MATERIALS

If classical control and estimation provide our starting point for creating systems that can *sense* the environment and *act* on those sensations, which of those ideas translate to a materials-based approach, where explicit computation is replaced—entirely or partially—with the designed physical response of a system component? In circuits, an archetype of this kind of equivalence is the interchangeability of a gain K with a physical amplifier. This interchangeability is so effective that much of the analysis of linear circuits (signal manipulation using physical elements) is exactly the same as the analysis of linear control (signal manipulation through computation and actuation) (Franklin et al. [1994]). Here we wish to create a similar analogy between computational control and material properties, but we will need to do so in a context of nonlinear and even nondifferentiable behavior.

Materials, even as they become more and more diverse in their designable physical properties, will often be *combinatoric* in terms of the control authority and sensing capability they provide; there will only be a finite number of choices of how to embed control and perception into the material properties of a system. For example, in the case of a micro-state machine (which will be introduced in the next few paragraphs) there are a finite number of sensors and actuators that can physically be on the state machine, and a finite number of ways that logical operators can connect these elements and their specific abilities. The design of this physical system is combinatoric because any combination of these elements exists and is countable, and (by definition of being in the feasible set of designs) is physically realizable.

Here we discuss reasons to believe that combinatoric capabilities should be sufficient for enabling autonomy in many systems. We discuss this in terms of two example systems, where the sensing, control, and/or computation have been severely constrained. First, the example of a very simple device with a one-bit sensor and a few control actions is used to demonstrate that control and estimation can potentially be carried out in a much simpler setting than those seen in Section 1.2. But where do these sensor definitions come from? Where do the actions come from? And what should one do if the computation required

to *process* simple sensory data and simple controls is not available?

The second example system with limited sensing, control, and computation is a micro-state machine—a mechanically designed device at the micro-level (Koman et al. [In Preparation]). Micro-state machines are just that: micro-particles that can act as state machines. They can be thought of as synthetic (engineered) cells. In the example, the machines exist in a chemical bath and are capable of movement using chemical inhibitors, and they contain onboard circuits that include simple sensors and very limited nonvolatile memory. At 100μm in size or less, classical computation is impossible. But the movement, sensory, and memory elements can potentially be combined with a series of logical operators to enable a specific task. We will also show that these discrete structures can sometimes be algorithmically derived using numerical methods, leading to finite state machines that define the sensing, control, and computation all at once.

1.3.1 Robots with One-Bit Sensors

How useful is a one bit sensor to a control system? We looked at this question in (Tovar and Murphey [2012]), where we investigated the use of a one-bit sensor in estimation and feedback control in a planar environment with a differential drive robot. This effort falls within a broader area of inquiry in robotics that falls under *minimalist robotics* (Bicchi and Goldberg [1996]) and *sensorless robotics* (Erdmann and Mason [1988]), and is also described in Strano Chapter. In this work we took a differential drive robot—with a configuration $q = (x, y, \theta)$, where (x, y) is the position in the plane and θ is the orientation between a body frame of the vehicle and the fixed world frame—with kinematics nominally described by

$$\dot{x} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_2. \tag{1.5}$$

(This is likely the most studied *nonholonomic* system in existence, since it is both simple and describes the geometry of a great number of wheeled vehicles.) Moreover, we assumed that the robot can only move in one of six directions, as in Fig. 1.5. Note that this model does not describe the dynamics of the real robot particularly well, but the model does well enough to predict time evolution in an actionable manner; this won't always be the case, as discussed later.

We equipped the robot with a "one-bit" sensor that can register zeros and ones in an environment, as seen in Fig. 1.3. Experimentally, we realized this sensor with a camera that was pointed down at the surface of a table that was white except for black tape, so that the sensor reads zero if the surface is light and one if the surface is dark. As a result, the robot could always sense if it was in a white or black region, such as those shown in Fig. 1.4.

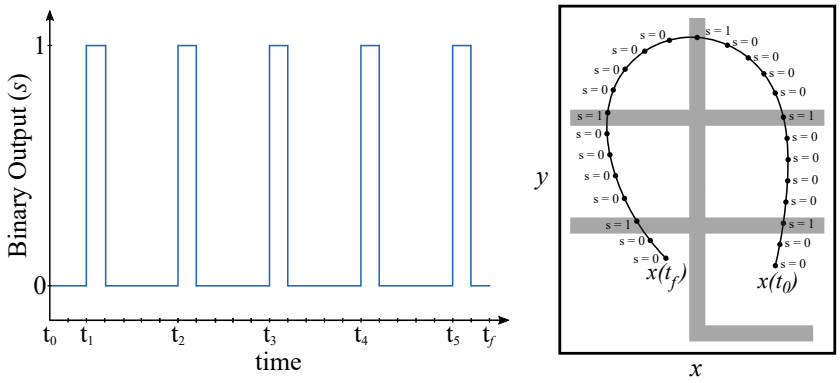


FIGURE 1.3 Example binary output from a one-bit sensor. The robot records a 0 when it detects the white surface and a 1 when it passes a dark line (or *beam*). For this trajectory in this environment, the robot records the following sequential data: {0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0}.

The state estimation question, as discussed earlier, is to take sensor readings—in the case of Fig. 1.3 a sequence of ones and zeros—e.g.,{0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0} at even time intervals—and reconstruct the state (x, y) from those readings. The key idea is as follows. When the robot starts moving, it will have very high uncertainty—until it registers a one, at which point the uncertainty plummets because the robot now knows it is in one of the sparse dark areas seen in Fig. 1.4. Now, as the robot continues moving, since it has a model of its own motion and access to the map, the number of possible states at time t will slowly dwindle, so long as the robot is moving; we will shortly describe how one can design this motion. This point about the necessity of motion is critical—the limited sensing capability is explicitly offset by the ability to record a map to memory, to make predictions about movement, and—most importantly—the ability to move by expending energy.

In (Tovar and Murphey [2012]), we showed that so long as the lines (technically features called *beams* (Tovar et al. [2009])) in these figures intersect, one can uniquely determine (x, y) after encountering *three* nonparallel dark regions. Moreover, the calculation for determining (x, y) is mathematically equivalent to the Kalman filter projection, but in this case the projections are event driven instead of time driven. In the same experiment we equipped the robot with a finite number of control capabilities—basically that it could move toward one of six landmarks (see Fig. 1.5, where the red targets are directions the robot can move towards at any given time). These six control modes and one bit of sensory information were sufficient to enable trajectory tracking of a curve that intersections the dark regions in the environment, both in simulation and experiment.

The key point in all this is that very limited amounts of data, and very

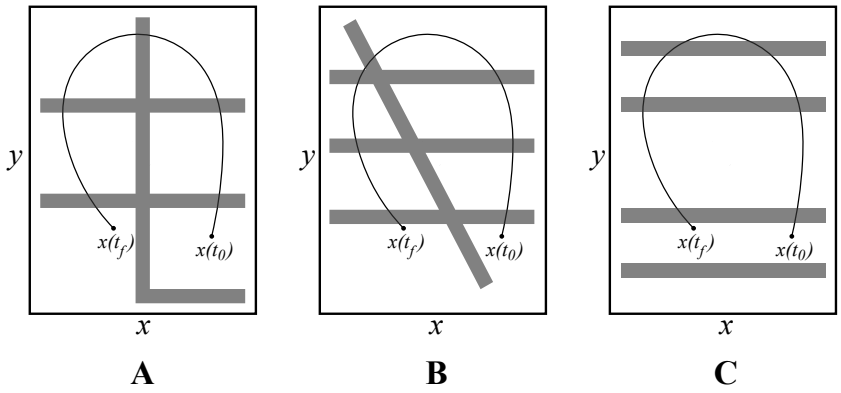


FIGURE 1.4 Differential environmental configurations have different amounts of information. Here the environment-sensor pairing has light and dark binary readings, so that trajectories through the environment will lead to a sequence of ones and zeros being registered in the sensor. Depending on the configuration of these regions (A-C and many others), the robot can completely localize itself in (x, y) coordinates using binary sensory data.

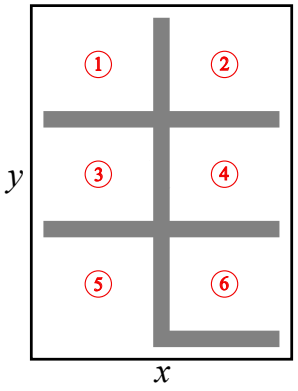


FIGURE 1.5 Control authority is modeled as the ability to go toward a landmark. At any location (x, y), the robot may choose $u \in \{u_1, u_2, u_3, u_4, u_5, u_6\}$. Scheduling u in time based on an objective is the control synthesis problem.

limited control authority, can provide sufficient combined capability to create an *operational autonomous system*. In the case of the one-bit sensor and six control modes, a similar model might arise from the movement of a micro-state machine in a hydrogel, where movement of the state machine is guided by its attraction to each distinct hydrogel, which can each be turned on and off through chemical inhibition.

Moreover, this example makes it clear that the choices of sensor and control depend on each other, on what is known about the environment, and what of that information can be stored in memory and used computationally. Hence, the one-bit sensor is sufficient for control, but only when paired with environmental information so that it is *meaningful*—e.g., actionable to the control system. However, as noted above, *motion for information* is critical, and we have not yet discussed how to automate what the autonomous system should do when it is not getting useful data. This is the focus of Section 1.3.2.

1.3.2 Exploration and Ergodic Control

The differential drive one-bit examples makes it clear that two things are true. First, the absolute signal from a sensor — the sequence $\{0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, ...\}$ — is meaningless without context (likely provided by the combination of a map and a motion model). Secondly, when environmental landmarks, such as the dark regions in Fig. 1.4, are rare , movement is essential in order to obtain information. The first of these comments is just the observation that information is in the changes in a signal rather than the signal itself—this is why a measure of *entropy* , defined as the average amount of information needed to specify the state of a random variable, is useful for determining motion . As a result, what we want is to move so that the sequence $\{0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, ...\}$ has as much variation (increasing its entropy) as possible, avoiding catastrophic outcomes like sitting at a single state recording $\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...\}$ indefinitely. How do we formulate that need as a control objective? This is where coverage, and later ergodicity, plays a role.

Moreover, once a "one" is detected, how we might explore a region should change dramatically. Suppose a region's map looks like Fig. 1.6. Before a "one" is detected, one might expect a uniform search policy is called for; that is, the agent should search in a way that does not prioritize any area over any other area, while avoiding areas already visited. Now note that this is not the same as a "random walk", partially because a random walk may involve excessive amounts of visiting the same region over and over; more importantly, a random walk is defined as an input (a random forcing signal) and generally the output (the state) will not be randomly structured at all. If one wants the output to look like a random walk, one must synthesize control or design the system in some other way to create an output that satisfies a statistical goal (i.e., make the state appear to be a random walk, even if the inputs are chosen deterministically).

To enable this type of random-looking trajectory synthesis—that is, the syn-

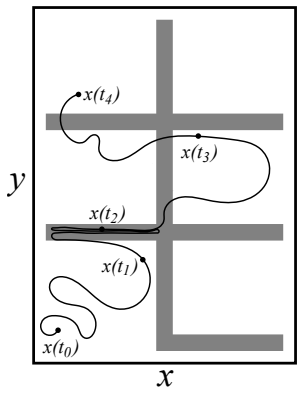


FIGURE 1.6 The goal of ergodic control is to drive the spatial statistics of a trajectory to match those of the distribution of expected information density. This trajectory starts out similar to a random walk, but when a "one" is detected, the EID changes dramatically and therefore so does the trajectory.

thesis of trajectories that have desirable spatial statistical properties—we developed *ergodic control* (Miller et al. [2016]), based on the work in Mathew and Mezic [2011]. The upshot of Miller et al. [2016] is that for a nonlinear system of the form (1.1), we can synthesize control that optimizes a measure of ergodicity of a trajectory $x(t)$ relative to a distribution $\Phi(x)$. This measure of ergoditicy is defined by the requirement that the percentage of time the trajectory $x(t)$ spends in any neighborhood of $\mathbb{R}^n$ must be proportional to the the integral of $\Phi(x)$ over that region . As a result, the need for exploration is met in much the same way as trajectory tracking from Section 1.2.1.

Ergodic control uses a measure of the *distance from ergodicity* between the time-averaged trajectory and the expected information density[2] (EID)—the distribution $\Phi(x)$—as a metric to be minimized for control synthesis. We assume a bounded, n-dimensional workspace (the search domain) $X \subset \mathbb{R}^n$. The spatial statistics of a trajectory $x(t)$ are quantified by the percentage of time spent in each region (open set) of the workspace,

2. An expected information density $EID(x)$ is generally computed based on a measurement model—for instance, of the form $z = \Upsilon(\alpha, x) + \delta$, where z is the measurement, α is the parameter being estimated, x is the state, and δ is zero-mean noise. This measurement model is then used to compute an information measure—e.g., the Fisher information $\mathcal{I}_{i,j}(x, \alpha) = \frac{\partial \Upsilon(\alpha,x)}{\partial \alpha_i}^T \Sigma^{-1} \frac{\partial \Upsilon(\alpha,x)}{\partial \alpha_j}$, where Σ is the noise covariance or the relative entropy, and entropy $H(x) = -\sum_x p(x) \log_2 p(x)$, where x is a random variable and $p(x)$ is its probability distribution. Finally, the expected value of this information measure is taken to calculate the amount of information anticipated in any region. The key point is that ergodic control does not depend on how the EID is computed, only that it exists and is representable as a distribution $\Phi(x)$. One may, for instance, use an entirely different information measure (e.g., one for detection (Streit [2013])) without changing the control approach.

$$C(x) = \frac{1}{T}\int_0^T \delta\left[x - x(t))\right] dt, \tag{1.6}$$

where δ is the Dirac delta (Mathew and Mezic [2011]). The goal of ergodic control is to drive the spatial statistics of a trajectory $x(t)$ to match as much as possible those of the distribution $EID(x)$; this requires the choice of a norm on the difference between the distributions $EID(x)$ and $C(x)$. Following (Mathew and Mezic [2011]), we quantify the difference between the distributions—i.e., the distance from ergodicity $\mathcal{E}$—using the sum of the weighted squared distance between the Fourier coefficients ϕ_k of the EID, and the coefficients c_k of the distribution representing the time-averaged trajectory,[3]

$$\mathcal{E}(x(t)) = \sum_{\boldsymbol{k}=0\in\mathbb{Z}^n}^{\boldsymbol{K}\in\mathbb{Z}^n} \Lambda_k \left[c_{\boldsymbol{k}}(x(t)) - \phi_{\boldsymbol{k}}\right]^2 \tag{1.7}$$

where $\boldsymbol{K}$ is the number of coefficients calculated along each of the n dimensions, and $\boldsymbol{k}$ is a multi-index $(k_1, k_2, ..., k_n)$. The coefficient $\Lambda_{\boldsymbol{k}} = \frac{1}{(1+||\boldsymbol{k}||^2)^s}$ is a weight where $s = \frac{n+1}{2}$, which places larger weight on lower frequency information.

Ergodic control enables coverage by insisting that the statistical distribution of a trajectory $x(t)$ match, as much as possible, a reference distribution $\Phi(x)$. So, if $\Phi(x)$ is a uniform distribution, a trajectory will look somewhat like a random walk. If, however, the distribution is non-uniform—say, because a "one" was encountered in our example—then the resulting trajectory will be statistically similar to a new distribution that reflects that the number of possible states has decreased dramatically. Why should this new distribution be chosen to be the EID? The EID from above measures the expected likelihood of information at a state x, where the information is measured using a metric—we choose the Fisher information (Frieden [2004]) in (Miller et al. [2016]), but any measure can be used. Indeed, any measure could be used and any distribution could be used, leaving the algorithm designer quite a bit of (potentially unwanted) flexibility in implementation.

With the ability to explore, the simplicity of a sensor may be compensated by the ability to *actively acquire data*. For the robot with a one-bit sensor and six control modes, this means that the robot needs to search, using its control authority, for that "one" sensor value. Moreover, the ability to explore the state will have other uses, including obtaining useful data for generating data-driven models. This is discussed next in Section 1.3.3.

3. The Fourier coefficients ϕ_k of the distribution $\Phi(x)$ are computed using an inner product, $\phi_{\boldsymbol{k}} = \int_X \phi(x) F_{\boldsymbol{k}}(x) dx$, and the Fourier coefficients of the basis functions along a trajectory $x(t)$, averaged over time, are calculated as $c_{\boldsymbol{k}}(x(t)) = \frac{1}{T}\int_0^T F_{\boldsymbol{k}}(x(t)) dt$, where T is the final time and F_k is a Fourier basis function.

1.3.3 Data-Driven Models

One of the fundamental problems in using model-based control is that a model generally must be present. However, novel materials—and novel configurations of materials—are often difficult to model based on first principles. If one wants to build up a model based on experimental data, and incorporate that model in control, how should the model be created?

Example situations where this might be matter occur at both the macroscale and microscale. At the macroscale, robotic systems generally have had rigid bodies connected by one or two degree-of-freedom joints. In the last two decades *compliant* (and even soft) robot designs have become more common (Pratt and Williamson [1995]), but these designs lead to substantial modeling challenges. Moreover, physical interfaces (e.g., the contact patch between a robotic finger and an object a robot is attempting to manipulate) are hard to model if they are inhomogenous (e.g., with nonisotropic friction). Even at the macroscale, novel materials lead to poorly posed models, particularly if the materials are inhomogenous in some way. At the microscale (e.g., at the scale of $100\mu m$), predictive models are challenging because of stochastic uncertainty (due to Brownian motion) and uncertainty about the underlying physics governing motion. The key point is that at both the macro scale and the micro scale one may need to use experimental data to generate models of the form in (1.1).

To approach the problem of generating models from data in the context of robotic systems, we have been using Koopman operators (Budišić et al. [2012], Abraham et al. [2017a]). The key idea of a Koopman operator is to represent (1.1) as a linear operator over functions of the state x (much like kernel methods, Gaussian processes, or nonlinear system indentification methods). That is, if $\psi = \{\psi_i(x)\}$ is an infinite set of functions of x (indexed by i), then $\frac{d}{dt}(\psi(x)) = K(\psi(x))$, where K is the Koopman operator. This somewhat abstract attitude towards the dynamics has the considerable upside that *all* possible equations of motion are linear. As a result, one can directly compute K in terms of the spectrum (e.g., eigenvalue/eigenvector pairs) of experimental data to generate a *dynamic mode decomposition* of a dynamic system. The considerable downside to this attitude about model derivation is that nominally an infinite number of $\psi(x)$ are required. In practice, ψ is chosen to be some choice of basis functions (e.g., polynomials, Fourier coefficients) and is then truncated to a finite number. The choice of ψ is an art at present, with no rigorous methods—of which the authors are aware—available for choosing basis functions and the order of truncation.

However, even with *ad hoc* decisions about basis functions and order, the Koopman operator representation of a mechanical control system is surprisingly useful. A simulation of Koopman-assisted modeling is shown in Fig. 1.7. This example consists of an unknown, highly nonlinear potential field (a) and a nominal model that is a simple, quadratic potential field (b). Starting with some initial conditions, a trajectory is predicted using the nominal model (shown in

(c)) which is significantly different from the actual trajectory of the point in the potential. The Koopman operator is then computed using a small amount of data points and twenty five basis functions (including first order polynomials and radial basis functions). The new model of the dynamic system resulted in the improved trajectory shown in (d). This process is repeated in (e) and (f), with increasing numbers of Koopman operator updates in the trajectory calculation. Even though the nominal model was quite inaccurate, the Koopman operator constructed a model of the system that is indistinguishable from the actual trajectory.

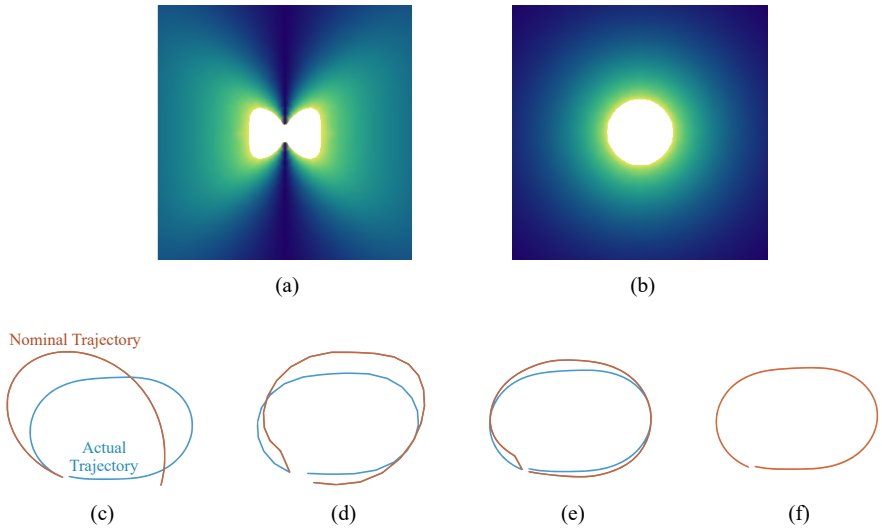


FIGURE 1.7 Koopman operator for one-bit, finite action robot. (a) Simulated "actual" potential field (b) Simulated "nominal" potential field (c) Nominal trajectory (orange) versus actual trajectory (blue) (d) Koopman operator trajectory with 50 updates versus actual trajectory (e) Koopman operator trajectory with 500 updates versus actual trajectory (f) Koopman operator trajectory with 5000 updates versus actual trajectory. Note that the last Koopman operator captures the dynamic behavior very well, without any additional analytical modeling.

To show the effects of the Koopman operator in a physical experiment, we took a sphero (by Sphero [2017]) robot—a sphere that has a kinematic internal mechanism similar to an animal toy—to run on the surface of the ground. Placing this robot on a sandy medium leads to granular mechanics—a traditionally challenging area for robotic vehicles. However, by building up a model of the robot/sand interface and finding dynamics based on data obtained from experiments running the robot through the sand, a much more aggressive model-based controller became possible (Abraham et al. [2017a]).

In many regards, data-based modeling for model-based control is a major en-

abling capability for robots with novel materials in them, including soft robots, robots operating in soft or nonhomogeneous environments, et cetera. It avoids the need for precise analytical modeling of the robot, instead using analytical models as the starting point for generating a data-based model. Moreover, although Koopman operators are the authors' current preference, other data-driven modeling techniques are available, including modeling state evolution using Gaussian processes. What is necessary, however, is the ability to accumulate data over a sufficiently large volume of the state space so that the models are predictive over the range of states one expects to encounter (e.g., using the techniques from Section 1.3.3). The feasibility of turning models, first-principle or data-driven, into feasible physical designs is the focus of the next section.

1.3.4 From Continuous Control to Finite State Machines

Thus far, we have focused on the analog between classical, continuous control and forms of control and autonomy that might be more amenable to implementation with materials (e.g., replacing continuous data with discrete detections that can be implemented using a binary comparator, identifying $f(x, u)$ based on data, et cetera). But even with these techniques in use, microscale devices face one last obstacle—computation. Computation on very small devices will be anywhere from energetically expensive to infeasible. If we wish to create microstate machines, such as those discussed in Section 1.3, how can we make them "smart" without equipping them with a computer?

The way we approach this question is to ask the following, technically precise variant. How can we take a control design and map it to a finite set of state-dependent actions? That is, how can we replace the control policy $u(x)$ with a finite state machine that captures the same essential control response as $u(x)$ (Mavrommati and Murphey [2016])? The way we approach this is to compute $u(x)$ using a finite number of control actions $\{u_1, u_2, ..., u_M\}$—called the *control alphabet*, all dependent on x and potentially data-driven—using an optimal control scheme to minimize an objective function $J(x, u)$. This policy, $u(x)$, is called a *control alphabet policy*. A gearbox, for example, has a finite number of controls (gears), where each gear constitutes a symbol in the control alphabet. The control alphabet policy indicates which gear to use as a function of the state measurements, to minimize some objective function (e.g. a measure of energy expenditure of the gearbox as it varies with state).

We created a special purpose control technique to approximate this optimal control solution efficiently, called switched sequential action control (Mavrommati and Murphey [2016]), but any optimal control method appropriate for a finite set of control choices may be used. Using this method, at any state x, u can be selected from the set of possible u_i. Then a cell subdivision algorithm is used to generate an x-dependent finite state machine (FSM). The finite state machine encodes the global policy as a finite number of dependencies, all a function of the sensor data. As a result, simple circuitry could completely encode

the relationship between sensor data and control action, without ever explicitly computing anything—creating *cyber-free* autonomy in the form of what we are calling a *micro-state machine*. We will explore this critical concept with an example system that is analogous to the robot with the one-bit sensor, but instead uses a micro-state machine in an environment of chemical sources.

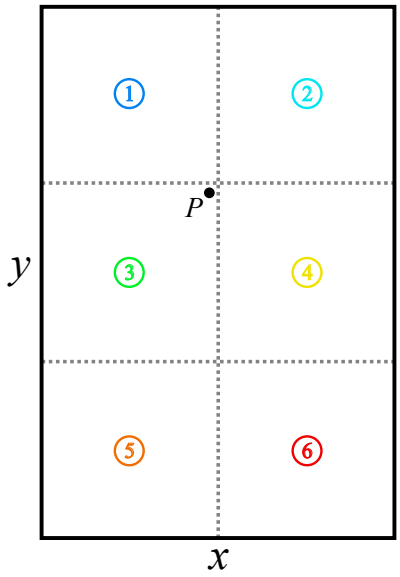


FIGURE 1.8 An example system limited by both sensory ability and control authority. At any location (x, y), the robot may choose u in $\{u_0, u_1, u_2, u_3, u_4, u_5, u_6\}$, where u_0 is the default state of zero control and $u_1 - u_6$ are chemical potentials centered around the indicated points. The control synthesis problem is to schedule u based on an objective (here, to approach a point P).

In this system, there are six controls (or chemical potentials) distributed throughout the space, shown in Fig. 1.8. The micro-state machine can move in the direction of any of these six sources, or do nothing—totaling seven possible controls. A desired point, P, is located near the middle of the environment. We will go into further detail of the capabilities of micro-state machines later in this section, but for now the state machine is moving around in a chemical bath, and it has sensors that can approximate its location based on the beams in the environment. The dotted lines indicate the beams, which in this case are lines of equipotential between the chemical sources detected using chemiresistors and comparators, rather than dark lines detected by a camera, as they were in the one-bit robot example in Sec. 1.3.1. Using the methods described above, a finite state machine can be generated for this system to directly map the control outputs to the sensory inputs. This control policy is shown in in Fig. 1.9.

If a micro-state machine happens to be in the upper right corner of the state

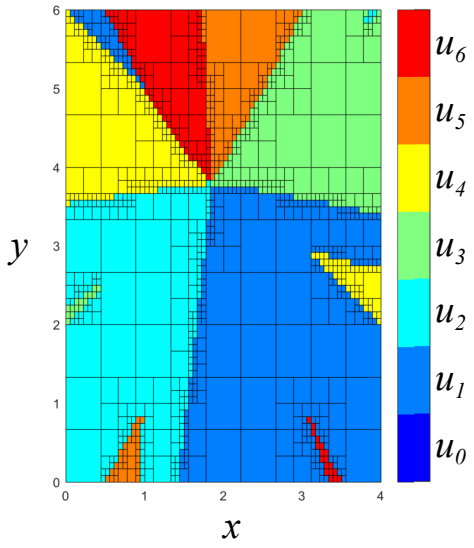


FIGURE 1.9 Control policy for the six source system. The different colors correspond to the different sources that are being tracked (the seven different control modes from Fig. 1.8). Dark blue for zero control, blue for tracking source 1, light blue for source 2, green for source 3, yellow for source 4, orange for source 5, and red for source 6. This phase plane plot of the control policy is a function of x and y, where $\dot{x}$ and $\dot{y}$ are both zero.

space , the control policy directs the state machine toward Source 3, which is below and to the left of the state machine's current location, therefore moving it toward the desired point P.

This example is a four-dimensional system, with state $(x, y, \dot{x}, \dot{y})$, but we only illustrate a two dimensional control policy map in (x, y) with $(\dot{x}, \dot{y}) = (0, 0)$. The rest of this chapter will use only the two-dimensional control policy in figure Fig. 1.9.

To control a micro-state machine with this policy, the state machine's memory would need to contain a list to look up the assigned control for its estimated state. The control policy in Fig. 1.9 has 1040 data points, which could be too much for a simple micro-state machine to physically encode. Fig. 1.10, on the next page, illustrates the same control policy generation method but with different levels of cell subdivision, so that the calculations are done on an increasingly coarse grid. Fig. 1.10 shows that when the control policy becomes very coarse, the Monte Carlo simulations still remain quite accurate—even if the coarse control policy maps are not necessarily very intuitive.

It should be noted that the success of this finite state machine generation method depends on the initial complexity of the system and the number of control and sensory options. But even with 7 possible discrete controls and

limited sensory information, the simulated micro-state machine exhibits good autonomous control accuracy. The key point is that replacing a control policy $u(x)$ with a finite state machine can potentially yield very reliable results. The benefit of having an approach that directly relates sensor data to control action is that it presents an opportunity for real-time control of complex robots with no online computation involved. This means that if the calculated finite state machine can be encoded using material properties, a computer is completely unnecessary in a system. Which brings us to our discussion of the micro-state machine.

A micro-state machine is an excellent example system to demonstrate the principle of cyber-free autonomy. Let us first review the capabilities of a micro-state machine, and then apply them to this example. Micro-state machines use chemical inhibitors to locomote, and they contain purposefully chosen materials and physical devices that react in desired ways, including, but certainly not limited to, sensors (e.g., photodetectors and chemiresistors) and nonvolatile memory (e.g., a memristor). And because explicit computation is infeasible, logical operators are used to convert these elements into a finite state machine that executes a specific task.

One of the easiest ways to implement state estimation in a micro-state machine is the use of chemical comparators. Typically, a comparator is an electronic device that compares two voltages (or currents) and outputs a digital signal indicating which of the two is larger. In this case, relative *chemical potentials* in the environment are being compared. In our example there are six different chemical potentials, and each comparator can only compare two. Fig. 1.11 (a) shows how two comparators can divide the space into four quadrants, which is all of the sensory information that is necessary to estimate its location and implement the most coarse control policy map from Fig. 1.10. Using these two chemical comparators, the micro-state machine can sense that, for example, it is closer to Source 1 than Source 2, meaning that it is in the left half of the map. And if the micro-state machine also senses that Source 5 is closer than Source 2, it can further deduce that it is in the lower left quadrant of the state space. Since the state machine already has the control policy encoded in a list, it knows that in this quadrant it should apply control u_2. This decision tree is shown in Fig. 1.11 (c) and is illustrated in the form of a finite state machine in Fig. 1.11 (d). This is a physically realizable design of a micro-state machine, incorporating existing sensors, actuators, and memory capabilities.

This method can be expanded to micro-state machines with more comparators and more complicated control policies. Control policy maps can be discretized to fit the divisions of the state space that are detectable by sensors on micro-states and—more importantly—micro-state machines can be designed with only the sensors deemed necessary by control policy maps. Once again, this implementation of control and estimation emphasizes the idea of designing physical elements and control policies together.

1.4 CONCLUSION

This chapter provided a brief description of classical modeling, control, and estimation. These three topics provided the foundation for our discussion of autonomy, and how we can rethink the sense-think-act cycle in terms of embedding what would normally be accomplished computationally into a materials-based, cyber-free setting. After the introduction to the classical view, we introduced a simple robotic system—a differential drive robot with six actuation states and two sensor states, analogous to micro-state machines composed of chemical comparators and inhibitors; this system is characterized by limited actuation, limited sensing, and limited computation in a dynamically rich environment with significant model uncertainty. We demonstrated how the discrete sensors and actuators could be used to generate useful control response, and ended by showing that the control response could be implemented without the use of explicit computing. Although a robotic system could rely on computation, it is unlikely that micro-state machines can be expected to, making this last contribution key to the chapter.

The framework described in this chapter is by no means the only way to realize autonomy using novel material properties as a proxy for explicit computation. The particular techniques by which we pursue those goals are not unique; data-driven models can be generated by other means than Koopman operators (e.g., Gaussian processes), exploration can be enabled in other ways (e.g., coverage algorithms), and presumably there are other approaches to generating interconnections between sensors and actions to accomplish a task. Nevertheless, we expect that any cell-sized system that exhibits autonomous, qualitatively rich behavior will be designed using roughly the components we describe here: models built on principles and data, limited sensing compensated for by exploration, and limited computation compensated by explicit, task-based physical design that exploits computation for continuous systems to create cyber-free designs for discrete systems.

ACKNOWLEDGEMENTS

This material is based upon work supported by Army Research Office grant W911NF-14-1-0461. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors.

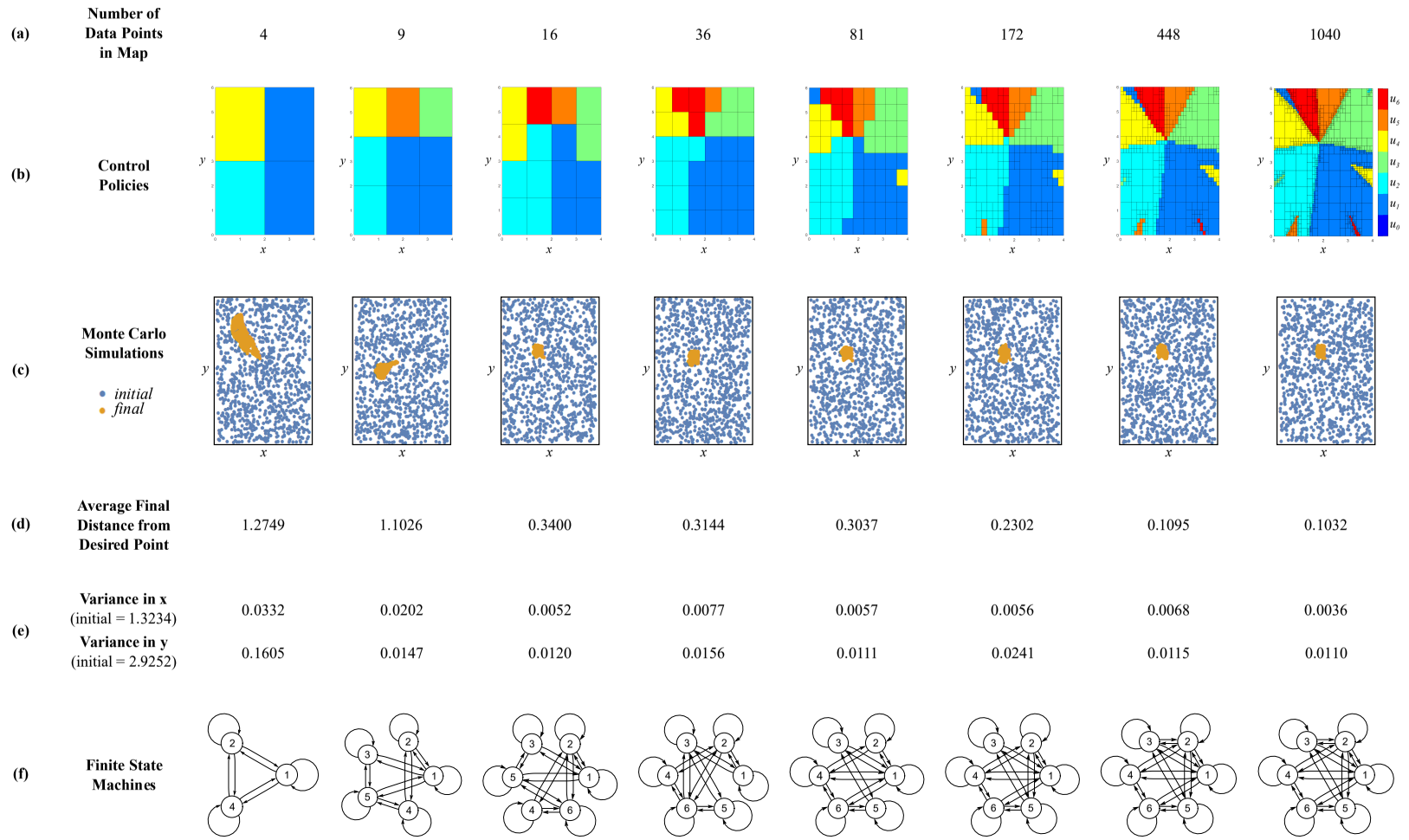


FIGURE 1.10 The control policies for the same system as Fig. 1.9, evaluated more and more discretely. (a) The number of data points in each map, indicative of the amount of memory required on the micro-state machine. (b) The control policy, mapping state to control. (c) 1000 Monte-Carlo iterations indicate susprisingly similar results in most cases. (d) The average final distance from the desired point increases as the map becomes coarser. (e) The variance of the final states about their mean. These are not directly related to coarseness of the grid, but must be considered in combination with the average final distance. (f) Finite state machine representations of the control policies, representative of the complexity (or simplicity) of the corresponding micro-state machine.

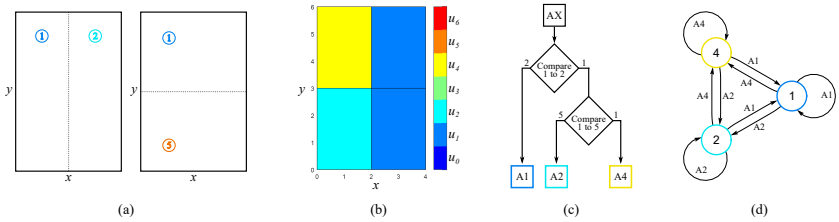


FIGURE 1.11 Physically realizable design of a micro-state machine. (a) Equipotential lines demonstrating how the micro-state machine can use chemcial comparators to estimate its location in the environment. LEFT: comparing chemcial Sources 1 and 2 divides the state space into left and right. RIGHT: comparing Sources 1 and 5 divides the space into top and bottom). (b) The simplest representation of the control policy from Fig. 1.10. (c) Decision diagram for this system. (d) Finite state machine for control.

References

Ian Abraham, Gerardo De La Torre, and Todd D. Murphey. Model-based control using Koopman operators. In *Robotics: Science and Systems Proceedings*, 2017a.

Ian Abraham, Ahalya Prabhakar, Mitra Hartmann, and Todd D. Murphey. Ergodic exploration using binary sensing for non-parametric shape estimation. *IEEE Robotics and Automation Letters*, 2 (2):827–834, 2017b.

Alex Ansari and Todd D. Murphey. Sequential Action Control: Closed-form optimal control for nonlinear and nonsmooth systems. *IEEE Transactions on Robotics*, 32(5):1196–1214, 2016.

Karl Johan Åström and Peter Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971.

Dimitri P Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

Antonio Bicchi and Kenneth Y Goldberg. Minimalism in robot manipulation. In *Lecture Notes, Workshop in 1996 IEEE International Conference on Robotics and Automation*, 1996.

Arthur Earl Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation and Control*. CRC Press, 1975.

Marko Budišić, Ryan Mohr, and Igor Mezić. Applied Koopmanism. *Chaos*, 22(4):047510, 2012.

Designed by Sphero. Sphero robot. `http://www.sphero.com/`, 2017. [Online; accessed 01-September-2017].

Sarah Cannon, Joshua J. Daymude, William Savoie, Ross Warkentin, Shengkai Li, Daniel I. Goldman, Dana Randall, and Andréa W. Richa. Phototactic supersmarticles. *ArXiv e-prints*, 1711.01327, 2017.

Andrea Censi. A mathematical theory of co-design. *ArXiv e-prints*, 1512.08055, 2015.

Gregory S. Chirikjian. *Stochastic Models, Information Theory, and Lie Groups, Volume 1: Classical Results and Geometric Methods.* Birkhäuser, 2009.

Gerardo De La Torre, Kathrin Flaßkamp, Ahalya Prabhakar, and Todd D. Murphey. Ergodic exploration with stochastic sensor dynamics. In *American Controls Conf. (ACC)*, pages 2971 – 2976, 2016.

Michael A. Erdmann and Matthew T. Mason. An exploration of sensorless manipulation. *IEEE Journal on Robotics and Automation*, 4(4):369–379, 1988.

Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison-Wesley, 1994.

Roy B. Frieden. *Science from Fisher Information: A Unification*. Cambridge University Press, 2004.

Qi Gong, Fariba Fahroo, and Isaac Michael Ross. Spectral algorithm for pseudospectral methods in optimal control. *Journal of Guidance Control and Dynamics*, 31(3):460–471, 2008.

Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II*. Springer-Verlag, 1996.

Matanya B. Horowitz, Anil Damle, and Joel W. Burdick. Linear hamilton jacobi bellman equations in high dimensions. In *IEEE Int. Conf. on Decision and Control (CDC)*, pages 5880–5887, 2014.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

Carl T. Kelley. *Iterative Methods for Optimization*. SIAM, 1999.

Hassan K. Khalil. Noninear systems. *Prentice-Hall*, 1996.

Volodymyr Koman, Pingwei Liu, Albert Liu, Daichi Kazowa, and Michael Strano. Micrometer-size electrical state machines based on 2D materials for aerosolizable electronics. In Preparation.

Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2012a.

Daniel Liberzon. *Switching In Systems and Control*. Springer, 2012b.

Jerrold E. Marsden and Tudor Ratiu. *Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems*. Springer, 2013.

George Mathew and Igor Mezic. Metrics for ergodicity and design of ergodic dynamics for multi-agent system. *Physica D-nonlinear Phenomena*, 240(4-5):432–442, 2011.

Anastasia Mavrommati and Todd D. Murphey. Automatic synthesis of control alphabet policies. In *IEEE Int. Conf. on Automation Science and Engineering (CASE)*, pages 313 – 320, 2016.

Anastasia Mavrommati, Emmanouil Tzorakoleftherakis, Ian Abraham, and Todd D. Murphey. Real-time area coverage and target localization using receding-horizon ergodic exploration. *IEEE Transactions on Robotics*, Conditionally Accepted.

Michael Andrew McEvoy and Nikolaus Correll. Materials that couple sensing, actuation, computation, and communication. *Science*, 347(6228), 2015.

Lauren M. Miller, Yonatan Silverman, Malcolm A. MacIver, and Todd D. Murphey. Ergodic exploration of distributed information. *IEEE Transactions on Robotics*, 32(1):36–52, 2016.

Gill A. Pratt and Matthew M. Williamson. Series elastic actuators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 399–406, 1995.

Roy L. Streit. The probability generating functional for finite point processes, and its application to the comparison of phd and intensity filters. *J. Adv. Inf. Fusion*, 8(2):119–132, 2013.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT press, 2005.

Benjamin Tovar and Todd D. Murphey. Trajectory tracking among landmarks and binary sensor beams. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2121–2127, 2012.

Benjamin Tovar, Fred Cohen, and Steven LaValle. Sensor beams, obstacles, and possible paths. *Algorithmic Foundation of Robotics VIII*, pages 317–332, 2009.

Andrew Wilson, Jarvis Schultz, and Todd D. Murphey. Trajectory synthesis for Fisher information maximization. *IEEE Transactions on Robotics*, 30(6):1358–1370, 2014.