Control Design for Cyberphysical Systems Using Slow Computing

Richard M. Murray
Control and Dynamical Systems
California Institute of Technology

27 February 2009

Current techniques for the design of software-enabled control systems rely on the existence of high performance sensing, actuation and computational devices that can be embedded within a physical system at modest cost. Driven by Moore's law, the success of this paradigm can be seen through the broad usage of feedback controllers in modern application areas, ranging from transportation to communications to medicine to robotics. The goal of this project lies at the other end of the computational spectrum: we seek to develop new principles and tools for the design of closed loop control systems using highly distributed, but *slow*, computational elements.

The motivation for control design using slow computing is to develop new architectures for feedback control systems that can be used in applications where computational power is extremely limited. One important class of such systems is that for which the energy usage of the system must remain small, either due to the source of power available (e.g. batteries or solar cells) or the physical size of the device (e.g. microscale and nanoscale robots). A longer term application area is in the design of control systems using novel computing substrates, such as biological circuits. A critical element in both cases is the tight coupling between the dynamics of the underlying process and the temporal properties of the algorithm that is controlling it.

Design of feedback systems using slow computing is particularly challenging due to the performance limitations placed on systems with computational delays that are comparable to the underlying dynamics. These systems are likely to use highly parallel, non-deterministic architectures to achieve what is normally accomplished through the tightly synchronized, serial interconnections of sensing, filtering, estimation, planning, regulation and actuation that are common in traditional control systems. Unfortunately, current techniques for systematic design of control systems assume a mostly serial processing architecture and techniques that make use of parallel architectures (such as neural networks) do not provide sufficiently systematic design methods. The goal of this project is to develop the architectures, theory and tools required to design controllers where computational delay does not allow current techniques to be utilized.

Intellectual Merit: This project will develop new, systematic methods for the design of control systems that can work in the presence of slow computing elements. As specific objectives, we seek to develop (1) an *architecture* for control using slow computing; (2) new *theory and tools* for design of controller for cyberphysical systems that scale to slow computing; and (3) *demonstrations* of the our methodology on university scale experiments in micro-vehicles. The proposed activity makes contributions to the *Foundations* and *Methods and Tools* themes of the CPS solicitation.

Broader Impact: The implementation plan for this project will involve students from multiple disciplines (including bioengineering, computer science, electrical engineering and mechanical engineering) as well as at multiple experience levels (sophomores through PhD students) working together on a set of interlinked research problems. The project will be centered in the Control and Dynamical Systems department at Caltech, which has a strong record of recruiting exceptional women and underrepresented minority students into undergraduate and graduate programs.

1 Motivation and Background

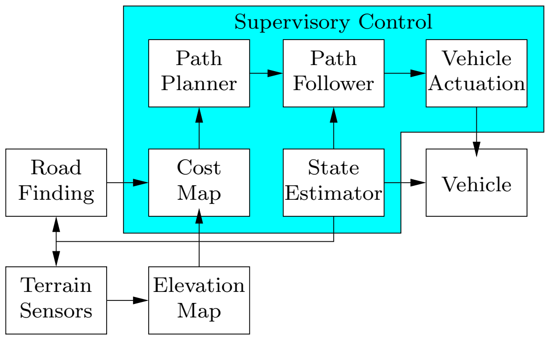
Current approaches to design of software-controlled systems make use of a combination of abstractions and design techniques that are often implicitly based on the assumption that significant computational capacity is available to implement computations and communications. This is a good assumption for many application areas where substantial amounts of computing can be embedded within a physical system to control the dynamical behavior of the underlying process. As a consequence, many of the approaches that are available for designing complex, cyberphysical systems rely on large amounts of computing to achieve complex and robust behavior.

As one example of a traditional complex system, consider the architecture used for the control of an autonomous vehicle, depicted in Figure 1. This particular system, built as part of the DARPA Grand Challenge competitions, made use of dozens of cores of computing, each running at over 2 GHz with dozens of programs and hundreds of independent threads of execution [10, 14]. The computing system for the second generation of the vehicle took approximately 0.5 m^3 of space and 2 kW of electrical power. Other teams made use of similar architectures and computing capabilities [9].

As a complementary example, consider instead the control system for a fruit fly, depicted in Figure 2. This system uses approximately 300,000 neurons with typical time constants in the range of 1–100 msec (10-1000 Hz) and has a physical size approximately equal to that of a sesame seed [18]. Yet it is able to take off, land, avoid obstacles, find food and mate (among other things), often with performance that is beyond what we can do in engineered systems at this size scale. As just two specific instances, the control system of a fly is capable of executing saccades (rapid changes in direction) that occur at angular rates of up to 1800 deg/sec [4] and it can fly in wind gusts that are up to 2X its flight speed in air [23].

The goal of this proposal is to develop some of the fundamental insights and tools that would allow us to design control systems that can perform the tasks of an autonomous car but using an architecture that is compatible with the speed of computation used by an insect. We believe that the development of such an architecture has the possibility of providing new ways of integrating

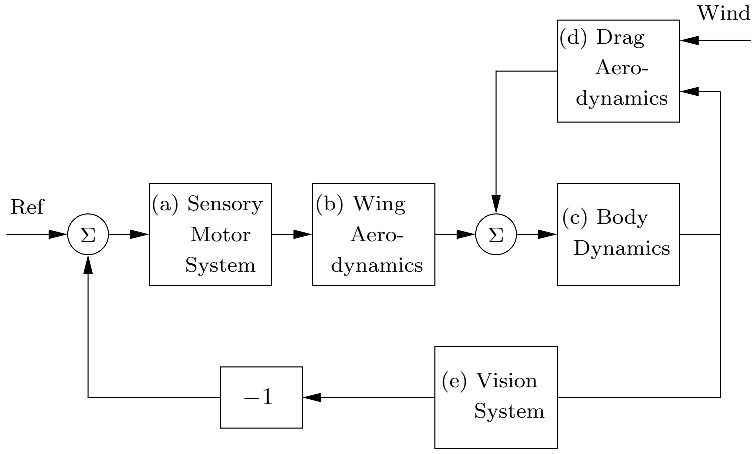
(a) "Alice"



(b) Control architecture

Figure 1: Control architecture for an autonomous vehicle [14]. (a) "Alice", a vehicle developed by undergraduates at Caltech for competition in the DARPA Grand Challenge. (b) Control architecture. The feedback system fuses data from terrain sensors (cameras and laser range finders) to determine a digital elevation map. This map is used to compute the vehicle's potential speed over the terrain, and an optimization-based path planner then commands a trajectory for the vehicle to follow. A supervisory control module performs higher-level tasks such as handling sensor and actuator failures.

(a) Insect in flight



(b) Control system block diagram

Figure 2: Control architecture for an autonomous vehicle [1]. (a) High speed camera image of *Drosophila melanogaster* encountering a wind gust [23]. (b) System model. The mechanical portion of the model consists of the rigid-body dynamics of the fly, the drag due to flying through the air and the forces generated by the wings. The motion of the body causes the visual environment of the fly to change, and this information is then used to control the motion of the wings (through the sensory motor system), closing the loop.

control into systems where large amounts of fast computation are not easily available, either due to limitations on power, physical size or choice of computing substrate. Although this type of computation was probably not the focus envisioned in the CPS solicitation, it is likely that many of the tools and insights required to design such systems will be central to design of other classes of systems in which the effects of time delay, asynchronous execution of parallel computations and highly complex interconnections play a defining role.

Related work There is substantial work in many different fields that touches on some of the issues that we plan to address in this project. We provide here a short analysis of some of the primary areas of overlap, with the goal of indicating some of the methodologies and tools that we hope to build upon from each of these areas. This list is obviously incomplete and we anticipate that new connections will be made early in the project with other areas of research that can help provide insights and solutions to important challenges that we will face.

Artificial neural networks. Artificial neural networks provide one of the most direct linkages to some of the architectural aspects required to develop sophisticated computational using complex interconnections of simple elements [26]. Much of the work on artificial neural networks does not include explicit dynamics within the network nor does it provide a modular structure for composition of basic elements, so extensions are likely to be required to extend existing techniques. The study of recurrent neural networks provides some insights into how dynamics can be incorporated, though they are not completely aligned with the needs of cyberphysical systems, especially the ability to reason about more complex behavior. For many applications using neural networks, it is natural to train the network to perform a task. We are more interested here in forward design and verification, but the basic framework and tools can still serve as a useful starting point.

Hybrid systems. The interconnection of continuous and discrete systems is a key element of the architecture that we propose here and the field of hybrid systems has made tremendous progress in developing methods for modeling and analyzing such systems [3]. The use of reachability analysis [36], bisimulation relations [25], composibility [2] and proof by construction techniques (e.g. [20]) provide a powerful set of tools that can be brought to bear.

Many of the hybrid systems modeling approaches are quite general, allowing them to capture a huge variety of engineering applications. The addition of time delay in the description of the continuous dynamics and asynchronous computation in the discrete dynamics may require some extensions of the current theory, although there is substantial work in the dynamical systems community on the former problem (see, for example, [49]) and the computer science community on the latter (described below). There are also emerging tools in "correct by construction" techniques that allow transformation from specification to design in a manner that provides an automatic proof certificate (see, for example, the work of Kress-Gazit et al. [20]), although most of the techniques do not yet optimize for slow computing architectures.

Distributed and networked systems. The computer science community has a long history of research in distributed and parallel computing, including frameworks such as I/O automata [37] and languages such as UNITY [12]. Tools are available for verifying correctness of concurrent programs using modeling checking [27, 35] and theorem proving [44] techniques. The chief limitation in most existing approaches is the way in which dynamic processes are modeled and analyzed, although symbolic model checkers such as PHAVer [22] are now available and widely used.

The area of networked control systems has emerged in the last 5 years and sought to combine some of the insights from computer science and control to allow analysis and design of systems that consist of multiple agents running across a network [39]. Design oriented techniques such as local temporal autonomy and shock absorbers [46] provide insights into how to design systems that work in the presence of timing uncertainty and information loss. These techniques tend to be focused on information-rich applications with substantial computing power, but some techniques will certainly be applicable in slow computing environments as well.

Simple control systems (Brockett). An intriguing direction of recent work is the design of "simple" control systems that tradeoff performance with computational complexity [8]. The basic idea is to include the complexity of the controller in the formulation of the performance metric, allowing optimization-based approaches to select for controllers that can be implemented with low amounts of computation and still achieve reasonable performance. This work is perhaps the closest in spirit to what we propose here, with the necessary addition of highly parallelized, asynchronous computing that we believe will be required for many applications.

Other areas. There are substantial other areas of research that are likely to be relevant that we do not describe in detail due to space limitations. Some areas with which we have particular familiarity include formal methods for verification of motion control systems (e.g., theorem proving and model checking) [6, 7, 50], methods of software-enabled control [41] and distributed control systems, including consensus algorithms [42, 43]. In each of these areas, powerful tools are available that we hope to adapt and modify to apply to the systems that we describe here. The combined challenges of highly parallel computing, distributed processing, asynchronous execution and slow computation require that many of these methods be modified and extended, but they provide a solid basis on which we can build.

Project Goals and Objectives The overarching goal of this project is to develop systematic methods for the design of control systems that can work in the presence of slow computing elements. As specific objectives, we seek to develop:

1. An *architecture* for control using slow computing. We anticipate that the architecture will make use of highly parallelized, simple computational elements that incorporate nonlinearities, time delay and asynchronous computation as integral design elements. Protocol-based control

systems that allow non-deterministic execution of message-passing based protocols will allow coordination between subsystems and networked operations between agents.

2. New *theory and tools* for design of control systems for cyberphysical systems that scale to highly distributed, slow computing platforms. The theory should capture fundamental limits of performance for closed loop behaviors that can be achieved within the proposed architecture, with supporting design tools that allow appropriate engineering tradeoffs to be made within those limitations. In addition, tools for the design of computationally simple, non-deterministic protocols that allow control at higher levels of abstraction must be developed.
3. *Demonstrations* of our methodology on university scale experiments using collections of autonomous vehicles. The specific application area is chosen to exploit existing infrastructure at Caltech, and will allow demonstration of the main design principles in a representative cyberphysical system.
4. An *integrated educational approach* that engages a diverse set of undergraduate and graduate students from multiple research areas. We will make use of the Control and Dynamical Systems (CDS) program at Caltech, which has a strong track record in attracting undergraduates, graduate and visiting students from different disciplines, including substantial numbers of women and underrepresented minorities.

The proposed activity makes contributions to the *Foundations* and also the *Methods and Tools* themes of the CPS solicitation. A fundamental aspect of the proposed research is the tight integration and explotation of the underlying attributes of the physical system with the computational elements of the control system.

2 Technical Approach

The approach we plan to take in this project builds on several specific areas of control and computer science that have been explored by the PI and his collaborators over the last 5 years. While innovations will be needed to develop systematic design techniques for the class of systems described in this proposal, we believe the initial results described below provide promising directions for some new approaches toward our goal.

2.1 Candidate architecture

A key element of this project will be to explore possible architectures for slow computing. At the high level, this architecture must describe the basic information flow of the control system. At a more detailed level, it will define the types of sensing, estimation, planning, reasoning, feedback regulation and error correction that are required and how these are interconnected.

A starting point for the architecture we plan to explore is shown in Figure 3. The system consists of a set of agents that are interconnected through a network. Each individual agent has a highly structured inner loop, described below, that interacts with a guarded commond language (GCL) protocol engine. This protocol-based feedback system modifies the inner loop dynamics, but also controls communications between other agents, using a packet-based communications network. We describe each of the aspects of architecture first, followed by a description of the features that are important in the context of slow computing systems.

We take the process to be a input/output dynamical system, possibly nonlinear, that has associated sensors and actuators. We assume that the sensing subsystem has a large array of individual sensors that provide information both about the underlying state of the system and its

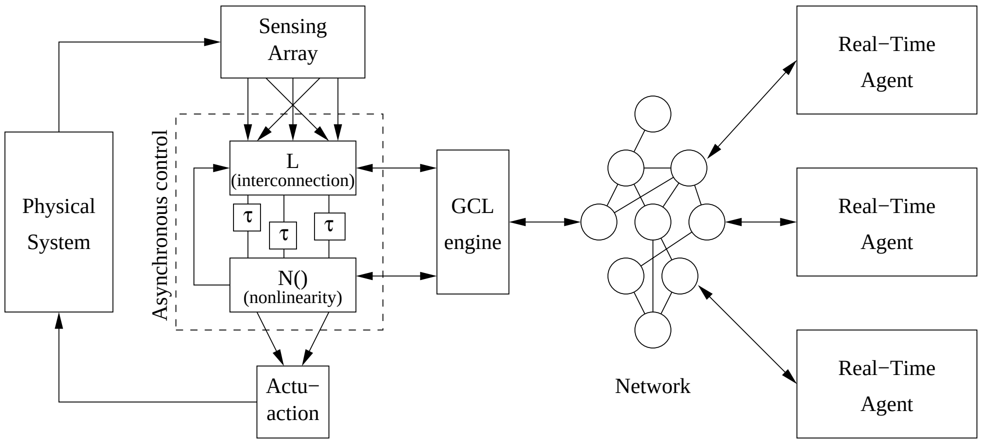


Figure 3: Candidate architecture for control using slow computing. Each system (agent) consists of a control system as shown on the left. The system consists of process dynamics, a (information-rich) sensory array, an asychronous control system (described in more detail in the text) and an actuation system. Multiple systems interact through a protocol-based (GCL) feedback engine operating across a network that incurs delays and possible information loss.

environment. We assume a much smaller number of actuators, so that the control system resembles the sensory-motor cascade that is present in many biological systems.

The "inner loop" control system makes use of an interconnection matrix L, a set of asynchronous delays (represented by the blocks labeled τ) and nonlinear elements $N(\cdot)$. Internal feedback between the nonlinear block and the interconnection block allows a general set of dynamical systems to be formed from this simple structure. This general structure is similar to a recurrent neural network in its basic form, but we intend to take a more structured approach to the analysis and design of the system. Two key features of this portion of the system are that it can be designed to allow a very short processing path between input and output, and it is designed to be run in an asynchronous manner. This level of the architecture is described more fully in Section 2.2.

The asynchronous inner loop interacts with a protocol-based control system, represented in the figure as the the "GCL engine". GCL refers to a guarded command language, a computational framework based on the semantics of the UNITY programming language [12]. GCL can be regarded as a type of non-deterministic finite state machine in which commands execute in an asynchronous manner. The GCL-based control program both interacts with the inner loop control system and provides a network interface for the agent. Both of these interactions occur in an asynchronous fashion, allowing easy parallelization of the rule set used to implement the protocols. Previous work on guarded command languages and their extensions have already proven useful for programming multi-agent control problems that tolerate network latency and asynchronous execution [32, 31], but new research is required to both link this to the inner loop dynamics and provide more systematic design procedures. This component of the architecture is described in more detail in Section 2.3

Finally, we assume that multiple agents interact with each other across a communcations network, using packet-based communication protocols. This network introduces another layer of variable time-delays and asynchronous behavior. It also enables the creation of complex interactions between agents, requiring formal tools for specification, design and verification.

Although this architecture has many familiar elements to researchers in networked embedded systems, there are several features of the execution and application environment that we are trying to capture in this particular formulation:

Information-rich sensory systems. We are interested in applications where the sensor array has large amounts of data (such as the hundreds of ommatidia in an insect flight control systems). It will not be possible to process this information in traditional ways (e.g., object detection and tracking), and hence we must use alternative techniques such as matched filters [45] and wide-field integration [28]. Given the complex tasks that we wish to accomplish, it is likely that we will use parallel pathways that correspond to different desired behaviors, with selection of those behaviors controlled by the protocol-based feedback system.

Minimal latency feedback paths. The primary feedback path in the individual systems will define the dynamics, robustness and behavioral properties of the individual agents. Given the desired limits on computing, this feedback path must minimize latency in order to avoid limitations in performance due to time delays. This can be achieved in the candidate architecture by exploiting the "single pass" nature of the basic inner loop architecture, and using (slower) feedback pathways to modulate the basic regulation function.

Highly distributed, asynchronous execution. Throughout the architecture, we seek to allow for multiple (slow) computing elements to be operating in parallel. This allows a highly distributed approach to control system design, but also drives the system to operate in a asynchronous (or at best loosely synchronized) manner. We anticipate that such asynchronous operations will occur in the inner loop, in the protocol-based supervisory control and in the networked communications between agents, all with potentially similar time scales.

Analyzable hybrid dynamics. Finally, we have sought to constrain the architecture, at least in this preliminary form, so that it contains enough structure to facilitate the specification, design and verification of the overall behavior of the system. At its most basic level, the system is a hybrid system combining continuous dynamics and logic, but we believe that we must limit the class of hybrid dynamics that are allowed in order to apply systematic design and analysis techniques.

There are many open questions concerning this candidate architecture in terms of its expressiveness, performance characteristics and verifiability. These will each be explored in the context of the overall research project, with the goal of modifying the candidate architecture to meet the features above. We now proceed to describe in more detail the various aspects of this candidate architecture and lay out the technical approach that we propose in more detail.

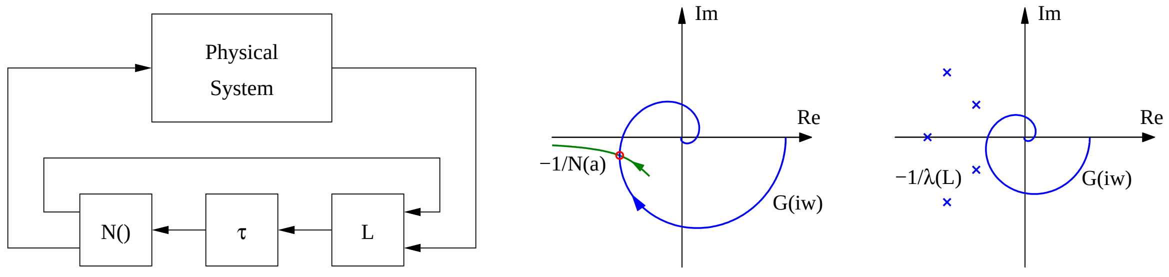
2.2 Design of dynamics

The first challenge in control design for cyberphysical systems using slow computing will be in the design of the (closed-loop) dynamics of the system. This element of the project focuses on inner loop shown in Figure 3.

Figure 4a shows the inner loop in a simplified form, consisting of an interconnection matrix L, a time delay matrix τ and a nonlinear map $N(\cdot)$. The interconnection matrix L is a simple linear operator that combines its two sets of inputs (the output of the sensor array and the feedback from the nonlinear map) to provide a vector of combined signals. The time delay matrix τ is a (square) multi-input multi-output operator that delays each of its signals by a nominal amount, but also models the asynchronous operation of the inner loop. Finally, the nonlinear map $N(\cdot)$ is a collection of single and multiple input nonlinearities that are used to tune the response of the system.

One of the appealing features of this particular interconnection structure is that special cases of it are well studied and characterized in the controls literature. For simplicity, we assume that the dynamics of the process can be modeled by its linearized dynamics about a desired operating point, which we denote

$$\dot{x} = Ax + Bu, \qquad y = Cx.$$



(a) Loop diagram (b) Describing function (c) Laplacian analysis

Figure 4: Loop analysis for nonlinear, networked systems. (a) A simplified view of the inner loop control system, omitting the GCL engine. (b) Stability analysis of the system with nonlinearities using describing functions. (c) Stability analysis of the system with an interconnection matrix, using the eigenvalues of the graph Laplacian to capture the effective loop gain.

The basic design problem is to find a control law u as a function of y that provides the desired dynamics for the closed loop. Ignoring the internal feedback loop for simplicity, the general form of the control law for our system is of the form

$$u_i = N_i(L_{jk}\, y_k(t - \tau_j)) \tag{1}$$

where τ_j is chosen from an appropriate distribution and N_i maps a vector of inputs to an appropriate output.

As one special case of this, if we remove the interconnection matrix L, the resulting system is in the form used for describing function analysis [1, pp. 288–290]. In this analysis, one searches for oscillatory solutions and replaces the nonlinearity by its first harmonic response, denoted $\mathcal{N}(\cdot)$. By plotting the frequency response of the system $P(i\omega) = C(i\omega I - A)^{-1}B$ in the complex plane and looking for intersections with the describing function curve $-1/\mathcal{N}(\cdot)$, as shown in Figure 4b, we can check for stability or the existence of limit cycles. There is a substantial literature on describing function analysis (and its generalization, the method of harmonic balance [30]) that we expect to provide design-oriented insights in our system.

Similarly, if we remove the nonlinearity N, we are left with a structure that has been studied in the context of cooperative control. In particular, if L is an interconnection matrix given by the graph Laplacian for the information flow and the process dynamics consists of set of identical linear systems, then the stability of the system can once again be determined by appealing to a Nyquist-like criterion. In this case, one plots $-1/\lambda_i(L)$ where $\lambda_i(L)$ is the ith eigenvalue of L. The system is stable if there are no net encirclements of any of the corresponding points in the complex plane [21].

This type of structure can also capture computations that have recently been explored in the context of bio-inspired approaches to control. One such approach, shown in Figure 5, makes of wide-field integration of sensory inputs to provide signals that are used to stabilize the dynamics of a vehicle [28]. The basic computation in the case of a planar vehicle is of the form

$$y_i = \langle \dot{Q}, F_i \rangle = \frac{1}{\pi} \int_0^{2\pi} \dot{Q}(\theta, x) \cdot F_i(\theta)\, d\theta, \tag{2}$$

where $\dot{Q}$ is the optical flow, F_i is a spatial filter (wide-field integration kernel), θ is the angle in the plane and $x \in \mathbb{R}^2$ represents the position of the vehicle. The environment is sensed through the optical flow $\dot{Q}$, which in turn depends on the location and speed of the vehicle in its environment.

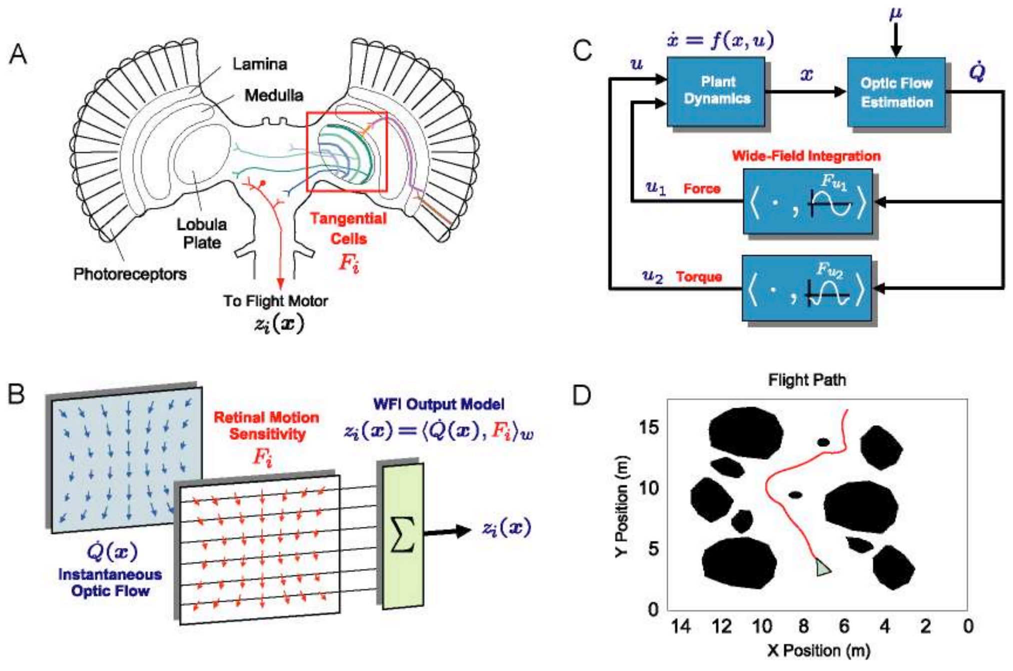


Figure 5: Bio-inspired flight control. (A) Insect visuomotor system. Wide-field retinal motion sensitive interneurons (tangential cells) parse spatially preserved visual information and transmit it to the flight motor. (B) Wide-field integration processing model. Spatial modes of optic flow $z_i(x)$ are extracted by retinal motion sensitivity kernels F_i. (C) Block diagram for static feedback of wide-field integration outputs. (D) Simulation of wide-field integration based visual navigation of a wheeled robot.

For a discretized sensor array, we can realize equation (2) as a linear sum that can be represented as $y = L\dot{Q}$. A linear controller with saturation can be formulated using additional linear combinations of these outputs, combined with appropriate time delays and nonlinearities.

A more geometric formulation of this type of sensory-driven control design has recently been developed by the PI and his students [11]. Here we consider the motion of a system on $SO(3)$ (orientations in $\mathbb{R}^3$) and assume that our sensor is defined by a map on the unit sphere, $m : S^2 \to \mathbb{R}$. The visual input can thus be written as $y(s,t) = m(R(t)s)$ where $s \in S^2$ is the spherical location of the sensor, t is time, $R(t)$ is the instantaneous orientation and m is the measurement map (intensities). We consider the problem of stabilizing a goal image g by appropriately setting the commanded angular torque T as a function of the error in the image space. Using this formalism, we are able to derive stabilizing controllers of the form

$$T = k_p(g^T My) + k_d(y^T(t-\tau)My), \tag{3}$$

where M represents an integration kernel (here written using a matrix style notation) and τ is a small time delay. We see that this controller is once gain in the form of equation (1) and hence we can implement it with a simple controller consisting of matrix multiplications and time delays. (What is even more interesting about this formalism is that if we set up a learning rule for M, then we can show that it has many of the features of the neural architecture of the insect visual system, including EMD-like operations between nearby visual elements.)

One of the interesting elements of this architecture that has not received much study is the use of the time delay as a dynamic element for tuning the dynamics of the controller. As illustrated

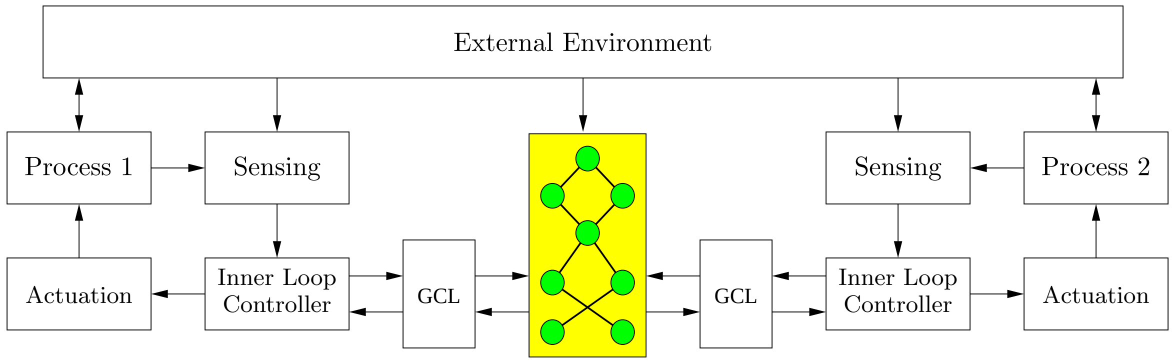


Figure 6: Framework for protocol-based control systems. Signals between control system modules for multiple processes are transmitted through a communication network.

in Figures 3 and 4a, we envision time delay as a central element in the architecture and it is the only dynamical element in our controller. Thus, we must use the combination of interconnection and time delay to obtain a design dynamic compensator as well as all other low level functionality. While this is straightforward to do in some sense (e.g., using a discrete-time representation of a continuous time system), as the delays begin to approach the time constants of the process to be controlled, we must use more intricate combinations of different time delays to obtain a desired input/output response. In addition, in an asynchronous environment the delays will not be precise and this "jitter" must be taken into account.

As already alluded to above, there are many open questions that remain to be explored in the design of dynamics using this formalism. New analysis tools must be developed to allow us to combine the combination of interconnection, time-delay and nonlinearity in the manner we have described and to allow uncertainty in the parameters describing these elements. In conjunction with the development of these analysis tools, specific design techniques must be formulated that allow a systematic process for constructing a system that satisfies an appropriate specification. In so doing, we plan to better understand the expressiveness of the proposed inner loop structure in terms of its ability to alter the dynamics of the underlying process.

2.3 Protocol-based control systems

Another important element in control design for cyberphysical systems using slow computing is the development of methods for interaction between multiple agents. We take the point of view here that such interactions will occur across information flow networks (including communication and sensing) and use the term "protocol-based" control to describe the resulting control system at this level of abstraction.

Figure 6 gives a possible view of the system, specializing some of the elements originally shown in Figure 3. We include here a more explicit role for the external environment, reflecting the fact that each agent senses its enviroment but also changes that environment (through its motion). Disturbances from the environment also come into play in many applications. We assume here that all communication between individual elements is done either via a communication network (using the GCL engine as its interface) or via the environment (through the sensing array).

Extensive work has been done on design of systems of this class, including work on the design of the information flow [21], cost/utility function approaches for cooperative control (see [40] for a survey) and the role of packet loss and time-delay [24, 43, 48]. We focus here on the asynchronous, protocol-based dynamics that govern the interaction between agents, following the formalism described in [40].

Let x^i represent the continuous state of the agent (for example, the location and speed of a vehicle) and u^i the input. We will assume that each agent also has a discrete state, α^i, which we define as the mode (or role) of the agent. The mode of the agent will be represented as an element of a discrete set $\mathcal{A}$ whose definition will depend on the specific control problem under consideration. As indicated by the terminology, we will generally consider the mode variable α^i to represent the portion of the agent's overall state that encodes its current actions and its relationship with the overall task being performed. We will assume that the mode of an agent can change at any time and we will write a change of mode as

$$\alpha' = r(x, \alpha),$$

where α' indicates the new value of α. We let $\alpha = (\alpha^1, \dots, \alpha^N)$ represent the modes of the collection of N agents and write $\alpha^i(t)$ for the mode of agent i at time t.

A *strategy* for a given task is an assignment of the inputs u^i for each agent and a selection of the modes of the agents. We will assume that the inputs to the agents' dynamics are given by control laws of the form

$$u^i = \gamma^i(x, \alpha)$$

where γ^i is a smooth function for simplicity, but which can be generalized to the inner loop for described earlier. For the choice of modes, we make use of the notion of a *guarded command language* (see [32]): a program is a set of commands of the form

$$\{g^i_j(x, \alpha) : r^i_j(x, \alpha)\}$$

where g^i_j is a guard that evaluates to either true or false and r^i_j is a rule that defines how the mode α^i should be updated if the rule evaluates to true. Thus, the mode evolves according to the update law

$$\alpha^{i\,\prime} = \begin{cases} r^i_j(x, \alpha) & \text{if } g^i_j(x, \alpha) = \text{true} \\ \text{unchanged} & \text{otherwise.} \end{cases}$$

This update is allowed to happen asynchronously.

We can also restrict this formalism to one in which the form of information flow is limited, for example in the case when sensors can only see for a limited range or in a certain direction. We model the set of communication channels by a graph $\mathcal{G}$ and write $\mathcal{N}^i(\mathcal{G})$ to represent the neighbors of agent i, that is, the set of agents that agent i is able to obtain information from (either by explicit communication or by sensing the information about the other agent). In general, $\mathcal{N}^i$ can depend on the locations and models of the agents, in which case we will write $\mathcal{N}^i(x, \alpha)$.

We say that a strategy is *centralized* if $\gamma^i(x, \alpha)$ or $g^i_j(x, \alpha) : r^i_j(x, \alpha)$ depend on the state or mode of any agent that is not a neighbor of i. A strategy is *decentralized* if

$$\begin{aligned} u^i(x, \alpha) &= u^i(x^i, \alpha^i, x^{-i}, \alpha^{-i}) \\ \{g^i_j(x, \alpha) : r^i_j(x, \alpha)\} &= \{g^i_j(x^i, \alpha^i, x^{-i}, \alpha^{-i}) : r^i_j(x^i, \alpha^i, x^{-i}, \alpha^{-i})\}, \end{aligned}$$

where we use the shorthand x^{-i} and α^{-i} to represent the state and modes of agent i's neighbors (hence $x^{-i} = \{x^{j_1}, \dots, x^{j_{m_i}}$ where $j_k \in \mathcal{N}^i$ and $m_i = |\mathcal{N}^i|\}$). We will mainly be interested in cooperative tasks that can be solved using a decentralized strategy.

We focus now on the evolution of the system mode α. To prove that a given specification for a system described by a guarded command language is satisfied, we must reason about the execution traces for the system, allowing any rule to be evaluated in any order. One specific technique that is well suited for analysis using these execution semantics is Lyapunov-based stability analysis. The

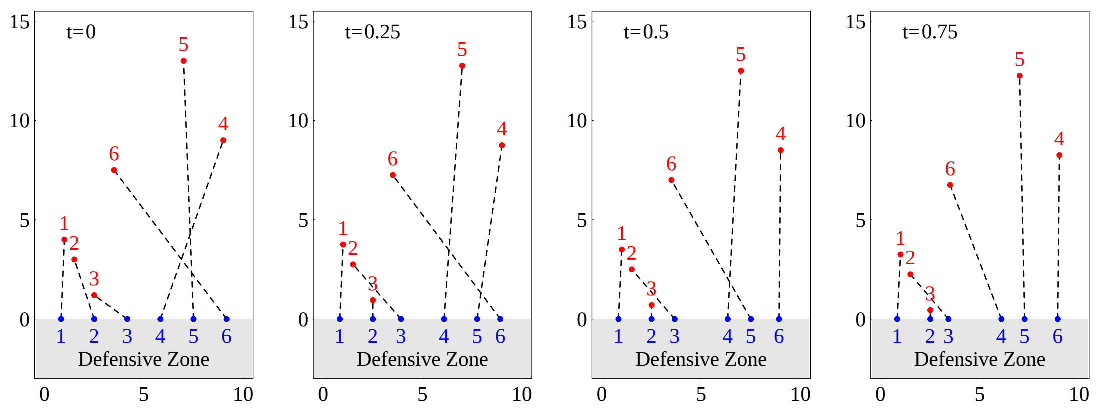


Figure 7: The RoboFlag Drill. The first four epochs of an execution of the CCL program along the x-axis represent blue defending robots. Other dots represent red attacking robots. Dashed lines represent the current assignment.

basic idea is to construct a Lyapunov function $V(x, \alpha)$ whose minimum value is obtained when the system is in the desired state and that satisfies

$$V'(x, \alpha) = V(x, r(\alpha)) \leq V(x, \alpha)$$

for any rule that can fire in a given state and mode. If we can in addition show that at least one rule will always fire that decreases V, then we can show that V goes to a minimum. The dynamics of the system can also be included, for example by describing them as a discrete time processes modeled in the guarded command framework (see [32] for details).

An example of this approach has been developed by Klavins [31] (while he was a postdoc at Caltech), who constructed a language for describing and verifying protocols for cooperative control. The computation and control language (CCL) uses the guarded command formalism to specify sets of actions for a collection of robots. Figure 7 gives an example of how a distributed area denial task can be solved in CCL. In this example, drawn from the RoboFlag game [15], 6 defensive robots are trying to protect a defense zone from an incoming set of robots, which descend vertically at a fixed speed. The defending robots must move underneath the incoming robots, but are not allowed to run into each other. The defenders are randomly assigned incoming robots and are allowed to talk to their neighbors and switch assignments under a given protocol. A protocol was developed by Klavins that is able to provably solve this problem, including ensuring that no two robots collide and that all defensive robots eventually end up assigned to an incoming robot with no crossing of assignments. Extensions to this approach for observability and controllability have also been developed [16, 17].

There are many open questions in this area that must be addressed before systematic techniques for GCL synthesis can be integrated into our candidate architecture. First, we must understand the effect of computational limits on GCL-based programs. While GCL-techniques are well suited for asynchronous execution, we do not currently have means of analyzing systems where multi-rate execution semantics might be used (so that some "high priority" rules are checked more often than others), nor do we have ways of specifying behavior that relies on more precise notions of time (though metric temporal logic [MTL] [33] and temporal logic of actions [TLA+] [35] are two possible starting points). And finally, the interaction between the protocol-based controller and the asynchronous, inner loop controller must be better formalized and cast into a design-oriented setting (e.g., how should the GCL-based controller change its mode as a function of the state of

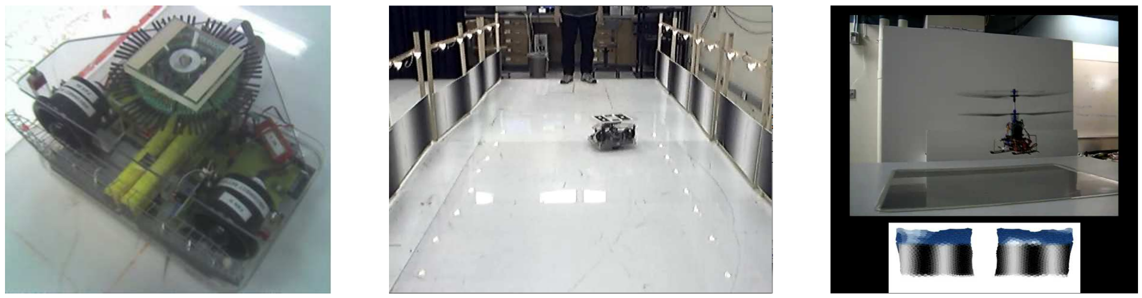

(a) "Flysight" (b) Flysight arena (c) Fsee rotorcraft

Figure 8: Caltech demonstration vehicles. (a) The Flysight platform consists of a planar vehicle driven by two fans, with an array of infrared sensors similar to insect ommatidia. (b) The vehicle can be operated in a number of different environments, including corridors with sinusoidally varying patterns on the wall (a standard environment for characterizing insect flight). (c) A helicopter based platform, interfaced to a real-time simulation of an insect's visual system, is also available for testing with more intricate vehicle dynamics. The image at the bottom is the simulated visual pattern provided to the vehicles bio-inspired control algorithm.

the underlying inner loop controller and vice versa).

2.4 Demonstration platform

To demonstrate the overall architecture and specific techniques that will be developed under this proposal, we plan to make use of several existing platforms available at Caltech. Figure 8 shows two of these platforms: a thrust-driven vehicle with a simple "compound eye" and a small, dual-rotor helicopter with a simulation-in-the-loop, insect vision system.

The "Flysight" testbed, shown in Figure 8a, incorporates a fan-actuated robot rolling on castors with a visual system that emulates the fly's. On Flysight we implemented a corridor-following algorithm that uses optic flow elementary motion detectors (EMD's) based on models from fly research (Figure 8b). The vehicle uses a simple microcontroller for the sensor interface and control algorithms are run on a Gumstix computer. The algorithm can successfully navigate the robot down the corridor under visual control.

A second platform that has been developed more recently makes use of a small, dual rotor helicopter interfaced to a real-time simulation of an insect visual system called "Fsee" (Figure 8c). This system differs from Flysight in a number of important ways: its motion is much more dynamic (and takes place in three dimensions, rather than two), the sensing array is much higher density (hundreds of individual visual sensors) and the computing is performed offboard (allowing more flexibility in the type of computational models that are tested). Initial work with this system has succeeded in stabilizing hover motion using (simulated) visual input and wide-field integration based controllers.

In addition to the individual vehicles described above, Caltech has built a testbed consisting of up to 12 mobile vehicles with embedded computing and communications capability for use in testing new approaches for command and control across dynamic networks [13]. The system allows testing of a variety of communications-related technologies, including distributed command and control algorithms, dynamically reconfigurable network topologies, source coding for real-time transmission of data in lossy environments, and multi-network communications.

2.5 Synergistic efforts

In addition to verifying the behavior of the system at the protocol-level, we must also eventually verify the overall behavior of the interconnected systems of systems (including the inner loop controller). This problem is one that many groups around the country are working on and, for the most part, we do not believe that there will be substantial differences between these problems for slow computing architectures versus traditional architectures. The PI is currently leading an AFOSR-sponsored MURI that has precisely this goal and the tools from that MURI (about to enter its fourth year of five) will be available for application in this domain. We briefly summarize here some of the work of the PI in this area as an indication of some of the tools that can be brought to bear.

We have begun to use tools from temporal logic to verify performance of complex, asynchronous systems such as the control logic in autonomous vehicles. In [51] we presented an approach that allows mission and contingency management to be achieved in a distributed and dynamic manner without any central control over multiple software modules. This approach comprises two key elements: a mission management subsystem and a Canonical Software Architecture (CSA) for a planning subsystem. The mission management subsystem works in conjunction with the planning subsystem to dynamically replan in reaction to contingencies. The CSA ensures the consistency of the states of all the software modules in the planning subsystem. System faults are identified and replanning strategies are performed in a distributed fashion by the planning and mission management subsystems through the CSA. The approach has been implemented and tested on Alice, an autonomous vehicle developed by Caltech students (and shown in Figure 1).

Fault tolerance and safety verification of control systems that have state variable estimation uncertainty are essential for the success of autonomous robotic systems. A software control architecture called Mission Data System, developed at the Jet Propulsion Laboratory, uses goal networks as the control program for autonomous systems. Certain types of goal networks can be converted into linear hybrid systems and verified for safety using existing symbolic model checking software [6, 7]. In [5], we present a process for calculating the probability of failure of certain classes of verifiable goal networks due to state estimation uncertainty. A verifiable example task is presented and the failure probability of the control program based on estimation uncertainty is found.

Finally, we are exploring the combination of receding horizon control with “proof by construction” techniques developed over the past few years for supervisory control systems. The basic idea is to efficiently explore all possible execution traces of a system and then eliminate those traces that do not satisfy a given temporal logic specification. The result can be stored as a high dimensional automaton whose executions necessarily satisfy the given specifications. While seemingly intractable, this technique is similar to trajectory generation techniques in control theory (e.g., [41]) and recent results have demonstrated some promising results [34].

3 Education and Mentoring Plan

We anticipate that in steady state the project will consist of 3–5 PhD students (some supported by outside fellowships), 2–4 undergraduate students (academic year and summer) and 1–2 visiting students (3–6 months each) working on this project. The graduate students would each work in one or more of the key technical areas described in the previous section, with undergraduate students and visiting students working on implementing specific designs and testing out new ideas. All students would work together on evolving the overall architecture and implementing their techniques on the demonstration platform.

The PI has a strong record of recruiting and mentoring both undergraduate and graduate students across the disciplines that are required for success in this program. Recent undergraduates, PhD students and postdocs include researchers in Computer Science (Klavins, Gillula), Control and Dynamical Systems (Bullo, Del Vecchio), Electrical Engineering (Gupta, Huang) and Mechanical Engineering (Kelly, Humbert). In addition, the PI was the team leader for Caltech DARPA Grand Challenge entries in 2004, 2005 and 2007, which involved students from all of the relevant disciplines required for success in this work.

The students who work on this program will be recruited through the Control and Dynamical Systems (CDS) program at Caltech. This highly interdisciplinary program includes an undergraduate minor, a graduate minor, a graduate PhD degree, established visiting programs with universities in the US and Europe, and substantial collaborations with other engineering and scientific disciplines at Caltech (including BE, CS, EE and ME). The CDS program has been very successful in recruiting a diverse group of students. Of the current set of CDS PhD students (including affiliated students who are essentially part of the CDS group), 37.5% are women, 56% are US citizens, and 33% of the US citizens are underrepresented minorities. The percentages of women and minorities are substantially higher than the average numbers for Caltech, partly due to the high rate of success in the program of attracting these highly sought after students to CDS.

The research activities that are proposed here will be naturally integrated into educational programs for both graduate students and undergraduates. The undergraduate controls course at Caltech (CDS 110) routinely includes an optional course project that is typically tied to existing research projects and for which this project would be an excellent example. Special topics courses (CDS 270-x) are offered on an annual basis to provide students with knowledge of the latest advances in areas of topic interest. These existing mechanisms provide a natural path for the transfer of knowledge beyond the students who are directly funded by the project.

4 Results of Prior NSF Support

The PI has recently completed an ITR grant that is related to the proposed effort and is a member of an NSF Expeditions team that may have some long term connections with the work described here.

Information Dynamics for Networked Feedback Systems (CCR-0326554) This four year project involved five faculty at Caltech (M. Effros, B. Hassibi, S. Low, R. Murray (PI), L. Schulman). The goal of this ITR was to develop theory, algorithms and experimental demonstrations for investigating the dynamics of information in complex, interconnected systems. Our work focused in four primary areas: (1) Real-time information theory for understanding fundamental limits of information flow in the presence of timing constraints and building a framework for the study of information dynamics; (2) Theory for robust control of networks to provide stable, high throughput data flow using source coding and feedback within and across protocol levels; (3). Packet-based control theory to allow feedback control of physical and information systems across networks, including issues of latency, multi-description coding, varying channel capacity, and feedback instabilities; (4) A framework for computational complexity of networked systems to understand the tradeoffs between computation, communication, and uncertainty in networked information processing systems.

Specific results of potential relevance to this program include the development of new techniques for state estimation over networks in the presence of packet loss [19, 29], design of network topologies for minimizing the usage of power while maintaining specified levels of performance in sensor networks [47], development of asynchronous protocols for distributed averaging across networks [38] and controls laws that respect information patterns determined by network structure [24].

Molecular Programming Project (CCF-0832824) The Molecular Programming Project (MPP) is an NSF Expedition to understand the principles and practice of molecular program. A molecular program is a collection of molecules that may perform a computation, fabricate an object, or control a system of molecular sensors and actuators. The best examples are the biomolecular programs of life—from the low-level operating system controlling cell metabolism to the high-level code for development, the process by which a single cell becomes an entire organism. Biomolecular programs are intrinsically parallel, distributed, asynchronous and error-prone, yet amazingly robust processes. The goal of the MPP is to develop the abstractions, languages and tools needed to design and implement artificial molecular programs. This expedition is led by Erik Winfree (Caltech) and is a five year expedition that started in 2008.

The inner loop architecture for slow computing described in this proposal shares many of the properties of molecular programs. Indeed, one approach to design of dynamics in molecular programs is through the manipulation of interconnections and time-delays, which are relatively easy to manipulate. While this proposal is not explicitly guided by molecular programming applications, there is a good possibility that some of the techniques developed here could also be used in the MPP.

References

[1] K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008. Available at `http://www.cds.caltech.edu/~murray/amwiki`.

[2] R. Alur, T. Dang, J. Esposito, Y. Hur, F. Ivancic, V. Kumar, P. Mishra, G.J. Pappas, and O. Sokolsky. Hierarchical modeling and analysis of embedded systems. *Proceedings of the IEEE*, 91(1):11–28, January 2003.

[3] P. J. Antsaklis. Special issue on hybrid systems: Theory and applications. *Proceedings of the IEEE*, 88(7), 2000.

[4] J. A. Bender and M. H. Dickinson. Visual stimulation of saccades in magnetically tethered *drosophila*. *Journal of Experimental Biology*, 209:3170–3182, 2006.

[5] J. M. B. Braman and R. M. Murray. Safety verification of fault tolerant goal-based control programs with estimation uncertainty. In *Proc. American Control Conference*, 2008.

[6] J. M. B. Braman, R. M. Murray, and M. D. Ingham. Verification procedure for generalized goal-based control programs. In *AIAA Infotech Conference*, 2007.

[7] J. M. B. Braman, R. M. Murray, and D. A. Wagner. Safety verification of a fault tolerant reconfigurable autonomous goal-based robotic control system. In *Proc. International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[8] R. W. Brockett. Reduced complexity control systems. In *Proc. IFAC World Congress*, 2007.

[9] M. Buehler, K. Iagnemma, and S. Singh. Special issues on the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 8–10, 2008.

[10] J. W. Burdick, N. duToit, A. Howard, C. Looman, J. Ma, R. M. Murray, and T. Wongpiromsarn. Sensing, navigation and reasoning technologies for the DARPA Urban Challenge. Available online at `http://gc.caltech.edu/media/papers/dgc07-final.pdf`, DARPA Urban Challenge Final Report, 2007.

[11] A. Censi, S. Han, S. Fuller, and R. M. Murray. A bioplausible arhictecture for vision-based attitude stabilization. In *Proc. IEEE Control and Decision Conference*, 2009. Submitted.

[12] K. Mani Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addison-Wesley, 1988.

[13] T. Chung, L. Cremean, W. B. Dunbar, Z. Jin, E. Klavins, D. Moore, A. Tiwari, D. van Gogh, and S. Waydo. A platform for cooperative and coordinated control of multiple vehicles: The caltech multi-vehicle wireless testbed. In *Conference on Cooperative Control and Optimization*, 2002.

[14] L. B. Cremean, T. B. Foote, J. H. Gillula, G. H. Hines, D. Kogan, K. L. Kriechbaum, J. C. Lamb, J. Leibs, L. Lindzey, C. E. Rasmussen, A. D. Stewart, J. W. Burdick, and R. M. Murray. Alice: An information-rich autonomous vehicle for high-speed desert navigation. *Journal of Field Robotics*, 23(9):777–810, 2006.

[15] R. D'Andrea and R. M. Murray. The RoboFlag competition. In *Proc. American Control Conference*, 2003.

[16] D. Del Vecchio. Discrete dynamic feedback for a class of hybrid systems on a lattice. In *IEEE International Symposium on Computer-Aided Control Systems Design*, 2006.

[17] D. Del Vecchio, R. M. Murray, and Erik Klavins. Discrete state estimators for systems on a lattice. *Automatica*, 2006.

[18] M. H. Dickinson. Personal Communication, 2009.

[19] M. Epstein, L. Shi, A. Tiwari, and R. M. Murray. Probabilistic performance of state estimation across a lossy network. *Automatica*, 44(12):3046–3053, 2008.

[20] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic mobile robots. *Automatica*, 45(2):343–352, 2009.

[21] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(5):1465–1476, 2004.

[22] G. Frehse. Phaver: algorithmic verification of hybrid systems past hytech. *International Journal on Software Tools for Technology Transfer*, 10:263–279, 2008.

[23] S. Fuller, A. Straw, M. H. Dickinson, and R. M. Murray. Gust response in *drosophila*. In preparation.

[24] V. Gupta, D. P. Spanos, B. Hassibi, and R. M. Murray. Optimal LQG control across a packet-dropping link. *Systems and Control Letters*, 56(6):439–446, 2007.

[25] E. Haghverdi, P. Tabuada, and G. J. Pappas. Bisimulation relations for dynamical, control, and hybrid systems. *Theoretical Computer Science*, 342(2–3):229–261, 2005.

[26] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley, 1991.

[27] G. J. Holzmann. *The SPIN Model Checker*. Addison-Wesley, 2003.

[28] J. Sean Humbert, R. M. Murray, and M. H. Dickinson. A control-oriented analysis of bio-inspired visuomotor convergence. In *Proc. IEEE Control and Decision Conference*, 2005.

[29] Z. Jin, V. Gupta, B. Hassibi, and R. M. Murray. State estimation utilization multiple description coding over lossy networks. In *Proc. IEEE Control and Decision Conference*, 2005.

[30] H. K. Khalil. *Nonlinear Systems*. Macmillan, 1992.

[31] E. Klavins. A language for modeling and programming cooperative control systems. In *Proceedings of the International Conference on Robotics and Automation*, 2004.

[32] E. Klavins and R. M. Murray. Distributed algorithms for cooperative control. *IEEE Pervasive Computing*, 3(1):56–65, 2004.

[33] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems*, 2(4):255–299, 1990.

[34] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Finding nemo: Reactive mission and motion planning. *IEEE Transactions on Robotics*, 2009. To appear.

[35] L. Lamport. *Specifying Systems*. Addison-Wesley, 2002.

[36] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35:349–370, 1999.

[37] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[38] M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, and R. M Murray. Asynchronous distributed averaging on communication networks. *IEEE/ACM Transactions on Networking*, 15(3):512–520, 2007.

[39] R. M. Murray, editor. *Control in an Information Rich World: Report of the Panel on Future Direcitons in Control, Dynamics and Systems*. SIAM, 2003. Available at `http://www.cds.caltech.edu/~murray/cdspanel`.

[40] R. M. Murray. Recent research in cooperative control of multi-vehicle systems. *ASME Journal of Dynamic Systems, Measurement and Control*, 129(5):571–583, 2007.

[41] R. M. Murray, J. Hauser, A. Jadbabaie, M. B. Milam, N. Petit, W. B. Dunbar, and R. Franz. Online control customization via optimization-based control. In T. Samad and G. Balas, editors, *Software-Enabled Control: Information Technology for Dynamical Systems*. IEEE Press, 2003.

[42] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[43] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 2004.

[44] S. Owre, J. Rushby, N. Shankar, and D. Stringer-Calvert. PVS: an experience report. In Dieter Hutter, Werner Stephan, Paolo Traverso, and Markus Ullman, editors, *Applied Formal Methods—FM-Trends 98*, volume 1641 of *Lecture Notes in Computer Science*, pages 338–345. Springer-Verlag, 1998.

[45] M. B. Reiser, J. S. Humbert, M. J. Dunlop, D. Del Vecchio, R. M. Murray, and M. H. Dickinson. Vision as a compensatory mechanism for disturbance rejection in upwind flight. In *Proc. American Control Conference*, 2004.

[46] C. L. Robinson, G. Baliga, and P. R. Kumar. Design patterns for robust and evolvable networked control. In *Third Annual Conference on Systems Engineering Research*, 2005.

[47] L. Shi, K. H. Johansson, and R. M. Murray. Optimal sensor hop selection: Sensor energy minimization and network lifetime maximization with guaranteed system performance. In *Proc. IEEE Control and Decision Conference*, 2008.

[48] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49:1453–1464, 2004.

[49] G. Stépán. *Retarded dynamical systems*. Longman, 1989.

[50] T. Wongpiromsarn, S. Mitra, R. M. Murray, and A. Lamperski. Periodically controlled hybrid systems: Verifying a controller for an autonomous vehicle. In *Hybrid Systems: Computation and Control*, 2009. To appear.

[51] T. Wongpiromsarn and R. M. Murray. Distributed mission and contingency management for the darpa urban challenge. In *International Workshop on Intelligent Vehicle Control Systems*, 2008.