Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2009, Article ID 984752, 9 pages
doi:10.1155/2009/984752

Research Article

Vector Field Driven Design for Lightweight Signal Processing and Control Schemes for Autonomous Robotic Navigation

Nebu John Mathai, Takis Zourntos, and Deepa Kundur

Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77840, USA

Correspondence should be addressed to Nebu John Mathai, mathai@ieee.org

Received 31 July 2008; Revised 26 February 2009; Accepted 8 April 2009

Recommended by Frank Ehlers

We address the problem of realizing lightweight signal processing and control architectures for agents in multirobot systems. Motivated by the promising results of neuromorphic engineering which suggest the efficacy of analog as an implementation substrate for computation, we present the design of an analog-amenable signal processing scheme. We use control and dynamical systems theory both as a description language and as a synthesis toolset to rigorously develop our computational machinery; these mechanisms are mated with structural insights from behavior-based robotics to compose overall algorithmic architectures. Our perspective is that robotic behaviors consist of actions taken by an agent to cause its sensory perception of the environment to evolve in a desired manner. To provide an intuitive aid for designing these behavioral primitives we present a novel visual tool, inspired vector field design, that helps the designer to exploit the dynamics of the environment. We present simulation results and animation videos to demonstrate the signal processing and control architecture in action.

1. Introduction

The problem of developing a control architecture for autonomous robotic agents involves numerous challenges pertaining to the best use of limited, nonideal information. Beyond this, given the remote, energy-scarce environments that robots have found application (e.g., space robotics, underwater exploration, mobile sensor networks deployed in inhospitable, unknown terrain) and the multiagent robotic paradigm, the need for signal processing with lightweight implementation (in terms of area and power complexity, and low-latency autonomous computation) has become increasingly important.

To minimize the economic cost of a multiagent system, it is important that the complexity of each agent be constrained. Moreover, in robotic exploration problems (where the agent must be able to maneuver effectively through challenging and inaccessible environments) and mobile sensor network applications, low agent complexity (e.g., in terms of compactness and energy usage) is demanded. Further, it has been suggested [1] that robotics, the endeavor of synthesizing artificial goal-directed machines, may offer insight to biology, the study of goal-directed organisms in nature. To that end, the development of synthesis methods for autonomous machines that aim to approach the economy of nature could be useful.

1.1. Why Analog Computation? Generally, the need for lightweight signal processing suggests the use of special purpose computers, as in the case of using a digital signal processor over a general purpose one to implement numerically-intensive algorithms. Taking this idea of application-specific processing hardware to the extreme, we are led to custom realizations where the operations required by the algorithm are mapped as directly as possible to the computing primitives provided by the implementation technology.

Of particular interest to us are custom analog systems, due to (1) the plethora of innate physical characteristics that can be exploited to obtain low-cost signal processing primitives (e.g., Kirchoff's current law can be used to realize an adder "for free"), (2) the reduced wiring complexity (e.g., for a 50 dB signal-to-noise ratio, an analog system requires one or two wires to convey signals, whereas a digital system requires eight wires), and (3) the ability to fine-tune the hardware at a very low level (for VLSI realizations, which are

preferable [2]). An excellent overview of the relative merits of analog and digital implementations of signal processing systems can be found in [3, 4]; in general, analog systems confer their greatest advantage for processing that requires moderate signal-to-noise ratios—levels that are pertinent to robotic control where noisy, nonlinear sensors restrict the fidelity of measurements of environmental data. Recent results from the field of neuromorphic engineering [5–9] demonstrate the efficacy of analog processing systems, from the perspective of functionality and economy of implementation. Hence, inspired by this, we consider analog-amenable signal processing and control architectures.

To that end, we need a principled means of synthesizing analog machines. Connectionist [10] and empirical [11] methods of realizing analog computation exist; however, the lack of a rigorous synthesis methodology is a drawback. In contrast, cybernetics—control theory and dynamical systems theory [12–15]—offers rigorous toolsets that enable the synthesis of analog automata. First, the continuous methods of control theory are an appealing match for agents coping with a physical environment that is, at practical scales of perception, continuous. Beyond this, the use of control theory can be viewed as the analog complement to the digital situated automata approach [16]—both use dynamical systems-based descriptions of the world, and rigorous synthesis toolsets to develop formulations of computational machinery that can be closely mapped to an implementation technology.

1.2. Contributions. In this work, we address the problem of realizing lightweight cognitive faculties for agents in multi-robot systems. Specifically, we extend the work of [17–19] in two directions: (1) towards purely reactive (i.e., memoryless) analog behaviors, and (2) towards multi-agent systems. We use control and dynamical systems theory both as a description language and as a synthesis toolset to realize signal processing schemes amenable to analog implementation. These mechanisms are mated with structural insights from behavior-based robotics to compose the overall control architecture.

We present the use of a novel visual tool—vector field design—to address the synthesis of reactive behaviors for single agent and multi-agent navigation; the tool is based on a dynamical model of the embodied agent-environment system, and it enables the roboticist to design behaviors that exploit these dynamics. A reactive action selection scheme is used to "stitch" together these behavioral controllers; simulation results of the resulting composite system are presented.

We note that vector field design has been seen in the context of computer graphics. In [20], a rigorous framework is developed for synthesizing vector fields with desirable properties; these vector fields are then computed, online, to assist with computer graphics and image processing applications. The proposed work, by contrast, has distinct challenges due to the lightweight processing requirements of practical field robotics. Hence, in the proposed work, we employ vector fields only at design time (or "compile time") in order to eliminate the cost of computing a spatially-extended two-dimensional function as a function of real-time sensor information. At run time, the product of this vector field *driven* design—a control law—is used to implement various robotic behaviors.

2. Preliminaries

2.1. Problem Formulation. Consider an autonomous navigation problem where a population of agents must reach a target. Moreover, we want the agents to self-organize into a spatially-distributed configuration in a region about the target (e.g., for a sensor network application, we would like to form a connected network of agents that covers this region). Since we desire lightweight signal processing and cognition, we assume that (1) the agent only has access to local information about its environment via short-range sensing faculties, (2) the agent does not have a priori information about the environment, and (3) the agent cannot use active communication to coordinate with other agents. Regarding the third point, we note that in many applications (e.g., in hostile environments), a communications channel may not always be available, and if one is we often want to maximize bandwidth for other, more pertinent uses (e.g., execution of distributed sensor fusion algorithms).

We note here that, in general, the design of control schemes for multi-agent systems is not restricted solely to physically-embodied robotic agents with such limited perceptual faculties. For example, in the computer graphics community, information which is not limited to physically-grounded local sensors can be used to great effect in achieving realistic, globally-optimal results as in [21].

2.2. Machine Organization. Robotic agents are situated in physical environments where they must contend with concurrent phenomena with dynamics over multiple time scales. Subsumption [22] is a structure for the design of reactive systems (i.e., systems where physically-grounded cognition [23] realizes a tight coupling between sensation and actuation) where a partititioning of functionality into levels of competance addesses the multi-scale nature of the world, and layering of control addresses parallelism. Behavior-based robotics [24] views the development of functionality in terms of the design of elementary behaviors (primitives for guiding the agent based on specifying temporally-extended action trends that tend to bring the agent to favorable circumstances) that can be combined—through an action selection [25] strategy—to realize more sophisticated composite behaviors.

In [17, 18] an analog subsumption architecture was presented—illustrated in Figure 1—in which the nesting of rigorously-derived control loops addressed the multi-scale nature of the environment. In the following we address the problem of designing concurrent behavioral primitives for the navigation layer (C_1/E_1); for brevity, in this work we subsume the competence provided by the (C_0/E_0) layer by assuming that velocity commands from C_1 are realized instantaneously by C_0. The time-scale separation between C_1/E_1 (slower) and C_0/E_0 (faster) justifies this.

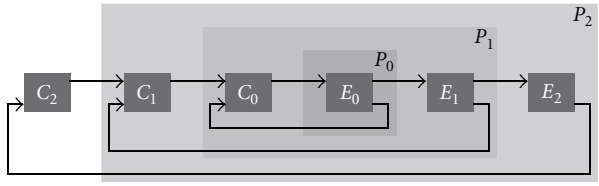


FIGURE 1: Nesting of controllers coupled to the environment; controller C_i regulates its sensory perception of the environment, E_i. The derivation of C_i considers a plant model, P_i, of the world "downstream" from it according to the recursion $P_0 := E_0$ and $P_i := C_{i-1}P_{i-1}E_i$.

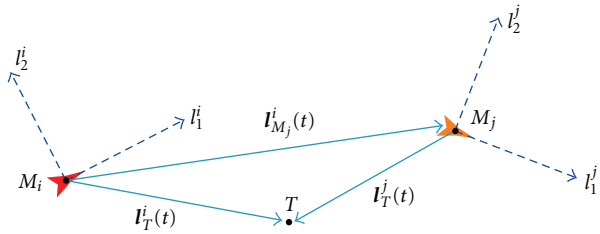


FIGURE 2: Two agents, M_i and M_j, whose respective local coordinate systems are specified by the l_1^k and l_2^k axes ($k \in \{i, j\}$). The displacements from each agent to the common target T, that is, $\mathbf{l}_T^k$ ($k \in \{i, j\}$), as well as the displacement from M_i to M_j, that is, $\mathbf{l}_{M_j}^i$, are shown.

2.3. Embodiment Details. Since the agent is coupled to the world by sensors that convey information from the environment and actuators that enable it to effect change to the environment, we first specify the details of the agent's sensori-motor embodiment.

2.3.1. Tracking Sensors. Consider an agent, M_i, in a planar world, to which a local frame of reference is attached, and let l_1^i and l_2^i denote the axes of a rectangular coordinate system in this frame with the agent at the origin. The local sensing faculties of the agent provide measurements of displacements between the agent and a target of interest with respect to this local coordinate system (with the agent at the origin). Figure 2 illustrates the case of an agent, M_i, sensing another agent, M_j, and where both agents sense a common target, T. Since practical sensors are nonideal measuring devices, these displacement measurements will be subject to various forms of distortion. We first set a minimum standard on the fidelity we expect from our sensors.

Definition 2.1 (measurement functions). Let:

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{for } x < 0, \\ 0 & \text{for } x = 0, \\ +1 & \text{for } x > 0, \end{cases} \tag{1}$$

$$\operatorname{sgn}^+(x) = \begin{cases} -1 & \text{for } x < 0, \\ +1 & \text{for } x \geq 0, \end{cases}$$

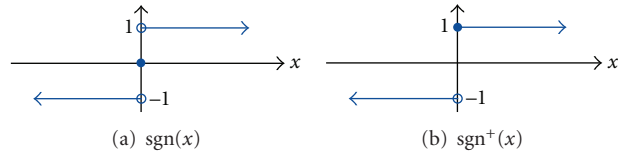


FIGURE 3: The signum definitions used in this work.

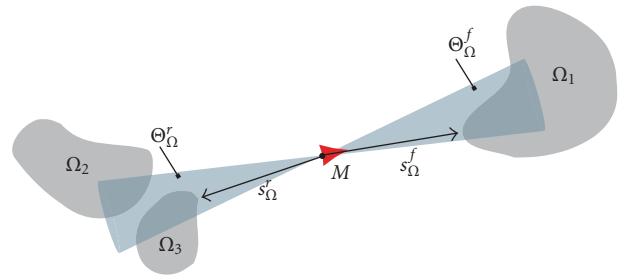


FIGURE 4: Specification of the obstacle sensor, where Ω denotes an obstacle.

as illustrated in Figure 3. The map $\sigma : \mathcal{R} \to \mathcal{R}$ is a measurement function if it is a bounded, continuous, bijection such that for all $x \in \mathcal{R}$, $\operatorname{sgn}(\sigma(x)) = \operatorname{sgn}(x)$.

Let $\eta = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix}$ denote the displacement between the agent and an object of interest. A sensor, S, is a memoryless system that returns its measurement of the position of this object, $\mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} \sigma_1(\eta_1) \\ \sigma_2(\eta_2) \end{bmatrix}$, where σ_1 and σ_2 are arbitrary measurement functions.

2.3.2. Obstacle Sensors. We specify minimal sensory apparatus to provide the agent with information regarding the presence of obstacles in the agent's local environment. Consider the situation shown in Figure 4. The agent, M, has short range sensors (with range $r_\Omega^{\max}$) at the front and rear of it that point along the l_1 axis of the agent's local frame of reference. Let the set Θ_Ω^f be a sector emanating from the agent's position that contains the positive l_1 axis. Similarly, the set Θ_Ω^r is a sector emanating from the agent's position that contains the negative l_1 axis. Let r^f and r^r denote the distance to the closest obstacle that is within the sectors Θ_Ω^f and Θ_Ω^r, respectively. Further, let $\sigma : \mathcal{R} \to [0, 1]$ be a continuous, bounded, monotonic decreasing function such that $\sigma(0) = 1$ and $\sigma(x) = 0 \Leftrightarrow x \geq r_\Omega^{\max}$. We define the forward obstacle sensor as a memoryless device that returns $\sigma(r^f)$, and the reverse obstacle sensor as a memoryless device that returns $\sigma(r^r)$.

2.3.3. Actuators. In this work we deal with behaviors for navigation, and so subsume the competence provided by a lower-level motor controller. We assume that the underlying vehicle kinematics are those of the simple unicycle model [26], where the motion of the vehicle is described by its signed translational speed, v, and its signed rotational speed, ω. The controllers we will synthesize actuate change by

specifying two independent motion commands—a_v and a_ω for translation and rotation, respectively—which are, effectively, instantaneously realized by the low-level motor controller (hence, we will model the effect of the low-level motor controller—which operates on a faster time scale than the navigation controller—by an identity operator taking a_v to v, and a_ω to ω). We note that positive a_v translates the agent's local frame of reference in the direction of the $l_1^i > 0$ ray, and positive a_ω rotates the frame in a counter-clockwise sense.

3. Synthesis of Behaviors

In this work, we address the problem of realizing robotic behaviors via agent-level signal processing schemes amenable to analog implementation. Our perspective is to make an association between behavior and sensor output regulation, that is, *we view behaviors as actions taken by an agent to cause its sensory perception of the environment to evolve in a desired manner.*

Casting the problem of behavioral design in control theoretic terms then, we need a model that describes how the agent's sensory perception of the world evolves with its actuation. Let $\boldsymbol{\eta}$ denote the actual displacement of an agent to a target of interest (e.g., a general target or another agent). Given the details of embodiment in Section 2.3, we can derive the plant model, P:

$$P : \begin{cases} \dot{\boldsymbol{\eta}} = \mathbf{p}(\boldsymbol{\eta}, \mathbf{a}) := \Upsilon(\boldsymbol{\eta})\mathbf{a} \\ \mathbf{s} = \begin{bmatrix} \sigma_1(\eta_1) \\ \sigma_2(\eta_2) \end{bmatrix}, \end{cases} \tag{2}$$

where

$$\Upsilon(\boldsymbol{\eta}) = \begin{bmatrix} -1 & \eta_2 \\ 0 & -\eta_1 \end{bmatrix} \tag{3}$$

and σ_1 and σ_2 are arbitrary measurement functions.

Now our task is to design a feedback control law, $\mathbf{a}(\boldsymbol{\eta})$, such that the resulting closed loop system:

$$\dot{\boldsymbol{\eta}} = \mathbf{p}(\boldsymbol{\eta}, \mathbf{a}(\boldsymbol{\eta})) := \hat{\mathbf{p}}(\boldsymbol{\eta}) \tag{4}$$

has the qualitative properties we desire, namely, we want $\boldsymbol{\eta} = 0$ (corresponding to zero displacement to the target of interest) to be a globally asymptotically stable equilibrium.

There are a variety of techniques that can be used to derive control laws for (2); we focus on the use of a visual tool, *vector field design* that appeals to the intuition in a manner we describe below. Recall that an n-dimensional vector field is a map $\mathbf{f} : \mathcal{R}^n \to \mathcal{R}^n$. When used as the right hand side of an ordinary differential equation (e.g., $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, $\mathbf{x} \in \mathcal{R}^n$) the vector field specifies how the states, $\mathbf{x}(t)$, evolve in time (i.e., how the trajectory $\mathbf{x}(t)$ "flows" through the state space $\mathcal{R}^n$ with respect to time). Hence the vector field describes the qualitative behavior of the system. Vector field design has proved to be a useful tool in diverse contexts where a dynamical systems formulation of the problem is natural, including computer graphics [20] and the design of chaotic oscillators [27].

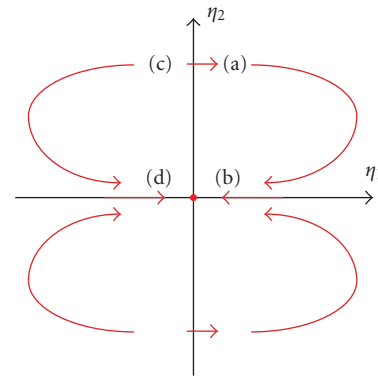


Figure 5: Structure of a candidate vector field for unconstrained taxis.

Our application of this toolset is similar to that of [27] where vector field design is only used at compile time as an aid to synthesize the run time control laws. Specifically, in the following we present the construction of reference vector fields, $\hat{\mathbf{p}}(\boldsymbol{\eta})$, that describe desirable sensor output dynamics that correspond to the robotic behaviors we are designing. Using these reference vector fields, we derive $\mathbf{a}(\boldsymbol{\eta})$ so that $\mathbf{p}(\boldsymbol{\eta}, \mathbf{a}(\boldsymbol{\eta})) = \hat{\mathbf{p}}(\boldsymbol{\eta})$—bringing the actual sensor output dynamics in compliance with the reference dynamics.

Before proceeding, we note that the vector fields we will be presenting are defined in terms of $\boldsymbol{\eta}$, which is in a coordinate system local to the agent and represents the relative displacement of the target with respect to the agent. The state $\boldsymbol{\eta} = 0$ corresponds to the condition where the agent's displacement to the target of interest is zero. For example, if we design a vector field where all states eventually flow to the goal state $\boldsymbol{\eta} = 0$, we will obtain an actuation law that corresponds to the robotic behavior of taxis.

3.1. Unconstrained Taxis. Here we present the construction of a reference vector field for taxis (target tracking behavior) where the agent's actuation is unconstrained. We first identify the qualitative properties that are required of $\hat{\mathbf{p}} = \begin{bmatrix} \hat{p}_1 : \mathcal{R}^2 \to \mathcal{R} \\ \hat{p}_2 : \mathcal{R}^2 \to \mathcal{R} \end{bmatrix}$. To globally asymptotically stabilize $\boldsymbol{\eta} = 0$ we must ensure $\boldsymbol{\eta} = 0$ is an equilibrium point (i.e., $\hat{\mathbf{p}}(\boldsymbol{\eta}) = 0 \Leftrightarrow \boldsymbol{\eta} = 0$) and that the trajectories induced by $\hat{\mathbf{p}}$ flow to $\boldsymbol{\eta} = 0$. Additionally, to facilitate the derivation of a control law we require the structure of $\hat{\mathbf{p}}$ be compatible with the plant model, that is, for all $\boldsymbol{\eta} = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix}$ such that $\eta_1 = 0$ we have $\hat{p}_2(\boldsymbol{\eta}) = 0$ (if this is not the case, then singularities in the control law will arise when $\eta_1 = 0$).

Figure 5 illustrates the qualitative structure of a vector field that satisfies these requirements. The behavior it implies (of which some representative cases are shown in Figure 6) is intuitively appealing. Trajectories flow to the η_1 axis, indicating that the agent acts to bring the target in front of (Figure 6(a)) or behind (Figure 6(c)) the agent; once this is achieved, the agent then closes in on the target (Figures 6(b) and 6(d), resp.).

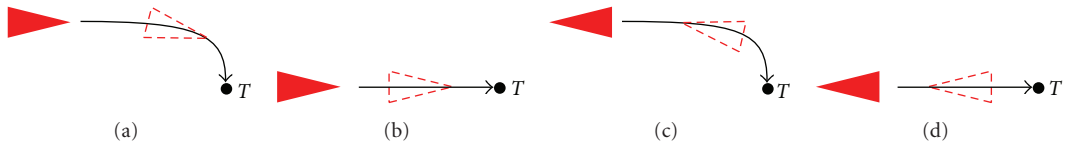


Figure 6: Behavior specified by the reference vector field of Figure 5.

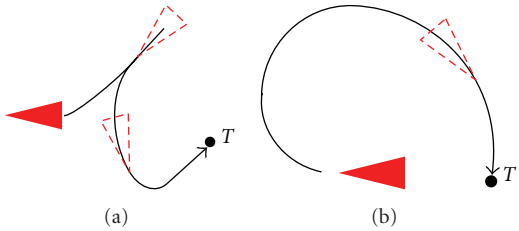


Figure 7: Behaviors specified by a reference vector field that biases forward motion (a), and uses only forward motion (b).

The flow of Figure 5 can be realized by:

$$\hat{\mathbf{p}} : \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \longmapsto \begin{bmatrix} -\mathrm{sgn}(\eta_1) + \mathrm{sgn}^+(\eta_1)\,|\,\eta_2\,| \\ -\mathrm{sgn}(\eta_2)\,|\,\eta_1\,| \end{bmatrix}. \tag{5}$$

Setting (2) and (5) equal, we obtain:

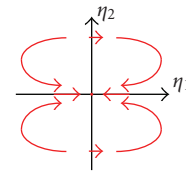
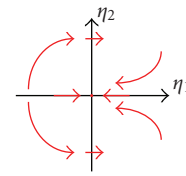
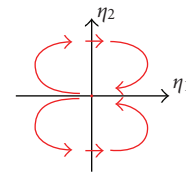
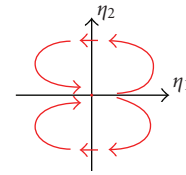
$$\mathbf{a} = \begin{bmatrix} \mathrm{sgn}(\eta_1) \\ \mathrm{sgn}^+(\eta_1)\mathrm{sgn}(\eta_2) \end{bmatrix} = \begin{bmatrix} \mathrm{sgn}(s_1) \\ \mathrm{sgn}^+(s_1)\mathrm{sgn}(s_2) \end{bmatrix} \tag{6}$$

(recall σ_1, σ_2 are measurement functions that preserve the signum of their arguments).

3.1.1. Biased Taxis. Suppose we wish to design a taxis behavior which, although unconstrained, is biased towards moving forwards towards the target (e.g., for agents which have the capability to reverse, but prefer—as most car drivers—forward motion where possible). Observe that in the second vector field of Table 1, all trajectories (except the ones where the target is directly behind the agent, i.e., $\eta_1 < 0$ and $\eta_2 = 0$) tend to flow towards the $\eta_1 > 0$ axis (i.e., where the target is ahead of the agent) and from there flow to the desired $\boldsymbol{\eta} = 0$ state. Figure 7(a) illustrates the actions of an agent that is regulating its sensor output according to these behavioral specifications. The agent reverses until it senses the target at an angle of $\pi/2$ (corresponding to a vector field trajectory hitting the η_2 axis from the left), moves to bring the target in front of the agent (corresponding to trajectories flowing towards the η_1 axis), and then closes in on the target.

3.2. Constrained Taxis. Constraints on the actions of an agent can be due to inherent limitations of the agent (e.g., the inability to move backwards) or imposed by external phenomena (e.g., obstacles in the agent's path). Consider the vector field illustrated in the third row of Table 1. The structure of this field indicates that all trajectories flow away from the region where $\eta_1 < 0$, towards the region where

Table 1: Summary of reference vector fields, $\hat{\mathbf{p}}(\boldsymbol{\eta})$.

Behavior	Desired vector field $\hat{\mathbf{p}}(\boldsymbol{\eta})$	Analytic form $\hat{\mathbf{p}}(\boldsymbol{\eta})$
Unconstrained taxis		$\begin{bmatrix} -\eta_1 + \mathrm{sgn}^+(\eta_1)\lvert\eta_2\rvert \\ -\mathrm{sgn}(\eta_2)\lvert\eta_1\rvert \end{bmatrix}$
Unconstrained taxis (forward bias)		$\begin{bmatrix} -\mathrm{sgn}(\eta_1) + \lvert\eta_2\rvert \\ -\eta_1\mathrm{sgn}(\eta_2) \end{bmatrix}$
Forward-only taxis		$\begin{bmatrix} -\lvert\eta_1\rvert + \lvert\eta_2\rvert \\ -\eta_1\mathrm{sgn}(\eta_2) \end{bmatrix}$
Reverse-only taxis		$\begin{bmatrix} \lvert\eta_1\rvert - \lvert\eta_2\rvert \\ \eta_1\mathrm{sgn}(\eta_2) \end{bmatrix}$

$\eta_1 > 0$, and from there flow to $\boldsymbol{\eta} = 0$. That is, the agent acts to bring the target in front of it, and then closes in, as illustrated in Figure 7(b). Hence, this field specifies target tracking by forward motion only. Reversing the direction of the vectors of this field, we obtain the fourth vector field of Table 1, which, by similar observations, specifies target tracking by purely reverse motion.

3.3. Antitaxis. To realize anti-taxis, that is, motion away from a target of interest, we note that this corresponds to driving $\boldsymbol{\eta}$ away from 0, to infinity. We can derive an anti-taxis vector field, $\hat{\bar{\mathbf{p}}}(\boldsymbol{\eta})$, by taking a base vector field like that of the second row of Table 1 and reversing the direction of flow: $\hat{\bar{\mathbf{p}}}(\boldsymbol{\eta}) := -\hat{\mathbf{p}}(\boldsymbol{\eta})$.

3.4. Comments. Table 1 summarizes the reference vector fields for taxis discussed in this section. Each vector field, in turn, gives rise to a robotic behavior when the corresponding control law is derived and used to specify velocity commands

for the agent. It is important to stress that these vector fields are used at *design time* to generate control laws that are employed by the robot at *run time*. Hence, the agent, when in action in the field, selects from a set of control laws, and not vector fields. Due to space restrictions, we do not present every control law (the actuation laws can be derived by setting $\mathbf{p}(\boldsymbol{\eta}, \mathbf{a}(\boldsymbol{\eta})) = \hat{\mathbf{p}}(\boldsymbol{\eta})$ and solving for $\mathbf{a}$); however, we note that these behavioral specifications give rise to purely reactive laws, which are amenable to very economical implementation. The economy of implementation of this compile time approach is seen more readily when we consider the computational load on the agent due to two scenarios: (1) computing vector fields at run time, or (2) computing control laws at run time. With the former, the agent would need to evaluate a two dimensional function over several points that adequately sample the state space; with the latter, it need only evaluate the control law at a single point—an operation requiring no memory or state, and, for the signum nonlinearities we employ, only requiring simple feedforward functions, for example, (6).

We also note that Table 1 presents more behaviors than are strictly needed for general taxis with the robotic kinematic model we employ in this work (i.e., one in which the robot can translate in the forward and reverse directions, and steer). For the embodiment we consider, there are four basic cases.

(1) The robot's translational motion is not impeded.

(2) Only the robot's forward translational motion is impeded.

(3) Only the robot's reverse translational motion is impeded.

(4) The robot's forward and reverse translational motion are both impeded.

For case (1), any of the four vector fields are sufficient, while for cases (2) and (3), the reverse-only and forward-only behaviors, respectively, are necessary. Case (4) is out of the scope of taxis behavior, since the agent is unable to immediately engage in any translational motion to get to the target: it must first free itself of the forward and/or reverse motion impediments. This requires it to engage in, for example, searching, as discussed in the next section. Hence, for the pure taxis behavior of cases (1)–(3), only two basic behaviors need be instantiated in the agent: forward-only and reverse-only taxis (the other behaviors are not useless; indeed, unconstrained taxis with forward bias is a simple model of how a human car driver operates under normal circumstances).

4. Action Selection and Simulation Results

4.1. Single Agents. We wish to "stitch" together the schemes presented in the preceding section to realize useful composite behaviors. Since the focus of this paper is on analog behavioral synthesis, for brevity we provide an overview of a technique for action selection and refer the reader to [17] where the synthesis of such a controller is presented in greater detail. We first construct a "virtual sense" that represents the level of urgency (analogous to the activation level of [25]) of situations that the agent is facing. Consider the case of deciding whether to employ forward taxis, reverse taxis, or unconstrained taxis. We can perform a "leaky integration" on the obstacle sensor outputs (e.g., using the system $\dot{\xi} = -\kappa\xi + u$, $y = \xi$, where ξ is the state of the filter, and u and y are the corresponding inputs and outputs) and then pass it through a hysteresis function. The output of this processing gives an indication of whether an obstacle sensor is being over-stimulated or not, which provides the required feedback for a controller to select an appropriate mitigating behavior.

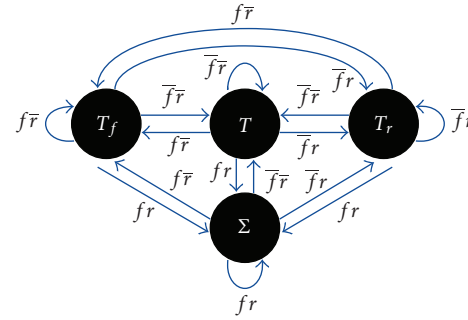
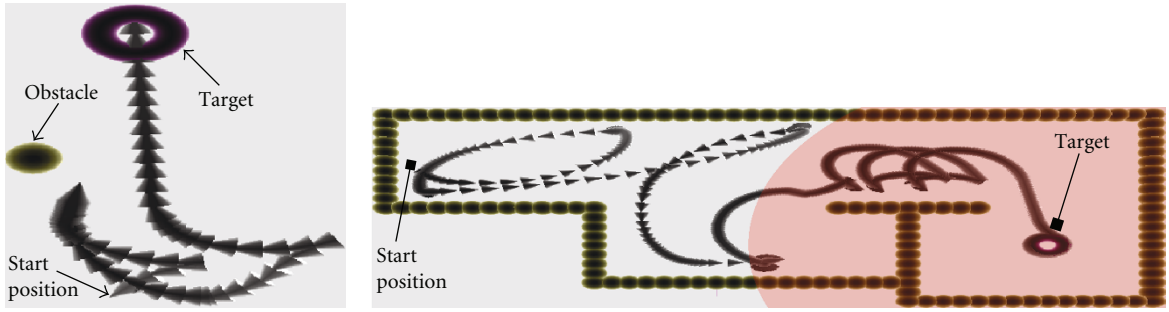


FIGURE 8: A finite state acceptor that describes the action selection scheme; T, T_f, and T_r represents unconstrained, forward-only and reverse-only taxis, respectively, while Σ is a searching behavior for the fall through case when neither T_f nor T_r can be used. The virtual senses f and r indicate that the forward and reverse obstacle sensors, respectively, are overstimulated, while $\bar{f}$ and $\bar{r}$ indicate the absence of overstimulation; the virtual senses are ANDed together to specify FSA transitions.

Figure 8 shows a finite state acceptor that describes the operation of our action selection controller (we stress that this FSA is used for descriptive purposes—the actual action selection mechanism is a feedback controller). Figure 9(a) presents simulation results of the agent avoiding an obstacle while tracking a target (the appendix provides details of the simulation methodology); unconstrained taxis (T) is first engaged, but the agent switches to the taxis-by-reversal controller (T_r) when confronted by the obstacle. After getting far enough away from the obstacle for the over-stimulation virtual sensors to relax, it re-engages unconstrained taxis behavior (T) to the target. Figure 9(b) illustrates the agent in a more complex obstacle ridden environment in which the target is initially out of sensor range (out of the shaded region about the target). It starts to search (using a reference oscillator [18] to cause it to execute a spiral traversal of space) until it senses the target, at which point it engages in various forms of constrained taxis (when near obstacles that impede its path) and unconstrained taxis (when free of obstacles) to get to the target. Searching behavior guarantees that the robot will eventually escape trapping regions or regions wherein it cannot sense the target, since the space-filling spiral search pattern will cause the agent to eventually traverse all space. For a lightweight agent with limited sensing faculties—whether a living organism or a robot—this is, of



(a) Taxis and obstacle avoidance

(b) Taxis, searching, and obstacle avoidance; the shaded region indicates the space within which the agent can sense the target

Figure 9: Single agent simulation results.

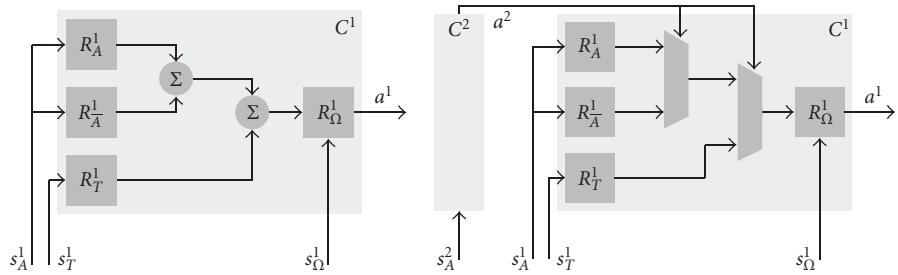


Figure 10: Two action selection schemes for multi-agent behavior; R_A and $R_{\overline{A}}$ are taxis and anti-taxis controllers, respectively, where the sensory feedback comes from other agents, R_T is a controller for tracking the common target, T, and R_Ω is a system that attenuates translational motion towards obstacles.

course, not without its cost: the inability to guarantee an upper bound on search time.

4.2. Multi-Agent Systems. We present some preliminary action selection schemes that result in useful emergent group behavior.

4.2.1. Superposition. Figure 10(a) illustrates a scheme where the outputs of the target and agent tracking controllers are superposed; this is akin to the approach taken in [28]. This scheme works well for unconstrained flocking, as illustrated in Figure 11. As can be seen, the six agents form two separate flocks as they navigate to the target; once at the target, they organize about the target. However, we note that when constraints, such as obstacles, are introduced, undesirable equilibrium points arise and agents are prone to getting "locked" at various points far from the target.

4.2.2. Multiplexing. The scheme in Figure 10(b) addresses the problem of undesirable equilibria by using a multiplexing scheme. Controller C^2 uses a combination of leaky integrators and hysteresis functions to realize an action selector that selects the action whose stimulating input is the most persistent over time. Figure 12 illustrates eight agents operating under this scheme. Whereas with a superposition scheme, some agents would have gotten stuck near the two obstacles, under this scheme spurious equilibria cannot emerge and all agents end up at the target. The mess of trajectories arises because the action selector is never at rest and so agents meander about the target.

5. Conclusions and Future Work

This work was concerned with the synthesis of efficient signal processing schemes for robotic behavior generation using analog-amenable computational machinery. We demonstrated the synthesis of several behaviors for taxis using a novel visual tool, vector field design. To demonstrate the operation of a control architecture based on these behaviors, we proposed two action selection mechanisms that realized the extreme cases of behavior superposition and behavior multiplexing.

Since this work is targeted to lightweight field robotics, we have taken an agent-centric approach; however, the field of multiagent systems design includes more global, optimal frameworks. Of particular interest is work in the computer graphics community on achieving realistic real-time simulations of multiagent phenomena. In [21], Treuille et al., propose a nonagent based scheme, which considers more global knowledge, and utilizes optimal path planning of aggregate quantities (and not per-agent dynamics); this approach enables real-time simulation of realistic multiagent behavior at the crowd level. An interesting item of future work would be to integrate the lightweight agent models of our work within the framework of [21], which might result

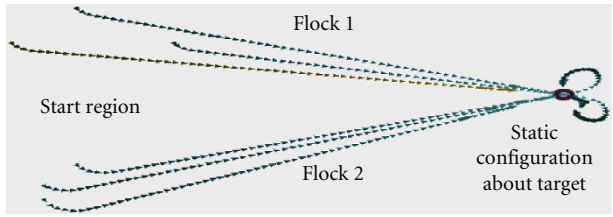


Figure 11: Six agents flocking to the target using the superposition action selection scheme.

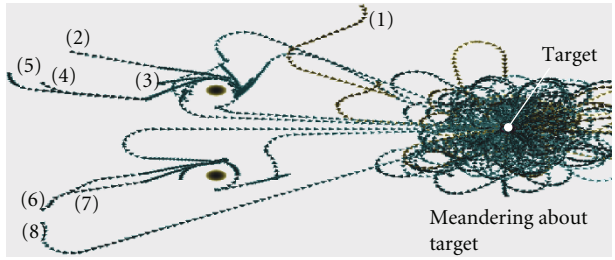


Figure 12: Eight agents flocking to the target using the multiplexed action selection scheme.

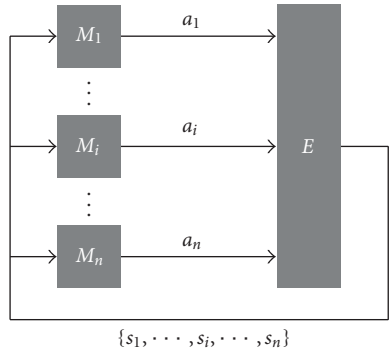


Figure 13: Overview of the multiagent simulation environment. The agents, M_i, generate actuation functions, $\mathbf{a}_i$, as static functions of their sensory perception, $\mathbf{s}_i$, of the environment, E.

in realistic behavior across scales, from the level of the group to that of the individual agents.

A further extension of our work would consider "second order" schemes where vector fields are produced at run time, as functions of the agent's sensory input. In such a scheme, as the agent operates in the field, it would compute a vector field, dynamically, as a function of sensor data. A control law would then be compiled, at run time, from this generated vector field and used to direct the agent. Although this would incur the higher computational cost of computing vector fields on-line, it might also impart more spatial awareness to the agent, reducing the need for searching behavior.

Appendix

MATLAB was used to simulate the agent interacting in an environment; the Runge-Kutta numerical integration scheme (MATLAB's ode4 solver) with fixed step size was used for all simulations. A custom OpenGL application was developed to animate the simulation results enabling us to verify behavioral characteristics in real-time. Beyond the simulation results presented here, we refer the reader to the accompanying supplementary video which better illustrates system behavior. Using a 1.33 GHz PowerBook G4, the most complex (eight agent) simulation presented in this work took less than ten minutes to simulate.

Figure 13 illustrates the scheme used for the simulations of this paper; the figure is also instructive from the perspective of understanding what computation is done in the agent, versus the environmental effects that are exploited by the agent. The environment model was used to track the evolution of each agent's orientation and position in the environment (based on the agent's velocity actuation commands), and generate sensory feedback (i.e., target, agent and obstacle sensor data). Let a global frame of reference be imposed on the environment, and with respect to this frame of reference let:

(i) $\mathbf{g}^i(t) = \begin{bmatrix} g_1^i \\ g_2^i \end{bmatrix}$ denote the position of agent M_i in the environment,

(ii) $\psi^i(t)$ denote the orientation of agent M_i in the environment.

Then the state of the environment (with initial conditions, $\mathbf{g}^i(0)$ and $\psi^i(0)$) evolves according to:

$$\begin{aligned} \dot{g}_1^i &= a_v^i(t)\cos\left(\psi^i(t)\right), \\ \dot{g}_2^i &= a_v^i(t)\sin\left(\psi^i(t)\right), \\ \dot{\psi}^i &= a_\omega^i(t), \end{aligned} \tag{A.1}$$

where a_v^i and a_ω^i are the commanded translational and rotational speeds, respectively, of agent M_i. Based on the

absolute positions of each agent and the targets of interest, E computes $\mathbf{s}_i$ which models the type of *relative*, local sensory feedback signals an agent receives from practical sensors. We note that in this scheme, the computational burden on the agent is limited merely to computing $\mathbf{a}_i$ as a static function of $\mathbf{s}_i$.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments. N. J. Mathai acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) PGS D Scholarship.

References

[1] B. Webb, "Robots in invertebrate neuroscience," *Nature*, vol. 417, no. 6886, pp. 359–363, 2002.

[2] R. A. Brooks, " A hardware retargetable distributed layered architecture for mobile robot control," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, pp. 106–110, Raleigh, NC, USA, March 1987.

[3] C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, Reading, Mass, USA, 1989.

[4] R. Sarpeshkar, "Analog versus digital: extrapolating from electronics to neurobiology," *Neural Computation*, vol. 10, no. 7, pp. 1601–1638, 1998.

[5] J. Van der Spiegel, P. Mueller, D. Blackman, et al., "An analog neural computer with modular architecture for real-time dynamic computations," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 1, pp. 82–92, 1992.

[6] M. Figueroa, D. Hsu, and C. Diorio, "A mixed-signal approach to high-performance low-power linear filters," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 5, pp. 816–822, 2001.

[7] R. Genov and G. Cauwenberghs, "Kerneltron: support vector "machine" in silicon," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1426–1434, 2003.

[8] R. M. Philipp and R. Etienne-Cummings, "Single-chip stereo imager," *Analog Integrated Circuits and Signal Processing*, vol. 39, no. 3, pp. 237–250, 2004.

[9] R. R. Harrison, "A biologically inspired analog IC for visual collision detection," *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 11, pp. 2308–2318, 2005.

[10] R. D. Beer, H. J. Chiel, and L. S. Sterling, "A biological perspective on autonomous agent design," *Robotics and Autonomous Systems*, vol. 6, no. 1-2, pp. 169–186, 1990.

[11] M. W. Tilden, "Adaptive robotic nervous systems and control circuits therefor," US patent no. 5325031, June 1994.

[12] W. R. Ashby, *Design for a Brain*, Chapman & Hall, London, UK, 2nd edition, 1960.

[13] W. R. Ashby, *An Introduction to Cybernetics*, Chapman & Hall, London, UK, 1968.

[14] R. E. Kalman and J. E. Bertram, "Control system analysis and design via the second method of Lyapunov," *Journal of Basic Engineering*, vol. 82, pp. 371–400, 1960.

[15] H. K. Khalil, *Nonlinear Systems*, Prentice Hall, Englewood Cliffs, NJ, USA, 3rd edition, 2002.

[16] S. J. Rosenschein and L. P. Kaelbling, "The synthesis of digital machines with provable epistemic properties," in *Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge (TARK '86)*, pp. 83–98, Monterey, Calif, USA, March 1986.

[17] N. J. Mathai and T. Zourntos, "Control-theoretic synthesis of hierarchical dynamics for embodied cognition in autonomous robots," in *Proceedings of IEEE Symposium on Artificial Life*, pp. 448–455, Honolulu, Hawaii, USA, April 2007.

[18] T. Zourntos and N. J. Mathai, "A BEAM-inspired lyapunov-based strategy for obstacle avoidance and target-seeking," in *Proceedings of the American Control Conference*, pp. 5302–5309, New York, NY, USA, July 2007.

[19] T. Zourntos, N. J. Mathai, S. Magierowski, and D. Kundur, "A bio-inspired analog scheme for navigational control of lightweight autonomous agents," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '08)*, pp. 1132–1137, Pasadena, Calif, USA, May 2008.

[20] E. Zhang, K. Mischaikow, and G. Turk, "Vector field design on surfaces," *ACM Transactions on Graphics*, vol. 25, no. 4, pp. 1294–1326, 2006.

[21] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 1160–1168, 2006.

[22] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[23] R. A. Brooks, "Elephants don't play chess," *Robotics and Autonomous Systems*, vol. 6, no. 1-2, pp. 3–15, 1990.

[24] L. Steels, "Intelligence with representation," *Philosophical Transactions of the Royal Society A*, vol. 361, no. 1811, pp. 2381–2395, 2003.

[25] P. Maes, "Situated agents can have goals," *Robotics and Autonomous Systems*, vol. 6, no. 1-2, pp. 49–70, 1990.

[26] S. M. LaValle, *Motion Planning*, Cambridge University Press, Cambridge, UK, 2006.

[27] N. J. Mathai, T. Zourntos, and D. Kundur, "Vector field design of screw-type chaos," to appear in *Fluctuation and Noise Letters*.

[28] M. J. Matarić, "Designing and understanding adaptive group behavior," *Adaptive Behavior*, vol. 4, no. 1, pp. 50–81, 1995.