

**CYBERNETIC AUTOMATA: AN APPROACH FOR THE  
REALIZATION OF ECONOMICAL COGNITION FOR  
MULTI-ROBOT SYSTEMS**

A Dissertation

by

NEBU JOHN MATHAI

Submitted to the Office of Graduate Studies of  
Texas A&M University  
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2008

Major Subject: Electrical Engineering

**CYBERNETIC AUTOMATA: AN APPROACH FOR THE  
REALIZATION OF ECONOMICAL COGNITION FOR  
MULTI-ROBOT SYSTEMS**

A Dissertation

by

NEBU JOHN MATHAI

Submitted to the Office of Graduate Studies of  
Texas A&M University  
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Co-Chairs of Committee,	Deepa Kundur Takis Zourntos
Committee Members,	Nancy M. Amato Karen L. Butler-Purpy Scott L. Miller
Head of Department,	Costas N. Georghiades

May 2008

Major Subject: Electrical Engineering

**ABSTRACT**

Cybernetic Automata: An Approach for the Realization of Economical  
Cognition for Multi-Robot Systems. (May 2008)

Nebu John Mathai, B.A.Sc., University of Toronto;  
M.Eng., University of Toronto

Co-Chairs of Advisory Committee: Dr. Deepa Kundur  
Dr. Takis Zourntos

The multi-agent robotics paradigm has attracted much attention due to the variety of pertinent applications that are well-served by the use of a multiplicity of agents (including space robotics, search and rescue, and mobile sensor networks). The use of this paradigm for most applications, however, demands economical, lightweight agent designs for reasons of longer operational life, lower economic cost, faster and easily-verified designs, etc.

An important contributing factor to an agent's cost is its control architecture. Due to the emergence of novel implementation technologies carrying the promise of economical implementation, we consider the development of a technology-independent specification for computational machinery. To that end, the use of cybernetics toolsets (control and dynamical systems theory) is appropriate, enabling a principled specification of robotic control architectures in mathematical terms that could be mapped directly to diverse implementation substrates.

This dissertation, hence, addresses the problem of developing a technology-independent specification for lightweight control architectures to enable robotic agents to serve in a multi-agent scheme. We present the principled design of static and dynamical regulators that elicit useful behaviors, and integrate these within an overall architecture for both single and multi-agent control. Since the use of control theory can be limited in unstructured environments, a major focus of the work is on the

engineering of emergent behavior.

The proposed scheme is highly decentralized, requiring only local sensing and no inter-agent communication. Beyond several simulation-based studies, we provide experimental results for a two-agent system, based on a custom implementation employing field-programmable gate arrays.

To my family and my teachers

## ACKNOWLEDGMENTS

I would like to thank my advisors, Dr. Takis Zourntos and Dr. Deepa Kundur, whose instruction, mentorship, and cultivation of creativity have been essential to my development as a scholar and a teacher. I could not have asked for better mentors: their friendship and genuine concern for their students make me very fortunate to have been under their guidance.

I would also like to thank the members of my committee, Dr. Nancy Amato, Dr. Karen Butler-Purry, and Dr. Scott Miller, for the care with which they went over this work, providing invaluable feedback.

I thank Dr. N. Sivakumar who served to increase my level of mathematical maturity and literacy.

During my studies at Texas A&M University, I had the privilege of serving as a lecturer; this experience helped to refine my pedagogical approach and discover the joy of teaching. In addition to my advisors who have supported these endeavors, I would like to thank Dr. Prasad Enjeti and Dr. Costas Georghiades for providing me with several invaluable teaching opportunities. I am grateful for Dr. Enjeti's mentorship, advice, and concern for my development as an academic. I'd also like to acknowledge Dr. Robert Nevels, Dr. Narasimha Reddy, and Dr. Chanan Singh for their roles in obtaining my first two positions.

The friends I made at Texas A&M certainly made for interesting times. In particular, Johnny Lee was always quick to lend a hand, whether in helping me sort out pressing affairs in College Station while I was in Canada, or by serving as co-pilot for an impromptu New Mexico expedition.

I thank the Natural Sciences and Engineering Research Council of Canada for its generous support via the NSERC PGS-D Scholarship.

My family has always been there to support me in my various endeavors. Their presence has made these endeavors meaningful—I shall be ever-thankful for this. My father, Kalladal John Mathai, inspired me to embark on this journey. I am an engineer and a good teacher by his example. My mother, Elizabeth John Mathai, cultivated a love for education in our house. My brother, Binu Samuel John Mathai, has always served as a most trusted—and capable—*consigliere*. I was very fortunate to have my family expanded during the course of my doctoral studies to include M. Joppan, Gracykutty Joppan, and Josen Joppan. I would like to thank them for entrusting me with Preetha, my love (who provided a vital spark that inspired vector field design). And finally—but certainly not least (except in size at this point!)—to my dearest Maria: when you can read this, you should know that your smiles carried me through the final leg of this journey.

## TABLE OF CONTENTS

CHAPTER		Page
I	INTRODUCTION . . . . .	1
	A. Multi-agent robotic systems . . . . .	1
	B. Technology-independent computation . . . . .	5
	C. Cybernetics . . . . .	7
	D. Contributions of this dissertation . . . . .	10
II	PRELIMINARIES . . . . .	12
	A. Background . . . . .	12
	1. Agents . . . . .	12
	2. Dynamical systems and control theory . . . . .	14
	a. Static versus dynamical control schemes . . . . .	15
	3. Embodiment and situatedness . . . . .	16
	B. Previous work on robot control architectures . . . . .	17
	C. Previous work on multi-agent systems . . . . .	24
	1. Explicit coordination . . . . .	24
	2. Implicit coordination . . . . .	25
	D. Perspective on our work . . . . .	26
III	PROBLEM FORMULATION AND TOOLSETS . . . . .	28
	A. Introduction . . . . .	28
	B. Preliminaries . . . . .	29
	1. The world and objects therein . . . . .	29
	2. Agent frame of reference . . . . .	31
	C. Formulation . . . . .	31
	1. Autonomous navigation . . . . .	32
	a. Exploration with obstacle avoidance . . . . .	32
	b. Target tracking with obstacle avoidance . . . . .	33
	2. Organization with respect to other agents . . . . .	33
	a. Flocking . . . . .	34
	b. Static organization . . . . .	35
	3. Remarks . . . . .	36
	D. Proposed toolset . . . . .	37
	1. Desired characteristics . . . . .	37
	2. Automata theory . . . . .	38



CHAPTER	Page
3. Computational intelligence . . . . .	40
4. Cybernetics tools . . . . .	41
IV     STATIC SCHEMES FOR AGENT BEHAVIOR . . . . .	42
A. Introduction . . . . .	42
B. Describing perception . . . . .	44
1. Sensors . . . . .	44
2. Actuators . . . . .	45
3. Plant model for perception . . . . .	46
C. Regulating perception . . . . .	49
1. An analytic approach . . . . .	50
2. Vector field design: a graphical approach . . . . .	51
a. An unsatisfactory solution . . . . .	51
b. A better solution . . . . .	53
D. Synthesis of taxis behaviors . . . . .	55
1. Unconstrained, biased taxis . . . . .	58
a. Qualitative structure of $\hat{\mathbf{p}}$ . . . . .	58
b. Construction of an analytic form for $\hat{\mathbf{p}}$ . . . . .	60
c. Derivation of the actuation command . . . . .	63
2. Taxis with constrained translational motion . . . . .	64
a. Qualitative structure of $\hat{\mathbf{p}}$ . . . . .	64
b. Construction of an analytic form for $\hat{\mathbf{p}}$ . . . . .	65
c. Derivation of the actuation command . . . . .	67
d. Taxis by reverse-only motion . . . . .	68
3. Taxis with constrained rotational motion . . . . .	68
E. Synthesis of non-taxis behaviors . . . . .	70
1. Anti-taxis . . . . .	71
2. Parking . . . . .	73
F. Summary of vector fields . . . . .	74
G. Simulation results . . . . .	74
H. Discussion . . . . .	78
V     DYNAMIC CONTROL SCHEMES FOR NAVIGATION . . . . .	87
A. Introduction . . . . .	87
B. Preliminaries . . . . .	87
1. Loosened coupling between sensing and action . . . . .	87
2. Unifying taxis and obstacle avoidance . . . . .	89
a. Obstacle sensors . . . . .	89
b. Disturbances . . . . .	91

CHAPTER	Page
C. Controller synthesis . . . . .	93
1. Derivation of virtual control, $\mathbf{s}^*$ . . . . .	93
2. Backstepping setup . . . . .	95
3. Stabilization . . . . .	96
D. Properties . . . . .	97
1. Structure . . . . .	97
a. Neural structure . . . . .	98
2. Scalability . . . . .	99
3. Singularity . . . . .	101
4. Relaxation of stability . . . . .	102
E. Simulation results . . . . .	102
F. Weak emergence of satisficing intelligence . . . . .	108
 VI	
INTEGRATED CONTROL ARCHITECTURES FOR SINGLE-AGENT SYSTEMS . . . . .	111
A. Introduction . . . . .	111
1. Integration of behaviors . . . . .	112
B. Machine organization . . . . .	113
1. Hierarchy . . . . .	113
a. The proposed architecture . . . . .	115
2. Layering of behaviors . . . . .	117
C. Static schemes for single agent systems . . . . .	119
1. Obstacle avoidance . . . . .	119
2. Searching . . . . .	124
a. Design of a reference oscillator for searching . . . . .	125
3. Integration . . . . .	128
 VII	
INTEGRATED CONTROL ARCHITECTURES FOR MULTI-AGENT SYSTEMS . . . . .	132
A. Introduction . . . . .	132
1. The “nearest neighbor” agent sensor . . . . .	132
2. Outline . . . . .	133
B. Regulating inter-agent boundaries . . . . .	134
C. Flocking . . . . .	135
1. Action superposition . . . . .	138
2. Action multiplexing . . . . .	140
D. Self-organization: passive coordinated deployment . . . . .	140
E. An integrated architecture . . . . .	145
F. Discussion . . . . .	152

CHAPTER	Page
1. Implementation of behaviors . . . . .	153
2. Flocking . . . . .	155
3. Self-organization . . . . .	157
VIII COMPARISONS WITH OTHER SCHEMES . . . . .	159
A. Introduction . . . . .	159
B. Taxis in an obstacle-free environment . . . . .	160
1. Constrained target sensing . . . . .	161
2. Algorithms compared . . . . .	161
a. Braitenberg vehicle 3a (Bra3a) . . . . .	163
b. Brooks-Matarić “homing” behavior (BroMat) . . . . .	164
c. Borenstein-Koren virtual force field (BorKor) . . . . .	165
d. Unconstrained taxis (uc) . . . . .	167
3. Methodology and performance metrics . . . . .	168
4. Results . . . . .	169
C. Navigation: taxis with obstacle avoidance . . . . .	170
1. Algorithms compared . . . . .	170
a. Braitenberg vehicle 3c (Bra3c) . . . . .	176
b. Matarić’s “avoid-everything-else” behavior (Mat) . . . . .	177
c. A dithered virtual force field method (BorKor-d) . . . . .	177
2. Methodology . . . . .	178
3. Results . . . . .	179
D. Discussion . . . . .	180
IX EXPERIMENTAL VALIDATION . . . . .	189
A. Introduction . . . . .	189
1. Methodology . . . . .	189
B. Testbed setup . . . . .	191
1. Chassis and electromechanical subsystems . . . . .	191
2. Sensors . . . . .	193
a. Obstacle sensing . . . . .	193
b. Target and agent sensing . . . . .	193
3. Actuators . . . . .	194
4. Computational substrate . . . . .	196
a. Control hardware . . . . .	196
5. Integrated vehicle . . . . .	197
C. Experiments . . . . .	197

CHAPTER	Page
X	CONCLUSION . . . . . 200
	A. Summary of the work . . . . . 200
	B. Future work . . . . . 201
	1. Extending vector field design . . . . . 201
	2. Improved perception . . . . . 202
	3. Machine “introspection” . . . . . 203
	4. Inter-agent communication . . . . . 203
	REFERENCES . . . . . 205
	APPENDIX A . . . . . 219
	APPENDIX B . . . . . 220
	VITA . . . . . 223

## LIST OF TABLES

TABLE		Page
I	Summary of toolset considerations. . . . .	41
II	Summary of reference vector fields for taxis. . . . .	75
III	Summary of reference vector fields for non-taxis behaviors. . . . .	76
IV	Quantities used in the dynamical control scheme. . . . .	100
V	Computations performed by the dynamical control scheme. . . . .	100
VI	A hierarchy of behaviors for navigation. . . . .	114
VII	Correspondence between the behaviors of a software-based scheme and our scheme. . . . .	154
VIII	Five different initial conditions for the agent were used in the comparison of taxis algorithms. . . . .	168
IX	Summary of algorithm performance characteristics for obstacle- free taxis: $d = 4, \psi = 0$ . . . . .	171
X	Summary of algorithm performance characteristics for obstacle- free taxis: $d = 4, \psi = \frac{3}{4}\pi$ . . . . .	173
XI	Summary of algorithm performance characteristics for obstacle- free taxis: $d = 4, \psi = \frac{4}{3}\pi$ . . . . .	174
XII	Summary of algorithm performance characteristics for obstacle- free taxis: $d = 8, \psi = \frac{4}{5}\pi$ . . . . .	175
XIII	Summary of algorithm performance characteristics for obstacle- free taxis: $d = 8, \psi = \frac{3}{2}\pi$ . . . . .	176
XIV	Summary of algorithm performance characteristics: taxis with obstacle avoidance. The dash (–) indicates that the algorithm did not converge, that is, the agent was unable to circumnavigate the obstacle. . . . .	184

TABLE	Page
XV Summary of accompanying video files. . . . .	219

## LIST OF FIGURES

FIGURE	Page
1	An agent (composed of sensors, $S$ , actuators, $A$ , and computational hardware, $C$ ) coupled to an environment, $E$ . . . . . 2
2	Overview of a general multi-agent robotics application. . . . . 4
3	Hardware topology of a dynamical system; $D$ represents a memory block. 6
4	An agent, $M$ , coupled to the environment, $E$ . . . . . 17
5	Classification of embodied control schemes. . . . . 19
6	A control scheme that uses $n$ processing steps to process sensor data and generate an action. Each step requires the others and can not, in isolation, control the agent. . . . . 21
7	A subsumption architecture with $n$ primitive controllers. Each controller has access to the sensors and actuators and hence can, in isolation, control the agent. Higher-level controllers influence the operation of lower ones via tuning channels (shown by diagonal arrows). The action selection subsystem enforces the scheme's policy on how the individual controller outputs determine the overall action of the agent. . . . . 22
8	The world and objects therein: $M_i$ and $M_j$ are agents, trying to get to a common target of interest, $T$ , in an environment with obstacles, $\Omega_1$ , $\Omega_2$ , and $\Omega_3$ . Agent $M_i$ senses the displacement to agent $M_j$ as $\mathbf{l}_{M_j}^i$ , and the displacement to $T$ as $\mathbf{l}_T^i$ . . . . . 30
9	The autonomous navigation problem. (1) An agent (in red) located in a region where it can not perceive the target, executes an approximation of a space-filling curve which eventually enters a region within sensing range of $T$ . (2) The agent navigates to $T$ , executing a collision-free path about the obstacles, $\Omega_1, \dots, \Omega_7$ , and (3) eventually reaches $T$ . . . . . 32

FIGURE	Page
10	An agent, $M$ , to which a local coordinate system has been attached. The agent is sensing a target of interest, $T$ , whose instantaneous position with respect to $M$ 's local frame of reference is $\boldsymbol{\eta}(t)$ . . . . . 44
11	Top-view of a simple unicycle (in grey) to which a local $l_1 - l_2$ coordinate system has been attached. The directions of positive $v$ and $\omega$ are indicated by red arrows. The unicycle's position and orientation with respect to a global $x_1 - x_2$ coordinate system (unattached to the unicycle) are denoted by $\boldsymbol{x}$ and $\psi$ . . . . . 46
12	The agent at time $t$ (shown in red with its local coordinate system), and at time $t + h$ (in blue). Also illustrated are the corresponding displacements to the target $T$ ( $\boldsymbol{\eta}(t)$ and $\boldsymbol{\eta}(t + h)$ , respectively), and the displacement, $\boldsymbol{b}$ , and angle, $\delta_\psi$ , between the two frames of reference. . . . . 48
13	A candidate vector field that globally asymptotically stabilizes $\boldsymbol{\eta} = \mathbf{0}$ . . . . . 52
14	Behaviors specified by the reference vector field of Figure 13. . . . . 53
15	Vector field for unconstrained taxis. . . . . 54
16	Behavior specified by the reference vector field of Figure 15. . . . . 54
17	Key manifolds and their behavioral implications. . . . . 56
18	The influence of flow curvature on behavior. Flow $c$ is circular, while the flows denoted by the subscripts $+$ and $-$ denote flows whose radii are increasing or decreasing, respectively. . . . . 57
19	Vector field for unconstrained taxis with forward bias. . . . . 59
20	Behavior 'c' specified by the reference vector field of Figure 19 that biases forward motion. . . . . 59
21	Candidates for $f(\cdot)$ in (4.32). . . . . 61
22	Behavior of an agent engaging in taxis by forward-only motion. . . . . 64
23	Vector field for taxis by forward motion. . . . . 65



FIGURE	Page
24	Candidates for $f(\cdot)$ in (4.43). . . . . 67
25	Behavior of an agent engaging in taxis by reverse-only motion. . . . . 68
26	Vector field for taxis by reverse motion. . . . . 69
27	Vector field for taxis with a rotational bias. . . . . 70
28	Vector field for asymptotic anti-taxis. . . . . 72
29	Vector field for rotational anti-taxis. . . . . 72
30	Vector field for parking. . . . . 73
31	Setup for simulations. . . . . 77
32	Simulation results for the unconstrained taxis behavior showing the agent's trajectory for four cases. The reference vector field for unconstrained taxis is shown in the center with annotations that correspond to the simulated trajectories. . . . . 79
33	Simulation results for the unconstrained taxis with forward bias behavior showing the agent's trajectory for two representative cases, and the associated reference vector field. . . . . 80
34	Simulation results for the constrained taxis by forward motion behavior showing the agent's trajectory for two representative cases, and the associated reference vector field. . . . . 81
35	Simulation results for the constrained taxis by reverse motion behavior showing the agent's trajectory for two representative cases, and the associated reference vector field. . . . . 82
36	Simulation results for the asymptotic anti-taxis behavior showing the agent's trajectory for two representative cases, and the associated reference vector field. . . . . 83
37	Simulation results for the rotational anti-taxis behavior showing the agent's trajectory for two representative cases, and the associated reference vector field. . . . . 84

FIGURE	Page
38	Simulation results for the parking behavior showing the agent's trajectory for two representative cases, and the associated reference vector field. . . . . 85
39	The structure of a dynamical controller. . . . . 88
40	Specification of an obstacle sensor. . . . . 90
41	The function $\mathbf{s}_\Omega : r_\Omega \mapsto [0, 1]$ is: (a) 0 for $r_\Omega \leq r_\Omega^{\min}$ , (b) monotonically increasing for $r_\Omega^{\min} \leq r_\Omega \leq r_\Omega^{\max}$ , and (c) 1 for $r_\Omega \geq r_\Omega^{\max}$ . . . 91
42	Revised controller motif with track and suppression channels. The designer must design $C$ so that the tracking objective (i.e., imposing desirable characteristics on the agent's perception of the world, $\mathbf{s}$ ) can be accomplished in the face of disturbances, $\mathbf{d}$ , perturbing $C$ 's action, $\mathbf{a}$ , upon $P$ . . . . . 93
43	Definitions of the pul : $\mathcal{R} \mapsto \mathcal{R}$ and sat : $\mathcal{R} \mapsto \mathcal{R}$ functions, parameterized by the constant $l_1 > 0$ . . . . . 95
44	Structure of the dynamical controller. . . . . 97
45	Neural aspects of the controller's structure. . . . . 99
46	Patched controller; $C$ is the dynamical controller of Figure 44 and $C^*$ implements (5.27). . . . . 101
47	Simulation results for the dynamical controller with no relaxation of stability ( $\kappa_1 = 0$ ); no obstacles are present. . . . . 103
48	Simulation of the dynamical controller with $\kappa_1 = 0$ ; the agent is impeded by a small ball. . . . . 103
49	Simulation of the dynamical controller with relaxed stability ( $\kappa_1 < 0$ ) for the obstacle-free case. . . . . 104
50	Simulation of the dynamical controller with $\kappa_1 < 0$ ; the agent is able to avoid a small ball placed in its path. . . . . 105
51	Simulation of the dynamical controller with $\kappa_1 < 0$ ; the agent is able to avoid a bigger ball placed in its path. . . . . 105

FIGURE	Page
52	Simulation of the dynamical controller with $\kappa_1 < 0$ ; the agent is able to circumnavigate a small wall in its path. . . . . 106
53	Simulation of the dynamical controller with $\kappa_1 < 0$ ; the agent is able to circumnavigate a big wall in its path. . . . . 106
54	Simulation of the dynamical controller with $\kappa_1 < 0$ ; the agent is able to circumnavigate a bigger wall in its path. . . . . 107
55	Simulation of the dynamical controller with $\kappa_1 < 0$ ; the wall is too big for the agent to circumnavigate. . . . . 107
56	Recursive development of our hierarchical control architecture. . . . . 116
57	The architecture of a general layered controller at the $i$ -th level of hierarchy, $C_i$ . Elementary controllers, $R_a^b$ , that realize various basis behaviors are grouped according to whether they address concurrent goals (in which case they have different superscripts) or exclusive goals (in which case they have different superscripts). . . 118
58	Agent setup with front and rear obstacle sensors. . . . . 119
59	Development of a reactive control architecture for single agent systems. . . . . 121
60	An Archimedean spiral (also known as an arithmetic spiral). Successive crossings of this curve across the $g_1$ axis (and, more generally, across any ray emanating from the spiral's origin) are separated by $\rho$ . . . . . 125
61	The trajectory of a vehicle starting at the origin, under (6.12) and $v(t) \equiv 0.1$ . . . . . 127
62	The vector field structure of oscillator (6.13). . . . . 129
63	A controller integrating taxis with an open-loop search behavior. . . 130
64	An example of basic obstacle avoidance. . . . . 130
65	An example of searching and taxis, both with obstacle avoidance. . . 131

FIGURE	Page
66	The agent sensor of $M$ returns a measurement of the displacement, $\mathbf{l}_j$ , to the closest agent, $M_j$ . . . . . 133
67	The “social boundaries” of agent $M_i$ : the inclusion zone, $Z_i$ is shown in blue, while the exclusion zone, $Z_e$ , is shown in yellow. Agent $M_i$ senses $M_j$ with respect to $M_i$ ’s local $l_1^i - l_2^i$ coordinate system, and strives to maintain $M_j$ within the inclusion zone (i.e., at a distance, $r_A$ , where $r_{A,\min} < r_A < r_{A,\max}$ ). . . . . 135
68	A regulation scheme to maintain social boundaries. . . . . 136
69	A regulation scheme for flocking using action superposition. The yellow-highlighted $R_T$ and $R_A$ blocks realize target tracking and agent tracking, respectively. The social boundary regulator is shown highlighted in dark blue; the summation is done at the output of this regulator. . . . . 137
70	The result of simulating six agents under the scheme of Figure 69. . . 138
71	A regulation scheme for flocking using action superposition. The social boundary regulator has been split into two (highlighted by the two dark blue boxes), and the superposition with target tracking is done in between the two halves. . . . . 139
72	The result of simulating six agents under the scheme of Figure 71. . . 140
73	A regulator for flocking; action multiplexing. . . . . 141
74	The result of simulating six agents under the scheme of Figure 73. . . 141
75	The result of simulating six agents under the scheme of Figure 73 with obstacles. . . . . 142
76	The case of a hexagonal formation of agents about a target. . . . . 143
77	Under the coordinated deployment behavior, the agent (shown as a red triangle at the origin of the $l_1 - l_2$ coordinate system) attempts to regulate $\mathbf{s}_A$ to within the grey regions (which are sectors of width $2\delta_C > 0$ offset by $\theta_C > 0$ with respect to $\mathbf{s}_T$ ). The blue shaded region indicates the range of the agent sensor. . . . 145

FIGURE	Page
78	A regulator for self-organization (highlighted in dark blue). A regulator for taxis is also instantiated to bring the agent to the target; once there, taxis will disengage, and self-organization will take over. . . . . 146
79	Self-organization of a single group of twelve agents. . . . . 146
80	Self-organization of twelve agents divided into two groups. . . . . 147
81	Self-organization of one group of thirteen agents. . . . . 147
82	Self-organization of twenty six agents, divided into two groups. Note the emergence of a symmetric final formation. In the final configuration, all agents except for four are stable in a static formation; the four that are not stationary, oscillate about stationary positions. . . . . 148
83	An integrated controller for a multi-agent scheme that includes flocking, self-organization about a target waypoint, and obstacle avoidance. . . . . 149
84	Two groups of agents (twenty-six in total) flock to the target and self-organize about it. The final configuration is mostly stationary (i.e., most of the agents are at rest), with isolated subgroups of agents occasionally moving about to re-organize. . . . . 150
85	A groups of twenty-six agents flock to the target and self-organize about it, navigating past two obstacles. . . . . 151
86	This simulation uses the same initial group configuration as Figure 85, but with the obstacles repositioned near the target to constrain that area. The collective self-organizes into a configuration that is distorted by the obstacles. . . . . 152
87	Constrained target sensing. The target sensor takes the actual displacement to the target, $\boldsymbol{\eta}$ , and returns a vector, $\boldsymbol{s}$ , constrained to one of $n$ directions (shown in red). . . . . 162

FIGURE	Page
88	The target sensor model used for the simulations of all algorithms in this section. The target sensor is a memoryless system that measures the displacement from the agent to the target, $\eta$ , constrains the direction (from $\theta$ to $\theta'$ through $Q$ ), and returns a vector to the target that is either in polar form, $(\rho, \theta')$ , or rectangular form, $\mathbf{s}$ . The blocks “rect2pol” and “pol2rect” effect the conversions between rectangular and polar coordinates. . . . . 162
89	Braitenberg vehicle 3a. Information from the target sensors (in red) are mapped directly to the motor actuators (in blue). The mapping is “inhibitory” so that when the target is near the sensor, the motors are actuated to a <i>lesser</i> degree than when the target is far from the sensor. The effect of this mapping is that the agent engages in taxis, coming to rest at the target. . . . . 163
90	The underlying “rotate-and-go” motion controller used in Brooks’ subsumption scheme. The “turn” module is first engaged to cause the agent to align with a desired heading; while rotating, the “forward” motion block is inhibited. Upon aligning with the commanded heading, the forward block is engaged, and the agent moves forwards. . . . . 165
91	Our implementation of Brooks-Matarić homing behavior. The red arrows indicate transitions caused by arrival at the target. . . . . 165
92	The underlying motion controller used in the scheme of Borenstein and Koren. . . . . 167
93	The agent was placed $d$ units away from the target, with an orientation of $\psi$ radians with respect to the target. . . . . 169
94	Path length versus the number of distinct target sensor directions ( $n$ ) for the Braitenberg vehicle 3a (Bra3a), Brooks-Matarić homing (BroMat), Borenstein-Koren virtual force field (BorKor), and unconstrained taxis (uc) algorithms; $d = 4$ and $\psi = 0$ . . . . . 171

FIGURE	Page
95	Path length versus the number of distinct target sensor directions ( $n$ ) for the Braitenberg vehicle 3a (Bra3a), Brooks-Matarić homing (BroMat), Borenstein-Koren virtual force field (BorKor), and unconstrained taxis (uc) algorithms; $d = 4$ and $\psi = \frac{3}{4}\pi$ . Note the high degree of insensitivity to $n$ of unconstrained taxis. . . . . 172
96	Path length versus the number of distinct target sensor directions ( $n$ ) for the Braitenberg vehicle 3a (Bra3a), Brooks-Matarić homing (BroMat), Borenstein-Koren virtual force field (BorKor), and unconstrained taxis (uc) algorithms; $d = 4$ and $\psi = \frac{4}{3}\pi$ . . . . . 173
97	Path length versus the number of distinct target sensor directions ( $n$ ) for the Braitenberg vehicle 3a (Bra3a), Brooks-Matarić homing (BroMat), Borenstein-Koren virtual force field (BorKor), and unconstrained taxis (uc) algorithms; $d = 8$ and $\psi = \frac{4}{5}\pi$ . . . . . 174
98	Path length versus the number of distinct target sensor directions ( $n$ ) for the Braitenberg vehicle 3a (Bra3a), Brooks-Matarić homing (BroMat), Borenstein-Koren virtual force field (BorKor), and unconstrained taxis (uc) algorithms; $d = 8$ and $\psi = \frac{3}{2}\pi$ . . . . . 175
99	A reactive scheme for obstacle avoidance based on Braitenberg vehicle “3c.” The vehicle engages in taxis when the obstacle sensor (in orange) is not stimulated; when stimulated by an obstacle, the vehicle stops and rotates until the stimulation ceases. . . . . 177
100	Our implementation Matarić’s avoid-everything-else behavior. The red arrow indicates a transition caused by arrival at the target. . . . 178
101	Paths executed by the agent around obstacle formation ‘A.’ . . . . . 180
102	Paths executed by the agent around obstacle formation ‘B.’ . . . . . 180
103	Paths executed by the agent around obstacle formation ‘C.’ . . . . . 181
104	Paths executed by the agent around obstacle formation ‘D.’ The BorKor-d method failed to circumnavigate the obstacle. . . . . 181
105	Paths executed by the agent around obstacle formation ‘E.’ Only cr and Mat enabled the agent to avoid the obstacle. . . . . 182

FIGURE	Page
106	Paths executed by the agent around obstacle formation ‘F.’ Only cr and Mat enabled the agent to avoid the obstacle. . . . . 182
107	Paths executed by the agent around obstacle formation ‘G.’ Only cr and Mat enabled the agent to avoid the obstacle. . . . . 183
108	Paths executed by the agent around obstacle formation ‘H’ (note: this concave obstacle was created by adding two horizontal extensions on the top and bottom of formation ‘E’). Only cr enabled the agent to avoid the obstacle. . . . . 183
109	Path executed by the agent around obstacle formation ‘I’ (note: this concave obstacle was created by lengthening the two horizontal extensions of obstacle ‘H’) under control of ‘cr.’ The obstacle is sufficiently concave as to cause ‘cr’ to fail to circumnavigate it. . . 184
110	The omni-directional obstacle sensor measures the displacement, $\mathbf{f}_i$ for $i \in \{1, 2, 3\}$ , from the obstacles (numbered 1, 2 and 3 in the figure) surrounding an agent (in red) to the agent, sums them (while attenuating their magnitudes so that closer obstacles yield longer vectors), and returns the resultant vector, $\mathbf{f}_r$ . . . . . 186
111	A comparison of cr and Mat (both using monocular obstacle sensors), and the virtual force field method using an omni-directional sensor (omni). The path lengths executed by the agent for each of these cases were 21.77, 11.85, and 9.16 respectively. . . . . 187
112	The chassis used for our experimental testbed with an outline of the key electromechanical subsystems. . . . . 192
113	Pulse width modulated scheme for obstacle sensor data read-out. The width of the pulse is directly proportional to the distance to the closest obstacle in front of it. . . . . 193
114	Hardware used to read obstacle sensor data. . . . . 194
115	Hardware for target sensor data conversion. . . . . 195
116	Pulse width modulation scheme for the drive motor power electronics and steering servomotors. . . . . 195



FIGURE	Page
117	Hardware approximation of a hysteresis function. . . . . 196
118	Hardware approximation of a leaky integrator. . . . . 197
119	Top view of agent. . . . . 198
120	Side view of agent. . . . . 199
121	Front view of agent. . . . . 199
122	The robot communication problem from the perspective of agent $M$ . 204
123	Overview of the simulation environment used in this work. The dark grey blocks indicate software that was written as part of the work of this dissertation, while the black boxes indicate third-party software and libraries. . . . . 220
124	Structure of the testbench. . . . . 221

## CHAPTER I

### INTRODUCTION

Significant developments in artificial intelligence . . . must await computers of an entirely different sort, of which the only existing prototype is the little-understood human brain. (Hubert Dreyfus [1])

By analogy with the evolution of natural intelligence, we believe that incrementally solving the control and perception problems of an autonomous mobile mechanism is one of the best ways of arriving at general artificial intelligence. (Hans P. Moravec [2])

In this work, we are concerned with the design of lightweight control architectures to endow robotic agents with various cognitive faculties to operate as part of a multi-agent scheme. To that end, we employ tools from cybernetics to specify technology-independent computational primitives. In this chapter we provide a brief primer on multi-agent robotic systems, the notion of technology-independent computation, and our justification for the use of cybernetics toolsets. We conclude with an overview of the contributions of this dissertation.

#### A. Multi-agent robotic systems

An *agent* is a computational system that is coupled to another system, the *environment*, via sensors (from which information about the environment is determined) and actuators (with which the agent is able to effect change on the environment) [3, 4]. The general structure of an agent is shown in Figure 1. *Robotic* agents are mechatronic

---

The journal model is *IEEE Transactions on Automatic Control*.

systems—artifacts integrating the electronic with the mechanical. In the setup of Figure 1, hardware that performs and measures mechanical work typically occupy the actuators and sensors,<sup>1</sup> while the computation is almost exclusively done with some form of electronic hardware (e.g., general-purpose computers such as microprocessors and microcontrollers, or special-purpose computers realized via field-programmable gate arrays (FPGAs), full-custom digital circuits, or analog circuits) and is interfaced to the sensors and actuators via electronic signaling.

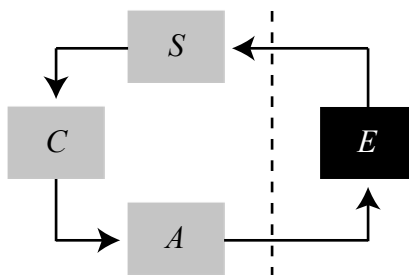


Fig. 1. An agent (composed of sensors,  $S$ , actuators,  $A$ , and computational hardware,  $C$ ) coupled to an environment,  $E$ .

There are a diversity of robotic schemes, which reflects the variety of environments where they have found application. Highly structured environments (e.g., an industrial manufacturing plant) require robotic systems that are capable of faithfully reproducing a *script* of action given to it; the design objectives for such systems do not extend to endowing faculties of autonomy. On the other hand, unstructured environments with very little a priori information (e.g., unknown terrains that need to be traversed and explored) require agents with cognitive faculties that enable them to *satisfice* [5, 6], that is, achieve tolerable (and not necessarily optimal) solutions using the limited resources they have, within an acceptable time-frame (where “acceptable”

---

<sup>1</sup>Examples of actuators include motors, engines and solenoids. Examples of sensors include accelerometers and gyroscopes (often realized via micro-electromechanical systems (MEMS) hardware), in addition to devices that measure physical phenomena such as sound, light, pressure, chemical concentration, etc.

depends on the faculties the agent is endowed with, as well as the application space). The field of autonomous mobile robotics deals with the latter type of environment.

Improving the sensing and computational resources of a robot can compensate for the lack of structure in an environment. For example, giving an autonomous mobile agent:

- sensors with sufficiently long range
- artificial vision
- a global map of the terrain and knowledge of its position with respect to this map (e.g., via a global positioning system)

and a sophisticated computer to fuse the high-quality information provided by these sensory faculties, would enable the agent to, in effect, perceive the underlying structure of a complex environment and plan a goal-directed strategy. However, the use of such highly capable agents is often not practical. Rather, many pertinent application spaces and the paradigm of multi-agent robotic systems require *lightweight* robotic agents—agents that are constrained in terms of economic cost, energy consumption, computational resources, size, mass, and time to act.

Multi-agent robotic systems—using several, possibly lightweight, agents instead of a single highly competent one—enable a diverse array of applications, such as robotic exploration (including space robotics, search and rescue, and other scientific and security-related uses) and mobile sensor networks. These applications are well-served by the benefits of:

- parallelism and distributed computation
- robustness to individual agent failure
- spatially-distributed sensing and actuation

that arise from the use of a multiplicity of agents to attack a problem. A prototypical example of such an application is shown in Figure 2. Aerial vehicles (which may or may not be autonomous) execute a coarse survey of a broad area, “marking” regions of interest (target waypoints) with a beacon (shown in green). Ground-based autonomous robotic agents (shown in red) then navigate, through a potentially obstacle-ridden environment, to the target waypoints where they self-organize into a structure that covers an area about the waypoint to conduct a finer search for phenomena of interest.

The use of agents with simpler, more easily characterized, verifiable and implemented functionality also confers benefits from a design perspective. To minimize the economic cost of a multi-agent scheme, it is important that the complexity of each agent be constrained. Moreover, in mobile sensor network applications (where long operational life is necessary) and robotic exploration problems (where the agent must be able to maneuver effectively through challenging and inaccessible environments), low agent complexity (e.g., in terms of compactness and energy usage) is demanded.

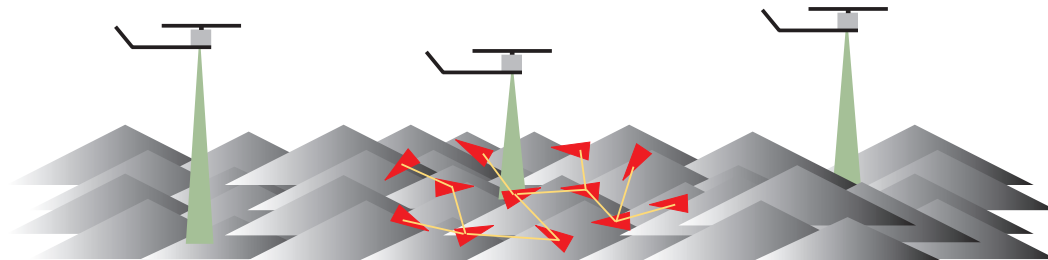


Fig. 2. Overview of a general multi-agent robotics application.

## B. Technology-independent computation

It is often not enough to simply scale down an agent by paring away its sensors and actuators; the design of economical *computation* must also be addressed.

Generally, the need for lightweight computation 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 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.

Practical sequential algorithms (e.g., finite state automata, software algorithms) can be expressed as dynamical systems of the form:

$$\begin{aligned}\mathbf{x}(t+1) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t))\end{aligned}\tag{1.1}$$

where the time variable,  $t$ , belongs to a countable ordered set, and the state,  $\mathbf{x}$ , evolves in a countable state space. Similarly, continuous-time continuous-valued computing systems (i.e., analog computers) can be described by systems of the form:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t))\end{aligned}\tag{1.2}$$

where the time,  $t$ , is a real variable, and the state,  $\mathbf{x}$ , evolves in a real-valued state space (i.e., a subset of  $\mathcal{R}^n$ ). Systems of either type can be directly mapped to hardware of the form shown in Figure 3; for (1.1)  $D$  consists of clocked multi-bit registers and  $\mathbf{f}$  and  $\mathbf{g}$  are Boolean functions, whereas for (1.2)  $D$  consists of  $n$  integrators and  $\mathbf{f}$  and  $\mathbf{g}$  are general functions.

In formulating a control scheme for lightweight robotic agents, we are interested

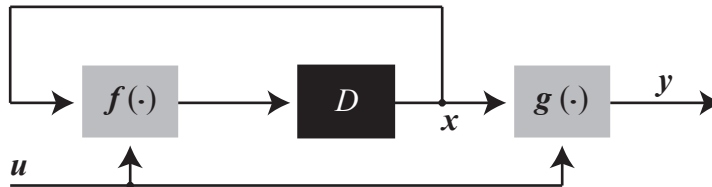


Fig. 3. Hardware topology of a dynamical system;  $D$  represents a memory block.

in a *technology-independent* specification of computational hardware, that is, one that is not tied to a specific hardware realization (e.g., a software algorithm is not a technology-independent scheme since it is predicated on the use of a general-purpose *digital* computer). Owing to the convergence of previously-disparate fields of science and engineering (such as chemistry, biology, electrical and mechanical engineering) as witnessed by the rise of nanotechnology, systems biology, and artificial life, the use of exotic, non-electronic<sup>2</sup> implementation media is foreseeable.

Of particular interest to us (and of more immediate use) is the potential for implementation via custom analog systems, due to:

- the plethora of innate physical characteristics that can be exploited to obtain low-cost computational primitives (e.g., Kirchoff’s current law can be exploited to realize addition “for free”)
- 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)

---

<sup>2</sup>For example, technology based on the use of chemical reaction systems [7, 8, 9, 10, 11]. Attempts to synthesize wet artificial life [12] and the work of systems biologists [13] in uncovering the regulatory circuits accomplished by biochemical reaction systems demonstrate that astonishing algorithmic and regulatory processes can be realized by non-traditional substrates.

- the ability to fine-tune the hardware at a very low level (for VLSI realizations, which are preferable [14])

An excellent overview of the relative merits of analog and digital implementations of computation can be found in [15, 16]; in general, analog systems confer their greatest advantage for processing that requires moderate signal-to-noise ratios—levels relevant to robotic control where noisy, nonlinear sensors practically restrict the fidelity of measurements of environmental data. Recent results from the field of neuromorphic engineering [17, 18, 19, 20, 21] demonstrate the efficacy of analog processing systems from the perspective of functionality and economy of implementation.

Inspired by this, we consider control architectures that are amenable to analog implementation, noting that an analog-amenable specification is simultaneously amenable to custom digital implementation (via discretization or digital redesign methods); however, although a software-based specification can be mapped to a custom digital implementation, it is not necessarily possible to map software to analog (at least, not *directly*<sup>3</sup>). Hence, we need a principled means of synthesizing technology-independent computation. Connectionist [22] and empirical [23] methods of realizing analog computation exist; however, the lack of a rigorous synthesis methodology is a drawback from a conservative engineering point of view.

### C. Cybernetics

The history of artificial automata can be extended back to the fourth century BC, at the very least. In the medieval period we have reports of designs or implementations of programmable automata by al-Jazari (thirteenth century), da Vinci (fifteenth

---

<sup>3</sup>The “directness” of the mapping from system specification to implementation technology is a crucial element of good hardware engineering methodology to ensure that the design is verifiable throughout the design and implementation process.



century) and Vaucanson<sup>4</sup> (eighteenth century). These automata were programmable, being able to execute a *script* of action; however, they were not autonomous.

The early twentieth century saw the advent of cybernetics, a mathematical science that sought to understand the origins of autonomy in teleological (i.e., goal-driven) artifacts.<sup>5</sup> Practitioners of the field produced some of the first examples of autonomous robots and other teleological mechanisms, including the “turtles” of Walter [24, 25], Theseus the mouse of Shannon [26], the “moth/bedbug” of Wiener [27], and the Homeostat of Ashby [3]. The vision of cybernetics was far more grand, however: the field was about the design of artificial brains, with abstract concepts of intelligence amplification<sup>6</sup> and even applications to government and control of national economies [29]. So where did cybernetics go? In short, though the interest in cybernetics did not die out, it did diminish with the rise of the modern digital computer and connectionism in the 1950s and 1960s. However, before waning it produced two very important offspring that have flourished: the modern field of control theory, and the inspiration behind the behavior-based paradigm of robotics.

The object of cybernetic inquiry was the engine of autonomy and goal-directedness in living organisms and machines, believing that the “stuff” of brains was the regulator. Yet, a glance at modern robotics work shows that regulation is not generally used as a motif for computation; rather, the paradigm of the programmed computer (and its custom analog, finite state machines) is dominant, with connectionism being used to a lesser extent. But should the role of cybernetics in robotics just be that of inspiration? We answer this to the contrary: cybernetics in this work offers both

---

<sup>4</sup>The work of Vaucanson influenced the Jacquard loom which in turn influenced Babbage’s analytical engine (nineteenth century).

<sup>5</sup>The precursors of the field stretch back through Maxwell and Watt to Heron of Alexandria and Ktesibios in antiquity.

<sup>6</sup>Intelligence amplifiers—theoretical constructs proposed in [28]—are analogous to electrical power amplifiers, with “intelligence” being the input and output.

inspiration *and* a science of synthesis.

Albus states:

“the same type of anatomical components which are used in the lower and mid levels of the control hierarchy to produce sensory interactive motor behavior may . . . be used at the upper levels of the same hierarchy to plan and solve problems.” [30]

This insight underlies the philosophy on the synthesis of artificial brains in this work: the use of *control systems* (traditionally restricted to the regulation of “low-level” activities) as a *computational paradigm* for all orders of agent behavior. To be sure, our motivation for adopting cybernetics stems not from a desire to resurrect an old, but venerable, line of inquiry. Rather it is based on engineering considerations:

- in contrast to connectionism (which generally relies on an iterative, empirical methodology), control theory provides a rigorous toolset for *principled* synthesis
- the continuous methods of control theory are an appealing match to a physical environment which is, at practical scales of perception, continuous [31]
- the tool is based on the language of dynamical systems theory, and as such it offers a mechanism to realize technology-independent specifications of computational machinery

Hence, this work is concerned with synthesizing the brains for cybernetic machines—autonomous robots; to that end, we employ cybernetic language and tools (dynamical systems and control theory) to perform this synthesis in a principled and technology-independent fashion.

## D. Contributions of this dissertation

Chapter II provides an overview of background information pertaining to multi-agent robotic systems, dynamical systems and control theory. A survey of relevant literature is also presented. In chapter III we formulate the problem addressed by this dissertation, and provide a discussion regarding our choice of cybernetics toolsets.

Next we deal with the design of elementary behaviors for our proposed scheme using cybernetics tools. Chapter IV presents a principled approach, grounded in dynamical systems theory, of synthesizing static (i.e., memoryless) behaviors. A novel visual tool—vector field design—is developed to aid the intuition as well as providing a rigorous basis for developing static couplings between sensing and actuation. Chapter V describes a control-theoretic approach to synthesizing behavior where sensation and actuation are decoupled through the interposing of a dynamical system. We highlight the weak emergence of satisficing intelligence—a key aspect of this controller.

Chapter VI presents an integrated control architecture for single agent systems using the elementary behaviors synthesized in chapter IV. The development of agent control architectures to enable collectives of agents to engage in useful multi-agent behaviors is then described in chapter VII. We highlight the engineering of weakly emergent behavior in these two chapters, and present a new scheme for inducing a collective of passively interacting agents to self-organize into a formation that covers a target.

Simulation results generated by our custom verification environment were used to characterize and illustrate the behaviors manifest by our robotic control schemes in a *virtual* planar environment. In chapter VIII, we study the performance of the navigation algorithms developed through our cybernetic approach with various other schemes found in the literature. Beyond simulation, however, since embodiment and

situatedness are central to our paradigm, chapter IX presents the design of a *physical* two-agent testbed and the associated experimental results that help to validate our approach.

We conclude the dissertation with some remarks on areas of future work.

## CHAPTER II

### PRELIMINARIES

Given an organism, its environment is defined as those variables whose changes affect the organism, and those variables which are changed by the organism's behaviour ... [The] organism and its environment form a single state-determined system ... (W. Ross Ashby [3])

#### A. Background

##### 1. Agents

An *agent* is a computational system coupled to an environment via sensors and actuators. Some examples of agents in general include human beings employed in a task, software agents (e.g., processes running on single computers, or across networks of computers), robotic agents (i.e., agents which are coupled to a physical environment), and, relevant to our work, mobile robots (agents that can change their position in a physical environment).

In this work we consider the design of *cognitive* faculties for autonomous robots, i.e., those faculties that enable an agent to derive, on its own, how to actuate change to the environment to achieve the design objectives of the agent. In particular, we consider autonomous robots with bounded resources (*lightweight* agents), constrained by:

- limited knowledge of the environment, including:
  - no a priori global information, such as a map of the environment
  - limited sensing radius
  - no knowledge of absolute position in space

- limited communications range
- bounded computational resources

and so invoke Simon’s concept of *satisficing* [5, 6] intelligence. Cognitive systems, in this sense, are systems that *satisfice*, forgoing optimality (which is often unachievable or impractical) to achieve tolerable solutions using whatever constrained resources they have in a timely fashion.

A *multi-agent system* (MAS) is a collection of agents; in particular, a *multi-agent robotic system* or *multi-robot system* (MRS) is a collection of robotic agents [32]. Consider a MRS consisting of autonomous robots. A *decentralized* MRS is one where the source of each agent’s autonomy lies purely within the agent itself and whatever local information it senses (precluding, for example, a central remote controller transmitting commands to agents). A trivial example of a multi-agent system is one where the agents are “asocial,” having no knowledge of other agents and operating as lone, isolated agents, unable to interact with others due to this ignorance.

Beyond this degenerate case, we can consider two agents,  $M$  and  $N$ , in more sophisticated schemes [33]. We define a *passive interaction* between  $M$  and  $N$  as an awareness on the part of  $M$  of an effect that  $N$  has on the environment that is purely a function of  $N$ ’s existence in the environment. That is, in a passive interaction  $N$  does not “decide” to exchange information to  $M$ . Rather,  $M$  by virtue of its sensory faculties is able to determine various aspects pertaining to  $N$  purely by virtue of  $N$ ’s existence. We say that  $N$  *actively interacts* with  $M$  if  $N$  “decides” to exchange information with  $M$ . That is, the information interchange occurs due to:

- $N$ ’s actuation of change in the environment, expecting  $M$  to sense this change
- $M$ ’s capacity to sense the changes induced by  $M$ ’s action, understanding that  $M$  actuated the change intentionally

The multi-agent extensions developed in this work (in chapter VII) are concerned with the design of lightweight cognitive faculties for mobile robotic agents, using passive interactions to achieve useful collective behaviors.

## 2. Dynamical systems and control theory

Our approach is cybernetic in the sense that we consider the agent and the environment to be coupled dynamical systems, where the cognitive faculties of the agent attempt to *control* (or regulate) the time evolution of various aspects of the environment [3, 34]. As such, concepts from dynamical systems theory figure prominently.

Suppose a system can be in any one of a number of states, and let its instantaneous state,  $\mathbf{x}(t)$ , belong to the *state space*,  $\mathcal{X}$  (the instantaneous time,  $t$ , being a member of some totally ordered *time set*,  $\mathcal{T}$ ). The evolution of  $\mathbf{x}(t)$  through  $\mathcal{X}$  as a function of time is specified mathematically by a dynamical system. Let  $\mathbf{f} : \mathcal{X} \rightarrow \mathcal{X}$ . If  $\mathcal{X} \subset \mathcal{R}^n$  and  $t \in \mathcal{T} \subset \mathcal{R}$ , then the ordinary differential equation:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \tag{2.1}$$

specifies a continuous-time continuous-valued dynamical system. When  $\mathcal{T}$  is equipotent to the set of integers,  $\mathcal{Z}$ , then the iterated map:

$$\mathbf{x}(t + 1) = \mathbf{f}(\mathbf{x}(t)) \tag{2.2}$$

specifies a discrete-time dynamical system (common state spaces for discrete-time dynamical systems include  $\mathcal{R}^n$  and  $\mathcal{Z}^n$ ). Given a system initially in state  $\mathbf{x}_0$  at time  $t_0$ , i.e.,  $\mathbf{x}(t_0) = \mathbf{x}_0$ , the *trajectory* of a dynamical system is the function  $\mathbf{x}(t)$  satisfying this initial condition and the equations of motion of the system (i.e., either (2.1) or (2.2)). In general, beyond linear dynamical systems, it is not easy to arrive at an analytically-tractable expression for the trajectory of a dynamical system. To analyze

the behavior of general systems, the qualitative theory of dynamical systems [35, 36] is used.

In this work, we use ordinary differential equations to specify continuous-time dynamical systems. For discrete-time systems we note that, beyond the specification of system equations of motion by iterated maps, digital redesign methods [37, 38] can be used to transform a continuous-time dynamical system to its corresponding discrete-time variant.

### a. Static versus dynamical control schemes

Consider the following dynamical system with input,  $\mathbf{a}$ , output,  $\mathbf{s}$ , and state,  $\boldsymbol{\eta}$ :

$$P : \begin{cases} \dot{\boldsymbol{\eta}} &= \mathbf{p}(\boldsymbol{\eta}, \mathbf{a}) \\ \mathbf{s} &= \mathbf{q}(\boldsymbol{\eta}, \mathbf{a}) \end{cases} \quad (2.3)$$

called a *plant*. Suppose we have some criterion we wish to impose on the time evolution of  $\boldsymbol{\eta}$ . For example, if  $\mathcal{X}$  is the state space of  $\boldsymbol{\eta}$ , suppose we wish to ensure that  $\lim_{t \rightarrow \infty} \boldsymbol{\eta}(t) \in \mathcal{X}^* \subset \mathcal{X}$ , where  $\mathcal{X}^*$  specifies a subspace of desirable plant states. The problem, then, is to specify a system,  $C$ , called the controller, that when coupled to  $P$  causes  $\boldsymbol{\eta}$  to evolve as desired.

A variety of rigorous techniques exist to synthesize  $C$ ; these form the corpus of control theory [39, 40]. Generally,  $C$  can be of two types:

- a *static* controller (dealt with in chapter IV), which is a memoryless system of the form:

$$C : \mathbf{a} = \mathbf{g}(\mathbf{s}) \quad (2.4)$$

- a *dynamical* controller (dealt with in chapter V), which refers to a dynamical



system of the form:

$$C : \begin{cases} \dot{\zeta} &= \mathbf{f}(\zeta, \mathbf{s}) \\ \mathbf{a} &= \mathbf{g}(\zeta, \mathbf{s}) \end{cases} \quad (2.5)$$

### 3. Embodiment and situatedness

An agent’s *embodiment* [41, 42] concerns the details of how it is coupled to the environment. The term is often used to differentiate embodied schemes of control from disembodied ones. In the former case, the details of the:

- sensors conveying information from the environment to the agent
- actuators conveying information from the agent to the environment

are important and are considered in formulating the control scheme. With the latter, the details of embodiment are abstracted away, and the control scheme operates on abstracted symbols and representations. The details of embodiment ultimately physically “ground” discussions of the system’s behavior with respect to the environment, that is, they enable one to cast the operation of the controller with respect to the environment.

The *situatedness* [43] of an agent refers to the influence of the environment’s dynamics on the agent.<sup>1</sup> Whereas the details of embodiment account for the information flow between the agent and the environment, the situatedness of an agent relates to how the environment evolves when the agent actuates change. An agent whose control scheme accounts for situatedness can often exploit the details of situatedness—i.e., the dynamics of the environment—to realize complex behaviors.

---

<sup>1</sup>As Arkin [44] puts it, situatedness is “... a strong two-way coupling between an organism and its environment.”

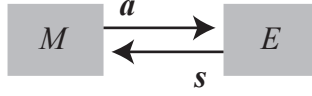


Fig. 4. An agent,  $M$ , coupled to the environment,  $E$ .

Consider the situation shown in Figure 4 where:

$$M : \begin{cases} \dot{\zeta} = \mathbf{f}(\zeta, \mathbf{s}) \\ \mathbf{a} = \mathbf{g}(\zeta, \mathbf{s}) \end{cases} \quad (2.6)$$

specifies the agent,  $M$ , and:

$$E : \begin{cases} \dot{\eta} = \mathbf{p}(\eta, \mathbf{a}) \\ \mathbf{s} = \mathbf{q}(\eta, \mathbf{a}) \end{cases} \quad (2.7)$$

specifies the environment,  $E$ . The embodiment of  $M$  concerns the sensory channel,  $\mathbf{s}$ , the actuation channel,  $\mathbf{a}$ , and their relation to the dynamics of  $M$ . Situatedness, on the other hand, refers to the dynamical structure of  $E$  and how the sensory and actuation channels are related to it. Note that by coupling  $M$  and  $E$ , the overall dimension of the composite state space,  $\begin{bmatrix} \eta \\ \zeta \end{bmatrix}$ , is increased; this suggests a greater behavioral richness that can arise when agents are situated—a richness that can be exploited.

## B. Previous work on robot control architectures

The control of mobile robotic agents is a problem with solutions from many disciplines. This survey focuses on control architectures to enable robots to deal, in real-time, with unknown environments using local information. Although the field of *machine planning* [45, 46] has developed a prolific array of powerful algorithms and techniques

for robotic motion planning, the frequent requirement for a priori information of the environment precludes its use in our application.

A plethora of diverse robot control architectures exist; we separate them into two broad classes:

- disembodied control (including deliberative or planner-based control [30, 47])
- embodied control (including reactive and behavior-based control [47])

Disembodied control is rooted in the *physical symbol system hypothesis* which states:

“[a] physical symbol system has the necessary and sufficient means for general intelligence action.” [48]

The idea here is that the cognition underlying machine actions can be realized by the execution of algorithmic processes that involve symbolic manipulation of an abstracted representation of the physical world (e.g., planning algorithms such as searches over a space of candidate solutions). These control architectures generally belong to the field of *core AI* [49] in computer science (often, somewhat disparagingly, referred to as “Good Old-Fashioned Artificial Intelligence”, or GOFAI [50]), and as such, generally use the constructs of automata theory [51] for synthesis and analysis. In this work we do not consider disembodied controllers on pragmatic grounds (regardless of our attitude towards this hypothesis) since the use of automata theory effectively restricts our implementation choice to digital hardware.<sup>2</sup>

Motivated by problems with the real-time performance of robots based on core AI techniques in complex dynamic environments [22], embodied approaches to control

---

<sup>2</sup>Although the work of [52] seems to suggest that, in principle, any Turing machine may be realized by an analog  $\mathcal{R}^3$  dynamical system.

were developed to produce more reactive systems where the mechanisms underlying the machine’s goal-directedness were closely “grounded” to physical details of the agents embodiment (i.e., its coupling to the environment via sensors and actuators) and situatedness within the physical environment. A statement of this point of view is the *physical grounding hypothesis*:

“...to build a system that is intelligent it is necessary to have its representations grounded in the physical world.” [53]

echoing earlier cybernetic points of view [3] (which themselves were preceded by earlier works in biology). Figure 5 presents a tree that classifies the major themes in the design of embodied cognition relevant to our work; the contribution of the proposed research to the field, *cybernetic automata*, is shown in perspective to other paradigms.

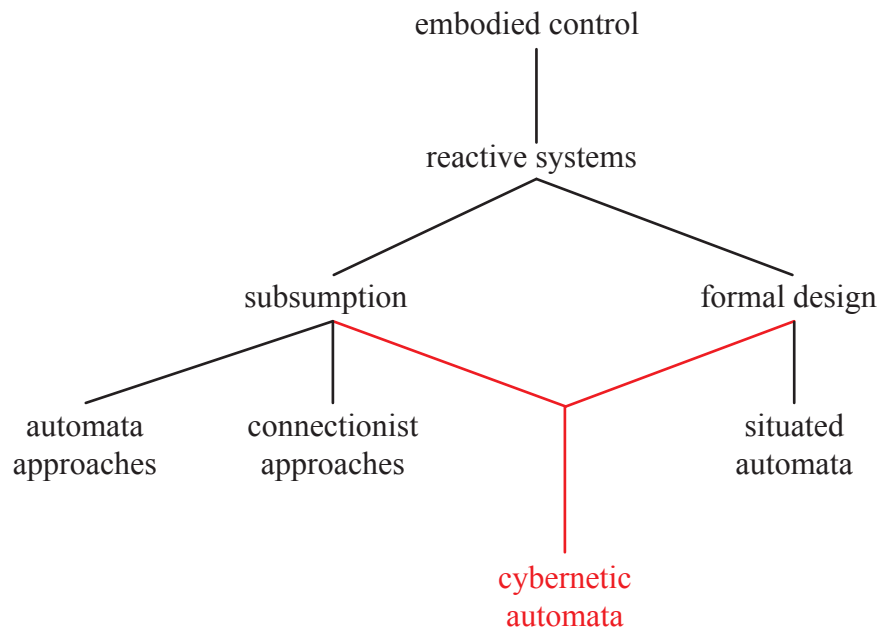


Fig. 5. Classification of embodied control schemes.

Often taking a dynamical systems approach to modeling and understanding the

embodiment of an agent within an environment, the historical antecedents of embodied control lie in cybernetics. Beyond the theoretical work regarding the underlying mechanisms of goal-directed machines and animals developed by researchers like Ashby, Wiener, and Shannon, the artificial turtles of Walter [24] provided some of the first physical examples of simple, reactive autonomous mobile robots. Taking an experimental approach, Walter investigated the emergence of behaviors reminiscent of living organisms that arose from basic reactive setups (e.g., static, feedforward connections from sensors to actuators). Adding feedback connections to introduce memory [25] resulted in elementary learning behavior. Continuing this empirical line of inquiry, the biological cyberneticist Braitenberg developed a series of thought experiments [54] to investigate the origin of decussations<sup>3</sup> in vertebrates. These thought experiments (virtual constructions of simple machines) were developed more fully in [55] to show how an array of seemingly sophisticated behavior can emerge from simple connections between sensory organs and motor organs.

The next phase of development contributed insights pertaining to the structural properties of cognition. Arbib’s work mated cybernetics and core AI with observations about the anatomy of brains and nervous systems, suggesting an approach that emphasized the “parallel activity of a multitude of operations within an array of interacting data and control schemes relevant to action” [56]. These insights into brain-like machine structure—hierarchical organization and layering of different control schemes—were further developed by Albus [30] who emphasized:

- the difference between planning algorithms and goal-seeking behavior in organisms

---

<sup>3</sup>A decussation is a set of nerve fibers connecting one side of the body and the opposite side of the brain. Braitenberg was interested in the influence of the connection topology on an organism’s movement in response to stimuli.

- the importance of sensing and reacting in real-time interactions with an environment, as opposed to planning
- a hierarchical structure for cognition based on control loops closed by feedback from the environment

Contemporary to these developments was Minsky’s *society of mind* theory which held:

“...each mind is made of many smaller processes ...each [process] by itself can only do some simple thing that needs no mind or thought at all. Yet when we join these [processes] in societies—in certain very special ways—this leads to true intelligence.” [57]

These insights were seminal to the subsequent prolific output in embodied control schemes.

The *subsumption architecture* [58] of Brooks is based on a set of elementary computational primitives that realize various levels of competence—behaviors that are germane to the agent’s overall goals. The units do not serve in a chain of command, like that illustrated in Figure 6, and do not require other primitives to handle sensing and action. Rather, each primitive has access to the agent’s sensors and actuators



Fig. 6. A control scheme that uses  $n$  processing steps to process sensor data and generate an action. Each step requires the others and can not, in isolation, control the agent.

and so can, individually, serve as a controller for the agent. To realize the overall control scheme, the primitives are organized in a layered architecture, as shown in Figure 7. With this topology, higher-levels controllers endow the agent with skills

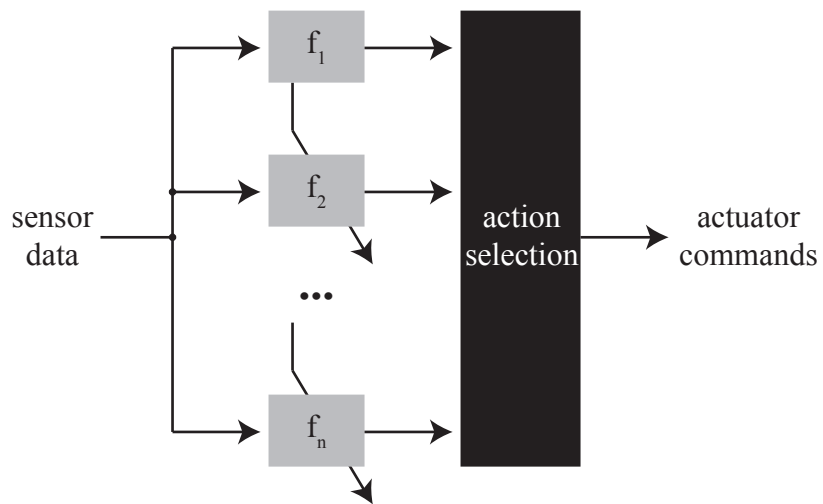


Fig. 7. A subsumption architecture with  $n$  primitive controllers. Each controller has access to the sensors and actuators and hence can, in isolation, control the agent. Higher-level controllers influence the operation of lower ones via tuning channels (shown by diagonal arrows). The action selection subsystem enforces the scheme's policy on how the individual controller outputs determine the overall action of the agent.

while subsuming the competencies provided by lower-level controllers (the upper levels are also able to influence the behavior of lower ones).

With the “society of minds” approach where several concurrent control mechanisms operate, a natural problem that arises is how to produce the overall action of the controller—*action selection* problem. If the controllers were all disjoint, dealing with separate problems and using separate actuators, the actuation signals could be output separately to the relevant actuators; however, in general, this is not possible. Brooks’ initial solution was to hardcode a priority scheme, while Arkin’s solution in early versions of the *AuRA* architecture [59] was to superpose (i.e., linearly sum) the outputs of each controller to produce a resultant action. Since it is not necessarily true that various controllers can have their outputs superposed or concatenated to create an overall actuator command, in [60] Maes considers an alternative approach based on a network of inhibitory and excitatory processes that work to produce a coherent resultant action. In this case, the action selection mechanism is a dynamic process evolving in the robot (as a function of its stimuli) that selects actions based on its state. Later versions of *AuRA* [61] used a hybrid approach of combining disembodied planning algorithms from core AI (to achieve action selection) with an underlying reactive controller (for real-time response).

Both Brooks and Arkin used finite state automata to realize their architectures. Beer’s approach of *computational neuroethology* [62], on the other hand, mated dynamical systems theory [63] with ethological studies of animal behavior [22], and applied connectionist methods [64, 65] as a synthesis toolset. Tilden’s work using *nervous nets* [23, 66] was an empirical continuation of Beer’s work, developing very economical controllers.

Contemporary to the development of the subsumption architecture, the *situated automata* approach of Rosenschein and Kaelbling [67, 68, 69, 70] introduced a



methodology for formal specification of reactive agents with *provable* properties, in contrast to other heuristic and empirical methods. This approach is based on modeling the agent and the environment by discrete dynamical systems (automata) coupled to one another. Agent design then becomes the synthesis of an automaton to influence the time-evolution of the environment automaton. The methodology provided tools (design languages) for a designer to specify characteristics of the environment and the agent (e.g., its goals) at a high-level which could then be compiled down to digital hardware realizations.

### C. Previous work on multi-agent systems

The work on multi-robot control architectures [33] can be divided into two major classes:

- schemes involving *explicit coordination*, that is, active inter-agent interaction
- schemes involving *implicit coordination*, that is, passive inter-agent interaction

#### 1. Explicit coordination

The former class is out of the scope of the proposed research, since the work of this proposal seeks a fully decentralized scheme with no requirements for active communications. However, we note some of the important contributions within this class, including:

- ACTRESS [71], a deliberative scheme where agents make heavy use of communications (e.g., for negotiation) to coordinate
- GOFER [72], a deliberative scheme using a centralized task planner and scheduler for coordination

- ALLIANCE [73], a behavior-based approach where agents use broadcast communications to inform other agents of their actions

Many of the control-theoretic works on the *state agreement*, *consensus*, and *formation control* problems [74, 75, 76, 77, 78] also fall within this class, as they require knowledge of the state of other agents. By contrast, [79] presents a scheme for formation control that involves minimal communications—agents periodically broadcast a message which enables the group to derive the status of the formation. Leonard [80] proposes a scheme for the cooperative control of mobile sensor networks based on the use of a centralized controller that uses information communicated by the agents to compute and communicate back a coordination signal.

## 2. Implicit coordination

Implicit coordination schemes use information about other agents passively sensed from the environment. This passive information can be in a variety of forms. For example, in Walter’s empirical work (described in [33]), agents were mounted with lights and endowed with light sensors. Hence, the interactions were passive as the agents could not modulate the lights, for example; however, they still formed a basis for interaction, as other agents could sense these lights and react based on this perception. Another example is that of *stigmergy* [81], whereby the actions of an agent produce a change in the environment, which in turn is sensed by other agents which respond. In this manner, agents have a basis to coordinate. Stigmergy is observed in the natural world (e.g., insect colonies); the principle has been applied to mobile robots (e.g., [82]).

Reynolds’ pioneering computer graphics work on behavioral models for the animation of bird flocking [83] provides an illustration of how superposition of simple behaviors such as:

- obstacle avoidance
- maintaining proximity to neighboring agents
- matching the velocity of neighboring agents

can lead to emergent behavior of the group as a whole.

Inspired by ethology, Mataric [84, 85] proposed the concept of *basis behaviors*:

“basis behaviors are stable, prototypical interactions between agents and the environment that evolve from the interaction dynamics and serve as a substrate for more complex interactions.” [85]

and presented a scheme based on superposition, to compose more complex group-level behavior, akin to Reynolds, but with an expanded repertoire of primitives.

In [86] Arkin presented a scheme for inter-agent cooperation based on agents being able to sense the presence of other agents, noting the emergence of phenomena such as the “recruitment” of multiple agents to work on a task. The relationship between inter-agent interactions in multi-agent robotic systems is further investigated in [87], with the observation that:

- in tasks with little passive interaction, the introduction of communications faculties improve performance
- in tasks where passive interactions exist, the introduction of further communication does not improve performance

#### **D. Perspective on our work**

The literature indicates a gap that this work seeks to address. First, note that the situated automata approach uses a discrete-time discrete-valued dynamical model of the

environment, from which a corresponding automaton is synthesized using automata-theoretic toolsets. This naturally begs the complementary use of continuous-time continuous-valued dynamical systems-based models of the environment, with corresponding control-theoretic means of rigorously synthesizing an analog automaton to regulate it.

We note that our work is not a conventional control-theoretic work. Our divergence from control theory is our focus on an overall control architecture for the agent. We use dynamical systems theory as a language to describe agents and their environments, and use control theory as a synthesis toolset to design basis behaviors. To realize the overall control scheme for an agent, however, we do not just mate a control algorithm with a robotic body. Rather, we develop an integrated architecture that stitches our elementary behaviors together to engineer the emergence of useful behavior. Hence, in this respect, we provide a cybernetic alternative to the connectionist and automata-theoretic approaches that exist in the literature. To our knowledge this combination of architectural insights with control-theoretic tools to engineer emergent behavior is a novel contribution to the field.

## CHAPTER III

### PROBLEM FORMULATION AND TOOLSETS

Be like the ant, consider her ways, and be wise: though having no ruler over her, neither anyone to guide her, she provides her bread in the summer and gathers her food in the harvest. (The Proverbs of Solomon)

#### A. Introduction

As discussed in the overview of literature in the previous chapter, approaches in robotic control can be broadly divided into embodied and disembodied ones; this work—a cybernetic behavior-based approach to robot cognition—belongs within the former class. The hallmarks of the cybernetic approach are an emphasis on embodiment and situatedness in grounding the formulation of artificial brains consistent with physical reality. In this work, embodiment enters through our use of realistic sensor and actuator models, while situatedness derives from:

- our modeling of how the agent’s actuation influences the evolution of the environment
- a consideration of which competencies are required for real-world applications

The later forms the topic of this chapter.

Search and rescue, and robotic exploration in general, are applications that are well-served by lightweight robotic agents with bounded resources, in contrast to a single “heavier” agent with greater resources [73, 80, 88]. The use of mobile sensor networks for such applications have been proposed in [89]. Our goal in this work is to develop a control architecture to serve as the robotic substrate underlying a mobile sensor network (i.e., the *mobility management* faculties) or an autonomous

multi-agent exploration system. In the following, we present a formulation of the competencies expected of such a substrate.

## B. Preliminaries

### 1. The world and objects therein

Let  $\mathcal{W} \subset \mathcal{R}^2$  represent a planar world. Consider a point object,  $X$ , in this world, and let  $\mathcal{T}$  denote a time set. The position of  $X$  with respect to an absolute global coordinate system imposed on  $\mathcal{W}$  is given by a map from  $\mathcal{T}$  into  $\mathcal{W}$ . We will denote this map by:

$$\mathbf{g}_X : \mathcal{T} \rightarrow \mathcal{W} \tag{3.1}$$

and refer to the value of the map at time  $t$ ,  $\mathbf{g}_X(t)$  as the *instantaneous position* of  $X$ . Further, we define the set of positions  $\mathbf{g}_X(\mathcal{T}) \subset \mathcal{W}$  (i.e., the image of  $\mathcal{T}$  under (3.1)) as the *path*<sup>1</sup> of  $X$  through  $\mathcal{W}$ . If  $X$  is static with respect to the global coordinate system, then the path of  $X$  is a singleton set and there is no confusion in referring to:

- the map  $\mathbf{g}_X(\cdot)$
- $X$ 's instantaneous position  $\mathbf{g}_X(t)$
- $X$ 's path

as simply  $\mathbf{g}_X$ . On the other hand if  $X$  moves with respect to the global coordinate system, then the various separate notations are needed. For a continuous-time system we have  $\mathcal{T} = [0, \infty) \subset \mathcal{R}$ , whereas for a discrete-time system,  $\mathcal{T} = \{0, 1, \dots\} \sim \mathcal{Z}$  (where  $A \sim B$  denotes the existence of a bijection between the sets  $A$  and  $B$ ). In the following we consider continuous-time systems.

---

<sup>1</sup>Also known as the *trace* of  $\mathbf{g}_X(\cdot)$  [90].

Figure 8 illustrates various objects (or *features*) in such a world at time  $t$ ; the global coordinate system is referenced to the axes  $g_1$  and  $g_2$ .  $M_i$  and  $M_j$  represent

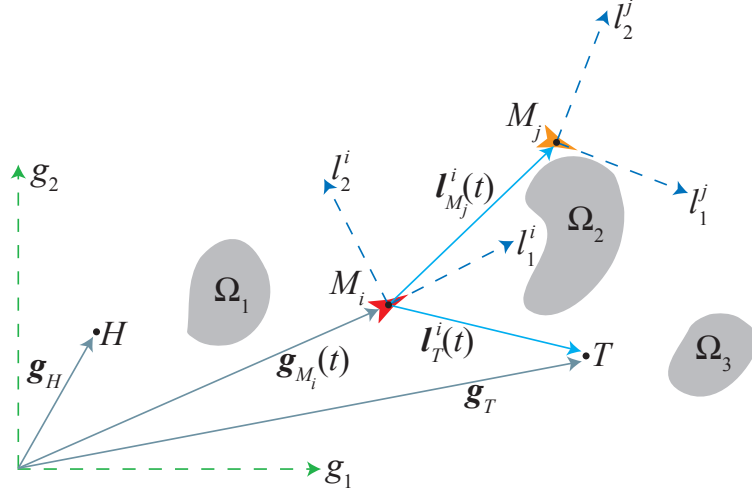


Fig. 8. The world and objects therein:  $M_i$  and  $M_j$  are agents, trying to get to a common target of interest,  $T$ , in an environment with obstacles,  $\Omega_1$ ,  $\Omega_2$ , and  $\Omega_3$ . Agent  $M_i$  senses the displacement to agent  $M_j$  as  $l_{M_j}^i$ , and the displacement to  $T$  as  $l_T^i$ .

agents  $i$  and  $j$ , respectively, of a group of  $N_M < \infty$  agents. Since mobile agents can change their position in  $\mathcal{W}$ , the corresponding path of a mobile agent  $M_j$ ,  $g_{M_j}$ , can have cardinality greater than 1.

We define an *obstacle* as a closed subset of  $\mathcal{W}$  within which an agent can not be located (i.e.,  $\forall i, g_{M_i}(T)$  can not intersect an obstacle). Given a set of  $N_\Omega < \infty$  obstacles in  $\mathcal{W}$ , we denote the  $j$ -th obstacle by  $\Omega_j$ .

We also define  $T$ , the *target way-point*, which is static with respect to the global coordinate system, and hence its corresponding path,  $g_T$ , is a singleton set.

## 2. Agent frame of reference

As we will later specify, we desire agents in  $\mathcal{W}$  to operate in a decentralized manner with only local information about their environments—that is, each agent observes the world with respect to a local frame of reference using sensors with limited sensing distance. Consider a local coordinate system attached to  $M_i$  such that  $M_i$  is at the origin. We orient the local axes of agent  $M_i$  as follows:

- the positive  $l_1^i$  axis,  $\{(l_1^i, l_2^i) \in \mathcal{W} : l_1^i > 0, l_2^i = 0\}$ , points in the *forward* direction of  $M_i$
- the negative  $l_1^i$  axis,  $\{(l_1^i, l_2^i) \in \mathcal{W} : l_1^i < 0, l_2^i = 0\}$ , points in the *reverse* direction of  $M_i$
- the positive  $l_2^i$  axis,  $\{(l_1^i, l_2^i) \in \mathcal{W} : l_1^i = 0, l_2^i > 0\}$ , is on the *left* side of  $M_i$
- the negative  $l_2^i$  axis,  $\{(l_1^i, l_2^i) \in \mathcal{W} : l_1^i = 0, l_2^i < 0\}$ , is on the *right* side of  $M_i$

Analogous to (3.1), we use:

$$l_X : \mathcal{T} \rightarrow \mathcal{W} \tag{3.2}$$

to denote the position of an object  $X$  with respect to the local coordinate system of  $M_i$ . Figure 8 illustrates the local coordinate systems for two agents and some sample local observations.

## C. Formulation

Here we formulate the competencies expected of a robotic substrate to serve in our application space. We are concerned with synthesizing the cognitive faculties that must be endowed in an individual agent so that a set of these agents can:

1. autonomously navigate to a target waypoint,  $T$



2. self-organize to cover a region about  $T$

### 1. Autonomous navigation

Figure 9 illustrates the general setup of the problem of autonomous navigation through an obstacle ridden environment.

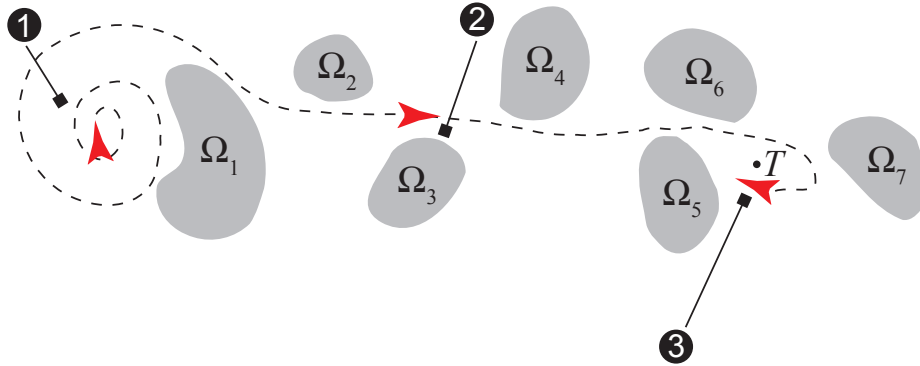


Fig. 9. The autonomous navigation problem. (1) An agent (in red) located in a region where it can not perceive the target, executes an approximation of a space-filling curve which eventually enters a region within sensing range of  $T$ . (2) The agent navigates to  $T$ , executing a collision-free path about the obstacles,  $\Omega_1, \dots, \Omega_7$ , and (3) eventually reaches  $T$ .

As illustrated, the agent requires two basic skills:

- the ability to search for a target
- the ability to engage in *taxis* behavior to track the target

constrained by the requirement for obstacle avoidance.

#### a. Exploration with obstacle avoidance

Exploration (or searching) is behavior that increases the agent's chance of coming within sensing range of the target. Let  $t_f > 0$ ,  $\Delta = [0, t_f] \subset \mathcal{T}$  be an interval in time,

and  $r_{T,\max}$  be the maximum distance from the target from which the target can still be sensed by an agent. Recall that the set  $B(\mathbf{g}_*; r)$  is the set of points,  $\mathbf{g}$ , such that  $\|\mathbf{g} - \mathbf{g}_*\|_2 < r$ . A search strategy for  $T$  is the selection of a path,  $\mathbf{g}(\Delta)$ , such that:

$$\mathbf{g}(\Delta) \cap B(\mathbf{g}_T; r_{T,\max}) \neq \emptyset \quad (3.3)$$

Let  $\mathbf{g}(0)$  be the initial position of the agent. A useful characteristic of a search strategy is when:

$$r_S(t) := \|\mathbf{g}(t) - \mathbf{g}(0)\|_2 \quad (3.4)$$

is a monotonic increasing function of  $t, \forall t \in \Delta$ . This indicates that the search strategy is an unbiased search about the agent's initial position. Finally, a viable search strategy must satisfy the constraint of obstacle avoidance:

$$\mathbf{g}(\Delta) \cap \left(\cup_{j=1}^{N_\Omega} \Omega_j\right) = \emptyset \quad (3.5)$$

### b. Target tracking with obstacle avoidance

Let  $t_f > t_e > 0$ , and  $r_{T,\min} > 0$  specify the desired proximity from  $T$  we wish the agent to reach. Then we can specify target tracking as the determination of a path,  $\mathbf{g}(\mathcal{T})$ , such that,

$$\mathbf{g}([t_e, t_f]) \in B(\mathbf{g}_T; r_{T,\min}) \quad (3.6)$$

Again, a viable path must be collision-free and satisfy:

$$\mathbf{g}([0, t_f]) \cap \left(\cup_{j=1}^{N_\Omega} \Omega_j\right) = \emptyset \quad (3.7)$$

## 2. Organization with respect to other agents

Autonomous navigation faculties are important; however, in a multi-robot system the agents are not intended to be single entities achieving their goals in isolation.

Rather, they are to be components of an overall group with group-level objectives. Specifically, in a sensor network we require the group to:

- maintain network connectivity (i.e., a communications channel for the sensor network)
- maintain network coverage (i.e., be distributed spatially to increase the set of points in  $\mathcal{W}$  from which the network can be accessed)

Moreover, we desire these group-level properties to be present both:

- while agents are navigating to the target
- when agents have arrived at the target

Since we are concerned with mobile robotic control, we re-cast these goals pertaining to network-level properties to goals of spatial organization since motion through space and sensing of spatial characteristics (e.g., displacements) are more elementary forms of information (and more readily accessible) to a lightweight mobile robot.

### a. Flocking

Flocking [83] is a behavior where agents move as a collective, maintaining proximity to other agents in addition to achieving their individual goals. This behavior can be helpful to (roughly) synchronize the arrival of the collective of agents to the target, as well as to ensure that agents that can not detect the target (e.g., due to an obstacle that obscures the target, due to failure of target sensors, or due to agents getting trapped in concave obstacle formations) have a secondary reference to track (in this case, neighboring agents). In cases where the target waypoint is out of detection range, flocking also enables “swarm”-like [91] searching of the territory wherein each agent engages in searching while maintaining the flocking conditions (3.8) and (3.9).

Let  $\mathcal{T}_f$  be a closed, connected subset of  $\mathcal{T}$ . We formulate flocking as the composition of the following three objectives that a group of agents strive to satisfy during the time interval  $\mathcal{T}_f$ :

- (safety) maintaining a collision-free path:

$$\begin{aligned} & (\forall i \in \{1, \dots, N_M\}) \\ & [\mathbf{g}_{M_i}(\mathcal{T}_f) \cap (\cup_{j=1}^{N_M} \Omega_j) = \emptyset] \end{aligned} \tag{3.8}$$

- (social goal) maintaining bounded proximity to other agents:

$$\begin{aligned} & (\forall i \in \{1, \dots, N_M\})(\forall t \in \mathcal{T}_f) \\ & \text{upper bound: } \left\{ \begin{array}{l} (\exists j \in \{1, \dots, N_M\}, j \neq i) \\ (\|\mathbf{g}_{M_j}(t) - \mathbf{g}_{M_i}(t)\|_2 < r_{A,\max}) \end{array} \right. \end{aligned} \tag{3.9}$$

$$\text{lower bound: } \left\{ \begin{array}{l} (\forall k \in \{1, \dots, N_M\}, k \neq i) \\ (\|\mathbf{g}_{M_j}(t) - \mathbf{g}_{M_i}(t)\|_2 > r_{A,\min}) \end{array} \right.$$

- (group goal) navigating to, and reaching, some region about the target waypoint,  $T$ :

$$\begin{aligned} & (\forall r_{T,\text{ref}} > 0)(\forall i \in \{1, \dots, N_M\}) \\ & [\exists \Delta \subset \mathcal{T}_f, \Delta \text{ is connected, } \max(\Delta) = \max(\mathcal{T}_f)] \\ & [\mathbf{g}_{M_i}(\Delta) \subset B(\mathbf{g}_T; r_{T,\text{ref}})] \end{aligned} \tag{3.10}$$

## b. Static organization

Having arrived at the target, the agents must organize into a network that maintains connectivity and coverage. In terms of spatial distribution, this is none other than maintaining the social goal of flocking (3.9).

### 3. Remarks

We note that the formulations in this section were made with respect to the global frame of reference. This was done to specify the problem in general terms, so as to be applicable to a variety of solution strategies (e.g., planning algorithms [45, 46]).

The proposed research addresses the problem of how to endow a set of homogeneous, lightweight, mobile sensor nodes with correspondingly lightweight cognition to cope with an unknown environment. We make the following assumptions to constrain our solution in line with this theme:

**Assumption 1 (No global knowledge).** *Agents do not have a priori knowledge (e.g., maps) of the locations of any features of  $\mathcal{W}$ .*

**Assumption 2 (Local sensing).** *Agents can only sense the position of features within a finite detection radius (specific to the feature under consideration) of the agent.*

**Assumption 3 (Local frame of reference).** *The sensed position of any feature is with respect to the agent's own local coordinate system.*

**Assumption 4 (Decentralization).** *There is no centralized coordinating controller—agent's must make decisions autonomously using the information that's locally available.*

**Assumption 5 (No communications).** *There is no communication channel available to the agent to engage in active interactions with other agents.*

In subsequent chapters, we will develop a solution that uses local strategies in line with the assumptions made here.

## D. Proposed toolset

We require tools to develop the algorithms that will realize the cognitive faculties of the agent. As illustrated in Figure 1, an agent is a system that is coupled to an environment via sensors and actuators. Cognitive faculties, in this sense, refer to some set of operators and dynamical processes that, based on:

- the internal state of the agent
- the perceived state of the environment (as determined by sensed measurements,  $\mathbf{s}$ , of  $E$ )

provides a specification for how the agent should actuate change to the environment via  $\mathbf{a}$ .

We identify three major classes of algorithm development tools:

- automata theory
- computational intelligence methods
- cybernetic tools based on control theory

In the following, we describe the characteristics desired in a good toolset and then discuss our choice of toolset.

### 1. Desired characteristics

**Rigor** Although the design of machines with well-characterized behavior is the hallmark of the engineering approach to system development (versus an empirical approach), in applications where human lives are involved (e.g., search and rescue, land-mine cleanup), funding is limited (e.g., ventures by public agencies including national security and “Big Science”), and windows of opportunity are limited (e.g.,

oceanic or space exploration), the need for a correctly behaving system is paramount. Hence, we want algorithm development toolsets that are rigorous, enabling:

- the quantitative analysis of the world
- the quantitative specification of our objectives
- the synthesis of a corresponding algorithm based on the above analyses
- the ability to prove that the synthesized algorithm operating in the world will achieve the above objectives

**Amenable to economical implementation** Due to the applications we are targeting, the cost of individual agents is an important factor as mentioned in the Introduction of this proposal. Thus, to facilitate the goal of lightweight agents, the algorithms we synthesize must be amenable to realization on economical implementation technologies, such as analog electronics, and customizable technologies, such as digital hardware.

**Direct mapping to an implementation substrate** Analogous to our interest in algorithms with provable properties, we also want a toolset that can be directly mapped onto an implementation substrate to ensure that a robust design methodology can be developed where the algorithm and its implementation can be easily verified to be equivalent.

## 2. Automata theory

The sequential automata tools of computer science involve the specification of algorithms as sequential (discrete-time) processes that manipulate discrete-valued quantities. Examples of such formulations are finite state automata, or sequential algorithms

for the various other classes of discrete automata [51]. The toolset is a very mature one with a long history of successful application to diverse areas; as a consequence there are many methodological tools available to help the designer, including design libraries, high-level specification languages, and verification tools.

With respect to the rigorous specification of behaviors for robotic agents, the work of Rosenschein and Kaelbling [67, 68, 69, 70] on the formal synthesis of automata based on discrete dynamical models of the environment illustrates the formal power of automata-theoretic tools. Moreover, automata-theoretic algorithms can be directly mapped to custom digital hardware. However, at the same time, direct maps between an algorithmic specification using automata theory and an analog implementation are generally not possible.<sup>2</sup>

In [93], Lumelsky cites two reasons to motivate continuous feedback-control based approaches to navigation. First, he mentions that the translation of continuous real-world phenomena to discrete structures (e.g., graphs) can be sensitive to approximation error, resulting in unacceptable planning choices. He further states that the approximation process itself can have high computational costs with non-intuitive results, degrading real-time performance. In the literature of the burgeoning fields of unconventional, natural, and organic computing, one occasionally finds views suggesting the automata-theoretic paradigm may not necessarily be the most natural means of realizing robotic control architectures:

“Any robot relying on a discrete representation for its successful functioning may become brittle, with small errors causing the robot’s beliefs

---

<sup>2</sup>Moore [52] and Sato [92] have obtained interesting results suggesting that a Turing machine, and other automata, can be embedded in the smooth flow of a  $\mathcal{R}^3$  dynamical system—enabling analog circuit implementations. However, these methods require the development of a switching map (which essentially realizes the algorithmic control flow) for which a synthesis method is not provided.



about the world to diverge from reality.” [31]

These views are usually based on the view that the physical world is dynamically complex, with a multitude of parallel processes interacting on different time scales, and question the “fit” of automata-based algorithms to real-world environments.

We do note, however, that concurrent sequential automata have been used by Brooks, Arkin, and others to realize robotic control architectures. Hence, we do not discount the utility of sequential automata as a mechanism for specifying cognitive systems (i.e., we do not enter the debate as to whether sequential automata is inherently applicable or inapplicable to robot cognition). Rather, we are motivated to look elsewhere for a paradigm that has more diverse implementation options.

### **3. Computational intelligence**

Computational intelligence methods mimic the phenomena of nature to solve problems. For example, evolutionary computing utilizes ideas culled from the development of an organism’s genome over several generations. Connectionist methods, on the other hand, use an underlying biomimetic template framework of a computational system modeled on the brains of organisms and combine this with a learning procedure.

Economic implementations of systems designed in this paradigm are possible via analog hardware. However, as computational intelligence tools use some form of evolution or training over time and are generally empirical, the synthesis procedure is not very direct. It is unclear as to whether rigorous statements can be made about the characteristics of the resulting systems (e.g., along the lines of sequential automata). Hence, we do not consider computational intelligence toolsets to be in line with our needs.

#### 4. Cybernetics tools

Given a dynamical systems model of the environment, and a specification of how the environment states should evolve, control-theoretic tools generally offer a means of synthesizing a corresponding controller. These controllers have provable properties (owing to the grounding of control theory in dynamical systems theory, analysis, algebra, geometry, etc.), and can be directly mapped to analog electronic hardware, and, via digital redesign methods, digital hardware.

Table I provides a summary of the discussion of this section.

Table I. Summary of toolset considerations.

	automata-theoretic tools	computational intelligence	cybernetic tools
rigorous?	yes	not yet	yes
analog?	not practical	yes	yes
direct map?	yes	yes	yes

## CHAPTER IV

### STATIC SCHEMES FOR AGENT BEHAVIOR

Yet to be written is the book, much larger in size, that shall show how all of the organism’s exteriorly-directed activities—its “higher” activities—are all similarly regulatory . . . (W. Ross Ashby [34])

#### A. Introduction

The behavior-based paradigm for multi-agent systems (succinctly summarized in [94]) involves:

- the design of low-level elementary behaviors that couple an agent’s sensory faculties to its actuators
- the design of integrated control architectures that employ these basis behaviors in a scheme where the various behaviors are selected in response to environmental stimuli
- the use of several of these agents in schemes that exploit their collective behavior

This work addresses all three aspects of the paradigm. In this chapter and the next, we present the design of elementary behaviors for autonomous navigation using cybernetic tools culled from control theory and the theory of dynamical systems to approach this synthesis in a principled manner.

Our perspective throughout this work is that behaviors spring from an agent’s regulation of its perception<sup>1</sup> of the environment, that is:

---

<sup>1</sup>We hence associate behavior with sensor output regulation [95].

*an agent acts to effect change in the world in order to cause its sensory perception of the world to evolve in a desirable manner.*

Embodiment and situatedness, hence, figure prominently in our philosophy: the details of embodiment are required in order to specify the sensory and actuation faculties of the agent, and the details of the agent’s situatedness specify how the environment reacts to the agent’s actions and, more broadly, govern the context for what “desirable” means.

In this chapter, we will be concerned with direct couplings from sensor information to actuator commands via static maps<sup>2</sup>—representing the extreme case of purely reactive<sup>3</sup> robotic control. In line with the important role played by embodiment and situatedness, we first describe the sensory and actuation faculties that couple the agent to the world. This enables us to derive a model describing how the agent’s perception of the world evolves as a function of its actuation. The model of agent perception forms the basis for synthesizing static control laws to realize various behavioral modes. For this synthesis, we present the use of a novel visual tool—*vector field design*—that appeals to the intuition while enabling mathematically rigorous specifications. Simulation results illustrating the nature of these behaviors are presented before concluding the chapter.

---

<sup>2</sup>That is, *memoryless* systems. Structural speaking, these are systems where there are only *feedforward* paths from sensation to actuation.

<sup>3</sup>“Reactive” is often loosely applied to a variety of robotic control schemes, including those in which state machines—dynamical systems—mediate between sensation and actuation. Hence, we emphasize the *pure* reactivity that the feed-forward maps of our work realize.

## B. Describing perception

Since we associate agent behavior with regulation of perception, we need a model that describes the temporal evolution of the agent’s sensory perception of the world as a function of the agent’s actions. In this section, we discuss the sensori-motor embodiment [41] of our prototypical agent, and derive a model of agent perception useful for autonomous navigation.

### 1. Sensors

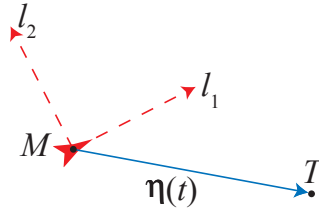


Fig. 10. An agent,  $M$ , to which a local coordinate system has been attached. The agent is sensing a target of interest,  $T$ , whose instantaneous position with respect to  $M$ ’s local frame of reference is  $\eta(t)$ .

Figure 10 illustrates an agent in a planar environment. Its sensors give the agent a measurement,  $\mathbf{s}$ , of the relative position,  $\eta$ , of a target of interest with respect to the agent’s local frame of reference. Practical sensors, however, are non-ideal devices which measure physical quantities subject to various forms of distortion. We first set a minimum standard on the fidelity we expect from our sensory apparatus.

**Definition 1 (Measurement Functions).** *The map  $\sigma : \mathcal{R} \rightarrow \mathcal{R}$  is a measurement function if it is a bounded, continuous, bijection such that  $\forall x \in \mathcal{R}, \text{sgn}(\sigma(x)) = \text{sgn}(x)$ .*

Consider the situation of Figure 10, where  $\eta = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix}$  denotes the position of

the target,  $T$ , with respect to the agent’s local  $l_1 - l_2$  coordinate system. Let  $\mathcal{S}$  be a compact subset of  $\mathcal{R}^2$  that contains  $\mathbf{0}$ . We specify the target sensor,  $S$ , as a memoryless system that returns its measurement of the target position,  $\mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \in \mathcal{S}$ :

$$\mathbf{s} = \boldsymbol{\sigma}(\boldsymbol{\eta}) := \begin{bmatrix} \sigma_1(\eta_1) \\ \sigma_2(\eta_2) \end{bmatrix} \quad (4.1)$$

where  $\sigma_1$  and  $\sigma_2$  are arbitrary measurement functions.

## 2. Actuators

In developing basic behaviors for navigation, we assume that low-level motor controllers exist with sufficient competence to physically realize these velocity commands<sup>4</sup> (e.g., by dealing with physical issues such as actuator dynamics). For the development we present, we deal with kinematic models [46] for vehicles, which abstract away low-level concerns pertaining to vehicle dynamics.

One such kinematic model for wheeled vehicles is that of the simple unicycle whose motion is described by its signed translational speed,  $v$ , and its signed rotational speed,  $\omega$ . This model can be used to describe the kinematics of differential-drive vehicles and model car-like ones. Figure 11 illustrates a unicycle with respect to a global frame of reference. The trajectory executed this vehicle with respect to this frame is described by:

$$\begin{aligned} \dot{\mathbf{x}} &= v \begin{bmatrix} \cos(\psi) \\ \sin(\psi) \end{bmatrix} \\ \dot{\psi} &= \omega \end{aligned} \quad (4.2)$$

Let the compact set  $\mathcal{A} \subset \mathcal{R}^2$  contain  $\mathbf{0}$ . We specify our actuator,  $A$ , as a

---

<sup>4</sup>That is, the low-level motor controller operates on a faster time scale than the controllers which regulate agent behavior.

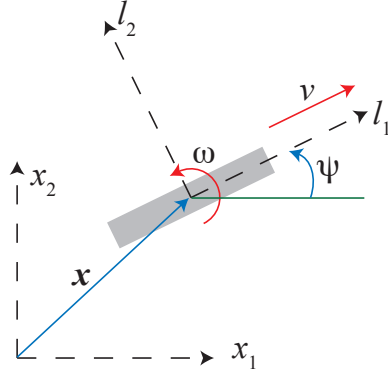


Fig. 11. Top-view of a simple unicycle (in grey) to which a local  $l_1 - l_2$  coordinate system has been attached. The directions of positive  $v$  and  $\omega$  are indicated by red arrows. The unicycle's position and orientation with respect to a global  $x_1 - x_2$  coordinate system (unattached to the unicycle) are denoted by  $\mathbf{x}$  and  $\psi$ .

memoryless system that is driven by the actuation signal  $\mathbf{a}$ :

$$\mathbf{a} = \begin{bmatrix} a_v \\ a_\omega \end{bmatrix} \in \mathcal{A} \quad (4.3)$$

where  $a_v$  and  $a_\omega$  are independent motion commands for translation and rotation, respectively. It instantaneously achieves this commanded velocity in the physical environment so that:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \mathbf{a} \quad (4.4)$$

We note that this specification models the set of lower-level controllers as an identity operator; we appeal to the time-scale separation between the behavioral controllers we discuss in this chapter and the lower-level controllers to justify this.

### 3. Plant model for perception

Given the aforementioned details of embodiment, we can now derive our model of agent perception,  $P$ , relating the time-evolution of the agent's sensory perception of

the world,  $\mathbf{s}$ , to the agent's actuation command,  $\mathbf{a}$ .

Let  $\boldsymbol{\eta}(t)$  be the displacement from the agent to a target of interest at time  $t$ . Consider an infinitesimal increment in time,  $h$ , during which the agent applies the actuation:

$$\mathbf{a} = \begin{bmatrix} a_v \\ a_\omega \end{bmatrix} \quad (4.5)$$

and recall our assumption that the lower-level motor controllers have the competence to achieve commanded velocities instantaneously. From the kinematics of a simple unicycle (4.2), we obtain the change in orientation of the agent:

$$\delta_\psi := \psi(t+h) - \psi(t) = a_\omega h \quad (4.6)$$

Moreover, from Figure 12, we see that during time interval  $h$  the agent translates to a new location whose displacement from its original position is:

$$\mathbf{b} = vh \begin{bmatrix} \cos(\delta_\psi) \\ \sin(\delta_\psi) \end{bmatrix} \quad (4.7)$$

With respect to the agent's frame of reference at time  $t$ , the displacement to the target is:

$$\mathbf{c} = \boldsymbol{\eta}(t) - \mathbf{b} \quad (4.8)$$

Since the frame of reference at time  $t+h$  been rotated by  $\delta_\psi$ , we obtain  $\boldsymbol{\eta}(t+h)$ , the displacement from the target to the agent—with respect to the frame of reference at time  $t+h$ )—by computing:

$$\boldsymbol{\eta}(t+h) = M(-\delta_\psi)\mathbf{c} \quad (4.9)$$



where:

$$M(z) = \begin{bmatrix} \cos(z) & -\sin(z) \\ \sin(z) & \cos(z) \end{bmatrix} \quad (4.10)$$

is the two-dimensional rotation matrix.<sup>5</sup>

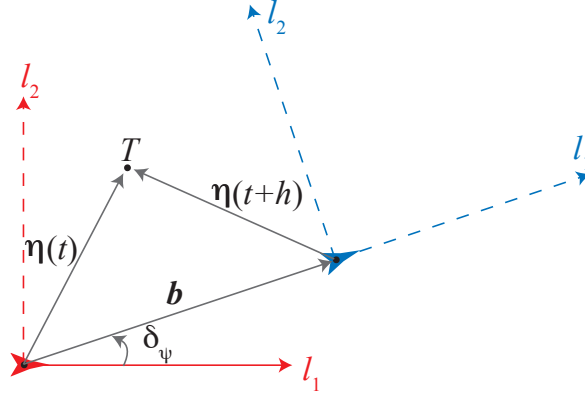


Fig. 12. The agent at time  $t$  (shown in red with its local coordinate system), and at time  $t + h$  (in blue). Also illustrated are the corresponding displacements to the target  $T$  ( $\boldsymbol{\eta}(t)$  and  $\boldsymbol{\eta}(t+h)$ , respectively), and the displacement,  $\mathbf{b}$ , and angle,  $\delta_\psi$ , between the two frames of reference.

Now we take the limit:

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \lim_{h \rightarrow 0} \frac{1}{h} [\boldsymbol{\eta}(t+h) - \boldsymbol{\eta}(t)] = \lim_{h \rightarrow 0} \frac{1}{h} [M(-\delta_\psi)(\boldsymbol{\eta}(t) - \mathbf{b}) - \boldsymbol{\eta}(t)] \\ &= \lim_{h \rightarrow 0} \frac{d}{dh} [M(-\delta_\psi)(\boldsymbol{\eta}(t) - \mathbf{b}) - \boldsymbol{\eta}(t)] \\ &= \begin{bmatrix} -a_v + \eta_2 a_\omega \\ -\eta_1 a_\omega \end{bmatrix} \end{aligned} \quad (4.11)$$

leading to the model:

$$P : \begin{cases} \dot{\boldsymbol{\eta}} = \mathbf{p}(\boldsymbol{\eta}, \mathbf{a}) := \Upsilon(\boldsymbol{\eta})\mathbf{a} \\ \mathbf{s} = \begin{bmatrix} \sigma_1(\eta_1) \\ \sigma_2(\eta_2) \end{bmatrix} \end{cases} \quad (4.12)$$

<sup>5</sup>That is, for  $\mathbf{x} \in \mathcal{R}^2$ ,  $\mathbf{y} = M(\phi)\mathbf{x}$  is the vector that results from rotating  $\mathbf{x}$  by  $\phi$  in a counter-clockwise sense about the origin.

where:

$$\Upsilon(\boldsymbol{\eta}) = \begin{bmatrix} -1 & \eta_2 \\ 0 & -\eta_1 \end{bmatrix} \quad (4.13)$$

and  $\sigma_1$  and  $\sigma_2$  are arbitrary measurement functions.

### C. Regulating perception

Armed with our model of agent perception,<sup>6</sup> (4.12), we are now in a position to synthesize our computational machinery, that is, the controllers to regulate this perception.

The goal of a taxis behavior is for the agent to reach the target; hence the agent must act to regulate its perception of the target (corresponding to its displacement to the target) so that this perception converges to the sensation  $\mathbf{s} = \mathbf{0}$ —that the agent *is* at the target.<sup>7</sup> We emphasize that from the agent’s point of view, it is striving to regulate its sensory perception of the world ( $\mathbf{s}$ ) to a desirable level ( $\mathbf{0}$ ).

With respect to the plant model (4.12), our task is to develop a feedback law that specifies the actuation,  $\mathbf{a}(\boldsymbol{\eta})$ , such that the resulting closed loop system:

$$\dot{\boldsymbol{\eta}} = \mathbf{p}(\boldsymbol{\eta}, \mathbf{a}(\boldsymbol{\eta})) := \widehat{\mathbf{p}}(\boldsymbol{\eta}) \quad (4.14)$$

has a single globally asymptotically stable equilibrium point at  $\boldsymbol{\eta} = \mathbf{0}$ :

$$\widehat{\mathbf{p}}(\boldsymbol{\eta}) = \mathbf{0} \iff \boldsymbol{\eta} = \mathbf{0} \quad (4.15)$$

$$\lim_{t \rightarrow \infty} \boldsymbol{\eta}(t) = \mathbf{0} \quad (4.16)$$

By regulating the agent’s perception to bring  $\boldsymbol{\eta}$  to  $\mathbf{0}$ , this actuation law will serve to

---

<sup>6</sup>We will refer to this model as the *plant model*.

<sup>7</sup>Recall that  $\mathbf{s}(\boldsymbol{\eta}) = \boldsymbol{\eta} = \mathbf{0} \iff \boldsymbol{\eta} = \mathbf{0}$  by definition of the measurement functions relating  $\mathbf{s}$  to  $\boldsymbol{\eta}$ .

bring the agent to the target.

In the following we present two approaches to synthesize controllers to this end.

### 1. An analytic approach

Using the methodology of Lyapunov synthesis, we consider the positive definite Lyapunov function candidate:

$$V := \frac{1}{2} \boldsymbol{\eta}^T \boldsymbol{\eta} \quad (4.17)$$

with time derivative:

$$\dot{V} = \boldsymbol{\eta}^T \dot{\boldsymbol{\eta}} = \boldsymbol{\eta}^T \mathbf{p}(\boldsymbol{\eta}, \mathbf{a}) \quad (4.18)$$

Since we only have access to the measured plant state,  $\mathbf{s}$ , we set:

$$\mathbf{a} = -\frac{1}{s_1} \begin{bmatrix} -s_1 & -s_2 \\ 0 & -1 \end{bmatrix} \mathbf{s}, \quad (4.19)$$

(when  $s_1 \neq 0$ ) which makes:

$$\dot{V} = -\left( \eta_1 s_1 + \frac{\eta_1}{s_1} s_2^2 \right) < 0, \boldsymbol{\eta} \neq 0 \quad (4.20)$$

since  $\text{sgn}(s_1) = \text{sgn}(\sigma_1(\eta_1)) = \text{sgn}(\eta_1)$  and  $\text{sgn}(s_2) = \text{sgn}(\sigma_2(\eta_2)) = \text{sgn}(\eta_2)$ . Thus, by Lyapunov's stability theorem,<sup>8</sup> (4.19.) can be used to asymptotically stabilize the state  $\boldsymbol{\eta} = \mathbf{0}$  of  $P$ —as long as  $s_1 \neq 0$ .

However, this restriction is unsatisfactory since:

- it precludes cases where the agent may need to enter a configuration that is at right angles to the target (since  $\eta_1 = s_1 = 0$  for these cases)
- there is nothing particularly special about this orientation that should preclude tracking (i.e., the agent knows where the target is, and so ought to be able to

---

<sup>8</sup>Theorem 4.1 of [96].

reach it)

## 2. Vector field design: a graphical approach

To synthesize a better controller, we note that our goal is to design  $\mathbf{a}(\boldsymbol{\eta})$  so that the right-hand side vector field<sup>9</sup> of the closed-loop plant equations (4.12),  $\mathbf{p}(\boldsymbol{\eta}, \mathbf{a})$ , has desirable properties. Hence, we can consider designing such a desirable vector field directly; if  $\widehat{\mathbf{p}}$  denotes the designed candidate vector field, we then need only solve  $\mathbf{p}(\boldsymbol{\eta}, \mathbf{a}) = \widehat{\mathbf{p}}(\boldsymbol{\eta})$  for  $\mathbf{a}$  to obtain our control law. Since visualizations of mathematical abstractions often aid the intuition, we present the use of a visual aids for designing desirable vector fields.

We first identify the qualitative properties required of  $\widehat{\mathbf{p}} = \begin{bmatrix} \widehat{p}_1 : \mathcal{R}^2 \rightarrow \mathcal{R} \\ \widehat{p}_2 : \mathcal{R}^2 \rightarrow \mathcal{R} \end{bmatrix}$ . As with our earlier controller synthesis, we require (4.15)-(4.16). Additionally, to facilitate the design of a control law that is compatible with the plant (e.g., to prevent the singularity that arose in our earlier synthesis), we require that the structure of  $\widehat{\mathbf{p}}$  be consistent with the plant model, in the sense:

$$(\forall \boldsymbol{\eta} = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \in \mathcal{R}^2 : \eta_1 = 0) (\widehat{p}_2(\boldsymbol{\eta}) = 0) \quad (4.21)$$

### a. An unsatisfactory solution

To highlight the need for this last requirement (4.21), consider a scheme that strives to regulate its perception of the target by bringing  $\boldsymbol{\eta}$  to  $\mathbf{0}$  in the most direct fashion possible. Figure 13(a) illustrates the qualitative properties of a vector field for just

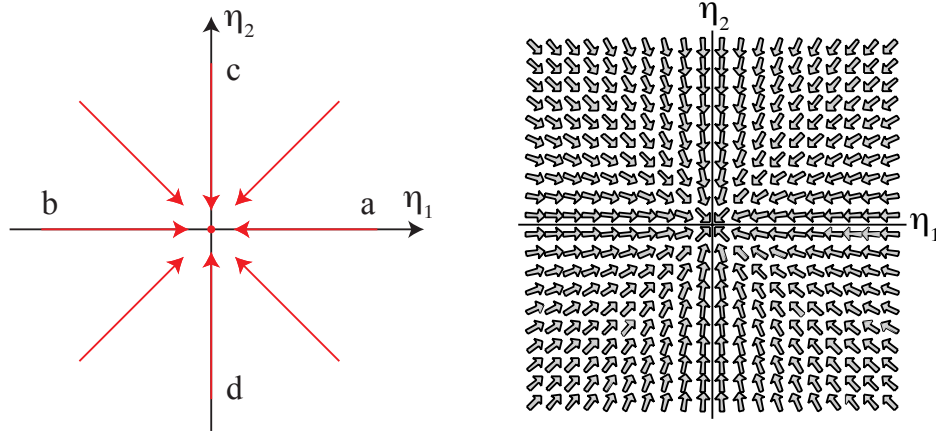
---

<sup>9</sup>A  $n$ -dimensional vector field is a map  $\mathbf{f} : \mathcal{R}^n \rightarrow \mathcal{R}^n$ . 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.

such a scheme: from any point in the state space, the flow is directly towards  $\boldsymbol{\eta} = \mathbf{0}$ . This qualitative structure can be realized by the vector field of the stable linear system:

$$\hat{\boldsymbol{p}} = -K\boldsymbol{\eta} \quad (4.22)$$

where  $K \in \mathcal{R}^{2 \times 2}$ ,  $K > 0$ , illustrated in Figure 13(b).



(a) Qualitative structure.

(b)  $-K\boldsymbol{\eta}$ ,  $K > 0$

Fig. 13. A candidate vector field that globally asymptotically stabilizes  $\boldsymbol{\eta} = \mathbf{0}$ .

With (4.22) the first two requirements (4.15) and (4.16) are satisfied. This can be plainly seen from a plot of the vector field  $-K\boldsymbol{\eta}$ , shown in Figure 13(b), since all vectors at every element in the state space point to the origin. However, notice that the vectors along the  $\eta_2$  axis point directly towards the origin—i.e., they have a non-zero  $\eta_2$  component. This shows—visually—that requirement (4.21) is not satisfied by our candidate  $\hat{\boldsymbol{p}}$ . Consequently, if we attempt to design a control law for  $P$  we will encounter the impossible task of reconciling the plant model (4.12), for which  $\dot{\eta}_2 = 0$  when  $\eta_1 = 0$ , with the reference vector field (4.22), for which  $\dot{\eta}_2 = 0$  only when  $\eta_2 = 0$ .

To understand the practical reason underlying the problem here, consider the behaviors that the vector field implies. Point  $a$  in Figure 13(a) corresponds to the agent perceiving the target in front of it; hence the flow from  $a$  to the origin indicates that the action implied by the vector field causes the agent to close in on the target by moving forward towards it, as shown in Figure 14(a). Similarly, the flow from point  $b$  (which corresponds to the target being behind the agent) to the origin closing in on the target while maintain it's back to the target, as shown in Figure 14(b). Now, recall that all points along the  $\eta_2$  axis (not including the origin) correspond to the target being to either the right or left of the agent. Hence, the flow from  $c$  or  $d$  to the origin corresponds to the agent maintaining its perpendicular orientation with respect to the target *while moving towards it* as illustrated in Figures 14(c) and 14(d). This is clearly incompatible with the differential drive vehicle kinematics discussed in the previous section—the vehicle simply can not slide sideways. Hence constraint (4.21) arises due to constraints imposed by our agent's embodiment.

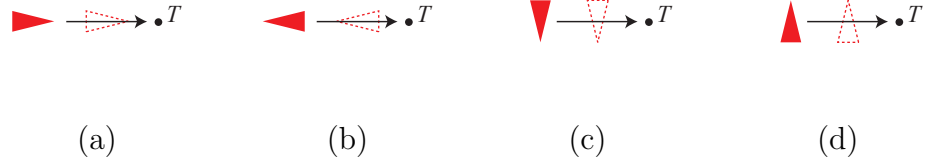


Fig. 14. Behaviors specified by the reference vector field of Figure 13.

## b. A better solution

Graphically speaking for any viable reference vector field,  $\hat{\mathbf{p}}$ , the requirements for taxis, (4.15) and (4.16), demand that all flows eventually lead to a single equilibrium at the origin. At the same time, the structure of our plant model (4.12) demands (via requirement (4.21)) that  $\hat{\mathbf{p}}$  have only *horizontal* vectors along the  $\eta_2$  axis. Figure 15(a) illustrates the qualitative structure of such a viable candidate.

The behavior this vector field implies is intuitively appealing. Some representative cases are shown in Figure 16. Trajectories flow to the  $\eta_1$  axis, indicating that the agent acts to bring the target in front of (Figure 16(c)) or behind (Figure 16(d)) the agent; once this is achieved, the agent then closes in on the target (Figure 16(a) and Figure 16(b), respectively).

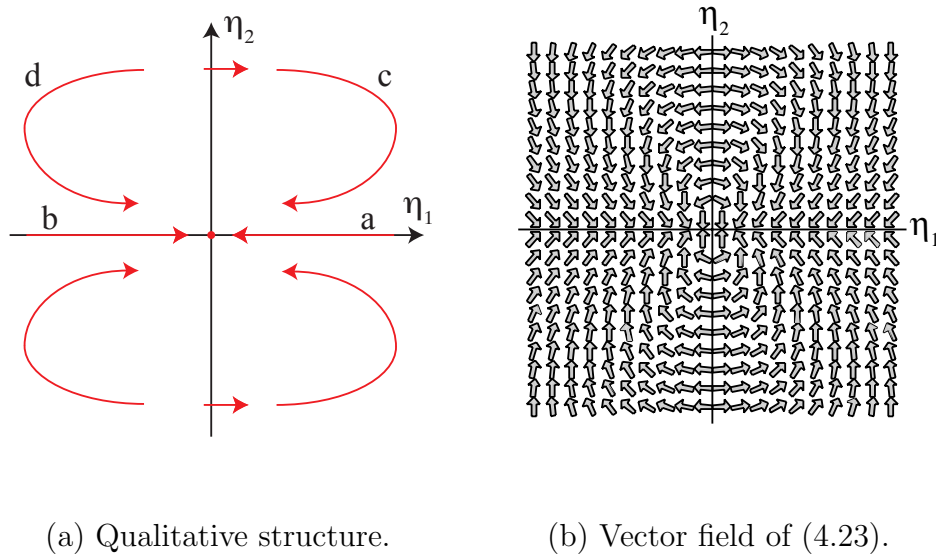


Fig. 15. Vector field for unconstrained taxis.

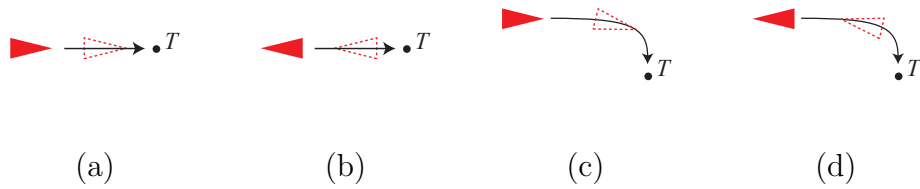


Fig. 16. Behavior specified by the reference vector field of Figure 15.

The qualitative structure of the flow of Figure 15(a) can be realized by the vector

field:

$$\widehat{\mathbf{p}} : \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \mapsto \begin{bmatrix} -\text{sgn}(\eta_1) + \text{sgn}^+(\eta_1)|\eta_2| \\ -\text{sgn}(\eta_2)|\eta_1| \end{bmatrix} \quad (4.23)$$

as illustrated in Figure 15(b).

Now, setting:

$$\mathbf{p}(\boldsymbol{\eta}, \mathbf{a}) = \widehat{\mathbf{p}}(\boldsymbol{\eta}) \quad (4.24)$$

we can easily (i.e., without inducing singularities) solve for  $\mathbf{a}$ :

$$\mathbf{a} = \begin{bmatrix} \text{sgn}(\eta_1) \\ \text{sgn}^+(\eta_1)\text{sgn}(\eta_2) \end{bmatrix} = \begin{bmatrix} \text{sgn}(s_1) \\ \text{sgn}^+(s_1)\text{sgn}(s_2) \end{bmatrix} \quad (4.25)$$

(recall  $\sigma_1$  and  $\sigma_2$  are measurement functions that preserve the signum of their arguments).

#### D. Synthesis of taxis behaviors

Thus far we have introduced the concept of vector field design as a promising methodology to develop static control laws for taxis behavior, applying it to the synthesis of the unconstrained taxis behavior of (4.25). The basic method is:

1. identify the qualitative structure of a vector field, compliant with (4.15), (4.16), and (4.21), that can be used to induce the agent to execute some desirable mode of behavior
2. arrive at a specific analytically-tractable expression for a vector field with this structure
3. set  $\widehat{\mathbf{p}}$  as the right-hand side of the plant model (4.12) and solve for the actuation law,  $\mathbf{a}$



In this section and the next, we develop several additional behaviors using vector field design. This will serve to demonstrate the intuition behind the tool, as well as produce a repertoire of reactive behaviors for use in chapters VI and VII. Before proceeding, however, in order to aid subsequent development, we make some remarks regarding how the structure of the vector field relates to agent behavior.

Figure 17 illustrates two key manifolds in the sensory state space:

$$\begin{aligned} L_- &= \{\boldsymbol{\eta} : \eta_1 < 0, \eta_2 = 0\} \\ L_+ &= \{\boldsymbol{\eta} : \eta_1 > 0, \eta_2 = 0\} \end{aligned} \quad (4.26)$$

and the actions that result when the sensor state is controlled to flow along these manifolds. When on  $L_-$  (i.e., when the target is behind the agent), if the sensor state flows towards the origin, then the agent is engaged in target tracking via reverse motion. For flows on  $L_-$  away from the origin, the agent is engaged in anti-taxis via forward motion. When on  $L_+$  (i.e., when the target is in front of the agent), if the sensor state flows towards the origin, then the agent is engaged in target tracking via forward motion. For flows on  $L_+$  away from the origin, the agent is engaged in anti-taxis via reversal.

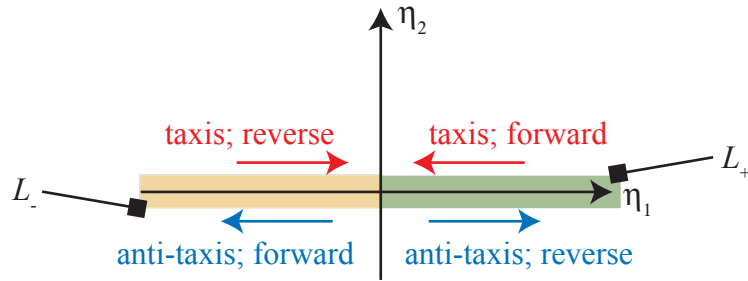


Fig. 17. Key manifolds and their behavioral implications.

With this, we can start our sketch of the visual characteristics of a candidate vector field by specifying how the agent should behave when it perceives the target

either in front of or behind the agent. But what about the rest of the sensory state space? With target tracking or evasion, at least, regulation is a matter of bringing states from any point in the state space to one of the key manifolds,  $L_+$  or  $L_-$ .<sup>10</sup> Now, how should these states flow to the manifolds? One way is by the use of curved flows, contrived to push states in a desirable manner. However, what is the relationship between the geometry of a flow and the resulting agent behavior? Figure 18 illustrates this relationship with respect to circular flows.

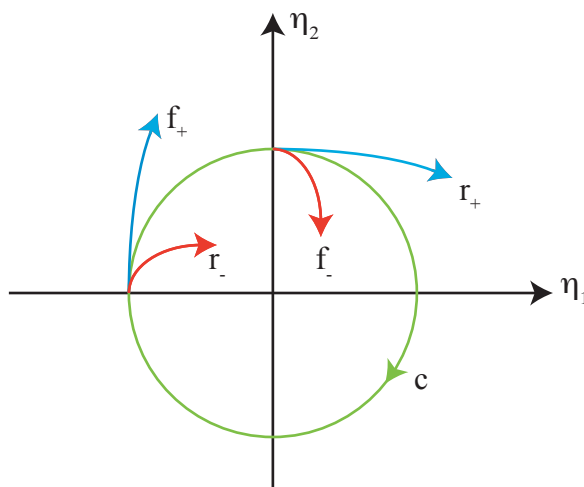


Fig. 18. The influence of flow curvature on behavior. Flow  $c$  is circular, while the flows denoted by the subscripts  $+$  and  $-$  denote flows whose radii are increasing or decreasing, respectively.

Consider the clockwise circular flow  $c$  which can be realized by the vector field:

$$\hat{\mathbf{p}} : \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \mapsto \begin{bmatrix} \eta_2 \\ -\eta_1 \end{bmatrix} \quad (4.27)$$

Setting (4.12) and (4.27) equal, we obtain  $a_\omega = 1$ , and  $a_v = 0$ ; hence, a circular flow corresponds to the agent rotating on the spot. Since  $f_+$  and  $r_+$  are expanding, they are crossing circles of increasing radius (which corresponds to distance from the target) as

<sup>10</sup>Recall, as per (4.21), we can not bring states to the origin via the  $\eta_2$  axis.

they flow, indicating anti-taxis. With  $f_+$ , since  $\eta_1 < 0$ , the agent is moving away from the target with its back to the target (and hence moving forwards). Whereas, with  $r_+$ , the agent is facing the target while moving away (and hence moving in reverse). Similar arguments demonstrate that  $f_-$  corresponds to taxis by forward motion, and  $r_-$  corresponds to taxis by reverse motion.

We have only presented a limited set of flow primitives to construct a vector field; however, as we will see in the remainder, these primitives can be “stitched” together in a variety of ways to realize a single composite vector field structure, enabling the specification of a remarkable diversity of behaviors.

## 1. Unconstrained, biased taxis

Suppose we wish to design a taxis behavior which, although unconstrained, is biased towards moving *forwards* toward the target (e.g., for agents which have the capability to reverse, but prefer — as most car drivers — forward motion where possible).

### a. Qualitative structure of $\hat{\mathbf{p}}$

Consider the vector field of Figure 19(a). The flow from  $a$  and  $b$  correspond to the agent closing in on the target by executing a straight line moving forwards (as in Figure 16(a)) or in reverse (as in Figure 16(b)). State trajectories from all other points (i.e., any state where  $\eta_2 \neq 0$ ) tend to flow towards  $L_+$  (i.e., where the target is ahead of the agent) and from there to the desired  $\boldsymbol{\eta} = \mathbf{0}$  state. Figure 20 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  $\frac{\pi}{2}$  (corresponding to a vector field trajectory hitting the  $\eta_2$  axis from the left, like for example, at point  $c_2$ ), moves to bring the target in front of the agent (corresponding to trajectories flowing towards the  $\eta_1$  axis), and then closes in on the target.

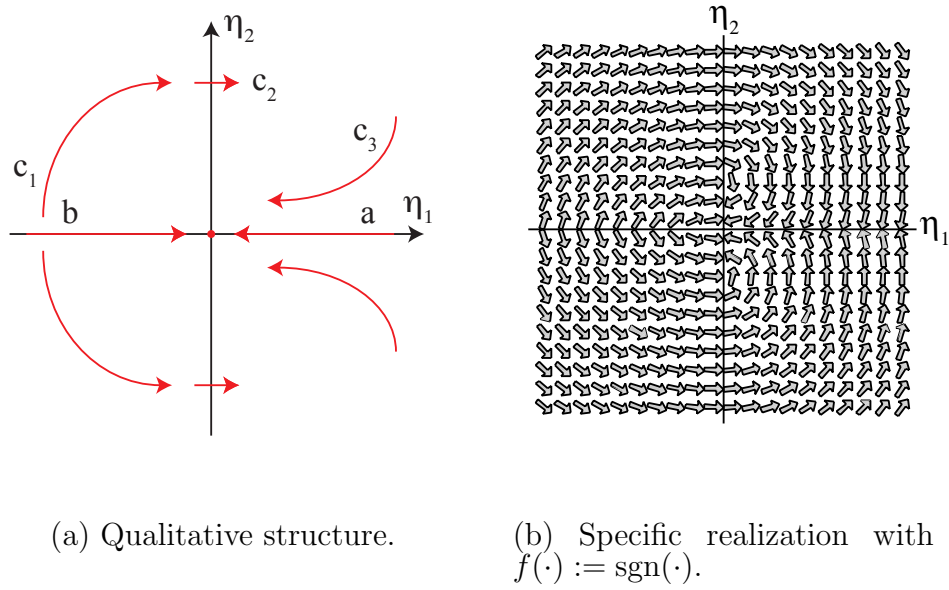


Fig. 19. Vector field for unconstrained taxis with forward bias.

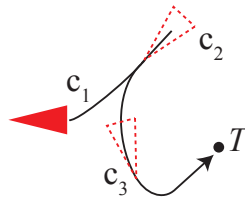


Fig. 20. Behavior ‘c’ specified by the reference vector field of Figure 19 that biases forward motion.

### b. Construction of an analytic form for $\hat{\mathbf{p}}$

We first divide the sensor state space into six subspaces:

$$\begin{aligned}
 X_+ &= \{\boldsymbol{\eta} : \eta_1 \leq 0, \eta_2 > 0\} & Y_+ &= \{\boldsymbol{\eta} : \eta_1 \geq 0, \eta_2 > 0\} \\
 L_- &= \{\boldsymbol{\eta} : \eta_1 < 0, \eta_2 = 0\} & L_+ &= \{\boldsymbol{\eta} : \eta_1 > 0, \eta_2 = 0\} \\
 X_- &= \{\boldsymbol{\eta} : \eta_1 \leq 0, \eta_2 < 0\} & Y_- &= \{\boldsymbol{\eta} : \eta_1 \geq 0, \eta_2 < 0\}
 \end{aligned} \tag{4.28}$$

and address the construction of the vector field within each.

Within  $X_+$ , rather than bring all flows towards  $L_-$  and from thence to the origin by reversing, we require clockwise rotation of the flow towards  $Y_+$  where taxis by forward motion is engaged; this represents a bias in favor of taxis by forward motion. Hence, the flow geometry must either be that of a circle (for zero translational motion) or a contracting spiral (for reverse taxis motion). Accordingly, let  $f(\eta_1)$  be a function that, for  $\eta_1 < 0$ , is non-positive. Then we can propose:

$$\hat{\mathbf{p}} = \begin{bmatrix} -f(\eta_1) + \eta_2 \\ -\eta_1 \end{bmatrix}, \boldsymbol{\eta} \in X_+ \cup Y_+ \tag{4.29}$$

for  $X_+$ . Within  $Y_+$ , we require clockwise rotation of the flow with taxis; hence, the flow geometry must be that of a contracting spiral. If we stipulate that for  $\eta_1 > 0$ ,  $f(\eta_1)$  must be positive, we can continue to use (4.29) within  $Y_+$ .

Within  $X_- \cup Y_-$  notice that the flow geometry has the same structure as for  $X_+ \cup Y_+$  but with a counter-clockwise sense of rotation. This suggests:

$$\hat{\mathbf{p}} = \begin{bmatrix} -f(\eta_1) - \eta_2 \\ +\eta_1 \end{bmatrix}, \boldsymbol{\eta} \in X_- \cup Y_- \tag{4.30}$$

Finally, for  $L_- \cup L_+$  we have:

$$\hat{\mathbf{p}} = \begin{bmatrix} -\eta_1 \\ 0 \end{bmatrix}, \boldsymbol{\eta} \in L_- \cup L_+ \quad (4.31)$$

Now, we must resolve (4.29)-(4.31) into a unified single expression. To do so, note that the function  $\text{sgn}(\eta_2)$  can be used to test for whether we are in:

- $X_+ \cup Y_+$  (for which  $\text{sgn}(\eta_2) = +1$ )
- $L_- \cup L_+$  (for which  $\text{sgn}(\eta_2) = 0$ )
- $X_- \cup Y_-$  (for which  $\text{sgn}(\eta_2) = -1$ )

This suggests the analytic form:

$$\hat{\mathbf{p}} : \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \mapsto \begin{bmatrix} -f(\eta_1) + \eta_2 \text{sgn}(\eta_2) \\ -\eta_1 \text{sgn}(\eta_2) \end{bmatrix} = \begin{bmatrix} -f(\eta_1) + |\eta_2| \\ -\eta_1 \text{sgn}(\eta_2) \end{bmatrix} \quad (4.32)$$

where  $f(\cdot)$  is of one of the forms shown in Figure 21.

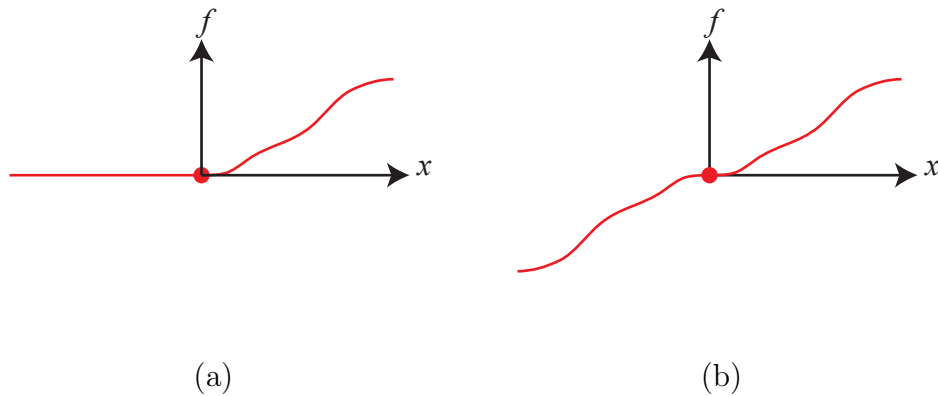


Fig. 21. Candidates for  $f(\cdot)$  in (4.32).

The next result confirms the stabilizing property of a family of vector fields that

includes (4.32).<sup>11</sup>

**Lemma 1.** *Consider the dynamical system  $\dot{\boldsymbol{\eta}} = \mathbf{q}(\boldsymbol{\eta})$  with right hand side:*

$$\mathbf{q} : \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \mapsto \begin{bmatrix} -f(\eta_1) + g(\eta_2) \\ -\eta_1 h(\eta_2) \end{bmatrix} \quad (4.33)$$

where:

- $\text{sgn}(f(x)) = \text{sgn}(x)$
- $g(x) = h(x) = 0 \iff x = 0$
- $xh(x) = g(x)$

The state  $\boldsymbol{\eta} = \mathbf{0}$  is globally asymptotically stable.

*Proof.* Let  $\boldsymbol{\eta} = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix}$  and  $\mathbf{q} = \begin{bmatrix} q_1 : \mathcal{R}^2 \rightarrow \mathcal{R} \\ q_2 : \mathcal{R}^2 \rightarrow \mathcal{R} \end{bmatrix}$ . Consider the continuously differentiable, radially unbounded, positive definite function:

$$V(\boldsymbol{\eta}) = \frac{1}{2} \boldsymbol{\eta}^T \boldsymbol{\eta} \quad (4.34)$$

with derivative:

$$\dot{V}(\boldsymbol{\eta}) = -\eta_1 f(\eta_1) \quad (4.35)$$

and the set:

$$\begin{aligned} S &= \{\boldsymbol{\eta} : \dot{V}(\boldsymbol{\eta}) = 0\} \\ &= \{\boldsymbol{\eta} : \eta_1 = 0\} \end{aligned} \quad (4.36)$$

Since:

- $(\forall \boldsymbol{\eta} \in S : \boldsymbol{\eta} \neq \mathbf{0}) (q_1(\boldsymbol{\eta}) \neq 0)$

---

<sup>11</sup>We will take  $f(\cdot)$  to be an odd function for clarity of the proof; however, we note that the proof can be extended for the case where  $f(\cdot)$  is 0 for negative arguments.

- $q(\boldsymbol{\eta}) = \mathbf{0} \iff \boldsymbol{\eta} = \mathbf{0}$

no solution can stay identically in  $S$  except for the solution  $\boldsymbol{\eta}(t) \equiv \mathbf{0}$ . Hence, by the theorem of Barbashin-Krasovskii-LaSalle,  $\boldsymbol{\eta} = \mathbf{0}$  is globally asymptotically stable.  $\square$

### c. Derivation of the actuation command

Setting (4.12) and (4.32) equal, we obtain:

$$\mathbf{a} = \begin{bmatrix} f(\eta_1) \\ \text{sgn}(\eta_2) \end{bmatrix} \quad (4.37)$$

which, since we only have access to the measured plant state, we modify to:

$$\mathbf{a} = \begin{bmatrix} f(\sigma_1(\eta_1)) \\ \text{sgn}(\sigma_2(\eta_2)) \end{bmatrix} = \begin{bmatrix} f(s_1) \\ \text{sgn}(s_2) \end{bmatrix} \quad (4.38)$$

where  $\sigma_1, \sigma_2$  are measurement functions. The next results confirms that this law indeed stabilizes  $P$ .

**Theorem 1.** *The plant model (4.12) under the feedback control of (4.38) has globally asymptotically stable equilibrium  $\boldsymbol{\eta} = \mathbf{0}$ .*

*Proof.* Since  $\text{sgn}(\sigma(x)) = \text{sgn}(x)$  for any measurement function  $\sigma$ , (4.38) and (4.37) are equivalent. Hence, setting  $h(x) = \text{sgn}(x)$  and  $g(x) = |x|$ , the result follows from Lemma 1.  $\square$

One of the simplest functions we can use for  $f(\cdot)$  is  $\text{sgn}(\cdot)$ ; Figure 19(b) illustrates the corresponding vector field. This leads to the computationally simple actuation:

$$\mathbf{a} = \begin{bmatrix} \text{sgn}(s_1) \\ \text{sgn}(s_2) \end{bmatrix} \quad (4.39)$$



## 2. Taxis with constrained translational motion

Constraints on the actions of an agent can be due to inherent limitations of the agent (e.g., the inability of the vehicle to move in reverse, damage to the vehicle preventing full use of actuators) or imposed by external phenomena (e.g., obstacles in the agent's path). Regardless, we desire the agent to accomplish its task (taxis) in the face of these constraints. In the following we present the synthesis of control laws that regulate perception, making due with constrained actuation.

### a. Qualitative structure of $\hat{p}$

From the behavioral modes of Figures 16 and 20, we can see that reverse motion is often used as part of the overall motion towards the target. Precluded from such reverse motion, however, an alternate regulation scheme is required. If the target is somewhere to the front of the agent (i.e.,  $\eta_1 > 0$ ) then clearly no reverse motion is necessary. On the other hand, if the target is behind (i.e.,  $\eta_1 < 0$ ) then the agent should strive to steer its perception of the target away from the agent's rear end and towards its front; once target is in front, the agent can close in. Figure 22 illustrates this desired behavior.

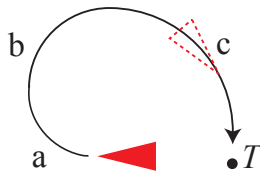
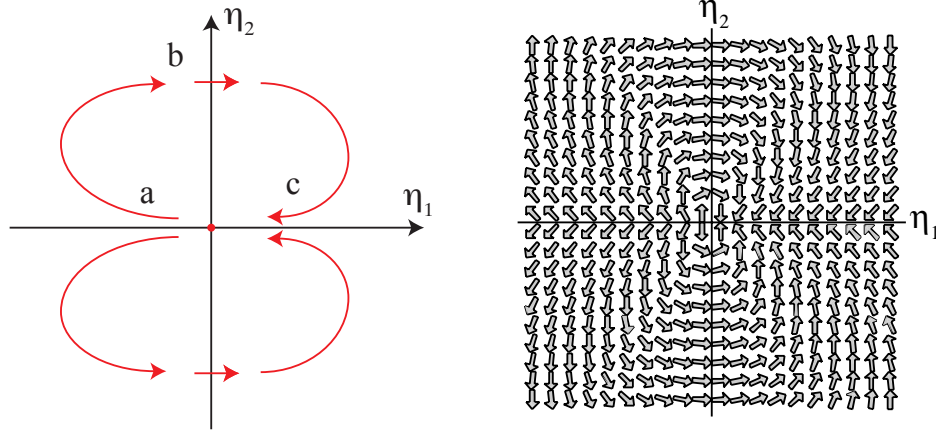


Fig. 22. Behavior of an agent engaging in taxis by forward-only motion.

A candidate vector field should have flows emanating from the  $L_-$  manifold (and away<sup>12</sup> from the origin), and entering the  $L_+$  manifold (and going towards the origin). Figure 23(a) presents the qualitative structure of such a vector field.



(a) Qualitative structure.

(b) Specific realization with  $f(\cdot) := |\cdot|$ .

Fig. 23. Vector field for taxis by forward motion.

### b. Construction of an analytic form for $\hat{\rho}$

Analogous to section b, we divide the sensor state space into four subspaces:

$$\begin{aligned} X_+ &= \{\boldsymbol{\eta} : \eta_1 \leq 0, \eta_2 \geq 0\} & Y_+ &= \{\boldsymbol{\eta} : \eta_1 \geq 0, \eta_2 \geq 0\} \\ X_- &= \{\boldsymbol{\eta} : \eta_1 \leq 0, \eta_2 < 0\} & Y_- &= \{\boldsymbol{\eta} : \eta_1 \geq 0, \eta_2 < 0\} \end{aligned} \quad (4.40)$$

and address the construction of the vector field within each.

Within  $X_+$  we require clockwise rotation of the flow with a constraint against reversing; hence, the flow geometry must either be that of a circle or an expanding

<sup>12</sup>Although the goal is taxis, if the target is behind the agent and moving towards the agent then reversal is occurring.

spiral (for forward motion). Let  $f(\eta_1)$  be a function that, for  $\eta_1 < 0$ , is either identically equal to 0 or a positive function (for an expanding spiral). Then we can propose:

$$\hat{\mathbf{p}} = \begin{bmatrix} -f(\eta_1) + \eta_2 \\ -\eta_1 \end{bmatrix}, \boldsymbol{\eta} \in X_+ \cup Y_+ \quad (4.41)$$

for  $X_+$ . Within  $Y_+$  we require clockwise rotation of the flow with taxis; hence, the flow geometry must be that of a contracting spiral. For  $\eta_1 > 0$ , let  $f(\eta_1)$  be an positive function; then we can use (4.41) within  $Y_+$ .

Within  $X_- \cup Y_-$  the flow geometry has the same structure as for  $X_+ \cup Y_+$  but with a counter-clockwise sense of rotation, suggesting:

$$\hat{\mathbf{p}} = \begin{bmatrix} -f(\eta_1) - \eta_2 \\ +\eta_1 \end{bmatrix}, \boldsymbol{\eta} \in X_- \cup Y_- \quad (4.42)$$

To resolve (4.41) and (4.42) into a unified single expression, we use  $\text{sgn}^+(\eta_2)$  to test whether we are in:

- $X_+ \cup Y_+$  (for which  $\text{sgn}^+(\eta_2) = +1$ )
- $X_- \cup Y_-$  (for which  $\text{sgn}^+(\eta_2) = -1$ )

leading to the analytic form:

$$\hat{\mathbf{p}} : \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \mapsto \begin{bmatrix} -f(\eta_1) + \eta_2 \text{sgn}^+(\eta_2) \\ -\eta_1 \text{sgn}(\eta_2) \end{bmatrix} = \begin{bmatrix} -f(\eta_1) + |\eta_2| \\ -\eta_1 \text{sgn}^+(\eta_2) \end{bmatrix} \quad (4.43)$$

where  $f(\cdot)$  is of one of the forms shown in Figure 24.

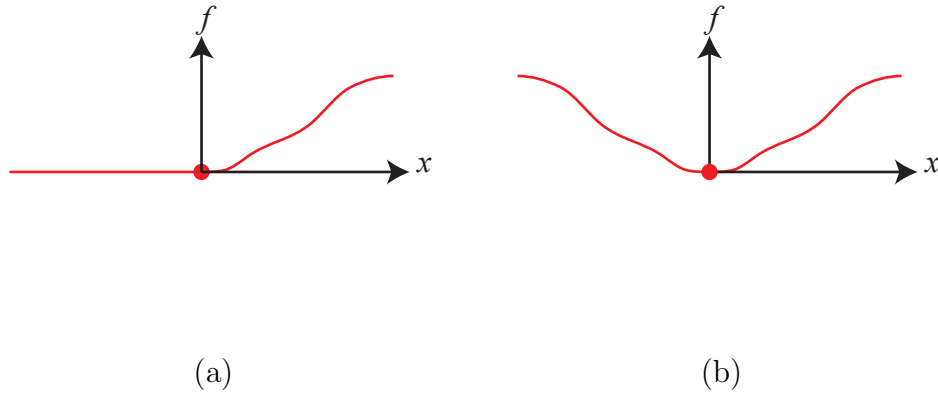


Fig. 24. Candidates for  $f(\cdot)$  in (4.43).

### c. Derivation of the actuation command

Setting (4.12) and (4.43) equal gives:

$$\mathbf{a} = \begin{bmatrix} f(\eta_1) \\ \text{sgn}^+(\eta_2) \end{bmatrix} \quad (4.44)$$

which, since we only have access to the measured plant state, we modify to:

$$\mathbf{a} = \begin{bmatrix} f(\sigma_1(\eta_1)) \\ \text{sgn}^+(\sigma_2(\eta_2)) \end{bmatrix} = \begin{bmatrix} f(s_1) \\ \text{sgn}^+(s_2) \end{bmatrix} \quad (4.45)$$

where  $\sigma_1, \sigma_2$  are measurement functions. A straightforward candidate for  $f(\cdot)$  is  $|\cdot|$ ; Figure 23(b) illustrates the corresponding vector field. This leads to the actuation:

$$\mathbf{a} = \begin{bmatrix} |s_1| \\ \text{sgn}^+(s_2) \end{bmatrix} \quad (4.46)$$

#### d. Taxis by reverse-only motion

For the complementary behavior of taxis via reverse-only motion (illustrated in Figure 25), by similar arguments to those in section a we obtain the vector field structure of Figure 26(a). The flow is identical to that of Figure 23(a) but for a change of sense in the rotation. Hence, negating vector field (4.43) we obtain:

$$\hat{\mathbf{p}} : \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \mapsto \begin{bmatrix} f(\eta_1) - \eta_2 \text{sgn}^+(\eta_2) \\ \eta_1 \text{sgn}(\eta_2) \end{bmatrix} = \begin{bmatrix} f(\eta_1) - |\eta_2| \\ \eta_1 \text{sgn}^+(\eta_2) \end{bmatrix} \quad (4.47)$$

(where, again,  $f(\cdot)$  is of one of the forms shown in Figure 24), with the corresponding actuation law:

$$\mathbf{a} = \begin{bmatrix} -f(\eta_1) \\ -\text{sgn}^+(\eta_2) \end{bmatrix} \quad (4.48)$$

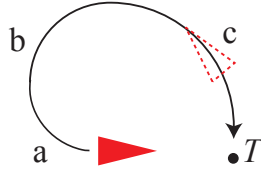
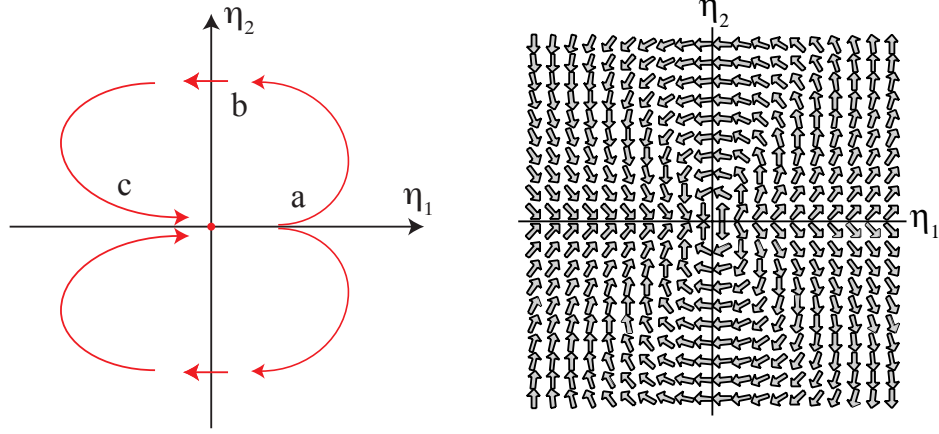


Fig. 25. Behavior of an agent engaging in taxis by reverse-only motion.

### 3. Taxis with constrained rotational motion

We can also consider the case where the agent is constrained to rotate in a specific direction as it translates. This may occur due to damage to the vehicle (e.g., loss of one motor of a differentially driven vehicle, or a servomotor in a steered one), introducing a rotational bias to any translational motion.

Another more interesting case is that of foiling certain radar detection systems. Suppose the agent is hostile to the target it is tracking, and suppose the target uses a



(a) Qualitative structure.

(b) Specific realization with  $f(\cdot) := |\cdot|$ .

Fig. 26. Vector field for taxis by reverse motion.

radar system to scan for whether the agent is moving towards the target. The agent will be most prone to detection when its radial component of velocity towards the target is higher. By executing a curved path towards the target, the radial component of velocity is transferred to a tangential one. If this is done sufficiently, it is possible for the radial component to become too small for a radar to detect (i.e., being less than the minimum detectable velocity of the radar [97, 98]) with the agent escaping detection.

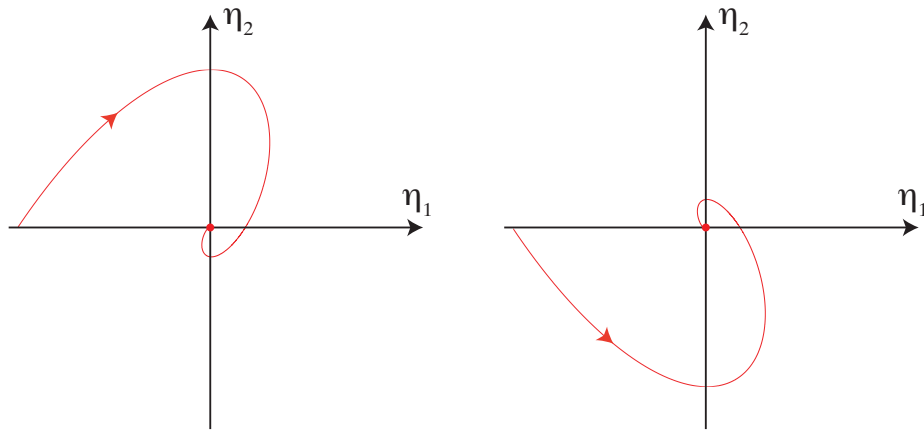
Figure 27 illustrates two vector fields that simultaneously induce rotational and translational motion for taxis. These vector fields have the qualitative structure of a stable focus, hence we can propose the following analytic form for the vector field:

$$\hat{\mathbf{p}} : \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \mapsto \begin{bmatrix} \pm\kappa\eta_2 - f(\eta_1) \\ \mp\kappa\eta_1 \end{bmatrix} \quad (4.49)$$

with the corresponding actuation law:

$$\mathbf{a} = \begin{bmatrix} f(\eta_1) \\ \pm\kappa \end{bmatrix} \quad (4.50)$$

where  $f(x)$  is an odd function of  $x$  such that  $\text{sgn}(f(x)) = \text{sgn}(x)$ , and  $\kappa > 0$ .



(a) Taxis with a rotational bias that induces positive (counter-clockwise) rotation in the agent.

(b) Taxis with a rotational bias that induces negative (clockwise) rotation in the agent.

Fig. 27. Vector field for taxis with a rotational bias.

## E. Synthesis of non-taxis behaviors

The control laws of the previous section bring the agent to the target; however, we can consider behaviors that do differently.

## 1. Anti-taxis

To realize anti-taxis, i.e., motion away from a target of interest, we must propose a vector field that takes  $\boldsymbol{\eta}$  as far away as possible from  $\mathbf{0}$ . We can consider two classes of such vector fields:

- ones that bring the flow in the anti-taxis sense asymptotically along the  $L_-$  and  $L_+$  manifolds of Figure 17 (call this asymptotic anti-taxis)
- ones that cause the flow to diverge in a radial unbounded manner via other regions of the sensory state space (call this rotational anti-taxis, since flow traversal of any manifold other than  $L_-$  and  $L_+$  will induce some rotation)

Figure 28(a) illustrates a vector field for asymptotic anti-taxis. Note, that this has the same structure as the vector field for biased unconstrained *taxis*, but with the direction of flow reversed. We propose the analytic form:

$$\widehat{\boldsymbol{p}} : \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \mapsto \begin{bmatrix} \text{sgn}(\eta_1) - |\eta_2| \\ \eta_1 \text{sgn}(\eta_2) \end{bmatrix} \quad (4.51)$$

(illustrated in Figure 28(b)). A vector field for rotational anti-taxis is shown in Figure 29(a) with the analytic form:

$$\widehat{\boldsymbol{p}} : \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \mapsto \begin{bmatrix} \text{sgn}(\eta_1) - |\eta_2| \\ |\eta_1| \text{sgn}(\eta_2) \end{bmatrix} \quad (4.52)$$

(illustrated in Figure 29(b)). As with all the vector fields in this chapter, the corresponding actuation law can be easily derived by setting the analytic form of the vector field equal to the right-hand side of (4.12).



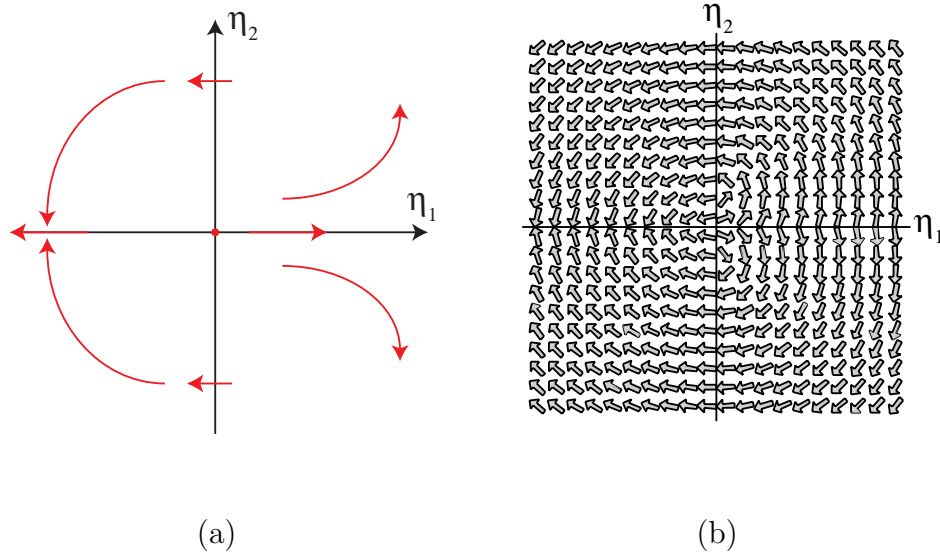


Fig. 28. Vector field for asymptotic anti-taxis.

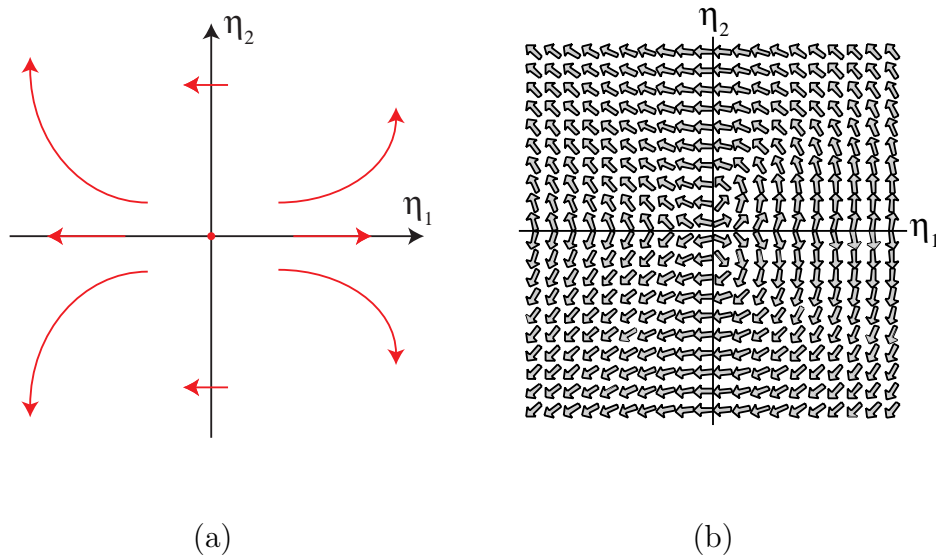


Fig. 29. Vector field for rotational anti-taxis.

## 2. Parking

Another useful behavior is that of “parking,” where the agent comes to a fixed distance from the target (the “parking spot”) oriented towards the target, and sits there. This can be realized by a vector field that stabilizes the non-zero equilibrium position corresponding to the parking spot, as shown in Figure 30(a). If  $\begin{bmatrix} \kappa \\ 0 \end{bmatrix}$  is the location of the parking spot, then this behavior can be realized by the vector field:

$$\hat{\mathbf{p}} : \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \mapsto \begin{bmatrix} \eta_1 - \kappa \\ -\eta_1 \text{sgn}(\eta_2) \end{bmatrix} \quad (4.53)$$

illustrated in Figure 30(b).

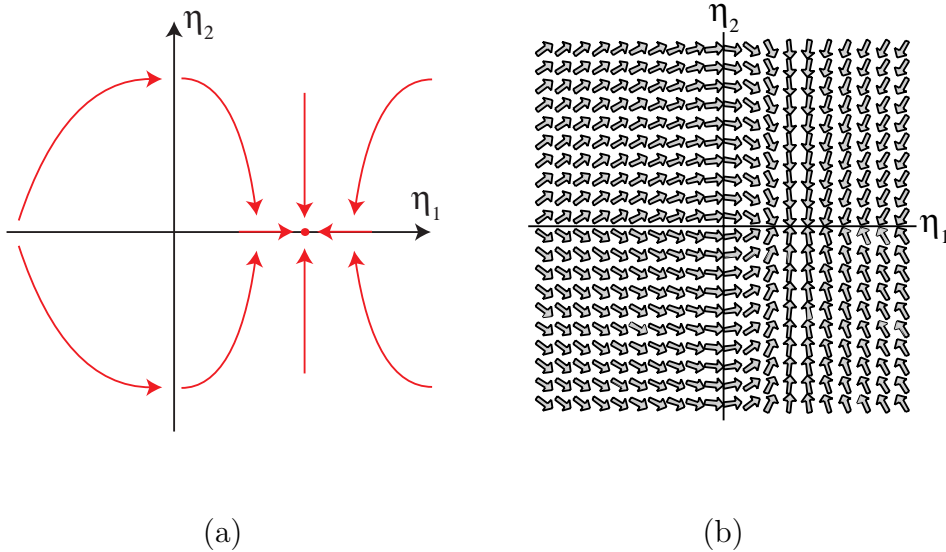


Fig. 30. Vector field for parking.

## F. Summary of vector fields

Tables II and III summarize the reference vector fields that we developed in this chapter, which in turn give rise to behaviors when the corresponding control law is derived<sup>13</sup> and used to specify velocity commands for the agent. These behavioral specifications give rise to purely reactive (static) laws requiring no memory and which are amenable to very economical implementation requiring simple nonlinear functions.

## G. Simulation results

In this section, we present simulations of the agent operating under the various control laws of this chapter. Figure 31 illustrates the setup of the agent and target for the simulations. In the figure (and for the subsequent simulation plots), the agent is indicated in red and the target in green. Let a global frame of reference be fixed to the target; the  $g_1 - g_2$  axes denote a coordinate system imposed on this frame of reference. Then  $\mathbf{g}_M(t)$  denotes the displacement from the target to the agent, referenced to the global frame of reference. Also, let  $\psi(t)$  be the angle between the agent's frame of reference and the global one.

In the following, we will provide the agent's initial conditions,  $\mathbf{g}_M(0)$  and  $\psi(0)$ , and present the trajectory that results from simulating the agent. We also illustrate the reference vector field,  $\hat{\mathbf{p}}$ , that underlies the control law for each simulation. The agent trajectories and the vector field are annotated so that the correspondence between agent behavior (as illustrated by its trajectory) and the vector field's structure can be understood.

The data for the simulations were obtained from a Simulink model simulated

---

<sup>13</sup>Setting  $\mathbf{p}(\boldsymbol{\eta}, \mathbf{a}(\boldsymbol{\eta})) = \hat{\mathbf{p}}(\boldsymbol{\eta})$  and solving for  $\mathbf{a}$ .

Table II. Summary of reference vector fields for taxis.

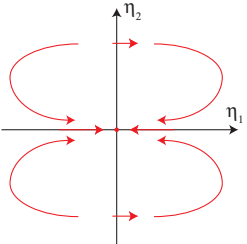
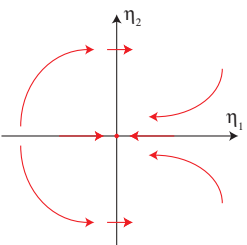
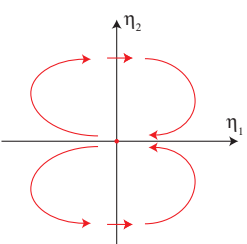
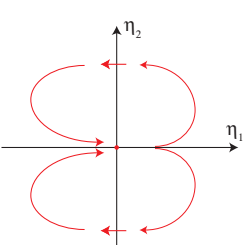
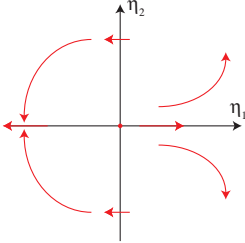
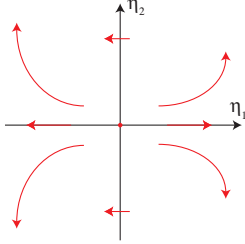
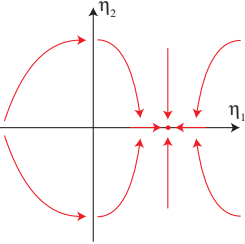
behavior	desired vector field $\hat{\mathbf{p}}(\boldsymbol{\eta})$	analytic form $\hat{\mathbf{p}}(\boldsymbol{\eta})$
unconstrained taxis		$\begin{bmatrix} -\eta_1 + \text{sgn}^+(\eta_1) \eta_2  \\ -\text{sgn}(\eta_2) \eta_1  \end{bmatrix}$
unconstrained taxis (forward bias)		$\begin{bmatrix} -\text{sgn}(\eta_1) +  \eta_2  \\ -\eta_1 \text{sgn}(\eta_2) \end{bmatrix}$
forward-only taxis		$\begin{bmatrix} - \eta_1  +  \eta_2  \\ -\eta_1 \text{sgn}(\eta_2) \end{bmatrix}$
reverse-only taxis		$\begin{bmatrix}  \eta_1  -  \eta_2  \\ \eta_1 \text{sgn}(\eta_2) \end{bmatrix}$

Table III. Summary of reference vector fields for non-taxis behaviors.

behavior	desired vector field $\hat{\mathbf{p}}(\boldsymbol{\eta})$	analytic form $\hat{\mathbf{p}}(\boldsymbol{\eta})$
anti-taxis (asymptotic)		$\begin{bmatrix} \text{sgn}(\eta_1) -  \eta_2  \\ \eta_1 \text{sgn}(\eta_2) \end{bmatrix}$
anti-taxis (rotational)		$\begin{bmatrix} \text{sgn}(\eta_1) -  \eta_2  \\  \eta_1  \text{sgn}(\eta_2) \end{bmatrix}$
parking $\boldsymbol{\eta} \rightarrow \begin{bmatrix} \kappa \\ 0 \end{bmatrix}$		$\begin{bmatrix} \eta_1 - \kappa \\ -\eta_1 \text{sgn}(\eta_2) \end{bmatrix}$

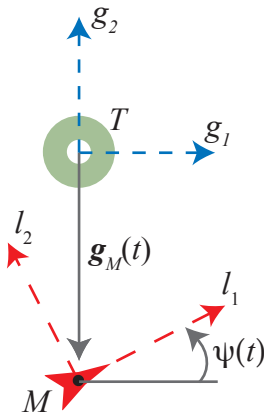


Fig. 31. Setup for simulations.

under Matlab. A custom animator (written in OpenGL by the author) was then used to visualize the data as an animation of an agent moving through the environment. The figures of the agent’s trajectories that we present in the following are time-lapsed images captured from the animator.

For example, Figure 32 shows four simulated trajectories for an agent operating under the unconstrained taxis law (4.25). In all four simulations, the agent (shown as a cone) is placed “due south” of the target (shown as a torus). For case (a), the agent is oriented such that the target is to its left, at an angle slightly ( $0.01$  radians) more than  $\frac{\pi}{2}$  radians, while for case (b) it is at an angle slightly less than  $\frac{\pi}{2}$  radians. For cases (c) and (d), the target is to the right of the agent, making angles with the target as in (a) and (b), respectively. As the figure illustrates, in all cases, the agent:

- starts at an initial orientation (indicated by annotations a1 though d1)
- translates (in reverse for cases (a) and (c), and forwards for cases (b) and (d)) *and* rotates to bring the agent in alignment with the target (indicated by annotations a2 through d2)
- translates (again, in reverse for cases (a) and (c), and forwards for cases (b)

and (d)) *without* rotating to close in on the target (indicated by annotations a3 through d3)

In a similar manner, Figures 33 to 38 present simulation results for the remaining behaviors developed in this chapter.

## H. Discussion

Structurally, the closest scheme to ours in the literature are the purely reactive Braitenberg vehicles [55]. Indeed, the work of this chapter can be viewed as a development of a series of Braitenberg-class machines using a more mathematically rigorous approach, while retaining the sort of intuitiveness that made Braitenberg’s work so influential. However, Braitenberg’s reactive schemes were not intended to serve in a robotic control architecture, but rather to help explain the coupling between sensation and actuation he observed in biological systems.

Matarić presents an outline of a design procedure<sup>14</sup> for behaviors in [100] which involves:

1. specifying the behavior’s qualitative characteristics in “observer space”
2. decomposing the behavior in terms of “observable, disjoint actions”
3. translating these disjoint actions into actuator inputs

This procedure is characteristic of a software-oriented approach, which differs from our cybernetic one in two key respects:

---

<sup>14</sup>Interestingly, the behavior-based paradigm is viewed by some [99] as being an example of a “dynamicist” (i.e., non-symbolicist) approach to cognition; however, a survey of the literature shows differently. In general, behavior-based robotics approaches tend to use either symbolic processing (i.e., software or finite-state automata), or, to a lesser extent, computational intelligence approaches (e.g., evolutionary computation, connectionism, etc.). In this work, we do not necessarily subscribe to the dynamicist position, however, since it is generally poorly-defined.

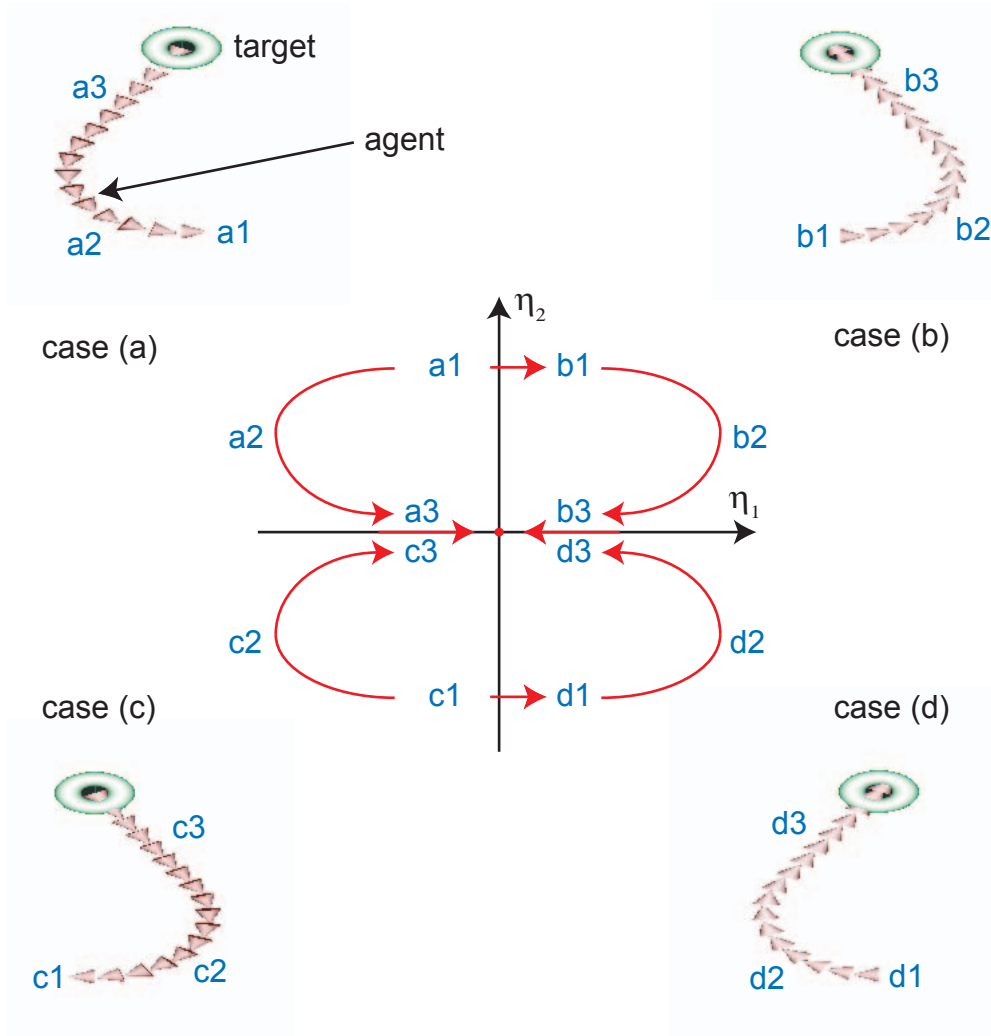


Fig. 32. Simulation results for the unconstrained taxis behavior showing the agent's trajectory for four cases. The reference vector field for unconstrained taxis is shown in the center with annotations that correspond to the simulated trajectories.



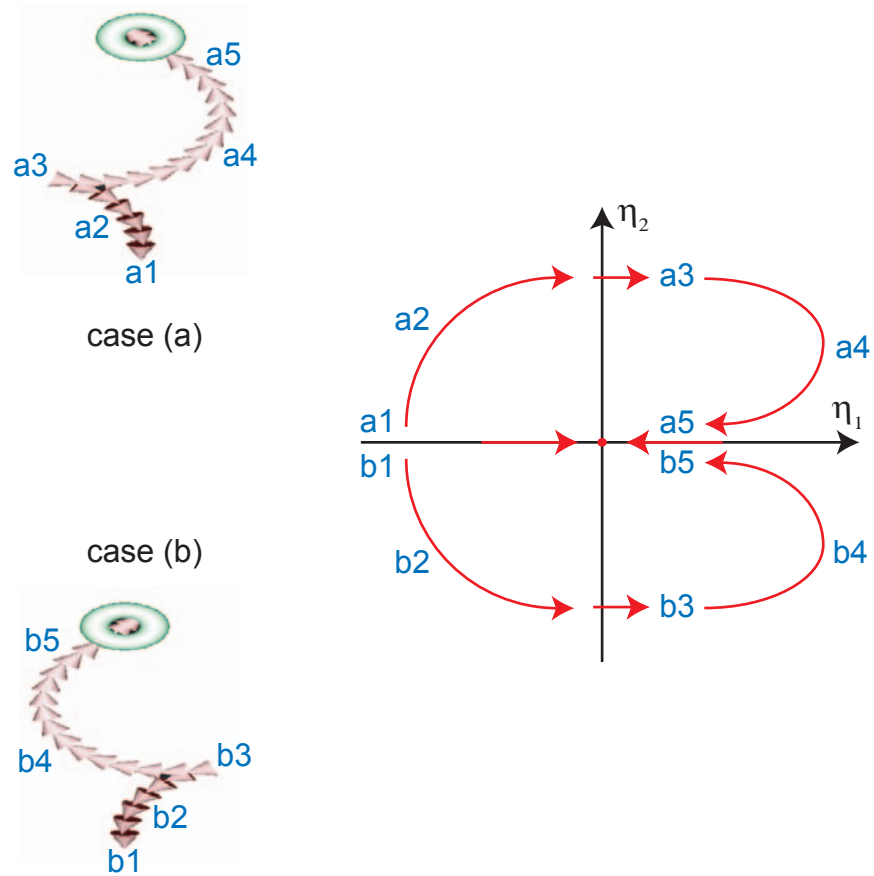


Fig. 33. Simulation results for the unconstrained taxis with forward bias behavior showing the agent's trajectory for two representative cases, and the associated reference vector field.

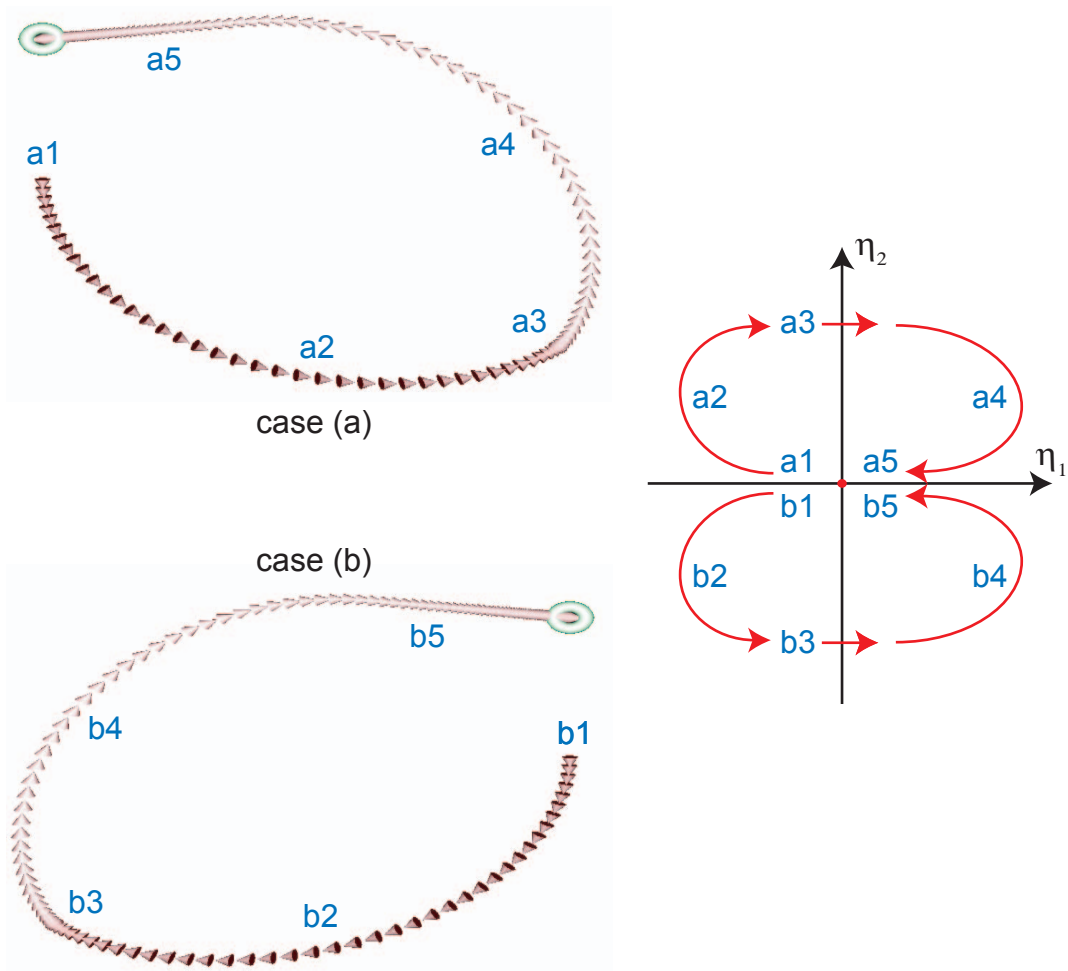


Fig. 34. Simulation results for the constrained taxis by forward motion behavior showing the agent's trajectory for two representative cases, and the associated reference vector field.

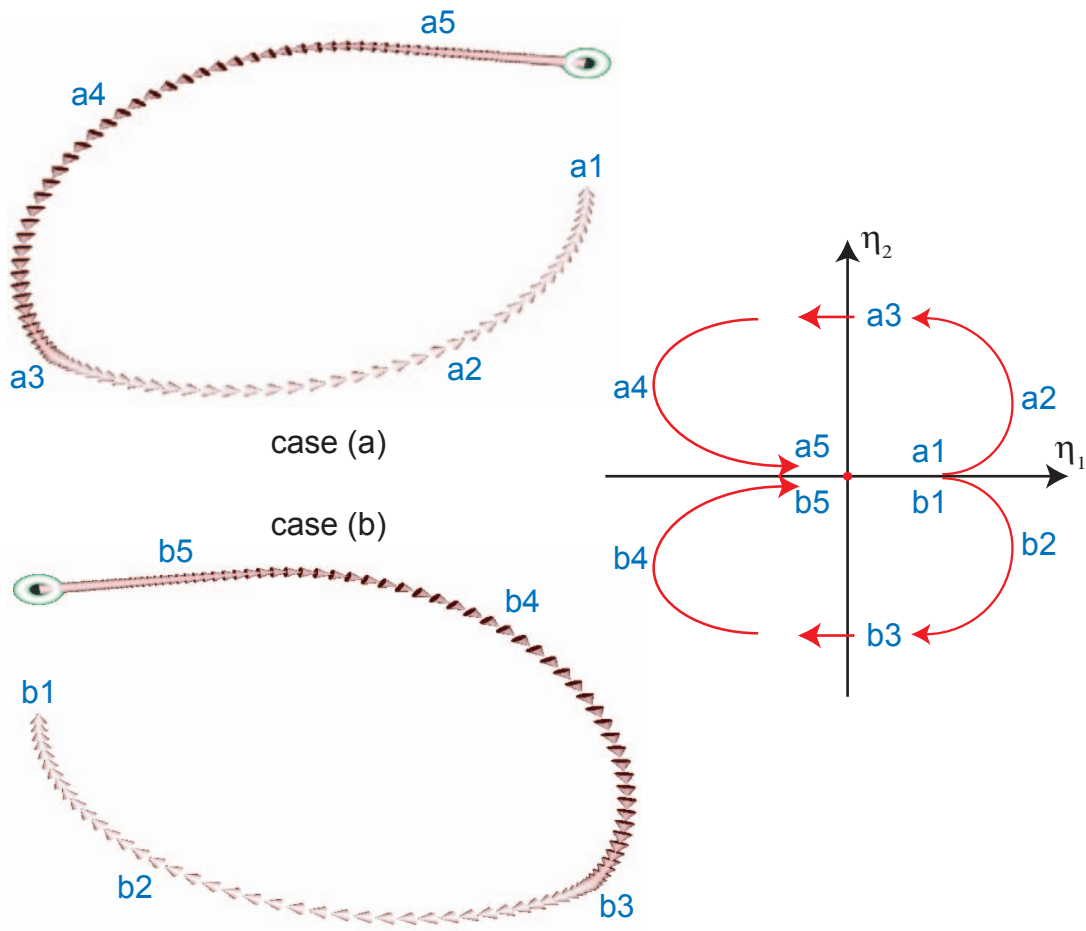


Fig. 35. Simulation results for the constrained taxis by reverse motion behavior showing the agent's trajectory for two representative cases, and the associated reference vector field.

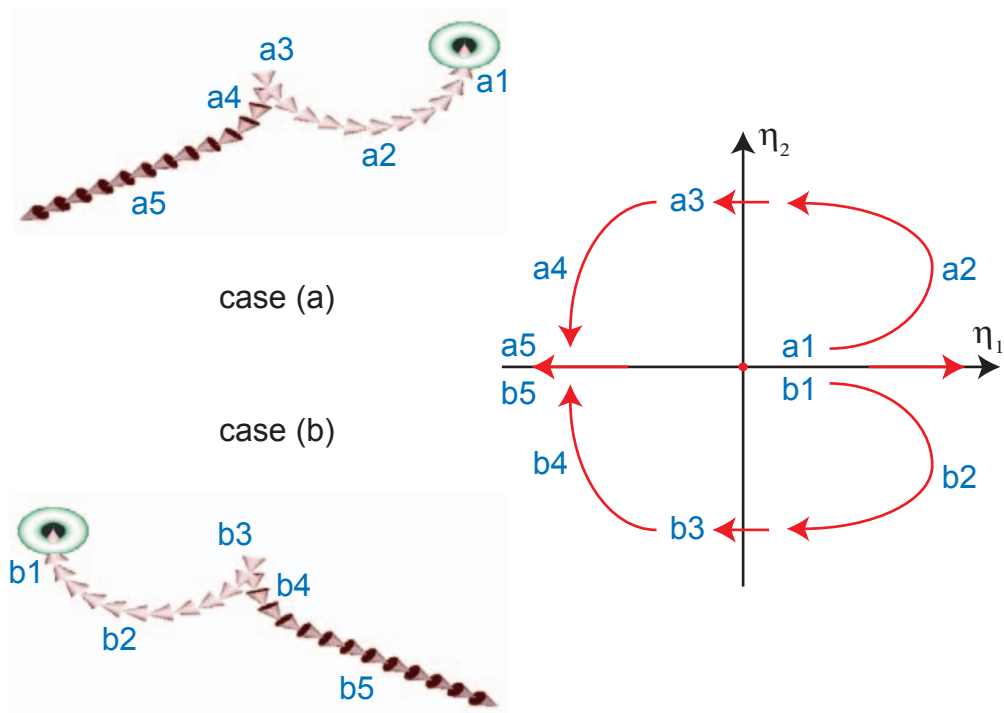


Fig. 36. Simulation results for the asymptotic anti-taxis behavior showing the agent's trajectory for two representative cases, and the associated reference vector field.

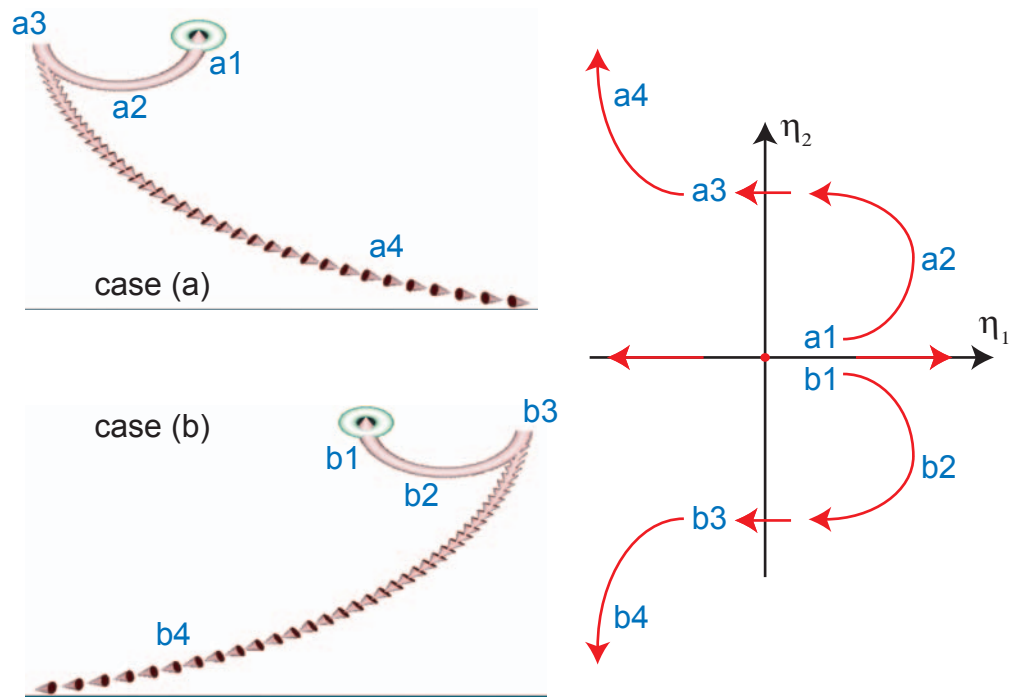


Fig. 37. Simulation results for the rotational anti-taxis behavior showing the agent's trajectory for two representative cases, and the associated reference vector field.

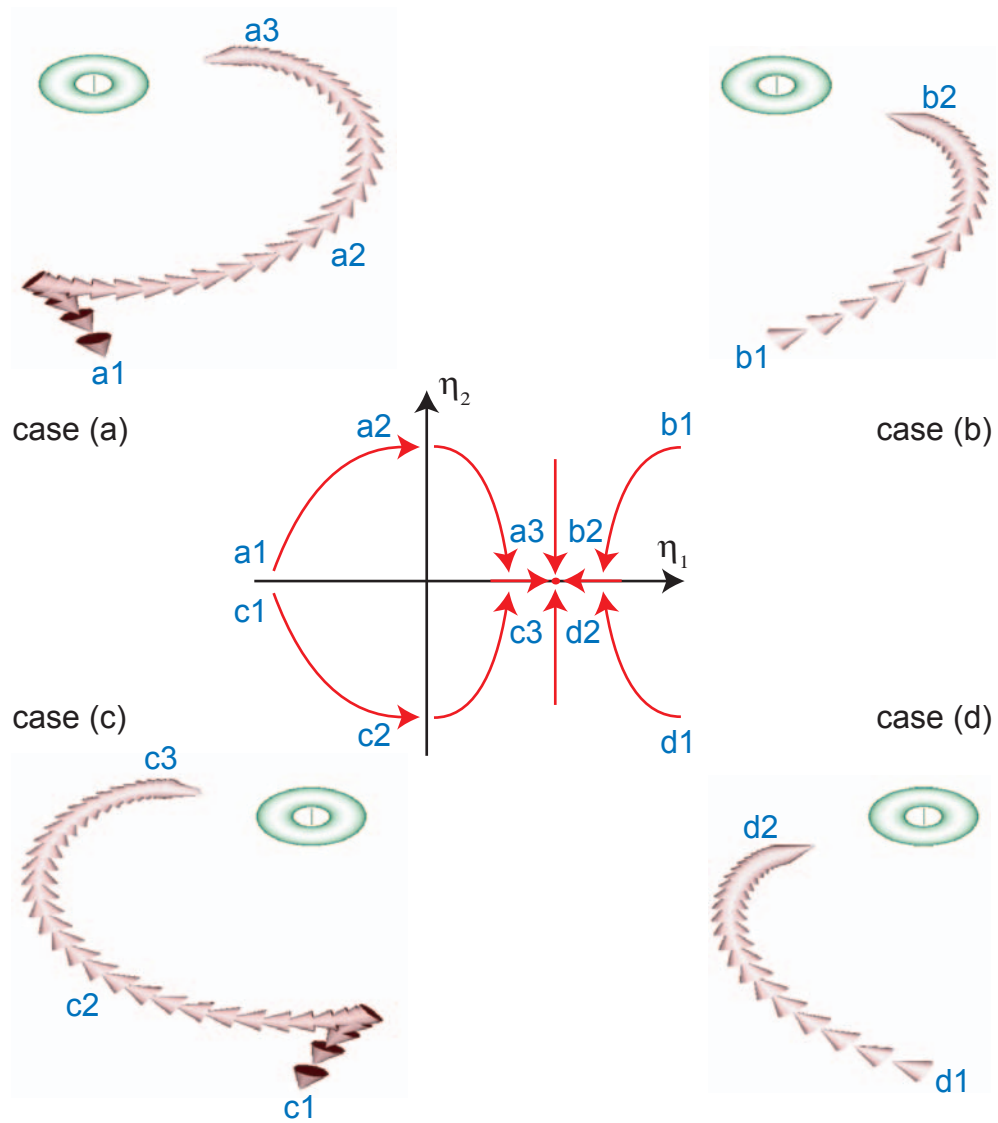


Fig. 38. Simulation results for the parking behavior showing the agent's trajectory for two representative cases, and the associated reference vector field.

- the description of behaviors from the vantage point of an external observer
- the explicit *scripting* of the *actions* that constitute the behavior

Our design approach, in comparison, views the design of behavior from an *agent-centric* point of view. From this perspective, the target sensor presents the agent with a signal (the agent’s “perception” of the target) and the behavior is specified (via a vector field) in terms of:

- where we want this signal to go (e.g., does it converge to an equilibrium point, does it go to infinity, etc.)
- the qualitative characteristics of how we want this signal to get there

Hence, in our methodology:

- at “design time” (i.e., when the vector field is constructed), we deal directly with the low-level nature of the agent’s *perception* (i.e., the sensor output signal) and the regulation of this perception—and *not* with abstractions concerning what the sensor output signal means to an external observer, nor with the nature of what the required actions of the agent are
- at “compile time,” we resolve the vector field with the plant model (that encapsulates the agent’s embodiment and situatedness) to obtain an actuation law that specifies how to couple sensory inputs to actuator outputs
- at “run time,” sensor outputs flow to the actuator inputs in accordance with this coupling

Hence, we do not design behaviors by scripting actions. Rather, we design a regulator to steer perception; the behavior—the sequence of goal-directed actions executed by the agent—emerges from this regulation of perception.

## CHAPTER V

### DYNAMIC CONTROL SCHEMES FOR NAVIGATION

#### A. Introduction

The reactive controllers we presented in chapter IV were purely reactive schemes for taxis that tightly coupled sensing and action via a memoryless controller. In this chapter we address two innovations [101, 102, 103] for an alternative dynamical scheme for navigation:

- the introduction of dynamics into the controller, “loosening” (in a sense that we will describe later) the coupling between sensing and action
- the use of a revised controller topology that enables a unified treatment of the two basic competencies for navigation—target tracking and obstacle avoidance

#### B. Preliminaries

##### 1. Loosened coupling between sensing and action

In the behavior-based robotics literature (e.g., [47]), a basic division is often made between *reactive* and *deliberative* controllers. The distinction is imprecise and often specified in a very qualitative manner, with the difference being one mainly of magnitude. That is, reactive controllers consist, essentially, of a *more* tighter coupling between sensors and actuators, either via a static function (e.g., the schemes we presented in chapter IV, Walter’s “turtles,” or Braitenberg’s vehicles) or small finite state machines.<sup>1</sup> In contrast, deliberative controllers employ state-based processing

---

<sup>1</sup>That schemes utilizing sets of small finite state machines, such as [58], are often classified as being “reactive” highlights the unsatisfying nature of the distinction since finite state machines are dynamical symbolic processors.



of sensory data (e.g., via larger state machines, symbolically processing abstracted representations of sensor data) to derive actuation commands.

Looking at the distinction from a structural point of view, a deliberative controller “loosens” the feed-forward coupling from sensation to actuation of a reactive system by interposing a dynamical system between sensation and actuation—actuation is buffered from sensation by this dynamical system.

Consider the situation shown in Figure 39, where:

$$P : \begin{cases} \dot{\boldsymbol{\eta}} = \mathbf{p}(\boldsymbol{\eta}, \mathbf{a}) \\ \mathbf{s} = \mathbf{q}(\boldsymbol{\eta}) \end{cases} \quad (5.1)$$

$$D : \dot{\boldsymbol{\zeta}} = \mathbf{f}(\boldsymbol{\zeta}, \mathbf{s}) \quad (5.2)$$

$$R : \mathbf{a} = \mathbf{g}(\boldsymbol{\zeta}, \mathbf{s}) \quad (5.3)$$

In order to design such a dynamical scheme for a cybernetic agent, an important question to ask is: why place a dynamical system between sensing and action? To address this we can consider conventional dynamical control architectures (such as in so-called deliberative control), or the piggy-backing of a deliberative controller on top of a behavior-based one (“hybrid” control [61]). In these schemes, the deliberative dynamics (in the form of a large state machine, possibly implemented via software and a general-purpose computer, executing planning algorithms) often provide the agent with faculties to make *longer-term* plans of action, rather than short-term reactions to stimuli.

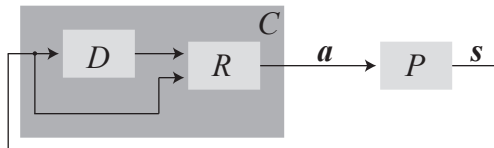


Fig. 39. The structure of a dynamical controller.

Following this motivation of achieving longer-term planning through the addition of a more complex state-based process (a dynamical system), consider the use of the linear time-invariant (LTI) filter,  $D$ :

$$D : \dot{\boldsymbol{\zeta}} = \mathbf{f}(\boldsymbol{\zeta}, \mathbf{s}) := A\boldsymbol{\zeta} + B\mathbf{s} \quad (5.4)$$

where  $\boldsymbol{\zeta} \in \mathcal{R}^{n_d}$  and  $\mathbf{s} \in \mathcal{R}^{n_e}$ . Conceptually, if  $D$  was a low-pass filter, the controller would be reactive to low-frequency longer-term trends in  $\mathbf{s}$ , acting to filter out the effect of spurious changes in  $\mathbf{s}$ —changes due, say, to the agent making a series of non-optimal actions (e.g., such as what would arise if it had to try out various options).

## 2. Unifying taxis and obstacle avoidance

The static schemes of chapter IV were solely concerned with target tracking; obstacle avoidance was only discussed as a motivation for the constrained taxis controllers. However, since autonomous navigation involves both target tracking and obstacle avoidance, it is natural to ask whether these two skills could be dealt with in a unified manner by the control scheme.

### a. Obstacle sensors

One way to address obstacle avoidance would be to consider (in a manner analogous to our development of the taxis plant model in chapter IV) an obstacle sensor that returns obstacle position information, and develop a plant model for how this information evolves in time as a function of the agent's actuation. Combined with the plant model for taxis, we could derive a composite plant model of the form:

$$\begin{aligned} \dot{\boldsymbol{\eta}}_T &= \mathbf{p}(\boldsymbol{\eta}_T, \mathbf{a}) \\ \dot{\boldsymbol{\eta}}_\Omega &= \mathbf{p}(\boldsymbol{\eta}_\Omega, \mathbf{a}) \end{aligned} \quad (5.5)$$

where  $\boldsymbol{\eta}_T$  and  $\boldsymbol{\eta}_\Omega$  correspond to the relative positions of the target and the obstacle under consideration, respectively. Based on this model, we could then formulate and attempt to solve the multivariable control problem of regulating  $\boldsymbol{\eta}_T$  and  $\boldsymbol{\eta}_\Omega$  via  $\mathbf{a}$ .

Based on the types of obstacle sensors commercially available that return scalar measurements of obstacle distance<sup>2</sup> within a detection sector about the agent, this may not be practical. More advanced sensors, involving machine vision for example, may make the problem of determining vector displacements more tractable, but this takes us away from the scope of our work—cybernetic formulations of lightweight *cognition*—and towards the design of advanced schemes for *perception*.<sup>3</sup>

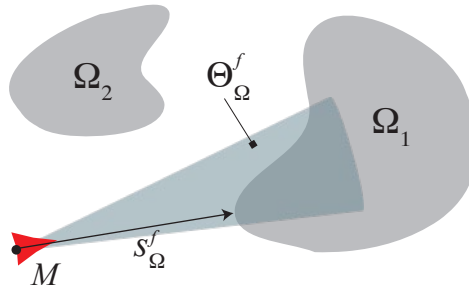


Fig. 40. Specification of an obstacle sensor.

Now, let's consider an alternative means to deal with scalar obstacle information. Figure 40 shows an agent,  $M$ , with a short range sensor (with sensing range  $r_\Omega^{\max}$ ) at its front that points along the  $l_1$  axis of the agent's local frame of reference. Let the set  $\Theta_\Omega^f$  be a sector emanating from the agent's position that contains the positive  $l_1$

<sup>2</sup>These transducers work by radiating a pulse of energy (e.g., ultrasound or infrared light), and then measuring the time it takes for the reflection of the pulse to return to a detector placed next to the source of the radiation. The distance to the object that caused the reflection is proportional to the pulse's transit time. Since source and detector are often placed close together, this restricts obstacle *distance* information to a relatively narrow sector ahead of the transducer.

<sup>3</sup>In addition to the increased complexity introduced by vision systems.

axis. Defining the set:

$$W_f := \Theta_\Omega^f \cap \left( \bigcup_{\forall i} \Omega_i \right) \quad (5.6)$$

we can specify the distance to the obstacle,  $r_\Omega$ , as:

$$r_\Omega = \min_{\forall \mathbf{w} \in W_f} \|\mathbf{w}\|_2 \quad (5.7)$$

Since the information returned by the sensor is unlikely to be *exactly* equal to the physical distance to an obstacle (due to nonlinear distortion, noise, etc.), we consider an obstacle sensor measurement function,  $s_\Omega(\cdot)$ , with the properties illustrated in Figure 41.

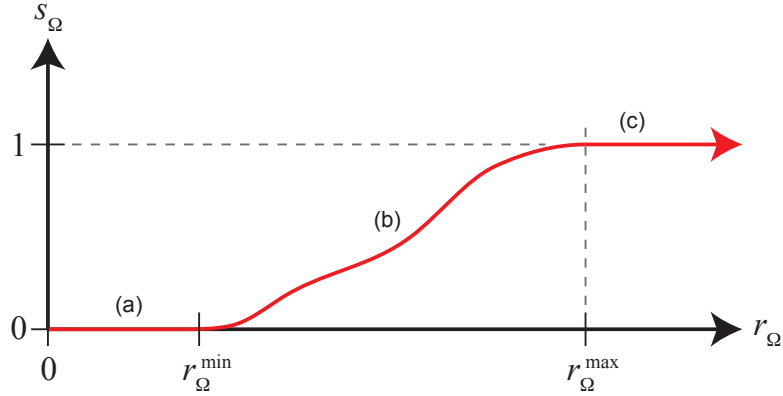


Fig. 41. The function  $s_\Omega : r_\Omega \mapsto [0, 1]$  is: (a) 0 for  $r_\Omega \leq r_\Omega^{\min}$ , (b) monotonically increasing for  $r_\Omega^{\min} \leq r_\Omega \leq r_\Omega^{\max}$ , and (c) 1 for  $r_\Omega \geq r_\Omega^{\max}$ .

## b. Disturbances

Based on the obstacle sensor specified in the preceding section, we can consider a reasonable safety mechanism for a robotic agent that attenuates translational motion when an obstacle is directly ahead. Given the function described in Figure 41, this attenuation can be accomplished by multiplying the agent's translational speed

command,  $a_v$ , with  $s_\Omega(r_\Omega)$ .

Now, the question arises: what is effect of this distortion of the agent's actuation command? To address this, we appeal to the control-theoretic concept of a *disturbance* which are:

“...by definition plant inputs ...which cannot be manipulated by the designer and are not completely known beforehand.” [104]

Hence, the presence of obstacles in the environment that can only be detected locally can be viewed as introducing a disturbance in the agent's actuation to the plant.

In [104], three basic attitudes toward dealing with disturbances in controller design are outlined:

- disturbances are undesirable and so:
  - they must be wholly *rejected* by cancellation
  - if disturbances can not be canceled, they must be *suppressed* and quashed to the greatest degree possible
- disturbances can be useful and, if so, should be *exploited*

Now, due to the assumptions we made in chapter III in which we precluded a priori global knowledge of the environment, totally rejection is not always possible. Rather, in this chapter we consider a means of suppressing disturbances. (A scheme that exploits the information implied by disturbances will be considered in chapter VI.)

In the next section we present the derivation of a controller that concurrently addresses the dual goals of navigation by tracking target position information and suppressing obstacle information. To that end, we alter our dynamical regulator motif (Figure 39) to that of Figure 42. Of note are the two channels into which information from the environment enter the controller:

- the *track* channel, corresponding to target position information which the controller attempts to regulate to  $\mathbf{0}$
- the *suppression* channel, corresponding to the disturbing influence of obstacle information on the agents actuation to the plant

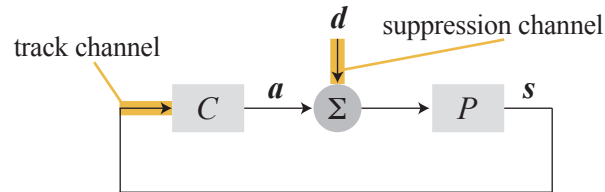


Fig. 42. Revised controller motif with track and suppression channels. The designer must design  $C$  so that the tracking objective (i.e., imposing desirable characteristics on the agent’s perception of the world,  $\mathbf{s}$ ) can be accomplished in the face of disturbances,  $\mathbf{d}$ , perturbing  $C$ ’s action,  $\mathbf{a}$ , upon  $P$ .

### C. Controller synthesis

The formulation of Figure 42 presents the autonomous navigation problem as a multi-variable feedback control problem in which the agent seeks to regulate its *filtered* perception of the world in the face of disturbances due to the presence of obstacles. We now provide the derivation of such a dynamical controller, using the nonlinear control-theoretic toolsets of Lyapunov synthesis [96] and backstepping [105].

#### 1. Derivation of virtual control, $\mathbf{s}^*$

We want to design a controller,  $R$ , to bring the states of  $D$ ,  $\zeta$ , to  $\mathbf{0}$ . From Figure 39, we observe that that  $R$  can only actuate change, via  $\mathbf{a}$ , to  $D$  indirectly through  $P$ , since  $\mathbf{s}$  is a static function of  $\boldsymbol{\eta}$  and not  $\mathbf{a}$ . Let’s suppose (a la backstepping) that  $R$  does have control over  $\mathbf{s}$  and define  $\mathbf{s}^*$  as what  $R$  would set  $\mathbf{s}$  to if it could—i.e., a virtual control.

Define the Lyapunov function candidate:

$$V_a := \zeta^T P \zeta \quad (5.8)$$

where  $P^T = P > 0$  is a positive definite matrix (the superscripted  $T$  denotes the transpose). Hence,

$$\dot{V}_a = \dot{\zeta}^T P \zeta + \zeta^T P \dot{\zeta} \quad (5.9)$$

into which (5.4) can be substituted yielding:

$$\dot{V}_a = [A\zeta + Bs]^T P \zeta + \zeta^T P [A\zeta + Bs] \quad (5.10)$$

Let  $K_0$  be a matrix that makes  $\bar{A} := A + BK_0$  Hurwitz, and define:

$$\mathbf{s}^{**} = \frac{\kappa_1}{2} \mathcal{S}(B^T P \zeta) \quad (5.11)$$

where:

$$\mathcal{S} : \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} \text{sat}_{l_1}(x_1) \\ \text{sat}_{l_1}(x_2) \end{bmatrix} \quad (5.12)$$

and, as illustrated in Figure 43:

$$\text{sat}_{l_1}(x) := \begin{cases} -1, & x \leq -l_1 \\ \frac{1}{l_1}x, & |x| < l_1 \\ +1, & x \geq +l_1 \end{cases} \quad (5.13)$$

Then setting:

$$\mathbf{s} = \mathbf{s}^* = K_0 \zeta + \mathbf{s}^{**} \quad (5.14)$$

and substituting this into (5.10) yields:

$$\dot{V}_a = \zeta^T \bar{A} P \zeta + \zeta^T P \bar{A} \zeta - \kappa_1 \mathcal{S}^T(B^T P \zeta) B^T P \zeta \quad (5.15)$$

Since  $\bar{A}$  is Hurwitz,  $\exists Q > 0$  such that

$$\zeta^T \bar{A} P \zeta + \zeta^T P \bar{A} \zeta = -\zeta^T Q \zeta \quad (5.16)$$

Substituting this into (5.15) shows that  $\dot{V}_a$  is negative definite, making  $\zeta = \mathbf{0}$  an asymptotically stable equilibrium point, as desired.

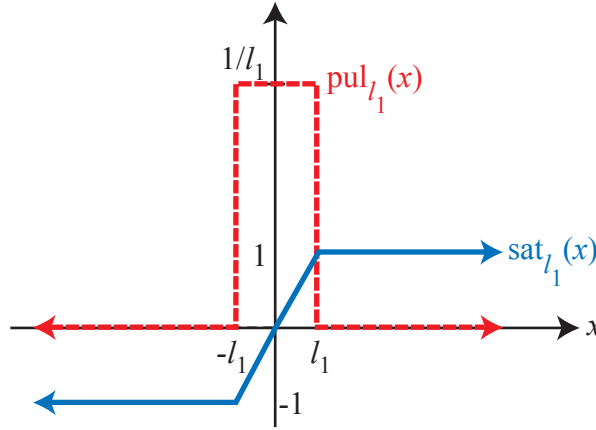


Fig. 43. Definitions of the  $\text{pul} : \mathcal{R} \mapsto \mathcal{R}$  and  $\text{sat} : \mathcal{R} \mapsto \mathcal{R}$  functions, parameterized by the constant  $l_1 > 0$ .

## 2. Backstepping setup

As noted earlier, we have only indirect control over  $\mathbf{s}$ —change actuated via  $\mathbf{a}$  must be integrated in  $P$  to influence  $\boldsymbol{\eta}$ , which in turn influences  $\mathbf{s}$ . The method of integrator backstepping [105] addresses the problem of designing  $\mathbf{a}$  for this case.

Let's first define an error signal that we will attempt to control to  $\mathbf{0}$ :

$$\boldsymbol{\epsilon} := \mathbf{s} - \mathbf{s}^* \quad (5.17)$$

and compute:

$$\begin{aligned} \dot{\boldsymbol{\epsilon}} &= \dot{\mathbf{s}} - \dot{\mathbf{s}}^* \\ &= \dot{\mathbf{s}} - K_0 \boldsymbol{\eta} - \frac{\kappa+1}{2} \Gamma (B^T P \zeta) B^T P \dot{\zeta} \end{aligned} \quad (5.18)$$



where:

$$\Gamma : \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} \text{pul}_{l_1}(x_1) & 0 \\ 0 & \text{pul}_{l_1}(x_2) \end{bmatrix} \quad (5.19)$$

and, as illustrated in Figure 43:

$$\text{pul}_{l_1}(x) := \begin{cases} 0, & x \leq -l_1 \\ \frac{1}{l_1}, & |x| < l_1 \\ 0, & x \geq +l_1 \end{cases} \quad (5.20)$$

Expanding and simplifying we obtain:

$$\dot{\boldsymbol{\epsilon}} = \dot{\boldsymbol{s}} - \mathbf{b}(\boldsymbol{\zeta}, \mathbf{s}) \quad (5.21)$$

where:

$$\mathbf{b}(\boldsymbol{\zeta}, \mathbf{s}) = [K_0 + \frac{\kappa_1}{2}\Gamma(B^T P \boldsymbol{\zeta})B^T P][A\boldsymbol{\zeta} + B\mathbf{s}] \quad (5.22)$$

### 3. Stabilization

Now we address the simultaneous stabilization of the states  $\boldsymbol{\zeta} = \mathbf{0}$  and  $\boldsymbol{\epsilon} = \mathbf{0}$ . The idea here is that if  $\boldsymbol{\epsilon} \rightarrow \mathbf{0}$  then  $\mathbf{s} \rightarrow \mathbf{s}^*$ , our virtual control; further, we want to ensure that in causing  $\boldsymbol{\epsilon} \rightarrow \mathbf{0}$ , we still have  $\boldsymbol{\zeta} \rightarrow \mathbf{0}$ .

Again resorting to Lyapunov synthesis, we propose the Lyapunov function candidate:

$$V := V_a + \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \quad (5.23)$$

and compute the derivative:

$$\begin{aligned} \dot{V} &= \dot{V}_a - 2\boldsymbol{\epsilon}^T B^T P \boldsymbol{\zeta} + 2\boldsymbol{\epsilon}^T \dot{\boldsymbol{\epsilon}} \\ &= \dot{V}_a - 2\boldsymbol{\epsilon}^T B^T P \boldsymbol{\zeta} + 2\boldsymbol{\epsilon}^T [\dot{\boldsymbol{s}} - \mathbf{b}(\boldsymbol{\zeta}, \mathbf{s})] \end{aligned} \quad (5.24)$$

Assume that  $\mathbf{s} = \boldsymbol{\eta}$  (i.e., the vector measurement function,  $\boldsymbol{\sigma}$ , is an identity

transform). If this is the case, then  $\dot{\mathbf{s}} = \dot{\boldsymbol{\eta}} = \Upsilon(\boldsymbol{\eta})\mathbf{a}$ , which suggests setting:

$$\mathbf{a} = \Upsilon^{-1}(\mathbf{s}) \left[ \mathbf{b}(\boldsymbol{\zeta}, \mathbf{s}) + B^T P \boldsymbol{\zeta} - \frac{\kappa_2}{2} \boldsymbol{\epsilon} \right] \quad (5.25)$$

leading to:

$$\dot{V} = \dot{V}_a - \kappa_2 \|\boldsymbol{\epsilon}\|_2 \quad (5.26)$$

which, recalling (5.15) regarding the negative definiteness of  $\dot{V}_a$ , is negative definite.

## D. Properties

### 1. Structure

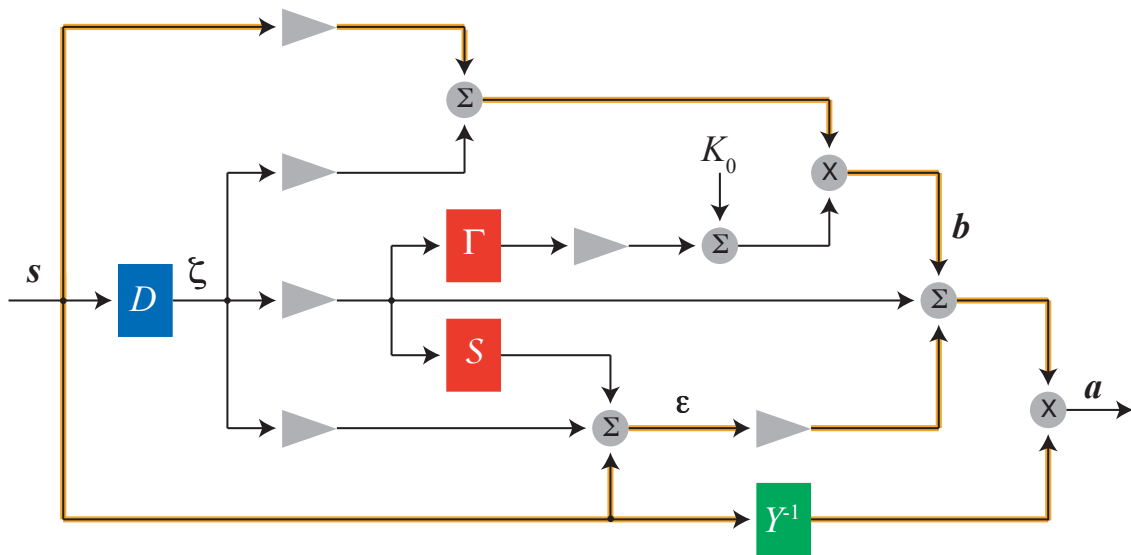


Fig. 44. Structure of the dynamical controller.

Figure 44 illustrates the structure of the controller derived in section C. A key feature of this scheme (distinguishing it from the purely reactive scheme of chapter IV, apart from the increased complexity) is the presence of memory in the form of the dynamical element  $D$  (shown in blue). The output of  $D$ ,  $\boldsymbol{\zeta}$ , strongly influences the

computation of the actuation signal; however, as the signal flow highlighted in yellow indicates, there are feed-forward channels directly from sensing to actuation that are not buffered by  $D$ . Hence, the scheme has both purely reactive and dynamical attributes.

Another significant feature is the  $\Upsilon^{-1}$  block (shown in green). Recall that  $\Upsilon$  describes the evolution of the agent’s perception of the environment as a function of actuation, embedding knowledge of the agent’s embodiment and situatedness. The presence of this static vestige of the plant model in the regulator reflects the cybernetic internal model principle [106]:

“the best regulator of a system is one which is a model of that system in the sense that the regulator’s actions are merely the system’s actions as seen through a mapping . . .”

### a. Neural structure

Two blocks of interest are the pulse ( $\Gamma$ ) and saturation ( $\mathcal{S}$ ) functions (shown in red) which impart a quasi-neural characteristic to the topology. Figure 45 redraws the controller structure, lumping the gains and summing junctions into a single signal combining node to emphasize this neural structure. We can take the view that the scheme includes a small feed-forward network of four *artificial* neurons, two of which have a classic sigmoidal transfer function and two of which are radial basis functions.<sup>4</sup> Alternatively, we can view the whole structure as reminiscent of a single biological neuron [107]. The stable dynamical system,  $D$ , performs a type of leaky integration of environmental stimuli. The result of this integration passes to a vector activa-

---

<sup>4</sup>A radial basis function (centered about  $\mathbf{c}$ ) is a real-valued function whose output is a result purely of the argument’s *distance* from  $\mathbf{c}$ . For example, for scalar  $x$  and  $c = 0$ ,  $\text{pul}(x) = \text{pul}(|x|)$ .

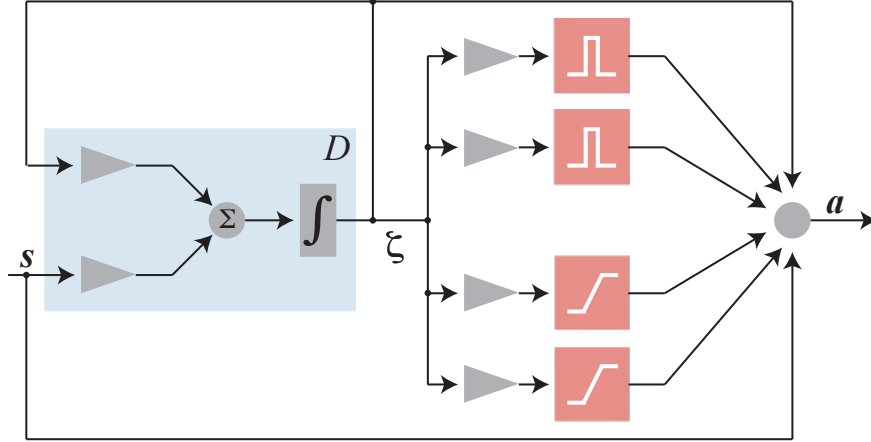


Fig. 45. Neural aspects of the controller's structure.

tion function comprised of the pulse and saturation functions (however, unlike the traditional integrate-and-fire neuron, no resetting of the integrator is done directly).

We note that  $\text{sat}(\cdot)$  and  $\text{pul}(\cdot)$  were chosen in our synthesis because they served as simple, economical function candidates to stabilize the plant. Although the controller derivation does not *require* these functions, it is interesting that our emphasis on economical choices seems to *suggest* their use.

## 2. Scalability

Since the derivation of section C does not restrict the dimensionality of the vectors or matrices involved, it can, in principle, scale in a straightforward way with the dimensionality of the environment and the regulator's dynamical component,  $D$ .

Table IV lists the characteristics of the various quantities used in the dynamical control scheme, while Table V summarizes the computations required by the controller.

Table IV. Quantities used in the dynamical control scheme.

quantity	dimensions	properties
$\boldsymbol{\eta}, \boldsymbol{s}$	$\mathcal{R}^{n_e}$	
$\boldsymbol{\zeta}$	$\mathcal{R}^{n_d}$	
$A$	$\mathcal{R}^{n_d \times n_d}$	
$B$	$\mathcal{R}^{n_d \times n_e}$	
$K_0$	$\mathcal{R}^{n_d \times n_d}$	$A + BK_0$ Hurwitz
$P$	$\mathcal{R}^{n_d \times n_d}$	$P^T = P > 0$
$l_1$	$\mathcal{R}$	$l_1 > 0$
$\kappa_1$	$\mathcal{R}$	
$\kappa_2$	$\mathcal{R}$	

Table V. Computations performed by the dynamical control scheme.

operation	quantity
binary multiply	$n_e^2$
multiply-by-constant	$2n_d(n_d + n_e) + n_e$
$\text{pul}(\cdot)$	$n_e$
$\text{sat}(\cdot)$	$n_e$

### 3. Singularity

In (5.25) the computation of  $\Upsilon^{-1}$  is required; however,  $\Upsilon$  is non-singular for  $s_1 = 0$ , precluding the use of (5.25) for this case. The situation here is analogous to that of section IV.C.1; however, we can not (with ease) bring a graphical method like vector field design to bear on the stabilization of  $\zeta$  and  $\epsilon$  since the high-dimensionality precludes visualization of the full state space.

Instead, we propose a patch: we establish a guard zone about  $s_1 = 0$ , such that whenever the agent senses the target within that zone it bypasses (5.25) and uses the actuation:

$$\mathbf{a} = \begin{bmatrix} a_v \\ a_\omega \end{bmatrix} = \begin{bmatrix} 0 \\ -s_2 \end{bmatrix} \quad (5.27)$$

That is, whenever  $s_1$  becomes sufficiently close to 0, the agent will rotate on the spot (the sense of the rotation will be so as to bring the target either in front of, or behind, the agent) until  $s_1$  is sufficiently far from 0.

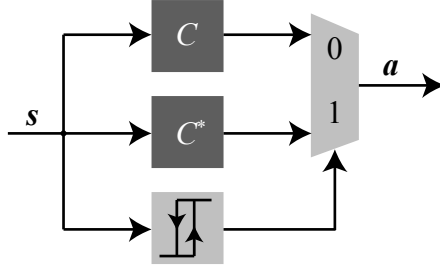


Fig. 46. Patched controller;  $C$  is the dynamical controller of Figure 44 and  $C^*$  implements (5.27).

#### 4. Relaxation of stability

In section C, note that we could have simply specified that the virtual control,  $\mathbf{s}^*$ , be defined:

$$\mathbf{s} = \mathbf{s}^* = K_0 \boldsymbol{\zeta} \quad (5.28)$$

resulting in:

$$\dot{V}_a = \boldsymbol{\zeta}^T \bar{A} P \boldsymbol{\zeta} + \boldsymbol{\zeta}^T P \bar{A} \boldsymbol{\zeta} = -\boldsymbol{\zeta}^T Q \boldsymbol{\zeta} \quad (5.29)$$

which is negative definite, making  $\boldsymbol{\zeta} = \mathbf{0}$  an asymptotically stable equilibrium point as desired. The question then arises: if ignoring  $\mathbf{s}^{**}$  (i.e., setting  $\kappa_1 = 0$ ) leads to a stable virtual control, why introduce  $\mathbf{s}^{**}$ ?

To answer this, let:

$$l_1 \leq \frac{\rho_0}{2} \|B^T P\|_1 \quad (5.30)$$

where:

$$\rho_0 = \frac{\sqrt{n_d}}{\lambda_{\min}(Q)} |\kappa_1| \|B^T P\|_1 \quad (5.31)$$

and  $\lambda_{\min}(Q)$  denotes the smallest eigenvalue of  $Q$ . Setting  $\kappa_1 < 0$  gives rise to a neighborhood of radius  $\rho_0$  about  $\boldsymbol{\zeta} = \mathbf{0}$  where  $\dot{V}_a < 0$  [102]. Hence, making  $\kappa_1$  negative *relaxes* the action of the controller such that, rather than trying to achieve *asymptotic stability* of  $\boldsymbol{\zeta} = \mathbf{0}$ , it strives for *ultimate boundedness* of  $\boldsymbol{\zeta}$  to within some region about  $\mathbf{0}$ .

#### E. Simulation results

Figures 47- 55 present simulation results for the dynamical controller of this chapter. The numerical annotations indicate the order of actions the agent engages in (with 1 denoting the initial agent configuration); the use of the “patched” controller ( $C^*$ ) to resolve singularities is indicated by the red  $\theta$  annotation.

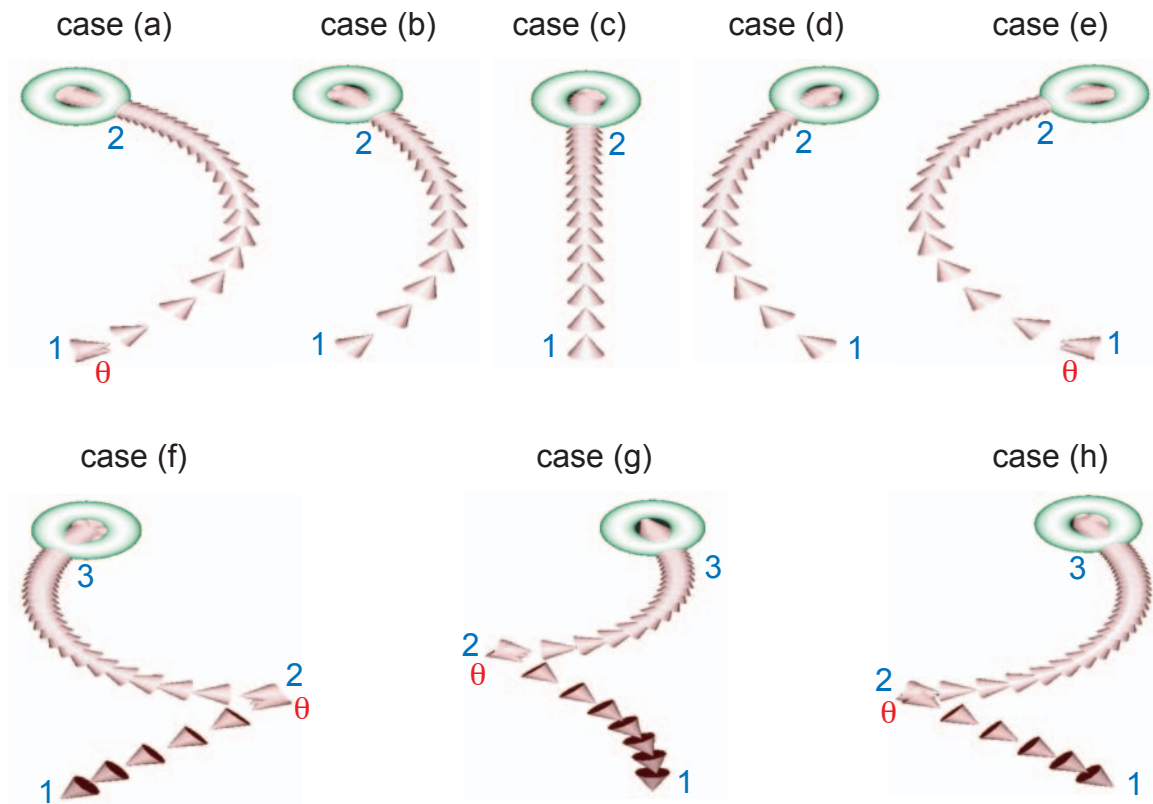


Fig. 47. Simulation results for the dynamical controller with no relaxation of stability ( $\kappa_1 = 0$ ); no obstacles are present.



Fig. 48. Simulation of the dynamical controller with  $\kappa_1 = 0$ ; the agent is impeded by a small ball.



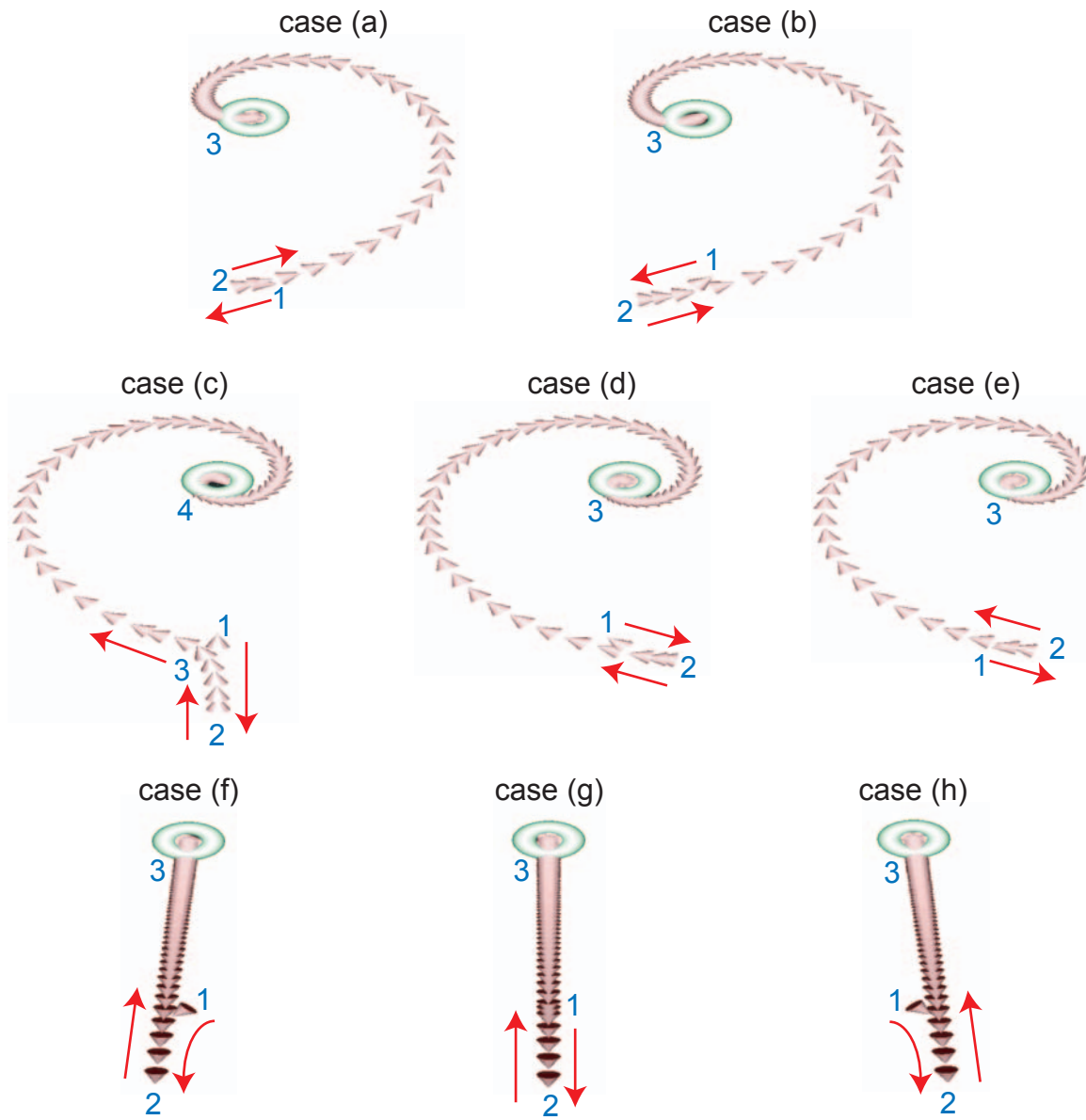


Fig. 49. Simulation of the dynamical controller with relaxed stability ( $\kappa_1 < 0$ ) for the obstacle-free case.

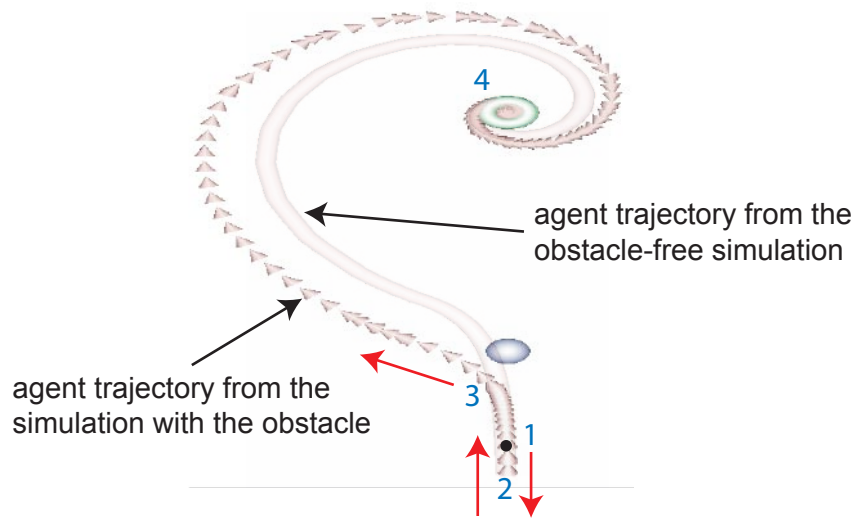


Fig. 50. Simulation of the dynamical controller with  $\kappa_1 < 0$ ; the agent is able to avoid a small ball placed in its path.

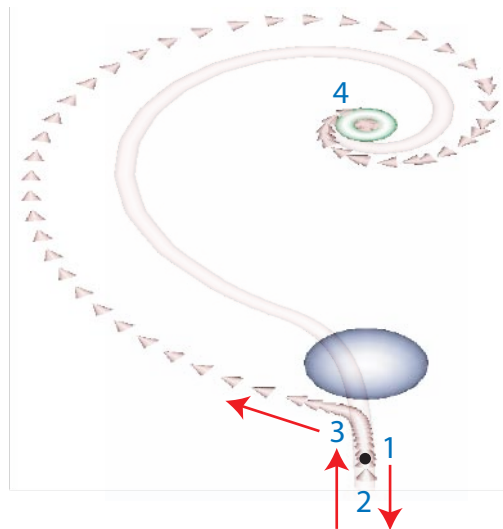


Fig. 51. Simulation of the dynamical controller with  $\kappa_1 < 0$ ; the agent is able to avoid a bigger ball placed in its path.

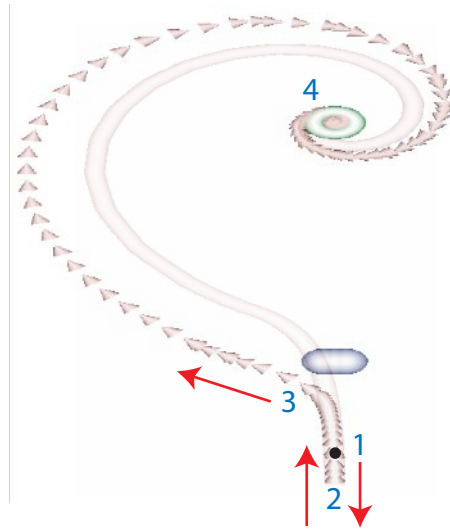


Fig. 52. Simulation of the dynamical controller with  $\kappa_1 < 0$ ; the agent is able to circumnavigate a small wall in its path.

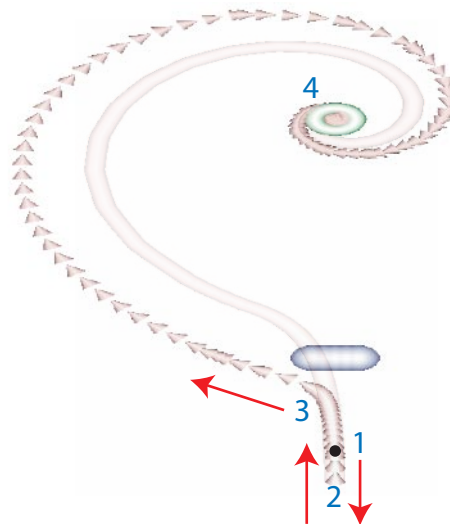


Fig. 53. Simulation of the dynamical controller with  $\kappa_1 < 0$ ; the agent is able to circumnavigate a big wall in its path.

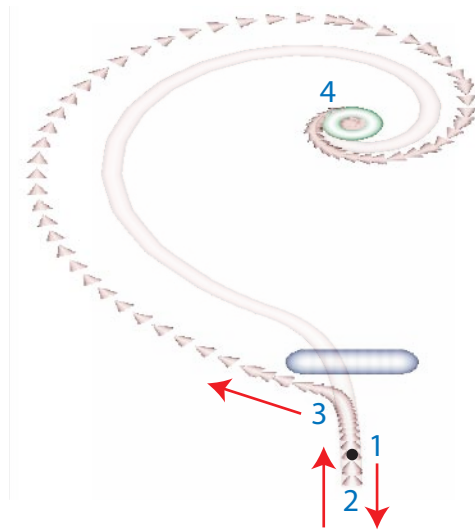


Fig. 54. Simulation of the dynamical controller with  $\kappa_1 < 0$ ; the agent is able to circumnavigate a bigger wall in its path.

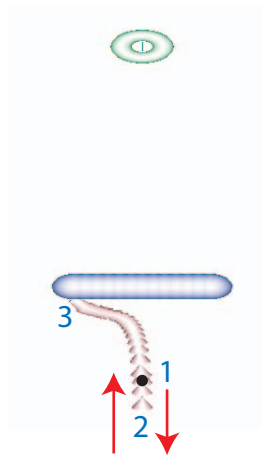


Fig. 55. Simulation of the dynamical controller with  $\kappa_1 < 0$ ; the wall is too big for the agent to circumnavigate.

## F. Weak emergence of satisficing intelligence

Consider Figure 48 which shows the machine getting stuck at a wall, when the parameter  $\kappa_1 = 0$ . Figures 50-54 shows the machine being able to get past various balls and walls when the  $\kappa_1$  is made negative. On the surface this may not seem earth-shattering, since it is clear that the changing of  $\kappa_1$  had something to do with this; however, on referring back to the derivation of the controller, we note that there is no coupling of obstacle information to the steering channel.

A second possibility is that altering  $\kappa_1$  causes the controller to bias trajectories in general—this, in fact, is true. However, it is still not the cause of the obstacle avoidance, since the trajectory deviation when  $\kappa_1 < 0$  and no obstacle is in front of it is *less* than the trajectory deviation when an obstacle is placed in front—hence, the controller is taking action due to the influence of the obstacle. The question arises: without being given obstacle information, how does this controller take the appropriate action?

The answer is that the effect of the feedback loop is to suppress disturbances. When the agent moves towards the obstacle it is forced to slow down. This shows up as a disturbance to the controller, and the controller compensates. However, now the question arises as to why this behavior does not manifest when  $\kappa_1 = 0$ . On referring to the controller equations, we note that when  $\kappa_1 = 0$  the controller is very aggressive, pursuing asymptotic stability—it wants to get to the target and will not admit deviations. However, when  $\kappa_1 < 0$ , the pursuit of asymptotic stability is relaxed to the pursuit of ultimate boundedness, i.e., the controller is open to admitting non-optimal solutions.

Simon [5, 6] suggests that cognitive systems are systems that *satisfice*, that is, systems that find “tolerable” rather than optimal solutions. With respect to our

system, the behavior of Figures 50-54 is an example of satisficing intelligence—the agent takes locally non-optimal actions that allow it to get around the wall. In [108], Bedau introduced the concept of *weak emergence*, reporting that weak emergence is manifest in all complex systems with [109] placing it as requisite property of complex adaptive systems.

**Definition 2 (Weak emergence).** *A phenomenon,  $P$ , of a dynamical system,  $S$ , with dynamics specified by  $D$ , is weakly emergent if and only if  $P$  can be derived from  $D$  and the external conditions of  $S$  but only by simulation.*

We appeal to this definition to show how satisficing intelligence is a weakly emergent property of the dynamical scheme presented in this chapter.

**Theorem 2.** *The satisficing obstacle avoidance behavior exhibited in Figures 50-54 is a weakly-emergent property of the dynamical control scheme presented in section C.*

*Proof.* ( $\Rightarrow$ ) We first show that the manifestation of  $P$  is rooted in  $\{S, D\}$ . This is straightforward as through experimentation we observe that  $P$  arises when  $\kappa_1 < 0$  (i.e., in the simulations of Figures 50-54). With  $\kappa_1 < 0$ , the agent exhibits more varied behavior (including taking locally non-optimal actions) when it meets an obstacle and reverses and/or turns to circumnavigate the obstacle. When  $\kappa_1 = 0$ , the agent is aggressive as it tracks the target—however, this *fanaticism* (to use the AI-inspired terminology of [110, 111]) prevents it from taking non-optimal deviations from its optimizing path towards the target.

Hence, the cause of  $P$  can be traced to  $\{S, D\}$  via the parameter  $\kappa_1$ —the degree to which the stability of  $C_1$  is relaxed.

( $\Leftarrow$ ) Now we show that  $P$  can *only* be seen to emerge through simulation, that is, we can not derive its manifestation purely by analyzing  $\{S, D\}$ . We note that in the synthesis of Section C, we did not explicitly design behavior for turning around

obstacles. This was because the agent we designed only had access to *local non-directional* obstacle sensing. Hence, we simply did not have access to the information required to design a regulator that could directly maneuver around an obstacle. Thus, by solely considering  $\{S, D\}$  it is not possible to deduce the emergence of  $P$  because the regulators in  $\{S, D\}$  do not receive sufficient information to, by design, engage in  $P$ -like behavior.

What we did design into the system, through the  $\mathbf{s}^{**}$  virtual control term (switched by setting  $\kappa_1 < 0$ ), was a relaxed requirement for stability. That is, we lessened the *constraint* on the level one controller providing it with the *freedom* to take more varied actions—but we can not say what it will exactly do. Interaction with the environment—through simulation—is necessary to observe the manifestation of  $P$ . □

## CHAPTER VI

### INTEGRATED CONTROL ARCHITECTURES FOR SINGLE-AGENT SYSTEMS

The implication is that a sensory-interactive goal-directed motor system is not simply an appendage to the intellect, but is rather the substrate in which intelligence evolved. There is, in fact, no evidence for a clear demarcation between the motor system and the intellect. Quite to the contrary, much anatomical, neurophysiological, and behavioral evidence suggests that complex behavior is generated in a multilevel control hierarchy where motor outputs are merely the terminal symbols . . . (James S. Albus [30])

#### **A. Introduction**

So far we have covered the development of basis behaviors to endow an autonomous robot with faculties to navigate. We started, in chapter IV, with static controllers which achieved the singular task of target sensor regulation for taxis and other target-referenced behaviors. Next, in chapter V, we developed dynamical control schemes for navigation, exploiting topological properties of the control loop to inject obstacle and target sensor information to both achieve taxis and obstacle avoidance.

In both cases we adopted toolsets from control theory and dynamical systems theory to develop controllers for which we could make rigorous statements. With this chapter and the next, however, we move past control-theoretic tools and into behavior-based robotics proper. Unlike in control theory, where we have quantitative models of the world and sensors that provide us with the information we require, now we begin to deal with an artificial organism in the world, with bounded resources



*but* still needing to cope in a competent manner (i.e., needing to satisfice). We may still use models in our development, but now they are more qualitative; we still have sensors, but now we acknowledge reality and the fact that our sensors often provide insufficient information. The use of the term organism is fitting, for this work is within the broader field of *hard* artificial life,<sup>1</sup> and as such we recall the concept of weak emergence. In this chapter and the next, hence, we will be *engineering* the emergence of useful behaviors, by providing our agent (via the control architecture) with the raw ingredients for this emergence: the basis behaviors of section IV.

### 1. Integration of behaviors

The dynamical controller represents a primitive example of the topic of this chapter—an *integrated control architecture*—being able to deal concurrently with both taxis and obstacle avoidance. In this chapter we continue in this vein, addressing the development of more sophisticated control schemes for single agent systems. We combine the computational machinery of chapters IV with architectural insights to yield an architecture for robot cognition that addresses:

- individual agents that navigate on their own and are unaware of other agents (which we address in this chapter)
- collective groups of agents that interact passively to realize useful collective behaviors (which will be addressed in the next)

---

<sup>1</sup>Artificial life (alife) can be divided into three broad groups: wet, soft, and hard. Wet alife seeks to develop artificial organisms from the perspective of biochemistry and systems biology; soft alife uses the immense processing power provided by modern computers to simulate models of artificial organisms. Hard alife seeks to realize embodied and situated artificial artifacts that can, at the very least, serve as an approximation of primitive life, exhibiting purposive cognitive behavior in the world. In placing our work within the alife tradition [101], we appeal to the perspective of Maturana and Varela [112] that “living systems are cognitive systems, and living is a process of cognition.”

## B. Machine organization

### 1. Hierarchy

Hierarchical structure is an observed characteristic of living organisms, and whose importance has been addressed in the artificial life literature [113, 114, 115, 116]. In fact, [113] poses the synthesis of dynamical hierarchies at all scales as one of fourteen crucial open problems for the synthesis of artificial life. An early formulation of a continuous-time hierarchical dynamic architecture was Ashby’s *ultrastable* system [3]. A two level hierarchy was specified consisting of a lower-level “reacting” part strongly coupled to the environment, and a higher-level system operating on a slower time scale that regulated the lower-level system. In the practical realization of an ultrastable system—the Homeostat—the higher-level system possessed the ability to *search* for successful controls to regulate the lower-level system. Another example is the cascaded architecture proposed by Albus [30] where the output of one level becomes the input to an adjacent lower level. Sensory feedback from the environment entered all levels of the hierarchy, with higher levels possibly using abstractions of lower-level senses (e.g., sensory information from which pertinent features have been extracted).

In autonomous systems, the need for hierarchical organization generally comes [101, 117] from two sources:

- the multi-scale nature of environmental phenomena that the agent must cope with, separating fast controllers from slow ones
- the varying degrees of abstractness in the regulation strategy, separating lower-level controllers (dealing with ‘concrete’ phenomena grounded in the agent’s sensori-motor embodiment) from higher-level ones (dealing with more abstract “decision making”)

An example of the former can be seen in the navigation problem that we are concerned with in this work where the situated agent must cope with phenomena occurring on a variety of time scales ranging its perception of obstacles (fast time scales), to the tracking of a distant target, to its perception of overall progress (slow, using long-term trends in sensor data). The same applies to spatial scales, as shown in Table VI which describes the separation in scale of various classes of behavior.

Issues of action selection often give rise to the need for a higher-level arbiter to make decisions that govern lower-level actions; these decisions are often based on abstracted sensory information. For example, to schedule the influence from a set of several regulators (with differing goals) onto a smaller set of actuators (e.g., the case where a robot has two independent processes that demand the use of a single kinematic actuator), a higher-level arbitration mechanism is needed.

Table VI. A hierarchy of behaviors for navigation.

level	behavior	spatial scale
0	motion	velocity
1	velocity tracking	velocity, position
	target tracking (taxis)	position
	obstacle avoidance	position
2	searching	path

### a. The proposed architecture

In our approach, the separation of regulation tasks due to varying abstractness and scale suggests a recursive view of the control strategy. Consider a low-level controller,<sup>2</sup>  $C_0$ , which is concerned with regulating fast phenomena in the environment. Let  $E_0$  model that portion of the environment,<sup>3</sup>  $E$ , that is responsible for this fast phenomena (and whose evolution is governed by the actuation output of  $C_0$ ,  $\mathbf{a}_0$ ). This is the situation of Figure 56(a).

At a higher level of abstraction, a slower temporal scale or a longer spatial scale, the agent’s controllers will be attempting to regulate a similarly higher level of phenomena—however, generally, to regulate this phenomena it will still have to actuate change via the lower levels.<sup>4</sup> Hence, in order to design a regulator for a higher level, the control strategy must account for the effect of all systems “downstream” from the controller. Consider the two-level hierarchy of Figure 56(b). The agent’s higher faculty,  $C_1$ , acts through  $C_0$  and  $E_0$  to influence the subject of its actions,  $E_1$ . Hence, the plant model used to synthesize  $C_1$ ,  $P_1$ , must account for all of these downstream systems.

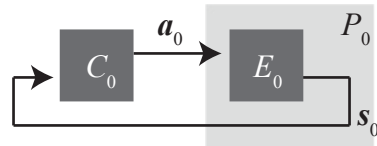
Hence, we describe our general architecture (illustrated in Figure 56(c)) as a hierarchy of regulators, where the controller at level  $i$ ,  $C_i$ , seeks to regulate its sensory

---

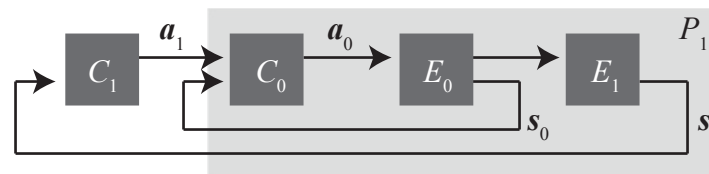
<sup>2</sup>For a mobile robotics problem, the zeroth level address the problem of motion causation, i.e.,  $E_0$  consists of the agent’s motor actuators and  $C_0$  is a system that, given a velocity command, controls  $E_0$  to achieve that velocity. We do not address the design of this level as it is out of the scope of our work. In our development, as mentioned in chapter IV, we subsume the competence provided by this zeroth level.

<sup>3</sup>The portion of the world that is external to the agent’s control architecture.

<sup>4</sup>An exception is when the agent can communicate with other agents; in this case, the communication channel can serve as a link between higher levels of cognition of separate agents, bypassing the need to effect change through the environment. This is in contrast to *stigmergic* cooperation between agents, where there is no direct communications channel. In this case, the agent must act through the environment (i.e., using its lower-order faculties); these actions are then sensed by others who then act in response.



(a) One level.



(b) Two levels.

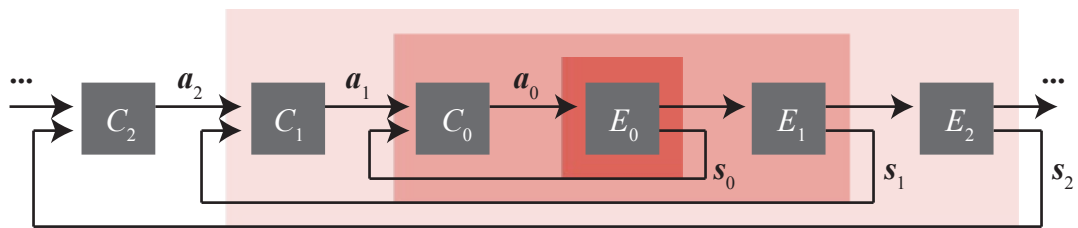
(c)  $n$  levels.

Fig. 56. Recursive development of our hierarchical control architecture.

perception of the environment at level  $i$ ,  $E_i$ ;  $E_i$  encapsulates those aspects of the overall environment that are relevant to the particular cognitive skill that  $C_i$  realizes. To synthesize goal-directed behavior, the derivation of  $C_i$  needs a plant model,  $P_i$ , of the world “downstream” from it (i.e., including all controllers  $C_j$ ,  $j < i$ , and environmental models,  $E_j$ ,  $j \leq i$ );  $P_i$  is defined according to the following recursion:

$$\begin{aligned} P_0 &:= E_0 \\ P_i &:= C_{i-1}P_{i-1}E_i \end{aligned} \tag{6.1}$$

## 2. Layering of behaviors

In a complex, dynamical environment, multi-scale phenomena generally bombard the agent in parallel, causing the agent to have multiple objectives that must be addressed concurrently. The society of minds theory suggested that intelligence emerges from this mix of parallel processes. Brooks developed a robotic control scheme—the subsumption architecture—that practically realized this idea, “vertically” decomposing tasks so that separate *layers* (as opposed to *levels*<sup>5</sup>) perform their objectives concurrently.

Apart from Minsky and Brooks, there is another more direct route to the structural insights of subsumption, a route that we take in this work, inspired by hardware design. Since we target custom hardware realizations (whether analog or digital), we are unrestricted by requirements for serial execution (as with general-purpose computers). Instead, we are free to instantiate hardware as needed, adopting parallel regulation hardware to deal with parallel phenomena. Hence, from this perspective,

---

<sup>5</sup>In this work, the term ‘level’ pertains to hierarchical structure, that is, one can define an ordering of levels based on some topological criteria (e.g., for us, a lower level is one that is being driven by a higher level). A ‘layer,’ on the other hand, denotes a parallel entity; there is no structural order here since two parallel layers can operate independently without passing information between each other.

a subsumption-like architecture is the natural solution.

Figure 57 illustrates a layered architecture for a regulator which, according to the scheme of Figure 56, would be placed within one of the control levels (i.e., within the  $C_i$  blocks). The individual regulators,  $R_a^b$ , correspond to the basis behavior controllers designed in chapters IV and V. As can be seen, mutually exclusive control actions enter multiplexers where only one action is selected; the actuation signal,  $\mathbf{a}_{i+1}$ , from a higher level controller governs this selection. The outputs from the multiplexers represent control actions that can be used concurrently. They enter a node where they are combined (possibly along with  $\mathbf{a}_{i+1}$ ) in some fashion to produce the resultant composite actuation,  $\mathbf{a}_i$ .

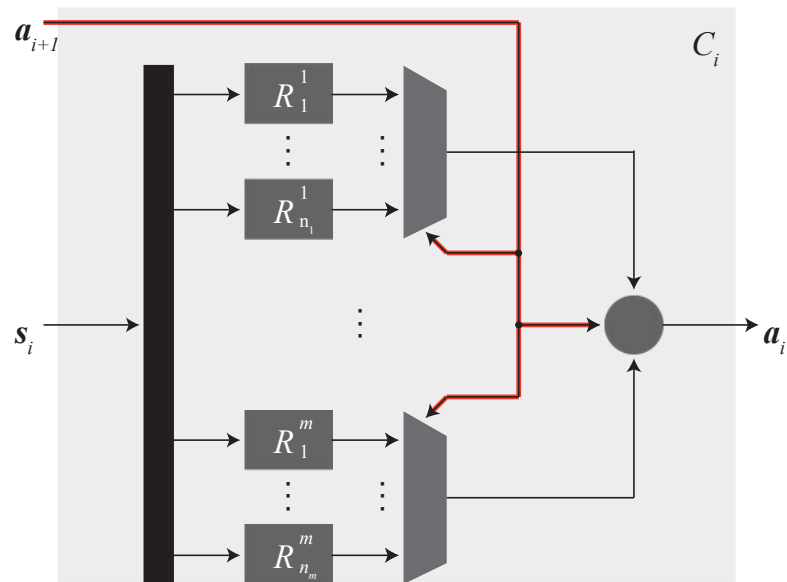


Fig. 57. The architecture of a general layered controller at the  $i$ -th level of hierarchy,  $C_i$ . Elementary controllers,  $R_a^b$ , that realize various basis behaviors are grouped according to whether they address concurrent goals (in which case they have different superscripts) or exclusive goals (in which case they have different superscripts).

## C. Static schemes for single agent systems

### 1. Obstacle avoidance

Consider an agent with front and rear obstacle sensors as shown in Figure 58. Let  $\eta_{\Omega,f}$  denote the distance between the agent,  $M$ , and  $\Theta_{\Omega,f} \cap (\cup_{i=1}^{n_{\Omega}} \Omega_i)$ , and let  $\eta_{\Omega,r}$  denote the distance between the  $M$  and  $\Theta_{\Omega,r} \cap (\cup_{i=1}^{n_{\Omega}} \Omega_i)$ , where ( $n_{\Omega}$  is the number of obstacles in the environment). The obstacle sensors, having finite sensing radius, return a measurement,  $\mathbf{s}_{\Omega}$ , of these values:

$$\mathbf{s}_{\Omega} = \begin{bmatrix} s_{\Omega,f}(\eta_{\Omega,f}) \\ s_{\Omega,r}(\eta_{\Omega,r}) \end{bmatrix} \quad (6.2)$$

where  $s_{\Omega,f}(\cdot)$  and  $s_{\Omega,r}(\cdot)$  are defined in Figure 41 of chapter V.

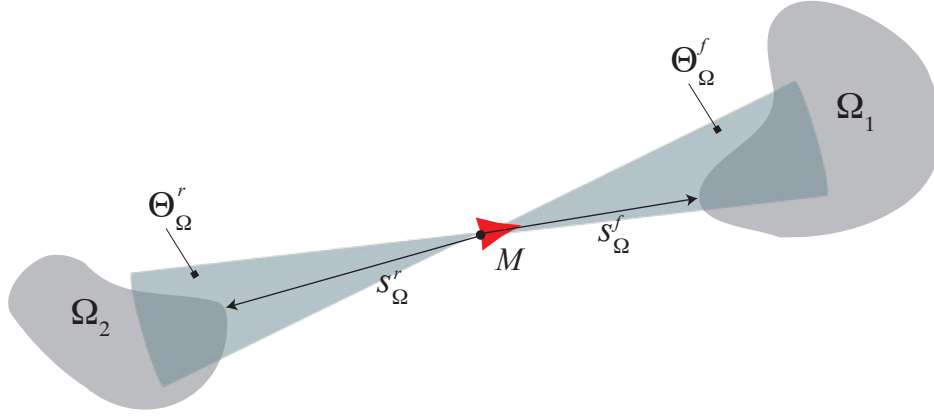


Fig. 58. Agent setup with front and rear obstacle sensors.

These sensors provide a monocular view of obstacles in front of and behind the agent. We can use them to design a reactive collision avoidance strategy by specifying a translational motion attenuator which filters commanded translational speeds by



the map:

$$a_v = \begin{cases} (1 - s_{\Omega,f})v & \text{for } v > 0 \\ (1 - s_{\Omega,r})v & \text{for } v < 0 \\ 0 & \text{for } v = 0 \end{cases} \quad (6.3)$$

preventing the agent from ever hitting an obstacle. Beyond this, however, the utility of these sensors in a reactive scheme for obstacle avoidance by navigation *around* obstacles is limited by the absence of directional information.

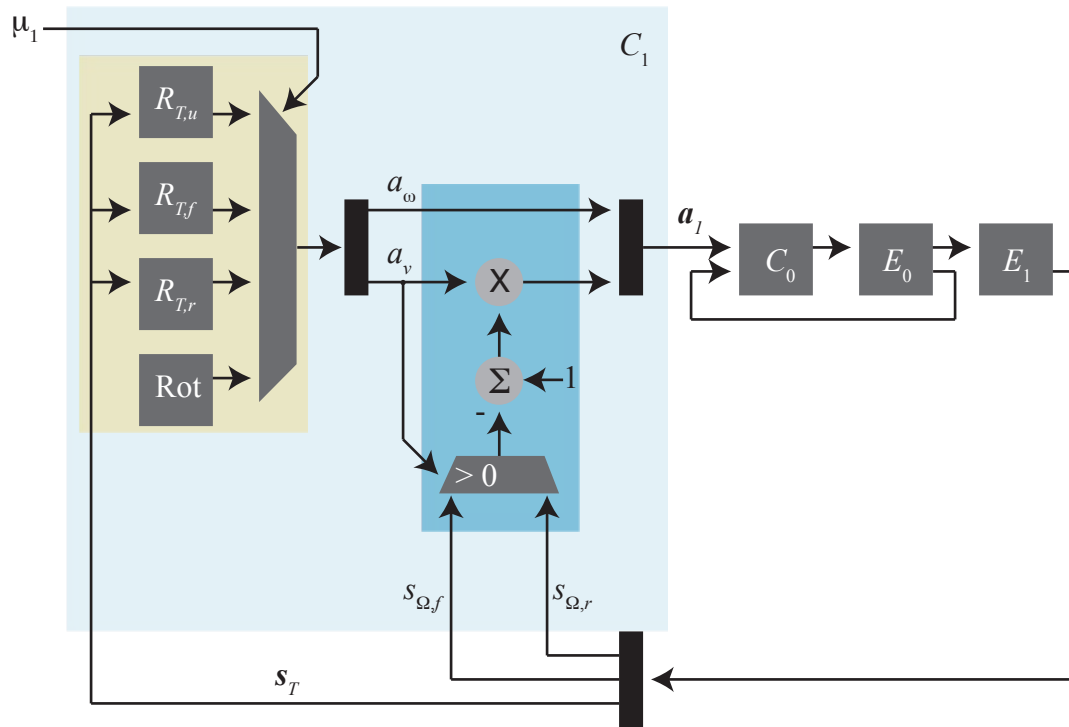
To develop our navigation scheme we use the obstacle sensors to provide the agent with motion constraints that it must operate under. Hence, we propose the scheme of Figure 59(a) which presents a level one controller,  $C_1$ , as a layering of three regulators:

- $R_{T,u}$  realizes *unconstrained* taxis
- $R_{T,f}$  realizes constrained taxis by *forward-only* motion
- $R_{T,r}$  realizes constrained taxis by *reverse-only* motion

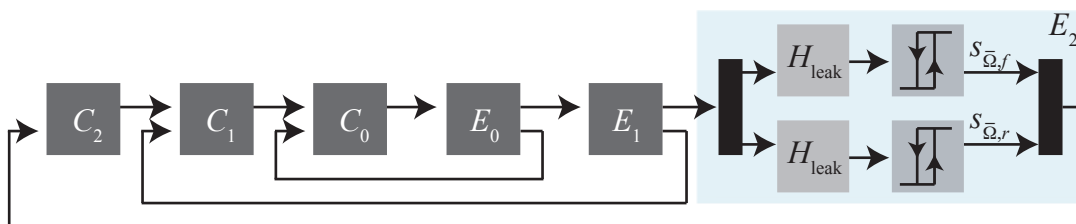
and an open-loop behavior,  $Rot$ , which drives the agent to rotate regardless of the target sensor.

Now the question is how do we drive  $\mu_1$ , the behavior selection signal? We can define it via a static map as a function of the obstacle sensor data,  $\mu_1(\mathbf{s}_\Omega)$ ; however, this gives rise to some potential problems:

- *elimination of useful motion*: many of the basis behaviors in this work involve rotation and translation (indeed, for the behaviors we designed in chapter IV, pure translation or rotation are rare). Hence, the combination of obstacle-based attenuation of translational speed (6.3) and whatever rotational motion the agent is engaged in *can* serve to steer the agent away from the obstacle and towards the target. By reactively switching between behaviors using a static



(a) Level one. The obstacle-based speed attenuator of (6.3) is highlighted in dark blue, while the sub-system for taxis is highlighted in yellow.



(b) Level two. The overstimulation filters are placed in the environment because they are a part of the agent's sensors, and are hence strictly outside of the controller.

Fig. 59. Development of a reactive control architecture for single agent systems.

function of the obstacle sensor data, however, we might be excluding this sort of useful motion.

- discontinuous switching: transient stimulation by obstacles (e.g., when the agent catches a brief sight of an obstacle during its motion, even though it may not be moving towards it) could result in many “hard” transients on the actuators as the agent makes discontinuous switches between behaviors (e.g., going from forward motion to reverse motion, without slowing down or stopping in between). This can result in increased wear on the actuators and drive electronics.
- chattering: the agent can get trapped in a limit cycle where it oscillates about the switching point between constrained and unconstrained behaviors, resulting in no net progress towards the target. For example, suppose an agent encounters an obstacle and then starts to reactively reverse away from the obstacle (due to it switching to constrained taxis). Once it gets sufficiently far from the obstacle, it will switch back to unconstrained taxis, moving forwards, and encountering the obstacle again. This repeating sequence of actions can trap the agent at the obstacle.

Alternatively, we can extract a longer-term trend from the obstacle sensors and govern behavior selection based on this. With the latter scheme a separation of time-scale and abstraction exists between the navigational basis behaviors and the selection of a behavior to adopt. This suggests the use of a controller at a higher level (level two) in the hierarchy.

To realize this scheme, we first create an abstracted measure of obstacle sensor *overstimulation* by passing obstacle sensor data through a leaky integrator,  $H_{\text{leak}}$ :

$$\dot{y} = -\kappa_{\text{leak}}y + u \tag{6.4}$$

(for  $\kappa_{\text{leak}} > 0$ ) and then through a hysteresis function to produce  $s_{\bar{\Omega},f}$  and  $s_{\bar{\Omega},r}$ , as illustrated in the  $E_2$  block of Figure 59(b).

Let the actuation signal from  $C_2$  be defined:

$$\mathbf{a}_2 = \begin{bmatrix} a_{2,f} \\ a_{2,r} \end{bmatrix} \quad (6.5)$$

where (for  $i \in \{f, r\}$ )  $a_{2,i} \in [0, 1]$  and  $a_{2,i} = 0$  means that motion in direction  $i$  is unconstrained, while  $a_{2,i} = 1$  means that motion in direction  $i$  is fully constrained (i.e., disallowed). We can thus map  $\mathbf{a}_2$  to the behavior selection multiplexor of Figure 56(b) such that  $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$  selects  $R_{T,u}$ ,  $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$  selects  $R_{T,f}$ ,  $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$  selects  $R_{T,r}$ , and  $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$  selects Rot. Now, to design controller  $C_2$  we need to specify a plant model,  $P_2$ , that reflects how  $C_2$ 's actuation of  $\begin{bmatrix} a_{2,f} \\ a_{2,r} \end{bmatrix}$ , passing through the cascade of systems  $\{C_1 C_0 E_0 E_1 E_2\}$  affects the evolution of  $\mathbf{s}_{\bar{\Omega}} := \begin{bmatrix} s_{\bar{\Omega},f} \\ s_{\bar{\Omega},r} \end{bmatrix}$ . Qualitatively, when  $s_{\bar{\Omega},f} > 0$  we expect that constraining forward motion (i.e., making  $a_{2,f} > 0$ ) should tend to decrease  $s_{\bar{\Omega},f}$ , and similarly when  $s_{\bar{\Omega},r} > 0$  then making  $a_{2,r} > 0$  to constrain reverse motion should tend to decrease  $s_{\bar{\Omega},r}$ . We can describe this dynamically by:

$$P_2 : \begin{cases} \dot{s}_{\bar{\Omega},f} = -a_{2,f} \\ \dot{s}_{\bar{\Omega},r} = -a_{2,r} \end{cases} \quad (6.6)$$

To synthesize a controller that will bring obstacle overstimulation (i.e.,  $\mathbf{s}_{\bar{\Omega}}$ ) to  $\mathbf{0}$ , we define the scalar-valued function:

$$V := \frac{1}{2} \mathbf{s}_{\bar{\Omega}}^T \mathbf{s}_{\bar{\Omega}} \quad (6.7)$$

which is positive-definite with respect to  $\mathbf{s}_{\bar{\Omega}}$ . Differentiating with respect to time, we

obtain:

$$\dot{V} = -s_{\bar{\Omega},f}a_{2,f} + s_{\bar{\Omega},r}a_{2,r} \quad (6.8)$$

Setting  $a_{2,f} = \text{sgn}(s_{\bar{\Omega},f})$  and  $a_{2,r} = \text{sgn}(s_{\bar{\Omega},r})$  makes (6.7) negative-definite with respect to  $\mathbf{s}_{\bar{\Omega}}$ , hence bringing  $\mathbf{s}_{\bar{\Omega}}$  to  $\mathbf{0}$ .

## 2. Searching

Searching in the absence of a priori knowledge of the search space—goal-directed trial-and-error [30]—is a fail-safe, often open-loop, behavior that an agent can engage in when confronted with a situation that it can not control (e.g., if it can not sense the presence of a target); Ashby’s Homeostat uses a random mechanism to search for favorable controls to properly regulate the system.

The Rot behavior used in the previous section realized a simple case of searching. In that case, the agent executed Rot whenever both its obstacle sensors were overstimulated, a case that prevented the agent from using any of its basic taxis behaviors since all translational motion was excluded; the agent’s rotation served to search for a favorable orientation to escape this situation.

Consider a target that is out of range of an agent that desires to track it. In the absence target sensor data, the agent would be unable to engage in taxis. In that case, what should the agent do? Ideally, the agent would execute a search of the space around it, with the goal that by covering a sufficient area it eventually enters sensing range of the target [118]. But what are the characteristics of an effective search?

An unbiased search strategy involves executing a path through space that, if done indefinitely, would eventually cover the entire space. Although a variety of “space-filling” [119] curves exist, here we focus on the Archimedean spiral [120], illustrated in Figure 60, as a prototypical path for a search strategy. As the figure illustrates, a useful characteristic of Archimedes’ spiral is the uniform separation,  $\rho$ , between

successive crossings of the spiral across any ray emanating from the origin of the spiral. If we consider an agent with a local sensing radius of  $r_s$ , then the agent executing an Archimedean spiral where  $\rho = 2r_s$  would cover all space enclosed by the spiral.<sup>6</sup> The parametric equations that describe this spiral are of the form:

$$\begin{aligned} g_1(t) &= t \sin(t) \\ g_2(t) &= t \cos(t) \end{aligned} \tag{6.9}$$

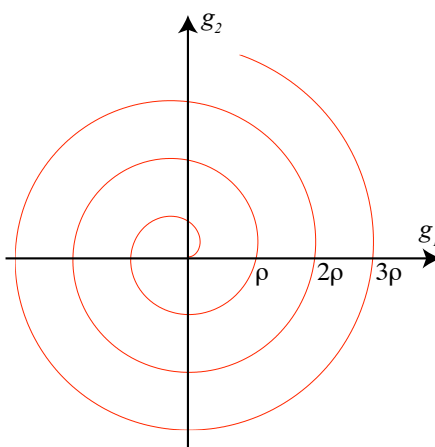


Fig. 60. An Archimedean spiral (also known as an arithmetic spiral). Successive crossings of this curve across the  $g_1$  axis (and, more generally, across any ray emanating from the spiral’s origin) are separated by  $\rho$ .

#### a. Design of a reference oscillator for searching

In the following we present the design of a dynamical system to generate a reference signal that an agent can track to execute approximations of Archimedean spirals through space.

---

<sup>6</sup>With respect to the figure, we can visualize the search by a circular “paintbrush” of radius  $\frac{1}{2}\rho$ , centered about the path, and tracing along the path.

**Searching forever** Ideally, we desire a signal generator that enables an endless search until a target is found. Consider a simple unicycle, with translational speed  $v$  and rotational speed  $\omega$ , and recall that the trajectory executed by this unicycle (with respect to the  $g_1 - g_2$  plane of Figure 60) is:

$$\begin{aligned}\dot{g}_1 &= v \cos(\psi) \\ \dot{g}_2 &= v \sin(\psi) \\ \dot{\psi} &= \omega\end{aligned}\tag{6.10}$$

(where  $\phi$  denotes the orientation of the vehicle in the  $g_1 - g_2$  plane). Note that if we hold the translational and rotational speeds constant, i.e.,  $v(t) \equiv v, \omega(t) \equiv \omega$ , then we have:

$$\begin{aligned}\psi(t) &= \int_0^t \omega d\tau = \omega t \\ g_1(t) &= \int_0^t v \cos(\psi) d\tau = \int_0^t v \cos(\omega\tau) d\tau = \frac{v}{\omega} \sin(\omega t) \\ g_2(t) &= \int_0^t v \sin(\psi) d\tau = \int_0^t v \sin(\omega\tau) d\tau = -\frac{v}{\omega} \cos(\omega t) + \frac{v}{\omega}\end{aligned}\tag{6.11}$$

and hence the radius of oscillation of the path executed,  $r(t) := \sqrt{g_1^2 + (g_2 - \frac{v}{\omega})^2}$ , is  $r(t) \equiv \frac{v}{\omega}$ . This suggests that we can control the radius of oscillation of the path by obtaining either:

- a suitable monotonic increasing reference function for  $v(t)$
- a suitable monotonic decreasing reference for  $\omega(t)$

Since it is not practical to increase the translational speed of a vehicle without bound, we consider the second option. It is out of the scope of this work (and may not be trivial) to derive explicit expressions for  $\omega(t)$  that would result in a vehicle executing the exact trajectory of (6.9), so we present an empirically-obtained

candidate that enables an approximation. Setting:

$$\omega(t) = \frac{1}{\sqrt{t}} \quad (6.12)$$

and holding  $v(t) \equiv v$ , yields the trajectory illustrated in Figure 61.

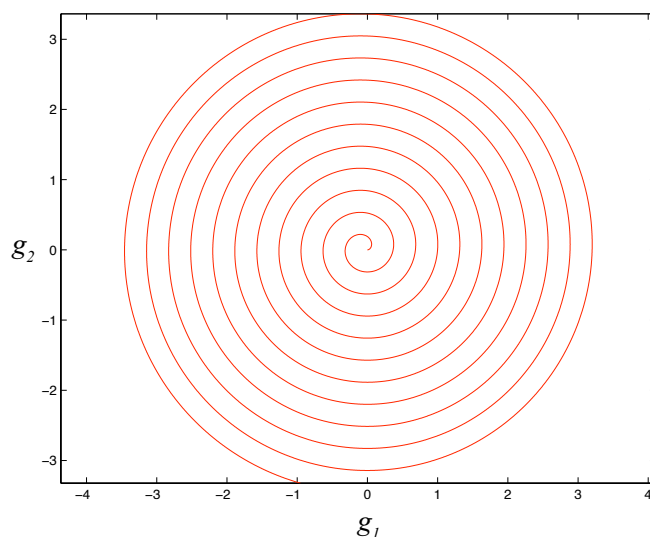


Fig. 61. The trajectory of a vehicle starting at the origin, under (6.12) and  $v(t) \equiv 0.1$ .

**Interleaved searching** The problem with the earlier development is the impracticality of accurately generating (6.12) for a long period of time—eventually, either:

- the finite precision of a digital realization
- analog integrator saturation and possibly drift due to offset errors

will prevent the generation over long time spans of the requisite linear ramp,  $t$ , that forms the basis for (6.12).

Consider a search strategy that is composed of two behaviors. First, the agent executes a spiral search for the finite duration  $t_s$ , which is the maximum period of



time that we can generate a practical ramp signal.<sup>7</sup> Next, it takes off in a bee-line in whatever direction it is in for duration  $t_b$  (whose duration is, again, constrained by practical considerations). In this manner, the system exhaustively explores a limited region and then moves off to potentially cover an unexplored area.

To realize this, we invoke a simple stable oscillator (illustrated in Figure 62):

$$\begin{aligned}\dot{\xi}_1 &= f \operatorname{sgn}(\xi_2) + \xi_1(1 - \|\boldsymbol{\xi}\|_1) \\ \dot{\xi}_2 &= -f \operatorname{sgn}(\xi_1) + \xi_2(1 - \|\boldsymbol{\xi}\|_1)\end{aligned}\tag{6.13}$$

where the frequency,  $f$ , of the oscillator is set by:

$$f = \begin{cases} \frac{2\pi}{t_s}, & \xi_2 \geq 0 \\ \frac{2\pi}{t_b}, & \xi_2 < 0 \end{cases}\tag{6.14}$$

Now, holding  $v(t) \equiv v$  we set:

$$\omega(\boldsymbol{\xi}) = \begin{cases} \frac{1}{\sqrt{\xi_1}}, & \xi_2 \geq 0 \\ 0, & \xi_2 < 0 \end{cases}\tag{6.15}$$

Since the oscillatory dynamics of (6.13) are piecewise constant,  $\xi_1$  and  $\xi_2$  evolve as linear functions (ramps) of time.

### 3. Integration

To realize an integrated control architecture, we stitch together the behaviors of taxis and searching with obstacle avoidance. Since these are mutually-exclusive, never being active simultaneously (since one is active when there *is* a target in range, and the other when that is not true) we can integrate them using a multiplexor as shown in Figure 63. As shown in the figure, we apply a leaky integrator to  $s_{\bar{T}}$  (a sense

---

<sup>7</sup>This period  $t_s$  will either be due to limitations in the underlying analog circuitry, or governed by our choice of digital word length.

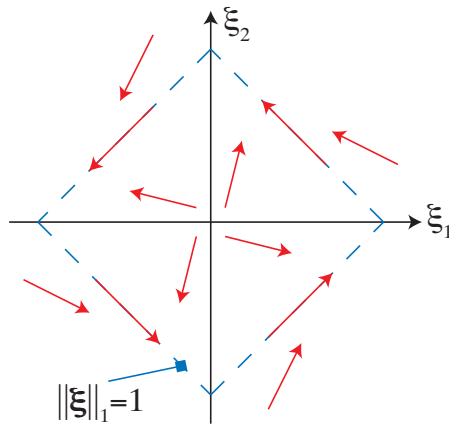


Fig. 62. The vector field structure of oscillator (6.13).

that indicates that the target is out of range) followed by hysteresis to perform the selection; we do this to decouple the actuators from fast transients where the target briefly goes out of range.

Figures 64 and 65 present the result of simulating an agent with this control architecture in a virtual environment.

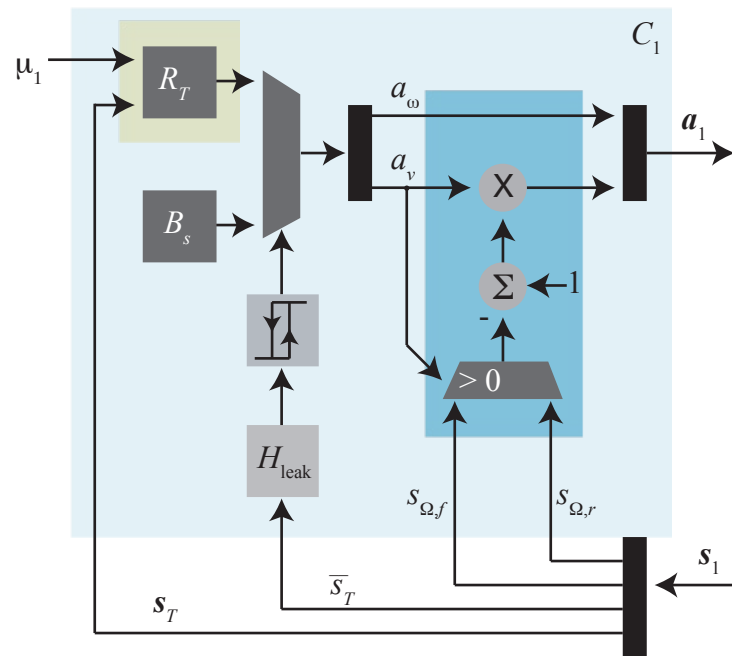


Fig. 63. A controller integrating taxis with an open-loop search behavior.

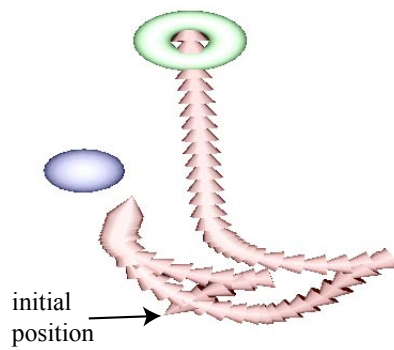


Fig. 64. An example of basic obstacle avoidance.

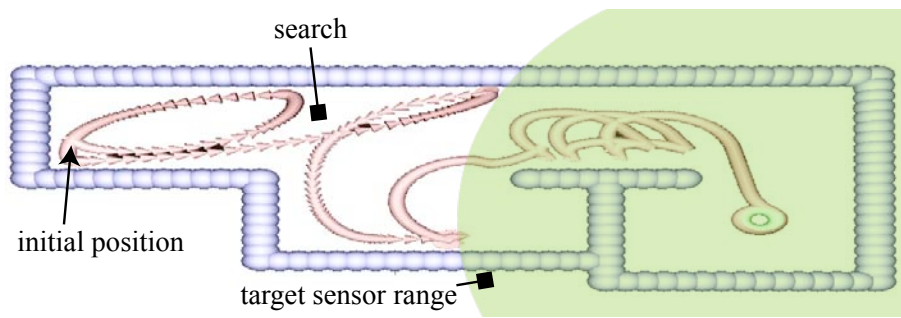


Fig. 65. An example of searching and taxis, both with obstacle avoidance.

## CHAPTER VII

### INTEGRATED CONTROL ARCHITECTURES FOR MULTI-AGENT SYSTEMS

#### A. Introduction

Having described and demonstrated the principles behind realizing an integrated control architecture for the single agent case in chapter VI, our goal now is to develop an integrated architecture for multi-agent systems. Our goal here is to realize a scheme for a set of homogeneous agents to engage in collective behavior [95] that would be useful for a robotic exploration or mobile sensor networks problem. Specifically, we want to engineering higher-order behaviors for:

- agents to navigate in the midst of other agents
- agents to flock together to a target
- agents to self-organize about that target

while using only passive interactions.

#### 1. The “nearest neighbor” agent sensor

We first endow the agents with faculties to perceive each other. Consider a local frame of reference attached to agent  $M$  and let  $\boldsymbol{\eta}_A = \begin{bmatrix} \eta_{A,1} \\ \eta_{A,2} \end{bmatrix}$  denote the position of the closest agent in this frame. For  $n$  agents within sensing range of  $M$  (illustrated in Figure 66), let  $\boldsymbol{l}_i, i \in \{1, \dots, n\}$  denote the displacement of between agent  $M_i$  and  $M$ . Then  $\boldsymbol{\eta} := \boldsymbol{l}_j$ , where

$$j = \underset{\forall i \in \{1, \dots, n\}}{\operatorname{argmin}} \|\boldsymbol{l}_i\| \quad (7.1)$$

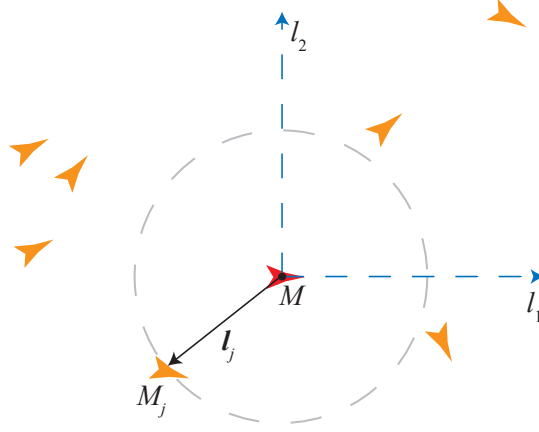


Fig. 66. The agent sensor of  $M$  returns a measurement of the displacement,  $\mathbf{l}_j$ , to the closest agent,  $M_j$ .

The agent sensor is a memoryless system that returns its measurement of the displacement to the closest agent,  $\mathbf{s}_A = \begin{bmatrix} s_{A,1} \\ s_{A,2} \end{bmatrix}$ :

$$\mathbf{s}_A = \boldsymbol{\sigma}(\boldsymbol{\eta}_A) := \begin{bmatrix} \sigma_1(\eta_{A,1}) \\ \sigma_2(\eta_{A,2}) \end{bmatrix} \quad (7.2)$$

where  $\sigma_1$  and  $\sigma_2$  are arbitrary measurement functions.

## 2. Outline

Before addressing the design of a fully integrated architecture, we first design a suite of higher-order behaviors using the elementary behaviors developed in chapter IV. In designing these, we illustrate three methods for realizing composite behaviors from elementary ones:

- time-division multiplexing of mutually exclusive primary behaviors
- superposition of complementary primary behaviors

- construction of composite sensory feedback

## B. Regulating inter-agent boundaries

By virtue of the agent sensor, agents are aware of the presence of other agents in their vicinity. This information can be used to realize a scheme of regulating inter-agent proximity. For example, in a multi-robot scheme, we desire agents to be within sensing range of each other, yet not so close as to cause collisions.<sup>1</sup> Moreover, in mobile sensor networks and robotic exploration, agents must be close enough to communicate, but also be far enough apart to ensure that the network achieves sufficient coverage over a territory.<sup>2</sup> We formulate this as the maintenance by each agent of “social boundaries.”

Consider Figure 67 which shows an agent,  $M_i$ , surrounded by two regions:

- the inclusion zone,  $Z_i := B(\mathbf{0}; r_{A,\max}) - B(\mathbf{0}; r_{A,\min})$ , within which  $M_i$  strives to maintain at least one other agent (e.g., agent  $M_j$  in the figure)
- the exclusion zone,  $Z_e := B(\mathbf{0}; r_{A,\min})$  within which  $M_i$  strives to ensure no other agent enters

To realize a composite behavior that strives to maintain these social boundaries, we can employ the elementary taxis and anti-taxis behaviors of chapter IV referenced to the agent sensor. Since these two behaviors are exclusive to each other, superposing them may result in actuation nulls where the antagonistic actions will sum to zero. Rather, we propose the scheme of Figure 68. The yellow box highlights an agent-based taxis scheme with obstacle avoidance (identical in structure to the target-based taxis

---

<sup>1</sup>The obstacle sensors may be unreliable in preventing collisions due to the dynamical nature of both agents. For example, two fast-moving agents close enough together may collide due to “sight” limitations in the obstacle sensors.

<sup>2</sup>Note that we map the *network* goal of maintaining connectivity with the *spatial* objective of ensuring an agent is nearby, and the network requirement of coverage to the spatial objective of ensuring agents are not too close.

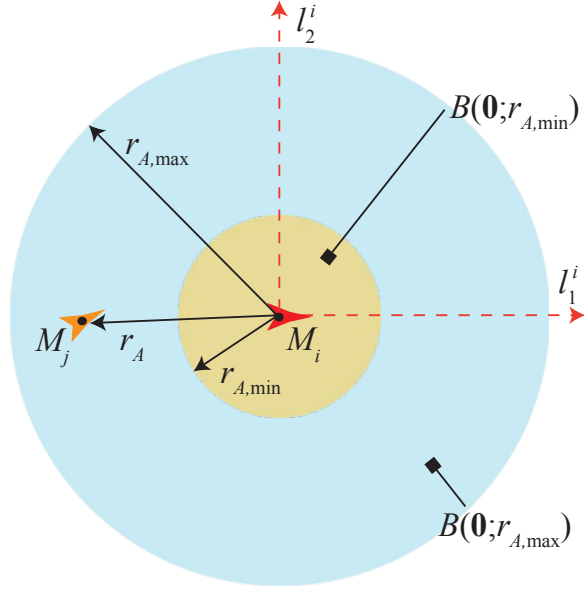


Fig. 67. The “social boundaries” of agent  $M_i$ : the inclusion zone,  $Z_i$  is shown in blue, while the exclusion zone,  $Z_e$ , is shown in yellow. Agent  $M_i$  senses  $M_j$  with respect to  $M_i$ 's local  $l_1^i - l_2^i$  coordinate system, and strives to maintain  $M_j$  within the inclusion zone (i.e., at a distance,  $r_A$ , where  $r_{A,\min} < r_A < r_{A,\max}$ ).

with obstacle avoidance scheme of chapter VI). We multiplex this with an agent-based anti-taxis behavior ( $R_{\bar{A}}$ ) and a null actuation (the  $\mathbf{0}$  block). The hysteresis function is defined such that it:

- selects taxis when the nearest agent goes sufficiently beyond  $Z_i$
- selects anti-taxis when the nearest agent enters  $Z_e$
- selects the null actuation  $a_v = 0, a_\omega = 0$  otherwise

### C. Flocking

Flocking is a mode of collective behavior where a mass of agents move together through the environment. We can realize a simple case of flocking by superposing a translational motion bias to the previous scheme; however, we want a more purposeful



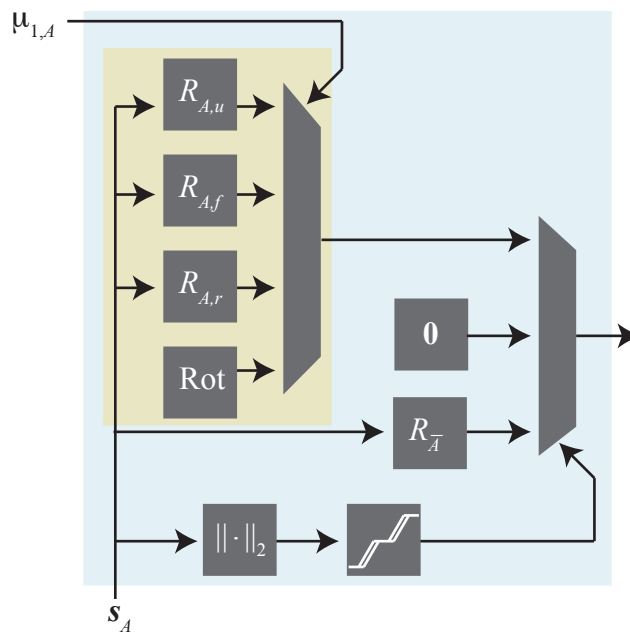


Fig. 68. A regulation scheme to maintain social boundaries.

behavior. To that end, we illustrate three architectures that enable a group of agents to flock to the target.

Suppose we wish to move a mass of agents from some point to a target location. Two behaviors will be at play in each agent:

- the primary behavior of target-based taxis<sup>3</sup>
- the secondary behavior of social boundary maintenance<sup>4</sup>

We have two options for composing these behaviors together: superposition of the behavior's actions and, as done in section B, multiplexing of the actions.

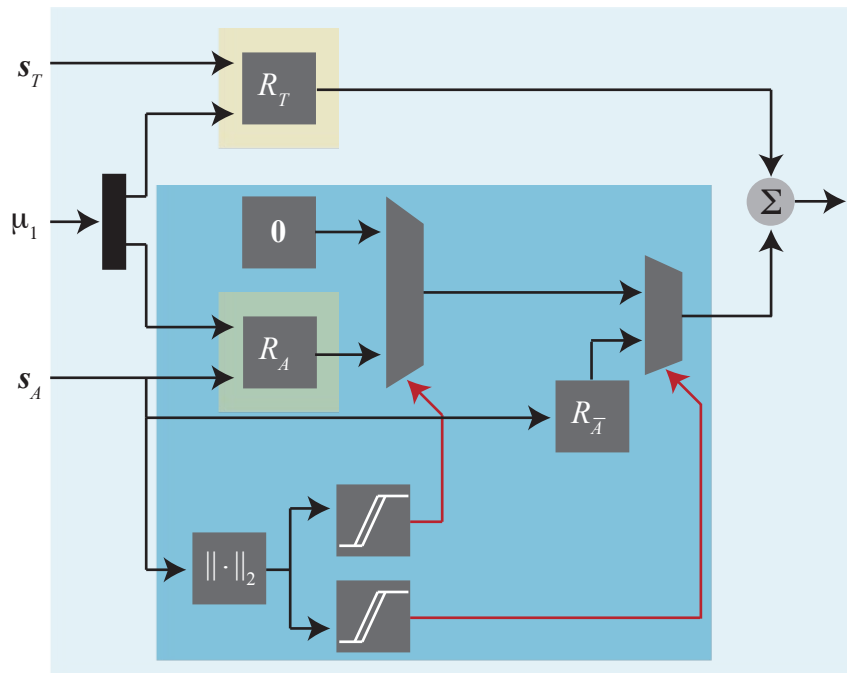


Fig. 69. A regulation scheme for flocking using action superposition. The yellow-highlighted  $R_T$  and  $R_A$  blocks realize target tracking and agent tracking, respectively. The social boundary regulator is shown highlighted in dark blue; the summation is done at the output of this regulator.

## 1. Action superposition

Consider the action superposition scheme of Figure 69 in which the outputs of the social boundary regulator and the target tracking regulator have been summed. Figure 70 shows a simulation where six agents flock to the target under this scheme. A major concern with superposition is that when the behaviors are not mutually exclusive, they can “fight” each other, that is, the sum of the actuation signals can result in a net output of zero. In the figure we can see that this does in fact happen as the agents get locked into a static configuration away from the target.

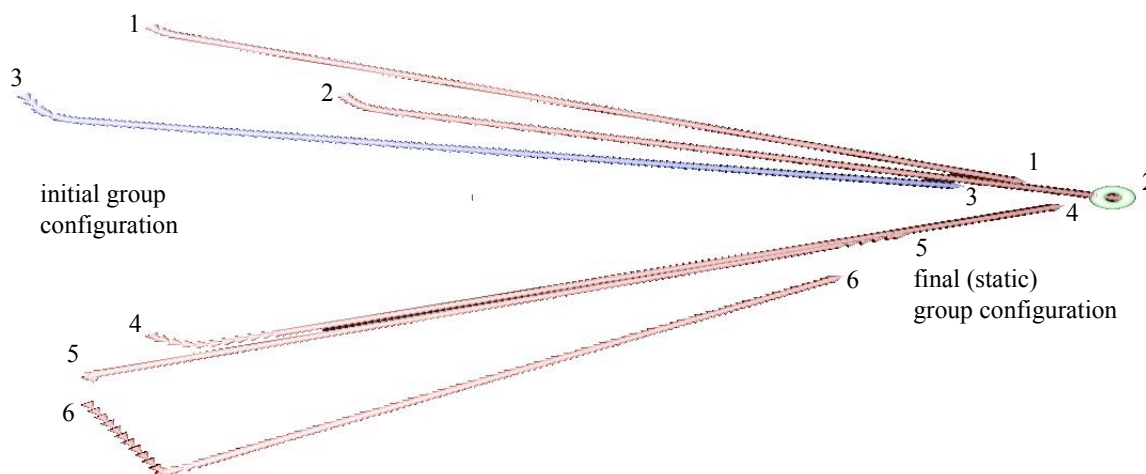


Fig. 70. The result of simulating six agents under the scheme of Figure 69.

Since target tracking and regulation of social boundaries are not necessarily exclusive to one another, we can propose the alternate superposition scheme of Figure 71. Here we “break” the social boundary regulator in two and superpose target tracking and agent tracking outputs. This then goes to the remaining half of the social boundary regulator that handles agent repulsion. Since the agent-repulsion

<sup>3</sup>For brevity, we will refer to target-based taxis as target tracking.

<sup>4</sup>Recall that this is composed of two primary behaviors: agent-based taxis (agent tracking) and agent-based anti-taxis (agent-repulsion).

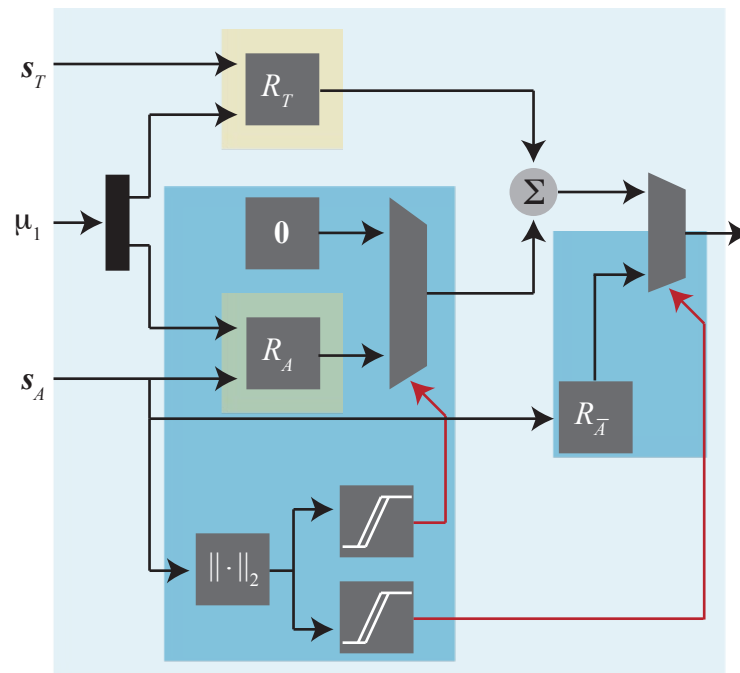


Fig. 71. A regulation scheme for flocking using action superposition. The social boundary regulator has been split into two (highlighted by the two dark blue boxes), and the superposition with target tracking is done in between the two halves.

regulator is closer to the output, it has higher-priority than either agent tracking or target tracking. Figure 72 shows the result of re-running the earlier simulation with the new scheme. Now, the agents “cloud” around the target, each getting close to it briefly, before breaking away (due to the switching of agent-repulsion behavior).

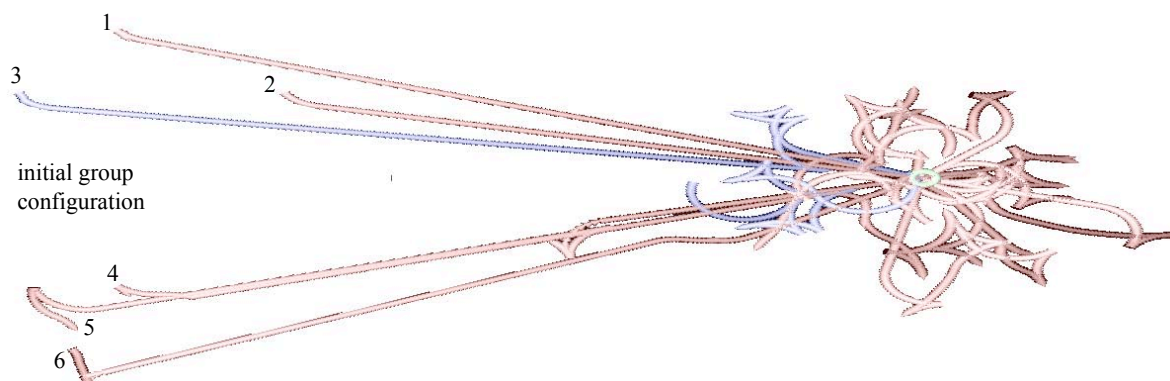


Fig. 72. The result of simulating six agents under the scheme of Figure 71.

## 2. Action multiplexing

The scheme of Figure 71 is still subject to actuation nulls when target tracking and agent tracking become antagonistic. Figure 73 illustrates an action multiplexing scheme that is inherently immune to null actuation and undesired equilibria away from the target, since the agent is always doing *something*. Figure 74 and 75 show two simulations of this scheme; in both, the agents cloud about the target.

### D. Self-organization: passive coordinated deployment

The action superposition scheme of Figure 69 had an appealing emergent characteristic: actuation nulls about the target led to the emergence to static formations of agents about the target. However, on closer inspection, the formation is not ideal since the coverage of the target is clearly biased, restricting the usefulness of the

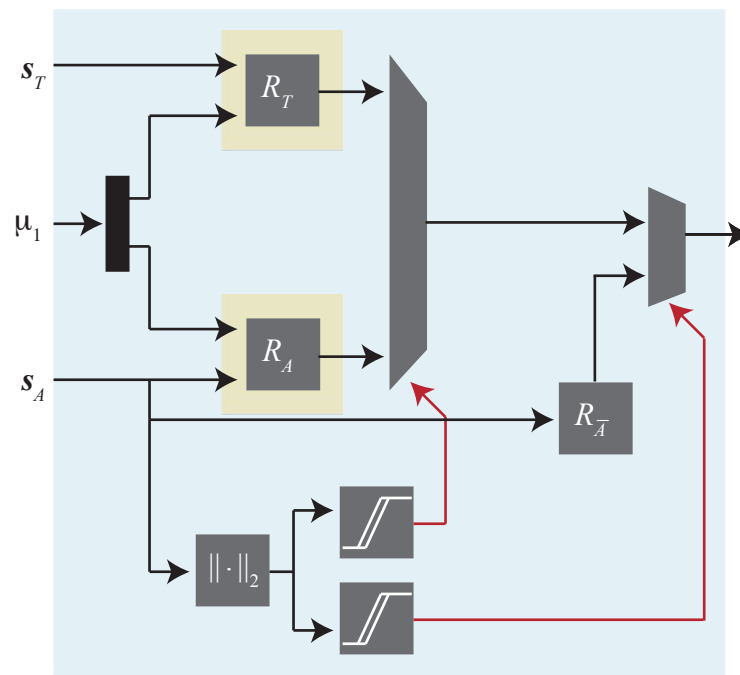


Fig. 73. A regulator for flocking; action multiplexing.

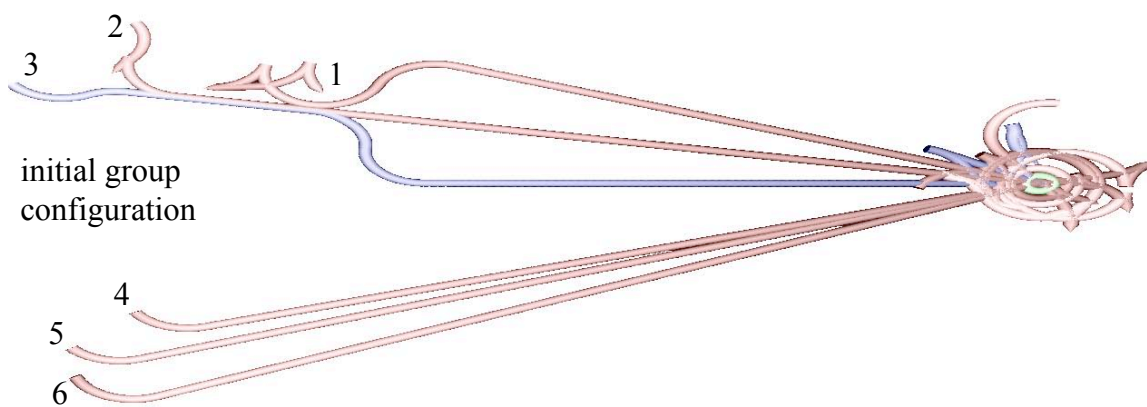


Fig. 74. The result of simulating six agents under the scheme of Figure 73.

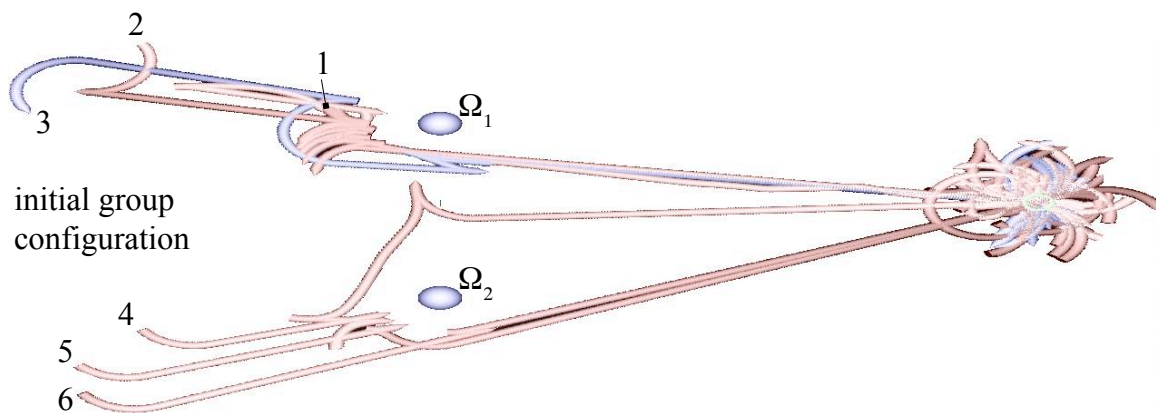


Fig. 75. The result of simulating six agents under the scheme of Figure 73 with obstacles.

scheme for mobile sensor networks or robotic exploration.

To rectify matters, we want to engineer the emergence of a more useful agent formation that better covers the target. Recall that under the assumptions of chapter III we are under a variety of constraints including:

- the lack of a global information pertaining to agent positions
- the lack of inter-agent communication faculties for active coordination between agents

Moreover, with the sensor model proposed at the beginning of this section, each agent only has access to the relative position of, at most, a single agent, the nearest neighbor. Hence, we can not appeal to the conventional multi-agent coordination- and formation-control schemes found in the literature [74, 75, 76, 77, 78] which generally require more sophisticated sensing.

To form a basis for a passive coordination scheme, we need some piece of common information that the distributed agents can use. To that end, we recognize that *every* agent performs a measurement (albeit a local one) of a common phenomenon: its displacement to the common target. Now, consider the ideal case where the agents

execute a regular simple  $n$ -sided polygonal<sup>5</sup> formation of agents centered about the target; Figure 76 illustrates the case for a hexagon. Let  $i$  denote any vertex of the polygon,  $M_i$  denote the agent at that vertex, and  $\mathbf{x}_{i,T}$  denote  $M_i$ 's displacement to the target. Let  $\Psi(\theta)$  be the two-dimensional rotation matrix:

$$\Psi(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (7.3)$$

and  $\theta_C > 0$  be half the measure of the interior angle of an  $n$ -sided regular polygon:

$$\theta_C = \frac{\pi}{2} \left( 1 - \frac{2}{n} \right) \quad (7.4)$$

We observe that the displacement from  $M_i$  to agent  $M_{i+1}$  (at the adjacent vertex, going clockwise) is co-linear with  $\Psi(\theta_C)\mathbf{x}_{i,T}$ , and similarly that the displacement from  $M_i$  to  $M_{i-1}$  is co-linear with  $\Psi(-\theta_C)\mathbf{x}_{i,T}$ .

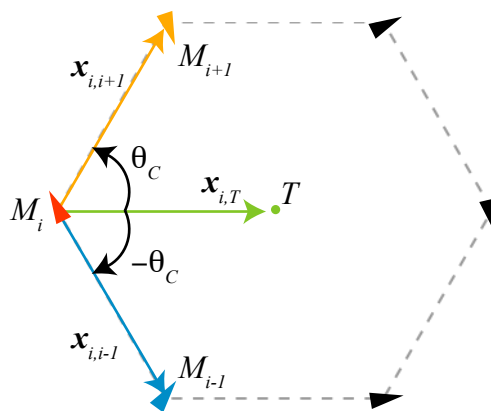


Fig. 76. The case of a hexagonal formation of agents about a target.

Hence, to give rise to a formation of agents,<sup>6</sup> we want to regulate our perception

<sup>5</sup>Recall that a regular polygon is one where all angles are congruent, and all sides have the same length; a simple polygon is one which does not intersect itself (in contrast to a star-like polygon).

<sup>6</sup>We note that so far we have specified an imprecise relationship between  $\mathbf{s}_T$  and



of the displacement to our nearest neighbor to be in compliance with this relationship between the target sensor and agent sensor measurements. To do so, we construct a composite sensory feedback signal for coordination,  $\mathbf{s}_C$ :

$$\mathbf{s}_C : (\mathbf{s}_T, \mathbf{s}_A) \mapsto \Psi(\theta_C)\mathbf{s}_T - \mathbf{s}_A \quad (7.5)$$

and use the taxis behavior to regulate  $\mathbf{s}_C$  to  $\mathbf{0}$ .

Now, will this work? Simulation results confirm that the use of this regulation scheme does *not* work (for the cases simulated)—even with ideal measurements of the displacements to the target and other agents. The reason behind this failure stems from the fact that there is only set of isolated configurations where the agents can come to rest: those configurations where  $\Psi(\theta_C)\mathbf{s}_T - \mathbf{s}_A$  goes to zero for *every* agent. It is unlikely that a system of multiple dynamic agents, where there is only very loose coupling between the agents (via a nearest neighbor agent sensor and a target sensor), will come to rest at precisely the right “sweet spot.” Moreover (7.5) is biased towards agents entering a relationship where the displacement between the target and the nearest neighbor is at  $\theta_C > 0$ —but this neglects the equally useful case of  $-\theta_C$ . Since agents do not have a global view of the current status of the formation, we can not obtain a basis for determining an appropriate sign to use locally.

We hence augment our scheme to create a band of acceptable orientations (about  $\pm\theta_C$ ) as illustrated in Figure 77:

$$\mathbf{s}_C = \begin{cases} \mathbf{0} & \text{for } ||\theta_{T,A}| - \theta_C| < \delta_C \\ \Psi(\theta_C)\mathbf{s}_T - \mathbf{s}_A & \text{otherwise} \end{cases} \quad (7.6)$$

---

$\mathbf{s}_A$ , that of co-linearity. To make this relationship an equality, we merely need invoke the law of cosines. However, since our agent and target sensors are only providing distorted (and likely noisy) measurements of  $\mathbf{x}_{i,T}$  and  $\mathbf{x}_{i,j}$  the utility of doing so is unjustified. Our goal here is to obtain a qualitative relationship that gives us a *principled* basis behind the engineering of *emergence* of a formation.

where:

$$\theta_{T,A} = \cos^{-1} \left( \frac{\mathbf{s}_T \cdot \mathbf{s}_A}{\|\mathbf{s}_T\|_2 \|\mathbf{s}_A\|_2} \right) \quad (7.7)$$

where  $0 < \delta_C < \theta_C$  specifies the width of this band.

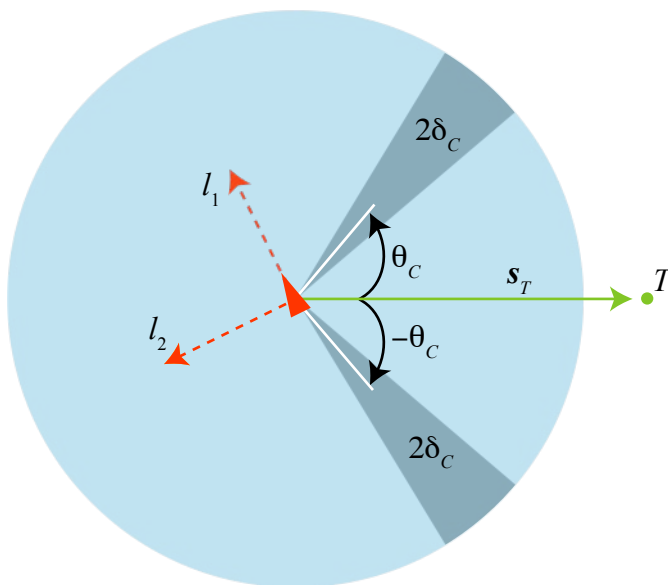


Fig. 77. Under the coordinated deployment behavior, the agent (shown as a red triangle at the origin of the  $l_1 - l_2$  coordinate system) attempts to regulate  $\mathbf{s}_A$  to within the grey regions (which are sectors of width  $2\delta_C > 0$  offset by  $\theta_C > 0$  with respect to  $\mathbf{s}_T$ ). The blue shaded region indicates the range of the agent sensor.

Figures 79-82 present simulation results for various configurations of agents operating under the control scheme of Figure 78.

### E. An integrated architecture

Figure 83 illustrates a layered control scheme that stitches together our regulators for flocking (which includes social boundary maintenance) and coordinated deployment; since these behaviors are mutually exclusive, multiplexors were used to compose these

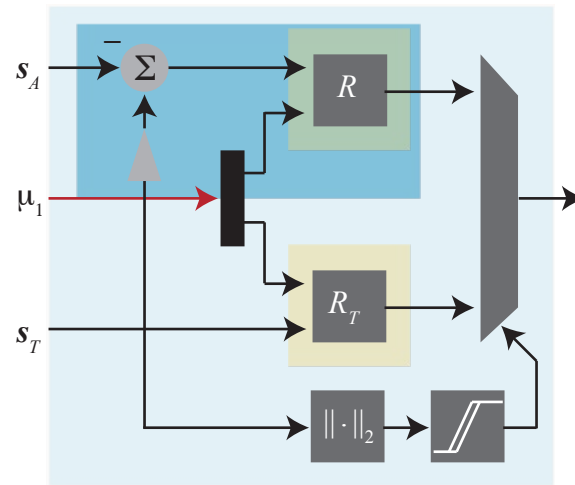


Fig. 78. A regulator for self-organization (highlighted in dark blue). A regulator for taxis is also instantiated to bring the agent to the target; once there, taxis will disengage, and self-organization will take over.

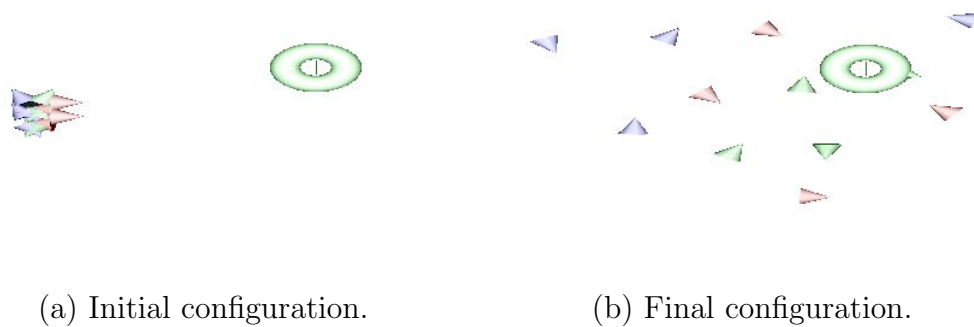


Fig. 79. Self-organization of a single group of twelve agents.



(a) Initial configuration.

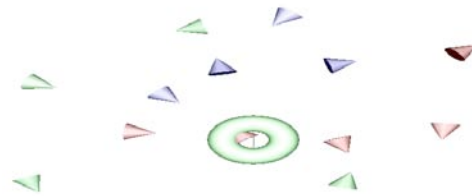


(b) Final configuration.

Fig. 80. Self-organization of twelve agents divided into two groups.

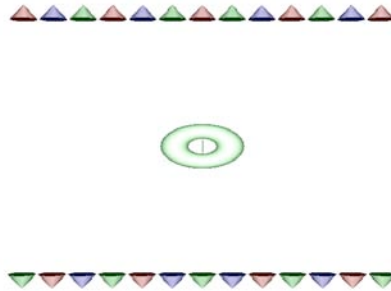


(a) Initial configuration.

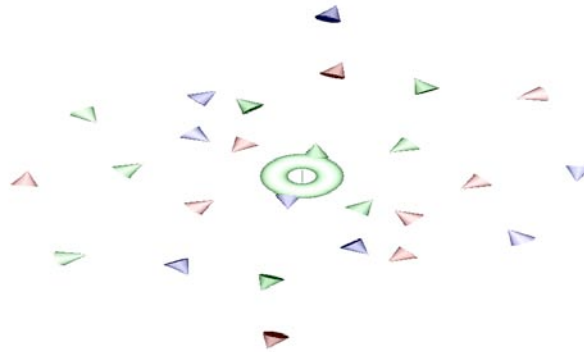


(b) Final configuration.

Fig. 81. Self-organization of one group of thirteen agents.



(a) Initial configuration.



(b) Final configuration.

Fig. 82. Self-organization of twenty six agents, divided into two groups. Note the emergence of a symmetric final formation. In the final configuration, all agents except for four are stable in a static formation; the four that are not stationary, oscillate about stationary positions.

behaviors.

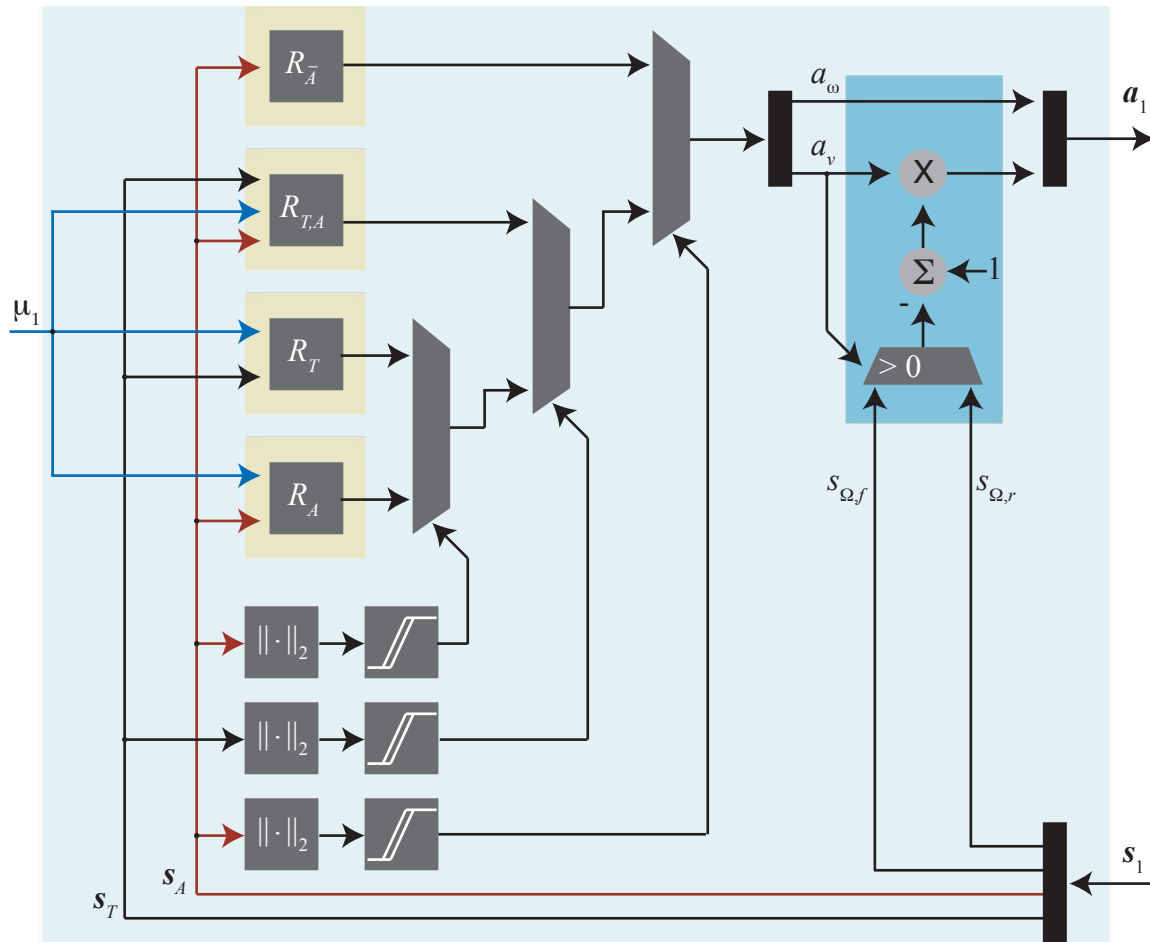


Fig. 83. An integrated controller for a multi-agent scheme that includes flocking, self-organization about a target waypoint, and obstacle avoidance.

Figure 84 show a time-lapsed view of a multi-agent simulation where twenty-six agents are divided into two groups. The first group arrives at the target and self-organizes; a second group arrives later, perturbing the organization by merging with the first group of agents. The overall group then continues to self-organize to cover the target, approximately uniformly.

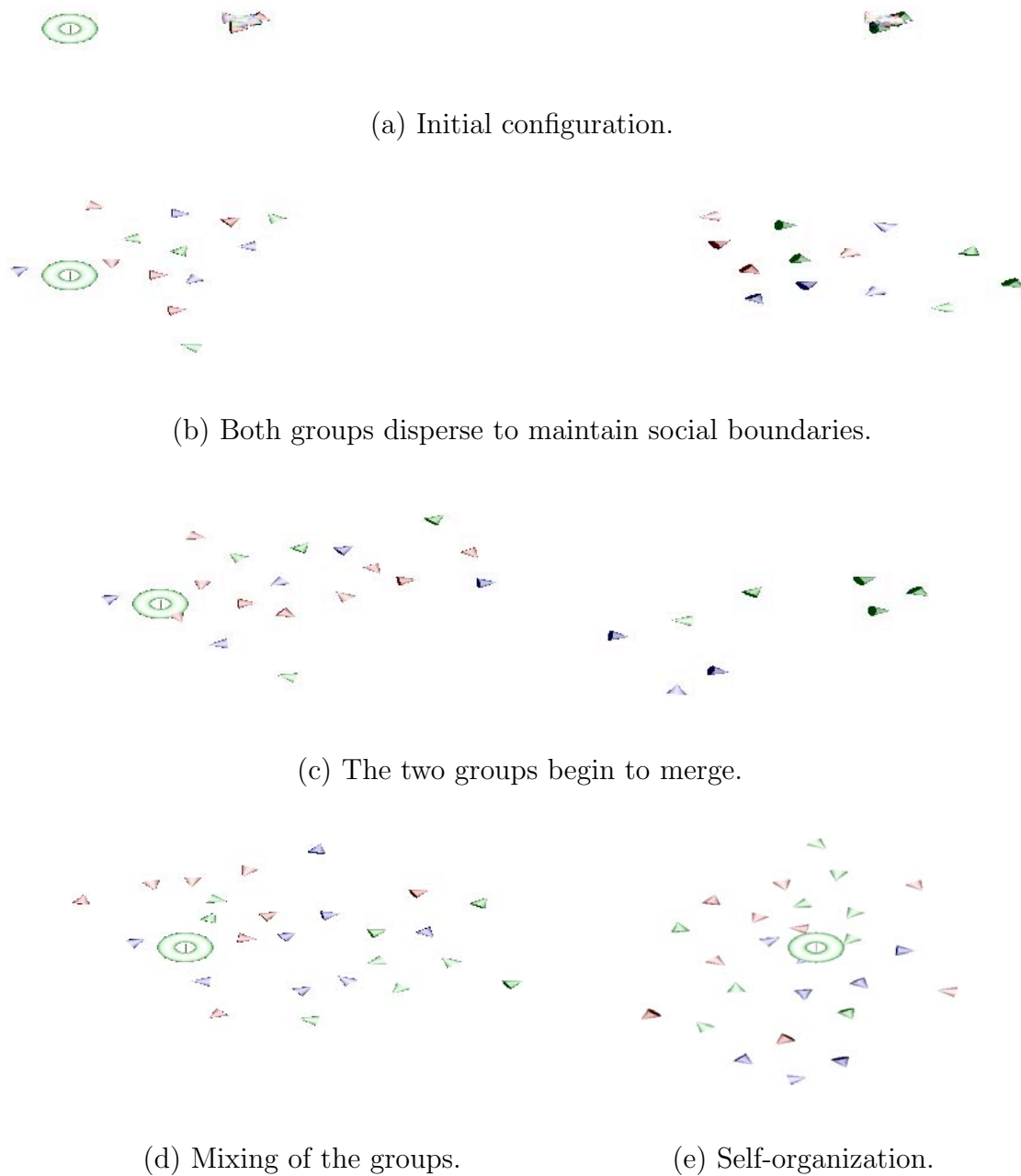


Fig. 84. Two groups of agents (twenty-six in total) flock to the target and self-organize about it. The final configuration is mostly stationary (i.e., most of the agents are at rest), with isolated subgroups of agents occasionally moving about to re-organize.

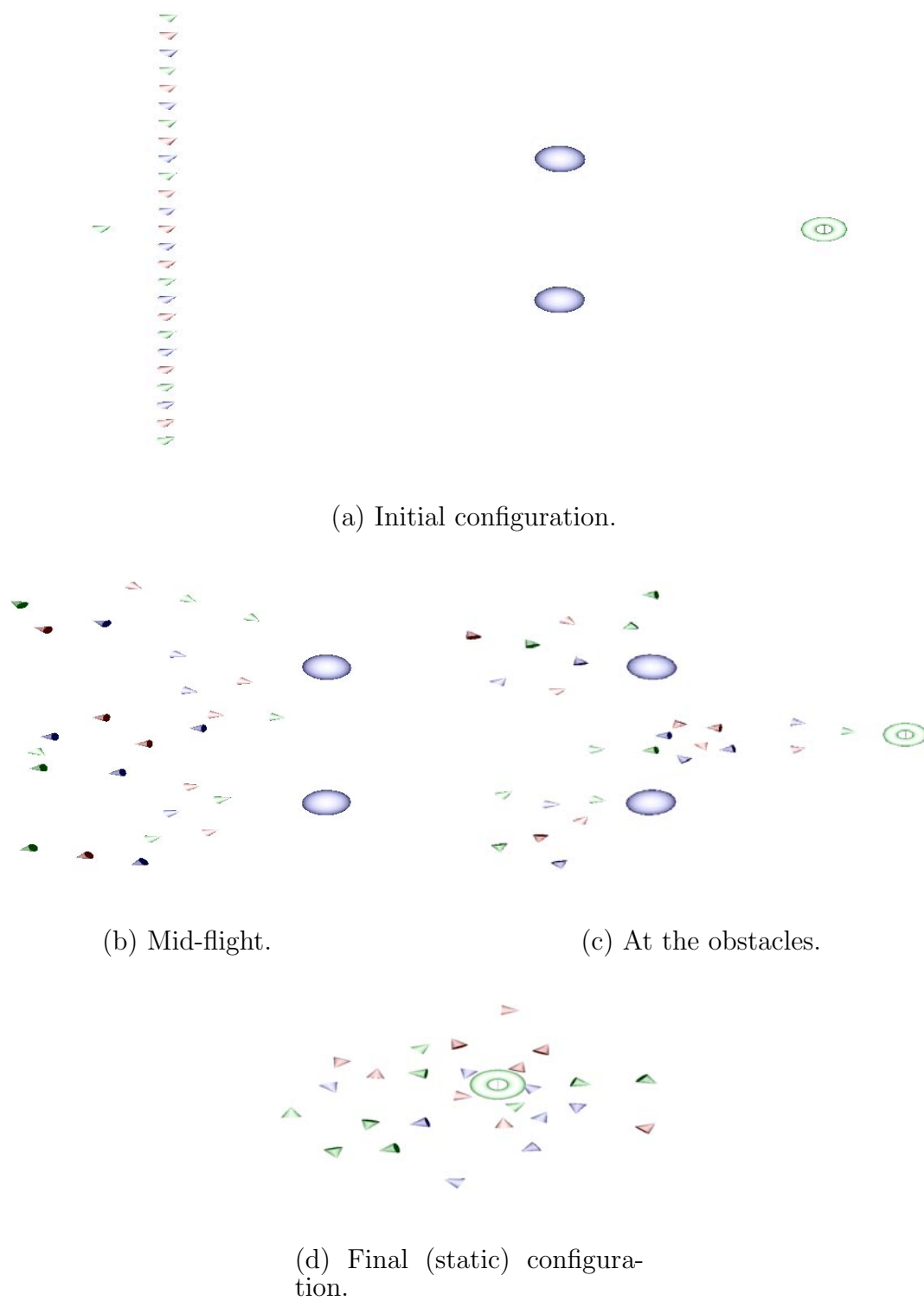


Fig. 85. A groups of twenty-six agents flock to the target and self-organize about it, navigating past two obstacles.



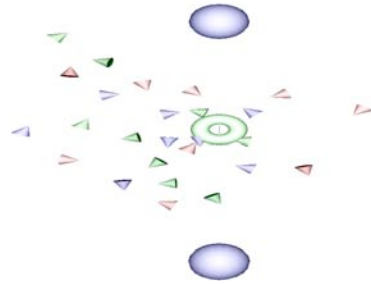


Fig. 86. This simulation uses the same initial group configuration as Figure 85, but with the obstacles repositioned near the target to constrain that area. The collective self-organizes into a configuration that is distorted by the obstacles.

## F. Discussion

In this chapter we presented the development of an integrated control architecture to enable groups of agents to function as a collective. Our approach was to employ the primary basis behaviors we designed in chapter IV in various behavioral composition schemes to realize useful secondary and tertiary behaviors.

We note that beyond providing a technology-independent formulation for robotic cognition, our scheme is:

- an agent-centric formulation
- very decentralized, each agent only knows of its *single* nearest neighbor (via an easy-to-realize sensing scheme)
- strongly homogeneous in terms of task allocation (i.e., there are no designated leaders or followers; any two agents can adopt identical behaviors)
- strongly homogeneous in terms of knowledge, that is, each agent is only aware of:
  - the relative displacement of the target

- the relative displacement of the single nearest neighbor
- the distance to the nearest point of an obstacle ahead of and behind the agent

and does not know of its position within the flock

We comment more on these points below.

## 1. Implementation of behaviors

We did not have to utilize many separate basis behaviors to design our scheme. In fact, we obtained a diversity of specialized behaviors from just one class of behavior—taxis—through:

- applying constraints to yield constrained taxis behaviors
- reversing the direction of flow to yield anti-taxis
- applying different sensory feedback signals to yield target tracking, agent tracking, and agent repulsion
- fusing target sensor and agent sensor outputs, and applying this composite signal to a taxis controller to yield coordinated deployment

By comparing our scheme with a related software-based one, we see the descriptive power of the cybernetic approach in being able to specify this diversity of behaviors with less design effort. For example, in [84] a similar repertoire of behaviors for collective multi-agent systems was developed via a software approach. In that work, six separate behaviors were specified by heuristic rules of the form:

```
if sensor-event-of-type-a occurs {
    do actuation-response-of-type-a
```

Table VII. Correspondence between the behaviors of a software-based scheme and our scheme.

behaviors of [84]	our construction
collision avoidance (obstacles)	primary behavior (taxis with speed attenuation)
collision avoidance (agents)	primary basis behavior (anti-taxis)
following	secondary behavior (social boundary regulation)
dispersion	secondary behavior (social boundary regulation)
aggregation	secondary behavior (social boundary regulation)
homing	primary basis behavior (taxis)
flocking	tertiary behavior (social regulation and taxis)

```

} else if sensor-event-of-type-b occurs {
    do actuation-response-of-type-b
} ...

```

Table VII illustrates the correspondence between our behaviors and those of [84]. Whereas the software approach utilized separate heuristic rules that were specified via `if-else` constructs, all of our behavioral repertoire were based on permutations of taxis,<sup>7</sup> specified as a closed-loop feedback law whose basic characteristics could be rigorously analyzed.

Hence, regardless of the technology-independence of a cybernetic formulation, the descriptive power of the toolset provides an appealing way of thinking about robotic control from a more agent-centric point-of-view. That is, whereas a heuristic approach requires the designer to think about behavioral design from the vantage

---

<sup>7</sup>That is, we only specified (“programmed”) one behavior—taxis—and through varying usage obtained our repertoire.

point of an observer (and often imbibing abstractions made by the observer), control-theoretic toolsets and dynamical systems theory bring the vantage point down to the basest of levels, that of the sensory feedback signals that are bombarding the agent. The problem then becomes one of *steering* the signals via an appropriate system. From this perspective, thus, we view the problem of robotic control for what it is to the agent: the regulation of a sensory *signal* from an undesirable characteristic to a desirable one.<sup>8</sup>

## 2. Flocking

Of relevance are the various leaderless<sup>9</sup> flocking schemes found in the literature of which the seminal work of Reynolds [83], as well as its descendants, stand as important representatives.

Reynolds' work was in the area of computer graphics (for which embodiment is generally not an issue), and formulated flocking as consisting of the composition of three primary behaviors (inter-agent collision avoidance, velocity matching of neighboring agents, staying close to neighboring agents). Two schemes for composition of new behaviors were also presented, weighted averaging and a priority-based scheme (to avoid the actuation nulls that can result from averaging). Matarić's work in [84] adapts Reynolds' formulation to a behavior-based robotics platform, giving rise to the flocking algorithm:

compute a weighted sum of the actuations from:

<collision avoidance>, <following>, <aggregation>, <dispersion>

---

<sup>8</sup>The designer still imparts something from the vantage point of an observer: the concept of what is desirable. However, the actions undertaken by the agent are *not* scripted from this perspective; they emerge from the agent's regulation of perception.

<sup>9</sup>For a discussion of non-homogeneous groups of agents that use leader-referenced schemes (among others), see [121].

```

if agent is at the front of the flock {
    slow down
} else if the agent is at the rear of the flock {
    speed up
}

```

where the behaviors enclosed in angular brackets correspond to those of table VII. Beyond the heuristic nature of the behavioral specification, this formulation differs from ours in that the agents of [84] can sense whether they are at the “front” or “rear” of the flock. Moreover, in determining the weighted sum, the distance to the centroid of the agent’s neighbors is needed. Both of these suggest a requirement for agent perception beyond that of our scheme. An element of Reynolds’ scheme that is absent in both Mataric’s and ours is the need for velocity information of an agent’s neighbors.

Olfati-Saber’s work [122] places the design of flocking agents on a very rigorous control-theoretic foundation, using Reynolds’ algorithm as a foundation. The approach employs a particle-based model to describe dynamical agents where, for agent  $i$ , the position,  $\mathbf{q}_i$ , and velocity,  $\mathbf{p}_i$ , are related to the actuation,  $\mathbf{a}_i$ , by:

$$\begin{aligned}\dot{\mathbf{q}}_i &= \mathbf{p}_i \\ \dot{\mathbf{p}}_i &= \mathbf{a}_i\end{aligned}\tag{7.8}$$

Two flocking schemes are presented, for free-flocking (in an obstacle-free environment) and flocking in the presence of obstacles; for both, the actuation signal includes the following terms:

$$\sum_{j \in N_i} f_1(\|\mathbf{q}_j - \mathbf{q}_i\|) \mathbf{n}_{ij} + \sum_{j \in N_i} f_2(\|\mathbf{q}_j - \mathbf{q}_i\|) (\mathbf{p}_j - \mathbf{p}_i)\tag{7.9}$$

where  $N_i$  denotes the indices of all agents that are neighbors of agent  $i$ . The first

term is based on computing a gradient, while the second aims for velocity consensus between agents.

The first point of difference with our scheme is the particle-based formulation of agents, which although capturing the dynamical attributes of agents, ignores vehicular kinematic restrictions that practical embodied agents must cope with. Beyond this, the computation of the actuation requires significant sensory feedback to measure the relative displacements and velocities of *each* local neighbor of an agent. Further, for the formulation of flocking with obstacle avoidance, the relative position of the obstacle (i.e., a vector quantity, as opposed to our scalar measurement) must be measured.

Now, the question arises: are the sensory requirements of these other schemes a problem? To be sure, they are forms of decentralized control. We note, however, that to perceive the relative displacements of a variably-sized population of neighboring agents, some form of vision or a radar-like system would be necessary<sup>10</sup> to image the surrounding environment. If only a population of constant size need be detected, then a simpler scheme using an onboard sensor array (with each array replicated for the number of agents, or time-multiplexed to detect each agent in sequence) could be used. For a lightweight agent, these sensing schemes are all costly prospects, in terms of either energy, space, or time (i.e., increased latency in processing the sensory data).

### 3. Self-organization

Control-theoretic formulations of the formation control problem strive to achieve faithful reproduction of a geometric formation, often invoking more sophisticated

---

<sup>10</sup>An alternative strategy would employ a triangulation scheme; however, this introduces a fixed global reference (i.e., a common coordinate system) for the system.

sensing schemes (as above, requiring the ability to sense several neighboring agents), which, although local, are generally difficult to realize.

In our approach, we instead design a system where the agents satisfice with a repertoire of *realizable* sensors. Perfect formations (that is, perfect geometric shapes) are not our goal; rather, we strive for the emergence of an approximately uniform coverage of the target, regardless of the specific geometric shape. Recall that our use of the term emergence is limited, appealing to Bedau's definition. That is, emergence is not simply a "magical" occurrence that we are happy to exploit when it occurs. Rather, a weakly emergent phenomenon is something that occurs due to a structural characteristic of the system (it is classified as emergent because we can not anticipate it arising purely by considering the often complex equations of motion of the system). Hence, we strive to engineer emergence by providing the raw materials for it: in this case, by endowing the agent with a regulator that fuses target and agent sensor data to provide the agent with guidance as to how to regulate both senses to achieve an acceptable formation about the target.

## CHAPTER VIII

### COMPARISONS WITH OTHER SCHEMES

#### A. Introduction

In this chapter we present some comparisons between the regulation-based agent control scheme advanced by our work, and those found in the literature.

Our philosophy here is to compare algorithms which are lightweight (i.e., of similar implementation complexity) and require similar sensory faculties. This choice was made to ensure comparisons would be as fair as possible, as well as to provide insight into how different algorithmic approaches (i.e., regulation versus heuristics) could be, potentially, combined in hybrid schemes.<sup>1</sup>

To ensure similar comparisons were made, when time constants were used for filters (in our analog scheme) or delays (in heuristic schemes) they were set equal across all techniques, vector summations (when performed) were normalized to unit vectors, and agent maximum speeds were set similarly. Beyond this, we also adopt performance metrics that do not look at absolute performance (i.e., optimal characteristics), but rather at robustness. Since this work is concerned with the design of behaviors for agents with bounded resources that ought to satisfice [5, 6] rather than optimize, such a characterization seems most appropriate.

In sections B and C, we present the comparison of robotic navigation schemes in different settings, namely:

- taxis in an obstacle-free environment, in which we limit the accuracy of the target sensor in an effort to ascertain the robustness of various schemes in the face of degraded perception

---

<sup>1</sup>This would be useful for future work.



- taxis with obstacle avoidance, in which we present the agent with progressively ‘difficult’ obstacle formations and ascertain its ability to circumnavigate these obstacles under the control of various algorithms

## B. Taxis in an obstacle-free environment

Here we consider the effect of limited resolution of target sensor directional information on the performance of various algorithms for taxis in obstacle-free environments. This is an important characterization, as the foundations of this work—the control schemes of chapter IV—are concerned with taxis behaviors. We address the more general case of taxis with obstacle avoidance in the next section.

Some of the applications that motivate the work of this dissertation are robotics exploration problems (e.g., search and rescue, space robotics, etc.) and mobile sensor networks. These applications generally demand lightweight agent designs (e.g., for reasons of agility in complex environments, economics, etc.), often precluding the use of expensive sensors with fine perceptual acuity. Practical sensors in this context, then, are often of limited accuracy and introduce distortion to measurements of environmental phenomenon. Further, in hostile environments it is possible for sensors to degrade (e.g., through injury to the robot); when such degradation is not catastrophic (i.e., when the sensor has not been totally destroyed, allowing some, possibly coarser, sensing), we desire graceful degradation of performance. Thus it is important to understand how a taxis<sup>2</sup> algorithm’s ability to track a target fares with sensor degradation.

---

<sup>2</sup>Which forms the basis of robotic navigation.

## 1. Constrained target sensing

We first describe how we model target sensors with constrained resolution. If the displacement from agent to target is  $\boldsymbol{\eta}$ , then the true direction to the target (referenced to the agent's local frame of reference),  $\theta$ , is given by:

$$\theta = \text{sgn}^+(\eta_2) \cos^{-1} \left( \frac{\eta_1}{\sqrt{\eta_1^2 + \eta_2^2}} \right) \quad (8.1)$$

where:

$$\text{sgn}^+(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ -1 & \text{for } x < 0 \end{cases} \quad (8.2)$$

To model the effect of limited target sensor resolution, let  $n$  denote the number of distinct directions that the target sensor can resolve.<sup>3</sup> Then  $\alpha(n) = \frac{2\pi}{n}$  is the angular spacing between these distinct directions. With this, we can define the coarse-resolution version of  $\theta$  as:

$$\theta'(n) = \text{floor} \left( \frac{|\theta| + \frac{\alpha(n)}{2}}{\alpha(n)} \right) \text{sgn}(\theta) \alpha(n) \quad (8.3)$$

Figure 87 illustrates the case for  $n \in \{3, 4, 5\}$ .

Figure 88 presents the target sensor model used for the simulation of all algorithms in this section, showing how  $Q$ , the block specified by (8.3), fits in the overall scheme.

## 2. Algorithms compared

We now provide an overview of the algorithms that we compare our unconstrained taxis scheme against. Three algorithms beyond our own were considered:

- a Braitenberg vehicle, representing perhaps the simplest class of taxis algorithm

---

<sup>3</sup>A high-resolution target sensor will have a correspondingly high number of distinct directions.

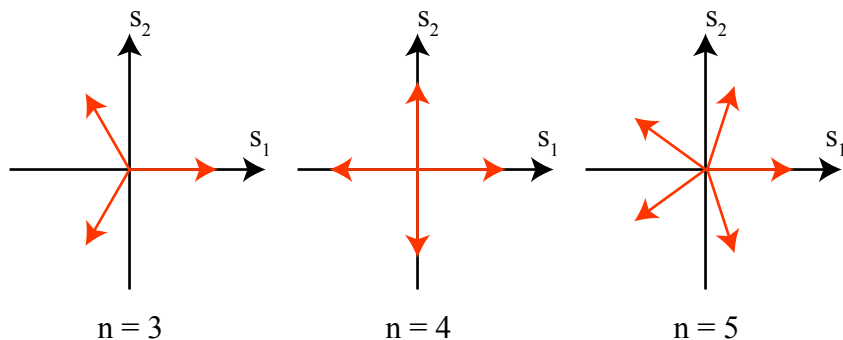


Fig. 87. Constrained target sensing. The target sensor takes the actual displacement to the target,  $\boldsymbol{\eta}$ , and returns a vector,  $\mathbf{s}$ , constrained to one of  $n$  directions (shown in red).

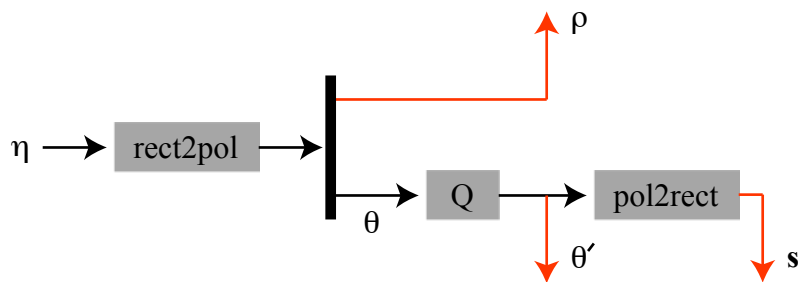


Fig. 88. The target sensor model used for the simulations of all algorithms in this section. The target sensor is a memoryless system that measures the displacement from the agent to the target,  $\boldsymbol{\eta}$ , constrains the direction (from  $\theta$  to  $\theta'$  through  $Q$ ), and returns a vector to the target that is either in polar form,  $(\rho, \theta')$ , or rectangular form,  $\mathbf{s}$ . The blocks “rect2pol” and “pol2rect” effect the conversions between rectangular and polar coordinates.

- a heuristic for taxis employed by Brooks, in his influential work on the subsumption architecture, and Mataric, in her behavior-based robotics work
- the virtual force field method of Borenstein and Koren, a lightweight potential field technique targeted to practical, fast-moving mobile robots

### a. Braitenberg vehicle 3a (Bra3a)

In the influential work of the biological cyberneticist Braitenberg [55], several thought experiments were conducted, resulting in a variety of virtual robotic “vehicles” exhibiting a diversity of behaviors. The origin of these thought experiments lie in his earlier study [54] which sought to understand how mappings from sensing to actuation (specifically, the decussations found in the neural topology of biological organisms) give rise to various behaviors such as taxis and kinesis.

In this section, we consider one such Braitenberg vehicle, vehicle “3a,” described in Figure 89, which exhibits taxis behavior. Beyond the simplicity of this scheme, we note that Braitenberg’s work, along with the earlier work of W. Grey Walter, influenced the reactive and behavior-based robotics paradigms. Hence, it is an important algorithm, from a historical perspective, to compare our scheme against.

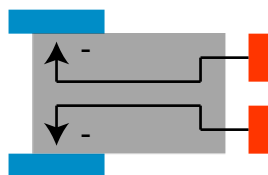


Fig. 89. Braitenberg vehicle 3a. Information from the target sensors (in red) are mapped directly to the motor actuators (in blue). The mapping is “inhibitory” so that when the target is near the sensor, the motors are actuated to a *lesser* degree than when the target is far from the sensor. The effect of this mapping is that the agent engages in taxis, coming to rest at the target.

In our implementation of this scheme, we re-cast the structural inhibitory mapping described by Figure 89 and [55] to the algebraic form:

$$\begin{aligned} v &= s_1 \\ \omega &= s_2 \end{aligned} \tag{8.4}$$

### b. Brooks-Matarić “homing” behavior (BroMat)

In Brooks’ seminal work [58] on the subsumption architecture, a simple “rotate-and-go” heuristic (described in Figure 90 using Brooks’ diagrammatic style) for differentially-driven vehicles was used to enable an agent to track a desired heading. Matarić presents the following heuristic for “homing” behavior [85] that more explicitly describes how taxis (in this case, tracking of “home”) occurs within such a scheme:

Home:

Whenever at home

stop

otherwise turn toward home, go.

Figure 91 presents a finite state machine that specifies our implementation of Brooks-Matarić homing behavior, with the motor control outputs for each state being given by:

$$\begin{aligned} \text{turn} &: \left\{ \begin{array}{l} v = 0 \\ \omega = \kappa_\omega \text{sgn}(\theta') \end{array} \right. \\ \text{fw} &: \left\{ \begin{array}{l} v = \kappa_v \\ \omega = 0 \end{array} \right. \\ \text{stop} &: \left\{ \begin{array}{l} v = 0 \\ \omega = 0 \end{array} \right. \end{aligned} \tag{8.5}$$

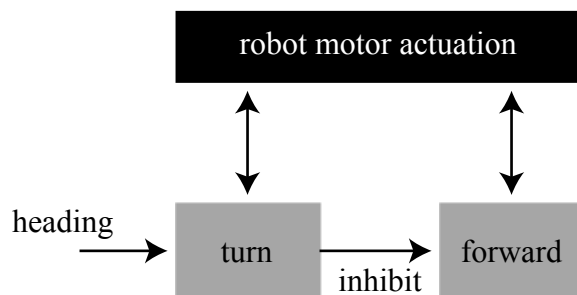


Fig. 90. The underlying “rotate-and-go” motion controller used in Brooks’ subsumption scheme. The “turn” module is first engaged to cause the agent to align with a desired heading; while rotating, the “forward” motion block is inhibited. Upon aligning with the commanded heading, the forward block is engaged, and the agent moves forwards.

where  $\kappa_v > 0$  and  $\kappa_\omega > 0$  are scaling factors that will be described later.

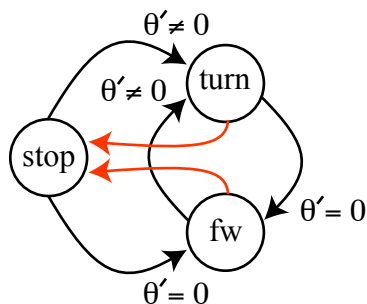


Fig. 91. Our implementation of Brooks-Matarić homing behavior. The red arrows indicate transitions caused by arrival at the target.

### c. Borenstein-Koren virtual force field (BorKor)

Potential field methods have their origin in the work of Krogh [123] and Khatib [124] on obstacle avoidance for robotic manipulators and mobile robots. Work in the mobile robotics community by Brooks and Arkin adapted potential field methods (which up to that point had *not* been used on physically-realized mobile robotic agents, due to

the requirements for global knowledge of the environment [125]) for use on mobile robots, leading to the navigation algorithms of [58] (using attractive and repulsive forces to guide the agent), and [59] (using directional fields called schemas that are instantiated in response to environmental stimuli).

The work of Borenstein and Koren in [125], inspired by the potential field work of Khatib and Krogh as well as the practical work of Brooks and Arkin, developed the *virtual force field* method to facilitate “fast, continuous, and smooth motion of the controlled vehicle” [125]. The technique uses virtual forces to obtain heading information that is used to derive a steering rate command—similar to Arkin [59]—but adds preprocessing of sensory data to make the algorithm less sensitive to sensor fluctuations.<sup>4</sup>

In this method a heading vector<sup>5</sup> is used to modulate the steering rate command,  $\omega$ , while the agent translates forwards. Hence, this method is a “continuous” version of Brooks-Matarić homing in which translation and rotation occur concurrently;<sup>6</sup> Figure 92 illustrates the scheme using conventions similar to those of [58] and Figure 90.

In our implementation of this scheme for taxis we use the following law:

$$\begin{aligned} v &= \kappa_v \tanh(\rho) \\ \omega &= \kappa_\omega [(-)\theta'] \end{aligned} \tag{8.6}$$

where, as in [125],  $(-)$  denotes an operator that returns the shortest rotational dis-

---

<sup>4</sup>In a simulated environment, the preprocessing scheme is not relevant since idealized sensor information can be used.

<sup>5</sup>For the purposes of this section, discussing taxis in an obstacle-free environment, the heading vector is an *attractive* vector directed from the agent *to* the target. The more general case of taxis and obstacle avoidance will be discussed in the next section.

<sup>6</sup>In the presence of obstacles, the translational speed is allowed to drop to prevent collisions.

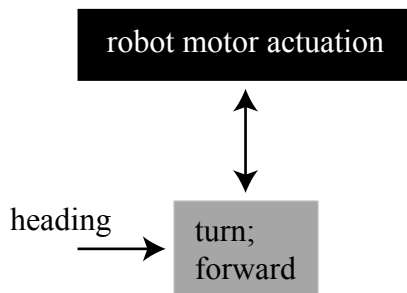


Fig. 92. The underlying motion controller used in the scheme of Borenstein and Koren.

placement between the agent's current direction of motion and the measured heading to the target,  $\theta'$ .

#### d. Unconstrained taxis (uc)

The development of this scheme was presented in chapter IV. To recapitulate, the foundation for this behavior is the regulation of the agent's sensory perception of the target,  $\mathbf{s}$ . We describe perception using the plant model:

$$P : \dot{\boldsymbol{\eta}} = \mathbf{p}(\boldsymbol{\eta}, \mathbf{a}) := \Upsilon(\boldsymbol{\eta})\mathbf{a} \quad (8.7)$$

where:

$$\Upsilon(\boldsymbol{\eta}) = \begin{bmatrix} -1 & \eta_2 \\ 0 & -\eta_1 \end{bmatrix} \quad (8.8)$$

and  $\mathbf{s}(\boldsymbol{\eta})$  is specified by Figure 88. From this, we derive a controller (using vector field design) to stabilize  $\boldsymbol{\eta} = \mathbf{0}$ , leading to the actuation law:

$$\begin{aligned} v &= \text{sgn}(s_1) \\ \omega &= \text{sgn}^+(s_1)\text{sgn}(s_2) \end{aligned} \quad (8.9)$$



Table VIII. Five different initial conditions for the agent were used in the comparison of taxis algorithms.

$d$	$\psi$ (radians)
4	0
4	$\frac{3}{4}\pi$
4	$\frac{4}{3}\pi$
8	$\frac{4}{5}\pi$
8	$\frac{3}{2}\pi$

### 3. Methodology and performance metrics

To compare these algorithms, the agent was placed at various initial conditions away from the target and, for each initial condition, was put under the control of each algorithm; Table VIII summarizes the initial conditions we consider with reference to the setup illustrated in Figure 93. A sweep of the parameter  $n$  (the number of distinct directions that could be resolved by the target sensor) from  $n = 3$  to  $n = 42$  was conducted for all combinations of initial condition and algorithm<sup>7</sup> and the length of the path taken by the agent from its initial position to a region of radius 0.3 units about the target was measured from the simulation data. For all 800 runs, the agents had the same maximum translational and rotational speeds (i.e.,  $\kappa_v = 5$  and  $\kappa_\omega = 5$ ); the only differences between runs were in the initial condition of the agent, the algorithm employed, and the value of  $n$ —with these parameters being varied *one* at a time.

In evaluating the *robustness* of the taxis algorithms to target sensor degradation,

---

<sup>7</sup>Hence, in total, the results of this section required  $5 \times 4 \times 40 = 800$  simulation runs.

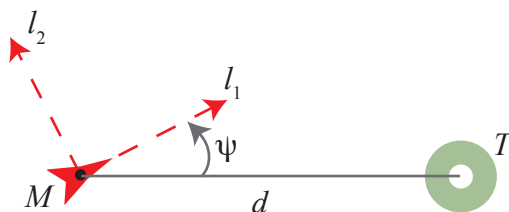


Fig. 93. The agent was placed  $d$  units away from the target, with an orientation of  $\psi$  radians with respect to the target.

we note that absolute path length is *not* as important as how the path length degrades as  $n$  is decreased.<sup>8</sup> Hence, our metric of robustness for a given algorithm is the standard deviation of the path lengths produced by that algorithm for  $n = 3$  to  $n = 42$ , with lower standard deviations indicating a more robust algorithm.

#### 4. Results

Figure 94 presents a plot of path length versus target sensor resolution ( $n$ ) for the four algorithms considered, when the agent is started with  $d = 4$  and  $\psi = 0$ . Pointed directly at the target, the agent under control of all schemes takes the shortest path to the target, for all  $n$ .

Beyond this degenerate, baseline case (summarized in Table IX), Figure 95 presents simulation results for the case  $d = 4$  and  $\psi = \frac{3}{4}\pi$ . The robustness of the various schemes to degraded target sensor resolution can be ascertained, qualitatively, by looking at how the path length increases as  $n$  is decreased. For example, although BroMat starts off with lower path lengths than BorKor, it degrades to a greater degree indicating less insensitivity to target sensor resolution. For sufficiently

---

<sup>8</sup>Recall, with mobile robotic agents possessing only local knowledge of the world, we are not concerned with optimality, but rather with how the agent is able to satisfice to achieve tolerable results. In this context, considering absolute path lengths is inappropriate.

high resolution (i.e., for  $n > 8$ ) we note that both BorKor and Bra3a exhibit a great deal of insensitivity to sensor degradation, whereas BroMat can be seen to degrade steadily as  $n$  is decreased from  $n = 42$ . For low sensor resolution, Bra3a degrades the worst, followed by BroMat, and then BorKor. Although the unconstrained taxis method yielded the lowest path length, we do not consider that as significant as the robustness it exhibits to degraded (i.e., lower-resolution) target sensor information over the other methods—path length stays constant in spite of  $n$  being decreased.

Figure 96 and Table XI present the case for  $d = 4$  and  $\psi = \frac{4}{3}\pi$ . Although uc performs better than the other methods (achieving a smaller standard deviation in path length), it does exhibit a sensitivity for the case  $n = 4$ . For that case, constrained target sensor directional information induced the agent to take a less optimal path towards the target in which it moved forwards; for all other values of  $n$ , the agent took a shorter route that involved it backing up to the target.

The results of Figure 97 and Table XII (for  $d = 8$  and  $\psi = \frac{4}{5}\pi$ ), exhibit similar trends as those for the case  $d = 4$  and  $\psi = \frac{3}{4}\pi$ . As well, the results of Figure 98 and Table XIII (for  $d = 8$  and  $\psi = \frac{3}{2}\pi$ ), exhibit similar trends as those for the case  $d = 4$  and  $\psi = \frac{4}{3}\pi$ . We note however, that uc for this case generally executes *longer* path lengths to the target; however, the standard deviation is still lower for uc.

### C. Navigation: taxis with obstacle avoidance

Here we evaluate the ability of our scheme to circumnavigate a variety of obstacle formations, and compare it against other approaches.

#### 1. Algorithms compared

The approach for navigation we use is the constrained regulation scheme for obstacle avoidance presented in Figure 59 of chapter VI. We compare its performance with

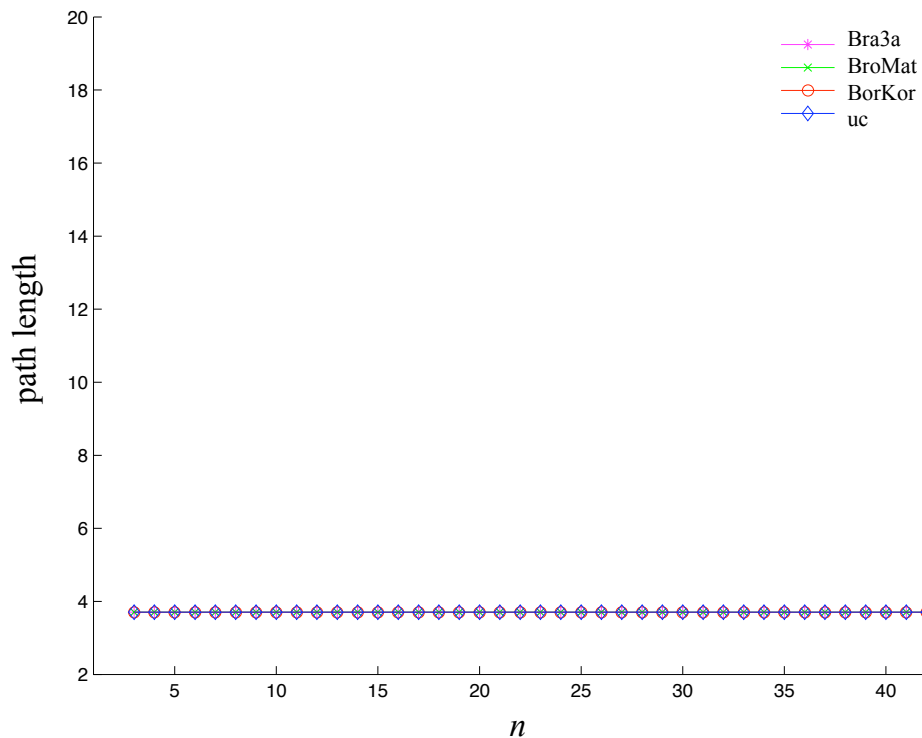


Fig. 94. Path length versus the number of distinct target sensor directions ( $n$ ) for the Braitenberg vehicle 3a (Bra3a), Brooks-Mataric homing (BroMat), Borenstein-Koren virtual force field (BorKor), and unconstrained taxis (uc) algorithms;  $d = 4$  and  $\psi = 0$ .

Table IX. Summary of algorithm performance characteristics for obstacle-free taxis:  $d = 4$ ,  $\psi = 0$ .

algorithm	path length		
	minimum	maximum	standard deviation
Bra3a	3.70	3.70	0.00
BroMat	3.70	3.70	0.00
BorKor	3.70	3.70	0.00
<b>uc</b>	3.70	3.70	0.00

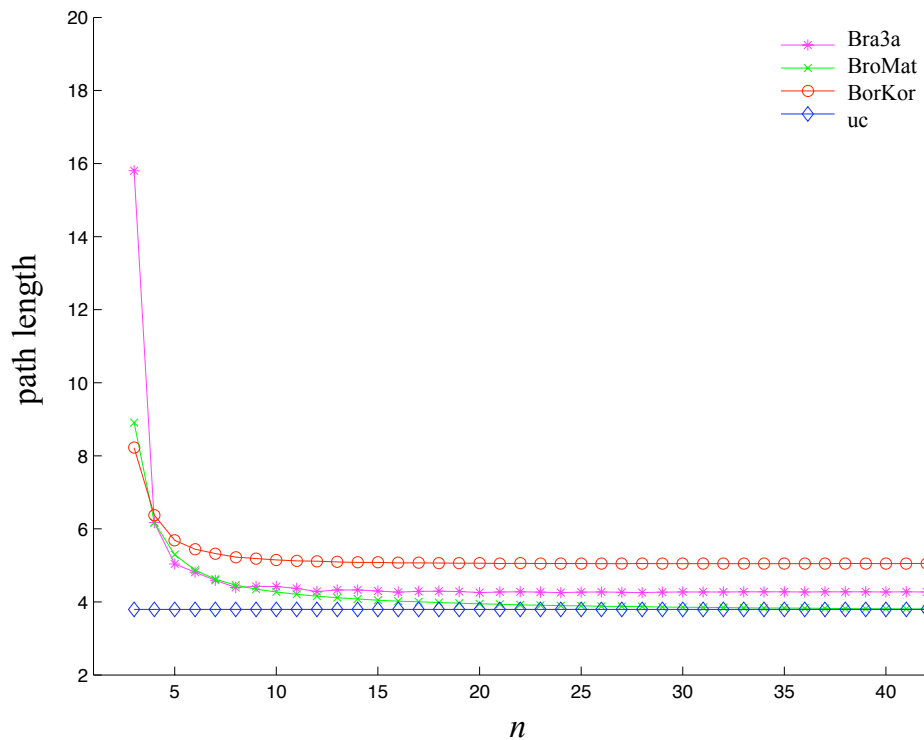


Fig. 95. Path length versus the number of distinct target sensor directions ( $n$ ) for the Braitenberg vehicle 3a (Bra3a), Brooks-Mataric homing (BroMat), Borenstein-Koren virtual force field (BorKor), and unconstrained taxis (uc) algorithms;  $d = 4$  and  $\psi = \frac{3}{4}\pi$ . Note the high degree of insensitivity to  $n$  of unconstrained taxis.

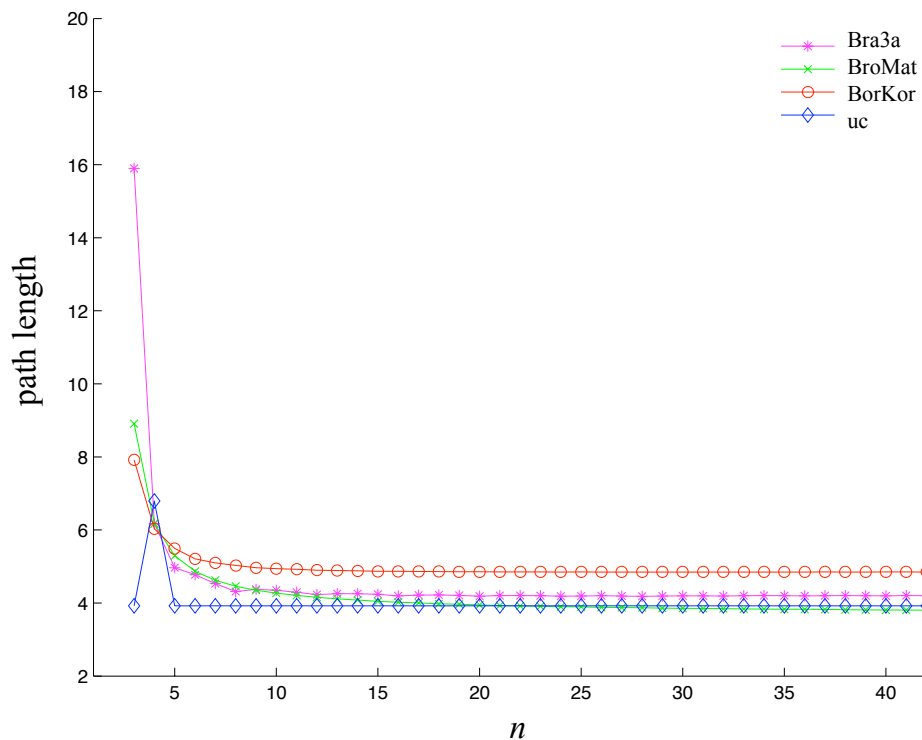


Fig. 96. Path length versus the number of distinct target sensor directions ( $n$ ) for the Braitenberg vehicle 3a (Bra3a), Brooks-Matarić homing (BroMat), Borenstein-Koren virtual force field (BorKor), and unconstrained taxis (uc) algorithms;  $d = 4$  and  $\psi = \frac{4}{3}\pi$ .

Table X. Summary of algorithm performance characteristics for obstacle-free taxis:  $d = 4$ ,  $\psi = \frac{3}{4}\pi$ .

algorithm	path length		
	minimum	maximum	standard deviation
Bra3a	4.25	15.80	1.84
BroMat	3.81	8.91	0.89
BorKor	5.05	8.22	0.54
<b>uc</b>	3.79	3.79	0.00

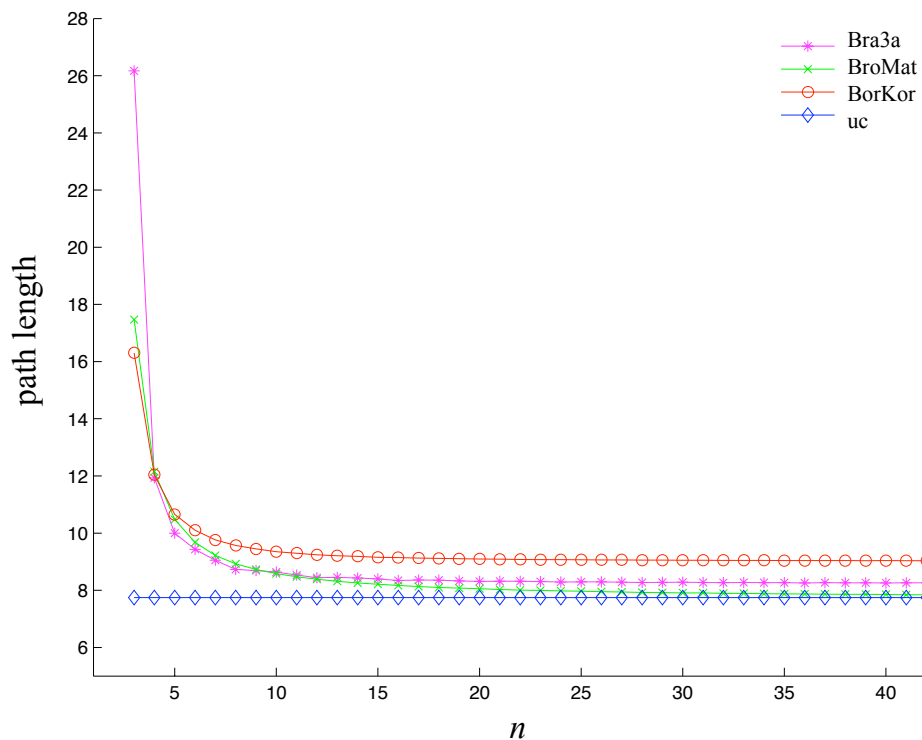


Fig. 97. Path length versus the number of distinct target sensor directions ( $n$ ) for the Braitenberg vehicle 3a (Bra3a), Brooks-Matarić homing (BroMat), Borenstein-Koren virtual force field (BorKor), and unconstrained taxis (uc) algorithms;  $d = 8$  and  $\psi = \frac{4}{5}\pi$ .

Table XI. Summary of algorithm performance characteristics for obstacle-free taxis:  $d = 4$ ,  $\psi = \frac{4}{3}\pi$ .

algorithm	path length		
	minimum	maximum	standard deviation
Bra3a	4.18	15.90	1.86
BroMat	3.80	8.91	0.89
BorKor	4.85	7.92	0.52
<b>uc</b>	3.92	6.79	0.45

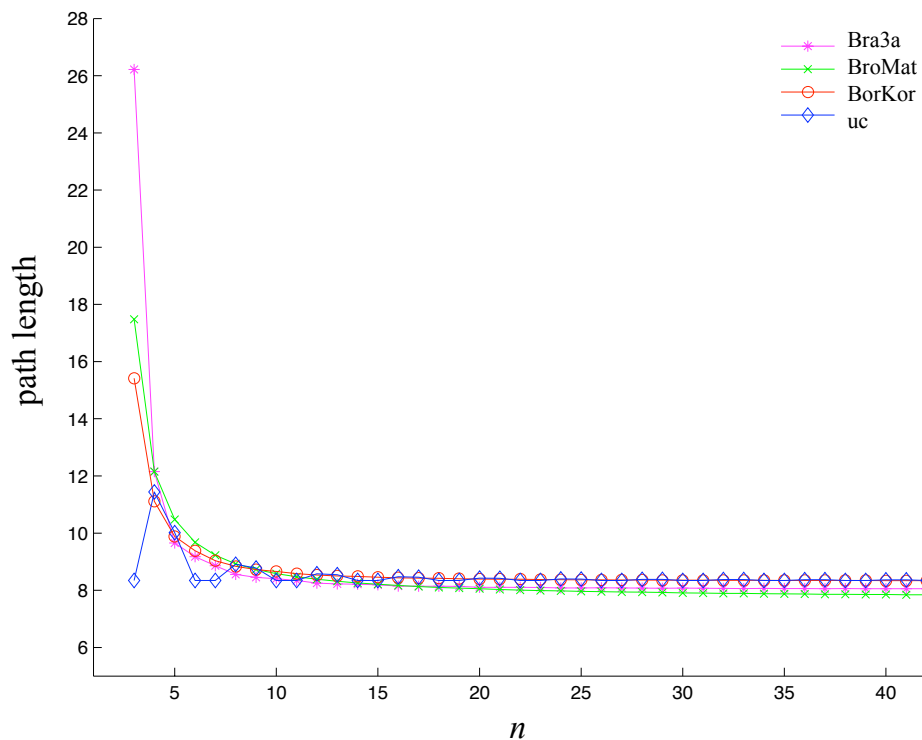


Fig. 98. Path length versus the number of distinct target sensor directions ( $n$ ) for the Braitenberg vehicle 3a (Bra3a), Brooks-Matarić homing (BroMat), Borenstein-Koren virtual force field (BorKor), and unconstrained taxis (uc) algorithms;  $d = 8$  and  $\psi = \frac{3}{2}\pi$ .

Table XII. Summary of algorithm performance characteristics for obstacle-free taxis:  $d = 8$ ,  $\psi = \frac{4}{5}\pi$ .

algorithm	path length		
	minimum	maximum	standard deviation
Bra3a	8.26	26.17	2.86
BroMat	7.84	17.47	1.67
BorKor	9.04	16.30	1.24
<b>uc</b>	7.74	7.74	0.00



Table XIII. Summary of algorithm performance characteristics for obstacle-free taxis:

$$d = 8, \psi = \frac{3}{2}\pi.$$

algorithm	path length		
	minimum	maximum	standard deviation
Bra3a	8.05	26.22	2.91
BroMat	7.84	17.48	1.67
BorKor	8.34	15.42	1.20
<b>uc</b>	8.34	11.44	0.55

respect to three schemes found in the literature:

- Braitenberg vehicle “3c” of [55]
- Matarić’s “avoid-everything-else” behavior of [85]
- a virtual force field method that uses the core of Borenstein and Koren [125, 126] and mates it with Arkin’s “noise schema instantiation” [59] technique (to help solve actuation nulls caused when obstacle sensor and target sensor outputs are superposed)

#### a. Braitenberg vehicle 3c (Bra3c)

Braitenberg presents a “multisensorial vehicle” [55]—vehicle “3c”—whose behavioral traits include taxis towards desirable stimuli as well as avoidance of undesirable stimuli. Figure 99 illustrates a version of this vehicle with two sensory inputs.

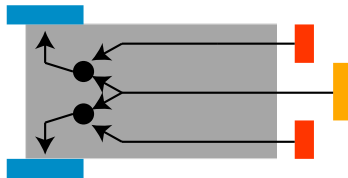


Fig. 99. A reactive scheme for obstacle avoidance based on Braitenberg vehicle “3c.” The vehicle engages in taxis when the obstacle sensor (in orange) is not stimulated; when stimulated by an obstacle, the vehicle stops and rotates until the stimulation ceases.

We implement this vehicle with the actuation law:

$$\begin{aligned} v &= \begin{cases} s_1 & \text{no obstacle in sight} \\ 0 & \text{otherwise} \end{cases} \\ \omega &= \begin{cases} s_2 & \text{no obstacle in sight} \\ \kappa_\omega & \text{otherwise} \end{cases} \end{aligned} \quad (8.10)$$

#### b. Matarić’s “avoid-everything-else” behavior (Mat)

In [85], Matarić specifies a heuristic for collision avoidance where the agent turns when confronted with an obstacle and backs up before proceeding forwards.

We implement this behavior and integrate it with a homing behavior using the finite state machine of Figure 100. We note that the timeout for backing off was matched to the obstacle overstimulation filter time constants used in our scheme.

#### c. A dithered virtual force field method (BorKor-d)

Here we adopt the lightweight potential field method of [125, 126], where the target exerts an attractive virtual force on the agent (i.e., directed from the agent to the target) and an obstacle exerts a repulsive one (i.e., directed from the obstacle to the agent). The agent translates forwards until it reaches the target, with the steering

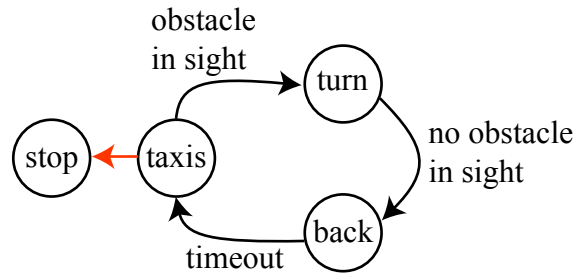


Fig. 100. Our implementation Matarić’s avoid-everything-else behavior. The red arrow indicates a transition caused by arrival at the target.

command being derived from the vector sum of the two forces; to preclude null actuation (i.e., where the repulsive and attractive forces cancel, leaving the agent “stuck” at an obstacle) a random dithering signal (Arkin’s noise “schema instantiation” of [59]) is added to the vector sum.

## 2. Methodology

We compare obstacle avoidance of the above algorithms in the face of various obstacle formations. To make as direct a comparison as possible between pure obstacle avoidance, we limit the obstacle sensor of this section to the monocular setup described in chapters IV and V.

To evaluate performance, we measure the length of the path executed by the agent under the control of each algorithm in the face of nine different obstacles. The obstacles were labeled ‘A’ through ‘I’ with varying size or shape; obstacle ‘A’ is a single spherical obstacle of size 0.5 units placed in the path of the agent, while obstacles ‘B,’ ‘C,’ ‘D,’ ‘E,’ ‘F,’ and ‘G’ are walls of length 1.5, 2, 3.5, 6.5, 12.50, and 17 units, respectively, placed transverse to the agent’s path to the target. Obstacle ‘H’ is a concave obstacle placed in the agent’s path; it is the same size (transversally) as ‘E’ but includes two extensions (1.25 units in length) making it concave with respect to

the agent’s straight-line path to the target. Obstacle ‘I’ is a variant of ‘H’ but with longer extensions (making it more concave than ‘H’).

As in the previous section, we stress that path length is not taken as the sole metric of performance: our aim here is to understand how the algorithm is able to cope with varying obstacles. That is, an algorithm that achieves great performance (i.e., low path lengths) for some obstacles but gets stuck at larger ones is clearly not as robust (and from our perspective as designers of “satisficing” systems, undesirable) as one that achieves *tolerable* performance for all obstacles. Hence, a higher performing algorithm (in the sense we adopt here) is one that can circumnavigate more obstacles.

### 3. Results

Figures 101, 102 and 103 illustrate paths executed by the agent under control of all four algorithms. Algorithms Bra3c and BorKor-d are shown superposed on each other (with BorKor-d’s path shown in a lighter shade), as are cr and Mat (with Mat shown in a lighter shade). All algorithms enable the agent to circumnavigate the obstacle, with Bra3c achieving the shortest path.

In Figure 104 we see BorKor-d’s inability to avoid obstacle formation ‘D’, while Figure 105 presents Bra3c’s similar failure. With Figures 106 and 107 only cr and Mat remain viable. Finally, Figure 108 shows that Mat is unable to get past the concave obstacle, while the proposed technique—cr—is.

The actual path length measurements are presented in Table XIV. As can be seen, for smaller obstacles Bra3c performs best, with cr achieving a moderate level of performance. With increasing obstacle size, however, cr (and Mat, upto a point) is able to cope quite well; it is also able to circumnavigate a moderately-sized concave obstacle, ‘H.’ We stress, however, that cr is still a local method and so its robustness is limited; the inability of cr to circumnavigate a more concave obstacle, ‘I,’ as seen

in Figure 109 is indicative of this.

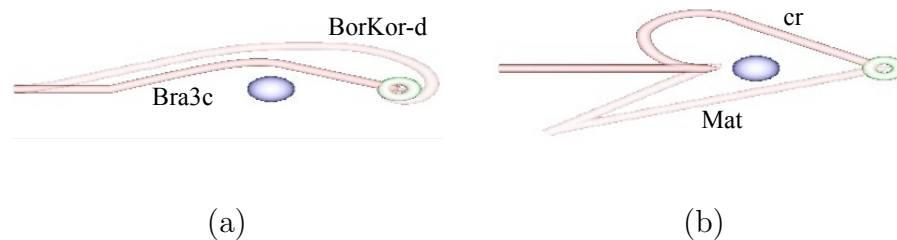


Fig. 101. Paths executed by the agent around obstacle formation ‘A.’

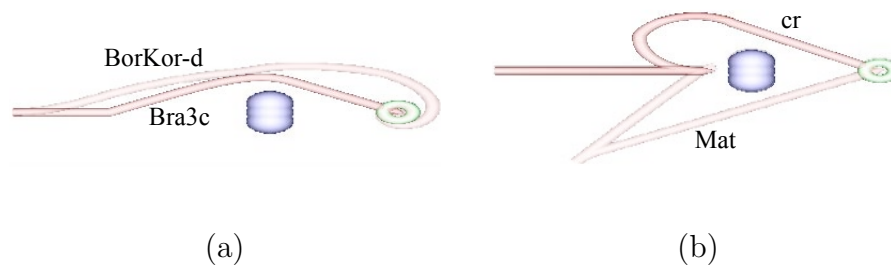


Fig. 102. Paths executed by the agent around obstacle formation ‘B.’

## D. Discussion

From the comparisons we looked at for taxis, we find one strength of our approach is robustness against target sensor degradation. This suggests its applicability in robotic schemes that require lightweight sensing faculties (e.g., a multi-agent scheme where economic constraints preclude the use of complex target sensors for a multiplicity of agents). Alternatively, it can also serve application spaces where there is a potential for the sensing faculties to suffer damage in the course of operation.

For general navigation, the results suggest that our technique is on par with the navigation behaviors proposed by Mataric. We note that our agent was able to

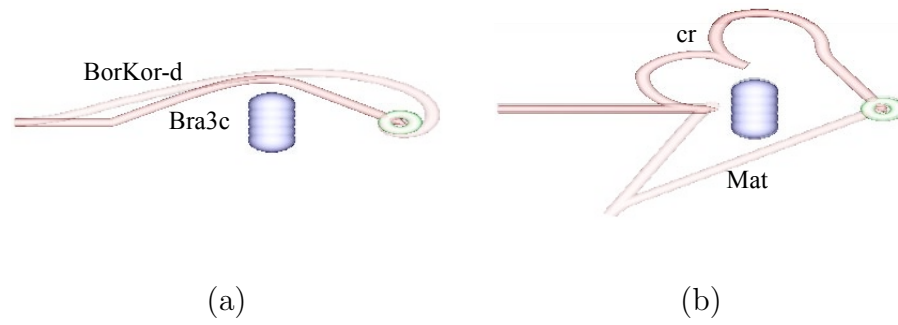


Fig. 103. Paths executed by the agent around obstacle formation 'C.'

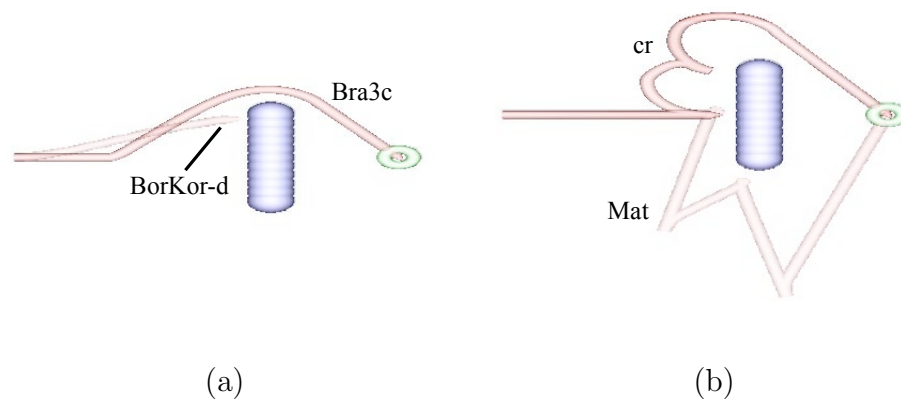


Fig. 104. Paths executed by the agent around obstacle formation 'D.' The BorKor-d method failed to circumnavigate the obstacle.

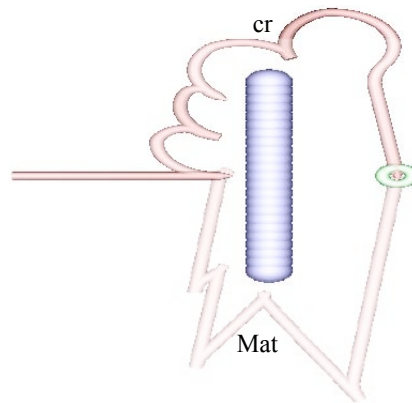


Fig. 105. Paths executed by the agent around obstacle formation 'E.' Only cr and Mat enabled the agent to avoid the obstacle.

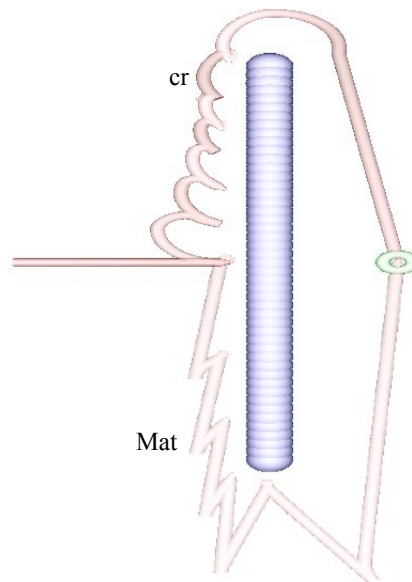


Fig. 106. Paths executed by the agent around obstacle formation 'F.' Only cr and Mat enabled the agent to avoid the obstacle.

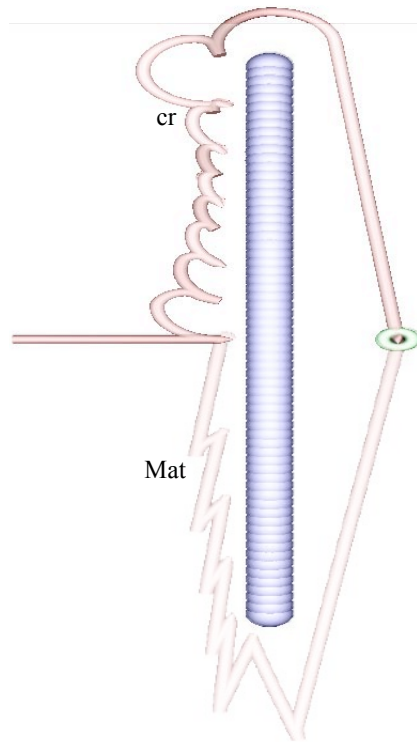


Fig. 107. Paths executed by the agent around obstacle formation 'G.' Only cr and Mat enabled the agent to avoid the obstacle.

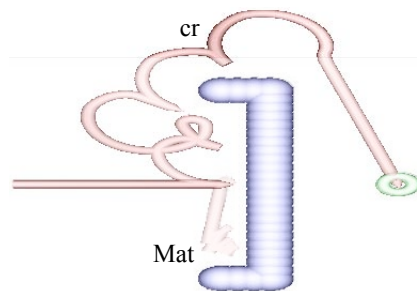


Fig. 108. Paths executed by the agent around obstacle formation 'H' (note: this concave obstacle was created by adding two horizontal extensions on the top and bottom of formation 'E'). Only cr enabled the agent to avoid the obstacle.



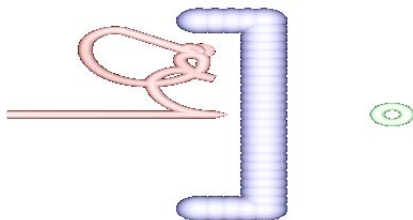


Fig. 109. Path executed by the agent around obstacle formation ‘I’ (note: this concave obstacle was created by lengthening the two horizontal extensions of obstacle ‘H’) under control of ‘cr.’ The obstacle is sufficiently concave as to cause ‘cr’ to fail to circumnavigate it.

Table XIV. Summary of algorithm performance characteristics: taxis with obstacle avoidance. The dash (–) indicates that the algorithm did not converge, that is, the agent was unable to circumnavigate the obstacle.

algorithm	path length for obstacle formation:								
	A	B	C	D	E	F	G	H	I
BorKor-d	9.77	9.70	9.70	–	–	–	–	–	–
Bra3c	6.72	6.92	7.16	8.18	–	–	–	–	–
Mat	11.89	11.92	11.90	17.33	22.36	32.77	37.81	–	–
<b>cr</b>	9.17	9.17	12.47	12.31	18.40	23.00	30.61	22.72	–

circumnavigate a concave obstacle formation that Mat was unable to; however, it is possible that with appropriate tuning of the backoff timer, Mat could be set to circumnavigate that case.<sup>9</sup> One benefit of our scheme that this demonstrates, on the other hand, is the fewer parameters that need to be tuned for it to operate tolerably well.

To reiterate, the focus of our work, and its strength, is for situations where an agent must cope with poor sensory information. This is reflected in our low-resolution requirements for target sensing (e.g., setting the sensor coarseness such that  $n = 3$  was sufficient for our scheme to perform very well) and obstacle sensing (monocular sensing). It is instructive, however, to compare our scheme against a technique with more sophisticated information available to it.

Consider the sensing scheme of Figure 110 which is able to perceive the relative displacement from an agent to a set of obstacles. This is considerably more complex than our monocular scheme, not only in terms of increased hardware costs (due to the requirement for several ranging sensors, such as ultrasound transducers) but also due to increased processing time. Even if processing was done in parallel, the fundamental problem with using several ranging transducers (e.g., the twelve-sensor sonar ring of Brooks [58]) that measure the transit time for an emitted signal to return to the device is that the firing of the ranging transducers must be done in sequence (or, at most, in pairs) to ensure that there is no cross-coupling between sent and received signals. This requirement for sequential firing, in turn, imposes limitations on the top speed of the agent.

If we disregard complexity, however, we find that a scheme that is able to use this high-quality information would perform better than ours. For example, Figure 111

---

<sup>9</sup>We do stress, however, that the backoff timer was not set arbitrarily, but instead was matched to the time constants of our overstimulation filters.

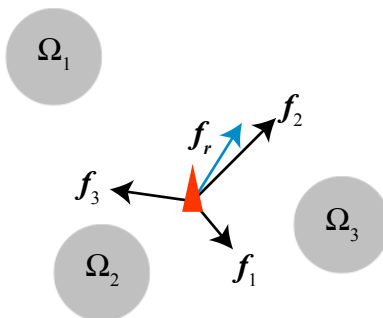
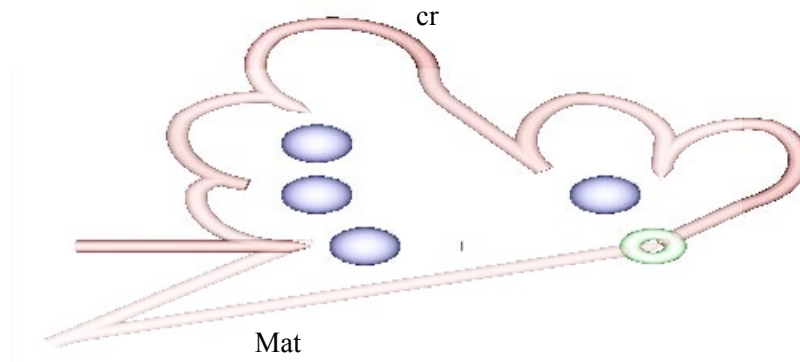


Fig. 110. The omni-directional obstacle sensor measures the displacement,  $\mathbf{f}_i$  for  $i \in \{1, 2, 3\}$ , from the obstacles (numbered 1, 2 and 3 in the figure) surrounding an agent (in red) to the agent, sums them (while attenuating their magnitudes so that closer obstacles yield longer vectors), and returns the resultant vector,  $\mathbf{f}_r$ .

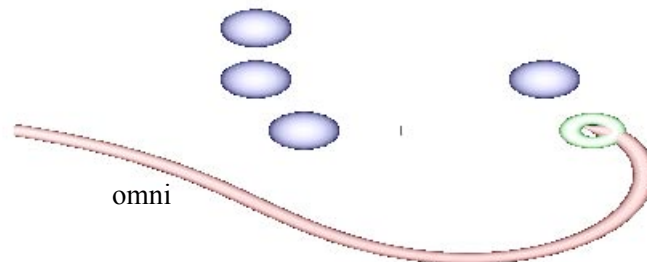
presents a comparison of Mat and cr (as implemented in the previous section) against the virtual force field method with an omni-directional obstacle sensor.<sup>10</sup> With global information, omni is able to perform better than the local techniques because of its enhanced long-range perceptual acuity—leading to a better path that avoid obstacles earlier, as well as circumventing problem obstacle formations (that cr is unable to, as can be seen in the figure).

Hence, we do not claim that our proposed control scheme is suitable for all applications. As mentioned in chapter I, by endowing an agent with sufficient perceptual and computational faculties, the need for local techniques vanishes. However, for the applications that motivate this work—search and rescue, robotic exploration, and mobile sensor networks—which often demand lightweight solutions, our lightweight scheme, being comparatively robust to degraded sensing and large obstacle forma-

<sup>10</sup>Given an omni-directional sensor with sufficiently long range, the virtual force field method begins to look more like a potential field method.



(a)



(b)

Fig. 111. A comparison of cr and Mat (both using monocular obstacle sensors), and the virtual force field method using an omni-directional sensor (omni). The path lengths executed by the agent for each of these cases were 21.77, 11.85, and 9.16 respectively.

tions, is a promising candidate.

## CHAPTER IX

### EXPERIMENTAL VALIDATION

A layman who has experienced things is more to be trusted than a sage who speaks on the basis of theoretical knowledge but without experience.  
(Mar Isaac of Nineveh)

#### A. Introduction

So far we have developed our control architecture from a mathematical point of view; having designed basis behaviors with provable characteristics, we then integrated them into control schemes, using a custom simulation environment<sup>1</sup> to verify the schemes.

In this chapter we present the design of a custom experimental testbed for our scheme, and present experimental results that help demonstrate the efficacy of our approach.

#### 1. Methodology

There are two basic philosophies for robotic testbeds found in the literature.

In the first, mobile mechatronic systems with limited autonomy are constructed with wireless faculties to communicate with an external (i.e., off-board) controller. An external sensing system (e.g., an overhead camera) measures phenomena relevant to each agent (e.g., the displacement from the agent to the target, other agents, or obstacles) and either:

---

<sup>1</sup>The simulation environment was developed using Matlab and Simulink; the results of integrating the equations of motion in Matlab were visualized using a custom Open-GL based application. An overview of this environment is presented in Appendix B.

- computes the relevant actuation signals for each agent (according to the control architecture that would be onboard the agent in a production system) and transmits this to the agents to control the actuators directly
- transmits this information to a controller onboard the agent, which then, according to its control architecture, drives the actuators

This is only a modest step towards reality above simulation, since often the external sensing system has more sensory acuity and access to more global information than a real sensor system onboard the agent would have. This can yield unrealistically positive experimental results. However, this scheme is relatively straightforward to realize, isolating the designer from having to deal with practical (and highly non-ideal) local sensors; rather, much of the effort here would be in the design of software to measure phenomena from the external sensing system (e.g., image processing from video input). A possible problem, however, is the introduction of artifacts in the experimental results due to limitations in communication bandwidth and real-time processing delays. Beyond this, however, a major problem is that the testbed does not bring the robotic scheme much closer to a production environment: often this sort of testbed serves only a singular function of providing some experimental results.

The second approach is agent-based. That is, the testbed is an implementation of a practical robotic scheme, using real sensors and computation onboard the agent, that validate the control architecture on a realistic real-world platform—ideally, one that has the attributes of a production system. The major disadvantage with this approach is the amount of time required to develop the system.

This work is an engineering dissertation, and as such, we intend our scheme to have path from a theoretical setting to real-world application. Hence, it serves neither the goals of academic nor professional engineering to realize a testbed purely for the

sake of experimental measurements. For this reason, we adopt the second approach and realize a testbed with the following characteristics of a production mobile robotic system (similar to a system that would be used for any of the key applications of our work, robotic exploration and mobile sensor networks):

- we use steered vehicles (as opposed to ideal, but unrealistic, differentially-driven vehicles) on a performance chassis
- we use a custom computational substrate onboard the agent (custom digital hardware implemented on a field-programmable gate array)
- all sensing is done locally with relatively inexpensive production sensors
- we operate at fast speeds

We constructed two such agents for our testbed.

## **B. Testbed setup**

### **1. Chassis and electromechanical subsystems**

A front-wheel steered vehicle was used for the testbed. This was chosen over a differentially driven vehicle (more commonly found in prototype robots) as it is more reflective of a production implementation. Specifically, a performance remote-controlled vehicle was chosen, the Team Losi Mini-LST miniature (1/18 scale) monster truck, illustrated in Figure 112. The drivetrain is powered by dual DC motors; power is distributed to all four wheels via a transmission consisting of a slipper clutch and three differentials. Dual servomotors at the front of the vehicle serve to steer it. A stock 7.2 V 1100 mAh Ni-MH battery pack provides power to the motors as well all on-board electronics.



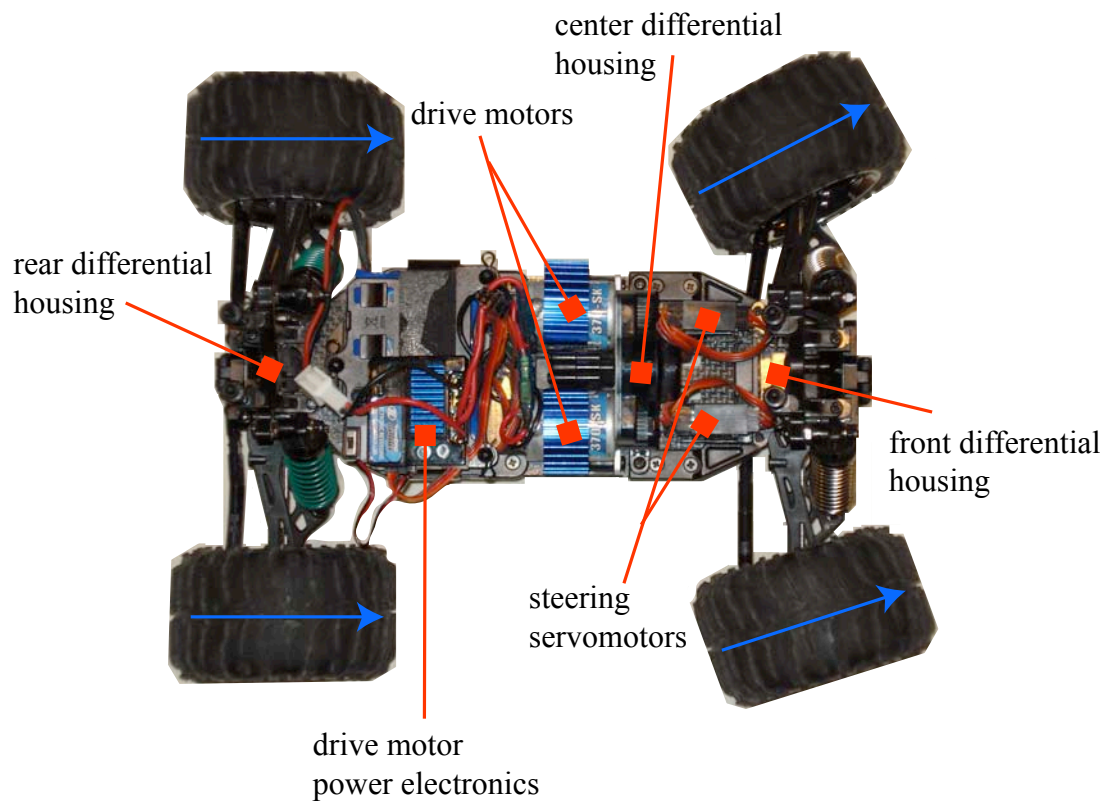


Fig. 112. The chassis used for our experimental testbed with an outline of the key electromechanical subsystems.

## 2. Sensors

### a. Obstacle sensing

To sense the presence of obstacles, we employed the LV-MaxSonar EZ-0 ultrasound transducers by MaxBotix Inc. These rangefinders have a mass of 4.3g, a volume of approximately  $16\text{cm}^3$ , and could be operated at voltages as low as 2.5V; operating in free-running mode they provide measurements every 49ms.

The parts had a tolerable precision of  $\pm 2.54\text{cm}$ , and could range obstacles at a distance 15cm to 645cm away from the sensor; for closer obstacles, the sensor could indicate the obstacle's presence or absence, but not its distance.

The sensor provides three interfaces for reading sensor data, RS-232, an analog voltage-based scheme, and the single-wire pulse width modulated scheme shown in Figure 113. The hardware we used to read data from this sensor is shown in Figure 114.

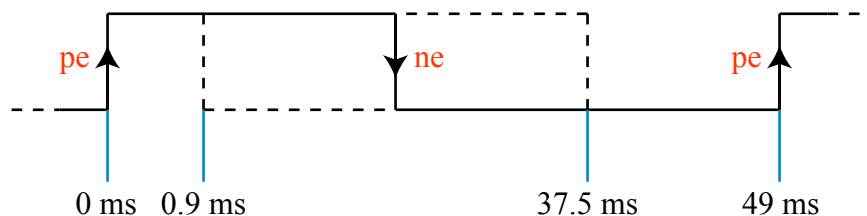


Fig. 113. Pulse width modulated scheme for obstacle sensor data read-out. The width of the pulse is directly proportional to the distance to the closest obstacle in front of it.

### b. Target and agent sensing

Polulu Inc. produces small form-factor infrared transceiver pairs which can be used by two agents to locate one another. The sensor is far from ideal: it provides no range measurements and produces a very coarse four-level quantization of target

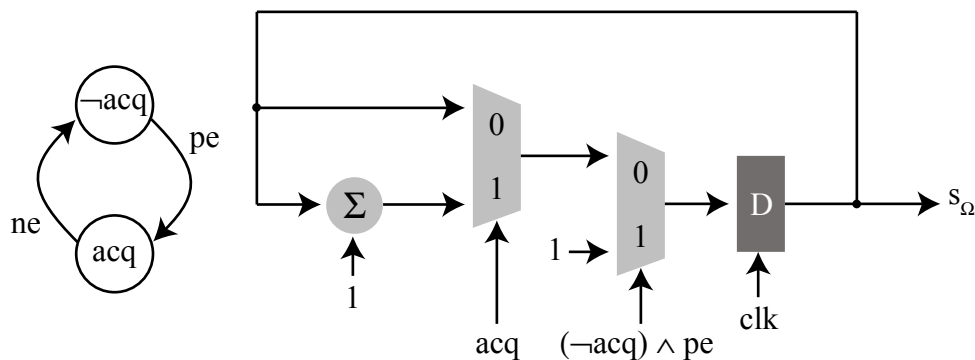


Fig. 114. Hardware used to read obstacle sensor data.

direction.<sup>2</sup> The advantages of this part, however, are the long detection range, and the straightforward interface, requiring the sampling of four wires (level shifting from 5V logic to 3.3V logic was required, and was accomplished via 74ACT244 octal driver).

Figure 115 illustrates the circuit used to convert the four bit measurement from the transceiver to the vector,  $s_T$ , used by the control laws ( $ir_n, ir_s, ir_e, ir_w$  are level-shifted versions of the output of this part).

### 3. Actuators

The drive motor power electronics and steering servomotors both take in pulse width modulated signals to set either speed and direction (with the former), or steering angle (with the latter). To implement the speed drivers a free-running digital counter was used to time a period of 7.6 ms. At the start of each period, a two-state flag would assert and remain asserted for a duration that corresponds to a control input from the control hardware.

<sup>2</sup>As such it provides a means of stressing those basis behaviors which very minimally satisfy our definition of a measurement function.

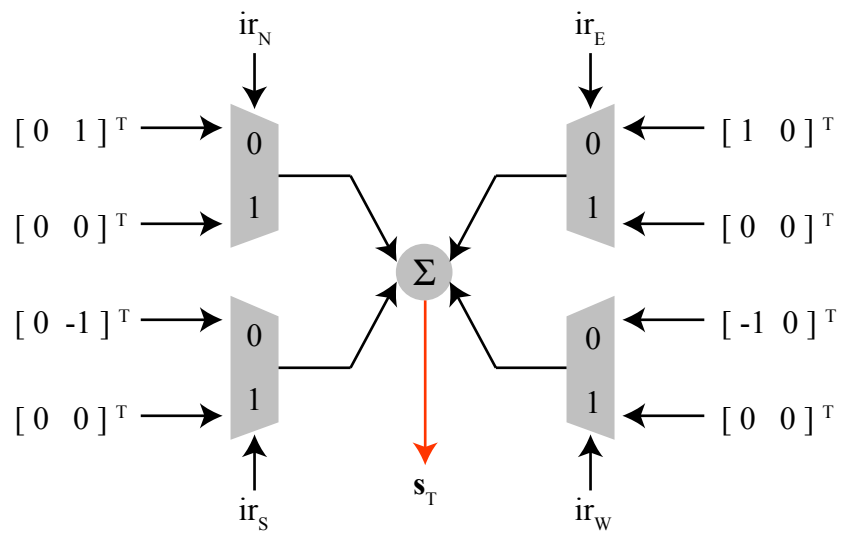


Fig. 115. Hardware for target sensor data conversion.

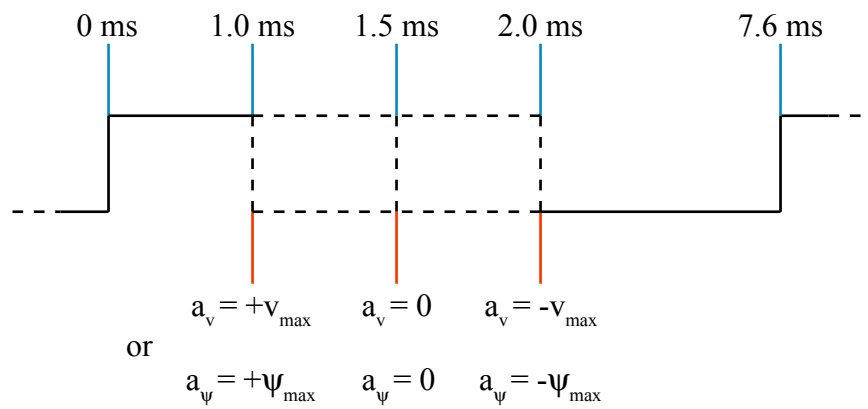


Fig. 116. Pulse width modulation scheme for the drive motor power electronics and steering servomotors.

#### 4. Computational substrate

To implement the control hardware as well as any interfacing hardware, we used field-programmable gate arrays (FPGAs). Opal Kelly Inc. produces modular development boards for relatively high-density field-programmable gate array devices. They are particularly well-suited to our robotics application due to the small form factors, and lack of unnecessary hardware (e.g., other than the FPGA and some RAM, they do not have other hardware peripherals onboard available to the designer). Moreover, they are well-supported by custom CAD software from Opal Kelly, as well as FPGA CAD from Xilinx.

We used two models for our testbed, the XEM3001 and XEM3005; both boards were over-specified for our lightweight control hardware.

##### a. Control hardware

The control hardware for the experimental results were mostly direct mappings of the control architectures presented in chapters VI and VII. The two exceptions were in the implementation of hysteresis functions and leaky integrators.

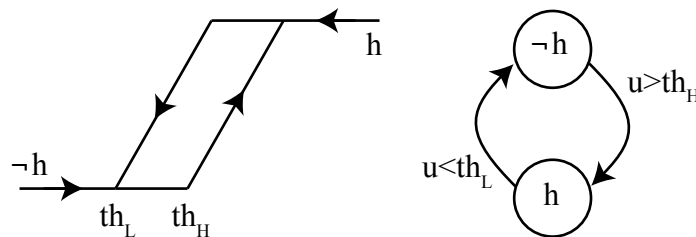


Fig. 117. Hardware approximation of a hysteresis function.

Our implementation of hysteresis function is shown in Figure 117. For a leaky integrator with input  $u$  and output  $y$  we employ the scheme of Figure 118 in which we first quantize  $u$  to  $\{0, 1\}$  (if it has not already been quantized) via the hysteresis

function of Figure 117 and use this to control a free-running counter.

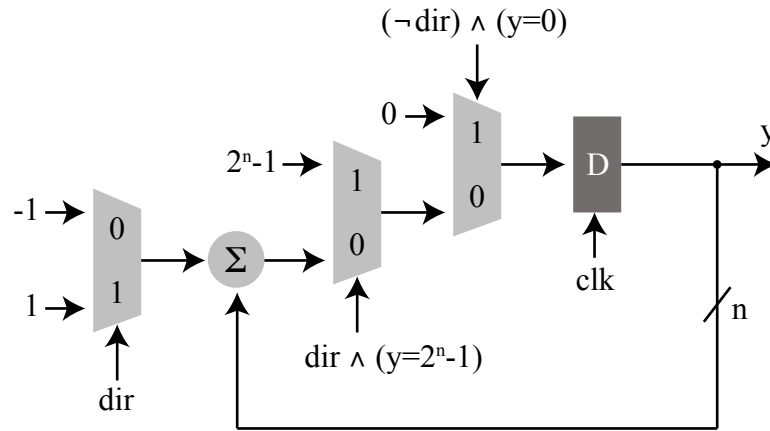


Fig. 118. Hardware approximation of a leaky integrator.

## 5. Integrated vehicle

Figures 119-121 show various views of the agent based on the XEM3001 board.

### C. Experiments

Several experiments were conducted to validate our theoretical development. They can be examined by referring to the video files that accompany this dissertation (the details of these files can be found in appendix A).

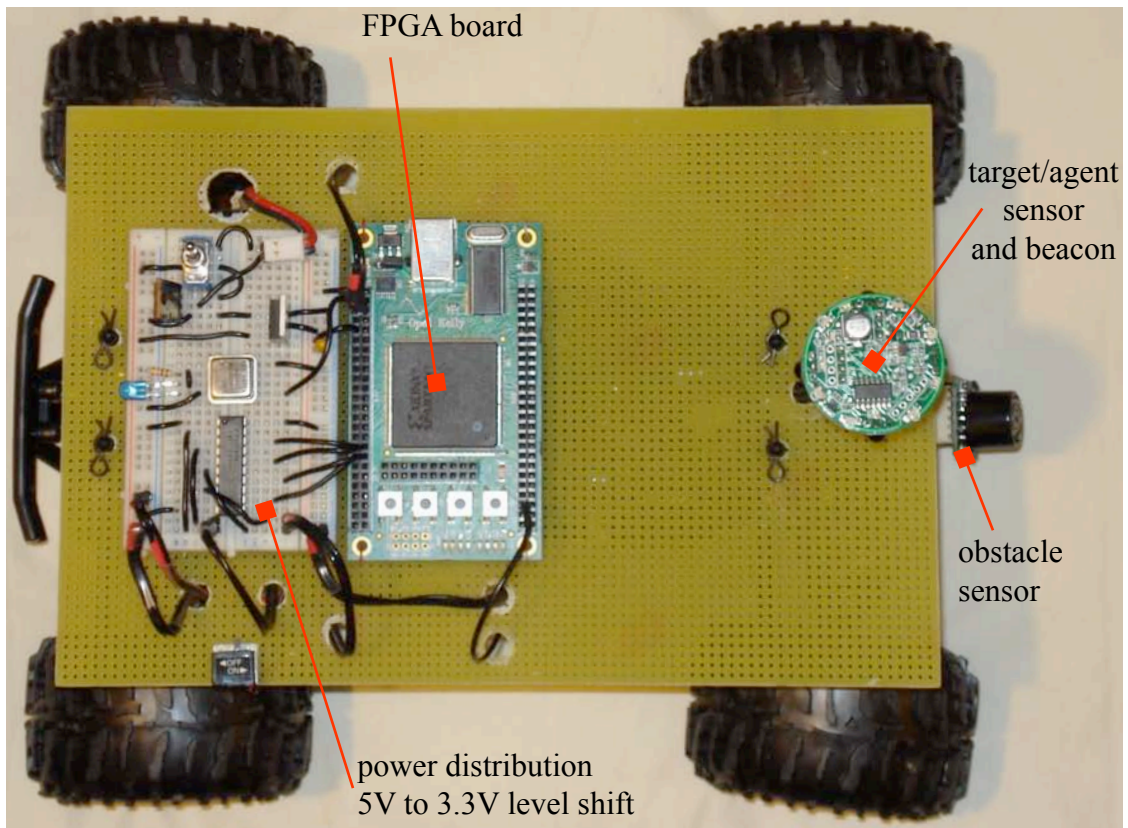


Fig. 119. Top view of agent.

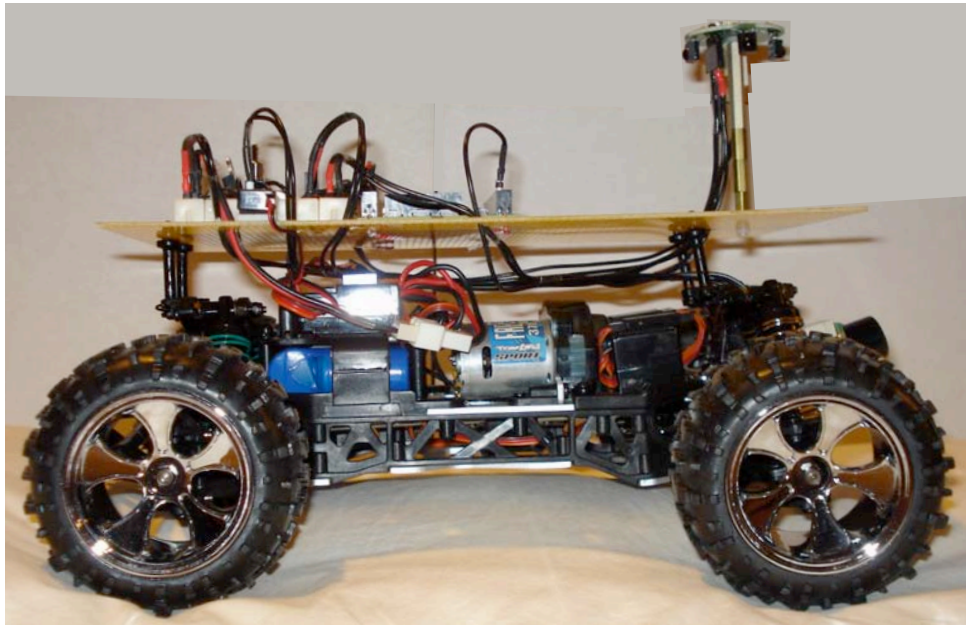


Fig. 120. Side view of agent.

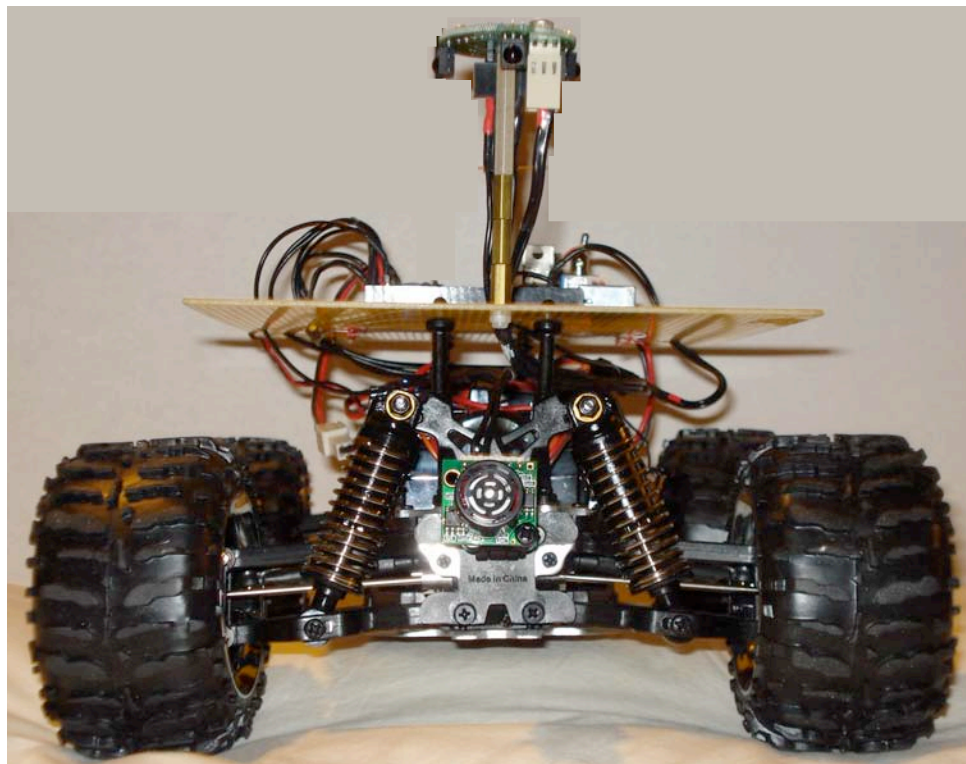


Fig. 121. Front view of agent.



## CHAPTER X

### CONCLUSION

Furthermore, my child, take heed: of writing many books there is no end, and much study is a weariness of the flesh. (Ecclesiastes)

#### A. Summary of the work

This work addressed the engineering of useful weakly emergent behaviors for mobile autonomous robots.

We began with the construction of static regulators to realize elementary behaviors for purely reactive systems. We also presented the design of a dynamical control scheme where obstacle avoidance was an emergent behavior. For both schemes, elementary behaviors that deal with well-posed problems (from the perspective of having adequate sensory and actuation faculties to cope with the problem) were realized using control theory.

To realize an integrated control architecture that would enable an agent to cope competently (i.e., satisfice) in an unstructured environment, we turned to the engineering of emergent behavior. A hierarchical and layered topology for structuring the regulators was proposed, based on the multi-scale nature of the environment. Composite behaviors that used our basis behaviors were realized, and several simulation results were used to qualify our scheme. An experimental two-agent testbed was used to validate the control architecture on a real-world mobile mechatronic artifact.

The contributions of this dissertation (many of which have been disseminated in [117, 101, 102, 95, 103]) include:

- a novel tool, vector field design, for the synthesis of static (purely reactive) behaviors

- the development of a dynamical control scheme that manifests weakly emergent behavior
- a novel mechanism<sup>1</sup> to realize composite behaviors, the simultaneous regulation of two senses
- the design of an integrated cybernetic control architecture for lightweight robotic exploration problems and mobile sensor networks
- the engineering of an emergent self-organization behavior which is highly decentralized for homogeneous multi-agent systems
- a software verification environment and visualization tool for qualification of cybernetic robotic research schemes
- a performance testbed based on custom hardware and FPGA-based computation

## B. Future work

We propose some areas of future work that extend this dissertation, and more broadly, should further the development of *cybernetic* brains for robotics.

### 1. Extending vector field design

In this work we presented the use of vector field design for the case of regulating a system evolving in a two-dimensional problem space. However, for problems that involve higher-dimensional state spaces (e.g., applications to unmanned aerial vehicles or underwater exploration, dynamical vehicle models, over-actuated kinematic

---

<sup>1</sup>Beyond action superposition and multiplexing.

models) the tool in its current form is of limited use. Future work would involve extending the tool to higher dimensions. To retain the visual nature of the tool—and its intuitiveness—a mechanism to go back and forth between two- or three-dimensional projections of a higher-dimensional state space (which could be used for design) would be necessary.

As well, in this work we described the relationship between flow geometry and agent behavior with respect to circular flows, and along the  $L_-$  and  $L_+$  manifolds. Future work could extend this relationship to different flow geometries and along other manifolds.

## 2. Improved perception

The perceptual schemes we considered in this work were intentionally limited to lightweight schemes that did not involve vision. However, considering natural organisms and some of the “heavier-weight” robotics schemes were looked at, it is clear some form of machine vision would be very useful, especially in enabling the agent to simultaneously perceive a multiplicity of different environmental phenomena—other agents, obstacles, targets—in a unified manner.

Moreover, the use of feedback based on optic flow<sup>2</sup> could provide a economical means of obtaining motion feedback for the agent (beyond the use of MEMS-based sensors which are, at present, expensive and of limited accuracy), as well as perceiving the relative velocity of neighboring phenomena.

Future work could involve developing schemes for the regulation of visual perception. For example, one could consider the agent’s field of vision as a spatially-distributed dynamical system (modeled by partial differential equations) whose evolution in time and distribution in *space* could be steered by the agent’s actuation.

---

<sup>2</sup>For which neuromorphic approaches have started to emerge.

### 3. Machine “introspection”

In chapters VI and VII we utilized multiplexors to prevent actuation nulls when competitive behaviors superpose. An alternative strategy would be to design a higher-level regulator that “looks” at the internal regulators to determine whether non-productive actuation (caused by antagonistic controller outputs) are occurring.

Work of this nature might provide a route to more abstract cognition, where rather than referencing all perception to the outside world, some perception could be inward-directed, with regulators controlling the robot’s own control architecture (its “brain”).

### 4. Inter-agent communication

If behavior is regulation of perception (as we have held in this work), then perhaps an appropriate metaphor for communication would be the regulation of another agent’s control architecture. To that end, as an extension of the previous section (where the object was regulation of the agent’s own “mind”), future work could address the design of agent interaction schemes using regulators that attempt to control other agents through communication.

In realizing truly autonomous agents that can be classified as artificial life (as opposed to scripted automata that are essentially heirs of Vaucanson’s duck), we ought to get beyond the mere use of communication as a means of reading another agent’s status or broadcasting our own to achieve coordinated action. Rather, we would like to see an emergent system of communication develop through a cybernetic formulation of the problem. For example, consider Figure 122 which shows two agents,  $M$  and  $N$ , operating in a common environment. A possible formulation for  $E$  for the case of stigmergic communication between  $M$  and  $N$  (i.e., in which  $M$  and  $N$  exchange

information via actuating change in the environment) could be:

$$E : \begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{a}_M, \mathbf{a}_N) \\ \mathbf{s}_M = \mathbf{g}(\mathbf{x}, \mathbf{a}_M) + \mathbf{h}(\mathbf{x}, \mathbf{a}_M, \mathbf{a}_N) \end{cases} \quad (10.1)$$

With this, we could propose the problem of designing a regulator (in  $M$ ) that would cause  $E$  to be under the sole control of  $M$ —in effect, bending  $N$  to  $M$ 's will.

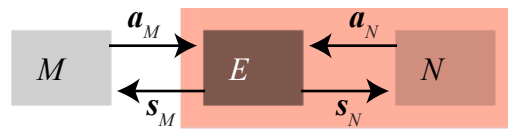


Fig. 122. The robot communication problem from the perspective of agent  $M$ .

## REFERENCES

- [1] Hubert L. Dreyfus, *Alchemy and Artificial Intelligence*, vol. P-3244, RAND Corporation, 1965.
- [2] Hans P. Moravec, “The Stanford cart and the CMU rover,” *Proceedings of the IEEE*, vol. 71, no. 7, pp. 872–884, 1983.
- [3] W. Ross Ashby, *Design for a Brain*, Chapman and Hall, second edition, 1960.
- [4] Pattie Maes, “Modeling adaptive autonomous agents,” in *Artificial Life: An Overview*, Christopher G. Langton, Ed., pp. 135–162. MIT Press, Cambridge, MA, 1995.
- [5] Herbert A. Simon, “A behavioral model of rational choice,” *The Quarterly Journal of Economics*, vol. 69, no. 1, pp. 99–118, 1955.
- [6] Herbert A. Simon, “Cognitive science: The newest science of the artificial,” *Cognitive Science*, vol. 4, pp. 33–46, 1980.
- [7] M. Okamoto, T. Sakai, and K. Hayashi, “Biochemical switching device realizing a McCulloch-Pitts type equation,” *Biological Cybernetics*, vol. 58, no. 5, pp. 295–299, 1988.
- [8] A. Hjelmfelt and J. Ross, “Chemical implementation and thermodynamics of collective neural networks,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 89, pp. 388–391, 1992.
- [9] A. Hjelmfelt, E. D. Weinberger, and J. Ross, “Chemical implementation of finite-state machines,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 89, pp. 383–387, 1992.

- [10] A. Hjelmfelt, E. D. Weinberger, and J. Ross, “Chemical implementation of neural networks and Turing machines,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 88, pp. 10983–10987, 1991.
- [11] M. Okamoto, T. Sakai, and K. Hayashi, “Switching mechanism of a cycle enzyme system: Role as a “chemical diode”,” *BioSystems*, vol. 21, pp. 1–11, 1987.
- [12] Steen Rasmussen, Liaohai Chen, David Deamer, David C. Krakauer, Norman H. Packard, Peter F. Stadler, and Mark A. Bedau, “Transitions from nonliving to living matter,” *Science*, vol. 303, pp. 963–965, 2004.
- [13] Uri Alon, *An Introduction to Systems Biology: Design Principles of Biological Circuits*, Chapman and Hall, 2006.
- [14] Rodney A. Brooks, “A hardware retargetable distributed layered architecture for mobile robot control,” in *IEEE International Conference on Robotics and Automation*, vol. 4, pp. 106–110, 1987.
- [15] Carver Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, 1989.
- [16] Rahul Sarpeshkar, “Analog versus digital: Extrapolating from electronics to neurobiology,” *Neural Computation*, vol. 10, pp. 1601–1638, 1998.
- [17] Jan Van der Spiegel, Paul Mueller, David Blackman, Peter Chance, Christopher Donham, Ralph Etienne-Cummings, and Peter Kinget, “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.
- [18] Miguel Figueroa, David Hsu, and Chris 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.
- [19] Roman Genov and Gert Cauwenberghs, “Kerneltron: Support vector machine in silicon,” *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1426–1434, September 2003.
- [20] Ralf M. Philipp and Ralph Etienne-Cummings, “Single-chip stereo imager,” *Analog Integrated Circuits and Signal Processing*, vol. 39, pp. 237–250, 2004.
- [21] Reid R. Harrison, “A biologically inspired analog IC for visual collision detection,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 11, pp. 2308–2318, 2005.
- [22] Randall D. Beer, Hillel J. Chiel, and Leon S. Sterling, “A biological perspective on autonomous agent design,” *Robotics and Autonomous Systems*, vol. 6, pp. 169–186, 1990.
- [23] Mark W. Tilden, “Adaptive robotic nervous systems and control circuits therefor,” U.S. Patent (Patent Number 5,325,031), June 1994.
- [24] W. Grey Walter, “An imitation of life,” *Scientific American*, vol. 182, no. 5, pp. 42–45, May 1950.
- [25] W. Grey Walter, “A machine that learns,” *Scientific American*, vol. 184, no. 8, pp. 60–63, August 1951.
- [26] Robert G. Gallager, “Claude E. Shannon: A retrospective on his life, work, and impact,” *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2681–2695, 2001.
- [27] Norbert Wiener, *The Human Use of Human Beings*, Houghton Mifflin, 1954.



- [28] W. Ross Ashby, “Design for an intelligence-amplifier,” in *Automata Studies*, C. E. Shannon and J. McCarthy, Eds. Princeton University Press, 1956.
- [29] Stafford Beer, “Fanfare for effective freedom—cybernetic praxis in government,” in *Platform for Change*, pp. 421–52. Wiley, 1975.
- [30] James S. Albus, “Mechanisms of planning and problem solving in the brain,” *Mathematical Biosciences*, vol. 45, pp. 247–293, 1979.
- [31] Dylan A. Shell and Maja J. Matarić, “Ergodic dynamics for large-scale distributed robot systems,” in *Proceedings of the 5th International Conference on Unconventional Computation*, pp. 254–266, 2006.
- [32] Jiming Liu and Jianbing Wu, *Multi-Agent Robotic Systems*, CRC Press, 2001.
- [33] Y. Uny Cao, Alex. S. Fukunaga, and Andrew B. Kahng, “Cooperative mobile robotics: Antecedents and directions,” *Autonomous Robots*, vol. 4, pp. 1–23, 1997.
- [34] W. Ross Ashby, *An Introduction to Cybernetics*, Chapman and Hall, 1968.
- [35] John Guckenheimer and Philip Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Springer-Verlag, 1983.
- [36] Lawrence Perko, *Differential Equations and Dynamical Systems*, Springer-Verlag, third edition, 2001.
- [37] Dragan Nesić and Dina S. Laila, “On preservation of dissipation inequalities under sampling,” in *Proceedings of the 39th IEEE Conference on Decision and Control*, pp. 2472–2477, 2000.

- [38] Y. Itoh, N. Hori, and H. Kamei, “Digital redesign of a nonlinear state-feedback control system based on the principle of equivalent areas,” in *SICE Annual Conference*, pp. 350–354, 2004.
- [39] 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.
- [40] Thomas L. Vincent and Walter J. Grantham, *Nonlinear and Optimal Control Systems*, John Wiley & Sons, 1997.
- [41] Tom Ziemke, “What’s that Thing Called Embodiment?,” in *Proceedings of the 25th Annual Meeting of the Cognitive Science Society*, pp. 1134–1139, 2003.
- [42] Susan Stepney, “Embodiment,” in *In Silico Immunology*, D. Flower and J. Timmis, Eds. Springer, 2007.
- [43] Stanley J. Rosenschein and Leslie Pack Kaelbling, “A situated view of representation and control,” *Artificial Intelligence*, vol. 73, pp. 149–173, 1995.
- [44] Ronald C. Arkin, *Behavior-Based Robotics*, MIT Press, 1998.
- [45] Jean-Claude Latombe, *Robot Motion Planning*, Kluwer, 1991.
- [46] Steven M. LaValle, *Motion Planning*, Cambridge University Press, 2006.
- [47] Dan Paluska, Maja J. Matarić, Robert Ambrose, and Jerry Pratt, “Biomimetic robot control,” in *Biologically Inspired Intelligent Robots*, Y. Bar-Cohen and C. Breazeal, Eds., vol. PM122. SPIE Press, 2003.
- [48] Allen Newell and Herbert A. Simon, “Computer science as empirical inquiry: Symbols and search,” *Communications of the ACM*, vol. 19, no. 3, pp. 113–126, 1976.

- [49] David Kirsh, “Foundations of AI: The big issues,” *Artificial Intelligence*, vol. 47, pp. 3–30, 1991.
- [50] John Haugeland, *Artificial Intelligence: The Very Idea*, MIT Press, 1985.
- [51] Michael Sipser, *Introduction to the Theory of Computation*, PWS Publishing Company, 1997.
- [52] C. Moore, “Generalized shifts: Unpredictability and undecidability in dynamical systems,” *Nonlinearity*, vol. 4, pp. 199–230, 1991.
- [53] Rodney A. Brooks, “Elephants don’t play chess,” *Robotics and Autonomous Systems*, vol. 6, pp. 3–15, 1990.
- [54] Valentino Braitenberg, “Taxis, kinesis and decussation,” *Progress in Brain Research*, vol. 17, pp. 210–222, 1965.
- [55] Valentino Braitenberg, *Vehicles: Experiments in Synthetic Psychology*, The MIT Press, 1984.
- [56] Michael A. Arbib, *The Metaphorical Brain*, Wiley-Interscience, 1972.
- [57] Marvin Minsky, *The Society of Mind*, Simon & Schuster, 1986.
- [58] Rodney A. Brooks, “A robust layered control system for a mobile robot,” *IEEE Transactions on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.
- [59] Ronald C. Arkin, “Motor schema-based mobile robot navigation,” *The International Journal of Robotics Research*, vol. 8, no. 4, pp. 92–112, 1989.
- [60] Pattie Maes, “Situated agents can have goals,” *Robotics and Autonomous Systems*, vol. 6, pp. 49–70, 1990.

- [61] Ronald C. Arkin and Tucker R. Balch, “AuRA: Principles and practice in review,” *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2/3, pp. 175–188, 1997.
- [62] Randall D. Beer, *Intelligence as Adaptive Behavior*, Academic Press, Inc., 1990.
- [63] R. D. Beer, “A dynamical systems perspective on agent-environment interaction,” *Artificial Intelligence*, vol. 72, pp. 173–215, 1995.
- [64] Simon Haykin, *Neural Networks*, Prentice Hall, 1999.
- [65] P. S. Churchland and T. J. Sejnowski, *The Computational Brain*, MIT Press, 1992.
- [66] Brosl Hasslacher and Mark W. Tilden, “Living machines,” in *Robotics and Autonomous Systems: The Biology and Technology of Intelligent Autonomous Agents*, L. Steels, Ed. Elsevier, 1995.
- [67] Stanley J. Rosenschein, “Formal theories of knowledge in AI and robotics,” *New Generation Computing*, vol. 3, no. 4, pp. 345–357, 1985.
- [68] Stanley J. Rosenschein and Leslie Pack Kaelbling, “The synthesis of digital machines with provable epistemic properties,” in *TARK '86: Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge*, pp. 83–98, 1986.
- [69] Leslie Pack Kaelbling, “An architecture for intelligent reactive systems,” in *Reasoning about actions & plans : Proceedings of the 1986 Workshop*, number 4, pp. 395–410, 1986.

- [70] Leslie Pack Kaelbling and Stanley J. Rosenschein, “Action and planning in embedded agents,” *Robotics and Autonomous Systems*, vol. 6, pp. 35–48, 1990.
- [71] H. Asama, A. Matsumoto, and Y. Ishida, “Design of an autonomous and distributed robot system: ACTRESS,” in *Intelligent Robots and Systems*, pp. 283–290, 1989.
- [72] Claude LePape, “A combination of centralized and distributed methods for multi-agent planning and scheduling,” in *IEEE ICRA*, pp. 488–493, 1990.
- [73] Lynne E. Parker, “ALLIANCE: An architecture for fault tolerant multi-robot cooperation,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, 1998.
- [74] Ruggero Carli, Fabio Fagnani, Alberto Speranzon, and Sandro Zampieri, “Communication constraints in coordinated consensus problems,” in *Proceedings of the 2006 American Control Conference*, pp. 4189–4194, 2006.
- [75] Dimos V. Dimarogonas and Kostas J. Kyriakopoulos, “On the state agreement problem for multiple unicycles,” in *Proceedings of the 2006 American Control Conference*, pp. 2016–2021, 2006.
- [76] Zhiyun Lin, Bruce Francis, and Manfredi Maggiore, “Coupled dynamic systems: From structure towards state agreement,” in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 3303–3308, 2005.
- [77] Zhiyun Lin, Bruce Francis, and Manfredi Maggiore, “Necessary and sufficient graphical conditions for formation control of unicycles,” *IEEE Transactions on Automatic Control*, vol. 50, no. 1, pp. 121–127, 2005.

- [78] Luc Moreau, “Stability of multiagent systems With time-dependent communication links,” *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.
- [79] Jakob Fredslund and Maja J. Matarić, “A general algorithm for robot formations using local sensing and minimal communication,” *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 837–846, 2002.
- [80] Petter Ögren, Edward Fiorelli, and Naomi Ehrich Leonard, “Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment,” *IEEE Transactions on Automatic Control*, vol. 49, no. 8, pp. 1292–1302, 2004.
- [81] Guy Theraulaz and Eric Bonabeau, “A brief history of stigmergy,” *Artificial Life*, vol. 5, pp. 97–116, 1999.
- [82] Owen Holland and Chris Melhuish, “Stigmergy, self-organization, and sorting in collective robotics,” *Artificial Life*, vol. 5, pp. 173–202, 1999.
- [83] C. W. Reynolds, “Flocks, herd, and schools: A distributed behavioral model,” in *SIGGRAPH*, pp. 25–34, 1987.
- [84] Maja J. Matarić, “Designing emergent behaviors: From local interactions to collective intelligence,” in *Proceedings, From Animals to Animats 2, Second International Conference on Simulation of Adaptive Behavior*. pp. 432–441, MIT Press, 1993.
- [85] Maja J. Matarić, “Designing and understanding adaptive group behavior,” *Adaptive Behavior*, vol. 4, no. 1, pp. 50–81, 1995.

- [86] Ronald C. Arkin, “Cooperation without communication: Multiagent schema-based robot navigation,” *Journal of Robotic Systems*, vol. 9, no. 3, pp. 351–364, 1992.
- [87] Tucker Balch and Ronald C. Arkin, “Communication in reactive multiagent robotic systems,” *Autonomous Robots*, vol. 1, pp. 1–25, 1994.
- [88] Lynne E. Parker and F. Tang, “Building multi-robot coalitions through automated task solution synthesis,” *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1289–1305, 2006.
- [89] Qun Li and Daniela Rus, “Navigation protocols in sensor networks,” *ACM Transactions on Sensor Networks*, vol. 1, no. 1, pp. 3–35, 2005.
- [90] Manfredo P. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice Hall, 1976.
- [91] Michael G. Hinchey, Roy Sterritt, and Chris Rouff, “Swarms and swarm intelligence,” *Computer*, vol. 40, no. 4, pp. 111–113, 2007.
- [92] Y. Sato and T. Ikegami, “Nonlinear computation with switching map systems,” *Journal of Universal Computer Science*, vol. 6, no. 9, pp. 881–905, 2000.
- [93] Vladimir J. Lumelsky, “Algorithmic issues of sensor-based robot motion planning,” in *Proceedings of the 26th IEEE Conference on Decision and Control*, vol. 3, pp. 1796–1801, 1987.
- [94] Luc Steels, “Intelligence with representation,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences*, vol. 361, no. 1811, pp. 2381–2395, 2003.

- [95] Nebu John Mathai and Takis Zourntos, “Emergent fluctuations in the trajectories of agent collectives,” *Fluctuation and Noise Letters*, vol. 7, no. 4, pp. L429–L437, 2007.
- [96] Hassan K. Khalil, *Nonlinear Systems*, Prentice Hall, third edition, 2002.
- [97] Richard J. Dunn, Price T. Bingham, and Charles A. Fowler, “Ground moving target indicator radar and the transformation of U.S. warfighting,” Tech. Rep., Northrop Grumman, Washington, 2004.
- [98] W. Koch and R. Klemm, “Ground target tracking with STAP radar,” *IEE Proceedings Radar, Sonar and Navigation*, vol. 148, no. 3, pp. 173–185, 2001.
- [99] Chris Eliasmith, “Computation and dynamical models of mind,” *Minds and Machines*, vol. 7, pp. 531–541, 1997.
- [100] Maja J. Matarić, “Behavior-based control: Main properties and implications,” in *Proceedings of the IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*, 1992.
- [101] Nebu John Mathai and Takis Zourntos, “Control-theoretic synthesis of hierarchical dynamics for embodied cognition in autonomous robots,” in *IEEE Symposium on Artificial Life*, pp. 448–455, 2007.
- [102] Takis Zourntos and Nebu John Mathai, “A BEAM-inspired Lyapunov-based strategy for obstacle avoidance and target-seeking,” in *American Control Conference*, pp. 5302–5309, 2007.
- [103] Takis Zourntos, Nebu John Mathai, S. Magierowski, and Deepa Kundur, “A bio-inspired layered analog scheme for navigational control of lightweight au-



- tonomous agents,” in *IEEE International Conference on Robotics and Automation*, 2008 (*to appear*).
- [104] C. D. Johnson, “Accommodation of external disturbances in linear regulator and servomechanism problems,” *IEEE Transactions on Automatic Control*, vol. 16, no. 6, pp. 635–644, 1971.
- [105] Miroslav Krstić, Ioannis Kanellakopoulos, and Petar V. Kokotović, *Nonlinear and Adaptive Control Design*, Wiley, 1995.
- [106] W. Ross Ashby and Roger C. Conant, “Every good regulator of a system must be a model of that system,” *International Journal of Systems Science*, vol. 1, no. 2, pp. 89–97, 1970.
- [107] Michael A. Arbib, Ed., *The Handbook of Brain Theory and Neural Networks*, MIT Press, 2003.
- [108] Mark A. Bedau, “Weak emergence,” in *Philosophical Perspectives: Mind, Causation, and World*, J. Tomberlin, Ed., vol. 11, pp. 375–399. 1997.
- [109] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, 2nd edition, 1992.
- [110] Michael Bratman, “Two faces of intention,” *The Philosophical Review*, vol. 93, no. 3, pp. 375–405, July 1984.
- [111] Philip R. Cohen and Hector J. Levesque, “Intention is choice with commitment,” *Artificial Intelligence*, vol. 42, pp. 213–261, 1990.
- [112] H. R. Maturana and F. J. Varela, *Autopoiesis and Cognition*, vol. 42 of *Boston Series in the Philosophy of Science*, D. Reidel Publishing Company, 1980.

- [113] Mark A. Bedau, John S. McCaskill, Norman H. Packard, Steen Rasmussen, Chris Adami, David G. Green, Takashi Ikegami, Kunihiko Kaneko, and Thomas S. Ray, “Open problems in artificial life,” *Artificial Life*, vol. 6, pp. 363–376, 2000.
- [114] Steen Rasmussen, Nils A. Baas, Bernd Mayer, Martin Nilsson, and Michael W. Olesen, “Ansatz for dynamical hierarchies,” *Artificial Life*, vol. 7, pp. 329–353, 2001.
- [115] Martin N. Jacobi, “Hierarchical organization in smooth dynamical systems,” *Artificial Life*, vol. 11, pp. 493–512, 2005.
- [116] Simon McGregor and Chrisantha Fernando, “Levels of description: A novel approach to dynamical hierarchies,” *Artificial Life*, vol. 11, pp. 459–472, 2005.
- [117] Nebu John Mathai and Takis Zourntos, “A hierarchical dynamical systems analog computation architecture for embodied cognition,” in *Unconventional Computation: Quo Vadis*, Los Alamos National Laboratory, March 2007.
- [118] Scott Burlington and Gregory Dudek, “Spiral search as an efficient mobile robotic search technique,” Tech. Rep., Center for Intelligent Machines, McGill University, Montreal, January 1999.
- [119] Hans Sagan, *Space-Filling Curves*, Springer-Verlag, 1994.
- [120] Archimedes, “On spirals,” in *The Works of Archimedes*, Sir Thomas Heath, Ed. Dover, 2002.
- [121] Tucker Balch and Ronald C. Arkin, “Behavior-based formation control for multirobot teams,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, 1998.

- [122] Reza Olfati-Saber, “Flocking for multi-agent dynamic systems: Algorithms and theory,” *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.
- [123] B. H. Krogh, “A generalized potential field approach to obstacle avoidance control,” in *International Robotics Research Conference*, August 1984.
- [124] Oussama Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.
- [125] Johann Borenstein and Yorem Koren, “Real-time obstacle avoidance for fast mobile robots,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, September/October 1989.
- [126] Y. Koren and J. Borenstein, “Potential field methods and their inherent limitations for mobile robot navigation,” in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 1398–1404, 1991.

**APPENDIX A**  
**EXPERIMENTAL RESULTS**

Video files (described in Table XV) that present some of our experimental results accompany this dissertation and are available for downloading.

Table XV. Summary of accompanying video files.

file	description
<code>taxis_obs.mov</code>	search, taxis, obstacle avoidance
<code>taxis_wall.mov</code>	taxis, obstacle avoidance (larger obstacle)
<code>capture.mov</code>	two-agent follower-leader scenario (involves the capture of one agent)
<code>escape.mov</code>	two-agent follower-leader scenario (involves the escape of one agent)

## APPENDIX B SIMULATION ENVIRONMENT

### A. Introduction

Figure 123 presents an overview of the simulation environment used to characterize the various single agent and multi-agent behaviors described in this work.

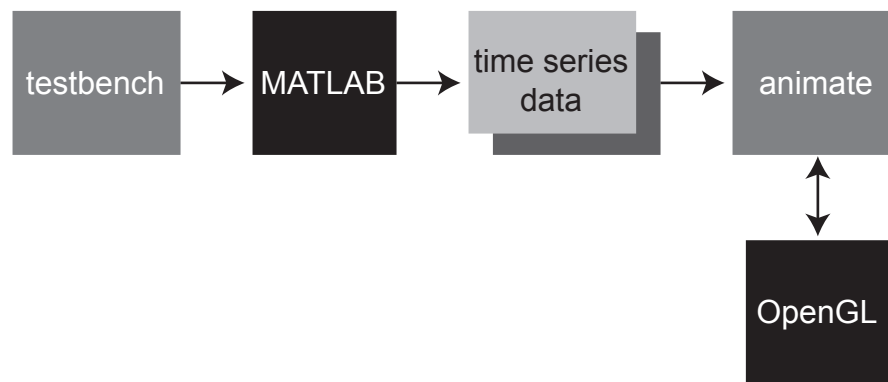


Fig. 123. Overview of the simulation environment used in this work. The dark grey blocks indicate software that was written as part of the work of this dissertation, while the black boxes indicate third-party software and libraries.

A software testbench, illustrated in Figure 124 that specified the coupled agent-environment dynamical system was specified using Simulink. Simulink was primarily used for its graphical block diagramming facilities and, apart from simple mathematical functions and signal conditioning blocks, the testbench was developed using custom s-functions, written using MATLAB’s “M-File” programming language (i.e., no third-party toolboxes were used).

MATLAB was used to simulate the testbench, and the resulting time series of agent position and orientation in the environment was dumped to a file. This file was then input to visualization software (animate), which produced an animation of

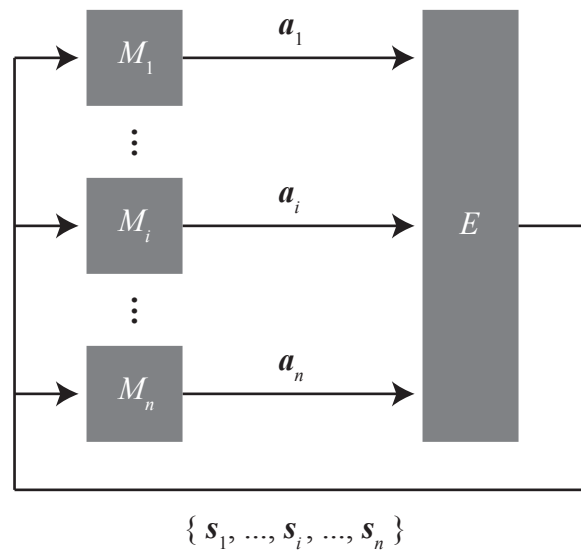


Fig. 124. Structure of the testbench.

the motion of agents in the environment. The visualizer was developed as part of the work of this dissertation, using the rendering facilities provided by the OpenGL (version 1.2) library.

## B. Environment model

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:

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

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

where  $a_v^i$  and  $a_\omega^i$  are the commanded translational and rotational speeds, respectively, of agent  $M_i$ .

## VITA

Nebu John Mathai, P.Eng., received his bachelor's degree in Engineering Science at the University of Toronto in 2000, and his master's degree in Electrical and Computer Engineering in 2004, also at the University of Toronto.

From 2000 to 2003, he worked as a design engineer at Cogency Semiconductor, Toronto, developing several hardware cores for signal processing, encryption, and forward error correction. He commenced his doctoral studies at Texas A&M University in 2004, graduating in 2008. He served as a lecturer in the Department of Electrical and Computer Engineering at Texas A&M for close to three years.

From 2005 to 2008 he held a PGS-D fellowship from the Natural Sciences and Engineering Research Council of Canada, and in 2008 was awarded a U.S. Senator Phil Gramm Doctoral Fellowship. He received the Texas A&M IEEE Students' Choice Award for Best Teaching Assistant in 2005 and the Students' Choice Award for Best Instructor in 2006.

N. J. Mathai may be reached through Dr. Deepa Kundur or Dr. Takis Zourntos at the Department of Electrical and Computer Engineering, Texas A&M University, College Station, Texas, 77843-3128. His e-mail address is mathai@ieee.org.