CONSTRUCTING GEOMETRIES FOR GROUP CONTROL: METHODS FOR REASONING ABOUT SOCIAL BEHAVIORS

A Dissertation

by

BENJAMIN THOMAS FINE

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

DOCTOR OF PHILOSOPHY

Chair of Committee,	Dylan Shell
Committee Members,	Nancy Amato
	Yoonsuck Choe
	Gregory Sword
Head of Department,	Dilma Da Silva

December 2015

Major Subject: Computer Science

ABSTRACT

Social behaviors in groups has been the subjects of hundreds of studies in a variety of research disciplines, including biology, physics, and robotics. In particular, flocking behaviors (commonly exhibited by birds and fish) are widely considered archetypical social behavioris and are due, in part, to the local interactions among the individuals and the environment. Despite a large number of investigations and a significant fraction of these providing algorithmic descriptions of flocking models, incompleteness and imprecision are readily identifiable in these algorithms, algorithmic input, and validation of the models. This has led to a limited understanding of the group level behaviors. Through two case-studies and a detailed meta-study of the literature, this dissertation shows that study of the individual behaviors are not adequate for understanding the behaviors displayed by the group.

To highlight the limitations in only studying the individuals, this dissertation introduces a set of tools, that together, unify many of the existing microscopic approaches. A meta-study of the literature using these tools reveal that there are many small differences and ambiguities in the flocking scenarios being studied by different researchers and domains; unfortunately, these differences are of considerable significance. To address this issue, this dissertation exploits the predictable nature of the group's behaviors in order to control the given group and thus hope to gain a fuller understanding of the collective.

From the current literature, it is clear the environment is an important determinant in the resulting collective behaviors. This dissertation presents a method for reasoning about the effects the geometry of an environment has on individuals that exhibit collective behaviors in order to control them. This work formalizes the prob-

lem of controlling such groups by means of changing the environment in which the group operates and shows this problem to be PSPACE-Hard. A general methodology and basic framework is presented to address this problem. The proposed approach is general in that it is agnostic to the individual's behaviors and geometric representations of the environment; allowing for a large variety in groups, desired behaviors, and environmental constraints to be considered. The results from both the simulations and over 80 robot trials show (1) the solution can automatically generate environments for reliably controlling various groups and (2) the solution can apply to other application domains; such as multi-agent formation planning for shepherding and piloting applications.

DEDICATION

To my mother, Papo and Ma-Ma-Re.

ACKNOWLEDGEMENTS

I would like to express my appreciation and thanks to my advisor Dr. Dylan Shell who has been a tremendous mentor to me. I would like to thank him for encouraging my research and for allowing me to grow as a researcher. His advice on both research; as well as, on my career have been essential to my success.

I would also like to thank my committee members, Dr. Nancy Amato, Dr. Yoonsuck Choe, and Dr. Gregory Sword for serving as my committee members. I also want to thank them for their helpful comments and suggestions throughout this process. I would like to acknowledge Jory Denny for his contributions to the work on the problem formalization and hardness proof in Section 5 and for his support in running the a number of the robot trials.

A special thanks to my family. I am extremely grateful to my mother and grandparents for all of the sacrifices they have made on my behalf through all my years of schooling. Additionally, I would like to thank all of the members of the Distributed Artificial Intelligence Robotics Laboratory for their support and advice over the years. Finally, I would also like to thank Brittany Duncan who is a good friend and valued colleague.

NOMENCLATURE

Microscopic Model Notation

Bold capital roman letter	Denotes a set (*e.g.*, $\mathbf{A}(t)$).
Bold lowercase roman letter	Denotes a vector (*e.g.*, $\mathbf{v}_i(t)$).
$\doteq$	Definition (*i.e.*, $x \doteq y$ should be read as "x is defined as y").
$*$	Denotes a preference (*e.g.*, $\mathbf{d}_{i*}(t) \doteq$ the direction vector from agent i to location $*$ at time t).
$\hat{\mathbf{x}}_i(t)$	Denotes the normalized form of $\mathbf{x}_i(t)$.
$\mathbf{x}_{i^x}(t)$	Denotes the x component of $\mathbf{x}_i(t)$ (We only use this notation for the first three components; x, y, and z).
$\perp(\mathbf{x}_i(t))$	Denotes the vector perpendicular to $\mathbf{x}_i(t)$.
$\angle(\mathbf{x}_i(t))$	Denotes the argument of the vector $\mathbf{x}_i(t)$ (*i.e.*, the angle describing the direction of a vector).
$\mathbf{A}(t)$	The set of all group members at time t.
$\mathbf{D}_i(t)$	The set of all group members detected by the i^{th} group member at time t (*i.e.*, $\mathbf{D}_i(t) \subseteq \mathbf{A}(t)$).

$\mathbf{I}_i(t)$	The set of all group members selected by the perception function (*i.e.*, $\mathbf{I}_i(t) \subseteq \mathbf{D}_i(t)$).
$\mathbf{r}_i(t)$	The position of the i^{th} group member at time t.
$\mathbf{d}_{ij}(t)$	The direction vector from the i^{th} group member to the j^{th} group member at time t.
$\mathbf{v}_i(t)$	The velocity of the i^{th} group member at time t.
$\mathbf{u}_i(t)$	The speed of the i^{th} group member at time t.
$\theta_i(t)$	The orientation of the i^{th} group member at time t (w.r.t. some true/global direction).
$\mathbf{R}_{[\alpha,\beta]}$	The set of all valid distances (*i.e.*, any group member which has a distance $d \in \mathbf{R}_{[\alpha,\beta]}$ would exists in $\mathbf{R}_{[\alpha,\beta]}$).
$L^2(\mathbf{r}_i, \mathbf{r}_j)$	The Euclidean norm between $\mathbf{r}_i(t)$ and $\mathbf{r}_j(t)$ (*i.e.*, $\|\| \cdot \|\|_2$).
w_k^i	The k^{th} gain/constant for the i^{th} group member (if the superscript is omitted then it is a global gain/constant).

Group Control Notation

A	Denotes a homogenous group of mobile agents.
m	Denotes the control-law agent in A obey.
b	Denotes the desired behavior.
E	Denotes an environment which is an enumerable set of primitives.
P	Denotes a finite set of attributes.
$\mathcal{P}$	Denotes the set of possible primitives.
C	Denotes a set of constraints.
W	Denotes an initial workspace.
C_g	Denotes a set of constraints on environment generation.
C_f	Denotes a set of feasible environments.

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iv

ACKNOWLEDGEMENTS v

NOMENCLATURE vi

TABLE OF CONTENTS ix

LIST OF FIGURES xii

LIST OF TABLES xix

1. INTRODUCTION 1

    1.1 Contributions 6
    1.2 Organization of Dissertation 6

2. MOTIVATING CASE-STUDY 8

    2.1 Hamilton's Theory Revisited 9
        2.1.1 The Selfish Herd Hypothesis 9
        2.1.2 Extending HA 11
    2.2 Sensor-based versus Agent-based Detection 13
        2.2.1 The Ideal Case 16
    2.3 Robot Implementation 16
        2.3.1 Implementation Specifics 16
    2.4 Results 17
        2.4.1 Effect of the Exclusion Distance 19
    2.5 Case-study Discussion 21
    2.6 Case-study Summary 23

3. UNIFYING MICROSCOPIC FLOCKING MODELS 25

    3.1 Scope 26
    3.2 Data-flow Template 27
        3.2.1 The Five Stages: Definitions 29

3.2.2 The Five Stages: Selected Literature . . . 32
3.3 Current Microscopic Models . . . 37
3.3.1 Definitions of Design Attributes . . . 41
3.3.2 Position, Velocity, and Orientation . . . 44
3.3.3 Observations of Chosen Design Attributes . . . 44
3.4 Specification of Flocking Models . . . 47
3.4.1 Literature Omitted from Model Specification . . . 47
3.5 Validation Methods . . . 51
3.5.1 Definitions of Attributes for Validation Methods . . . 54
3.5.2 Observations of Chosen Validation Methods . . . 56
3.6 Discussion of Unifying Microscopic Flocking Models . . . 58
3.6.1 Collision Avoidance in the Data-flow Template . . . 58
3.6.2 Neighbor Identification . . . 60
3.6.3 Association . . . 61
3.6.4 Data-Centric Approach to Determining Parameter Values . . . 61
3.6.5 Flocking Behaviors are Independent of Information Types . . . 62
3.7 Summary of Model Unification . . . 62

4. MANY MICROSCOPIC MODELS YIELD SIMILAR BEHAVIORS . . . 63

4.1 Local Crowded Horizon . . . 64
4.2 Robotic Implementation Specifics . . . 69
4.2.1 The Effect of a Limited Field of View . . . 70
4.3 Information-abstracted Flocking . . . 72
4.4 Case-study Summary . . . 74

5. UNDERSTANDING GROUP BEHAVIORS VIA CONTROL . . . 76

5.1 Group Control with External Agents . . . 76
5.1.1 Shepherding . . . 76
5.1.2 Caging . . . 78
5.2 Group Control with Internal Group Members . . . 78
5.3 Group Control via the Environment . . . 79
5.4 Generating Environments for Group Control: Problem Definition . . . 82
5.5 k-MAEBE Variant . . . 84

6. APPROACH AND IMPLEMENTATION TO GROUP CONTROL . . . 85

6.1 Environment Generation . . . 86
6.1.1 Shape Grammars . . . 86
6.1.2 Computational Schemas . . . 88
6.1.3 Hierarchical . . . 89
6.2 Environment Validation . . . 90
6.3 System Implementation . . . 91

6.3.1 Shape Grammars . . . 91
6.3.2 Predefined Behaviors . . . 91
6.3.3 Microscopic Models . . . 93

7. EXPERIMENTS AND RESULTS . . . 95

7.1 Hardness of MAEBE . . . 95
7.2 System Validation: Simulation Results . . . 98
7.3 System Validation: Robot Results . . . 100
7.4 Evaluating the Set of Generated Constructs . . . 103
7.4.1 Environment Generation . . . 105
7.4.2 Empirical Mapping Results . . . 106
7.4.3 Understanding the Space of Constructs . . . 112

8. DISCUSSION . . . 114

8.1 Finding the Right Primitives . . . 114
8.1.1 Environment Decomposition . . . 114
8.1.2 *De novo* Primitives . . . 116
8.1.3 Bottom-up versus Top-down . . . 118
8.2 Constructs for Multi-Robot Formations . . . 120

9. CONCLUSIONS . . . 128

9.1 Microscopic Flocking Models . . . 128
9.1.1 Recommendations from Unification Tool-Set . . . 129
9.1.2 Directions for Future Flocking Model Investigations . . . 131
9.2 Group Control . . . 132
9.2.1 Directions for Future Group Control Investigations . . . 132

REFERENCES . . . 134

APPENDIX A. MICROSCOPIC FLOCKING MODEL SPECIFICATIONS . 148

APPENDIX B. ALGORITHM PROOFS . . . 163

LIST OF FIGURES

FIGURE Page

1.1 Figure (a) shows a *Judas goat* influence a group of sheep. Figure (b) shows a school of fish being caged (or corralled) by another. Figure (c) shows a fishing weir designed to corral schools of fish. Figure (d) shows a cattle pen design to assist in the transportation of livestock. 4

1.2 Two time series showing the effect an environment can have on a group. These robot trials show that a single robot and a group of robots running the same motion model can exhibit different behaviors given a particular environment. 5

2.1 Detailed pictorial description of the extended HA model. The first and last frames show the effect of the exclusion distance on the selection of group members for way-point calculation and the middle frame shows the HA way-point calculation if two group members are selected. . . 12

2.2 The data flow from raw sensor data to the control-law input using member-based detection. For the implementation, the constraint is a *a priori* shape database and the *Convert to Group Member* step averages the foreground pixels to single (x,y)-coordinates for each detected individual. 13

2.3 The data flow from raw sensor data to control-law input using sensor-based detection. For the implementation, the constraints for each sensor reading is a laser intensity threshold. 14

2.4 A single robot wrapped in high reflectance material for the fiducial and equipped with a Hokuyo URG-04LX-UG01 scanning laser rangefinder and an ASUS Eee PC. 17

2.5 Starting formations for six robots. 18

2.6 A single trial of member-based detection with an exclusion distance of 0. Marker **B** shows the issue of FN for member-based input. . . . 20

2.7 A single trial of sensor-based detection with an exclusion distance of 0. Marker **B** shows the robustness to FN. 20

2.8 A single trial of sensor-based detection with an exclusion distance of 0.5. 21

2.9 A single trial of sensor-based detection with an exclusion distance of 1. 21

2.10 The data flow from raw sensor data to control-law input using member-extrapolated detection. For the implementation, the constraints are the same as in member-based detection, but the foreground pixels are not averaged to single (x,y)-coordinates. 22

3.1 A diagrammatic representation of the proposed DT for microscopic flocking models. It details the main aspects for the generation of flocking behaviors via the five boxes (stages). The connections between the stages encode the data that propagates between them. In particular, the connections between the sensing stage and group member detection stage represents the raw sensor information from each sensor (*e.g.*, laser range-finder, camera, GPS). The connection between the group member detection and neighbor selection stage is the set of detected group members ($\mathbf{D}_i(t)$). The neighbor selection stage passes at least one set of selected group members ($\mathbf{I}_i(t)$) to the motion computation stage which passes the next computed motion to the physical motion stage. 28

3.2 Specific examples of data-flow between the various stages in the DM. Note that the examples listed here are from a combination of various publications. There is no known example of a publication that unambiguously describes the data-flow between all stages. 38

3.3 The modified DT that includes the obstacle detection and obstacle selection stages. These two stages allow for the addition of obstacle avoidance behaviors and the generation of virtual group members. The connection at ① can be treated as a place holder for adding a method that generates the required virtual group members, as in (Olfati-Saber, 2006). 59

4.1 The four variations of the LCH flocking algorithm used in this study. Variations 4.1a and 4.1b require pose information where 4.1c and 4.1d use bearing information. Variations 4.1b and 4.1d utilize all of the detected group members where 4.1a and 4.1c only use a subset of the detected group members. 65

4.2 Each time series shows the motions of the multi-robot system running one of the four LCH variations. 70

4.3 Figure 4.3a shows that a group member will continue in the same direction when no other group members are observed. This behavior causes the resulting motion of the group to be more directional than the computer simulations in Viscido et al. (2002). Figure 4.3b shows how the follow the leader behavior is generated when there are only a few detected neighbors. 71

4.4 These two motion figures were generated from 50 simulated group members using the member-centric pose variation of the LCH. The blue squares represents the starting formation, which was randomly generated within a squared region, and the green squares represent the ending formation of the group members. 71

4.5 These motion figures are typical results of simulations conducted for this study. Each trial simulates 50 group members over 75 iterations (only plotting every third pose per group member), where st_{distance} is set to 0.5 units and st_{angular} is set to 10 degrees. The top row of motion figures (figures 4.5a,4.5b, 4.5c, and 4.5d) were generated using group-centric information and the bottom row of figures (figures 4.5e, 4.5f, 4.5g, and 4.5h) were generated using member-centric information. The blue squares represent the starting positions of the group members, and the green squares represent the end positions of the group members. Motion figures 4.5e, 4.5f, 4.5a, and 4.5b were simulated with the single-group starting formation and a sensing range of 3000 units, where figures 4.5g, 4.5h, 4.5c, and 4.5d were simulated using the split-group starting formation and a sensing range of 3 units. 73

5.1 These simulation results were generated using identical parameters with exception to group size. Together these results show how the corralling behavior is only elicited when the group is of sufficient size (Figure 5.1c). 81

6.1 A diagram showing the proposed method for automatically enumerating a set of environments that elicit a particular behavior from a group. The environment schema (C_g), group (A), and behavior (b) are all user defined inputs to the system and remain constant during the system's execution. The output is a set of environments (E) that can elicit the specified behavior from the given group. 85

6.2 Three common rules used in shape grammars. Here, the straight line and the square would be considered primitives. 88

6.3 Pictorial representation of a Galton board (the pegs arranged in a triangle) along with a funnel and bins for containing the *agents* . . . 89

6.4 A simple execution of the splitting grammar. Step 0 is the starting shape and a non-terminal. The only primitives that are physically realized are the line primitives. 92

6.5 The splitting grammar uses a non-terminal shape (square), which is used to help define a tree-like structure of the environment and a single terminal shape (straight line). The weiring grammar uses one terminal shape (straight line) and one marker (cross) that defines the predesignated area for the corralling behavior. 92

6.6 Motion models used in this work. The gray circle centered on the group member is the interaction radius for selecting neighbors. 94

7.1 Reduction from a Warehouseman's problem instance (a) to an instance of the Multi-Agent Environmental Behavior Elicitation problem (b). Each piece-wise linear tunnel from a start to goal position is equivalent to a time-varying trajectory of a rectangle in the Warehouseman's problem. 97

7.2 Both figures show three environments that were generated using the splitting and corralling shape grammars and that passed the respective filters defined in Table 7.1. 99

7.3 Both figures show environments that were generated using the corralling shape grammar but *failed* the corralling filter. 100

7.4 Three different environments that successfully elicit the simple split behavior for each motion model. 100

7.5 Three different environments that successfully elicit the corralling behavior for each motion model. 101

7.6 These two time series shows a multi-robot system obeying the SNN+ motion model being influenced by environments that were automatically generated by the implemented system. 102

7.7 Motivating examples of how small perturbations in the environment can effect the exhibited behaviors of a group. Figure 7.7a shows the effect the workspace has on the trajectories of the simulated agents, where Figure 7.7b and 7.7c show examples of how small changes can drastically change the groups' exhibited behavior. The desired action for these motivating examples was influencing at least 90 percent of the group to turn right at the junction. Shading of the trajectories represent time starting with white and fading to black. 104

7.8 Figures 7.8a and 7.8b represent the two computational schemas. The workspace is the solid black line where the dotted line represents a possible construct. The labels a and l represent the various angle and length parameters. 106

7.9 An example environment generated by the DIRT method from the fourth anchor position with a starting formation for seven agents. The thin-blue line represents the goal line for the turn right behavior. . . 106

7.10 Parameter space map for the DIRT method on a group of seven agents obeying BOIDS from the four anchor locations. 108

7.11 Parameter space map for the DIRT method on a group of seven agents obeying BOIDS from the fourth anchor position. 109

7.12 Parameter space map for the DIRT method on a single agent obeying BOIDS from the fourth anchor position. 110

7.13 Parameter space map for the DIRT method on three agents obeying BOIDS from the forth anchor position. 111

7.14 Represents the fitness values of 1,024 different parameter configurations from the four dimensional parameter space of the DSDR schema. All results are for a group of 10 agents obeying the BOIDS flocking model. 112

8.1 Parameter space map for the SIRT schema on a group of seven agents obeying BOIDS from the fourth anchor position. The probabilities of the three rules in the SIRT method are all dependent on each other; therefore the parameter space is projected on the rule 1 and rule 2 dimensions. 118

8.2 The left figure shows an environment that was generated via the presented system that is eliciting the corralling behavior from a group of agents. The right figure shows the same agents being controlled by a set of point (robots) that are in a formation based on the generated environment. 121

8.3 Additions to the presented system for use in formation design investigations and applications. 122

8.4 The four task used in the multi-robot formation trials. 123

8.5 Example solutions generated by the presented system for all four task. The solid yellow lines represent the physical construct generated by the system. 124

8.6 Pictorial representation of a single trial for *Task 3* for six shepherding robots (red circles) and a target group of five robots (green circles). The desired task is to guide the target group from the blue circle (dark grey) to the green circle (light grey) while following the given trajectory (black dotted line). Black lines represent the trajectories of the robots while the yellow solid lines represent the physical constructs the shepherding robots are using as formation templates. 125

8.7 Time series of two successful multi-robot trials for *Task 1*. Time series (a) shows four robots following the BOIDS motion model eliciting *Task 1* in a static environment. Time series (b) shows four robots following the BOIDS motion model eliciting *Task 1* in a static environment with six mobile piloting robots. 126

8.8 Time series of two successful multi-robot trials for *Task 2*. Time series (a) shows four robots following the BOIDS motion model eliciting *Task 2* in a static environment. Time series (b) shows four robots following the BOIDS motion model eliciting *Task 2* in a static environment with six mobile piloting robots. 126

8.9 Time series of two successful multi-robot trials for *Task 3*. Time series (a) shows four robots following the BOIDS motion model eliciting *Task 3* in a static environment. Time series (b) shows four robots following the BOIDS motion model eliciting *Task 3* in a static environment with six mobile piloting robots. 127

8.10 Time series of two successful multi-robot trials for *Task 4*. Time series (a) shows four robots following the BOIDS motion model eliciting *Task 4* in a static environment. Time series (b) shows four robots following the BOIDS motion model eliciting *Task 4* in a static environment with six mobile piloting robots. 127

B.1 Example grammar with equal rule weights (r1 and r2) where $\mathcal{S}$ is the starting non-terminal. All strings of a have a probability of being generated from this grammar. 164

LIST OF TABLES

TABLE Page

3.1 Descriptions of perception functions found in the selected literature. Most neighbor selection stages could be created using a combination of the functions listed here, but these may not be the only possible functions. 31

3.2 A categorical review of the information and group member requirements for each of the selected models. This table can be used as a first pass to identify which models could be useful when designing a new model or when identifying models that may benefit from additional investigations. Entries marked with a ($\surd$) signifies that the particular attribute is utilized. For the collision avoidance attribute, (Member) signifies that the model only considers member to member collision avoidance, while (All) signifies that the model considers both member and environment collision avoidance. 39

3.3 Both entries in this table are possible implementations of the control-law in Reynolds (1987). The two implementations only vary slightly; however, the differences have effects on the complexity of the group members and the underlying causes of flocking behaviors. 50

3.4 Details the validation methods chosen for the selected motion models. The form of validation employed varies significantly within the publications. 52

4.1 The MMD (in units) from all of the simulations that had a sensing range of 3 units. For each simulation, the median distance from the center of all 50 group members is calculated from the ending formation. The medians from all ten trials were averaged to yield the MMD for the given parameterization. 74

6.1 Specifications of the behaviors used in this study. 93

7.1 Definitions and percentages of environments that passed the filtering process. 99

7.2 The percentage of environments that generated the desired behaviors for the three motion models. These percentages only consider the number of environments that passed the filtering process and are rounded to the nearest hundreth. 101

7.3 Validation results for two environments for *robotic* agents. Each behavior was tested with one environment selected from the set of valid environments and conducted over five trials, totaling 30 trials. The simple split behavior was tested using four agents and the corralling behavior was tested using three. 102

7.4 Validation results for two environments for *simulated* agents. Each behavior was tested with one environment selected from the set of valid environments and simulated over five trials, totaling 30 simulation trials. The simple split behavior was tested using four agents and the corralling behavior was tested using three. 102

8.1 Average fitness values over the robot trials for all four task. Each trial had a group of four autonomous agents following the BOIDS motion model. For the case when the static primitives where approximated, six mobile agents where used. Each trial scenario was run a total of 10 times. 125

A.1 The translation of the neighbor selection and motion computation stages from the selected flocking models. Together, these two stages constitute the control-law, which is the primary focus of the vast majority of the literature. The motion rules presented in this table have been translated into the common notation (Table 1) presented earlier in this study. The neighbor selection column details the required perception functions (Table 3.1) along with what set(s) of neighbors will be considered. The motion computation column details the low-level control law, or algorithm, which computes the next motion of the flock member. 149

1. INTRODUCTION

Flocking behaviors are some of the most commonly observed collective spatial behaviors displayed by groups. These behaviors have been extensively studied for decades in multiple research communities, including biology (Aoki, 1984; Ballerini et al., 2008a; Bazazi et al., 2008; Couzin et al., 2005; Giardina, 2008; Gueron et al., 1996; Hamilton, 1971; Viscido et al., 2002), physics (Csahók and Vicsek, 1995; Czirók and Vicsek, 2000; Vicsek et al., 1995), robotics (Arkin and Balch, 1999; Codling et al., 2007; Desai et al., 1998; Ferrante et al., 2012; Hauert et al., 2011; Lien et al., 2004; Vaughan et al., 2000), computer graphics (Funge et al., 1999; Reynolds, 1987), transportation science (Bender and Fenton, 1970; Helbing and Molnár, 1995; Helbing et al., 2005), and mathematics (Babak et al., 2004; Blomqvist et al., 2012; Edelstein-Keshet, 2001; Mogilner and Edelstein-Keshet, 1999). Although some works have expressed confidence in our comprehension of this phenomenon (Goldstone and Janssen, 2005), understanding of flocking behaviors remains incomplete (Giardina, 2008; Hildenbrandt et al., 2010; Lopez et al., 2012; Parrish et al., 2002; Vicsek and Zafeiris, 2012). The focus of this dissertation is to:

1. show study of the individual group member is not adequate for complete understanding of the group level behaviors;

2. introduce a novel approach for reasoning about these groups through adeptly constructed geometries (environments).

The current literature does provide several candidate models to explain this phenomena (*e.g.*, Hauert et al. (2011); Helbing et al. (2000); James et al. (2004); Moussaïd et al. (2009); Okubo (1986); Rauch et al. (1995); Warburton and Lazarus

(1991); Wood and Ackland (2007)). However, there is no consensus on the precise details of the motions needed to produce rich flocking behaviors under realistic sensing models, actuation, and dynamics constraints. This stems, in part, from a poor definition of what constitutes a flocking behavior.

Although diverse research communities study different varieties of this problem and questions surrounding the phenomenon (Bender and Fenton, 1970; Clark and Evans, 1954; Dingle and Drake, 2007; Edelstein-Keshet, 2001; Emlen, Jr., 1952; Hutto, 1988; Miki and Nakamura, 2006; Parrish, 1989; Parrish and Edelstein-Keshet, 1999; Partridge, 1982; Pitcher et al., 1976; Rands et al., 2004; Simons, 2004; Viscido and Wethey, 2002; Whitfield, 2003), the vast majority of the flocking literature aims at bottom-up production of flocking motions (Goldstone and Janssen, 2005; Parrish et al., 2002) focusing on the behaviors of the individual group members. Generally, studies are reported without explicitly detailing the sensing capabilities, limiting assumptions, and/or computation capabilities of the individuals. Additionally, there is currently no common or accepted method for the design, validation, and/or presentation of flocking models (Parrish et al., 2002; Vicsek and Zafeiris, 2012), which makes it difficult to determine the current state of the literature and to compare existing models.

To show a microscopic approach (alone) is not adequate for reasoning about these groups, this dissertation first addresses the limitations in the current literature with an extensive meta-study of 32 publications selected from over 100 publications reviewed. These publications were carefully chosen to maximize coverage of the common design choices and assumptions found throughout the literature and to be a representative cross-section of the literature as a whole.

The meta-study reveals the current presentation of proposed flocking models lack either completeness, precision, or both, which significantly hinders repeatability and

understanding. In this work, **completeness** refers to how many of the *key* aspects of the flocking model are presented and **precision** (or lack thereof) is the quality of the specification/presentation of the various aspects of the model. Additionally, there are small (sometimes subtle) implicit and/or explicit assumptions that are currently overlooked (examples which have been teased out include: type of member detection required, lack of occlusions, and perfect sensing), which may impact the produced motions.

Furthermore, even models that are completely and precisely presented are not always realized exactly when validated (*e.g.*, a model designed for local sensing might be validated using global sensing). Therefore, it is difficult to know the degree to which a particular model is actually capable of producing flocking behaviors in a practical scenario or if the model's assumptions are realistic. To address the limitations in an microscopic approach to understanding groups that exhibit collective structure, this dissertation explores methods for reliably controlling such groups.

The ability to control groups has been the focus of much interest and study from the early history of animal management to recent investigations (Erickson, 2000; Grandin, 1980; Halloy et al., 2007; Helbing and Molnár, 1995; Lien et al., 2005; Petersen et al., 1994; Rodriguez et al., 2012b; Zheng et al., 2009). Numerous techniques have been developed and studied, ranging from the use of external agents (better known as shepherding (Lien et al., 2004, 2005; Vaughan et al., 2000)) to heterogeneous groups, where a subset of the individuals in the group have extra knowledge or goals (Conradt et al., 2009; Couzin et al., 2005; Gueron et al., 1996); see Figures 1a and 1b. In addition to using other individuals to control these groups, observations and empirical studies (Bobadilla et al., 2012; Butler et al., 2006; Despland et al., 2000; Umstatter, 2011; Vaughan et al., 2000) have shown the environment in which these groups operate is an important determinant of the resulting behav-

ior (e.g., locust following vegetation patterns, cattle herded by fencing). Thus, the exhibited behaviors of any group is a combination of the inter-member interactions and the interactions each individual has with the local environment. This can be seen in both Figures 1c, 1d.

(a) (b) (c) (d)

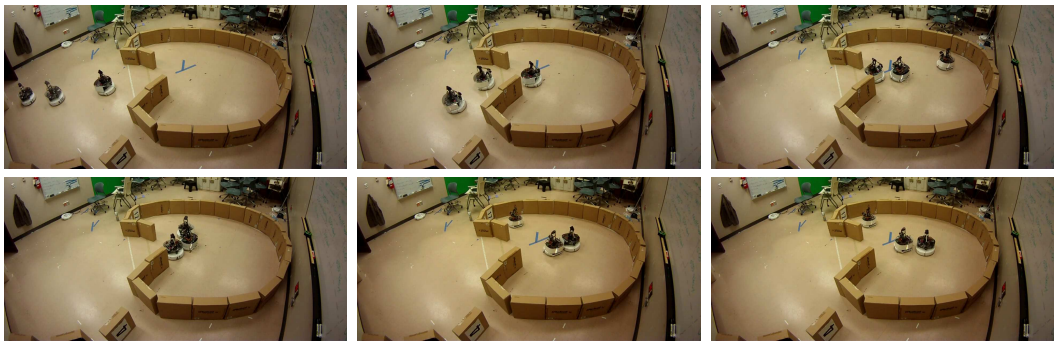
Figure 1.1: Figure (a) shows a *Judas goat* influence a group of sheep. Figure (b) shows a school of fish being caged (or corralled) by another. Figure (c) shows a fishing weir designed to corral schools of fish. Figure (d) shows a cattle pen design to assist in the transportation of livestock.

Corralling cattle with fencing (Grandin, 1980), controlling pedestrian flows with turnstiles (Helbing and Molnár, 1995), and shaping robot formations with obstacles (Becker et al., 2013) are just three examples of groups being controlled by adeptly constructed environments (or constructs *). While specialized barriers, passive mechanism, and carefully shaped obstacles are useful in a variety of applications (Figure 1.2), comparatively little is known about this family of physical constructs as a general class of apparatus for controlling (or influencing) collective behavior. This work is interested in understanding how one could reason about the effects the geometry of an environment has on groups and use this to automatically generate environments that can control a target group. In particular the focus is on environments that induce a particular group to perform a desired task, and an au-

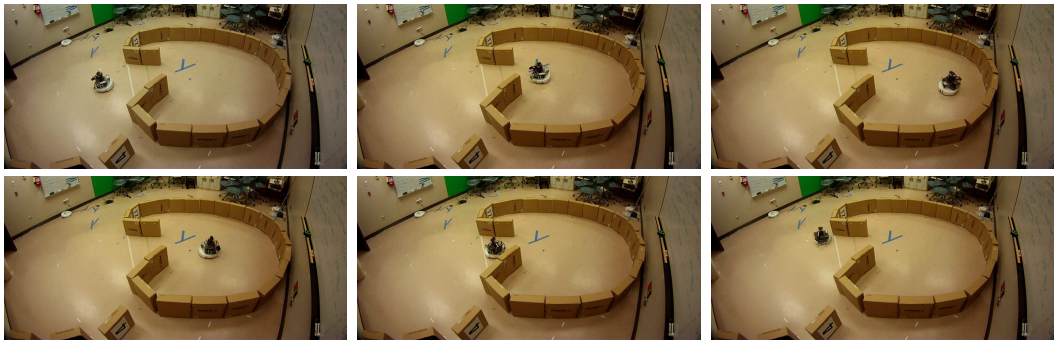
*See Section 5 for the differences between environment and construct

tomated process for searching, planning over, and generating the physical structures or constructs which comprise such environments is explored.

This dissertation further shows being able to reason about these constructs and their effects on group behavior can apply to other methods of group control. Currently, much of the literature in shepherding formations lacks a rigorous method for developing these formations for reliable control of the target group. This investigation shows how these generated constructs can be used to inform multi-agent formations for use in shepherding and piloting applications.



(a) Group of robots following the BOIDS motion model being corralled by static environment.



(b) Single robot following the BOIDS motion model failing to be corralled by static environment.

Figure 1.2: Two time series showing the effect an environment can have on a group. These robot trials show that a single robot and a group of robots running the same motion model can exhibit different behaviors given a particular environment.

1.1 Contributions

The contributions of this dissertation are as follows.

1. A detailed study of the current flocking model literature that highlights subtle (but important) inconsistencies and ambiguities (Section 3).

2. A tool-set that will allow for more complete and precise studies of flocking models (Section 3).

3. Support for flocking models that produce flocking behaviors that are agnostic to the resolution and type of information given to the control algorithm (Section 4).

4. Compares (for the first time) the effects different types of information have on the produced flocking motions through the implementation of a single algorithm (Section 4).

5. Shows the comparison of one flocking algorithm with another purely on the basis of the motion they produce is inadequate (Sections 2, 3, and 4).

6. Formalizes the problem of generating environments to control group behavior and proves the hardness of this problem to be PSPACE-Hard (Sections 5 and 7).

7. Introduces and validates a methodology and implementation to solve the above problem (Sections 6 and 7).

1.2 Organization of Dissertation

The remainder of the dissertation is organized as follows. Section 2 is a detailed motivating example of how current literature contains subtle ambiguities that lead

to a limited understanding of the exhibited group behaviors. Additionally, the multi-robot system used for all experiments in this dissertation is presented.

The detailed meta-study of the current microscopic flocking model literature is presented in Section 3. This section introduces a tool-set that, if used, will reduce incompleteness and imprecision in future investigations. Section 4 is a case-study that will show the study of the individual is not adequate for complete understanding of the group, and thus motivation for the remainder of work in this dissertation.

The scope of the group control work is set by highlighting the related literature to group control in Section 5. Additionally Section 5 formalizes the problem of controlling groups with automatically generated environments. Section 6 then presents the approach and implementation of the system introduced in this work. In addition to the specific implementation chosen for this dissertation, Section 6 outlines other possible methods and compares them to the chosen methods. Theoretical hardness proof, computer simulations, and robot experiments that support the presented solution can be seen in Section 7.

Section 8 discusses an important limitation in the presented approach and presents methods for lifting the assumption. Additionally, Section 8 highlights the generality and applicability of the methodology by using the generated constricts for multi-robot formations for group control applications. This dissertation will conclude and present recommendations for future studies of microscopic flocking models and avenues for future investigations will be presented in Section 9.

2. MOTIVATING CASE-STUDY *

Still today, questions regarding the mechanisms by which flocking behaviors are formed and why they persist remain far from entirely resolved. This case-study illustrates the delicate process of translating a prose description, which is common in much of the literature, into an algorithmic model of flocking behaviors. Through a study and robot implementation of a flocking model first proposed by Hamilton in 1971 (Hamilton, 1971) this study explores the effects different forms of perceptual input have on the exhibited behaviors.

Here, **perception** is the process of taking raw sensor information and converting it into usable input for the control-law. The control-law governs the individual's behavior and, therefore, understanding the preceding perceptual processing is vital for understanding the resultant behavior. For clarity, when this work refers to the **focal member**, it refers to the member-centric reference frame of a single individual running the given model.

The majority of the flocking literature neither considers nor reports the process of converting the raw sensor data into usable input. As a consequence, several models employ raw sensor data which is unavailable or unrealistic for a simple individual (*e.g.*, a robot, bird, ant), oftentimes ignoring uncertainty from noise or occlusion. This case-study directly focuses on sensory processing to highlight the need for a unified description of flocking models. Three different perception functions are introduced and their implications for the underlying models are discussed. The three functions are (1) member recognition, (2) member-extrapolated and (3) sensor-based.

The results from computer simulations and robot trials support the following relationships:

- With perfect sensing the behaviors produced by the three functions are equivalent.
- Member-extrapolated is similar to member-based detection due to the constraints on the sensor data.
- Sensor-based input is more sensitive to false positives (FP) where member-based and member-extrapolated are sensitive to false negatives (FN). Where a FN is when the focal member fails to detect a group member and a FP is when the focal member detects a group member which does not exists.

In addition, the results of this case-study suggest the resulting behaviors from a given model are essentially independent of the perception function used (under certain parameter constraints).

2.1 Hamilton's Theory Revisited

2.1.1 The Selfish Herd Hypothesis

Many theories attempt to explain the causes and mechanisms of flocking (Barbosa, 1995; Buhl et al., 2006; Hamilton, 1971; Simons, 2004), but no single theory has had complete success. The most commonly discussed theory is Hamilton's *Selfish Herd Hypothesis* (Hamilton, 1971). From field observations, Hamilton suggests flocking behaviors emerge due to a selfish need for survival, and individuals do not have a notion of the collective, but only a *sense* for individuals within a certain field of view. In this case, the field of view is a radius from the given group member.

Hamilton (1971) presents the Hamilton (HA) and the Simple Nearest Neighbor (SNN) flocking models. In the HA model, an individual selects the nearest group

member (**nearest neighbor**) and then selects the nearest group member to the nearest neighbor. The individual will then select a way-point which is two body-lengths (**repulsion distance**) away from the nearest neighbor, which is on the line between the two selected group members. If the nearest neighbor is within the repulsion distance of the focal member, the way-point will be perpendicular (sign chosen at random) to the line between the two selected group members. In the SNN rule, the individual will move directly towards the nearest neighbor. If the nearest neighbor is within the repulsion distance, the focal member will not move.

Algorithm 1 Hamilton's original motion rule

Input: List of inputs (I) in a robot-centric coordinate frame, where $|I| > 1$.
Parameters: $r :=$ *repulsion distance*
Output: Way-point in robot-centric coordinate frame.

```
1: a_1 ← min(∀i ∈ I, dist(i, (0, 0)))
2: a_2 ← min(∀i ∈ I, dist(i, a_1)
3: if dist(a_1, (0, 0)) > r then
4:     return ComputeHAWayPoint(a_1, a_2)
5: return ComputePerpendicularWayPoint(a_1, a_2)
```

Hamilton supported these models by showing, through computer simulation, the group formed densely packed clusters from random starting formations and discussed how these two models are biologically feasible. In his simulations, the group members are homogeneous holonomic agents starting in random positions within the environment. At each simulation step, all group members would synchronously compute their next way-point and move one body-length in that direction. It must be noted, the HA rule was validated in one dimension, while SNN was validated in two.

Unfortunately, Hamilton's proposed flocking models fails to exhibit the same behaviors the models were originally motivated by. In the observed biological agents (Hamil-

ton, 1971), the group members formed one large centrally compact group and in Hamilton's simulations the group formed many small compact subgroups. From the simulations, Hamilton suggested there must be a group level aggregation behavior not covered by the proposed models. Hamilton proposed the smaller groups would then move toward the nearest group, producing a single compact group.

2.1.2 Extending HA

The original description of the HA model assumes the focal member will always have at least two other group members to calculate the next way-point. In a physical system, this assumption is not guaranteed to hold (*e.g.*, due to occlusions created by the environment and other group members). To account for this, the SNN way-point should be computed when only one group member is detected. To justify this additional behavior interpretations of Hamilton's work (Hamilton, 1971) must be discussed. Lines 1 and 4 in Algorithm 2 show the modifications to Algorithm 1.

Algorithm 2 Extended Hamiltonian motion rule

Input: List of inputs (I) in a robot-centric coordinate frame.
Parameters: $r :=$ *repulsion distance*
$ed :=$ *exclusion distance*
Output: Way-point in robot-centric coordinate frame.
1: **if** $|I| = 0$ **then**
2: **return**
3: **if** $|I| = 1$ **then**
4: **return** ComputeSNNWayPoint(I)
5: $a_1 \leftarrow \min(\forall i \in I, dist(i, (0, 0)))$
6: $a_2 \leftarrow \min(\forall i \in I, dist(i, a_1))$
7: **if** $\nexists i \in I$, where dist(a_1, a_i) $> ed$ **then**
8: **return** ComputeSNNWayPoint(a_1)
9: **if** dist$(a_1, (0, 0)) > r$ **then**
10: **return** ComputeHAWayPoint(a_1, a_2)
11: **return** ComputePerpendicularWayPoint(a_1, a_2)

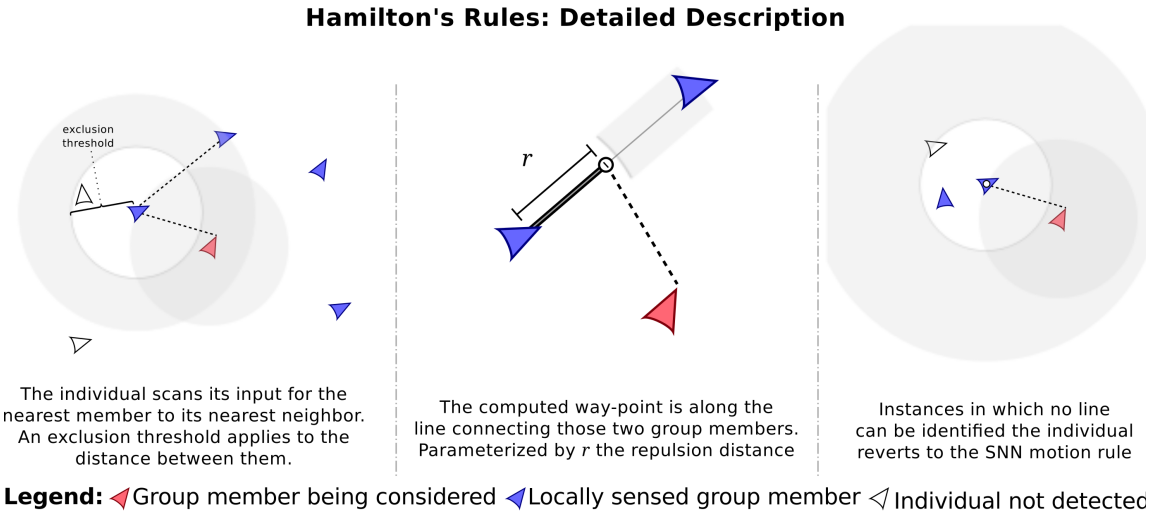


Figure 2.1: Detailed pictorial description of the extended HA model. The first and last frames show the effect of the exclusion distance on the selection of group members for way-point calculation and the middle frame shows the HA way-point calculation if two group members are selected.

A common interpretation of Hamilton's work is each group member attempts to individually decrease their chances of predation. Using this interpretation alone, the additional SNN behavior to the HA motion rule cannot be justified. One may argue the SNN represents a *searching* behavior and, it stands to reason, an individual has a higher probability of finding a group by following the sole detected group member.

However, it is unclear whether a particular individual is concerned with or even aware of its current predation risk. The observed behaviors seen in many animals, including red tail deer (Hamilton, 1971), suggest the members within the perimeter of the group all have smaller predation risk than those on the perimeter. If the group members only *desire* to be within the perimeter of the group, then the addition of the SNN behavior to the HA motion rule is justified.

For completeness, the case when the focal member does not detect any other individual is also considered. Here, the group member maintains the same motor

commands as the previous computation. Figure 2.1 is a pictorial representation of the extended version of the HA rule.

2.2 Sensor-based versus Agent-based Detection

Member-based input involves representing the location of each detected group member with a single point in space. Another method of translating sensor data to input for a control-law is sensor-based detection. Sensor-based input is every sensor reading which corresponds to an group member in the field of view.

In **member-based detection** an individual has an *a priori* description of a group member that is used to distinguish foreground pixels from background pixels. **Foreground** pixels are any pixels in raw sensor data which will be used in the classification of a group member where **background** pixels are all other pixels (e.g., environmental features). Figure 2.2 shows the data flow from the raw sensor input to the input to the control-law for member-based detection. The raw sensor data are first separated into foreground and background pixels based on the *a priori* description of an individual. Then the foreground pixels are averaged together to a single location for each detected group member.

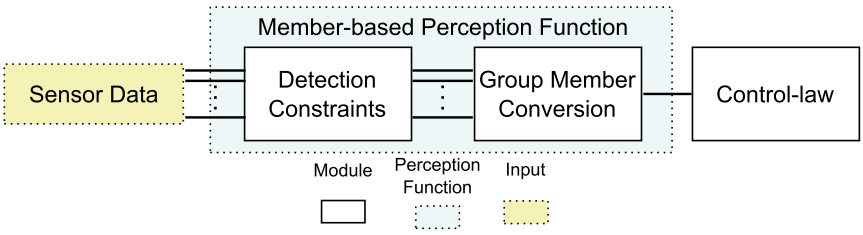


Figure 2.2: The data flow from raw sensor data to the control-law input using member-based detection. For the implementation, the constraint is a *a priori* shape database and the *Convert to Group Member* step averages the foreground pixels to single (x,y)-coordinates for each detected individual.

In **sensor-based detection** an individual does not have an *a priori* description of a group member. The only information given to the individual is an intensity threshold used to identify foreground pixels from background pixels. Figure 2.3 shows the data flow from the raw sensor input to the input to the motion control-law for sensor-based detection. The raw sensor data are classified as high or low intensity pixels; foreground and background pixels respectively. Then the locations of the foreground pixels are passed directly to the control-law.

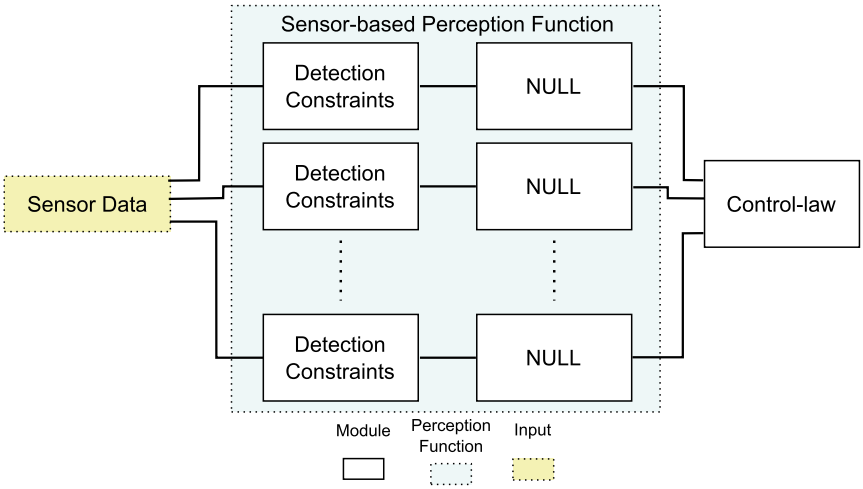


Figure 2.3: The data flow from raw sensor data to control-law input using sensor-based detection. For the implementation, the constraints for each sensor reading is a laser intensity threshold.

The initial set of experiments using the sensor-based detection exhibited SNN behavior and not the HA behavior as expected because of an implicit assumption built into the HA flocking model. In some cases, multiple sensor readings may result from the same spatially extended group member. Algorithm 1 does not consider the case when the two chosen inputs are consecutive (or nearby) sensor readings, because

agent-based detection enforces a separation distance between inputs. However, in sensor-based detection, the chosen HA way-point can be observed to be the same as the SNN way-point because the selected inputs are very nearby readings.

In the original HA rule, there are no distance constraints on the selection of the group member nearest to the nearest neighbor. Neither Hamilton's or Viscido's work explains what occurs when the distance between the two selected individuals is less than the repulsion distance. If the focal member's *intent* is to find a location which will minimize its predation risk, should it not select a position which is feasible? That is, a position which has no group members within the repulsion radius. For the implementation the **exclusion distance** is used to find two group members which are far enough apart for a feasible point to exist on the line between the two selected individuals.

The necessity of a way-point being feasible depends on the mobility and sensing frequency of the group members. If the individual will arrive at the way-point before the next sensing cycle, then the way-point should be a feasible location. However, if the individual cannot achieve the target way-point before the next sensor reading, then one could argue the exclusion distance is superfluous for individuals in continuous motion. If the computed way-point is always more than one sensation away, then the focal member will never arrive at the way-point, making the feasibility argument moot.

Now consider when the focal member may reach the way-point before the next sensation. The way-point must be a feasible point in the focal member's configuration space, but should the way-point put the focal member in a location where the nearest group member is within the repulsion radius? Under the assumption of constant motion, it is impossible to select a way-point which is feasible in the configuration space while maintaining the repulsion distance. For this reason one could argue the

exclusion distance is only needed for sensory-based input. Line 7 in Algorithm 2 adds the exclusion distance criterion.

2.2.1 The Ideal Case

In the ideal case, the detection process would have neither FN nor FP. In other words, every pixel in the raw sensor data will be correctly identified as either foreground or background. Given a properly calibrated exclusion distance, the expectation is the behaviors of the HA model using the two detection processes would be equivalent. It follows, if the exclusion distance is approximately the same as the diameter of the group members, then it will function as member-based detection.

2.3 Robot Implementation

Throughout this dissertation, multi-robot trials are conducted to support the various claims. For all trials presented in this work, iRobot Create robots are equipped with a Hokuyo URG-04LX-UG01 scanning laser rangefinder (for all sensing requirements) and an ASUS Eee PC (for computation and control); see Figure 2.4. The robots are wrapped in a specific type of fiducial paper according to the trial being conducted. On the control side, each robot's control-law is written for use with the Player-3.0.2 and Gearbox-9.07 drivers. Each driver has been modified for the specific trials in this dissertation. Specific details outside of the base platform for each type of trial will be detailed later in this work.

2.3.1 Implementation Specifics

For member-based detection, group members are detected by segmenting the laser scan into individual segments based on distance between consecutive pixels. Then each scan is *time-warped* to have an equal number of readings and compared to the *a priori* database of group members (Berndt, 1994). For each detected segment, all

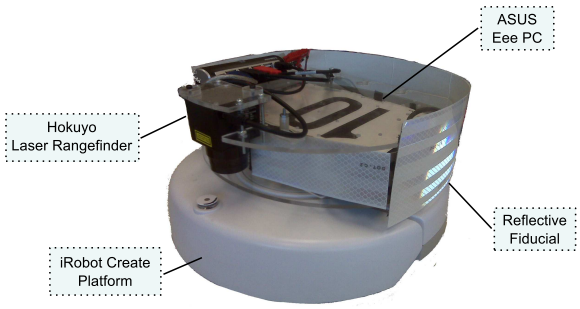


Figure 2.4: A single robot wrapped in high reflectance material for the fiducial and equipped with a Hokuyo URG-04LX-UG01 scanning laser rangefinder and an ASUS Eee PC.

of the corresponding pixels are averaged into a single (x,y)-position.

For the sensor-based detection model, the Gearbox 9.07 drivers to allow for the detection of intensity. The intensity of the pixels are compared to a given threshold value. Values exceeding the threshold are labeled foreground pixels, or background otherwise.

The experiments for this case-study consisted of three different starting formations (Figure 2.5) and various parameter settings including three different exclusion distances (0, 0.5, 1 meter). To compare the two perception functions, trials with six robots in a obstacle free corridor approximately seven robot diameters wide and long enough to be considered infinite for these results were conducted.

2.4 Results

Based on analysis and observations of 63 robot trials, sensor-based input into the HA flocking model does indeed still produce flocking behaviors. This suggests the common assumption of member-based detection is not necessarily justified. As hypothesised, these experiments support the following claim:

For any motion rule designed with member-based detection, similar be-

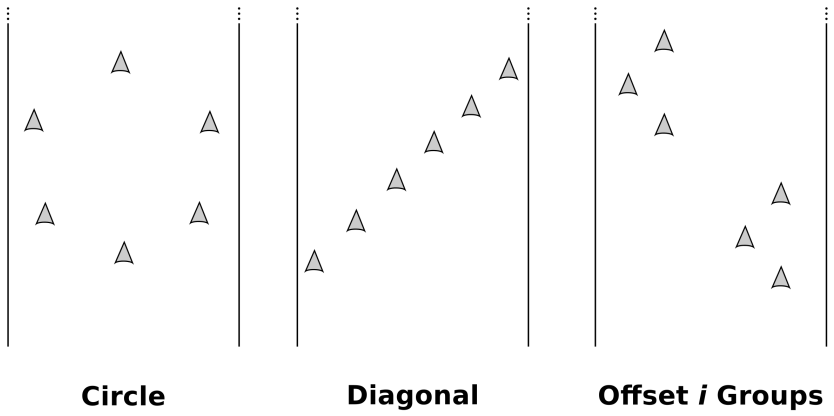


Figure 2.5: Starting formations for six robots.

haviors can be exhibited with sensor-based detection given a particular set of parameters.

More specific to the HA flocking model, there is some range of exclusion distances which produce similar behaviors no matter what detection process is used.

Figures 2.6, 2.7, 2.8, and 2.9 show the inter-robot distance plots for all six robots for single trials. The inter-robot distance is the distance between the focal robot and all inputs given to the HA control-law. Each color represents the raw sensor data from one of the six robots.

Figures 2.6 and 2.7 are from trials with the same parameter settings but different detection methods. Other than the sheer quantity of inputs, the inter-robot distances are similar in these two trials. In both plots, the majority of the robots sense one other robot, and this is at a similar distance. This single robot observation is because the robots are reverting to SNN behavior.

These plots also show the effects FP and FN have on the inputs to the flocking model. Marker **B** highlights the sensor signature of robots that only appear briefly

in Figure 2.6. Comparing these two figures it is apparent that both trials had a strong tendency to the SNN behaviors. These plots, along with the similarity in the observed behaviors, strongly support the claim above.

2.4.1 Effect of the Exclusion Distance

As expected, the exclusion distance did not have any observable affect on the behaviors of the HA motion rule when using member-based detection, but it did have a substantial effect on the behaviors when using sensor-based detection. When the exclusion distance is zero, the group (using sensor-based detection) always behaves identically to the SNN behavior, even though the group members are always calculating the HA way-point (due to the implicit assumption described above). This is to be expected because the two selected group members will almost always be adjacent to each other. The only time this assumption fails is when a FP is selected as one of the two inputs. As the exclusion distance increases, the behaviors start to exhibit the HA behaviors more than the SNN behaviors (after some lower-bound of the exclusion distance). At some point, if the exclusion distance becomes too large, the behaviors will again exhibit SNN behaviors. This upper-bound exists: a trivial case is when the exclusion distance is equal to the size of the individual's field of view.

Figures 2.7, 2.8, and 2.9 use sensor-based input with the exclusion distances of 0, 0.5, and 1 meter; respectively. These three support the existence of an interval for the exclusion distance where the system can successfully exhibit the HA behaviors. When the exclusion distance is 0 and 1, the system begins to converge (this convergence is the same as described above) to the SNN behavior (Marker **A** in the respective figures). However, in Figure 2.9 (see Marker **C**), there are two robots that detect robots which are not in the SNN formation (straight line). This means the

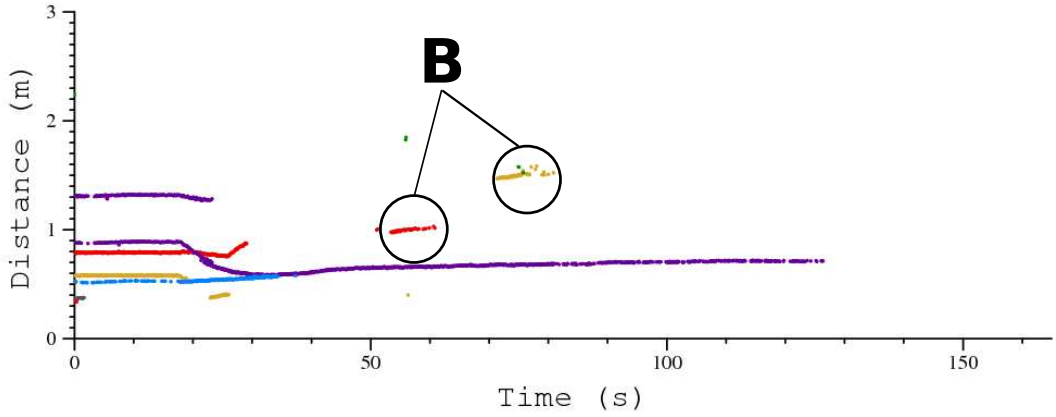


Figure 2.6: A single trial of member-based detection with an exclusion distance of 0. Marker **B** shows the issue of FN for member-based input.

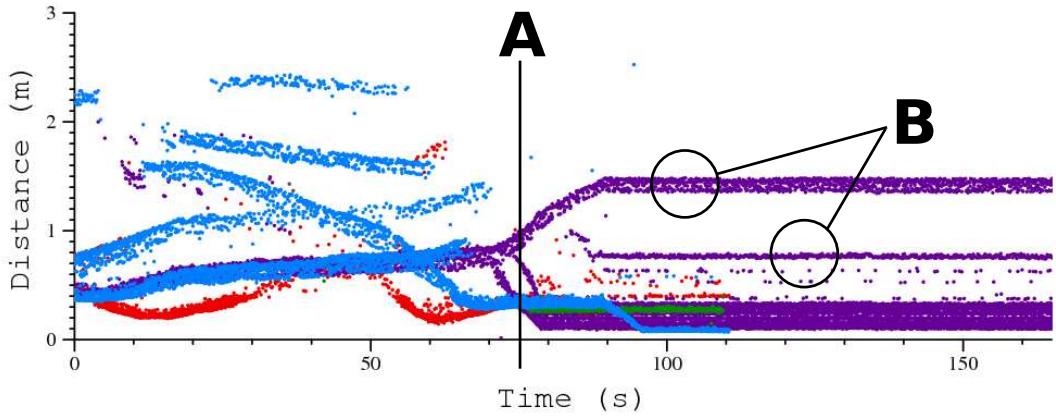


Figure 2.7: A single trial of sensor-based detection with an exclusion distance of 0. Marker **B** shows the robustness to FN.

percentage of HA way-point behaviors is higher in Figure 2.9 than Figure 2.7. When the exclusion distance is 0.5 there is no appearance of strong convergence to the SNN behavior.

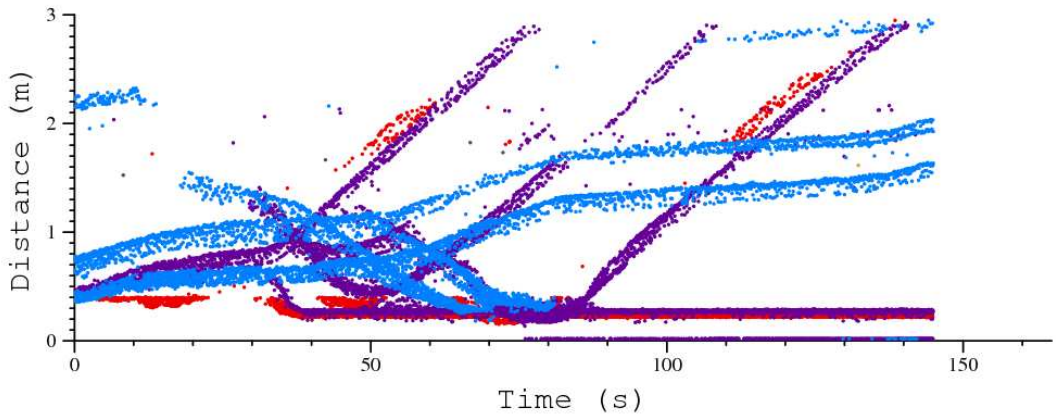


Figure 2.8: A single trial of sensor-based detection with an exclusion distance of 0.5.

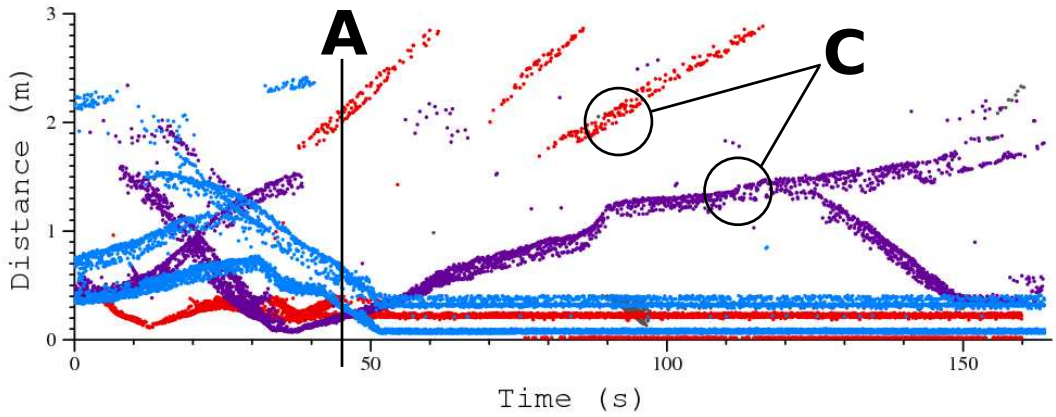


Figure 2.9: A single trial of sensor-based detection with an exclusion distance of 1.

2.5 Case-study Discussion

This case-study has identified two behavioral dimensions which are useful in understanding and describing the HA flocking model; smoothness and behavior.

Smoothness is the variance in the chosen way-point over a trial and **behavior** is the percentage the HA motion rule behaviors are observed. Ideally, the behaviors should be completely smooth (rather than jittery) and the behaviors should always exhibit the HA behaviors.

Based on the above criteria, there are certain parameter settings where sensor-based detection out performs member-based detection. To understand why, a third detection method was implemented: member-extrapolated detection. **Member-extrapolated** detection uses the same *a priori* description to detect foreground pixels; however, the foreground pixels are passed directly to the control-law without being averaged. Figure 2.10 shows the data flow from the raw sensor data to the input to the control-law for member-extrapolated detection.

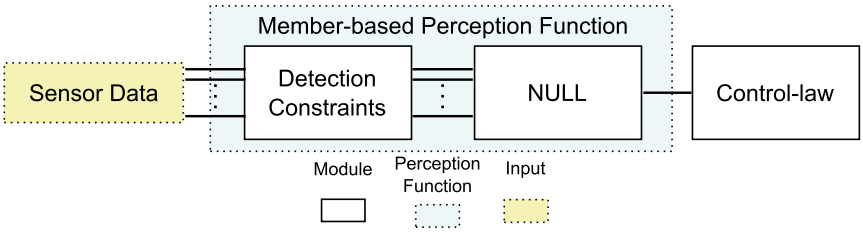


Figure 2.10: The data flow from raw sensor data to control-law input using member-extrapolated detection. For the implementation, the constraints are the same as in member-based detection, but the foreground pixels are not averaged to single (x,y)-coordinates.

The trials show the behaviors with member-extrapolated detection are similar to the sensor-based detection with respect to the exclusion distance, however, the overall behaviors are more similar to the member-based detection. This is (likely) due to the effects of sensor noise, occlusions, and the constraints used during the detection process. In the ideal case (given the proper parameter settings) the behaviors using

all three detection processes would be equivalent. With a realistic sensor, one can expect a certain ratio of FP and FN.

The results from the trials suggest member-based and member-extrapolated detection minimizes the FP while increasing the FN. Sensor-based detection minimizes the FN while increasing the FP. Member-extrapolated detection has a similar effect on the FP/FN ratio as member-based detection because both are using the same set of constraints on the sensor data. The only observable difference in the performance of member-extrapolated and member-based detection is member-extrapolated tends to be less *smooth* than member-based. The *smoothness* in member-based detection is attained through the averaging of the foreground pixels.

Using sensor-based detection, the behaviors become considerably less *smooth* (due to FP) but the group members do not revert to the SNN behavior as often as member-based and member-extrapolated. Since there are fewer constraints on the selection of inputs, it is more likely to detect at least one pixel from all of the neighboring group members, which increases the probability the focal member can find two inputs which are at least the exclusion distance apart. Since two inputs are likely to be selected, the HA behaviors more often than SNN (assuming a proper parameter set).

2.6 Case-study Summary

The results from the above experiments have shown that the widely used assumption of whole member detection is not required for individuals to produce flocking behaviors. The realization sensor-based detection is a valid perception function for flocking behaviors is very important when considering a models applicably for biological group members. Sensor-based detection requires less computational resources and requires less *a priori* knowledge than a segmentation based approach, such as member-based detection. Thus, it is reasonable to conclude that for simpler animals

such as ants and bees, member-based perception functions are not used.

Additionally, this treatment of the HA flocking model has highlighted the necessity of implementing flocking models on a multi-robot system and show the need for a more rigorous approach to studying this phenomena. Without validating a model on a physical system (like a multi-robot system), it is difficult to prove the model's completeness and its biological plausibility. Many of the issues highlighted above can be addressed by having a better understanding of the key aspects of flocking models.

3. UNIFYING MICROSCOPIC FLOCKING MODELS *

To better understand all aspects involved in the generation of flocking behaviors, a series of analytical tools that help reduce incompleteness and imprecision in the design and presentation of these models is introduced. Together, these three distinct tools use three different approaches for the understanding of flocking behaviors; (1) organizational, (2) categorical, and (3) structural. To gain a complete and precise understanding of a given model, all three tools must be applied, as they highlight different aspects of the model.

The first tool, the data-flow template (DT), identifies and addresses the key aspects for the production of flocking behaviors and how these aspects relate. Together, the five aspects, or stages, of the DT (see Section 3.2) give an organizational description for the production of flocking behaviors that can be used for understanding what information is required and what stage(s) utilize it. Additionally, the DT can be a useful *blueprint* for the design of new flocking models.

Moving beyond the organization of each particular model, the second tool aims to chart the relationships across existing models; the objective is to categorize existing work concisely while retaining sufficient precision to allow a practitioner to resolve implementation questions. A taxonomy is developed to detail the computation, sensing, and motion capabilities of the individual group members. Additionally, a second taxonomy is introduced that aids in the classification of validation methods used for a particular flocking model. When using the two taxonomies together, one may gain insight into which assumptions or capabilities may be infeasible or

impractical for a robotic or biological group member. Using the two taxonomies together also affords the ability to identify which model design choices have been fully validated (*e.g.*, local sensing versus global sensing).

Flocking models that appear to be similar at both the organizational and categorical levels may actually differ significantly in the structure of the motion computation. The third tool used for gaining a fuller understanding of the model is a consolidated notation and formalization which focuses on the motion computation of a given control-law. The **control-law** is a combination of the neighbor selection and the motion computation stages and can be considered the *algorithm* of the model. Formalization of the current control-laws facilitates understanding the implementation differences between the various models. Even though many of the flocking models have the same aim, the formalization and notation shows the implementation of the models are typically quite different, which can lead to different modeling assumptions.

3.1 Scope

Of the publications reviewed from the flocking model literature, only literature that considers the individual group members model for investigating flocking behaviors are considered. Microscopic flocking models have been the primary focus of the many diverse research communities, thus resulting in several reviews and surveys of the literature (Blomqvist et al., 2012; Giardina, 2008; Goldstone and Janssen, 2005; Parrish et al., 2002; Vicsek and Zafeiris, 2012). These models typically focus on the control-law used to produce the motions of the individual group members. Such models have been used to explore why flocks exist (Barbosa, 1995; Bazazi et al., 2008; Hamilton, 1971; Partridge, 1982; Viscido and Wethey, 2002; Viscido et al., 2002), what is required for the production of flocking behaviors (Fine and Shell, 2011, 2012;

Pitcher et al., 1976; Reynolds, 1987), how much influence individuals have on the group (Conradt et al., 2009; Couzin et al., 2005; Warburton and Lazarus, 1991), what special/unique properties might exist in such a group (Vicsek et al., 1995), and how these motions can be useful in robot applications (Arkin and Balch, 1999; Ferrante et al., 2012; Turgut et al., 2008).

In addition to investigating flocking motion using microscopic models, some work uses macroscopic models (Babak et al., 2004; Mogilner and Edelstein-Keshet, 1999; Vaughan et al., 2000). In some of the literature, macroscopic models are used as a tool for the validation of microscopic models and are not used as a *stand alone* model (Albi and Pareschi, 2013; Cavagna et al., 2012). In the works considering a stand alone macroscopic model, there is still difficulties in model the complex interactions between all members of the group and the environment. Due, in part, to these difficulties, the group control work starting in Section 6 will use simulations of microscopic models to predict the group level behaviors.

3.2 Data-flow Template

The data-flow template (DT) aids in designing and presenting complete microscopic flocking models. Each of the five stages (sensing, group member detection, neighbor selection, motion computation, and physical motion) of the DT represent the key aspects for the generation of flocking behaviors. The five stages of the DT are connected by the information passed between them. With an explicit understanding of the five stages and the relative connections, one gains a complete understanding of the control-law and it facilitates repeatability among researchers. The DT differs from the other tools presented in this work, in that the DT defines microscopic models at the logical level. In this way, the DT is a blueprint for flocking models, as it details the major building blocks and how they fit together.

In addition to serving as a blueprint for the creation of flocking models, the DT is a useful tool for understanding the complexity of a given model. Complexity (in this section) refers to how much computation, in the motion computation stage, is required to execute a given model. Therefore, a model using all of the detected group members is less complex than a model that only uses a subset of the detected group members, since this model performs extra computation to produce the required subset. The DT used in this way can be useful for gaining a better understanding for which stages can be executed at the hardware level, thus making the model less complex. Figure 3.1 shows a generic view of the DT along with the connections between the various stages.

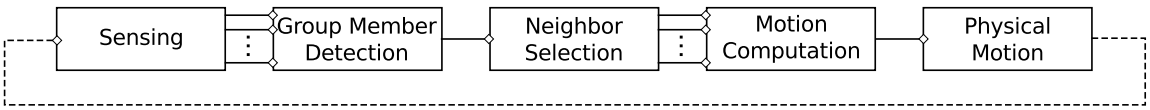


Figure 3.1: A diagrammatic representation of the proposed DT for microscopic flocking models. It details the main aspects for the generation of flocking behaviors via the five boxes (stages). The connections between the stages encode the data that propagates between them. In particular, the connections between the sensing stage and group member detection stage represents the raw sensor information from each sensor (*e.g.*, laser range-finder, camera, GPS). The connection between the group member detection and neighbor selection stage is the set of detected group members ($\mathbf{D}_i(t)$). The neighbor selection stage passes at least one set of selected group members ($\mathbf{I}_i(t)$) to the motion computation stage which passes the next computed motion to the physical motion stage.

3.2.1 The Five Stages: Definitions

Sensing: The sensing stage translates the visible environment (from the individual sensors reference frame[†] into usable input for the later stages (*e.g.*, a laser range-finder converts the visible environment into a list of ranges). A group member's internal representation of the visible environment is based on the sensors used, therefore the design of the following stages is directly affected by this stage. For example, if a group member is equipped with a laser range-finder the group member detection stage may use shape (based on the type of raw sensor data) to detect other group members in the environment. Although there is no formal input to the sensing stage, the DT in Figure 3.1 shows input to the sensing stage from the physical motion stage because the resulting motion of that stage may affect the visible environment.

Group Member Detection: The group member detection stage uses the raw sensor information provided by the sensing stage and outputs the set of all detected group members ($\mathbf{D}_i(t)$). The set $\mathbf{D}_i(t)$ is a subset of all possible group members, represented by the set $\mathbf{A}(t)$, that are within the visible sensing region ($\mathbf{R}_{[\alpha,\,\beta]}$). In other words, if the senors defined in the sensing stage only senses information within a two meter radius ($\mathbf{R}_{[0,\,2\text{ meters}]}$), then the set $\mathbf{D}_i(t)$ will only contain group members within a two meter radius. Additionally, each group member in the set $\mathbf{D}_i(t)$ encompasses all of the required information (*e.g.*, position and velocity) and the type of information used for the description of a group member has no effect on the DT. Therefore, if the model requires position and orientation information, then for each group member in $\mathbf{D}_i(t)$, there will be a corresponding $\mathbf{r}_j(t)$ and $\theta_j(t)$.

[†]Typically the sensor's reference frame will be that of the group member, but when the model uses global sensors (*e.g.*, overhead camera) the two reference frames will differ.

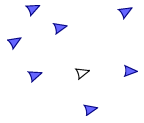
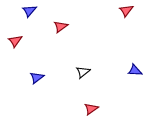
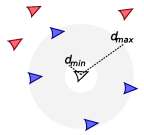
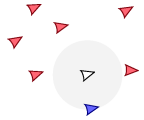
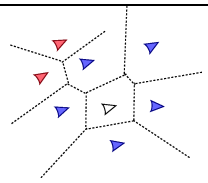
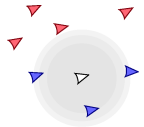
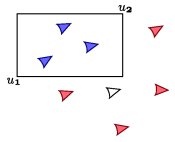
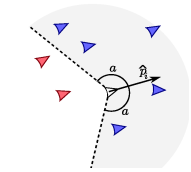
Neighbor Selection: The neighbor selection stage takes the set $\mathbf{D}_i(t)$ provided by the group member detection stage and outputs at least one subset of the set $\mathbf{D}_i(t)$. The set(s) generated by the neighbor selection stage only contain group members which will be used in the motion computation stage. Therefore, if the motion computation stage only uses the nearest neighbor[‡] (in distance) to compute the next motion, then the neighbor selection stage will only output a set that contains the nearest neighbor.

To reduce the set $\mathbf{D}_i(t)$ to the desired output, the neighbor selection stage uses a set of perception functions (see Table 3.1 for a list of perception function definitions). The neighbor selection function can use any number and combination of these functions in order to reduce the set $\mathbf{D}_i(t)$ into usable input for the motion computation stage. The perception functions could be used in succession (*e.g.*, union of the output of two perception functions) or in parallel (*i.e.*, the neighbor selection stage would output more than one set). For example, if the motion computation stage may require two sets as input (*e.g.*, attraction and repulsion sets), the neighbor selection stage will output two sets; the set of 'attraction-zone' group members and the set of 'repulsion-zone' group members. One possible representation of the neighbor selection function that considers the attraction-repulsion zones, using the notation in Table 1, is $(\mathbf{I}_{Repulsion}(t) \cap \mathbf{I}_{Attraction}(t)) \subseteq \mathbf{D}_i(t)$.

Motion Computation: The motion computation stage uses the set(s) generated by the neighbor selection stage to calculate the next motion of the group member (*e.g.*, this stage can update any combination of $\mathbf{r}_i(t)$, $\theta_i(t)$, $\mathbf{u}_i(t)$, or $\mathbf{v}_i(t)$). It is important to note, this stage only describes the internal representation of the next motion and does not describe how the internal representation is translated into low-level control commands for the group member. In the case of the attraction/repulsion zone

[‡]In this dissertation, group member X calls member Y a neighbor if and only if Y can be sensed by X. Note, there is no assumption of symmetry in this notation.

Table 3.1: Descriptions of perception functions found in the selected literature. Most neighbor selection stages could be created using a combination of the functions listed here, but these may not be the only possible functions.

$\text{All}() = \{j \in \mathbf{A} : j\} = \mathbf{A}$		All group members are selected.
$\text{FixedSet}_i() = \{j \in \mathbf{S}_i \subseteq \mathbf{D}_i : j\} = \mathbf{S}_i$		Each group member is given a fixed set of group members $\mathbf{S}_i$ (*e.g.*, group members are nodes in a graph with fixed connectivity).
$\text{DistanceBased}_i(d(\cdot), \mathbf{R}_{[d_{min},\, d_{max}]}) = \{j \in \mathbf{D}_i : d_{min} \leq d(i,j) < d_{max}\}$		All group members within some distance from the sensing group member are selected.
$\text{Nearest}_i(d(\cdot)) = k \in \mathbf{D}_i$ where $k \in \text{t}(\underset{j \in \mathbf{D}_i}{\text{argmin}} d(i,j))$		Only the nearest group member from the sensing group member is selected.
$\text{VoronoiBased}_i() = \{j \in \mathbf{V}_i \subseteq \mathbf{A} : j\} = \mathbf{V}_i$		The voronoi neighbors, represented by the set $\mathbf{V}_i(t)$, of the sensing group member are selected.
$\text{k-Nearest}_i(d(\cdot)) = \{j \in \mathbf{D}_i : \text{k-}\underset{j \in \mathbf{D}_i}{\text{argmin}} d(i,j)\}$		A generalized form of the nearest neighbor perception function which selects the k-nearest group members from the sensing group member.
$\text{BoundingBox}_i(\mathbf{u}_1, \mathbf{u}_2) = \{j \in \mathbf{D}_i : (\mathbf{u}_1 x \leq \mathbf{r}_j x \leq \mathbf{u}_2 x) \wedge (\mathbf{u}_1 y \leq \mathbf{r}_j y \leq \mathbf{u}_2 y)\}$		All group members within a defined bounding box are selected (this function is designed for a 2D plane but it is trivial to extend to higher dimensions).
$\text{Angle}_i(\alpha) = \{j \in \mathbf{D}_i : \arccos(\hat{\mathbf{r}}_j \cdot \hat{\rho}_i) < \alpha\}$		All group members that are within a specified field of view are selected.

Legend: Detected group member Selected group member Focal member

example, the motion computation stage may compute two different motion vectors that are summed together to produce the next motion. Table A.1 describes most of the neighbor selection and motion computation stages in the selected publications.

Physical Motion: The physical motion stage takes the computed motion from the motion computation stage and translates it into a form that can be realized in either a simulated or physical robot (*e.g.*, a kinematic control-law or left and right motor speeds for a two wheeled robot). Like the sensing stage, there is no formal output given by the physical motion stage; however, the resulting motions have an impact on the visible environment used by the sensing stage (*e.g.*, physical motion could affect which group members belong to $\mathbf{D}_i(t + \Delta t)$). Therefore, the sensing and physical motion stages are connected in Figure 3.1.

3.2.2 The Five Stages: Selected Literature

To show the five stages are indeed key aspects for the generation of flocking behaviors, gain a better understanding of the current microscopic flocking models, and determine which aspects of flocking motion generation have not been fully studied, the DT is applied to the selected literature. This section highlights specific examples found in the literature that have various levels of completeness in regards to specific DT stages. Figure 3.2 gives examples of specific data-flows seen in the literature between the five stages of the DT.

Sensing: Surveying the literature using the DT shows different research communities tend to focus only on particular aspects of flocking motion generation. Biology and physics models simplify both the sensing and physical motion stages, where control literature models simplify the sensing stage but introduce motion constraints and various types of noise. There are few works which investigate the sensing stage in great detail, with one example being (Kelly and Keating, 1996). The treatment of

the sensing stage in (Kelly and Keating, 1996) describes all of the sensors used and the various properties associated with those specific sensors. Additionally, they give a detailed description of the physical group member, which could have an impact on the sensing, group member detection, and neighbor selection stages. The data-flow between the sensing and group member detection stages for the model presented by (Kelly and Keating, 1996) can be seen in Figure 3.2a.

Although the level of detail given in (Kelly and Keating, 1996) is desirable, it is not always feasible to describe all five stages in a detailed manner due to space limitations. In contrast to the verbose description in (Kelly and Keating, 1996), (Gazi and Passino, 2005) gives a brief, yet complete description of the sensing capabilities of the group members. Even though the sensing capabilities for the group members rely on strong assumptions (instantaneous and perfect sensing with an infinite range), the treatment of the sensing stage does not allow for ambiguities in the understanding of the presented approach.

The vast majority of the literature either does not discuss or present a complete description of the sensing stage. An example of a publication that does not treat the sensing stage can be seen in (Tanner et al., 2003a). (Tanner et al., 2003a) simply state there are n group members moving in a plane that contain a position ($\mathbf{r}_i(t)$) and a velocity ($\mathbf{v}_i(t)$). There is no discussion of how the positions and velocities are sensed and/or calculated. Only from the context could the fact the model in (Tanner et al., 2003a) uses an oracle to maintain the information of the group members be inferred.

For the publications that have a partial treatment of the sensing stage (Czirók et al., 1997; Vicsek et al., 1995; Viscido et al., 2002), it is typical to see the following style of description: "The group member can detect all members within a radius of r". From this description the reader cannot disambiguate between the case where

the sensing radius is a simulation of a sensor limitation or if the sensing radius is a part of the control-law. From the biology literature, it is known, that some species only use a limited/specific number of groups members to calculate their next motion, thus the distinction of what the sensing radius actually represents is important for the understanding of the overall model.

Group Member Detection: The group member detection stage is one of the least completely treated and discussed aspects of flocking motion generation. The vast majority of the literature does not consider group member detection and makes the assumption all of the group members within the sensing range are included in the set $\mathbf{D}_i(t)$. Additionally, the group member detection stage is also responsible for the detection of the required information. If in the sensing stage a *velocity* sensor is not specified and the control-law requires the group member's velocity, then the group member detection stage should detail how the velocity information is calculated (*e.g.*, the velocity is inferred from the displacement of $\mathbf{r}_i(t)$ and $\mathbf{r}_i(t\ +\ 1)$).

A couple of good treatments of the group member detection stage can be seen in (Arkin and Balch, 1999; Kelly and Keating, 1996; Turgut et al., 2008). (Turgut et al., 2008) give an adequate description through describing how all of the required information is sensed (*e.g.*, velocity, identification, orientation, etc.) by the group members. However, (Turgut et al., 2008) does not discuss if any noise or detection error exist in the group member detection stage. A common data-flow between the group member detection and neighbor selection stage can be seen in Figure 3.2b.

To this point, only group member detection stages that detects real group members have been considered (*i.e.*, $\mathbf{D}_i(t)\ \subset\ \mathbf{A}_i(t)$). Another possible role of the group member detection stage is the detection/creation of virtual group members. Virtual group members can be used to avoid obstacles (Olfati-Saber, 2006) and/or to as-

sist in maintaining a desired group structure (Lindhé et al., 2005). A virtual group member is created from the sensed information. The information typically used for the creation of virtual group members is encoded in the set $\mathbf{A}_i(t)$ and any detected obstacles. In the notation, a virtual group member is a group member that exist in the set $\mathbf{D}_i(t)$ but does not exist in the set $\mathbf{A}_i(t)$. It is important to note, a virtual group member in the set $\mathbf{D}_i(t)$ cannot be distinguished from a non-virtual group member in the same set.

Neighbor Selection: The neighbor selection stage is treated in almost all of the literature and is the only stage that is generally completely described. One of the most complete treatments can be found in (Viscido et al., 2002). For each of the perception functions, (Viscido et al., 2002) explicitly states which group members will be selected and passed to the motion computation stage. The data-flow for one of the models presented by (Viscido et al., 2002) between the neighbor selection and motion computation stage can be seen in Figure 3.2c. In addition to which group members are to be selected, (Viscido et al., 2002) gives brief explanations for the selection decisions. The one exception to this is in their treatment of the local crowded horizon (LCH) control-law. (Viscido et al., 2002) do not clarify if all of the detected group members are selected or if only the members apart of the 'most crowded horizon' are selected.

In contrast to (Viscido et al., 2002) fairly complete treatment of the neighbor selection stage, (Mikhailov and Zanette, 1999) do not explicitly treat this stage at all. With that said, it is clear from the context, (Mikhailov and Zanette, 1999) are using a perception function that selects all of the group members in the set $\mathbf{D}_i(t)$ (see the perception function All() in Table 3.1).

The majority of the literature typically uses one perception function in the neigh-

bor selection stage; however, there are some works which use a combination of perception functions. The perception functions in Table 3.1 shows the control-law proposed by (Gueron et al., 1996) uses over six perception functions. This is because the motion computation stage for (Gueron et al., 1996) has a large number of conditions which affect the computation output (similar to the "attraction/repulsion" zone example used earlier). For a more detailed treatment of this work please see Section 3.4.1.

In recent years, microscopic flocking model literature has been using the term topological to describe a particular set of flocking models (Ballerini et al., 2008b; Bode et al., 2011; Cavagna et al., 2010; Ginelli and Chaté, 2010; Niizato and Gunji, 2011; Tanner et al., 2003a,b). **Topological models** only differ from *metric* models in the neighbor selection stage of the DT, however, this distinction is not clear in the majority of the topological literature. Niizato and Gunji (2011) is effective in teasing out the difference between topological and metric flocking models. Additionally, (Niizato and Gunji, 2011) presents a control-law that utilizes both a metric and topological neighbor selection function.

Motion Computation: Even though the motion computation stage is always treated, the description of this stage is typically incomplete, which causes ambiguities when attempting to formalize or implement a control-law. One reason for the lack of completeness is from not explicitly addressing all of the outputs generated by the neighbor selection stage. For example, the SNN flocking model (Viscido et al., 2002) does not detail the group members computed motions when there are no neighbors. In other words, if the set $\mathbf{D}_i(t)$ is empty, the motion computation stage is undefined.

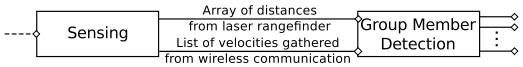
Another cause for the lack of incompleteness in the motion computation stage is from the use of vague terminology such as *back-up* and *turn left*. As discussed later in

this Section (Section 3.4.1), the use of vague terminology prevents us from formalizing such models, and thus it prevents us from implementing a model as it was intended to be implemented. A third reason for incompleteness of the motion computation stage can be seen in Viscido et al. (2002). Within the same publication there are two possible interpretations of the LCH control-law, but the authors never distinguish between the two of them. The first prose description of the LCH control-law and the implemented version of the model are similar but not exactly the same. Both of these interpretations have been implemented on a physical system and, fortunately, found the generated motions only differ slightly (Fine and Shell, 2012). However, the assumptions made by the two different interpretations do have an impact of the complexity of the control-law.

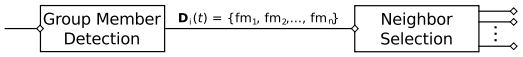
Physical Motion: The physical motion stage is rarely treated in the flocking literature. A common input to this stage can be seen in Figure 3.2d, where the motion computation stage takes a computed position ($\mathbf{r}_i(t)$) and translates it into low-level motor commands for the group member.

3.3 Current Microscopic Models

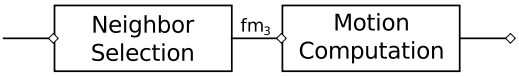
This section analyzes the high-level design choices for the selected microscopic flocking models. Table 3.2 is a categorical view of the sensing and computational requirements of individual group members; as well as, the composition of the entire group (*i.e.*, is the group heterogeneous or homogeneous). The eight attributes (group composition, group member mobility, continuous/discrete time, collision avoidance, neighbor identification, neighbor's position, neighbor's velocity, and neighbor's orientation) highlighted in Table 3.2, together, give an adequate description of the model's required information. Additionally, Table 3.2 allows one to gain a better understanding for if a particular model is biologically feasible, or how easily the model may be to



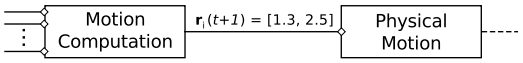
(a) The sensing stage reduces the visible environment into an array of distances from a laser rangefinder and a list of velocities gathered through wireless communication; both being common sensors chosen for robotic applications. The array and list are passed to the group member detection stage as the only inputs. This particular data-flow is seen in Kelly and Keating (1996).



(b) The group member detection stage translates the information given by the sensing stage into a set of n group members. It is important to note that $\mathbf{D}_i(t)$ is a set of abstract group members and that a individual group member (fm) can be thought of as a feature vector. The features of a group member are defined by the required attributes (e.g., $\mathbf{r}_j(t)$, $\mathbf{v}_j(t)$, $\theta_j(t)$, etc.). The set of group members is passed to the neighbor selection stage. This data flow is used in all of the selected literature.



(c) The neighbor selection stage selects a subset ($\mathbf{I}_i(t)$) of the set $\mathbf{D}_i(t)$ to be passed to the motion computation stage. The selection function used in this example only selects the nearest group member. This example can be seen in Hamilton (1971) and Viscido et al. (2002).



(d) The motion computation stage (i.e., control-law) uses the set $\mathbf{I}_i(t)$ to compute the next motion for the i^{th} group member. In this example, the next position for the group member is (1.3, 2.5) and it is passed to the physical motion stage. This example can be seen in Viscido et al. (2002), Hamilton (1971), Szabó et al. (2008), Szabó et al. (2009), and Gazi and Passino (2003).

Figure 3.2: Specific examples of data-flow between the various stages in the DM. Note that the examples listed here are from a combination of various publications. There is no known example of a publication that unambiguously describes the data-flow between all stages.

implement on a robot system. For example, if a control-law requires idealize/perfect motion, then the resulting motions of this rule implemented on a robot (*i.e.*, noisy) system may not be equivalent to the desired motions. In general, Table 3.2 can be used to answer three questions; (1) what is the composition of the group, (2) how realistic are the constraints, and (3) what sensing information is required?

Table 3.2: A categorical review of the information and group member requirements for each of the selected models. This table can be used as a first pass to identify which models could be useful when designing a new model or when identifying models that may benefit from additional investigations. Entries marked with a ($\surd$) signifies that the particular attribute is utilized. For the collision avoidance attribute, (Member) signifies that the model only considers member to member collision avoidance, while (All) signifies that the model considers both member and environment collision avoidance.

Paper	Group Composition	Mobility	Continuous/Discrete	Collision Avoidance	Identification	Position	Velocity	Orientation
Viscido et al. (2002)(SNN/HA/LCH)	□	○	⊙	–	–	√	–	–
Conradt et al. (2009)	⊞	○	⊖	Member	–	√	√	–
Gueron et al. (1996)	⊞	○	⊙	Member	–	√	–	–
Couzin et al. (2005)	⊞	○	⊙	Member	–	√	√	–
Lopez et al. (2012)	⊞	○	⊙	–	–	√	–	–
Hamilton (1971)	□	○	⊙	–	–	√	–	–
Vicsek et al. (1995)	□	○	⊙	–	–	√	–	√
Dong (2012)	□	○	⊙	–	–	√	√	–
Smith and Martin (2009)	□	○	⊙	–	–	√	–	√
Shimoyama et al. (1996)	□	○	⊖	Member	–	√	√	√
Czirók et al. (1997)	□	○	⊙	–	–	√	–	√
Szabó et al. (2008)	□	○	⊙	–	–	√	√	√
Szabó et al. (2009)	□	○	⊙	–	–	√	√	√

Continued on next page

Table 3.2 Continued

Paper	Group Composition	Mobility	Continuous/Discrete	Collision Avoidance	Identification	Position	Velocity	Orientation
Levine et al. (2000)	□	◯	⊖	Member	–	√	–	–
Toner and Tu (1998)	□	◯	⊙	–	–	√	–	√
Grégoire et al. (2003)	□	◯	⊙	Member	–	√	–	√
Camperi et al. (2012)	□	◯	⊙	Member	–	√	√	–
Helbing et al. (2005)	⊞	◯	⊖	All	–	√	√	–
Matarić (1993)	□	△	⊖	All	–	√	–	–
Reynolds (1987)	□	△	⊖	All	–	√	√	√
Kelly and Keating (1996)	□	△	⊖	All	√	√	√	√
Turgut et al. (2008)	□	△	⊙	All	–	√	√	–
Gökçe and Şahin (2009)	⊞	△	⊙	All	–	√	√	√
Tanner et al. (2003a)	□	◯	⊖	Member	√[§]	√	√	√
Tanner et al. (2003b)	□	◯	⊖	Member	–	√	√	√
Jadbabaie et al. (2002)	□	◯	⊙	–	–	√	–	√
Gazi and Passino (2005)	□	◯	⊙	–	√	√	–	–
Gazi and Passino (2003)	□	◯	⊙	Member	–	√	√	√
Olfati-Saber (2006)	□	◯	⊙	All	–	√	–	√
Arkin and Balch (1999)	□	△	⊖	All	√	√	–	–
Fine and Shell (2011)	□	△	⊙	–	–	√	–	–
Hauert et al. (2011)	□	△	⊙	Member	–	√	√	–

Legend: □ Homogeneous ⊞ Heterogeneous ◯ Idealized △ Constrained ⊖ Continuous ⊙ Discrete

[§]Identification is only used in the perception function.

3.3.1 Definitions of Design Attributes

Group Composition: Several studies investigate the possibility of differences (sometimes subtle, sometime significant) in the group members. Apart from physical differences (*e.g.*, in sensing capability, size, appearance), there can be differences in the underlying control law; more precisely, the motion computation stage in the DT may vary between group members. A group's composition can either be homogeneous ($\square$), where all group members have identical motion computation stages, or heterogeneous ($\boxplus$), where at least one group member has a different motion computation stage.

This definition of group composition only considers the motion computation stage when determining if a group is homogeneous or not, and does not consider the motions exhibited by individual group members. For example, if all group members have an identical probabilistic motion model (Shimoyama et al., 1996; Viscido et al., 2002), then the group is considered homogeneous even though the motions of the individual group members given identical input may not be the same. However, if the model has parameters which are unique to a subset of group members (Conradt et al., 2009; Couzin et al., 2005), the group is considered heterogeneous because the unique parameters can drastically affect the group member's motion. In the case of (Couzin et al., 2005) there are two types of group members (informed and uninformed), thus $\mathbf{A}_i(t)$ can be partitioned into two distinct subsets of group members, and therefore considered to be heterogeneous.

This study does not consider either the sensor configuration or the physical appearance of the group members in regards to the group's composition, as the focus is only on the chosen control-law. However, a variation of group composition could consider the sensor configuration of the group members. Knowing the particular

sensor configuration may reveal implementation details that may affect the design of the model. Even though not considered here, the sensor configuration should be discussed when presenting the sensing stage. Studies have also investigated the effects of heterogeneous group members (in appearance) on the human perception of a group (Ip et al., 2006). Heterogeneous flocks (in appearance) are not considered under the assumption that the physical appearance of the group members (for the selected literature) does not affect the exhibited flocking behaviors.

Mobility: The mobility of the group details the physical motion individual group members can perform and it is assumed all of the group members have the same mobility (*i.e.*, the group is homogeneous in regards to mobility). A group member's mobility can either be idealized ($\bigcirc$), where the group member can immediately and perfectly perform the computed motion, or constrained ($\triangle$), where the group member has a restricted set of possible motions (*e.g.*, grid-based motion) or imperfect (noisy) motion. For example, if a group member that has idealized motion executes a model which calculates the next position ($\mathbf{r}_i(t+1)$) at time t, the group member will arrive at $\mathbf{r}_i(t+1)$ (with no error). Conversely, if the same group member now has constrained motion, there could be a difference between the calculated and observed $\mathbf{r}_i(t+1)$ (*i.e.*, the group member is not guaranteed to arrive at $\mathbf{r}_i(t+1)$).

Discrete/Continuous: Group members can execute the given model in either discrete time ($\odot$) or in continuous time ($\ominus$). Group members operating in discrete time only sense, compute, and act (move) in distinct time intervals (*i.e.*, a group member may only sense, compute, and act every 0.1 seconds, regardless of the sensors sensing frequency). Conversely, group members that execute the given model in continuous time will sense, compute, and act at the frequency of the given sensors and the time it takes to preform the computations. The important distinction between the two

time choices is that in discrete time, the group members may ignore information if the time interval is too large, where in continuous time, no information will be ignored (assuming all sensors have the same frequency). However, if the information from the sensors is noisy and the group member is operating in continuous time, the resulting motion may be sporadic and may not exhibit the desired motions.

Collision Avoidance: The collision avoidance of the group details the individual group member's ability to avoid various types of collisions. The two types of collisions studied in the literature are member-member collisions (a group member collides with another group member) and member-environment collisions (a group member collides with an object in the environment that is not another group member). Group members can avoid collisions with other group members (Member), the environment (Environmental), both group member and environmental obstacles (All), or group members have no collision avoidance capabilities (–).

Identification: Models that utilize identification assume each group member has a unique label (ID) (*e.g.*, leader or group member 42). Furthermore, these models assume each group member has the ability to *detect* other group member's ID at any time. Identification can be used to assist in member-to-member communication, as well as, allowing *follower* group members to identify the *leader* group member(s).

It is important to note, if a group member has the ability to identify other members, this does not imply the member has the ability to associate information between sensing steps (for the definition of association see Section 3.6). Furthermore, it is important to note that the index of the group members in the sets $\mathbf{A}_i(t)$ and $\mathbf{D}_i(t)$ cannot be used for identification (*i.e.*, the group member $j \in \mathbf{D}_i(t)$ is not guaranteed to be member $j \in \mathbf{D}_i(t+1)$). Therefore, if identification is required, there must be an identification attribute associated with each member, just as all other

attributes (*e.g.*, position, velocity).

3.3.2 Position, Velocity, and Orientation

The position, velocity, and orientation columns identify whether or not a model utilizes that particular type of information. These three columns do not distinguish how the information is sensed or computed (*e.g.*, global versus local reference frame), rather they simple state what information is required. In other words, these columns identify the minimum informational requirements the control-law must have in order to compute the next motion for the group member.

3.3.3 Observations of Chosen Design Attributes

Group Composition: Observing Table 3.2, six of the investigations consider heterogeneous group members. Even though the current definition of heterogeneity is restricted, all of the studies which consider homogeneous group members, use truly homogeneous members (*i.e.*, all aspects of the group members are identical). For the publications which do consider heterogeneous flocks, the studies typically investigate how *informed members* or *leaders* can affect the motions of the group (Conradt et al., 2009; Couzin et al., 2005). These investigations have parameters which differ according to the group member's classification (*e.g.*, leader or follower). In (Gueron et al., 1996), heterogeneity represents *strong* and *weak* group members, where *strong* group members move faster than *weak* members. This is not considered to be heterogeneity with respect to mobility because the feasible motions of the group members are still the same, one type of group member simply performs the motion faster than the other (i.e., a different gain in the motion computation stage).

Mobility: Table 3.2 reveals eight publications which consider constrained motion, which all exist in the robotics and control literature. The majority of the literature assumes if a group member computes it's next position as $\mathbf{r}$, then at time $t + 1$

the group member will be at **r** (with no error). Of the literature that does consider constrained motion, none of the models explicitly handle the motion constraints in any of the DT stages. Even though Vicsek et al. (1995) is probabilistic in regards to the motion noise, the noise is added in the motion computation stage and thus classified as idealized motion in Table 3.2. When the group member moves to the calculated position in the physical motion stage, there is no error; thus, the (Vicsek et al., 1995) model is considered to use idealized motion.

Continuous/Discrete Time: Table 3.2 shows a divide among research groups with respect to the use of continuous or discrete time models. Almost all of the selected literature from the biology and physics communities use discrete time, where the robotics and control groups predominately use continuous time. One possible cause for this dichotomy are the chosen forms of validation; see Table 3.4 (*e.g.*, computer simulation versus robot implementation).

Continuous time models are typically more complete (in the specific sense outlined in the Introduction) than discrete time models, although, there are some exceptions. (Gueron et al., 1996) is an example of a discrete time motion model that is completely described. As seen in (Gueron et al., 1996), the authors took great care to present what motion output would occur given any possible input. Another example of a completely described discrete time model can be found in (Conradt et al., 2009), where the authors detail what group members do in the absence of neighbors (*i.e.*, the neighbor selection stage outputs an empty list).

Collision Avoidance: Eight of the publications consider both environmental and group member collision avoidance, with nine other publications considering member-member collision avoidance. The group members in the Olfati-Saber (2006) study perform collision avoidance with both the environment and other group members;

however, the motion computation stage does not handle any environmental obstacles. Group members generate virtual group members (see Section 3.6) which travel along the boundary of the detected obstacles, thus the motion computation stage only considers group member avoidance. Additionally, none of the investigations explicitly describe, with the exception of Reynolds (1987), how the obstacle avoidance is performed.

Position/Velocity/Orientation: When considering the complexity of a given control-law, only what the model requires (*e.g.*, position, velocity, and/or orientation information) is considered. There does not appear to be a difference in the design of the motion computation stage with respect to the way in which the information is sensed (*e.g.*, global, local, or inferred). However, there is a difference in how the sensed information is used.

The majority of the models use the information in the motion computation stage, but some models (Jadbabaie et al., 2002; Vicsek et al., 1995) use some of the information in the neighbor selection stage. In (Viscido et al., 2002) the motion computation stage only requires the velocities ($\mathbf{v}_j(t)$) of the selected group members; however, position information ($\mathbf{r}_i(t)$) is used to select a subset of group members from $\mathbf{A}_i(t)$. Table 3.2 does not make a distinction on where the information is used.

With exception of the prose description of the LCH motion rule in (Hamilton, 1971), all of the selected models require position information. In the proposed, but not validated, description of the LCH motion rule each group member moves towards the center of the highest density of detected group members. With respect to velocity and orientation information, there does not seem to be any major trend seen in the literature.

3.4 Specification of Flocking Models

To aid in the comparison of various flocking models and to assist in the understanding of the current state of the literature, the selected models are translated into a common notation and formalized stages 3 and 4, which together create the motion rule. Table A.1 shows the formalization of the neighbor selection stage and motion computation stage for each of the selected models. Observing Table A.1 reveals even though each of these models have the same aim, there are many ways in which the flocking problem can be solved.

3.4.1 Literature Omitted from Model Specification

Not all of the models from the selected literature can be easily formalized in the proposed framework (Arkin and Balch, 1999; Gueron et al., 1996; Kelly and Keating, 1996; Matarić, 1993; Reynolds, 1987), which have been labeled as "See Section X.Y.Z" in Table A.1. The most common reason (found in all omitted works except for Gueron et al. (1996)) for omission is ambiguity in the details of the low-level control law used to produce the flocking behaviors. Gueron et al. (1996) is a unique case in which the motion rule is completely and precisely described, but the model is so complex, it does not lend it self to being formalized.

The Dynamics of Herds: From Individuals to Aggregations (Gueron et al., 1996): In Gueron et al. (1996), the authors presented the motion rule in enough detail where formalization is possible, but the rule is extremely verbose, which made it impractical to completely formalize the model in Table A.1. Table A.1 shows the formalization of one of the four spatial regions (*e.g.*, attraction, repulsion) which affects the group member's motion. As shown in Table A.1, the high number of discrete conditions in the motion computation stage requires a high number of perception functions in the neighbor selection stage. This suggests some care should be taken when designing

the neighbor selection and motion computation stages.

Self-Organized Pedestrian Crowd Dynamics: Experiments, Simulations, and Design Solutions (Helbing et al., 2005): Helbing et al. (2005) presents a control-law based on the social force model presented in (Helbing and Molnár, 1995). This model, along with other similar models Moussaïd et al. (2009), could be formalized into the presented notation if a complete and precise description of all of the forces were given. Specifically in (Helbing et al., 2005), the authors present the force $\mathbf{f}_{\alpha i}(\mathbf{r}_\alpha, \mathbf{r}_i, t)$ but only define the force as "attraction effects". Furthermore, the term $\xi_\alpha(t)$ is defined to be a "fluctuation term [that] reflects random behavioral variations". $\xi_\alpha(t)$ is clearly a noise term but the authors do not sufficiently define the properties of this term.

Designing Emergent Behaviors: From Local Interactions to Collective Intelligence (Matarić, 1993): In Matarić (1993), the flocking behaviors are created from multiple behaviors, such as *Follow*, *Avoid*, *Aggregate*, and *Disperse*. Within the descriptions of each behavior there are ambiguities which make it difficult to formalize the behaviors. For example, the avoidance behavior in Matarić (1993) is composed of two types of avoidance; environmental and member. Within the environmental avoidance computation, there are ambiguous phrases such as "backup and turn", "turn right, go.", and "if an [obstacle] is on the right". It is difficult to translate these phrases into the proposed framework and notation without making certain decisions which could skew the original model design.

Another difficulty with the formalization of the avoidance behavior is the dynamics between the two types of avoidance. Both avoidance methods have the similar statement of "If an [obstacle/group member] is on right...". If an obstacle is on the left and a group member is on the right, it is unclear what the resulting behavior should be according to the description given in Matarić (1993). Similar ambiguities

can be found in the other behaviors and the dynamics between them.

Even without these ambiguities, it would be difficult to recreate the flocking model presented in (Matarić, 1993). The presentation of the behaviors uses phrases such as "backup" and "turn right". If these behaviors were formalized, the majority of the terms would be parameters and not calculations based on input. This is not a problem when it comes to validating the model which produces flocking behaviors, but it does make it difficult to reproduce the work for further study and comparisons.

Flocks, herds and schools: A distributed behavioral model (Reynolds, 1987): As in the previous studies discussed, Reynolds (1987) is difficult to formalize without introducing bias. Even though the three rules for flocking presented in Reynolds (1987) are highly cited in the robotics literature, the details of the three rules are ambiguous. Again, the problem lies in the details of the low-level behaviors and vague descriptions of the various parameters needed.

Table 3.3 shows two possible formalizations of the flocking models in Reynolds (1987). Both Reynolds (2004) and Kline (1996) were formalized using actual software implementations of Reynolds proposed model. Even though these formalizations are similar and produce similar flocking behaviors, there are a few key differences.

First, Kline (1996) only considers the closest group member within a given radius when computing $\mathbf{a}_i(t)$ and $\mathbf{s}_i(t)$ where Reynolds (2004) considers all group members within the sensing radius. The computational differences between these two approaches have a direct affect on the neighbor selection function. Please note, in the formalization of the neighbor selection function in Kline (1996), the union of $\text{DistanceBased}(\cdot) \cap \text{Nearest}()$ had to be included in order to select the nearest neighbor, which is explicitly used in the motion computation stage.

Table 3.3: Both entries in this table are possible implementations of the control-law in Reynolds (1987). The two implementations only vary slightly; however, the differences have effects on the complexity of the group members and the underlying causes of flocking behaviors.

Version	**Neighbor Selection**	**Motion Computation**
Reynolds (2004)	$\mathbf{I} = \text{DistanceBased}(L^2, \mathbf{R}_{[0,\ \beta]})$	$\mathbf{v}_i(t) = w_1\hat{\mathbf{s}}_i(t) + w_2\hat{\mathbf{a}}_i(t) + w_3\hat{\mathbf{c}}_i(t)$ $\mathbf{s}_i(t) = \frac{\sum_{j\in\mathbf{I}} \frac{\mathbf{r}_i(t) - \mathbf{r}_j(t)}{-\delta_2(\mathbf{r}_i(t),\ \mathbf{r}_j(t))^2}}{\lvert\mathbf{I}\rvert}$ $\mathbf{a}_i(t) = \frac{\sum_{j\in\mathbf{I}} \theta_j(t)}{\lvert\mathbf{I}\rvert} - \theta_i(t)$ $\mathbf{c}_i(t) = \frac{\sum_{j\in\mathbf{I}} \mathbf{r}_j(t)}{\lvert\mathbf{I}\rvert} - \mathbf{r}_i(t)$
Kline (1996)	$\mathbf{I} = \text{DistanceBased}(L^2, \mathbf{R}_{[0,\ \beta]})$	$\mathbf{v}_i(t) = w_1\hat{\mathbf{s}}_i(t) + w_2\hat{\mathbf{a}}_i(t) + w_3\hat{\mathbf{c}}_i(t)$ $\mathbf{s}_i(t) = \begin{cases} 0 & \text{if } k = \emptyset, \\ \delta_2(\mathbf{r}_i(t),\ \mathbf{r}_j(t)) & \text{otherwise.} \end{cases}$
	$\mathbf{I} = \text{DistanceBased}\ (L^2, \mathbf{R}_{[0,\ \alpha]}) \cap \text{Nearest}()$	$\mathbf{a}_i(t) = \begin{cases} 0 & \text{if } k = \emptyset, \\ \mathbf{a}_{ix}(t) = \begin{cases} \mathbf{a}_{ix}(t) - w_5 & \text{if } (\mathbf{r}_k(t) - \mathbf{r}_i(t))x < w_4, \\ \mathbf{a}_{ix}(t) + w_6 & \text{otherwise.} \end{cases} & \\ \mathbf{a}_{iy}(t) = \begin{cases} \mathbf{a}_{iy}(t) - w_5 & \text{if } (\mathbf{r}_k(t) - \mathbf{r}_i(t))y < w_4, \\ \mathbf{a}_{iy}(t) + w_6 & \text{otherwise.} \end{cases} & \text{otherwise.} \\ \mathbf{a}_{iz}(t) = \begin{cases} \mathbf{a}_{iz}(t) - w_5 & \text{if } (\mathbf{r}_k(t) - \mathbf{r}_i(t))z < w_4, \\ \mathbf{a}_{iz}(t) + w_6 & \text{otherwise.} \end{cases} & \end{cases}$ $\mathbf{c}_i(t) = \frac{\sum_{j\in\mathbf{I}} \mathbf{r}_j(t)}{\lvert\mathbf{I}\rvert} - \mathbf{r}_i(t)$

On Flocking By The Fusion Of Sonar And Active Infrared Sensors (Kelly and Keating, 1996): Similar to Matarić (1993), Kelly and Keating (1996) has ambiguous behavior descriptions, which makes it difficult to formalize into the unifying framework. In Kelly and Keating (1996) the authors present the dynamics between the different behaviors as a hierarchy but does not fully present the underling behaviors. The authors use phrases such as "try to maintain position" and "speed up", which are difficult to formalize without making assumptions on the author's intent.

Behavior-based Formation Control for Multi-robot Teams (Arkin and Balch, 1999): This work is a prime example of how researchers should report all of the various parameters and gains used in the validation process. In addition to there being no ambiguities in respect to the parameters and gains, the authors list the actual values used, thus allowing for repeatability. However, this work was unable to be formalized due to some ambiguity in the presentation of the primitive schemas used. For example, the *Move-to-goal* schema states "Attract to goal with variable gain. Set high when heading for goal." Even though, it seems straight forward to formulate this schema, formalization is not attempted here for the same reasons as in Section 3.4.1.

3.5 Validation Methods

The various methods of control-law validation found in the selected literature are analyzed in this section. Table 3.4 is a review of the validation choices (*e.g.*, synchronous versus asynchronous group members) and the various validation methods (*e.g.*, computer simulations versus robot implementations). The six attributes (validation method, synchrony, neighbor's position, neighbor's velocity, neighbor's orientation, group's environment) highlighted in Table 3.4 give an adequate description of how current models are being validated. The information provided in Table 3.4 also affords us the ability to gain insight into which models may be more effective in

a real world situation. In other words, if a control-law is only validated using global information in an obstacle free environment, this model may not produce flocking behaviors when introduced in a more realistic environment. Additionally, Table 3.4 can be used to cross-check the assumptions made in the design of a given model (*e.g.*, a model design for local information should be validated with local information).

Table 3.4: Details the validation methods chosen for the selected motion models. The form of validation employed varies significantly within the publications.

Paper	Validation Method	Synchrony	Position	Velocity	Orientation	Environment
Viscido et al. (2002)(SNN/HA/LCH)	$\mathcal{S}$	√	Global	–	–	Bounded Free Space
Warburton and Lazarus (1991)	$\mathcal{S}$	√	Global	Global	Global	Unbounded Free Space
Conradt et al. (2009)	$\mathcal{S}$	√	Global	Global	Global	Unbounded Free Space
Codling et al. (2007)	$\mathcal{S}$	√	Global	Global	Global	Unbounded Free Space
Gueron et al. (1996)	$\mathcal{S}$	√	Local	Global	Global	Unbounded Free Space
Couzin et al. (2005)	$\mathcal{S}$	√	Global	Global	Global	Unbounded Free Space
Lopez et al. (2012)	$\mathcal{S}$	√	Global	Global	Global	Unbounded Free Space
Huth and Wissel (1992)	$\mathcal{S}$	√	Global	Global	Global	Unbounded Free Space
Hamilton (1971)	$\mathcal{S}$	√	Global	–	–	Unbounded Free Space
Vicsek et al. (1995)	$\mathcal{S}$	√	Global	Constant	Global	Periodic Free Space
Dong (2012)	$\mathcal{M}$	√	Global	Global	–	Unbounded Free Space

Continued on next page

Table 3.4 Continued

Paper	Validation Method	Synchrony	Position	Velocity	Orientation	Environment
Smith and Martin (2009)	$\mathcal{S}$	√	Global	Constant	Global	Periodic Free Space
Shimoyama et al. (1996)	$\mathcal{S}$	√	Global	Global	Global	Unbounded Free Space
Czirók et al. (1997)	$\mathcal{S}$	√	Global	Global	Global	Periodic Free Space
Szabó et al. (2008)	$\mathcal{S}$	√	Global	Global	Global	Periodic Free Space
Szabó et al. (2009)	$\mathcal{S}$	√	Global	Global	Global	Periodic Free Space
Levine et al. (2000)	$\mathcal{S}$	–	Global	Constant	Global	Periodic Free Space
Toner and Tu (1998)	$\mathcal{M}$	√	Global	Constant	Global	Bounded Free Space
Grégoire et al. (2003)	$\mathcal{S}$	√	Global	Constant	Global	Periodic Free Space
Camperi et al. (2012)	$\mathcal{S}$	√	Global	Global	–	Unbounded Free Space
Helbing et al. (2000)	$\mathcal{SP}$¶	–	Global	Global	–	Bounded Obstacles
Matarić (1993)	$\mathcal{P}$	–	Local	–	Local	Bounded Free Space
Reynolds (1987)	$\mathcal{S}$	√	Global	Global	Global	Unbounded Obstacles
Kelly and Keating (1996)	$\mathcal{P}$	–	Local	Inferred	Inferred	Bounded Free Space
Turgut et al. (2008)	$\mathcal{SP}$	–	Global	–	Local	Bounded Free Space
Gökçe and Şahin (2009)	$\mathcal{SP}$	–	Global	–	Local	Bounded Free Space
Tanner et al. (2003a)	$\mathcal{MS}$	√	Global	Global	Global	Bounded Free Space
Tanner et al. (2003b)	$\mathcal{MS}$	√	Global	Global	Global	Unbounded Free Space
Jadbabaie et al. (2002)	$\mathcal{M}$	√	Global	Constant	Global	–
Gazi and Passino (2005)	$\mathcal{MS}$	–	Global	Global	Global	Unbounded Free Space
Gazi and Passino (2003)	$\mathcal{M}$	√	Global	–	–	–

Continued on next page

¶The physical agent were human participants.

Table 3.4 Continued

Paper	Validation Method	Synchrony	Position	Velocity	Orientation	Environment
Olfati-Saber (2006)	$\mathcal{MS}$	✓	Global	Global	Global	Unbounded Obstacles
Arkin and Balch (1999)	$\mathcal{SP}$	–	Global	–	–	Unbounded Obstacles
Fine and Shell (2011)	$\mathcal{P}$	–	Local	–	–	Bounded Free Space
Hauert et al. (2011)	$\mathcal{SP}$	–	Local	Local	–	Unbounded Free Space

Legend: $\mathcal{M}$: Mathematical $\mathcal{S}$: Simulation $\mathcal{P}$: Physical

3.5.1 Definitions of Attributes for Validation Methods

Validation Method: The validation method attribute details what combination of the possible methods where used to validate the control-law and characteristics of the flocking motion produced by the model. The methods seen in the selected literature are mathematical verification ($\mathcal{M}$) of flocking motion characteristics (*e.g.*, group stability), computer simulation ($\mathcal{S}$), and the use of robot implementations ($\mathcal{P}$). Both computer simulations and robot implementations rely on implementing the control-law and studying the exhibited motions of the group. Literature using mathematical methods typically prove the existence of various group-level characteristics (*e.g.*, does the group converge to a stable formation or does a phase shift occur). The difference between validation (computer simulations and robot implementations) and verification (mathematical) lies in the scope of the chosen method.

Verification methods consider particular aspects of the flocking behaviors (*e.g.*, convergence and group stability). In other words verification methods simply check to see if the model performs the particular expected behaviors, such as convergence. Verification methods describe here adheres to the definition in Oberkampf and Trucano (2000), that the developed model does indeed *output* the desired behaviors (or

solution).

Validation methods consider the overall motions produced by the group. These methods look at how *flock-like* the group's behaviors are. This differs from the definition of validation in Oberkampf and Trucano (2000) in that there is no consistent or standard measure of degree of accuracy.

Synchrony: Synchrony defines whether or not group members sense, compute, and act in unison. If one group member executes any of the five stages at a different time or frequency, validation is said to be asynchronous. The key distinction between synchronous and asynchronous validation is if the model is asynchronous, then one cannot assume the sets $\mathbf{A}_i(t)$ and $\mathbf{A}_j(t)$ are identical because the sensing of the group members could have been executed at different times.

Position/Velocity/Orientation: The position, velocity, and orientation attributes describe the way in which the required information (see Table 3.2) is actually sensed. The three sensing methods found in the literature are global (Global), local (Local), or inferred from other information (Inferred). In the selected publications global information is either sensed by an overhead sensor or by an oracle that maintains the required information. Local information is gathered by the group member via sensors or member-to-member communication. Inferred information can be gathered in two ways, (1) by using two distinct types of information (*e.g.*, speed and orientation can yield velocity) or (2) by associating data from multiple sensing iterations (*e.g.*, displacement of position over time can yield velocity).

Environment: The environment attribute describes the type of environment considered in the validation of the proposed model. The environment can be any combination of bounded, unbounded, or periodic space, with or without obstacles. For all of the literature reviewed in this study, all of the models where designed with regards

to a particular environment. There are no cases in the selected literature where a control-law was tested in an environment it was not designed for.

3.5.2 Observations of Chosen Validation Methods

Validation Methods: The most commonly seen form of validation is the use of computer simulations which is then followed by robot implementations. It is important to note there are various degrees of simulation validation (*e.g.*, physics-based, sensor-based, etc.), which are not covered in this meta-study. Mathematical verifications are typically used to show certain known group properties hold given a particular model. In Tanner et al. (2003a) and Tanner et al. (2003b) the authors use graph theory to show the group members maintain a stable group (*i.e.*, all group members maintain common velocities and avoid collisions with other group members). However, mathematical verifications are also used to explore certain properties of the group, such as phase shifts in Mikhailov and Zanette (1999)[‖]. The most complete validation treatment of a proposed microscopic model found in the selected literature was done in Lindhé et al. (2005). This work uses all three methods of validation covered in this meta-study.

Synchrony: From Table 3.4, it is observed that ten of the publications consider asynchronous group members. Additionally, notice all of the publications that consider asynchronous group members are from the control and robotics literature. The low number of asynchronous flocks is surprising because from Şamiloğlu et al. (2006), it is shown asynchronism can have negative impacts on the exhibited flocking behaviors.

Position/Velocity/Orientation: All of the models from the selected literature only require local information (see formalization of control-laws in Table A.1); however, many of the studies use global information for the validation. This disconnect be-

[‖]Phase shifts refer to the moments in time when a group becomes or discontinues being a group.

tween design and validation leaves room for subtle assumptions which could affect the group's overall motion; such as the effects of occlusions. Physical group members using local sensing may not be able to detect the same set of neighbors as a global sensor due to occlusions from group members and environmental obstacles (*i.e.* the set $\mathbf{D}_i(t)$ using a global sensor may differ from the same set sensed via a local sensor). For example, if there are three collinear group members, a member on the end of that line may only be able to detect one neighbor using local sensing rather than both neighbors. This issue becomes apparent when considering the HA model from Viscido et al. (2002).

If the motion computation stage presented in Viscido et al. (2002) uses local information, the motion computation stage becomes undefined in some cases. Since the HA control-law was validated using global information (with the absence of occlusions), the motion computation stage does not treat cases where only one neighbor is detected; thus presenting a potential problem when local information is used. For a more detailed treatment of the HA model please see Viscido et al. (2002) and Section 2. There is no instance in the selected literature where the effects of occlusions are properly treated or studied.

Velocity information is, arguably, the most complex type of information used in the literature; being a combination of a member's speed and orientation. Typically, group members need to employ extra strategies and information in order to attain the velocity of its neighbors; either through communication and identification, or data association. There are only a few works in the selected literature (Czirók et al., 1997; Kelly and Keating, 1996; Vicsek et al., 1995) which describe how the velocity information is attained. The other literature simply states the motion computation stage uses velocity information.

Environment: The environment used for robot and simulated validation is an important aspect to consider when planning to implement a particular approach on a robot system. None of the selected literature has investigated the effects the environment has on the motions of the group, with very few studies considering environmental obstacles (Arkin and Balch, 1999; Lindhé et al., 2005; Olfati-Saber, 2006; Reynolds, 1987). Investigations which do consider environmental obstacles typically select environments with few obstacles which are widely spaced from each other. Olfati-Saber (2006) is one of the more complete treatments with respect to the effects the environment has on the group's behaviors. The importance of the environment's effect on a group of individuals will be discussed in greater detail later in this dissertation.

3.6 Discussion of Unifying Microscopic Flocking Models

This meta-study presented three types of tools (data-flow model, two taxonomies, and a notation/formalization) to assist in the reduction of incompleteness and imprecision in microscopic flocking models. The proposed DT along with the two taxonomies and the notation/formalization allow for better understanding and comparison of the current literature on flocking models; however, there do exist some cases and investigations in which the current tool-set does not work as well as it could. Through the exploration of these outlying cases, future avenues of research that could prove beneficial to the overall understanding of the flocking phenomenon are highlighted.

3.6.1 Collision Avoidance in the Data-flow Template

Although the model is general, it imposes enough constraints to serve as a constructive guide toward consistent, complete, and precise descriptions of flocking models. One consideration not covered in the current DT is the motion computation stage's ability to handle collisions. The majority of the literature states that group

members avoid collisions with the environment, but the publications rarely describe the process/computations required to perform such motions (*e.g.*, how does the group member detect the environment, which parts (objects) are used in the computation stage, and/or how do the avoidance computations affect the motion computations). To allow for a better understanding of the collision avoidance capabilities, the DT must be modified as presented in Figure 3.1; which only defines the control-law when there are no obstacles in the environment.

To address collision avoidance in the DT, the modified DT in Figure 3.3 is introduced, which shows the addition of a sixth and seventh stage (obstacle detection and obstacle selection). These two additional stages are executed in parallel with the group member detection and neighbor selection stages, respectively. The obstacle detection stage takes the same input as the group member detection stage (input from the group member's sensors) and produces a set of obstacles. This set can be passed to the obstacle selection stage, which outputs a subset of detected obstacles for the motion computation stage, or the set can be paired with the group members in the set $\mathbf{D}_i(t)$ to generate virtual group members (see ① in Figure 3.3).

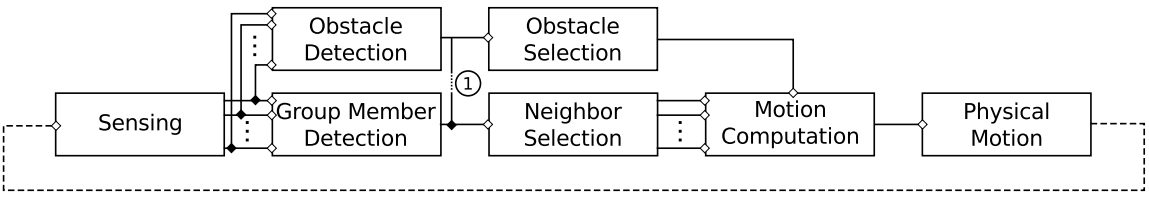


Figure 3.3: The modified DT that includes the obstacle detection and obstacle selection stages. These two stages allow for the addition of obstacle avoidance behaviors and the generation of virtual group members. The connection at ① can be treated as a place holder for adding a method that generates the required virtual group members, as in (Olfati-Saber, 2006).

Another aspect of group member collisions, either with other members or the environment, not covered in the current body of the flocking literature is the ability to use collisions as input to the control-law. There have been studies in biology that have suggested certain insects use information form member-to-member collisions in order to adjust their behaviors (Jackson and Ratnieks, 2006; Moglich et al., 1974). Even though these studies focus on task switching, it is plausible to assume certain flocking models may use collisions as input to the motion computation stage (*e.g.*, group members using collisions to navigate the environment). It would be interesting to see if it is possible to create flocking behaviors with only using information sensed from direct contact and how that might affect the common assumptions made in the flocking literature.

3.6.2 Neighbor Identification

In theory, identification can prove useful for maintaining the group's structure and determining a group member's membership within the group (*e.g.*, is the group member a leader or follower). Unfortunately, obtaining identification information in practice (on a robot implementation) is difficult and error prone. On a robot system there are three common ways to obtain the required identification information; (1) member-to-member communication, (2) direct sensing (*e.g.*, group members can identify the 'color' of its neighbor), and/or (3) association. To date, it is not clear if biological group members utilize identification, but Occam's razor would suggest that identification would not be required since flocking behaviors can be produced without it. Clearly, further investigations are needed in order to understand the role, if any, identification plays in the production of flocking motion.

3.6.3 Association

Association is the ability for a group member to pair sensor information from two consecutive sensor readings. Association can be used in three ways, (1) as a standalone part of a given model (i.e., using prior information as input to the control-law), (2) inferring information from multiple readings (*e.g.*, using a group member's position displacement to determine that member's velocity), and/or (3) using association to aid in identification (*e.g.*, if the group member knows the starting positions of its neighbors, the group member could use association to keep track of its neighbors' IDs). It is important to note, association and identification are distinct attributes that are independent of each other (*i.e.*, it is possible to use any combination of these attributes). The association column has been omitted from Table 3.2 due to the lack of proper presentation in the selected literature.

3.6.4 Data-Centric Approach to Determining Parameter Values

As seen in Gökçe and Şahin (2009); Kline (1996); Olfati-Saber (2006); Shimoyama et al. (1996) many publications present flocking models that contain many different parameters or gains. Very few works, if any, describe how the parameters for the control-law's validation where selected, where some works (Vicsek et al., 1995) validate the model over a range of parameter settings. Even in the investigations that study a range of parameter values, the justification of the chosen values remains unclear. The majority of the time the values are artificially 'tuned' until the desired flocking behaviors are produced from the given model. Useful information about the effects of parameter values may be found if parameter values can be derived from biological flocks (Butler et al., 2006; Lopez et al., 2012; Lukeman et al., 2010; Moussaïd et al., 2009). For example, the repulsion radius of the group members could be determined by analyzing the average distance maintained between the members of the

biological group.

3.6.5 Flocking Behaviors are Independent of Information Types

Observing Table 3.2, it can be seen almost every possible combination of required information was used in the studied flocking models. If only the information required by the motion computation stage is considered, the number of combinations seen in the literature increases. From this, it is reasonable to assume flocking models do not require a specific type of information. Further support for this observation will be shown in Section 4 with the introduction of information-abstracted flocking. This case-study will show the existence of a flocking model that is structured in a way as to allow for the use of different combinations of information without modifying the control-law.

3.7 Summary of Model Unification

This section has detailed the current state of flocking model literature that focuses on the individual member's models for the creation of flocking behaviors. Using the tool-set presented the commonly seen designs and assumptions in the current literature have been identified and discussed. As a result from this review it is clear, failure to properly treat all of the five stages of the DT could lead to incompleteness and/or imprecision in the presentation of the control-law. To demonstrate this, examples were adhering to the DT, leads to a more complete and precise understanding of the model (Section 3.2.2) were given. Unfortunately, from the review of the literature above and the work from Section 2, it seems the ability to understand the group level flocking behaviors from the study of the individual is not possible. Section 4 will further show this through computer and robot experiments using another commonly cited microscopic flocking model.

4. MANY MICROSCOPIC MODELS YIELD SIMILAR BEHAVIORS *

The aim of this section is to show how different microscopic flocking models can yield the same (or very similar) group level behaviors. To do this, the notion of information-abstracted flocking models is introduced. **Information-abstracted flocking models** are structured in such a way, the resulting behaviors is agnostic to the detail of the observation and/or to the type of information sensed. In other words, if the behaviors produced by a flocking model are equivalent under different types of information (*e.g.,* pose, bearing), then the model is considered to be information-abstracted on type.

The concept of information-abstracted flocking models is rooted from the work of generic programming (Musser and Stepanov, 1988), where a single implementation of an algorithm can be *instantiated* with different data representations, an idea analogous to abstraction in abstract algebra. Information-abstraction is actually stronger, as it actually produces comparable emergent behavior despite being instantiated with different data representations.

Additionally, the work in this section will show that:

- flocking models may be structured so the resulting behaviors are independent to the detail of the observation or the type of information given to the algorithm;
- comparison of one flocking model with another purely on the basis of the group behaviors they produce is inadequate. The existence of information-abstracted flocking models imply, despite the emergent behavior appearing equivalent, major pieces of the puzzle could remain underspecified;

*

- the vast majority of the literature assumes pose information is required to produce flocking behaviors, this study further supports the suggestion that biological flocking behavior is possible using bearing information only.

Furthermore, the presented implementation of the Local Crowded Horizon model (detailed below) using only bearing information is one of the most simplistic and biologically plausible flocking models to date.

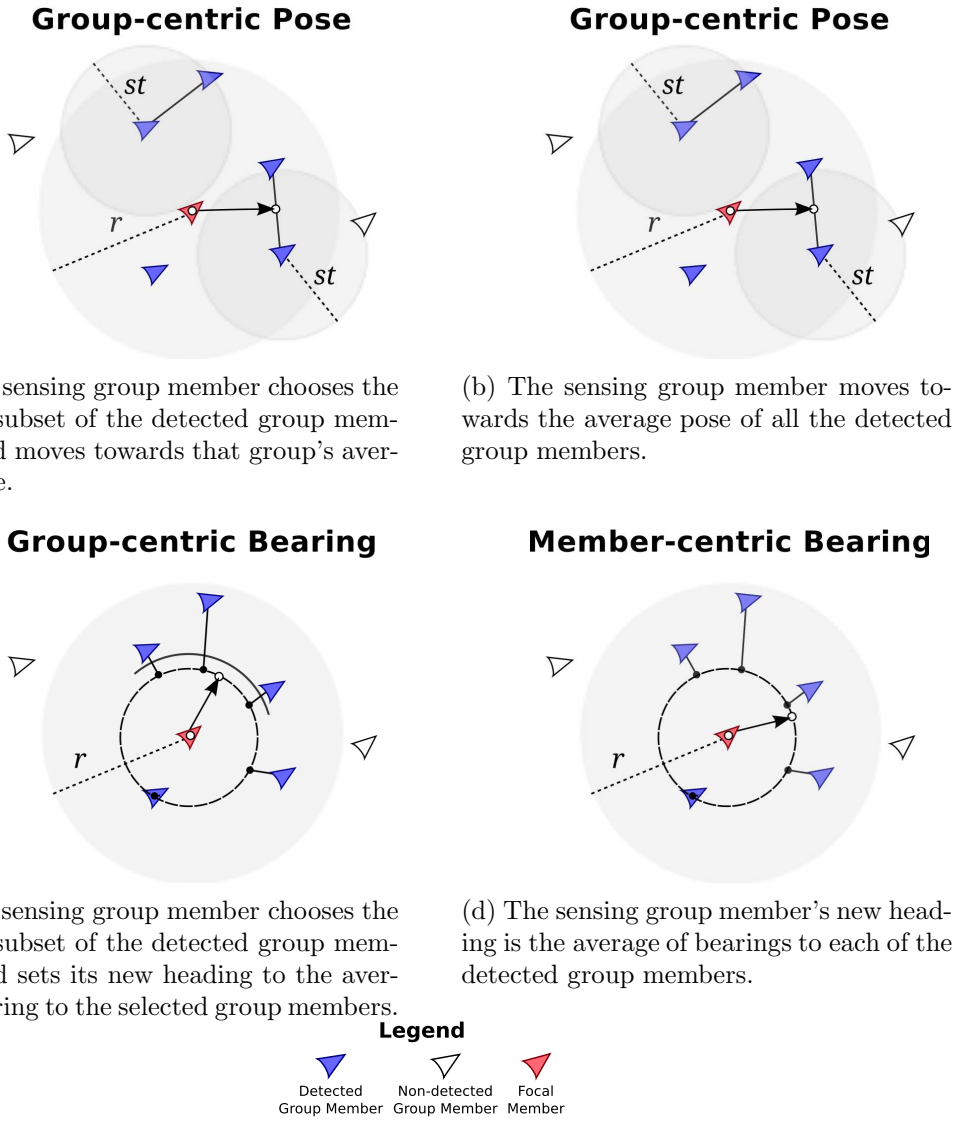
4.1 Local Crowded Horizon

The Local Crowded Horizon (LCH) microscopic flocking model was presented by Viscido *et al.* Viscido et al. (2002) as a biologically plausible explanation for flocking motion exhibited in the presence of a predator. In the original work there is a discrepancy between the theoretical design and the simulated version of the LCH. The authors describe the LCH by stating "[Group members] use the density of the entire [group] to determine their [next pose]"; however, their implementation has group members "move toward the average [pose] of [all of the detected group members.]" This discrepancy leaves the LCH with at least four different variations in regards to the information requirements; (1) group-centric pose, (2) group-centric bearing, (3) member-centric pose, and (4) member-centric bearing. The two **group-centric** variations use a subset of the detected group members where the two **member-centric** variations use all of the detected group members to compute the next pose. Figure 4.1 is a pictorial and prose description of the four different variations.

Both member-centric variations follow the same structure for the production of flocking behaviors. The member-centric variations take a set of all the detected group members ($\mathbf{I}$) and moves one unit (d) towards the average of the feature vectors in $\mathbf{I}$. A **feature vector** contains all of the sensed information required for the computation of the next pose (*e.g.*, pose, bearing, velocity) for each group member in $\mathbf{I}$. The only

difference between the two member-centric variations is in how the averaging of the feature vectors are handled.

In the member-centric variation using pose (see control-law named Variation 2) the function AveragePose($\mathbf{I}$) calculates the average pose of the set $\mathbf{I}$. To correctly av-



(a) The sensing group member chooses the largest subset of the detected group members and moves towards that group's average pose.

(b) The sensing group member moves towards the average pose of all the detected group members.

(c) The sensing group member chooses the largest subset of the detected group members and sets its new heading to the average bearing to the selected group members.

(d) The sensing group member's new heading is the average of bearings to each of the detected group members.

Figure 4.1: The four variations of the LCH flocking algorithm used in this study. Variations 4.1a and 4.1b require pose information where 4.1c and 4.1d use bearing information. Variations 4.1b and 4.1d utilize all of the detected group members where 4.1a and 4.1c only use a subset of the detected group members.

erage the group member bearings (Variation 4), AverageBearing($\mathbf{I}$) replaces AveragePose($\mathbf{I}$) and calculates the bearing from the average unit vector from the group member and all members in the set $\mathbf{I}$, *i.e.* 2-dimensional pose information is not required, only 1-dimensional bearing information.

The group-centric variations (Variations 1 and 3) are identical to the member-centric variations, except that the motion command that is computed depends on only a subset ($\mathbf{I}'$) of the set $\mathbf{I}$. This reflects the idea that the motions follow some strict prioritization where attention need only be paid to some salient (or dense, or tightly-clustered) individuals. In both group-centric variations, the group member computes the set $\mathbf{I}'$ based on density ($\mathbf{I}'$ contains all group members that exist in the highest density cluster) and moves one unit towards the average of the feature vectors in the set $\mathbf{I}'$.

Analogous to the case described above, different density selection functions are needed to handle the differences in the feature vectors. In the implementation of the group-centric variations, member-centric information is used to calculate the group-centric information to utilize the same detection process for all parameterizations. Of course, group-centric pose computations are only permitted to use group-centric information.

LCH Variation 1: Group-centric LCH Pose (Corresp. to Fig. 1(a))

Input: Set of detected group members ($\mathbf{I}$) in egocentric coordinate frame.

Parameters:

$st \doteq$ *splitting threshold (distance)*

$d \doteq$ *distance to travel*

$\mathbf{r}_i \doteq$ *current pose*

Output: Desired pose in a member-centric coordinate frame.

1: **if** $|\mathbf{I}| = 0$ **then**

2: **return** $[0,\ 0]$

3: **else**

4: $\mathbf{CC} \leftarrow \text{ConnComponents}(\mathbf{I}, st)$, where $\mathbf{CC} \geq 1$

5: $\mathbf{I}' \leftarrow \text{MaxConnComponent}(\mathbf{CC})$

6: $\mathbf{v} \leftarrow \text{AveragePose}(\mathbf{I}')$

7: **return** $\left[\mathbf{r}_{i^x} + \left(\frac{\mathbf{v}}{||\mathbf{v}||} * d\right),\ \mathbf{r}_{i^y} + \left(\frac{\mathbf{v}}{||\mathbf{v}||} * d\right)\right]$

LCH Variation 2: Member-centric LCH Pose (Corresp. to Fig. 1(b))

Input: Set of detected group members ($\mathbf{I}$) in egocentric coordinate frame.

Parameters:

$d \doteq$ *distance to travel*

$\mathbf{r}_i \doteq$ *current pose*

Output: Desired pose in a member-centric coordinate frame.

1: **if** $|\mathbf{I}| = 0$ **then**

2: **return** $[0,\ 0]$

3: **else**

4: $\mathbf{v} \leftarrow \text{AveragePose}(\mathbf{I})$

5: **return** $\left[\mathbf{r}_{i^x} + \left(\frac{\mathbf{v}}{||\mathbf{v}||} * d\right),\ \mathbf{r}_{i^y} + \left(\frac{\mathbf{v}}{||\mathbf{v}||} * d\right)\right]$

LCH Variation 3: Group-centric LCH Bearing (Corresp. to Fig. 1(c))

Input: Set of detected group members ($\mathbf{I}$) in egocentric coordinate frame.

Parameters:

$st \doteq$ *splitting threshold (angle)*

$d \doteq$ *distance to travel*

$\mathbf{r}_i \doteq$ *current pose*

Output: Desired pose in a member-centric coordinate frame.

if $|\mathbf{I}| = 0$ **then**
  return $[0,\ 0]$
else
  $\mathbf{CC} \leftarrow \text{ConnComponents}(\mathbf{I}, st)$, where $\mathbf{CC} \geq 1$
  $\mathbf{I}' \leftarrow \text{MaxConnComponent}(\mathbf{CC})$
  $\theta \leftarrow \text{AverageBearing}(\mathbf{I}')$
  return $[\mathbf{r}_{i^x} + (\cos(\theta) * d)\,,\ \mathbf{r}_{i^y} + (\sin(\theta) * d)]$

LCH Variation 4: Member-centric LCH Bearing (Corresp. to Fig. 1(d))

Input: Set of detected group members ($\mathbf{I}$) in egocentric coordinate frame.

Parameters:

$d \doteq$ *distance to travel*

$\mathbf{r}_i \doteq$ *current pose*

Output: Desired pose in a member-centric coordinate frame.

if $|\mathbf{I}| = 0$ **then**
  return $[0,\ 0]$
else
  $\theta \leftarrow \text{AverageBearing}(\mathbf{I})$
  return $[\mathbf{r}_{i^x} + (\cos(\theta) * d)\,,\ \mathbf{r}_{i^y} + (\sin(\theta) * d)]$

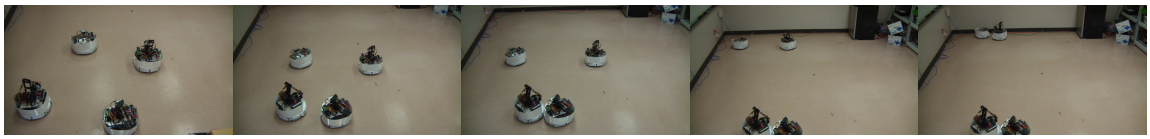
In both group-centric variations, the group member groups the detected other members based on a threshold (st) in the set $\mathbf{CC}$ (st could either be a distance or

angular based threshold). The control-law then selects the largest group from the set **CC** to compute the next pose. The function MaxConnComponent(**CC**) takes **CC** and returns the largest group of group members. The details of the group-centric variation using bearing information can be seen in Variation 3.

4.2 Robotic Implementation Specifics

The experiments here utilize four robots in the single-group starting formation (Figure 2.5) in a obstacle free space that can be considered infinite for the presented trials. Several trials for each of the four LCH variations were conducted; Figure 4.2 shows time series from a few of these trials. There is no significant difference observed in the behaviors produced by the four different variations of the LCH. It is observed that the behaviors produced by the robot implementation do not collapse into a single group, as observed in the simulations of Viscido *et al.* Viscido et al. (2002). However, the robots do indeed collapse into a single group, but because of the robot's limited field of view (FOV), the group exhibits directional motion. This directional motion occurs when one or more robots do not observe other robots in the group; therefore, they continue moving in their current direction; see Figure 4.3a. Computer simulations show that the FOV limitations of the robots is the primary cause of the motion differences.

Notice that Figure 7.6 does not show any results from trials using the group-centric variation because the size of the system is not large enough to show the desired behaviors. When the there is only one robot in the selected group, the robots will exhibit motions that can best be described as a *follow the leader* behavior. Figure 4.3b is a pictorial representation showing how the follow the leader motions are generated.



(a) Single robot trial with four robots running the member-centric pose variation.

(b) Single robot trial with four robots running the member-centric pose variation.

(c) Single robot trial with four robots running the member-centric bearing variation.

(d) Single robot trial with four robots running the member-centric bearing variation.

Figure 4.2: Each time series shows the motions of the multi-robot system running one of the four LCH variations.

4.2.1 The Effect of a Limited Field of View

Using MatLab (version R2011b) all four variations of the LCH similar to the implementation in Viscido et al. (2002) were implemented. Each group member has access to the global information for every other group member but each member is only able to *detect* group members within the sensing radius (r). Each group member will calculate their next pose ($\mathbf{r}_i$) according to the given LCH variation. Additionally, the group members FOV was included into the implementation in order to account for the limited FOV of the robots.

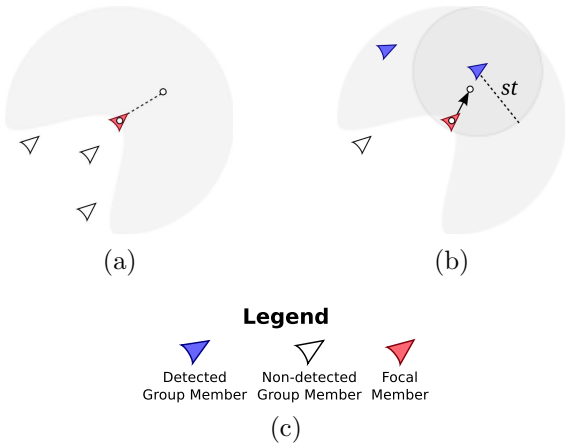
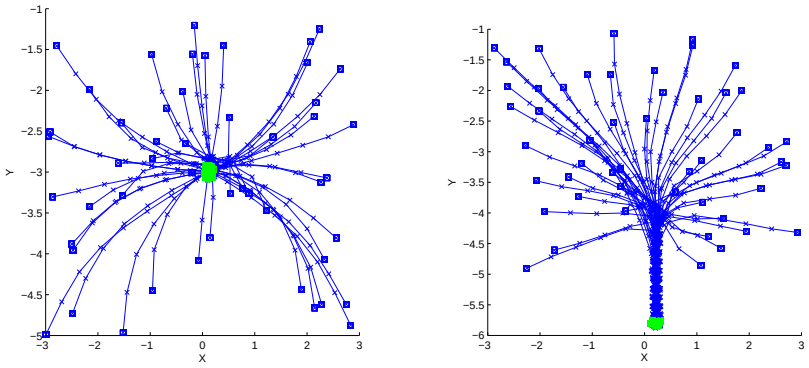


Figure 4.3: Figure 4.3a shows that a group member will continue in the same direction when no other group members are observed. This behavior causes the resulting motion of the group to be more directional than the computer simulations in Viscido et al. (2002). Figure 4.3b shows how the follow the leader behavior is generated when there are only a few detected neighbors.



(a) No limitations on FOV (b) Limited FOV

Figure 4.4: These two motion figures were generated from 50 simulated group members using the member-centric pose variation of the LCH. The blue squares represents the starting formation, which was randomly generated within a squared region, and the green squares represent the ending formation of the group members.

Figure 4.4a shows the simulated motions of 50 group members for the member-centric variation using pose with no limitations on their FOV. These motions reveal

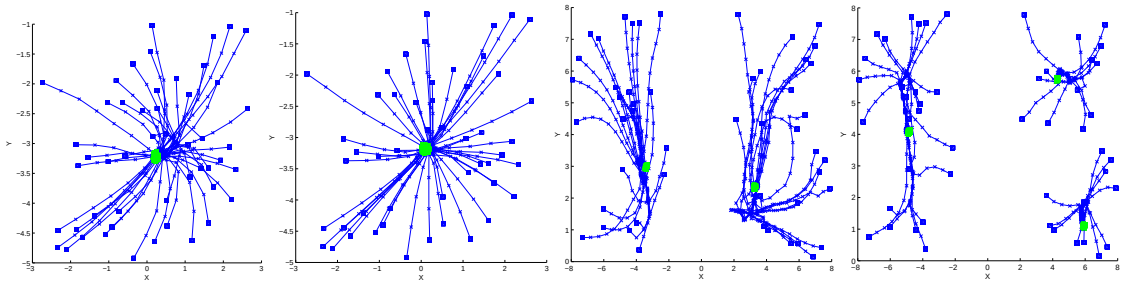
that the implementation of the LCH is similar to the LCH implementation in Viscido et al. (2002). Furthermore, the motions in Figure 4.4b show the simulated motions for the 50 group members with a limited FOV similar to the FOV of the robots. Comparing the motions in Figure 4.4b and Figure 7.6 it was noticed that the simulation, with a limited FOV, can produce equivalent motion to the robotic implementation. Since the implementation can produce the motions of the original motivating work and the motions produced by a robotic implementation, it is reasonable to assume the simulation is adequate for further investigation of information (type and detail) on flocking.

4.3 Information-abstracted Flocking

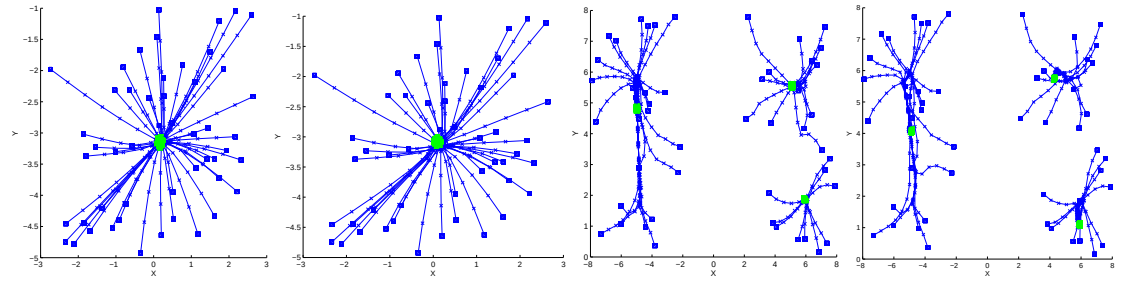
To investigate the existence of information-abstracted flocking algorithms, computer simulations for each of the 16 different parameterizations of sensing range (3000, which can be considered infinite, and 3 units), starting formation (single-group and split-group), type of information (pose and bearing), and observation detail (member-centric and group-centric) were conducted. Ten trials were conducted for each parameterization, each with random starting positions for the group members, resulting in a total of 160 simulations of 75 iterations for 50 group members. To aid in the study of the underlying motions produced by the chosen variations, the group members' FOV was not limited.

Using motion equivalence to compare the results in Figure 4.5, very little difference in the simulated motions is seen. Even when the worst case is considered, Figures 4.5c and 4.5g, there are no real differences in the resultant motion. Even though the motions are not identical, they still resemble the motions of the motivating biological flocks as presented in Viscido et al. (2002).

In addition to motion equivalence, Viscido et al. (2002) proposed the use of mean



(a) Pose (b) Bearing (c) Pose (d) Bearing



(e) Pose (f) Bearing (g) Pose (h) Bearing

Figure 4.5: These motion figures are typical results of simulations conducted for this study. Each trial simulates 50 group members over 75 iterations (only plotting every third pose per group member), where st_{distance} is set to 0.5 units and st_{angular} is set to 10 degrees. The top row of motion figures (figures 4.5a,4.5b, 4.5c, and 4.5d) were generated using group-centric information and the bottom row of figures (figures 4.5e, 4.5f, 4.5g, and 4.5h) were generated using member-centric information. The blue squares represent the starting positions of the group members, and the green squares represent the end positions of the group members. Motion figures 4.5e, 4.5f, 4.5a, and 4.5b were simulated with the single-group starting formation and a sensing range of 3000 units, where figures 4.5g, 4.5h, 4.5c, and 4.5d were simulated using the split-group starting formation and a sensing range of 3 units.

median distance (MMD) from the center of the group as a metric to describe the motions of a group that compresses during a predator attack. To supplement the observations of motion equivalence made form the motion figures in Figure 4.5, the MMD is calculated for all of the parameterizations over all trials. The computed MMDs for the trials that used a sensing range of 3 units are not significantly different

from the trials that used a sensing range of 3000 units, therefore only the MMDs from the trials that used a sensing range of 3 units are reported; see Table 4.1.

Table 4.1: The MMD (in units) from all of the simulations that had a sensing range of 3 units. For each simulation, the median distance from the center of all 50 group members is calculated from the ending formation. The medians from all ten trials were averaged to yield the MMD for the given parameterization.

	Single-group		Split-group	
	Pose	**Bearing**	**Pose**	**Bearing**
Group-centric	0.05	0.05	3.23	4.94
Member-centric	0.05	0.05	4.95	4.94

When the group starts in a single-group, Table 4.1 shows that there is no difference in the group's MMD for any of the 16 parameterization. When the group starts in two separate groups, there is a slight difference in the computed MMDs. As expected from the motions reported in Figure 4.5, there is a slight difference between the MMDs of the group-centric pose and member-centric pose trials when the group started in two groups. These results not only show that flocking motions are possible using group-centric information, but since the group-centric variation produces a smaller MMD, this suggests that group-centric information may be preferred in certain situations.

4.4 Case-study Summary

The ability to produce biologically motivated flocking motions using either group-centric or member-centric information suggests that flocking motions may be possible using a combination of the two. In other words, the group member may observe individual group members when they are nearby, but may still observe groups that

are further away. Furthermore, if one considers the case when group members only make observations based on image-size, it is plausible that the group member may not be aware of the differences between the two types of information. If this holds true, this may offer an explanation to how multiple flocks merge a split over time as occurs in large flocks of birds.

Additional thought is required if one is to understand what individual group members are sensing or computing. At least two distinct aspects are worthy of consideration: the detail involved in the observation and the type of information extracted from that observation. Moreover, metrics that are intended to evaluate the group's motion (*e.g.*, MMD) are also insufficient in of themselves. Therefore, future work may need to focus less on metrics designed to study the resulting group motion, and instead on the impact that the requisite information has on the biological plausibility, or the ability to implement the given algorithm on a robotic system, or both.

The preceding results show that the information available to a group member, while very important from an implementation and biological modeling point of view, are not necessarily distinguishable in terms of the flocking behaviors they produce. From these results and the results from Sections 2 and 3 it is clear that other approaches to understanding this phenomena must be explored.

5. UNDERSTANDING GROUP BEHAVIORS VIA CONTROL

To this point, this dissertation has explored bottom-up approaches to the understanding of flocking behaviors by investigating the individuals and their motions that give rise to flocking behaviors. Unfortunately, as highlighted in the previous sections, a complete understanding of these behaviors are still lacking. Even with limited models and incomplete understanding of the individuals, investigations have been capable of exploiting these groups to reliably produce desirable aberrant behaviors. The ability to reliably manipulate and control groups has been the subject of study from early human history to a diverse field of research today (Becker et al., 2013; Bobadilla et al., 2011; Butler et al., 2006; Couzin et al., 2005; Despland et al., 2000; Erickson, 2000; Grandin, 1980; Lien et al., 2004, 2005; Pereira et al., 2004; Petersen et al., 1994; Umstatter, 2011; Vaughan et al., 2000; Weiwei et al., 2012). The three approaches to group control seen in related literature are (1) using external agents, (2) using *informed* group members, and (3) adeptly constructed environments.

5.1 Group Control with External Agents

5.1.1 Shepherding

One of the more common approaches to influencing groups is through the use of external mobile agents (*e.g.* shepherds, sheepdogs, robots) that guide the given group (Lien et al., 2004, 2005; Pereira et al., 2004; Pierson and Schwager, 2015; Vaughan et al., 2000; Weiwei et al., 2012); referred from this point on as shepherding behaviors. These works typically utilize direct sensing of the target group to dynamically change the position or formation of the shepherd(s) responsible for controlling the group and assume that the individuals in the target group are (in some way) repulsed by the shepherds. Recent works have employed this technique for both a

single shepherd (Lien et al., 2004; Vaughan et al., 2000) and multiple shepherds (Lien et al., 2005; Pereira et al., 2004; Pierson and Schwager, 2015; Weiwei et al., 2012).

Using a single mobile robot, Vaughan et al. (2000) demonstrated a control-law for guiding a group of ducks to a desired location within a obstacle-free circular environment. To successfully guide the ducks, the robot (in this case a over-head tracking system) had to constantly sense the state of the group and change positions throughout the control task. The interesting aspect to this work was the use of a simple macroscopic description of the target group. Vaughan et al. (2000) employed a model that only required a centroid and radius of the encompassing circle. Even though this model is simplistic, it proved to be adequate due to the gregarious nature of the target group.

Lien et al. (2005) also presented strategies for guiding gregarious agents (similar to the ducks in Vaughan et al. (2000)), but in contrast to Vaughan et al. (2000), Lien et al. (2005) employ multiple mobile agents. The use of multiple shepherds allow for greater control over the group with less variation in the external agents position over time, but determining the formations for the shepherds is still lacking in rigor. For example, in Lien et al. (2005), the authors make assumptions on the basic form of the shepherds and does not derive these formations from interactions with the target group.

Similar to Lien et al. (2005), in that a basic form is already chosen, Pierson and Schwager (2015) approaches the multiple shepherd problem in a different manner than other known work in that they rely on certain geometric constraints between the shepherds. Given these constraints, Pierson and Schwager (2015) developed a continuous shepherding strategy that guarantees the target group to converge asymptomatically to the goal location. However, like in Lien et al. (2005), Pierson and Schwager (2015) still rely on a previously determined basic form for the shepherds.

5.1.2 Caging

Closely related to shepherding is a group control method known as caging. Caging is a common object manipulation method employed in robotic grasping applications (Diankov et al., 2008; Fukui et al., 2010; Rodriguez et al., 2012a) but has also been employed in object manipulation using a group of mobile robots (Fink et al., 2008; Wang and Kumar, 2002; Wang et al., 2004). Recently, this approach has been used to shepherd groups of autonomous agents instead of a ridged object (Pereira et al., 2004; Weiwei et al., 2012). These studies aim to design formations which prevent any members of the group to escape the convex hull of the shepherds.

These caging techniques are also seen in nature, dolphins when feeding on a school of fish. In open water environments, dolphins have been known to encircle (or cage) large schools of fish in order to entice the school into a small, dense mass, thus making feeding more efficient. Similar to dolphins, humpback whales utilize a feeding technique known as bubble net feeding. A group of whales will swim in circles while blowing bubbles around a school of fish (Leighton et al., 2004). As the whales circle they decrease the radius of the circle, thus resulting in tighter and tighter schools of fish. In other words, the whales are using bubbles to cage a school of fish in order to increase the density of the school before feeding.

5.2 Group Control with Internal Group Members

More closely related to a bottom-up approach to understanding flocking behaviors, some studies have investigated heterogeneous flocks (Conradt et al., 2009; Couzin et al., 2005; Gueron et al., 1996; Halloy et al., 2007; King and Cowlishaw, 2009). Work in this area can be split into to general categories: (1) use of informed group members or (2) use of directly controlled group members. **Informed** individuals are any group member that has some extra knowledge or desired behavior

(*e.g.* knowledge of food source location). **Directly controlled** individuals are group members that are either specifically trained to perform a particular task, behavior, or remotely operated group members. A good example of a directly controlled group members are *Judas goats*, which are used at livestock management facilities.

One dilemma with this approach is that it is not always practical to introduce informed individuals into a given group (*e.g.*, how would one introduce an *informed* group member into a flock of wild birds?). Furthermore, it is not always clear that the use of internal group members could elicit more complex behaviors, such as segregation based on agent characteristics (*e.g.*, separate the males from the females).

Recently, a new classification of an agent has been introduced by Yeh et al. (2008), known as a composite agent, in order to model the complex interactions in agent-based crowd simulations. **Composite agents** are agents (similar to the group members described above) that are associated with proxy agents. **Proxy agents** can be seen as additional behaviors added to the group member. With the addition of various proxy agents to certain group members, composite agents are capable of displaying a large variety of group behaviors.

5.3 Group Control via the Environment

Other work in group control has looked into the effects that the environment has on the exhibited behaviors of the group (Becker et al., 2013; Bobadilla et al., 2011; Butler et al., 2006; Despland et al., 2000; Erickson, 2000; Grandin, 1980; Petersen et al., 1994; Umstatter, 2011). Recently, investigations have started exploring methods for implicitly controlling autonomous agents by way of adeptly constructed environments (Becker et al., 2013; Bobadilla et al., 2011; Butler et al., 2006; Umstatter, 2011). Such work demonstrates that the environment can be designed in such a way that it can reliably induce a particular action from a given group of agents.

Bobadilla et al. (2011) show that a group of simple robots can be successfully *guided* to a specific location in space through an environment comprised of static walls and one-way passive gates. Their study used prior knowledge of the controller that the robots would employ to build an environment using geometric patterns that were capable of influencing the group. Similarly, Becker et al. (2013) shows that a large number of simple robots directed by a global broadcast signal can perform fairly complex behaviors and tasks without explicit communication within the group given a properly structured environment.

Taking more of a field application angle to this problem, Butler et al. (2006) has shown applications of this approach by herding grazing cattle in open fields. The authors employ a combination of GPS enabled shock collars and the notion of virtual boundaries. When cattle approach a virtual wall they receive a mild shock which forces them to move away from the virtual wall. The algorithms presented in Butler et al. (2006) relocate the virtual walls based on the desired behavior of the grazing cattle.

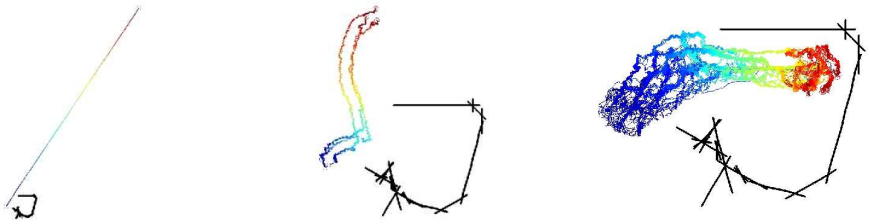
The above approaches and the work presented here are similar to the work done in automatic part manipulation and sorting (Bohringer et al., 1995; Erdmann and Mason, 1988). These works attempt to orient ridged objects by applying a combination of forces to the objects. The key difference between part manipulation and group control, as studied here, are the models used for the planning of the control actions.

Although there has not been much work done for this other than the few works mentioned above (Becker et al., 2013; Bobadilla et al., 2011) there are works in shape grammars (Gips, 1974; Stiny, 1980), product design (Hsiao and Chen, 1997; Orsborn et al., 2006), and even urban planning applications (Halatsch et al., 2008; Koutsourakis et al., 2009). A large number of these publications do not consider the

semantics of the *environments* that are generated. In the ones that do, the semantics are evaluated based on the environments themselves and not between the interactions of another agents or group.

There are a handful of real-world example environments (fish weirs and cattle handling facilities) that have been designed to effectively and reliably control particular groups into eliciting specific behaviors. However, the creation of these environments generally rely on domain experts and, in some cases, unformalized domain knowledge (Grandin, 1980). This (*ad hoc*) approach limits the variety of group and behavior pairings that can be investigated. To address the lack of generality, there needs to be a way of automatically generating useful environments.

The key aspect of the fish weirs that is distinct from many of the other works is that the weirs are static environments that are still capable of controlling groups of individuals. Such environments exploit the group's structure (*e.g.*, aggregation) in order to control the given group. Figure 5.1 shows three simulations of different size group's obeying the BOIDS flocking model. These simulations show that only a group of sufficient size is successfully corralled within the environment.



(a) Individual (b) Group of Three (c) Group of Fifty

Figure 5.1: These simulation results were generated using identical parameters with exception to group size. Together these results show how the corralling behavior is only elicited when the group is of sufficient size (Figure 5.1c).

The remainder of this dissertation will use the fact that the collective structure of groups can be exploited by static environments. If one could develop/design environments that could exploit a given group in various ways, one may be able to better understand the group as a whole. This dissertation will present a methodology for automatically generating such environments.

5.4 Generating Environments for Group Control: Problem Definition

An **environment** (E) is considered to be an enumerable set of simple geometric shapes known as primitives. Each **primitive** (P) is a finite set of continuous and/or discrete attributes that rigorously define a geometric shape; see Definition 1. Example primitives include a 2-dimensional line segment defined by two sets of Cartesian coordinates, a circle defined by coordinates and a radius, and a polygonal shape defined by a set of line segments.

Definition 1. *Let* $\mathcal{P} = \{\emptyset, P_1, P_2, \ldots, P_\varphi\}$ *denote the set of all possible primitive types. Where a primitive* $P = \{\alpha_1, \alpha_2, \ldots, \alpha_\gamma\}$ *and* $\alpha_i = \{\alpha^i_{min}, \alpha^i_{max}\}$ *for continuous primitives or* $\alpha_i = [\alpha^i_1, \alpha^i_2, \ldots]$ *for discrete.*

The problem of generating an environment e, given a set of primitives $\mathcal{P}$, becomes a matter of planning over all possible combinations of the primitives. The set of possible primitive combinations is further reduced by a set of constraints (C). These constraints detail the initial workspace (W), the procedures for adding primitives to the set e (C_g), and constraints on feasible sets (C_f). Definition 2 defines the constraints on possible environments. For this dissertation a **construct** refers to the geometry generated by C_g and an environment is the result after being applied to the given workspace. If the workspace is the empty set then the construct and the environment are identical.

Definition 2. *Let $C = \{C_g, C_f, W\}$ where C_g denotes constraints on the generation procedures, C_f denotes constraints on feasible generations, and W denotes an initial geometric representation of a 2- or 3-space workspace.*

The **group of agents** (A) is a homogeneous group of mobile individuals that have some degree of autonomy. All group members in A follow the same motion model, which can be either deterministic or stochastic. The **motion model** (m) represents the low-level control law each agent will obey when operating in the environment (*e.g.*, Reynold's rules (Reynolds, 1987)) and is encapsulated in the definition of A. For reviews of control-laws for autonomous agents that exhibit some level of collective structure please see Edelstein-Keshet (2001); Giardina (2008); Goldstone and Janssen (2005); Lerman et al. (2005); Parrish et al. (2002); Partridge (1982), or refer back to Section 3.

The specified **behavior** (b) can be any group level behavior ranging from simple point-to-point navigation to segregation based on agent classification (*e.g.*, removing the female sheep from the herd). Each behavior is defined as a function that takes, as input, position of each member in A over a given time and the environment, and returns a fitness value between 0 and 1; with one being complete compliance with behavior. More formally

Definition 3. *Let $b(\Pi) \rightarrow (0, 1]$ where $\Pi = \{\pi_1, \pi_2, \ldots, \pi_{|A|}\}$, and π_i denotes the trajectory of the i^{th} agent in A.*

Given the above definitions, the complete definition of the **Multi-Agent Environmental Behavior Elicitation (MAEBE)** problem is as follows:

> Given a set of mobile agents A, a behavior b, a set of environmental primitives $\mathcal{P}$, and a set of generation constraints C. Does there exist an environment e constructed from $\mathcal{P}$ satisfying C, that elicits b from A?

5.5 k-MAEBE Variant

For the remainder of the work we are going to consider a variant of the MAEBE where we have a finite set of primitives. It is important to note, we are not restricting the number of primitives, just the different types of primitives. For example, the restricted set of primitives may only include a line segment primitive (wall) and a wedge style primitive. The formal definition of the **k-MAEBE** problem is as follows:

> Given a set of mobile agents A, a behavior b, a set of k environmental primitives selected from $\mathcal{P}$, and a set of generation constraints C. Does there exist an environment e constructed from the selected k primitives satisfying C, that elicits b from A?

6. APPROACH AND IMPLEMENTATION TO GROUP CONTROL *

To solve the MAEBE problem without relying on experts or specific knowledge, we break the problem down into three key aspects; (1) primitive generation, (2) environment generation, and (3) environment validation. First, let us focus on both the environment generation and validation. We assume that the set of primitives is given, but will lift this assumption later (Section 8.1). A diagrammatic overview of the system considering only the generation and validation of environments can be seen in Figure 6.1.

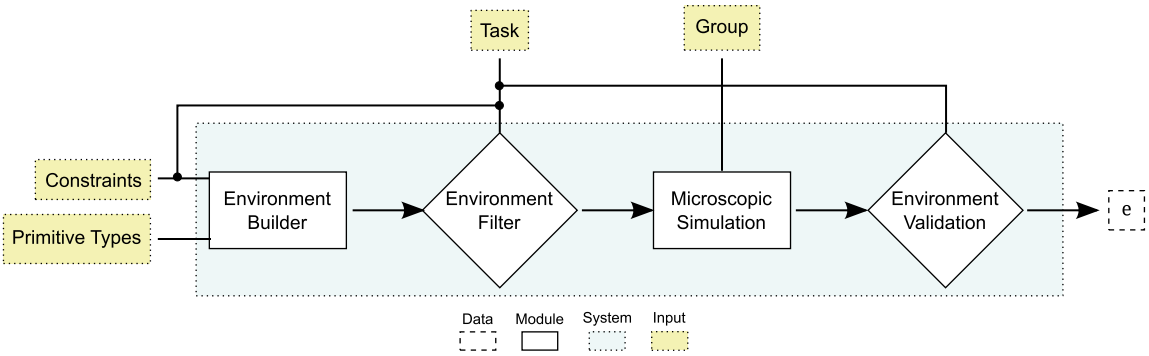


Figure 6.1: A diagram showing the proposed method for automatically enumerating a set of environments that elicit a particular behavior from a group. The environment schema (C_g), group (A), and behavior (b) are all user defined inputs to the system and remain constant during the system's execution. The output is a set of environments (E) that can elicit the specified behavior from the given group.

*©2013 IEEE. Part of the work presented here is reprinted with permission from "Eliciting collective behaviors through automatically generated environments" Benjamin T. Fine and Dylan A. Shell, 2013. IEEE/RSJ International Conference on Intelligent Robots and Systems.

6.1 Environment Generation

To increase the generality and applicability of our method we treat the motion model of the agents as a black-box. Without any information to how the group behaves and interacts with the environment, we needed to develop a way to enumerate through a large set of environments. To do this, we employed environmental schemas (C_g). Simply put, **environmental schemas** are rules for how to combine primitives in order to construct environments. In this study, we consider three general types of schemas; (1) shape grammars, (2) computational schemas, and (3) hierarchical schemas.

The general form of the proposed environment generation algorithm can be seen in Algorithm 3. The `GenerateCandidateEnvironments` function is the user defined environment schema. For this work, this function is either a shape grammar or a computational schema.

Algorithm 3 General Environment Generation (GEG)

Input: $k, \mathcal{P}, C, A, b, \rho$
Output: A set environments E generated from $\mathcal{P}$ given C that elicits b from A.

1: $E_{cand} \leftarrow$ `GenerateCandidateEnvironments` $(k, \mathcal{P}, C)$
2: $E_{cand} \leftarrow$ `AddCandidatesToWorkspace`(E_{cand}, W)
3: **for** $e \in E_{cand}$ **do**
4: **if** `ValidateEnvironment` (e, b, A, ρ) **then**
5: $E \leftarrow e$

6.1.1 Shape Grammars

Shape grammars, first introduced by Stiny (1980), are similar to typical symbol grammars (Chomsky, 1956) in that they have a set of symbols (primitives) and rules

(C_g), that together, generate syntactically valid strings (environments). The key difference in a shape grammar is that the rules encode spatial and geometric properties, such as pose and orientation. In this study we only consider two-dimensional polygonal environments but, in general, shape grammars can be used in higher dimensions (Chau et al., 2004) and with non-polygonal shapes (Jowers and Earl, 2010). For the original definitions and a more detailed treatment of shape grammars, please see the seminal shape grammar study of Stiny (1980).

The general form of grammar based schemas can be seen in Algorithm 4. In the Appendix, we show that algorithms of this form are probabilistically complete and their time complexity is exponential with respect to the number of possible permeations of the k selected primitives.

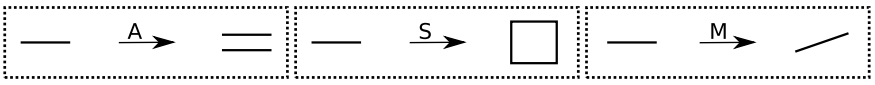
Algorithm 4 Grammar-based Candidate Generation (GCG)

Input: $k, \mathcal{P}, C$
Requires: ϕ, denotes number of candidate environments to generate.
Output: A set environments $E_{cand} = \{e_1, e_2, ..., e_\phi\}$.

```
1: while |E_cand| < φ do
2:     e ← GenerateInstance (k, P, C_g)
3:     if FilterEnvironment(e, C_f) then
4:         E_cand ← e
```

The basic structure of a shape grammar consists of left-side and right-side shapes. Left-side shapes represent which shape the rule will apply to and the right-side shape represents the final shape after the rule is applied. Potential rules range from addition, where another shape is added to the left-side shape (Figure 6.2a), to substitution, where the left-side shape is replaced by another shape (Figure 6.2b). Another common rule found in shape grammars is modification, where the left-

side shape is modified is a particular way (*e.g.*, the left-side shape is rotated by 15 degrees); see Figure 6.2c.



(a) Addition (b) Substitution (c) Modification

Figure 6.2: Three common rules used in shape grammars. Here, the straight line and the square would be considered primitives.

6.1.2 Computational Schemas

Computational schemas are (generally) more programmatic and predictable than shape grammars. Shape grammars are useful tools when the structure of semantically valid environments is unknown, but when the structure of the desired environments is known, shape grammars may be less efficient than computational schemas. For this work, when we refer to the efficiency of a grammar we mean the percentage of environments generated by the given schema that elicit behavior b from A.

The general form of a computation schema can be seen in Algorithm 5. In the Appendix B, we show algorithms of this form to be resolution complete and we show the expected runtime is exponential.

Take for example a Galton board (Kozlov and Mitrofanova, 2003), as depicted in Figure 6.3. It is possible to generate a shape grammar for generating Galton boards but the rules would be overly complex for the underlying structure. Instead, one could simply mathematically define the given structure with a few parameters, thus producing a less complex and potentially more accurate schema. We consider

Algorithm 5 General Computational Candidate Generation (GCCG)

Input: $k, \mathcal{P}, C$

Requires: $\mathcal{R} = \{R_1, R_2, ..., R_\varphi\}$, where $R_i = \{r_1, r_2, ..., r_\gamma\}$, denotes the resolutions for each attribute of each primitive type.

Output: A set environments E_{cand}.

1: $E_{cand} \leftarrow$ `GenerateAllPermutations`$(k, \mathcal{P}, C, \mathcal{R})$

accuracy to be the percentage of valid environments generated by a schema. For example, a schema generates valid environments 90% of the time is considered more accurate than a schema that generates valid environments 50% of the time.

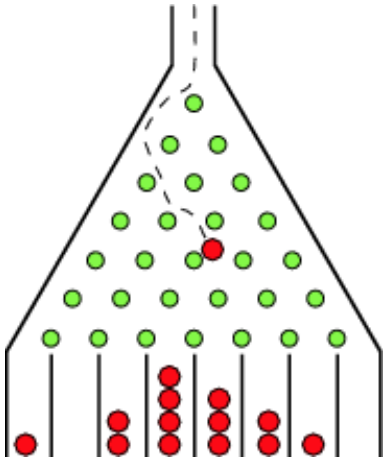


Figure 6.3: Pictorial representation of a Galton board (the pegs arranged in a triangle) along with a funnel and bins for containing the *agents*

.

6.1.3 Hierarchical

Hierarchical schemas are combinations of other schemas (any combination of both shape grammars or computational schemas) that together form a larger schema. The motivation for this schema style can be seen when considering complex behaviors,

such as gathering and sorting male and female cattle from a grazing herd. There are three key sub-behaviors that can be identified: (1) corralling the cattle, (2) moving the cattle to a line formation, and (3) sorting the cattle based on sex.

To develop a single schema for all three sub-behaviors (while not impossible) would be very challenging. The aim of a hierarchical schema would be to merge a schema for each of the three desired behaviors. Thus allowing simpler schemas to be used in joint to generate environments that can elicit more complex behaviors from a group.

6.2 Environment Validation

After the system generates a syntactically valid environment based on the chosen schema we must check if the environment is semantically valid. A **semantically valid** environment is any environment that elicits the desired behavior b from the given group A. The semantic validation is conducted in two stages. First, each generated environment must exist in the set of feasible environments defined by C_f. For example, if the user of the system only has 100 meters of fencing, then an environment using more than 100 meters would be considered outside the set of feasible environments.

Once the environment is determined to be feasible, the environment must be validated on semantics (*i.e.*, does e elicit b from A). The presented approach uses a microscopic simulation of A following the motion model m. If the resulting fitness value computed by b is above a given threshold, then the generated environment is considered semantically valid, and thus added to the set E.

6.3 System Implementation

6.3.1 Shape Grammars

We implemented two different (rather simplistic) grammars using the Shape Grammar Interpreter[†] (SGI) given in (Trešťák et al., 2009). The implemented shape grammars use a combination of non-terminal shapes, terminal shapes, and markers. Both **non-terminal shapes** and **markers** are only used in the generation of the environment and are never part of the environment itself. These shapes are used to help define underlying structure of the environment. In other words, if one was to physically build an environment consisting of all three shape types, only the terminal shapes would be constructed. The only difference between non-terminals and markers are that markers are used by b to compute the fitness value (*e.g.*, goal location).

Terminal shapes, such as walls, are shapes the agents will interact with during the validation procedure and are items we must physically construct. Figure B.1 details the two shape grammars implemented for the validation of the proposed system. To aid in a better understanding of how the grammars presented in Figure 6.5 work, Figure 6.4 is a step by step walk through for the generation of a single environment using the splitting grammar.

6.3.2 Predefined Behaviors

For the validation of the proposed approach and system implementation, we consider two behaviors; *splitting* and *corralling*. These behaviors where chosen because they are commonly exhibited by real (biological and/or robotic) systems. Splitting behaviors are defined by any behavior where the group fragments into at least two groups for k consecutive simulation steps. We consider a group to be performing

[†]SGI version 1.31 from the Source Forge repository (Trešťák, 2012).

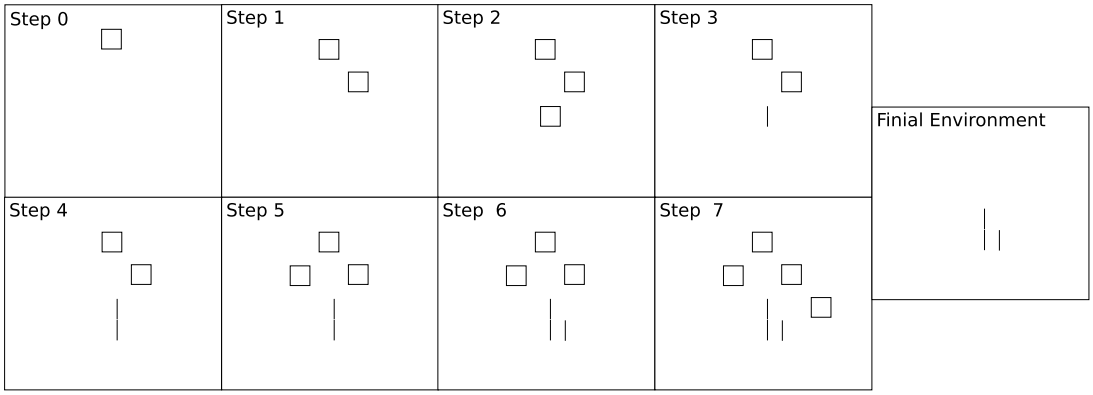
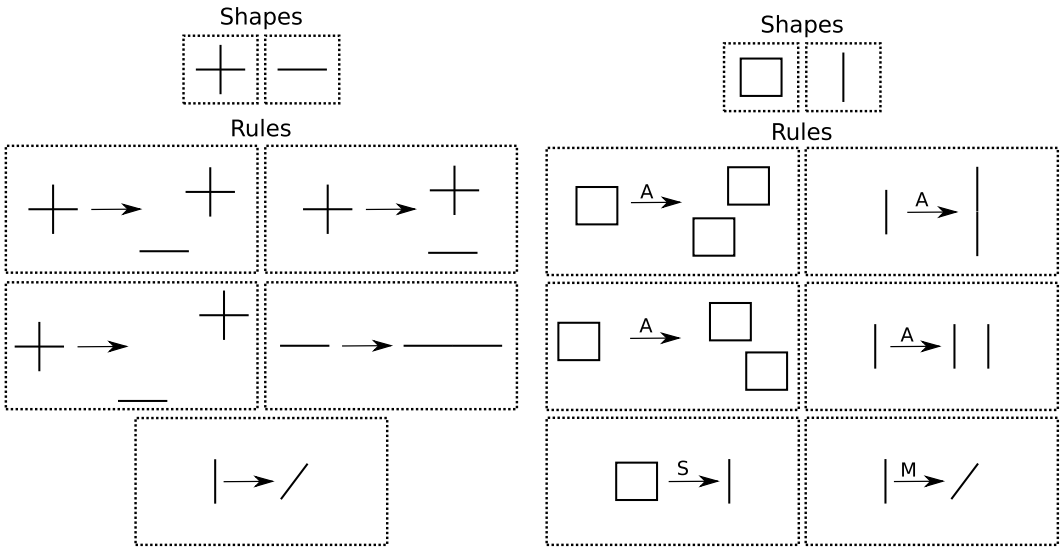


Figure 6.4: A simple execution of the splitting grammar. Step 0 is the starting shape and a non-terminal. The only primitives that are physically realized are the line primitives.



(a) Splitting Grammar (b) Corralling Grammar

Figure 6.5: The splitting grammar uses a non-terminal shape (square), which is used to help define a tree-like structure of the environment and a single terminal shape (straight line). The weiring grammar uses one terminal shape (straight line) and one marker (cross) that defines the predesignated area for the corralling behavior.

the corralling behavior if and only if the group does not fragment and the centroid of the group remains within radius r from the origin (the marker in the corralling

grammar) for at least k consecutive iterations of the simulation.

The definitions of the behaviors can be seen in Table 6.1. Each behavior returns a binary fitness value. A value of 1 if the behavior is exhibited and a value of 0 if it is not.

Table 6.1: Specifications of the behaviors used in this study.

ing

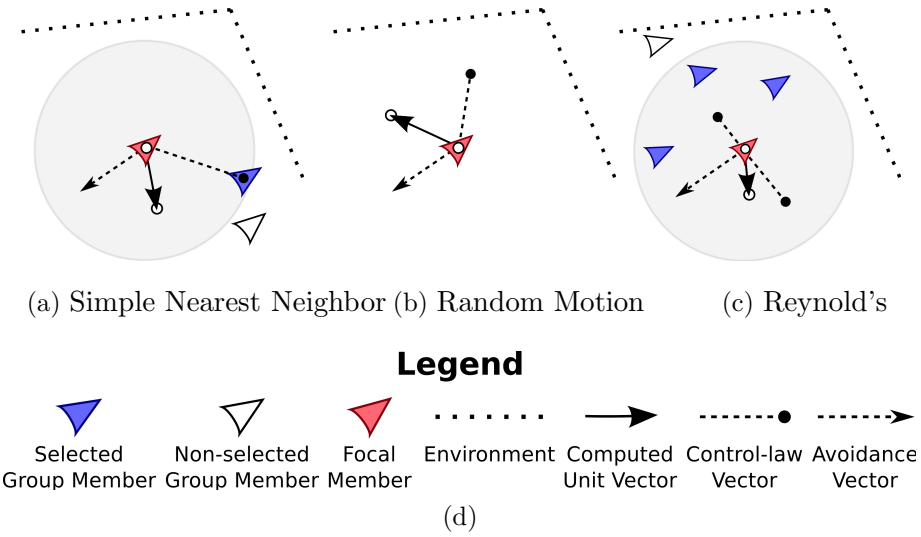
Simple Split A group exhibits this behavior when the agents fragment into at least two distinct groups for $k = 50$ consecutive iterations. Group membership is determined based on the fragmentation threshold of 25 units. This behavior must be elicited within 200 iterations.
Balanced Split A group exhibits this behavior when the agents fragment into at least two distinct groups for $k = 5$ consecutive iterations. The groups are considered balanced when the entropy of the system is at least 30% of the maximum entropy. Group membership is determined based on the fragmentation threshold of 25 units. This behavior must be elicited within 200 iterations.
Corralling A group exhibits this behavior when the centroid of the agent's position is within a threshold distance of 25 units of the origin $(0, 0)$ for at least $k = 50$ consecutive iterations. Additionally, the group must not fragment (based on the fragmentation threshold of 25 units). This behavior must be elicited within 500 iterations.

6.3.3 Microscopic Models

For the microscopic models, we implemented three flocking models that are pulled directly from the flocking literature; (1) Random Motion Plus[‡] Viscido et al.

[‡]The *plus* term signifies the addition of environment avoidance capabilities to the motion model if it was not included in the original model.

(2002) (RM+), (2) Simple Nearest Neighbor Plus Viscido et al. (2002) (SNN+), and (3) Reynold's Boids Reynolds (1987) (BOIDS). These motion models where chosen because they either represent a staple motion model in the literature (SNN+ and BOIDS) or they are useful for control cases in the validation process (RM+). Figure 6.6 is a pictorial representation of the three motion models implemented.



(a) Simple Nearest Neighbor (b) Random Motion (c) Reynold's

(d)

Figure 6.6: Motion models used in this work. The gray circle centered on the group member is the interaction radius for selecting neighbors.

It is important highlight that the proposed methodology and implementation in this dissertation is agnostic to the chosen model. The generation of environments does not consider the model of the group members; the individual's model is only utilized during the validation procedure. Thus, it does not matter if the group is heterogeneous, homogeneous, or comprised of composite agents, which allows one to study the general problem of group control and not just specific instances.

7. EXPERIMENTS AND RESULTS *

7.1 Hardness of MAEBE

To show the hardness of the MAEBE problem, we reduce the Warehousemans's problem (a multi-agent motion planning problem), which was shown to be PSPACE-Hard in Hopcroft et al. (1984). The definition of the Warehouseman's problem is as follows:

Definition 4. *Given a set of rectangular objects A, a two-dimensional rectangular box R, a starting configuration q_s, and an ending configuration q_g. The* Warehouseman's Problem *is as follows: Does there exist a collision-free path for A from q_s to q_g?*

Theorem. *The MAEBE problem is PSPACE-hard.*

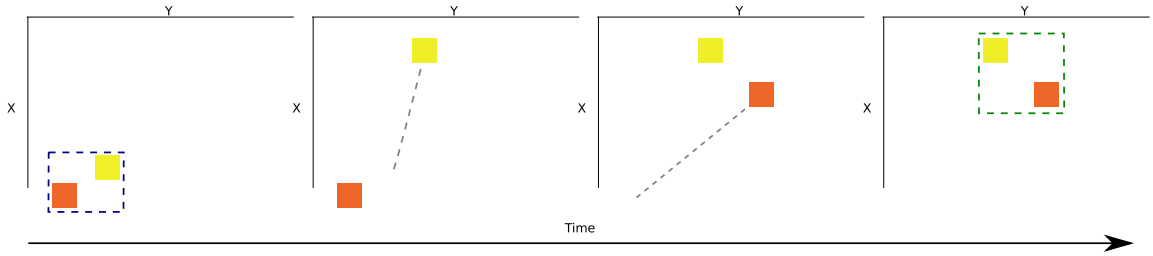
Proof. To prove that the MAEBE problem is PSPACE-hard, we will give a polynomial-time reduction of the Warehouseman's Problem to the MAEBE problem, and then we will prove the correctness of such a reduction.

Essentially, each agent in the MAEBE problem will follow paths (tunnels) in a 3-dimensional space according to its control-law. A set of tunnels (one per agent in A) defines a solution, such that each (x, y, z) coordinate of the tunnels map to a (x, y, t) coordinate for a path in 2-dimensional space, where t is time. The path will be collision-free per the constraints needing to be satisfied in the reduction (*i.e.*, no tunnel (path) intersections).

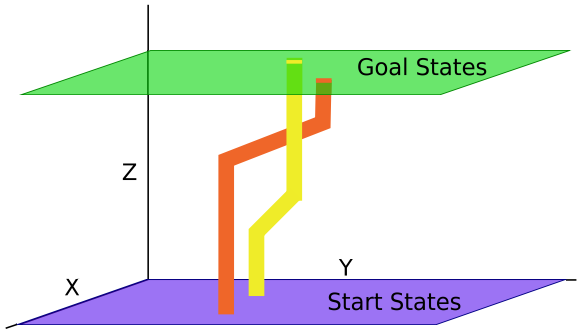
Reduction Given an instance of the Warehouseman's problem $wp = \{A, R, q_s, q_g\}$, we map wp to an instance of the MAEBE problem $maebe = \{A, b, \mathcal{P}, C\}$ as follows and is shown in Figure 7.1:

1. $A = A$, as in each rectangular agent in wp corresponds to an agent in $maebe$.
2. m (encapsulated in A) is a tunnel following routine by which each rectangular agent $a \in A$ is restricted to motions through tunnels.
3. b is a move to goal behavior from q_s to q_g, as in the agents must arrive at q_g.
4. $P = \{tunnel\}$. Each tunnel has a position, rotation, and elevation which corresponds to a translational motion of a rectangular agent through time or a waiting motion. Each tunnel can have the width of any agent in A such that each agents swept volume of motion will be contained within a tunnel.
5. C contains two constraints: (1) none of the primitives can intersect in 3–space and (2) a tunnel must lead from each starting position of the agents with the z component being 0, and similarly, a tunnel must reach each goal position of the agents.
6. $W = \{[R_x, R_y, z]\}$ where z is unbounded.

The reduction of the decision problem can be seen as: Find a three-dimensional environment e composed of non-intersecting tunnels to lead all agents from a starting position to a goal location, or determine that none exists. This is a linear time reduction. It is linear in the number of agents needing to be mapped, as all other components are constant time.



(a)



(b)

Figure 7.1: Reduction from a Warehouseman's problem instance (a) to an instance of the Multi-Agent Environmental Behavior Elicitation problem (b). Each piece-wise linear tunnel from a start to goal position is equivalent to a time-varying trajectory of a rectangle in the Warehouseman's problem.

Reduction Correctness: Each agent will follow the tunnels, as its control-law dictates. As stated above, the solution of tunnels lays out a path such that each (x, y, z) coordinate of the tunnels map to a (x, y, t) coordinate for a path. The path will be collision-free per the constraints needing to be satisfied in the reduction (*i.e.*, no tunnel (path) intersections). Thus, each *yes* answer to *maebe* maps to a collision-free path for the corresponding *wp*. A *no* answer implies that no path exists because if a path had existed then an environment of tunnels would exist to elicit such a behavior. □

7.2 System Validation: Simulation Results

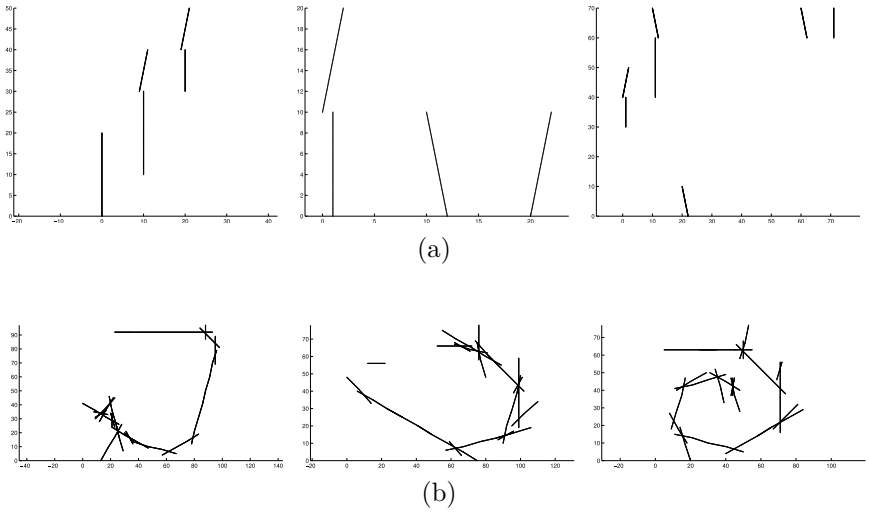
Using MatLab (version R2011b) we implemented the three selected motion models, both the splitting behaviors and the corralling behavior (Table 6.1), two environment filters for both of the implemented shape grammars (Table 7.1), and the framework of the proposed system (excluding the SGI software). We generated a 1000 environments with the corralling grammar and a 1000 environments for the splitting grammar. The same 1000 environments generated by the splitting grammar were used for both the simple split and balanced split behaviors. The results presented here show that the system successfully generates, filters, and validates environments that elicit a given behavior b from a group A.

Table 7.1 shows the percentage of environments for both the splitting and corralling grammars that passed the filtering process. In other words, Table 7.1 shows the percentage of environments that are in the feasible set of environments defined by C_f. In both instances the filters discard roughly 50% of the generated environments. Figure 7.2 shows example environments that were generated by the SGI software and passed the filtering process. It is clear what type of splitting environments would fail the filtering process (thus not pictured here), but Figure 7.3 shows examples of environments that failed the filtering process for the corralling grammar.

For each environment that passed the environment filter, the system conducted a single simulation trial to determine if the environment elicited the specified behavior from the implemented motion models. Table 7.2 shows the results from the validation process using 25 homogeneous agents for each simulation. In the case of the worst performance (SNN+ and the corralling behavior), 22.4% of the generated environments elicited the specified behavior, and in best case (RM+ and the balanced split behavior), 46.8% of the generated environments elicited the specified behavior.

Filter	Definition	Pass Rate
Splitting	The environment must contain at least three terminal shapes.	53.7%
Corralling	The origin marker must be within the convex hull of the terminal shapes and there must be a obstacle free path from the origin marker to an arbitrary point outside of the convex hull of the terminal shapes.	50.1%

Table 7.1: Definitions and percentages of environments that passed the filtering process.

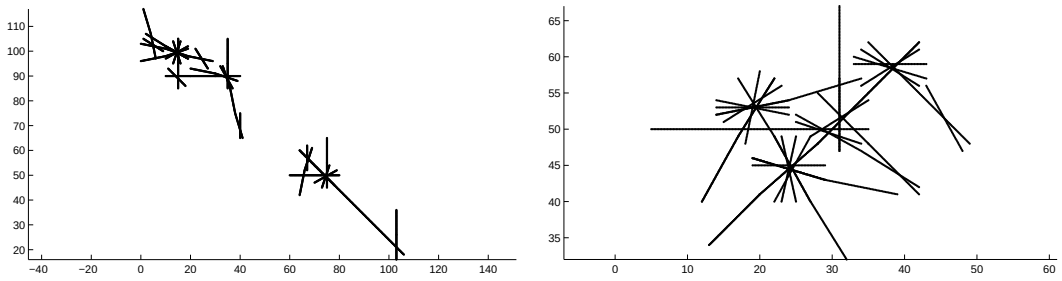


(a)

(b)

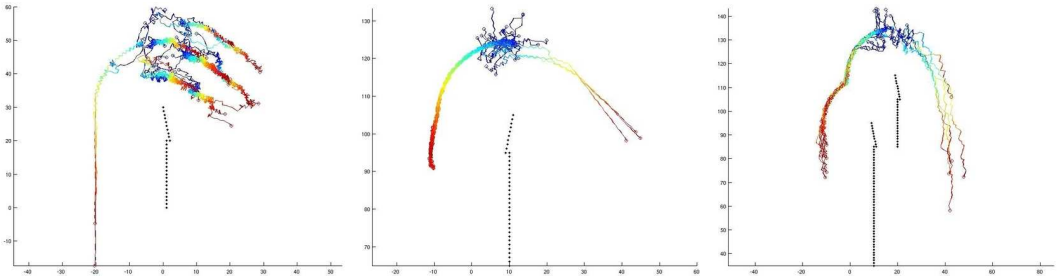
Figure 7.2: Both figures show three environments that were generated using the splitting and corralling shape grammars and that passed the respective filters defined in Table 7.1.

Figure 7.4 are simulation results from three environments that elicited the simple split behavior from groups executing the three motion models. For all plots in the remainder of this publication, the color gradient (blue to red) represents simulation time.



(a) (b)

Figure 7.3: Both figures show environments that were generated using the corralling shape grammar but *failed* the corralling filter.



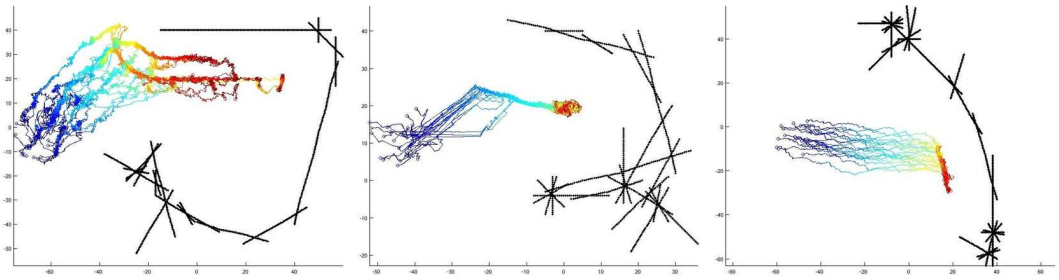
(a) BOIDS Model (b) SNN+ Model (c) RM+ Model

Figure 7.4: Three different environments that successfully elicit the simple split behavior for each motion model.

7.3 System Validation: Robot Results

To support the claim that the environments generated and validated by the presented system can reliably control multi-robot systems (or even biological agents), we conducted robot trials with four iRobot Creates. Each robot is equipped with a Hokuyo URG-04LX-UG01 laser range finder for all sensing requirements and an ASUS Eee PC for control. Each robot is also marked with reflective tape to aid in neighbor detection.

We constructed environments that were generated and validated by our system



(a) BOIDS Model (b) SNN+ Model (c) RM+ Model

Figure 7.5: Three different environments that successfully elicit the corralling behavior for each motion model.

Table 7.2: The percentage of environments that generated the desired behaviors for the three motion models. These percentages only consider the number of environments that passed the filtering process and are rounded to the nearest hundreth.

	Reynolds	Simple Nearest Neighbor	Random
Simple Split	0.49	0.70	0.86
Balanced Split	0.49	0.71	0.87
Corralling	0.60	0.45	0.51

for both the splitting and corralling grammars. Figure 7.6 shows time series from two separate successful robot trials, and Tables 7.3 and 7.4 show the results of all 30 robot and comparable simulation trials.

It is important to note, for the robot trials, we slightly modified the corralling behavior to include groups that only maintain two-thirds connectivity. In other words, if two of the three robots are consider to be performing the corralling behavior, then the environment is said to elicit the specified behavior. This modification is advantageous because of (1) the noise in the sensing, perception, and action of the robots, (2) the difficulties in scaling the environment, and (3) the fewer number of agents means that proportions for single agent events are larger.

Table 7.3: Validation results for two environments for *robotic* agents. Each behavior was tested with one environment selected from the set of valid environments and conducted over five trials, totaling 30 trials. The simple split behavior was tested using four agents and the corralling behavior was tested using three.

	Reynolds	Simple Nearest Neighbor	Random
Simple Split	0.40	1.00	0.80
Corralling	0.20	0.20	0.00

Table 7.4: Validation results for two environments for *simulated* agents. Each behavior was tested with one environment selected from the set of valid environments and simulated over five trials, totaling 30 simulation trials. The simple split behavior was tested using four agents and the corralling behavior was tested using three.

	Reynolds	Simple Nearest Neighbor	Random
Simple Split	0.60	0.80	1.0
Corralling	0.40	0.20	0.00

(a) Splitting

(b) Corralling

Figure 7.6: These two time series shows a multi-robot system obeying the SNN+ motion model being influenced by environments that were automatically generated by the implemented system.

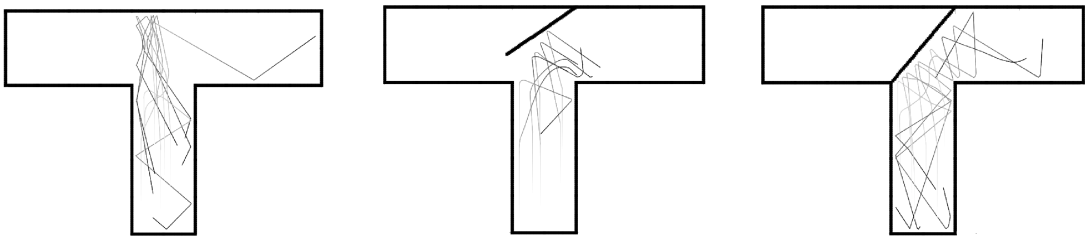
7.4 Evaluating the Set of Generated Constructs

The results to this point show through simulations and robot trials that the presented solution can indeed solve the MAEBE problem. However, the experiments in Section 7 do not give one a notion to the *tightness* of the set of generated constructs. Here, **tightness** refers to the percentage of generated constructs that elicit the desired behavior from the given group. In other words, if 90% of the constructs produced by *Schema A* elicit b from A and only 80% of the constructs produced by *Schema B* elicit the behavior, then Schema A is considered tighter than Schema B.

This notion of tightness is important when considering the applicability of the methodology to real world applications. Determining the tightness value is straight forward once a large number of constructs have been generated, but how can this value be used to inform the chosen schemas, or even parameters with in the schemas? To explore this, the preceding section investigates the connection between the robustness of the group's behavior to changes in the produced constructs.

The commonality among much of the literature discussed here is it relies on the interplay between the environment and the group level dynamics exhibited by the agents (*e.g.*, cohesion, repulsion, polarization, velocity matching). Unfortunately, it is difficult to quantify, or even identify, the key interactions that determine the exhibited behaviors of the group. As a result, it is challenging to determine how robust the group is to perturbations in the environment. Figure 7.7 shows how changes to the environment can effect the exhibited behaviors of the group.

Without a good understanding of the group's sensitivity to the changes in the environment, it is difficult to evaluate and/or improve current environmental designs for group control. Current methods for generating such environments (Becker et al., 2013; Bobadilla et al., 2011) are useful for identifying a single, or a small number of,



(a) (b) (c)

Figure 7.7: Motivating examples of how small perturbations in the environment can effect the exhibited behaviors of a group. Figure 7.7a shows the effect the workspace has on the trajectories of the simulated agents, where Figure 7.7b and 7.7c show examples of how small changes can drastically change the groups' exhibited behavior. The desired action for these motivating examples was influencing at least 90 percent of the group to turn right at the junction. Shading of the trajectories represent time starting with white and fading to black.

instance(s) of environments that successfully (and reliably) induce a specific action from a particular group. However, these methods do not give one a notion to what the set of environments looks like or what properties it may have. In other words, we would like a way in which to enquire about the set of generated environments. We do so by exploring the parameter space of the generation methods used. Questions of interest would include:

- Are there regions in this space that have a high density of useful environments?
- What is the connectivity in these regions (*i.e.*, how sparse are the regions of high density)?
- What properties of the group can be inferred from the structure of these regions in the design space?

To begin to answer these and other related questions, this work empirically maps the parameter space of two different computational schemas through the use of com-

puter simulations. We identify two properties (density and connectivity of regions) of these parameter space maps that allow us the ability to describe the sensitivity of a group of agents obeying the well-known Reynolds' flocking rules (Reynolds, 1987). Additionally, we discuss the relationship between these properties and our ability to explore the full set of realizable environments.

7.4.1 Environment Generation

Methods for generating environments to influence a group do not typically start from a blank slate, they must account for (and potentially exploit) the geometry of preexisting structures like hallways in a building or trees in a field. As defined in Section 5, these preexistent structures, are the workspace (W). The environments studied here can be broken down into two main parts; (1) the workspace and (2) the physical construct that was generated to influence the group. Multiple schemas are explored in this section, but each take, as input, a workspace (W) and a set of anchor points (A) that designate potential starting positions to which the constructs that are generated may be affixed.

Two schemas are considered: the (1) Deterministic Influence Right Turn (DIRT) schema and the (2) Deterministic Standard Deviation Reduction (DSDR) schema. Both computational schemas utilize a top-down approach for generating the constructs, in that they utilize the notion of a *wall*, which is a known construct for influencing autonomous agents. Figures 7.8a and 7.8b are pictorial representations of the DIRT and DSDR schemas, respectively. Additionally, the algorithmic description of the DIRT method can be seen in Algorithm 6. The algorithmic description of the DSDR method is similar to Algorithm 6, with the exception of two extra parameters, and thus not shown. For each of these methods, the parameters of construct angle and length are all independent of each other.

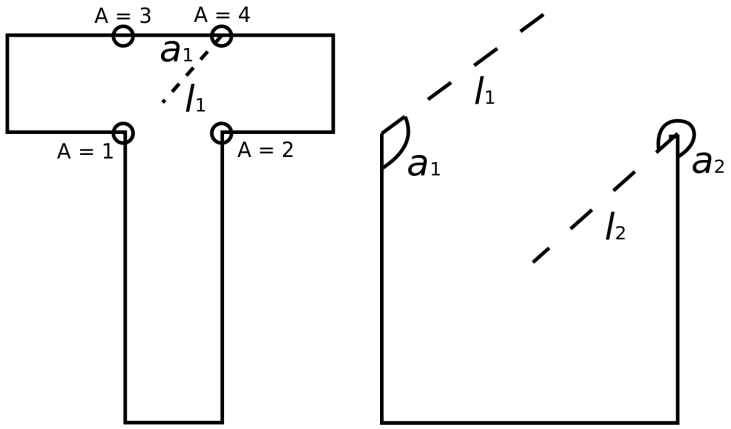


Figure 7.8: Figures 7.8a and 7.8b represent the two computational schemas. The workspace is the solid black line where the dotted line represents a possible construct. The labels a and l represent the various angle and length parameters.

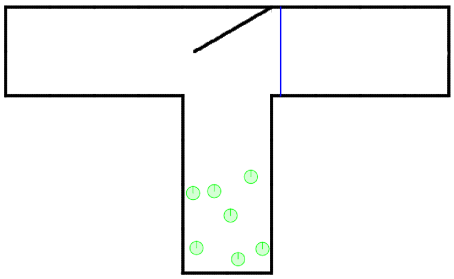


Figure 7.9: An example environment generated by the DIRT method from the fourth anchor position with a starting formation for seven agents. The thin-blue line represents the goal line for the turn right behavior.

7.4.2 Empirical Mapping Results

To gain an understanding of how perturbations in the environment effect the exhibited behavior of a group the parameter space for each of the presented schemas

Algorithm 6 Deterministic Influence Right Turn (DIRT)

Input: $theta \doteq$ *angle of the wall construct*
$length \doteq$ *length of the wall construct*
$A \doteq$ *anchor point for the wall construct*
$W \doteq$ *workspace (set of line segments)*
Output: An environment e that is comprised of W and the generated construct.
1: $point1 \leftarrow A$
2: $point2 \leftarrow A + [length * cos(theta),\ length * sin(theta)]$
3: $construct \leftarrow [point1, point2]$
4: $e \leftarrow [W, construct]$
5: return e

is empirically mapped. The parameter space is mapped by evaluating the environment's ability to successfully induce a group to perform a particular task. For all of the simulations in this work, the simulated agents are obeying the flocking model presented in Reynolds (1987); hereafter referred to as BOIDS.

This model was chosen because of its notoriety in the fields related to collective motion and for its emergent properties. The BOIDS flocking model only has three behaviors each agent follows (cohesion, separation, and velocity matching) but complex group level behaviors emerge from the member to member interactions, such as group decision making (Couzin et al., 2005).

A systematic grid-based search of the parameter space was used to pick parameter settings. The search employed dynamic resolution for each dimension similarly to the quad-tree methods for searching and representing regions in a space (Samet, 1980). Resolution is adjusted on basis of the success in achieving the task of the current parameter configuration. The algorithm efficiently searches regions in the parameter space with high densities of configurations that produce useful constructs. All of the parameter space maps adhere to the following symbols. A + designates parameter configurations that generate environments that successfully elicit the turn

right behavior from the group, where a • designates parameter configurations that generate environments that fail to elicit the turn right behavior from the group.

Parameter Space Maps (Turn Right Action): We mapped the parameter space for the DIRT schema where the desired action is to get at least 90% of the group to turn right at the junction in the t-junction workspace (Figure 7.9). The DIRT schema started with four anchor points for the t-junction workspace (the four corners at the junction). The first mapping iteration produced the map in Figure 7.10. The remaining mapping iterations are only applied to anchor point four, as it produced the highest number of successful environments in the first iteration. Figure 7.11 is the results of mapping the DIRT parameter states with six additional iterations of increasing resolution (around successful configurations) in both the angle and length parameters.

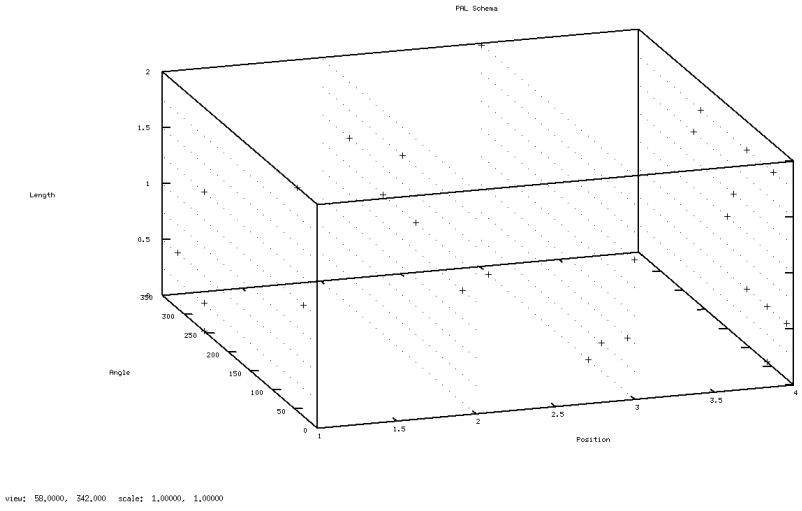


Figure 7.10: Parameter space map for the DIRT method on a group of seven agents obeying BOIDS from the four anchor locations.

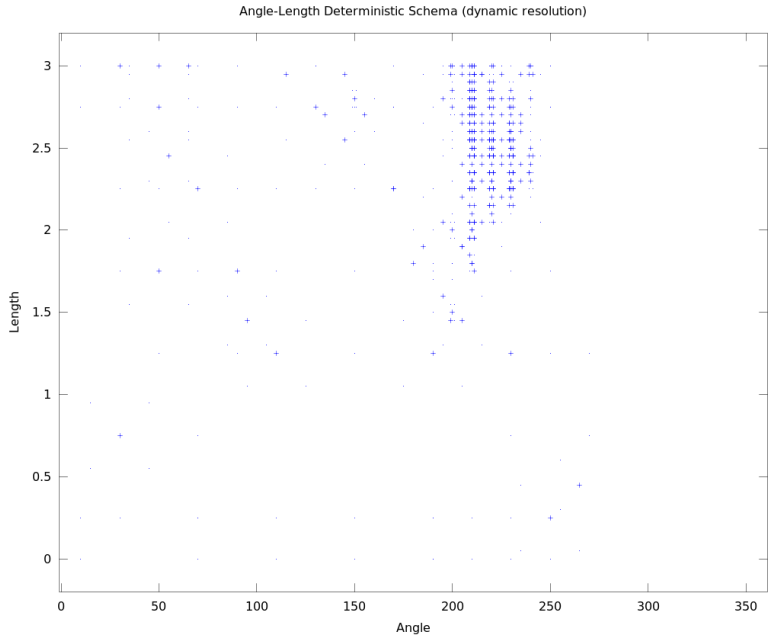


Figure 7.11: Parameter space map for the DIRT method on a group of seven agents obeying BOIDS from the fourth anchor position.

Figure 7.11 shows that the region in which the DIRT schema generates useful environments has a high density in respect to the rest of the parameter space. This suggests that the group which generated this map is fairly robust to variations in the environment as long as the constructs angle is between 200° and 250°; as well as, the length being between 2 and 3 meters. Additionally, the region in Figure 7.11 is relatively small when compared to the size of the set of possible configurations, which suggests that methods that employ random parameter selection may not be as efficient in searching this set of environments.

To examine the effect of group size on the region in the parameter space generated by the DIRT method, we map the same parameter space for a single agent and for a group of three agents; Figures 7.12 and 7.13 respectively. Notice that in both cases, the high density area seen in Figure 7.11 no longer exists. In fact, very few parameter

configurations were successful in generating environments that could reliably induce the right turn action from the agents. We conclude from these figures that the success of these environment depends on group level behaviors exhibited the agents (*e.g.*, cohesion). This is important because it allows one to then build connections between the structure of the parameter space and the macroscopic properties of the group.

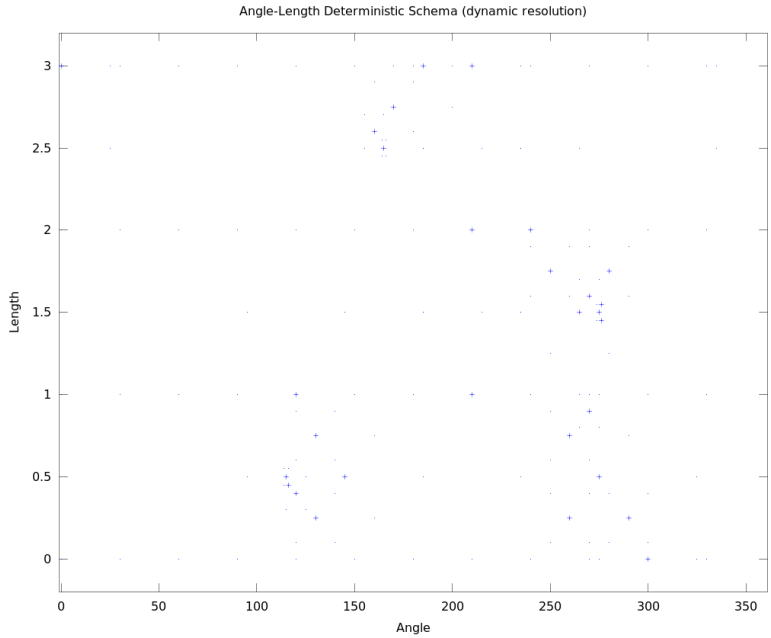


Figure 7.12: Parameter space map for the DIRT method on a single agent obeying BOIDS from the fourth anchor position.

Parameter Space Maps (Reduce x-axis Standard Deviation): In addition to the right turn behavior we explore a computational schema for reducing the group x-axis standard deviation. The motivating action behind this method is to coerce the group into a line (*i.e.*, reduce the x-axis standard deviation to 0), which has

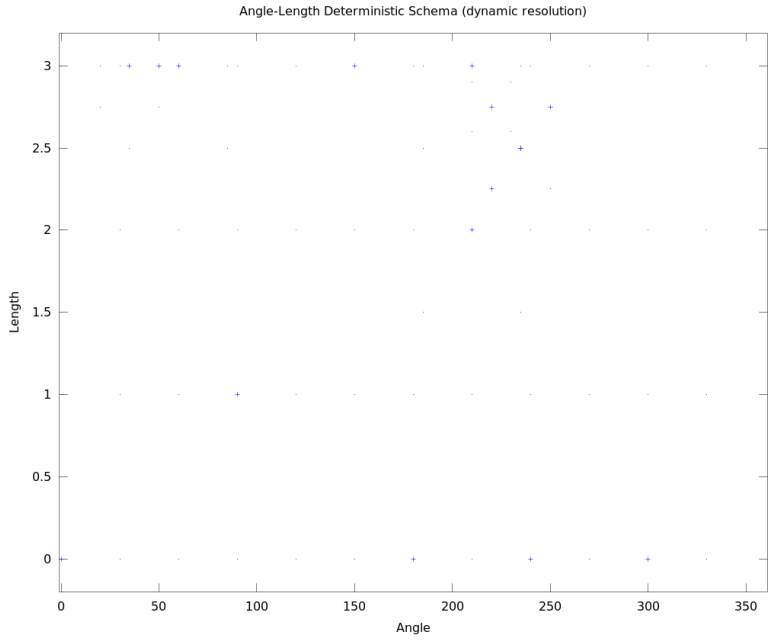


Figure 7.13: Parameter space map for the DIRT method on three agents obeying BOIDS from the forth anchor position.

motivation rooted in livestock management. Due to the four parameters used in the DSDR schema we plot all of the resulting deviations along the $y = 1$ axis in Figure 7.14.

Observing Figure 7.14 we see that the majority of the parameter configurations produce environments that induce a x-axis standard deviation in the group of in the range of 1.5 and 4.0. In regards to the desired single line action, there are only a few configurations that would generate desirable environments. This suggests that the generation method is either not capable of producing useful environments, or that some group level property (*e.g.*, repulsion) is impeding the environments' ability to manipulate the group.

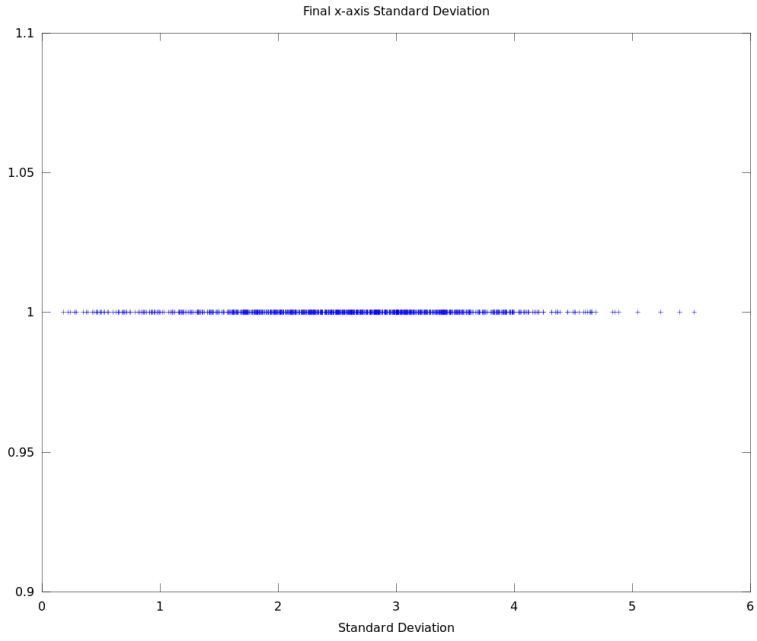


Figure 7.14: Represents the fitness values of 1,024 different parameter configurations from the four dimensional parameter space of the DSDR schema. All results are for a group of 10 agents obeying the BOIDS flocking model.

7.4.3 Understanding the Space of Constructs

One of the important limitations to note with the methodology introduced in Section 6 is in choosing a particular environmental schema. Since the schema is the aspect of the methodology that defines the set of possible environments it is important to understand the quality of that set. Here, quality refers to the ratio of valid to non-valid environments in the set generated from the given schema. By gaining a better understanding of the structure of the space in which these generated environments exist, better exploration methods for increasing the generalization and/or quality of the environmental schemas can be developed.

For example, if a schema with a large number of parameters where developed, parameter space would grow and the difficulty of mapping and searching it would

grow as well. One could use these high density regions within the parameter space to pick ranges of parameters which yield the largest set of valid environments. This could be done by employing various parameter optimization techniques.

8. DISCUSSION

8.1 Finding the Right Primitives

To this point, all the work done was under the assumption that the primitives used for environment generation are known. Although this is a reasonable assumption for many applications (*e.g.*, livestock management and traffic control), it still limits the generality of the presented approach for automatic generation of constructs for group control. In order to lift this assumption there are at least two plausible approaches: (1) environment decomposition and (2) bottom-up generation from a simple point primitive.

8.1.1 Environment Decomposition

The large majority of the group control work via adeptly constructed mechanisms and static constructs utilize experts and domain knowledge (*e.g.*, fishing and livestock management). However, as in livestock management, there still remains challenges reproducible results and robustness in regards to the variations in similar groups (*e.g.*, herds of tamed versus wild cattle). Although, ideal constructs for various groups and behaviors have not been discovered there are many current constructs that are useful and currently used for actual group control. Since there exist current constructs that elicit the desired behavior from the given group, it is reasonable to start with these constructs for finding useful primitives.

Decomposing known constructs into sets of primitives can be done using either directed or undirected methods. A **directed approach** would allow for an expert to identify regions of the known construct as potential primitives and the expected behavior of the group. These regions would be further reduced to a set of simple geometric representations (*e.g.*, line segments and arcs). At this point algorithms

could be design that would test variations of these geometries to determine an *optimal* primitive or a range of primitives that are useful in eliciting the expected behavior. It is important to note that the expected behavior in this case may not be the same as the behavior for the overall construct. For example, the overall desired behavior may be a trajectory through a workspace consisting of left and right turns (similar to the tasks in Section 7), where the expected behavior for a selected region may be a simple left turn.

An **undirected** approach would automatically generate the regions that were selected by the expert in the directed approach. This could be done in a variety of ways from segmentation algorithms to sliding windows along the construct. The difficulty comes when trying to determine the expected behavior of the group. Since this would be impossible, the candidate primitives would need to be clustered with ones that elicit similar behaviors. In the example above, primitives that elicit a turn right action would be clustered together. At this point, an *optimal* primitive or range of primitives could be generated for all of the behaviors seen during the decomposition.

One of the key benefits to an undirected approach is that it is a closed-loop system thus (potentially) allowing all possible primitives to be studied. Unfortunately, the generality of this approach also is a draw-back. For more complex constructs, it could be very computationally expensive to conduct this approach, with no runtime guarantees of finding useful primitives. On the other hand, a directed approach would limit the generality of the primitives found but may lead to more predictable running times.

8.1.2 De novo *Primitives*

Another approach to generating new primitives is starting with an empty slate. This could be done by generating random primitives from the combinations of simple point primitives. A bottom-up approach such as this makes no assumptions about the structure of the primitives; instead, primitives are generated by adding single pixel primitives to a empty workspace. The only input required from a user would be the desired or expect behavior of the group, such as turn right *. The bottom-up approach is similar to the work done with shape grammars, where the shape is a point and the rules define the way in which a point can be added. One difficulty with this approach is the dimensionality of the search space.

If no prior notion of what a useful primitive might look like is known, then the grammar employed would have to be general (*i.e.*, have a large number of rules that could be applied or a single rule that randomly places the next primitive). If the grammar is general enough to explore a large variation of primitives, then the feasibility in regards to running time comes into question.

One possible solution to this large search space draws motivation from Section 8. By using a shape grammar with a large number of rules, accompanied by probabilities, parameter optimization algorithms could be used to *focus* the generation of constructs within a closed-loop system. As a proof of concept, the Stochastic Influence Right Turn (SIRT) schema (Algorithm 7) was designed. For each iteration of the SIRT method (seen in Algorithm 7) a single rule is applied to the current primitive; the only addition to the standard shape grammar format is the rule probabilities.

As done in Section 8, the parameter space for the SIRT schema was dynamically mapped. Figure 8.1 shows the parameter space of the three rules in the SIRT schema.

*This could be excluded as done in the undirected approach if desired. The drawback would come in the computational resources required.

Algorithm 7 Stochastic Influence Right Turn (SIRT)

Input: $k \doteq$ *number of grammar iterations*
$x_{res} \doteq$ *rule resolution along x-coordinate axis*
$y_{res} \doteq$ *rule resolution along y-coordinate axis*
$P \doteq$ *Set of probabilities for each grammar rule*
$A \doteq$ *anchor point for the construct*
$e_b \doteq$ *workspace (set of line segments)*

Output: An environment e that is comprised of W and the generated construct.

1: $pointer = A$
2: **for** $i = 1; i < k; i++$ **do**
3: $rule = rand\_choice(P)$
4: **switch** $(rule)$
5: **case** 1**:**
6: $pointer \leftarrow pointer + [x_{res}, 0.0]$
7: **case** 2**:**
8: $pointer \leftarrow pointer + [0.0, y_{res}]$
9: **case** 3**:**
10: $pointer \leftarrow pointer + [x_{res}, y_{res}]$
11: **end switch**
12: $construct \leftarrow [construct, pointer]$
13: **end for**
14: $e \leftarrow [W, construct]$
15: return e

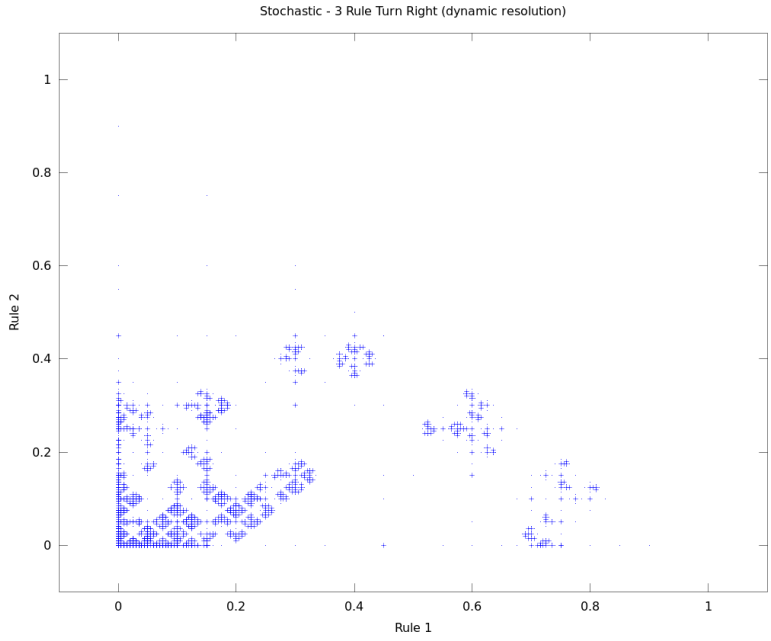


Figure 8.1: Parameter space map for the SIRT schema on a group of seven agents obeying BOIDS from the fourth anchor position. The probabilities of the three rules in the SIRT method are all dependent on each other; therefore the parameter space is projected on the rule 1 and rule 2 dimensions.

This space mapping reveals that the turn right behavior is exhibited by the group more often when the third rule has a high probability of use. Of course, based on the three rules this is expected, but it does support this approach working for larger rule sets.

8.1.3 Bottom-up versus Top-down

Two questions arise when considering a top-down or bottom-up approach:

(1) In general, is one approach better than the other (in regards to the quality of the generated constructs)?

(2) Is there any duality between the two approaches that we can exploit?

In regards to the first question, one way is by empirically computing the number of configurations in the schemas parameter space that produce useful environments for each of the methods. Clearly, the method that produces the highest number will be deemed better. However, we are only considering parameter space maps that were generated from a group of a single size. To expand the notion of *better* we would want to consider how the parameter space regions change in respect to group size. We would argue that if the regions persist through multiple group sizes, this would signify a better generation method.

Another, way of evaluating the approach would be in terms of generality of the environments produced. With top-down approaches the assumption is that the known primitives (*e.g*, walls) have a desirable/applicable influence on the group. However, these known primitives are not guaranteed to have the same effects on different groups or groups of varying size. As bottom-up approaches do not carry these assumptions, it is reasonable to conclude that the bottom up approach are more general, and in this context better than top-down approaches.

In regards to the second question, it is not possible to directly compare these approaches and the set of environments they produce with the results presented here. However, we hypothesize that it is possible to generate or learn the higher level (known) primitives from bottom up approaches. One possible way of constructing these higher level primitives is through a multi-step process of mapping the bottom-up approach for many simple group actions. With this data, we could then identify structures that successfully influence the group. The benefit to explore this possibility would be in produce more general environmental generations that could be applied to a wide variety of application domains and groups without prior domain knowledge.

8.2 Constructs for Multi-Robot Formations

Up to this point, only applications where either complete control over the environment or the environment was the medium that could be manipulated were considered. It is not always feasible or practical to attempt control of a group by means of the environment. As we see in the shepherding and caging literature (Lien et al., 2004, 2005; Pereira et al., 2004; Vaughan et al., 2000; Weiwei et al., 2012), it may be desirable to control or influence a group with a number of shepherds. One of the difficulties in this approach is determining or learning the formations the shepherds should form in order to effectively influence the target group.

Currently, many of the formations used for these applications are *ad hoc* or hand-tuned by experts. To address this, we suggest using environments as templates for potential robot formations. Figure 8.2 is an example of a valid environment for a given group and behavior being used as a formation template for a piloting application.

To apply our system to this application, we had to make two additions. First, we had to add the ability to convert the physical constructs generated by the system into robot formations and second, we needed to add in task allocation in order to determine which location in the formation each robot would occupy. Figure 8.3 is the diagrammatic overview of the system with the two additions. Note, that in Figure 8.3 the *Environment Generation* block contains both the environment filter and validation. Also, as before, we will assume the parameters are given, thus the *Parameter Generation* block will be *NULL*.

To show that our proposed approach would map to formation design, we conducted 80 robot trials using both static environments and six shepherding robots that use formations derived from the generated environments. We generated envi-

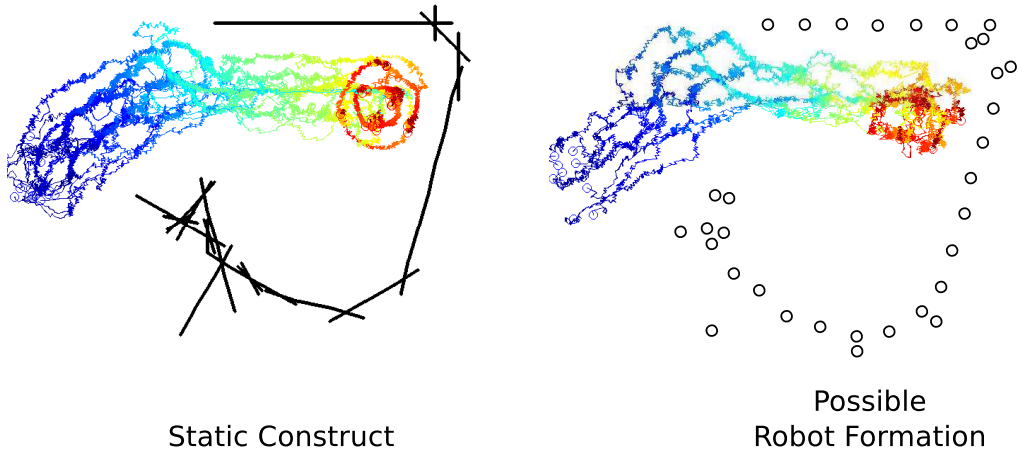


Figure 8.2: The left figure shows an environment that was generated via the presented system that is eliciting the corralling behavior from a group of agents. The right figure shows the same agents being controlled by a set of point (robots) that are in a formation based on the generated environment.

ronments and formations for four behaviors (tasks). Each task being to coerce the target group into following a particular trajectory through the given workspace. Pictorial representations of the four task are shown in Figure 8.4. For the remaining figures in this work, the green (light grey) gradient represents the goal location for the group, the blue (dark grey) gradient represents starting location of the group, and the dotted black line represents the desired group trajectory.

To generate candidate environments we implemented a simple computational schema that generates a single line environment at different angles given a anchor point in the workspace (Algorithm 8). Using our system, we generated valid environments for each of the four tasks.

The group of shepherding robots comprised of six mobile robots that are marked in a way that the target group robots sense them the same as they sensed the environment. Thus, the shepherd robots and the environment are treated the same

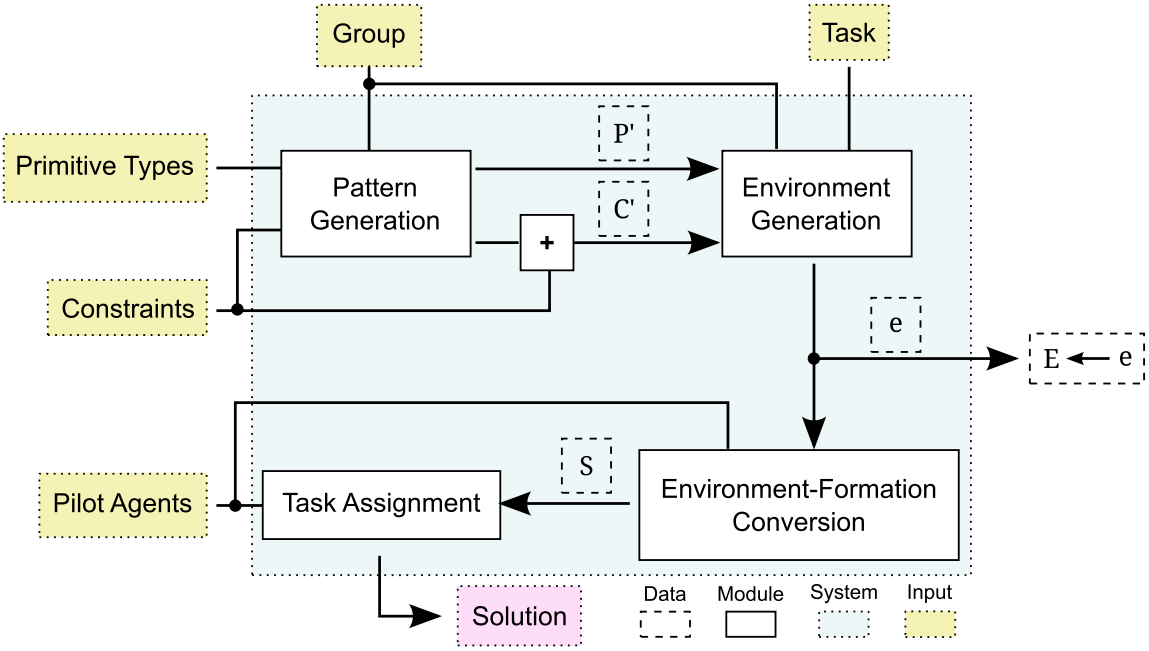


Figure 8.3: Additions to the presented system for use in formation design investigations and applications.

in regards to the motion model obeyed by the target group. Each construct (wall) generated by the system was approximated using three shepherd robots (a hand-tuned parameter).

After the walls where determined the system generated formations by assigning three robots equal distance along the wall. For *Task 1* and *4* the system generated solutions with three walls, for *Task 2* the solution had four walls, and for *Task 3* the solution had five walls.

It is important to note, that there was no solution found for *Task 3* when considering a completely static environment. To find a solution we had to allow for walls to be modified over time. Example solutions for all task can be seen in Figure 8.5.

After the templates where generated we needed to determine when and where each shepherding robot would be during the trial. To determine this we employed

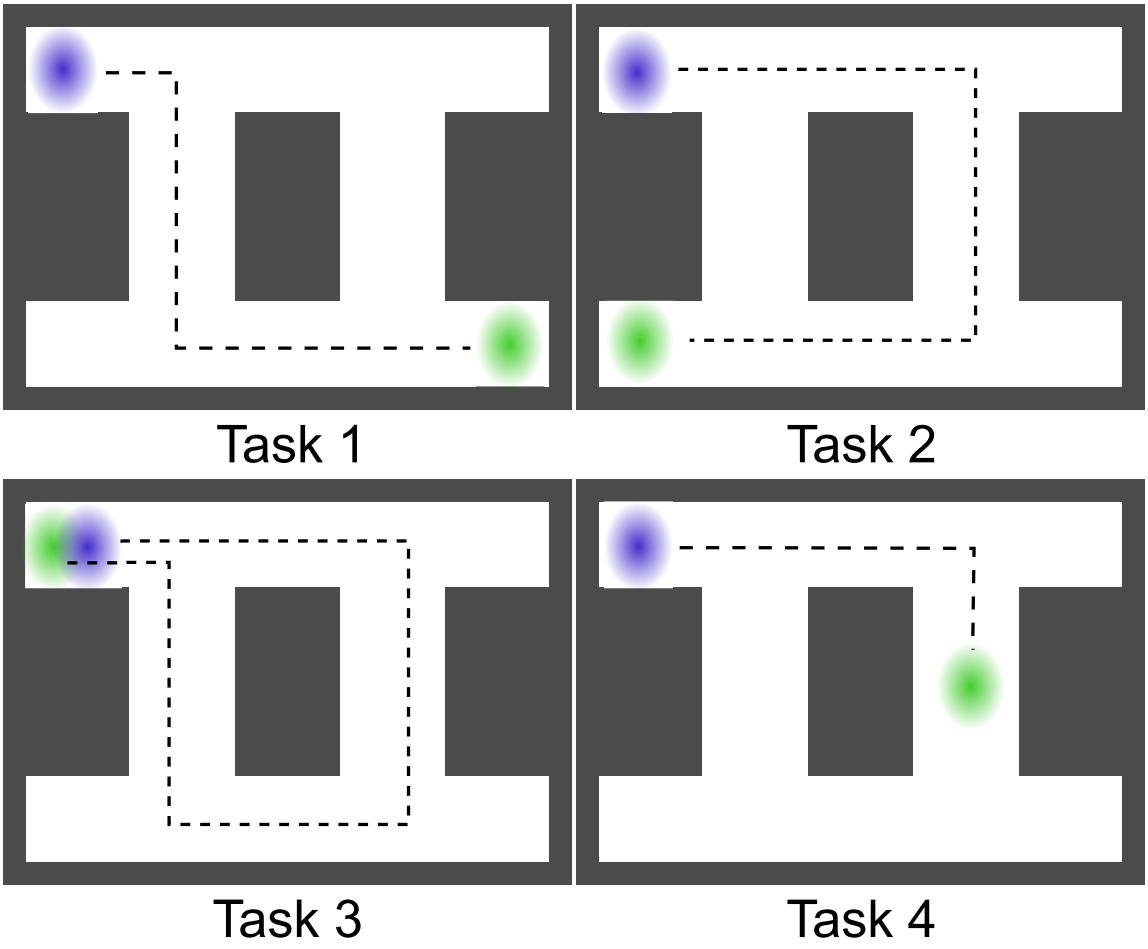


Figure 8.4: The four task used in the multi-robot formation trials.

the Hungarian method (Kuhn, 1955). After the locations were determined the user triggers the changes (during runtime) in formations when the target group cleared the current wall.

The results of the four task can be seen in Table 8.1. Each task for both static walls and mobile shepherds were conducted 10 times. Figures 8.7, 8.8, 8.9, and 8.10 are time series of selected trials.

Algorithm 8 Influence Turn Behavior (ITB)

Input: $angle, length, anchorPoint, W$
Output: An environment e that is comprised of W and the generated construct.

1: $point1 \leftarrow anchorPoint$
2: $point2 \leftarrow anchorPoint + [length * cos(theta),\ length * sin(theta)]$
3: $construct \leftarrow [point1, point2]$
4: $e \leftarrow [W, construct]$
5: return e

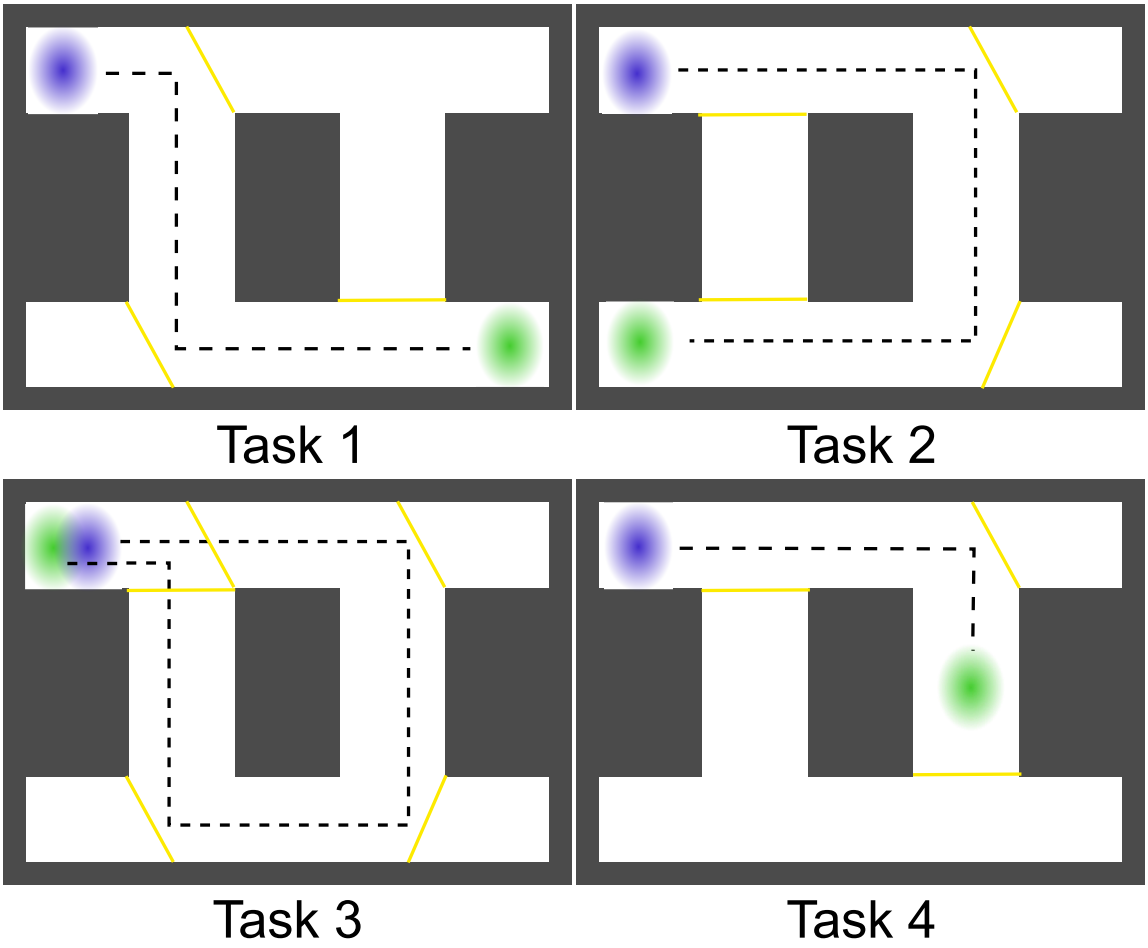


Figure 8.5: Example solutions generated by the presented system for all four task. The solid yellow lines represent the physical construct generated by the system.

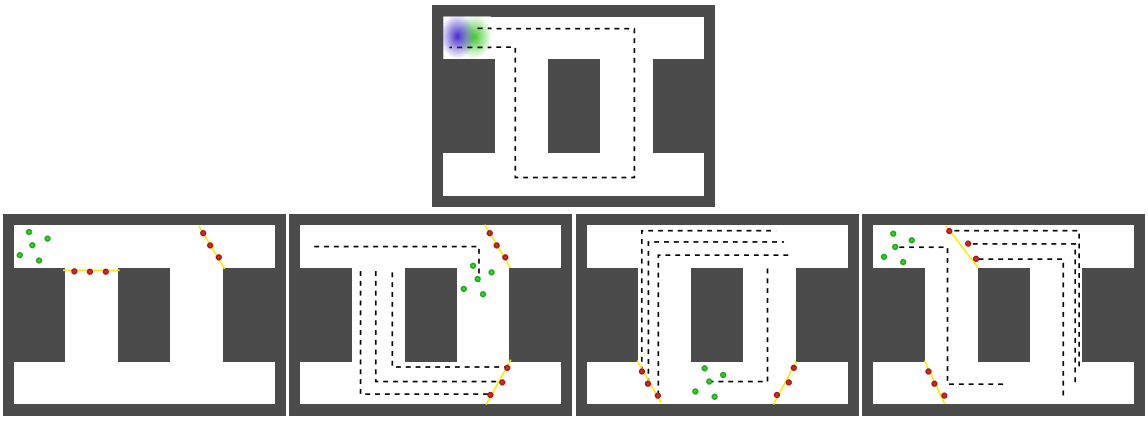


Figure 8.6: Pictorial representation of a single trial for *Task 3* for six shepherding robots (red circles) and a target group of five robots (green circles). The desired task is to guide the target group from the blue circle (dark grey) to the green circle (light grey) while following the given trajectory (black dotted line). Black lines represent the trajectories of the robots while the yellow solid lines represent the physical constructs the shepherding robots are using as formation templates.

Table 8.1: Average fitness values over the robot trials for all four task. Each trial had a group of four autonomous agents following the BOIDS motion model. For the case when the static primitives where approximated, six mobile agents where used. Each trial scenario was run a total of 10 times.

	Static Environment	Dynamic Agents
Task 1	0.975	0.875
Task 2	0.9	0.675
Task 3	0.925	0.775
Task 4	0.975	0.925

(a)

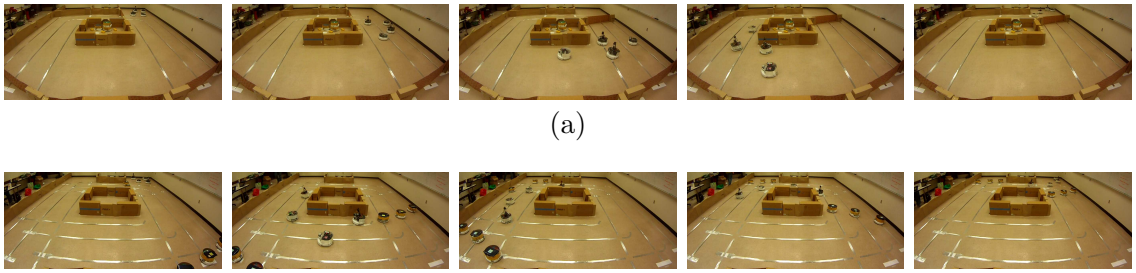
(b)

Figure 8.7: Time series of two successful multi-robot trials for *Task 1*. Time series (a) shows four robots following the BOIDS motion model eliciting *Task 1* in a static environment. Time series (b) shows four robots following the BOIDS motion model eliciting *Task 1* in a static environment with six mobile piloting robots.

(a)

(b)

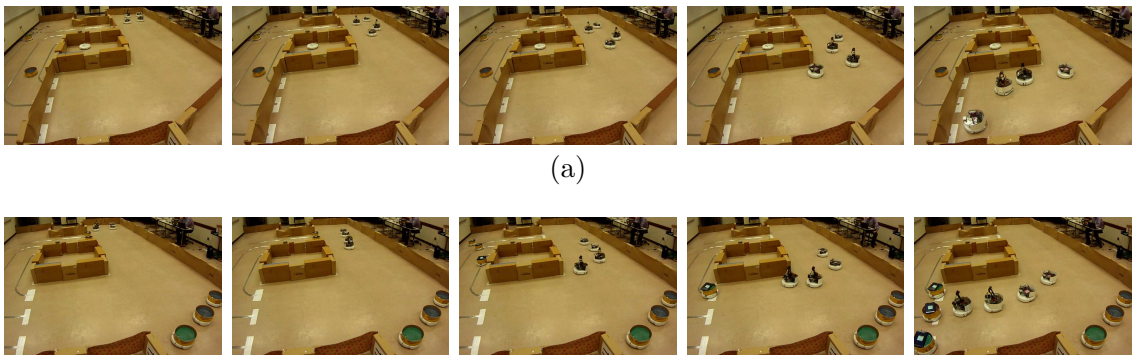
Figure 8.8: Time series of two successful multi-robot trials for *Task 2*. Time series (a) shows four robots following the BOIDS motion model eliciting *Task 2* in a static environment. Time series (b) shows four robots following the BOIDS motion model eliciting *Task 2* in a static environment with six mobile piloting robots.



(a)

(b)

Figure 8.9: Time series of two successful multi-robot trials for *Task 3*. Time series (a) shows four robots following the BOIDS motion model eliciting *Task 3* in a static environment. Time series (b) shows four robots following the BOIDS motion model eliciting *Task 3* in a static environment with six mobile piloting robots.



(a)

(b)

Figure 8.10: Time series of two successful multi-robot trials for *Task 4*. Time series (a) shows four robots following the BOIDS motion model eliciting *Task 4* in a static environment. Time series (b) shows four robots following the BOIDS motion model eliciting *Task 4* in a static environment with six mobile piloting robots.

9. CONCLUSIONS

The desire to understand and reliably control groups of autonomous individuals has been a focus of much research and interest throughout history. This dissertation has explored the current understanding and ability to control such groups in two ways: (1) a detailed meta-study of the current state of flocking model literature that focuses on microscopic flocking motion models, and (2) a methodology for exploiting the flocks group structure for reliable control. The presented meta-study revealed the need and importance for a more unified and rigorous approach to the study of the underlying individual behaviors, where the proposed methodology for group control has shown that rather simple and static environments can be automatically generated to reliably control various groups of gregarious individuals.

9.1 Microscopic Flocking Models

Using the three presented tools (data-flow template, two taxonomies, and a formalization/notation) the commonly seen designs and assumptions in the current motion models were identified and detailed. Through the use of the presented DT five critical aspects of the flocking problem are apparent. This dissertation has shown that failure to properly treat all of the five stages of the DT could lead to incompleteness and/or imprecision in the presentation of the motion model. To demonstrate this, examples were adhering to the DT, leads to a more complete and precise understanding of the model (Section 3.2.2) are presented. Additionally, through analysis of the selected publications using the two taxonomies (Tables 3.2 and 3.4), common assumptions made in the literature are identified, and this work shows that the majority of the investigations of microscopic motion models have the same underlying aim. Therefore, in an attempt to increase the breadth of flocking motion research,

the following recommendations for future research investigations.

9.1.1 Recommendations from Unification Tool-Set

The goal of the following recommendations are to help outline a framework/style for the presentation and design of future microscopic flocking motion models. All of the recommendations have resulted from applying the previously presented tools to the selected publications. Based on the study above, if these recommendations are followed for future publications, the overall understanding of the flocking problem could significantly be enhanced.

Recommendations from the Data-flow Template

1. The available and type of raw sensor information in the sensing stage affects all other aspects of the flocking motion model. Therefore, the sensing stage should explicitly list how all of the required information (*e.g.*, position, velocity, identification, etc.) is sensed from the group member's environment.

2. Observations from multiple studies (Vicsek et al., 1995) show that not all required information is used in the motion computation stage. For those works, it remains unclear if that information is a part of the motion model, or if that information is simulating a sensor or group member limitation. Therefore, the purpose and use of all required information should be clearly described.

3. Specific sensing attributes and limitations (*e.g.*, agent-based detection) may affect the exhibited flocking motions and/or the design of the motion model. Therefore, the sensing stage and the group member detection stage should present any and all assumptions made.

4. As seen in the presentation of the (Viscido et al., 2002) and (Reynolds, 1987) motion models, there is a possibility for multiple interpretations of a given

model. To help reduce ambiguities, algorithmic presentations of motion computation stages should be preferred over prose descriptions.

5. The vast majority of the flocking motion literature treats/presents multiple stages as one stage (typically the neighbor selection and motion computation stages), which could lead to incompleteness and/or imprecision. Therefore, it is recommended that all five DT stages should be logically separated and should also be treated/presented separately.

Recommendations from Design and Validation Taxonomies

1. Ambiguities and/or omissions of important information in the current motion models make it difficult to compare works across the literature. Therefore, future publication should explicitly state what attributes and assumptions the motion model requires and utilizes, respectively.

2. To better accommodate the implementation of flocking motion models on physical systems (robotic or biological), continuous time models should be preferred over discrete time models.

3. In some of the literature, the motion model was designed for local information, but when the model was validated the group members had access to global information. This inconsistency in the motion model validation could lead to models which are difficult to implement on a physical system. Therefore, the method in which the required information is sensed should reflect the type of information required by motion computation stage (*i.e.*, globally versus locally sensed).

4. To better simulate real-world situations, future motion models should be validated using asynchronous group members (when applicable).

5. The phenomenon of flocking is created through the interaction of many distributed individual group members. Therefore, motion models should only use locally sensed information when investigating the flocking phenomenon.

6. Due to flocks existing and operating in real-world environments, future investigations should validate motion models in obstacle filled environments (similar to environments biological flocks may encounter) or detail the assumptions that make this unnecessary or potentially detrimental to the model.

9.1.2 Directions for Future Flocking Model Investigations

1. The vast majority of the literature uses agent-based group member detection, therefore it is suggested that future robotic investigations explore other detection methods, such as sensor-based detection.

2. As mentioned in Section 3.3, there are other possible definitions of group composition; thus, future studies could consider the effects the various definitions have on the motion model and the exhibited motions of the group.

3. Future investigations could explore what flocking motions are afforded by using collisions as input to the motion computation stage.

4. The vast majority of the flocking motion models require position information from the group member's neighbors. Therefore, future investigations could explore the production of flocking motions without the use of position information.

5. It is reasonable to assume that the output of the motion computation stage will affect the input of the sensing stage. Future investigations could explore this connection in depth and study various aspects of a physical group, such as occlusions.

9.2 Group Control

From observations of groups of gregarious individuals, it is clear that the environment is a key determinant of the group's behavior. Recent works have explored methods for generating environments that can elicit a desired behavior from a given group but have failed to solve the general problem. This dissertation defines and formalized the problem of automatically generating such environments. Through a reduction of the Wharehouseman's problem, it is shown that this problem is PSPACE-Hard.

This dissertation presents a methodology and framework that can automatically generate solutions to this problem. Through computer simulations and multi-robot trials the presented approach and implemented system are validated. Furthermore, the results support that the approach is general enough to branch into other applications domains for group control; specifically shepherding applications. This is shown through the use of physical constructs generated by the system to serve as formation templates for shepherding robots.

9.2.1 Directions for Future Group Control Investigations

1. The vast majority of the control literature utilizes models of the individual group members. Future work should consider methods for modeling the interactions between a group and its environment at a macroscopic level. This could make way for many advances to other limitations to solving this and other closely related problems.

2. As mentioned in Section 8.1, little to no work has studied the basic *building-blocks* or primitives that are critical for group control. Investigations should explore methods for finding useful primitives as discussed in Section 8.1.

3. This work has discussed deriving useful primitives for group control but has

not covered the question of how these newly derived primitives are combined. Future work should consider ways of deriving schemas for the newly created primitives.

4. As seen in Section 7, the constructs generated from the proposed system are useful for the generation of robot formations. It may be desirable to merge this work with the internal group control of heterogeneous groups. Select members of a group can use knowledge of these constructs to help guide the other group members.

REFERENCES

Albi G, Pareschi L (2013) Modeling of self-organized systems interacting with a few individuals: From microscopic to macroscopic dynamics. Applied Mathematics Letters 26(4):397 – 401

Aoki I (1984) Internal Dynamics of Fish Schools in Relation To Inter-Fish Distance. Bulletin of the Japanese Society of Scientific Fisheries 48(3):1081–1088

Arkin RC, Balch T (1999) Behavior-based Formation Control for Multi-robot Teams. IEEE Transactions on Robotics and Automation 14(6):926–939

Babak P, Magnsson KG, Sigurdsson S (2004) Dynamics of group formation in collective motion of organisms. Mathematical Medicine and Biology 21(4):269–292

Ballerini M, Cabibbo N, Candelier R, Cavagna A, Cisbani E, Giardina I, Lecomte V, Orlandi A, Parisi G, Procaccini A, Viale M, Zdravkovic V (2008a) Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. Proceedings of the National Academy of Sciences 105(4):1232–1237

Ballerini M, Cabibbo N, Candelier R, Cavagna A, Cisbani E, Giardina I, Lecomte V, Orlandi A, Parisi G, Procaccini A, Viale M, Zdravkovic V (2008b) Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. Proceedings of the National Academy of Sciences 105(4):1232–1237

Barbosa A (1995) Foraging Strategies and Their Influence on Scanning and Flocking Behaviour of Waders. Journal of Avian Biology 26(3):182–186

Bazazi S, Buhl J, Hale JJ, Anstey ML, Sword GA, Simpson SJ, Couzin ID (2008) Collective Motion and Cannibalism in Locust Migratory Bands. Current Biology

18(10):735–739

Becker A, Habibi G, Werfel J, Rubenstein M, McLurkin J (2013) Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp 520–527

Bender JG, Fenton RE (1970) On the Flow Capacity of Automated Highways. Transportation Science 4(1):52–63

Berndt CD (1994) Using dynamic time warping to find patterns in time series. In: AAAI-94: Knowledge discovery in databases, pp 229–248

Blomqvist O, Bremberg S, Zauer R (2012) Mathematical modeling of flocking behavior. PhD thesis, KTH, Optimization and Systems Theory

Bobadilla L, Sanchez O, Czarnowski J (2011) Controlling wild bodies using discrete transition systems. Advanced Robots

Bobadilla L, Martinez F, Gobst E, Gossman K, LaValle SM (2012) Controlling wild mobile robots using virtual gates and discrete transitions. American Control Conference (Invited)

Bode NWF, Franks DW, Wood AJ (2011) Limited interactions in flocks: relating model simulations to empirical data. Journal of The Royal Society Interface 8(55):301–304

Bohringer KF, Bhatt V, Goldberg KY (1995) Sensorless manipulation using transverse vibrations of a plate. In: IEEE International Conference on Robotics and Automation, IEEE, vol 2, pp 1989–1996

Buhl J, Sumpter DJT, Couzin ID, Hale JJ, Despland E, Miller ER, Simpson SJ (2006) From Disorder to Order in Marching Locusts. Science 312(5778):1402–1406

Butler Z, Corke P, Peterson R, Rus D (2006) From robots to animals: Virtual fences for controlling cattle. International Journal of Robotics Research 25(5–6):485–508

Camperi M, Cavagna A, Giardina I, Parisi G, Silvestri E (2012) Spatially balanced topological interaction grants optimal cohesion in flocking models. Interface Focus 2(6):715–725

Cavagna A, Cimarelli A, Giardina I, Parisi G, Santagati R, Stefanini F, Tavarone R (2010) From empirical data to inter-individual interactions: Unveiling the rules of collective animal behavior. Mathematical Models and Methods in Applied Sciences 20:1491–1510

Cavagna A, Giardina I, Ginelli F (2012) Boundary information inflow enhances correlation in flocking. ArXiv e-prints

Chau H, Chen X, McKay A, de Pennington A (2004) Evaluation of a 3D Shape Grammar Implementation. In: Design Computing and Cognition, Klumer Academic Publishers, pp 357–376

Chomsky N (1956) Three models for the description of language. IRE Transactions on Information Theory 2(3):113–124

Clark PJ, Evans FC (1954) Distance to Nearest Neighbor as a Measure of Spatial Relationships in Populations. Ecology 35(4):445–453

Codling EA, Pitchford JW, Simpson SD (2007) Group navigation and the "many-wrongs principle" in models of animal movement. Ecology 88(7):1864–1870

Conradt L, Krause J, Couzin ID, Roper TJ (2009) 'Leading According to Need' in Self-Organizing Groups. The American Naturalist 173(3):304–312

Couzin ID, Krause J, Franks NR, Levin SA (2005) Effective leadership and decision-making in animal groups on the move. Nature 433(7025):513–516

Csahók Z, Vicsek T (1995) Lattice-gas model for collective biological motion. Physical Review E 52(5):5297–5303

Şamiloğlu AT, Gazi V, Koku AB (2006) Effects of asynchronism and neighborhood size on clustering in self-propelled particle systems. In: Computer and Information

Sciences ISCIS 2006, Lecture Notes in Computer Science, vol 4263, Springer Berlin / Heidelberg, pp 665–676

Czirók A, Vicsek T (2000) Collective behavior of interacting self-propelled particles. Physica A: Statistical Mechanics and its Applications 281(1–4):17–29

Czirók A, Stanley HE, Vicsek T (1997) Spontaneously ordered motion of self-propelled particles. Journal of Physics A: Mathematical and General 30(5):1375–1385

Desai JP, Ostrowski J, Kumar V (1998) Controlling formations of multiple mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'98), Leuven, Belgium, pp 2864–2869

Despland E, Collett M, Simpson SJ (2000) Small-scale processes in desert locust swarm formation: how vegetation patterns influence gregarization. Oikos 88(3):652–662

Diankov R, Srinivasa S, Kuffner DFJ (2008) Manipulation planning with caging grasps. In: Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on, IEEE, pp 285–292

Dingle H, Drake AV (2007) What Is Migration? BioScience 57(2):113–121

Dong JG (2012) Flocking under hierarchical leadership with a free-will leader. International Journal of Robust and Nonlinear Control

Edelstein-Keshet L (2001) Mathematical models of swarming and social aggregation. In: International Symposium on Nonlinear Theory and its Applications, Miyagi, Japan

Emlen, Jr JT (1952) Flocking Behavior in Birds. The Auk 69(2):160–170

Erdmann MA, Mason MT (1988) An exploration of sensorless manipulation. Robotics and Automation, IEEE Journal of 4(4):369–379

Erickson CL (2000) An artificial landscape-scale fishery in the bolivian amazon.

Nature 408(6809):190–193

Ferrante E, Turgut AE, Huepe C, Stranieri A, Pinciroli C, Dorigo M (2012) Self-organized flocking with a mobile robot swarm: a novel motion control method. Adaptive Behavior 20(6):460–477

Fine BT, Shell DA (2011) Flocking: don't need no stink'n robot recoginition. In: IEEE/RSJ International Conference on Robotics and Automation, San Francisco, CA, USA

Fine BT, Shell DA (2012) Examining the information requirements for flocking motion. In: 12th International Conference on Adaptive Behavior, Odense, Denmark

Fink J, Hsieh MA, Kumar V (2008) Multi-robot manipulation via caging in environments with obstacles. In: Robotics and Automation, IEEE International Conference on, IEEE, pp 1471–1476

Fukui R, Mori T, Sato T (2010) Application of caging manipulation and compliant mechanism for a container case hand-over task. In: Robotics and Automation, 2010 IEEE International Conference on, pp 4511–4518

Funge J, Tu X, Terzopoulos D (1999) Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '99, pp 29–38

Gazi V, Passino KM (2003) Stability Analysis of Swarms. IEEE Transactions on Automatic Control 48(4):692–697

Gazi V, Passino KM (2005) Stability of a one-dimensional discrete-time asynchronous swarm. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 35(4):834–841

Giardina I (2008) Collective behavior in animal groups: Theoretical models and empirical studies. HFSP Journal 2(4):205–219

Ginelli F, Chaté H (2010) Relevance of metric-free interactions in flocking phenomena. Phys Rev Lett 105:168,103

Gips J (1974) Shape grammars and their uses. PhD thesis, Stanford University Palo Alto, CA

Gökçe F, Şahin E (2009) To flock or not to flock: the pros and cons of flocking in long-range "migration" of mobile robot swarms. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '09), Budapest, Hungary, pp 65–72

Goldstone RL, Janssen MA (2005) Computational models of collective behavior. Trends in Cognitive Sciences 9(9):424 – 430

Grandin T (1980) Observations of cattle behavior applied to the design of cattle-handling facilities. Applied Animal Ethology 6(1):19–31

Grégoire G, Chaté H, Tuj Y (2003) Moving and staying together without a leader. Physica D: Nonlinear Phenomena 181(30-4):157–170

Gueron S, Levin SA, Rubenstein DI (1996) The Dynamics of Herds: From Individuals to Aggregations. Journal of Theoretical Biology 182(1):85–98

Halatsch J, Kunze A, Schmitt G (2008) Using shape grammars for master planning. In: Design Computing and Cognition'08, Springer, pp 655–673

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

Hamilton WD (1971) Geometry for the Selfish Herd. Journal of Theoretical Biology 31(2):295–311

Hauert S, Leven S, Varga M, Ruini F, Cangelosi A, Zufferey JC, Floreano D (2011) Reynolds flocking in reality with fixed-wing robots: communication range vs. max-

imum turning rate. In: IEEE/RSJ International Conference on Robotics and Automation, San Francisco, CA, USA

Helbing D, Molnár P (1995) Social force model for pedestrian dynamics. Phys Rev E 51:4282–4286

Helbing D, Farkas I, Vicsek T (2000) Simulating dynamical features of escape panic. Nature 407:487–490

Helbing D, Buzna L, Johansson A, Werner T (2005) Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. Transportation Science 39(1):1–24

Hildenbrandt H, Carere C, Hemelrijk C (2010) Self-organized aerial displays of thousands of starlings: a model. Behavioral Ecology 21(6):1349–1359

Hopcroft J, Schwartz J, Sharir M (1984) On the complexity of motion planning for multiple independent objects; pspace-hardness of the "warehouseman's problem". The International Journal of Robotics Research 3(4):76–88

Hsiao SW, Chen CH (1997) A semantic and shape grammar based approach for product design. Design studies 18(3):275–296

Huth A, Wissel C (1992) The simulation of the movement of fish schools. Journal of Theoretical Biology 156(3):365–385

Hutto RL (1988) Foraging Behavior Patterns Suggest a Possible Cost Associated with Participation in Mixed-Species Bird Flocks. Oikos 51(1):79–83

Ip GW, Chiu Cy, Wan C (2006) Birds of a feather and birds flocking together: Physical versus behavioral cues may lead to trait- versus goal-based group perception. Journal of personality and social psychology 90(3):368–368–381

Jackson DE, Ratnieks FL (2006) Communication in ants. Current Biology 16(15):R570 – R574

Jadbabaie A, Lin J, Morse AS (2002) Coordination of Groups of Mobile Autonomous

Agents Using Nearest Neighbor Rules. IEEE Transactions on Automatic Control 48(6):988–1001

James R, Bennett PG, Krause J (2004) Geometry for mutualistic and selfish herds: the limited domain of danger. Journal of Theoretical Biology 228(1):107–113

Jowers I, Earl C (2010) The Construction of Curved Shapes. Environment and Planning B: Planning and Design 37(1):42–58

Kelly ID, Keating DA (1996) On Flocking By The Fusion Of Sonar And Active Infrared Sensors. In: Proceedings of the Conference on Mechatronics and Machine Vision in Practice, Guimarães, Portugal, vol 1, pp 14–17

King AJ, Cowlishaw G (2009) Leaders, followers, and group decision-making. Communicative & Integrative Biology 2(2):147–150, DOI 10.4161/cib.7562

Kline C (1996) C++ boids. http://www.behaviorworks.com/people/ckline/ cornell-www/boid/boids.html, Last viewed in March 2012

Koutsourakis P, Simon L, Teboul O, Tziritas G, Paragios N (2009) Single view reconstruction using shape grammars for urban environments. In: Computer Vision, IEEE 12th International Conference on, IEEE, pp 1795–1802

Kozlov VV, Mitrofanova MY (2003) Galton board. Regular and Chaotic Dynamics 8(4):431–439

Kuhn HW (1955) The hungarian method for the assignment problem. Naval research logistics quarterly 2(1-2):83–97

Leighton T, Richards S, White P (2004) Trapped within a wall of sound. Acoustics Bulletin 29:24–29

Lerman K, Martinoli A, Galstyan A (2005) A review of probabilistic macroscopic models for swarm robotic systems. In: Sahin E, Spears W (eds) Swarm Robotics, Lecture Notes in Computer Science, vol 3342, Springer Berlin / Heidelberg, pp 143–152

Levine H, Rappel WJ, Cohen I (2000) Self-organization in systems of self-propelled particles. Physical Review E 63(1):017,101–017,104

Lien JM, Bayazit O, Sowell R, Rodriguez S, Amato NM (2004) Shepherding behaviors. In: Proc. of the IEEE International Conference on Robotics and Automation, vol 4, pp 4159 – 4164

Lien JM, Rodriguez S, Malric JPJ, Amato NM (2005) Shepherding behaviors with multiple shepherds. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, Barcelona, Spain, pp 3413–3418

Lindhé M, Ögren P, Johansson KH (2005) Flocking with Obstacle Avoidance: A New Distributed Coordination Algorithm Based on Voronoi Partitions. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'05), Barcelona, Spain, pp 1785–1790

Lopez U, Gautrais J, Couzin ID, Theraulaz G (2012) From behavioural analyses to models of collective motion in fish schools. Interface Focus

Lukeman R, Li YX, Edelstein-Keshet L (2010) Inferring individual rules from collective behavior. Proceedings of the National Academy of Sciences 107(28):12,576–12,580

Matarić MJ (1993) Designing Emergent Behaviors: From Local Interactions to Collective Intelligence. In: Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB '93), Honolulu, Hawai, USA, pp 432–441

Mikhailov AS, Zanette DH (1999) Noise-induced breakdown of coherent collective motion in swarms. Physical Review E 60(4):4571–4575

Miki T, Nakamura T (2006) An effective simple shepherding algorithm suitable for implementation to a multi-mmobile robot system. In: Innovative Computing, Information and Control, 2006. ICICIC '06. First International Conference on, vol 3, pp 161 –165

Mogilner A, Edelstein-Keshet L (1999) A Non-Local Model for a Swarm. Journal of Mathematical Biology 38:534–570

Moglich M, Maschwitz U, Holldobler B (1974) Tandem calling: A new kind of signal in ant communication. Science 186(4168):1046–1047

Moussaïd M, Helbing D, Simon Garnier MC Anders Johansson, Theraulaz G (2009) Experimental study of the Behavioural mechanisms underlying self-organization in human crowds. Proceedings of the Royal Society B 276:2755–2762

Musser DR, Stepanov AA (1988) Generic Programming. International Symposium on Symbolic and Algebraic Computation (ISSAC) pp 13–25

Niizato T, Gunji YP (2011) Metrictopological interaction model of collective behavior. Ecological Modelling 222(17):3041 – 3049

Oberkampf WL, Trucano TG (2000) Validation methodology in computational fluid dynamics. AIAA paper 2549:19–22

Okubo A (1986) Dynamical aspects of animal grouping: Swarms, schools, flocks, and herds. Advances in Biophysics 22:1–94

Olfati-Saber R (2006) Flocking for Multi-agent Dynamic Systems: Algorithms and Theory. IEEE Transactions on Automatic Control 51(3):401–420

Orsborn S, Cagan J, Pawlicki R, Smith R (2006) Creating cross-over vehicles: Defining and combining vehicle classes using shape grammars. AIE EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing 20(03):217–246

Parrish JK (1989) Re-examining the selfish herd: are central fish safer? Animal Behaviour 38(6):1048–1053

Parrish JK, Edelstein-Keshet L (1999) Complexity, Pattern, and Evolutionary Trade-Offs in Animal Aggregation. Science 284(5411):99–101

Parrish JK, Viscido SV, Grunbaum D (2002) Self-Organized Fish Schools: An Examination of Emergent Properties. Biological Bulletin 202(3):296–305

Partridge BL (1982) The Structure and Function of Fish Schools. Scientific American 246(6):114–123

Pereira GA, Campos MF, Kumar V (2004) Decentralized algorithms for multi-robot manipulation via caging. The International Journal of Robotics Research 23(7-8):783–795

Petersen JB, Robinson BS, Belknap DF, Stark J, Kaplan LK (1994) An archaic and woodland period fish weir complex in central maine. Archaeology of Eastern North America pp 197–222

Pierson A, Schwager M (2015) Bio-inspired non-cooperative multi-robot herding. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol 4

Pitcher TJ, Partridge BL, Wardle CS (1976) A blind fish can school. Science 194(4268):963–965

Rands SA, Pettifor RA, Rowcliffe JM, Cowlishaw G (2004) State-dependent foraging rules for social animals in selfish herds. Proceedings of the Royal Society of London Series B: Biological Sciences 271(1557):2613–2620

Rauch EM, Millonas MM, Chialvo DR (1995) Pattern formation and functionality in swarm models. Physics Letters A 207(3–4):185–193

Reynolds CW (1987) Flocks, herds and schools: A distributed behavioral model. Computer Graphics 21(4)

Reynolds CW (2004) Opensteer: Steering behaviors for autonomous characters. http://opensteer.sourceforge.net/, Last viewed in March 2012

Rodriguez A, Mason M, Ferry S (2012a) From caging to grasping. The International Journal of Robotics Research p 0278364912442972

Rodriguez S, Giese A, Amato N, Zarrinmehr S, Al-Douri F, Clayton M (2012b) Environmental effect on egress simulation. In: Kallmann M, Bekris K (eds) Motion in

Games, Lecture Notes in Computer Science, vol 7660, Springer Berlin Heidelberg, pp 7–18

Samet H (1980) Region representation: Quadtrees from boundary codes. Communications ACM 23(3):163–170

Shimoyama N, Sugawara K, Mizuguchi T, Hayakawa Y, Sano M (1996) Collective Motion in a System of Motile Elements. Physical Review Letters 76(20):3870–3873

Simons AM (2004) Many wrongs: the advantage of group navigation. Trends in Ecology & Evolution 19(9):453–455

Smith J, Martin A (2009) Comparison of Hard-Core and Soft-Core Potentials for Modelling Flocking in Free Space. ArXiv e-prints

Stiny G (1980) Introduction to shape and shape grammars. Environment and Planning B: Planning and Design 7(3):343–351

Sugawara K (2012) Personal communications

Szabó P, Nagy M, Vicsek T (2008) Turning with the others: novel transitions in an SPP model with coupling of accelerations. In: Proceedings of the IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO '08), Venice, Italy, pp 463–464

Szabó P, Nagy M, Vicsek T (2009) Transitions in a self-propelled-particles model with coupling of accelerations. Physical Review E 79(2):0219,080–021,913

Tanner HG, Jadbabaie A, Pappas GJ (2003a) Stable Flocking of Mobile Agents, Part I: Fixed Topology. In: Proceedings of the IEEE Conference on Decision and Control, pp 2010–2015

Tanner HG, Jadbabaie A, Pappas GJ (2003b) Stable Flocking of Mobile Agents, Part II: Dynamic Topology. In: Proceedings of the IEEE Conference on Decision and Control, pp 2016–2021

Toner J, Tu Y (1998) Flocks, herds, and schools: A quantitative theory of flocking.

Physical Review E 58(4):4828–4858

Treščák T (2012) The shape grammar interpreter source forge repository. Sourceforge.net/projects/sginterpreter

Treščák T, Rodriguez I, Esteva M (2009) General Shape Grammar Interpreter for Intelligent Designs Generations. In: The 6th Int. Conf. on Computer Graphics, Imaging and Visulization

Turgut A, Çelikkanat H, Gökçe F, Şahin E (2008) Self-Organized flocking in mobile robot swarms. Swarm Intelligence 2(2–4):97–120

Umstatter C (2011) The evolution of virtual fences: A review. Computers and Electronics in Agriculture 75(1):10–22

Vaughan R, Sumpter N, Henderson J, Frost A, Cameron S (2000) Experiments in Automatic Flock Control. Robotics and Autonomous Systems 31:109–117

Vicsek T, Zafeiris A (2012) Collective motion. Physics Reports 517(34):71 – 140

Vicsek T, Czirók A, Ben-Jacob E, Cohen I, Shochet O (1995) Novel Type of Phase Transition in a System of Self-Driven Particles. Physical Review Letters 75(6):1226–1229

Viscido SV, Wethey DS (2002) Quantitative analysis of fiddler crab flock movement: evidence for 'selfish herd' behaviour. Animal Behaviour 63(4):735–741

Viscido SV, Miller M, Wethey DS (2002) The Dilemma of the Selfish Herd: The Search for a Realistic Movement Rule. Jour of Theor Biology 217(2):183–194

Wang Z, Kumar V (2002) Object closure and manipulation by multiple cooperating mobile robots. In: Robotics and Automation, IEEE International Conference on, IEEE, vol 1, pp 394–399

Wang Z, Hirata Y, Kosuge K (2004) Control a rigid caging formation for cooperative object transportation by multiple mobile robots. In: Robotics and Automation, IEEE International Conference on, IEEE, vol 2, pp 1580–1585

Warburton K, Lazarus J (1991) Tendency-distance models of social cohesion in animal groups. Journal of Theoretical Biology 150(4):473–488

Weiwei W, Fukui R, Shimosaka M, Sato T, Kuniyoshi Y (2012) Cooperative manipulation with least number of robots via robust caging. In: Advanced Intelligent Mechatronics, 2012 IEEE/ASME International Conference on, pp 896–903

Whitfield DP (2003) Redshank Tringa totanus flocking behaviour, distance from cover and vulnerability to sparrowhawk Accipiter nisus predation. Journal of Avian Biology 34(7):163–169

Wood AJ, Ackland GJ (2007) Evolving the selfish herd: emergence of distinct aggregating strategies in an individual-based model. Proceedings of the Royal Society of London Series B: Biological Sciences 274(1618):1637–1642

Yeh H, Curtis S, Patil S, van den Berg J, Manocha D, Lin M (2008) Composite agents. In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, pp 39–47

Zheng X, Zhong T, Liu M (2009) Modeling crowd evacuation of a building based on seven methodological approaches. Building and Environment 44(3):437–445

APPENDIX A

MICROSCOPIC FLOCKING MODEL SPECIFICATIONS

Table A.1: The translation of the neighbor selection and motion computation stages from the selected flocking models. Together, these two stages constitute the control-law, which is the primary focus of the vast majority of the literature. The motion rules presented in this table have been translated into the common notation (Table 1) presented earlier in this study. The neighbor selection column details the required perception functions (Table 3.1) along with what set(s) of neighbors will be considered. The motion computation column details the low-level control law, or algorithm, which computes the next motion of the flock member.

Paper	Neighbor Selection	Motion Computation
Viscido et al. (2002) (SNN)	$k = \text{Nearest}_i(L^2)$	$\mathbf{r}_i(t+\Delta t) = \begin{cases} \mathbf{r}_i(t) + \hat{\mathbf{d}}_{ik}(t) & \text{if } L^2(\mathbf{r}_i(t), \mathbf{r}_k(t)) > r_\rho, \\ \mathbf{r}_i(t) & \text{otherwise.} \end{cases}$
Viscido et al. (2002) (HA)	$k = \text{Nearest}_i(L^2)$	$\mathbf{r}_i(t+\Delta t) = \begin{cases} \mathbf{r}_i(t) + \hat{\mathbf{e}}(\mathbf{t}) & \text{if } L^2(\mathbf{r}_i(t), \mathbf{r}_k(t)) > r_\rho, \\ \perp (\mathbf{r}_i(t) + \hat{\mathbf{e}}(\mathbf{t})) & \text{otherwise.} \end{cases}$

Continued on next page

Table A.1

Paper	Neighbor Selection	Motion Computation
	l = Nearest$_k(L^2)$	$\mathbf{e}(t) = (\hat{\mathbf{d}}_{lk}(t) + \mathbf{r}_l(t)) - \mathbf{r}_i(t)$
Viscido et al. (2002) (LCH)*		$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \hat{\mathbf{e}}$
	$\mathbf{I}_i$ = All()	$\mathbf{e} = \sum_{j \in \mathbf{I}_i} \frac{\mathbf{d}_{ij}}{1 + w_1 L^2(\mathbf{r}_i(t),\ \mathbf{r}_j(t))}$
		Where w_1 has units of $\frac{1}{\text{distance}}$
Conradt et al. (2009)	$\mathbf{I}_{i\alpha}$ = DistanceBased$_i(L^2,\ \mathbf{R}_{[0,\ \alpha]})$	$\mathbf{d}_i(t) = \begin{cases} -\frac{1}{\lvert\mathbf{I}_{i\alpha}\rvert} \sum_{j \in \mathbf{I}_{i\alpha}} \frac{\mathbf{r}_j(t) - \mathbf{r}_i(t)}{L^2(\mathbf{r}_i(t), \mathbf{r}_j(t))} & \text{if } \lvert\mathbf{I}_{i\alpha}\rvert \neq 0, \\ = \frac{1}{4\lvert\mathbf{I}_{i\beta}\rvert} \sum_{j \in \mathbf{I}_{i\beta}} \frac{\mathbf{r}_j(t) - \mathbf{r}_i(t)}{L^2(\mathbf{r}_j(t), \mathbf{r}_i(t))} + \frac{1}{4\lvert\mathbf{I}_{i\beta}\rvert} \sum_{j \in \mathbf{I}_{i\beta}} \hat{\mathbf{v}}_j(t) + w_1^i \frac{\mathbf{r}_*(t) - \mathbf{r}_i(t)}{2L^2(\mathbf{r}_*(t), \mathbf{r}_i(t))} & \text{if } \lvert\mathbf{I}_{i\beta}\rvert \neq 0, \\ \frac{\mathbf{r}_*(t) - \mathbf{r}_i(t)}{\lvert\mathbf{r}_*(t) - \mathbf{r}_i(t)\rvert} & \text{otherwise.} \end{cases}$
	$\mathbf{I}_{i\beta}$ = DistanceBased$_i(L^2,\ \mathbf{R}_{[\alpha,\ \beta]})$	

Continued on next page

*This is the formalization of the LCH motion rule that was validated in Viscido et al. (2002). The other version of the motion rule is discussed in more detail through out this meta-study (see Section 3.3.3).

Table A.1

Paper	Neighbor Selection	Motion Computation
Gueron et al. (1996)	$\mathbf{I}_S = \text{BoundingBox}_i((\mathbf{r}_i x - w_1, \mathbf{r}_i x + w_1), (\mathbf{r}_i y - w_2, \mathbf{r}_i y + w_2))$ $\mathbf{F} = \text{Angle}_i() \cap \mathbf{I}_S$ $\mathbf{LF} = \text{Angle}_i() \cap \mathbf{I}_S$ $\mathbf{RF} = \text{Angle}_i() \cap \mathbf{I}_S$ $\mathbf{LB} = \text{Angle}_i() \cap \mathbf{I}_S$ $\mathbf{RB} = \text{Angle}_i() \cap \mathbf{I}_S$	$\theta_i(t + \Delta t) = \begin{cases} \theta_i(t) & \text{if } (\mathbf{F} \wedge \neg\mathbf{LF} \wedge \neg\mathbf{RF} \wedge \neg\mathbf{LB} \wedge \neg\mathbf{RB}) \vee \\ & (\neg\mathbf{F} \wedge \mathbf{LF} \wedge \mathbf{RF} \wedge \neg\mathbf{LB} \wedge \neg\mathbf{RB}) \vee \\ & (\neg\mathbf{F} \wedge \neg\mathbf{LF} \wedge \neg\mathbf{RF} \wedge \mathbf{LB} \wedge \mathbf{RB}), \\ \theta_i(t) - 90^\circ & \text{if } (\neg\mathbf{F} \wedge \mathbf{LF} \wedge \neg\mathbf{RF} \wedge \neg\mathbf{LB} \wedge \neg\mathbf{RB}) \vee \\ & (\neg\mathbf{F} \wedge \neg\mathbf{LF} \wedge \neg\mathbf{RF} \wedge \mathbf{LB} \wedge \neg\mathbf{RB}), \\ \theta_i(t) + 90^\circ & \text{if } (\neg\mathbf{F} \wedge \neg\mathbf{LF} \wedge \mathbf{RF} \wedge \neg\mathbf{LB} \wedge \neg\mathbf{RB}) \vee \\ & (\neg\mathbf{F} \wedge \neg\mathbf{LF} \wedge \neg\mathbf{RF} \wedge \neg\mathbf{LB} \wedge \mathbf{RB}). \end{cases}$ $u_i(t + \Delta t) = \begin{cases} u_i(t)\frac{1}{w_i} & \text{if } (\mathbf{F} \wedge \neg\mathbf{LF} \wedge \neg\mathbf{RF} \wedge \neg\mathbf{LB} \wedge \neg\mathbf{RB}), \\ u_i(t) & \text{if } (\neg\mathbf{F} \wedge \mathbf{LF} \wedge \neg\mathbf{RF} \wedge \neg\mathbf{LB} \wedge \neg\mathbf{RB}) \vee \\ & (\neg\mathbf{F} \wedge \neg\mathbf{LF} \wedge \neg\mathbf{RF} \wedge \mathbf{LB} \wedge \neg\mathbf{RB}) \vee \\ & (\neg\mathbf{F} \wedge \neg\mathbf{LF} \wedge \mathbf{RF} \wedge \neg\mathbf{LB} \wedge \neg\mathbf{RB}) \vee \\ & (\neg\mathbf{F} \wedge \neg\mathbf{LF} \wedge \neg\mathbf{RF} \wedge \neg\mathbf{LB} \wedge \mathbf{RB}), \\ 0 & \text{if } (\neg\mathbf{F} \wedge \mathbf{LF} \wedge \mathbf{RF} \wedge \neg\mathbf{LB} \wedge \neg\mathbf{RB}), \\ u_i(t)w_i & \text{if } (\neg\mathbf{F} \wedge \neg\mathbf{LF} \wedge \neg\mathbf{RF} \wedge \mathbf{LB} \wedge \mathbf{RB}). \end{cases}$ See Section 3.4.1

Continued on next page

Table A.1

Paper	Neighbor Selection	Motion Computation
Couzin et al. (2005)	$\mathbf{I} = \text{DistanceBased}_i(L^2, \mathbf{R}_{[0,\beta]})$	$\mathbf{d}_i(t+\Delta t) = \frac{\hat{\mathbf{e}}+w_1\mathbf{d}_{i*}(t)}{\lvert\hat{\mathbf{e}}+w_1\mathbf{d}_{i*}(t)\rvert}$ $\mathbf{e} = \sum_{j\in\mathbf{I}} \frac{\mathbf{r}_j(t)-\mathbf{r}_i(t)}{\lvert\mathbf{r}_j(t)-\mathbf{r}_i(t)\rvert} + \sum_{j\in\mathbf{I}} \frac{\mathbf{v}_j(t)}{\lvert\mathbf{v}_j(t)\rvert}$
Lopez et al. (2012)	$\mathbf{I} = \text{DistanceBased}_i(L^2, \mathbf{R}_{[0,\beta]})$	$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + w_1\Delta t * \theta_i(t+\Delta t)$ $\theta_i(t+\Delta t) = \frac{1}{\lvert\mathbf{I}\rvert}\sum_{j\in\mathbf{I}} w_2(\lvert r_j(t) - r_i(t)\rvert)\theta_j(t) + \frac{1}{\lvert\mathbf{I}\rvert}\sum_{j\in\mathbf{I}} w_3(\lvert\mathbf{r}_j(t) - \mathbf{r}_i(t)\rvert)\frac{\mathbf{r}_j(t)-\mathbf{r}_j(t)}{\lvert\mathbf{r}_j(t)-\mathbf{r}_j(t)\rvert} + \eta_i(t)$ Where $\eta_i(t) \doteq$ a stochastic component
Hamilton (1971) (1D)	$\mathbf{I} = \text{VoronoiBased}_i()$ $k = \text{VoronoiBased}_{\mathbf{I}_1}()\backslash i$ $l = \text{VoronoiBased}_{\mathbf{I}_2}()\backslash i$	$\mathbf{r}_i(t+\Delta t)\begin{cases}\mathbf{r}_i(t) & \text{if } L^2(\mathbf{r}_{\mathbf{I}_0}(t),\mathbf{r}_{\mathbf{I}_1}(t)) < min(L^2(\mathbf{r}_l(t),\mathbf{r}_i(t)), L^2(\mathbf{r}_k(t),\mathbf{r}_i(t))),\\ \frac{\mathbf{r}_l(t)-\mathbf{r}_i(t)}{2} & \text{if } L^2(\mathbf{r}_l(t),\mathbf{r}_i(t)) < L^2(\mathbf{r}_k(t),\mathbf{r}_i(t)),\\ \frac{\mathbf{r}_k(t)-\mathbf{r}_i(t)}{2} & \text{otherwise.}\end{cases}$

Continued on next page

Table A.1

Paper	Neighbor Selection	Motion Computation
Hamilton (1971) (2D)	$k = \text{Nearest}_i(L^2, \mathbf{R}_{[0,\ \beta]})$	$\mathbf{r}_i(t+\Delta t) = \begin{cases} \mathbf{r}_i(t) + \hat{\mathbf{d}}_{ik}(t) & \text{if } L^2(\mathbf{r}_i(t), \mathbf{r}_k(t)) > r_\rho, \\ \mathbf{r}_i(t) & \text{otherwise.} \end{cases}$
Czirók et al. (1997); Smith and Martin (2009); Vicsek et al. (1995)[†]	$\mathbf{I} = \text{DistanceBased}_i(L^2, \mathbf{R}_{[0,\ \beta]})$	$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\Delta t$ $\theta_i(t+\Delta t) = \sum_{j\in\mathbf{I}} \arctan\left(\frac{\sin\theta_j(t)}{\cos\theta_j(t)}\right) + \Delta\theta$ Where $\Delta\theta \in \mathcal{U}(-\frac{\eta}{2},\ \frac{\eta}{2})$
Dong (2012)	$\mathbf{I} = \text{DistanceBased}_i(L^2, \mathbf{R}_{[0,\ \beta]})$	$\mathbf{r}_i(t\ +\ \Delta t)\ =\ \mathbf{r}_i(t)\ +\ \mathbf{v}_i(t)\Delta t$ $\mathbf{v}_i(t\ +\ \Delta t)\ =\ \sum_{j\in\mathbf{I}} \Delta t a_{ij}(L^2(\mathbf{r}_i(t),\ \mathbf{r}_j(t)))(\mathbf{v}_j(t)\ -\ \mathbf{v}_i(t))\ +\ \mathbf{v}_i(t)$ $a_{ij}(x)\ =\ \begin{cases} 1 & \text{if } x\ \leq\ w_1, \\ 0 & \text{otherwise.} \end{cases}$

Continued on next page

[†]Uses a common absolute velocity.

Table A.1

Paper	Neighbor Selection	Motion Computation
		Where $w_1 > 0$
Shimoyama et al. (1996)		$\dot{\mathbf{v}}_i(t) = \frac{1}{m}(-w_1\mathbf{v}_i(t) + w_2\mathbf{n}_i(t) + \sum_{j\in\mathbf{I}} \alpha_{ij}(t)\mathbf{f}_{ij}(t) + \mathbf{g}_i(t))$
(Also based on Sugawara (2012))	$\mathbf{I} = \text{DistanceBased}_i(L^2, \mathbf{R}_{[0,\ \beta]})$	$\dot{\mathbf{n}}_i(t) = \frac{1}{\tau}(\mathbf{n}_i(t) \times \hat{\mathbf{v}}_i(t) \times \mathbf{n}_i(t))$
		$\alpha_{ij}(t) = 1 + d\left[\mathbf{n}_i(t) \cdot \frac{\mathbf{r}_j(t)-\mathbf{r}_i(t)}{\lvert\mathbf{r}_j(t)-\mathbf{r}_i(t)\rvert}\right]$
		$\mathbf{f}_{ij}(t) = -w_3\left[\left(\frac{\lvert\mathbf{r}_j(t)-\mathbf{r}_i(t)\rvert}{w_4}\right)^{-3} - \left(\frac{\lvert\mathbf{r}_j(t)-\mathbf{r}_i(t)\rvert}{w_4}\right)^{-2}\right] \times \left(\frac{\mathbf{r}_j(t)-\mathbf{r}_i(t)}{w_4}\right) \exp\left(\frac{-\lvert\mathbf{r}_j(t)-\mathbf{r}_i(t)\rvert}{w_4}\right)$
		$\mathbf{g}_i(t) = w_5\left(\frac{\mathbf{e}_i(t)-\mathbf{r}_i(t)}{\lvert\mathbf{I}\rvert\lvert\mathbf{e}_i(t)-\mathbf{r}_i(t)\rvert}\right)$
		$\mathbf{e}_i(t) = \frac{\sum_{j\in\mathbf{I}} \mathbf{r}_j(t)}{\lvert\mathbf{I}\rvert}$
		Where $m \doteq$ agent's mass, $\tau \doteq$ rotational relaxation time, and $(0 \leq d \leq 1)$ for $\alpha_{ij}(t)$.

Continued on next page

Table A.1

Paper	Neighbor Selection	Motion Computation
Szabó et al. (2008, 2009)		$\mathbf{e}_i(t+\Delta t) = \nu \cdot \mathcal{M}(\gamma,\xi) \cdot N(s \cdot \frac{\sum_{j\in\mathbf{I}} \mathbf{v}_j(t)}{\lvert\mathbf{I}\rvert} + (1-s) \cdot\ a(t)\Delta\ t)$
	$\mathbf{I}\ = \text{DistanceBased}_i(L^2, \mathbf{R}_{[0,\ \beta]})$	$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{e}_i(t)\Delta t$
		Where $\nu \doteq \lvert\mathbf{v}\rvert$, $\mathcal{M}(e,\ \xi) \doteq$ rational tensor representing random perturbation with $\gamma :=$ random unit vector chosen uniformly vectors $\perp N(\frac{\sum_{j\in\mathbf{I}} v_j(t)}{\lvert I\rvert})$ and $\xi \in \mathcal{U}(-\eta\pi,\eta\pi), N(e) = \frac{e}{\lvert e\rvert}, s \in (0,1], a(t) = \frac{\mathbf{v}(t)-\mathbf{v}(t-\Delta t)}{\Delta t}$
Levine et al. (2000)		$\hat{\mathbf{v}}_i(t) = \frac{1}{m}\left(w_1\hat{\mathbf{f}}_i(t) - w_2\mathbf{v}_i(t) - \nabla U_i(t)\right)$
	$\mathbf{I}\ = \text{DistanceBased}_i(L^2, \mathbf{R}_{[0,\ \beta]})$	$\hat{\mathbf{f}}_i(t) = \sum_{j\in\mathbf{I}} \hat{\mathbf{v}}_j(t) \exp\left(-\frac{\lvert\mathbf{r}_i(t) - \mathbf{r}_j(t)\rvert}{w_3}\right)$
		$U_i(t) = \sum_{j\in\mathbf{I}} w_4 \exp\left(-\frac{\lvert\mathbf{r}_i(t) - \mathbf{r}_j(t)\rvert}{w_5}\right) - \sum_{j\in\mathbf{I}} w_6 \exp\left(-\frac{\lvert\mathbf{r}_i(t) - \mathbf{r}_j(t)\rvert}{w_7}\right)$
Toner and Tu (1998)		$\theta_i(t+\Delta t) = \sum_{j\in\mathbf{I}} \theta_j(t) + \eta_i(t)$
	$\mathbf{I}\ = \text{DistanceBased}_i(L^2, \mathbf{R}_{[0,\ \beta]})$	$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{e}$ where $\mathbf{e} = [\cos\theta_i(t+\Delta t), \sin\theta_i(t+\Delta t)]$

Continued on next page

Table A.1

Paper	Neighbor Selection	Motion Computation
Grégoire et al. (2003)	$\mathbf{I} = \text{DistanceBased}_i(L^2, \mathbf{R}_{[0,\ \beta]})$	**Model 1:** $\theta_i(t+\Delta t) = \arg\left[w_1 \sum_{j\in\mathbf{I}} \mathbf{v}_j(t) + w_2 \sum_{j\in\mathbf{I}} \mathbf{f}_{ij}\right] + \eta\xi_i(t)$ **Model 2:** $\theta_i(t+\Delta t) = \arg\left[w_1 \sum_{j\in\mathbf{I}} \mathbf{v}_j(t) + w_2 \sum_{j\in\mathbf{I}} \mathbf{f}_{ij} + |\mathbf{I}|\eta\mathbf{e}_i(t)\right]$ $\mathbf{f}_{ij} = \frac{\mathbf{r}_j(t) - \mathbf{r}_i(t)}{L^2(\mathbf{r}_j(t) - \mathbf{r}_i(t))} \begin{cases} -\infty & \text{if } L^2(\mathbf{r}_j(t) - \mathbf{r}_i(t) < w_3), \\ \frac{1}{4}\frac{L^2(\mathbf{r}_j(t) - \mathbf{R}_i(t)) - w_4}{w_5 - w_4} & \text{if } w_3 < L^2(\mathbf{r}_j(t) - \mathbf{r}_i(t)) < w_5, \\ 1 & \text{otherwise.} \end{cases}$
Camperi et al. (2012)	$\mathbf{I} = \text{DistanceBased}_i(L^2, \mathbf{R}_{[0,\ \beta]})$	$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_0\hat{\mathbf{e}}$ $\mathbf{e} = w_1 \sum_{j\in\mathbf{I}} \mathbf{v}_j(t) + w_2 \sum_{j\in\mathbf{I}} \mathbf{f}_{ij} + |\mathbf{I}|\eta\mathbf{e}_i(t)$ $\mathbf{f}_{ij} = \frac{\mathbf{r}_j(t) - \mathbf{r}_i(t)}{L^2(\mathbf{r}_j(t) - \mathbf{r}_i(t))} \begin{cases} -\infty & \text{if } L^2(\mathbf{r}_j(t) - \mathbf{r}_i(t) < w_3), \\ \frac{1}{4}\frac{L^2(\mathbf{r}_j(t) - \mathbf{R}_i(t)) - w_4}{w_5 - w_4} & \text{if } w_3 < L^2(\mathbf{r}_j(t) - \mathbf{r}_i(t)) < w_5, \\ 1 & \text{otherwise.} \end{cases}$ $\mathbf{v}_0 \doteq$ constant speed of member
Helbing et al. (2005)	See Section 3.4.1	

Continued on next page

Table A.1

Paper	Neighbor Selection	Motion Computation
Matarić (1993)	See Section 3.4.1	
Reynolds (1987)	See Section 3.4.1	
Kelly and Keating (1996)	See Section 3.4.1	
Turgut et al. (2008)	$\mathbf{I} = \text{DistanceBased}(L^2, \mathbf{R}_{[0,\ \beta]})$	$\mathbf{d}_i(t+\Delta t) = \hat{\mathbf{e}}$ $\mathbf{e} = \sum_{j \in \mathbf{I}(t)} e^{\tilde{\theta}_j(t)} + w_1 \frac{1}{8} \sum_{k=1}^{8} f_k e^{\frac{4}{\pi}k}$ $f_k = \begin{cases} -\frac{(L^2(\mathbf{r}_i(t) - \mathbf{r}_{O_k}(t)) - r_\rho)^2}{w_2} & \text{if } L^2(\mathbf{r}_i(t) - \mathbf{r}_{O_k}(t)) \geq r_\rho, \\ \frac{(L^2(\mathbf{r}_i(t) - \mathbf{r}_{O_k}(t)) - r_\rho)^2}{w_2} & \text{otherwise.} \end{cases}$ Where $\tilde{\theta}_j(t) = \angle\left(e^{\left(\theta_i(t) - \theta_j(t) + \frac{\pi}{2}\right)} + w_3 e^{\eta}\right)$, and $\eta = \mathcal{N}\left(\sum_{j \in \mathbf{I}_i(t)} \left(\theta_i(t) - \theta_j(t) + \frac{\pi}{2}\right), \sigma\right)$

Continued on next page

Table A.1

Paper	Neighbor Selection	Motion Computation
Gökçe and Şahin (2009)	$\mathbf{I} = \text{DistanceBased}(L^2, \mathbf{R}_{[0,\ \beta]})$	$\mathbf{d}_i(t+\Delta t) = \hat{\mathbf{e}}$ $\mathbf{e} = \sum_{j \in \mathbf{I}(t)} e^{\tilde{\theta}_j(t)} + w_1 \frac{1}{8} \sum_{k=1}^{8} f_k e^{\frac{4}{\pi}k} + w_2(\mathbf{d}_{i*}(t) - \mathbf{d}_i(t))$ $f_k = \begin{cases} -\frac{(L^2(\mathbf{r}_i(t) - \mathbf{r}_{O_k}(t)) - r_\rho)^2}{w_3} & \text{if } L^2(\mathbf{r}_i(t) - \mathbf{r}_{O_k}(t)) \geq r_\rho, \\ \frac{(L^2(\mathbf{r}_i(t) - \mathbf{r}_{O_k}(t)) - r_\rho)^2}{w_3} & \text{otherwise.} \end{cases}$ Where $\tilde{\theta}_j(t) = \theta_i(t) - \theta_j(t) + \frac{\pi}{2}$ and $\mathbf{r}_{O_k}(t) \doteq$ the pose of the k^{th} obstacle.
Tanner et al. (2003a)	$\mathbf{I} = \text{FixedSet}\, i()$	$\dot{\mathbf{r}}_\mathbf{i} = \mathbf{V}_i(t)$ $\dot{\mathbf{V}}_\mathbf{i} = \mathbf{u}_i(t)$ $\mathbf{u}_i(t) = \sum_{j \in \mathbf{I}} (\mathbf{v}_i(t) - \mathbf{v}_j(t)) - \sum_{j \in \mathbf{I}} \nabla_{\mathbf{r}_i(t)} U_{ij}$ $U_{ij} = \begin{cases} i \frac{1}{(L^2)^2 + \log(L^2)^2} & L^2 < r_\rho \\ w_1 & L^2 \geq r_\rho \end{cases}$

Continued on next page

Table A.1

Paper	Neighbor Selection	Motion Computation
Tanner et al. (2003b)		$\dot{\mathbf{r}}_\mathbf{i} = \mathbf{V}_i(t)$
	$\mathbf{I} = \text{DistanceBased}(L^2, \mathbf{R}_{[0,\ \beta]})$	$\dot{\mathbf{V}}_\mathbf{i} = \mathbf{u}_i(t)$
		$\mathbf{u}_i(t) = \sum_{j \in \mathbf{I}} (\mathbf{v}_i(t) - \mathbf{v}_j(t)) - \sum_{j \in \mathbf{I}} \nabla_{\mathbf{r}_i(t)} U_{ij}$
		$U_{ij} = \begin{cases} i \frac{1}{(L^2)^2 + \log{(L^2)^2}} & L^2 < r_\rho \\ w_1 & L^2 \geq r_\rho \end{cases}$
Jadbabaie et al. (2002) (Leaderless)	$\mathbf{I} = \text{DistanceBased}(L^2, \mathbf{R}_{[0,\ \beta]})$	$\theta_i(t + \Delta t) = \frac{1}{w_1}(\lvert\mathbf{I}\rvert\theta_i(t) + \sum_{j \in \mathbf{I}} \theta_j(t))$
		Where $w_1 > \lvert\mathbf{I}\rvert$
Jadbabaie et al. (2002) (Leader-Follower)	$\mathbf{I}_f = \text{DistanceBased}(L^2, \mathbf{R}_{[0,\ \beta]})$	$\theta_i(t + \Delta t) = \frac{1}{1 + \lvert\mathbf{I}_f \cap \mathbf{I}_l\rvert}(\theta_i(t) + \sum_{j \in \mathbf{I}_f \cap \mathbf{I}_l} \theta_j(t))$
	$\mathbf{I}_l = \text{FixedSet}()$	

Continued on next page

Table A.1

Paper	Neighbor Selection	Motion Computation
Gazi and Passino (2003)	$\mathbf{I} = \text{All}()$	$\mathbf{r}_i(t+\Delta t) = \sum_{j\in\mathbf{I}} f(L^2(\mathbf{r}_i(t),\mathbf{r}_j(t)))$ where $f(x) = -x(w_1 - w_2 exp(\frac{\sqrt{x^\top x}^2}{w_3}))$ with $w_2 > w_1$
Gazi and Passino (2005)	$\mathbf{I} = \text{VoronoiBased}_i()$ $k = \text{VoronoiBased}_{\mathbf{I}_1}()\backslash i$ $l = \text{VoronoiBased}_{\mathbf{I}_2}()\backslash i$	$x_i(t+\Delta t) = \begin{cases} x_i(0) & \text{if } i = 1, \\ \max\left(\begin{array}{l} x_{i-1}(t) + w_1, \\ \min\left(\begin{array}{l} x_i(t) - g\left(x_i(t) - \frac{x_{i-1}(\tau_{i-1}) + x_{i+1}(\tau_{i+1})}{2}\right), \\ x_{i-1}(t) - w_1 \end{array}\right) \end{array}\right) & \text{if } i \neq 1, N, \\ \max\left(\begin{array}{l} x_{N-1}(t) + w_1, \\ x_N(t) - g(x_N(t) - x_{N-1}(\tau_{N-1}) - r) \end{array}\right) & \text{if } i = N. \end{cases}$ Where $g(\cdot) \doteq$ a scaling function and $\tau_i \doteq$ the time when x_i was last sensed.
Olfati-Saber (2006)	$\mathbf{I} = \text{DistanceBased}(L^2, \mathbf{R}_{[0,\ \beta]})$	$\mathbf{v}_i(t+\Delta t) = \sum_{j\in\mathbf{I}} \phi_\alpha(\|\|\mathbf{r}_j(t) - \mathbf{r}_i(t)\|\|_\sigma)\mathbf{n}_{ij} + \sum_{j\in\mathbf{I}} a_{ij}(\mathbf{v}_j(t) - \mathbf{v}_i(t)) + f_i^\gamma$ $\phi_\alpha(x) = ph\left(\frac{x}{\|\|r\|\|_\sigma}\right)\frac{1}{2}\left[(w_1 + w_2)\left(\frac{x + w_3}{\sqrt{1 + (x + w_3)^2}}\right) + (w_1 - w_2)\right]$

Continued on next page

Table A.1

Paper	Neighbor Selection	Motion Computation
		$\|\|x\|\|_\sigma = \frac{1}{\epsilon}\left[\sqrt{1 + \epsilon\|\|x\|\|^2} - 1\right]$
		$\mathbf{n}_{ij} = \frac{\mathbf{r}_j(t) - \mathbf{r}_i(t)}{\sqrt{1 + \epsilon\|\|\mathbf{r}_j(t) - \mathbf{r}_i(t)\|\|^2}}$
		$a_{ij} = ph\left(\frac{\|\|\mathbf{r}_j(t) - \mathbf{r}_i(t)\|\|_\sigma}{\|\|r\|\|_\sigma}\right)\ in\ [0,\ 1]$
		$ph(x) = \begin{cases} 1 & x \in [0,\ h) \\ \frac{1}{2}\left[1 + \cos\left(\pi\frac{x-h}{1-h}\right)\right] & x \in [h,\ 1] \\ 0 & \text{otherwise.} \end{cases}$
		$f_i^\gamma = -w_4\left(\mathbf{r}_i(t) - \mathbf{r}_*(t)\right) - w_5\left(\mathbf{v}_i(t) - \mathbf{v}_*(t)\right)$
		Where $w_4,\ w_5 > 0,\ h \in (0,\ 1)$
Arkin and Balch (1999)	See Section 3.4.1	
Hauert et al. (2011)‡		$\theta_i(t + \Delta t) = \theta_i(t) - w_1\hat{\mathbf{s}}_i(t) + w_2\hat{\mathbf{a}}_i(t) + w_3\hat{\mathbf{c}}_i(t) + w_4\mathbf{d}_{i*}(t)$
	$\mathbf{I} = \text{DistanceBased}(L^2, \mathbf{R}_{[0,\ \beta]})$	

Continued on next page

‡Uses a fixed velocity.

Table A.1

Paper	Neighbor Selection	Motion Computation
		$\mathbf{s}_i(t) = \frac{\sum_{j \in \mathbf{I}} \frac{1}{\mathbf{r}_i(t) - \mathbf{r}_j(t)}}{\lvert\mathbf{I}\rvert}$
		$\mathbf{a}_i(t) = \frac{\sum_{j \in \mathbf{I}} \mathbf{v}_j(t)}{\lvert\mathbf{I}\rvert}$
		$\mathbf{c}_i(t) = \frac{\sum_{j \in \mathbf{I}} \mathbf{r}_j(t)}{\lvert\mathbf{I}\rvert}$

APPENDIX B

ALGORITHM PROOFS

Theorem. *The GCCG algorithm is resolution complete.*

Proof. The GCCG method generates all possible environments given the resolution and bound constraints given by C_g and $\mathcal{P}$. Each $e \in E_{cand}$ is validated to either a *yes* or *no* solution, thus the GCCG algorithm is resolution complete. □

Theorem. *The complexity of the GCCG algorithm is exponential.*

Proof. Given there are k primitives chosen from $\mathcal{P}$ when have $k\varphi$ possible combinations. Each primitive has α^{γ} permutations given by

$$P_i = \prod_{j=0}^{\gamma_i} \frac{\alpha_{max}^j - \alpha_{min}^j}{\mathcal{R}_{ij}} = \mathcal{O}(\alpha_i^{\gamma_i}) = \mathcal{O}(\alpha^{\gamma})$$

where,

$$\alpha_i = \max_{j=0 \to \gamma_i} \frac{\alpha_{max}^j - \alpha_{min}^j}{\mathcal{R}_{ij}},$$

$$\alpha = \max_{i=0 \to \varphi} \alpha_i,$$

$$\gamma = \max_{i=0 \to \varphi} \gamma_i.$$

Thus, resulting in a runtime of $\mathcal{O}\left(k\varphi\alpha^{\gamma}\right)$.

□

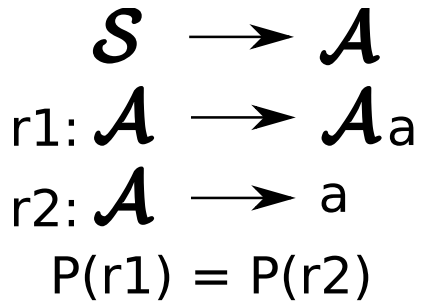


Figure B.1: Example grammar with equal rule weights (r1 and r2) where $\mathcal{S}$ is the starting non-terminal. All strings of a have a probability of being generated from this grammar.

Definition 5. *ϵ–variability states that any valid environment can have ϵ-perturbations and still be considered valid.*

Theorem. *The GCG algorithm is probabilistically complete.*

Proof. Due to the assumption of ϵ–variability we know there exist a volume of valid environments in the configuration space of possible environments. The GCG algorithm is probabilistically complete if all possible environments have a probability of being generated. If the rules defined in C_g have equal probabilities of being selected, such as the grammar in Figure B.1, then the GCG algorithm is probabilistically complete. □