Eliciting Collective Behaviors through Automatically Generated Environments

Benjamin T. Fine and Dylan A. Shell

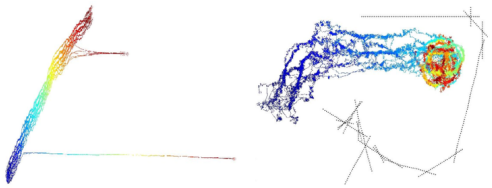
***Abstract*— Many groups of agents exhibit emergent collective behaviors. The environment in which the agents operate is one determinant of the resulting behaviors. This work shows how automatic enumeration of environments enables exploration of various collective behaviors that perform useful group functions (*e.g.* segregation, corralling, shape formation). Although groups of agents, such as mobile robots, can be manipulated through explicit control, this study shows that these systems can be usefully manipulated without resorting to such imperative means. This method has obvious uses for heterogeneous robot systems, especially those which include large numbers of simple agents. The method introduced is general, in that it takes as input: (1) algorithmic specifications of the environment generation, (2) a *black-box* model of the individual agent's control laws, and (3) a mathematical description of the task objective. To show the validity of the proposed method this investigation studies two behaviors (splitting and corralling) for three commonly studied motion models, including the well known Reynold's model. Simulations and physical multi-robot trials show that automatically generated environments can elicit pre-specified behaviors from a group of individual agents. Additionally, this work investigates the effects of a group's emergent properties on the ability to elicit the specified behavior via the environment. The findings suggest that automatically exploring environments can lead to better exploration and understanding of collective behaviors, including the identification of previously unknown emergent behaviors.**

I. Introduction

Collective emergent behaviors exhibited by groups of individuals are seen in our everyday world (*e.g.*, birds flocking, fish schooling, pedestrians forming lines). Certain aspects of these observed behaviors can be explained by understanding the individual agent's control laws that depend on local interactions among neighboring agents. However, it is difficult to define which resultant collective behaviors are elicited by the individual control laws. Some of these behaviors may only exist due to the effects the environment has on the individual agents, thus influencing the observed behaviors. Figure 1 (a) shows a commonly used structure (fishing weir) for corralling fish into a predetermined location. Using computer simulated agents, Figure 1 (b) shows how the environment can determine which group behaviors are exhibited.

Recent works [1, 2] have shown that a properly designed environment can be useful in manipulating the exhibited behaviors of a group of agents. Currently *ad hoc* techniques are used to design such environments, which relies heavily on experience and specific domain knowledge. While useful environments have been produced, the process is time consuming and cannot easily be generalized for a variety of agents, tasks, or application domains. Moreover, these approaches do not easily allow one the ability to investigate

(a) Human-made fishing weirs designed to corral fish in a predetermined location.



(b) The left figure shows motions of simulated agents in an empty environment and the right figure shows the effect the environment has on the same agents.

Fig. 1: Figure 1a shows motivating examples of human-made fishing weirs while Figure 1b demonstrates the effect an environment can have on simulated agents.

emergent properties of a group.

How an environment influences an agent's behavior depends on whether that agent is operating in a group; generating environments to achieve objectives from a collective, must take into account that the collective behaviors exhibited by the group may be very different from that exhibited by a lone individual. In practice this can help rather than hinder the process of control, as there are environments that can exploit collective dynamics. However, the connections between the local environment-agent interactions and the observed collective behaviors remain far from adequately understood. Section III-B shows, through simulations, the influence collective and emergent behaviors have on the environment's ability to manipulate a group of agents.

The generation and validation of environments that can elicit a pre-specified behavior from a group is difficult for two reasons: (1) how does one describe the infinite set of possible environments, and (2) how does one practically search among this set? To address these difficulties, this work proposes the use of *shape grammars* for the automatic generation of environments. Shape grammars [3] are a proven way of automatically generating structured shapes (environments) from a small set of primitive shapes and rules. Through implementations of two distinct grammars, this investigation shows that it is feasible to automatically generate a searchable set of environments and are able to

show that such environments do influence a group's behavior.

A. Related Works

Beyond explicit control, the most common approaches for manipulating a group of agents is either through the use of heterogeneous group members [4, 5, 6] or through the use of external agents [7, 8, 2] (*e.g.*, shepherds). Works that utilize heterogeneous group members typically use *informed* agents that have a preferred behavior bias (*e.g.*, preferred location in the environment) or a different set of control law gains, or additional information. One dilemma with this approach is that it is not always practical to introduce informed agents into a given group (*e.g.*, how would one introduce an *informed* agent into a flock of wild birds?). Furthermore, it is not always clear that the use of internal agents could elicit more complex behaviors, such as segregation based on agent characteristics (*e.g.*, separate the males from the females).

In the case of shepherding methods, individual group members are generally repulsed by the external agent(s), thus the typical question studied by these works is "What motions must the external agent(s) perform in order to get the group to the goal state?" The method presented in this work differs in two ways: (1) it illustrates that static environments suffice because they take advantage of inter-agent interactions to form predictable group behaviors and (2) it does not require a model of the individual agents for the generation of environments. It is important to note that this approach does not utilize a model of the agents to be manipulated, thus the approach is agnostic to the individual agents.

Other manipulation work has utilized the implicit agent control afforded by the environment [1]. These works show that a group of very simple robots can be successfully *guided* to a specific location in space through an environment comprised of static walls and one-way passive gates. Bobadilla *et al.* [1] used prior knowledge of how the simple robots will move and react with the environment in order to build an environment which was capable of eliciting the pre-specified behavior. The robots in this work do not communicate with the other robots, explicitly, which suggest this group is incapable of exhibiting more complex collective behaviors in the absence of an environment. Even though it is possible to manipulate the simple robots in [1], the slightly more complex agents studied in this work allow for environments to take advantage of the group's emergent properties (*e.g.*, aggregation), and thus can accomplish more complex tasks with less complexity in the environment.

II. Problem Statement and Approach

The approach aims to enumerate a set of environments ($\mathbb{E}$) that elicit a pre-specified behavior (B) from a group of agents that obey the given motion model[1] (M). In other words, the pre-specified behavior and the agent's motion model are provided to the system, which outputs a set of environments that elicit the behavior B from a group of agents acting according to M. The specified behavior 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).

The agent motion model represents the low-level control law each agent will execute during the simulation process (*e.g.*, Reynold's rules [9]). This model is only used in the validation of the generated environment and not in the enumeration process, therefore the generation of environments is agnostic to whether or not the group is homogeneous. Figure 2 presents an overview of the structure and data-flow of the environment generation and validation method.

To generate the set of valid environments for the specified behavior and motion model, the approach requires a schema of the environment, denoted by S. For this work, a valid environment is any environment that elicits the pre-specified behavior from the given group of agents. The environment schema represents both the environmental *building blocks* and the *rules* for the creation of environments. The building blocks of an environment describe the low-level features, such as a wall, starting region, or a wedge. The rules within a given schema detail how the various building-blocks can be manipulated. For example, a rule can state that two walls can be joined together to yield a wall twice the length of a single wall.

Using the given environment schema, the environment generation module applies the schema's rules to create a single environment (e_i) that is passed through an environment filter, denoted by the right facing arrow in Figure 2. The purpose of an environment filter is to reduce the number of environments that are rigorously tested through the use of basic domain knowledge. For instance, there may be resource limitations for constructing environments (*e.g.* limited length of fencing material); the filter could exclude any environments that exceed the upper bound on environmental resources. Any environment e_i that passes the filter is added to the environment queue[2]. It is important to note that a filter is not always required; the primary purpose of a filter is to allow the schemas to be as general as possible.

The system then validates the filtered environments by simulating the motions of the agents within the environment e_i. The motions generated in simulation will be analyzed and compared with the behavior B. If the simulated motions of the agents represent the specified behavior, then the environment e_i will be added to the set $\mathbb{E}$. When the simulated motions do not match the specified behavior B, e_i is discarded.

A. Implementation

The description to this point has been given in high-level and generic terms, thus the various modules and inputs to the system can be implemented in a variety of ways. This section details the current implementation of the approach, which includes a novel procedure for the enumeration of

[1]The motion model for the agents is a *black-box*; the details which determine the motion of the agents is not used for generating environments and is only used for environment validation.

[2]For the implementation in this work the queue is a simple first in first out queue.

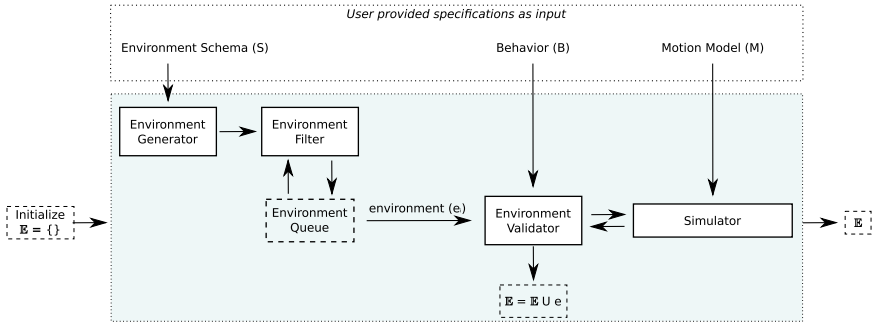


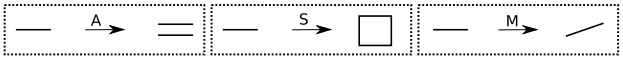
Fig. 2: A diagram of the presented method to automatically enumerate a set of environments that elicit a pre-specified behavior from a group of agents. The Environment Schema (*S*), Motion Model (*M*), and Behavior (*B*) are all user defined inputs to the system and remain constant during the system's execution. The output is a set of environments ($\mathbb{E}$) that elicit the specified behavior from the group of agents.

environments through the use of shape grammars (*i.e.*, shape grammars are used to implement the environment schemas).

Shape Grammars

Shape grammars are similar to typical symbol grammars [10] in that they have a set of symbols (shapes or building-blocks) and rules, 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. This study only considers two-dimensional polygonal shapes but, in general, shape grammars can be used in higher dimensions [11] and with non-polygonal shapes [12]. For the original definitions and a more detailed treatment of shape grammars, please see the seminal shape grammar study [3].

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 3a), to substitution, where the left-side shape is replaced by another shape (Figure 3b). 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 3c.

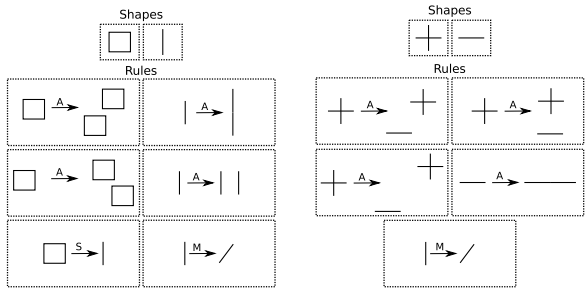


(a) Addition (b) Substitution (c) Modification

Fig. 3: Examples of three common rules in shape grammars.

Shape grammars can also use combinations 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 a part of the environment itself. These shapes are used to help define the underlying structure of the environment. Terminal shapes, such as walls (lines), are shapes the agents will interact with during the validation procedure and are items that must be physically constructed.

This work implements two grammars using the Shape Grammar Interpreter[3] (SGI) given in [14] along with two corresponding environmental filters. Figure 4 outlines the splitting and corralling shape grammars used in this work. For both of the grammars in Figure 4 there is a corresponding environmental filter. The *splitting grammar filter* only accepts environments that contain at least three terminal shapes. The *corralling grammar filter* only accepts environments that have an obstacle free path from the origin marker, which must be within the convex hull of the terminal shapes, to an arbitrary point outside of the convex hull of the terminals.



(a) Splitting Grammar (b) Corralling Grammar

Fig. 4: The splitting grammar uses a non-terminal shape (square), which is used to help define the structure of the environment and a single terminal shape (straight line). The corralling grammar uses one terminal shape (straight line) and one non-terminal (marker), which defines the origin.

Pre-specified Behaviors

The two behaviors chosen in this study are splitting and corralling. These behaviors where chosen because they are commonly exhibited by real (biological or physical implementation) systems. Splitting behaviors are defined by any behavior where the group fragments into at least two groups for k consecutive iterations. This study considers a group to be performing the corralling behavior if and only if the group does not fragment and the centroid of the agent's positions remains within radius r from the origin for at least k consecutive iterations of the simulation.

Motion Models

The agent's motion model is the low-level control law, or algorithm, which determines the next location the agent will move towards. For validation purposes, this work implements three motion models from the flocking literature; (1) Random Plus[4] [15] (RM+), (2) Simple Nearest Neighbor Plus [15] (SNN+), and (3) Reynold's Boids [9] (RB). It must be noted that in addition to the motion model, each agent is biased toward a single direction. This addition was necessary to ensure the group would pass through the environment. This

[3]SGI version 1.31 from the Source Forge repository [13].

[4]The *plus* term signifies the addition of environment avoidance to the motion model.

additional motion can be seen as fish moving with the flow of a river's current.

III. Findings

A. System Validation

Using MatLab (version R2011b) this study implements the aforementioned motion models, three specified behaviors (Table I), the two environment filters, and the framework of the proposed system (excluding the SGI software). The system is executed for 1000 environment generations for each of the three behaviors to verify the system was capable of generating environments that elicit a pre-specified behavior. Note that the same 1000 environments were used for both the simple split and balanced split behaviors as both behaviors where tested on environments generated by the splitting grammar. The results presented in this section show that the system successfully generates, filters, and validates environments that elicit a pre-specified behavior from a given group of agents obeying a particular motion model.

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.

TABLE I: Specifications of the behaviors used in this study. Each of the parameter values for the behaviors where arbitrary chosen before executing the system. This was done to avoid any validation errors due to over-tunning the parameter values.

Of the 1000 environments that where generated from both the splitting and corralling grammars, 53.7% and 50.0% passed the respective filters. Figures 5 and 6 are example environments that were generated by the SGI and that passed the respective environment filters. Examples of environments that did not pass the filtering process are not shown here due to space limitations.

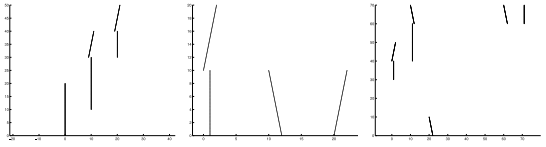


Fig. 5: Three environments that were generated using the grammar in Figure 4a and passed the splitting grammar filter.

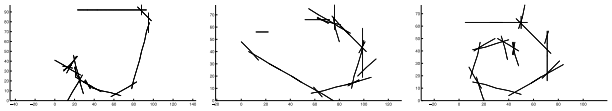
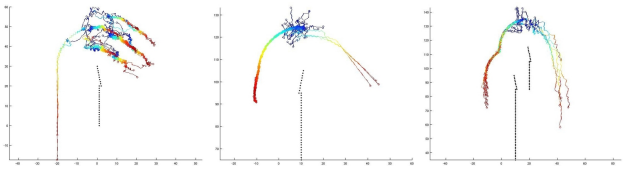


Fig. 6: Three environments that were generated using the grammar in Figure 4b and passed the corralling grammar filter.

For each environment that passed the environment filter procedure, the system conducted a single simulation to determine if the environment elicited the specified behavior from the given motion models. Table II 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. Figure 7 shows simulation results from three environments that elicited the simple split behavior. For all plots in this publication, the color gradient (blue to red) represents simulation time.

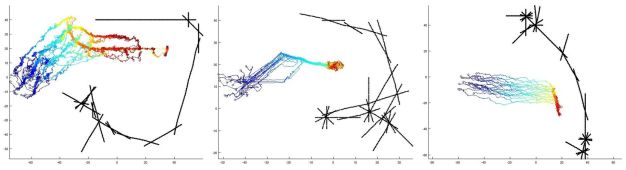
	RB	SNN+	RM+
Simple Split	48.98%	69.65%	86.41%
Balanced Split	49.35%	71.32%	87.15%
Corralling	60.28%	44.71%	50.70%

TABLE II: The percentage of environments that generated the pre-specified behaviors for the three motion models. These percentages only consider the number of environments that passed the filtering process.



(a) RB (b) SNN+ (c) RM+

Fig. 7: Three different environments that successfully elicit the simple split behavior for each motion model.



(a) RB (b) SNN+ (c) RM+

Fig. 8: Three different environments that successfully elicit the corralling behavior for each motion model.

To support the claim that the environments generated and validated by the system can successfully manipulate multi-robot systems, physical trials with four iRobot Creates were conducted. Each robot is equipped with a Hokuyo URG-04LX-UG01 laser range finder and an ASUS Eee PC. For the trials, a single environment that was generated and validated by the system for both the splitting and corralling grammars

was constructed. Figure 9 shows time series from two separate successful physical robot trials and Table III shows the results of all 30 robot trials in addition to comparable simulation trials.

	Robotic		
	RB	SNN+	RM+
Simple Split	40.0%	100.0%	80.0%
Corralling	20.0%	20.0%	0.00%
	Simulated		
	RB	SNN+	RM+
Simple Split	60.0%	80.0%	100.0%
Corralling	40.0%	20.0%	0.00%

TABLE III: Validation results of two environments (one generated from each grammar) for both robotic and simulated agents. Each pre-specified behavior was tested with one environment selected from the set of valid environments, that were generated for the initial validation trials, and simulated over five trials; totaling 30 robotic and 30 simulation trials. The simple split behavior was tested using four agents, while the corralling behavior was tested using three.

For the physical robot experiments the corralling behavior was slightly modified to include groups that only maintain two-thirds connectivity. In other words, if two of the three robots are considered 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.

B. Emergent Properties Influence of Controllability

From the previous literature and from the above simulations, it is clear that the environment can have an influence on the exhibited behaviors of a group. However, the effect of collective behaviors on the environment's ability to elicit a given behavior has not been the subject of extensive study. To demonstrate the effect of emergent properties, detailed simulations for five environments, that were validated for the simple split behavior, were conducted. For each environment, four sets of experiments, each with ten trials, were conducted. The only parameters modified for each set was group size and the starting radius of the group. Table IV shows the parameters[5] used and the percentage of simulations that elicited the simple split behavior.

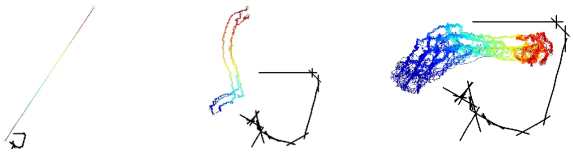
Examining the total percentages from Table IV one can see that the RM+ model exhibits the simple split behavior more frequently than both the RB and SNN+ models. This is due, in part, to the aggregation properties of the RB and SNN+ motion models. Moreover, notice that as the group size decreases the success rate increases. When the group size is two (Set 1) all of the environments almost always elicit the simple split behaviors. This suggest that the decrease in group size reduces the influence of the collective properties exhibited by the group, and thus affects the controllability of the group.

[5]The values in this work were hand-selected in order to highlight the influence of emergent properties. Further work must be done to properly classify the influence.

	# Agents	Start Radius	RB	SNN+	RM+
Set 1	2	10 units	100%	96.0%	100%
Set 2	25	10 units	60.0%	40.0%	84.0%
Set 3	25	40 units	0.00%	46.0%	62.0%
Set 4	100	40 units	96.0%	74.0%	96.0%
Total			64.0%	64.0%	85.5%

TABLE IV: Results from simulations exploring the simple split behavior. Each set contains simulation results from five environments and ten samples for each environment.

Another example where collective behaviors influence the environment's ability to manipulate the group can be seen in Figure 10. The simulation results in Figure 10 show the motions of agents obeying the RB motion model in the presence of an environment generated by the corralling grammar. The only difference in the results is the size of the group. These results show that the corralling behavior is only exhibited when the group is of sufficient size. Again, this suggests that some collective properties may afford more controllability of the group by the environment.



(a) One Agent (b) Three Agents (c) 50 Agents

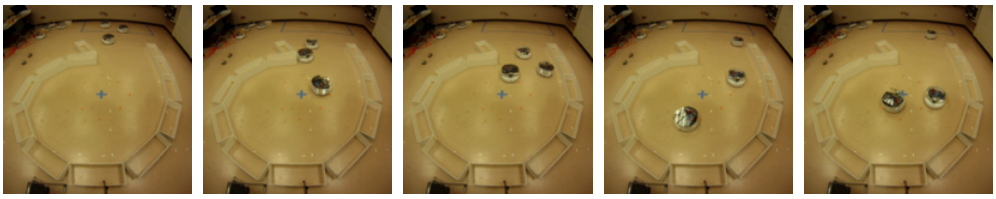
Fig. 10: These simulation results were generated using identical parameters with exception to group size. Together these results show how the corralling behavior is only elicited when the group is of sufficient size (Figure 10c).

IV. Conclusion

Observing groups of individual agents in the real world, it is clear that the environment plays a role in what behaviors the agents exhibit. Until this work, there have only been *ad hoc* methods for generating environments to elicit a pre-specified behavior from a group. This investigation has introduced and implemented a system that can automatically generate and validate environments that elicit a specified behavior from a group of individual agents obeying a given motion model. Through computer simulations and physical robot trials, this study has shown that the proposed system can indeed produce valid environments. Additionally, this work has shown (through simulation) that emergent properties of a group have an influence on the environments ability to elicit the specified behavior.

The use of the proposed system will allow one to explore many aspects of emergent and collective behaviors. For

(a) Splitting

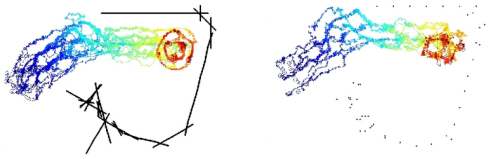


(b) Corralling

Fig. 9: These two time series shows a multi-robot system obeying the SNN+ motion model being manipulated by environments that were generated by the presented system.

example, the system could be easily extended to work that considers manipulating groups with external agents. If the environment is reduced down to a set of individual points, as shown in Figure 11, it is possible to use the generated environment as a *blueprint* for multi-robot formations; similar to the formations used in [8].

In addition to exploring various aspects of collective behaviors and emergence and what behaviors the environment exploits, many questions regarding the development of grammars and behavior specifications still need to be investigated. This tool needs to be applied to more complex behaviors, such as segregation of fast and slow (strong and weak) agents and splitting agents into individual pens. Possible approaches including adding in attractive terminal shapes (*e.g.*, food source) and/or adding in dynamic terminal shape



(a) Environment (b) Possible Robot Formation

Fig. 11: The left figure shows an environment that was generated via the proposed 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.

ACKNOWLEDGMENTS

The authors would like to thank Tomáš Treščák for the time and kind support he gave us in the understanding and use of the Shape Grammar Interpreter software. We would also like to thank the members of the Artificial Intelligent Robotics Laboratory at Texas A&M University for their role in developing the aforementioned concepts and their assistance in conducting the physical robot trials.

REFERENCES

[1] L. Bobadilla, O. Sanchez, and J. Czarnowski, "Controlling wild bodies using discrete transition systems," *Advanced Robots*, 2011.

[2] R. Vaughan, N. Sumpter, J. Henderson, A. Frost, and S. Cameron, "Experiments in Automatic Flock Control," *Robotics and Autonomous Systems*, vol. 31, pp. 109–117, Apr. 2000.

[3] G. Stiny, "Introduction to shape and shape grammars," *Environment and Planning B: Planning and Design*, vol. 7, no. 3, pp. 343–351, 1980.

[4] L. Conradt, J. Krause, I. D. Couzin, and T. J. Roper, ""Leading According to Need" in Self-Organizing Groups," *The American Naturalist*, vol. 173, no. 3, pp. 304–312, Mar. 2009.

[5] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision making in animal groups on the move," *Nature*, vol. 433, no. 3, pp. 513–516, Feb. 2005.

[6] S. Gueron, S. A. Levin, and D. I. Rubenstein, "The Dynamics of Herds: From Individuals to Aggregations," *Jour. of Theor. Biology*, vol. 182, no. 1, pp. 85–98, Sep. 1996.

[7] J.-M. Lien, O. Bayazit, R. Sowell, S. Rodriguez, and N. M. Amato, "Shepherding behaviors," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 4, May 2004, pp. 4159 – 4164.

[8] J.-M. Lien, S. Rodriguez, J.-P. J. Malric, and N. M. Amato, "Shepherding behaviors with multiple shepherds," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Barcelona, Spain, Apr. 2005, pp. 3413–3418.

[9] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *Computer Graphics* 21(4), pp. 25–34, 1987.

[10] N. Chomsky, "Three models for the description of language," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 113–124, 1956.

[11] H. Chau, X. Chen, A. McKay, and A. de Pennington, "Evaluation of a 3D Shape Grammar Implementation," in *Design Computing and Cognition*. Klumer Academic Publishers, 2004, pp. 357–376.

[12] I. Jowers and C. Earl, "The Construction of Curved Shapes," *Environment and Planning B: Planning and Design*, vol. 37, no. 1, pp. 42–58, 2010.

[13] T. Treščák. (2012, Jul.) The shape grammar interpreter source forge repository. Sourceforge.net/projects/sginterpreter.

[14] T. Treščák, I. Rodriguez, and M. Esteva, "General Shape Grammar Interpreter for Intelligent Designs Generations," in *The 6th Int. Conf. on Computer Graphics, Imaging and Visulization*, Aug. 2009.

[15] S. V. Viscido, M. Miller, and D. S. Wethey, "The Dilemma of the Selfish Herd: The Search for a Realistic Movement Rule," *Jour. of Theor. Biology*, vol. 217, no. 2, pp. 183–194, 2002.