Self-Reconfiguring Modular Robot Learning for Lower-Cost Space Applications

Andrew B. Jones
Dept. of Computer Science
North Dakota State University
1320 Albrecht Blvd
Fargo, ND 58102
andrew.jones.4@ndsu.edu

Thomas Cameron Jr.
Dept. of Mechanical Engineering
North Dakota State University
1320 Albrecht Blvd
Fargo, ND 58102
thomas.cameronjr@ndsu.edu

Benjamin Eichholz
Dept. of Mechanical Engineering
North Dakota State University
1320 Albrecht Blvd
Fargo, ND 58102
benjamin.eichholz@ndsu.edu

David Loegering
Dept. of Computer Science
North Dakota State University
1320 Albrecht Blvd
Fargo, ND 58102
david.w.loegering@ndsu.edu

Taylor Kray
Dept. of Mechanical Engineering
North Dakota State University
1320 Albrecht Blvd
Fargo, ND 58102
taylor.kray@ndsu.edu

Jeremy Straub
Dept. of Computer Science
North Dakota State University
1320 Albrecht Blvd
Fargo, ND 58102
jeremy.straub@ndsu.edu

***Abstract*—The applications for self-reconfiguring modular robots are far reaching. Their advantage comes from their ability to form into many different shapes which allows them to perform a multitude of tasks. This is especially advantageous for space applications, due to the associated high cost of launching equipment into space. Self-reconfiguring robots are not a panacea, however. A trade-off between flexibility and capability exists. A modular robot that configures itself to do a specific task may be unable to perform as well as a robot that was designed specifically for that task. This disadvantage can be mitigated by having the units form into optimal configurations for the associated tasks. This is difficult to pre-define for tasks that are not known a-priori. To this end, the robots are designed to learn and choose the most efficient ways of reorganizing into the desired configuration. In this paper, the use of machine learning for commanding a self-reconfiguring modular robot system is investigated. A method is proposed under which the robot system would be able to learn through a trial and error process to form itself into different configurations, more optimally.**

TABLE OF CONTENTS

1. INTRODUCTION

Modular robot systems consist of multiple similar interconnecting units that can arrange into different configurations [1]. Self-reconfiguring modular robots take this a step farther by doing this autonomously.

The applications for self-reconfiguring modular robots are far reaching. Their advantage comes from their ability to form into many different shapes which allows them to perform a multitude of tasks, such as moving through small passages or climbing over obstacles [2]. This is especially advantageous for space applications, due to the associated high cost of launching equipment into space.

Self-reconfiguring robots are not a panacea, however. A trade-off between flexibility and capability exists. A modular robot that configures itself to do a specific task may be unable to perform as well as a robot that was designed specifically for that task. This disadvantage can be mitigated by having the units form into optimal configurations for the associated tasks. This is difficult to pre-define for tasks that are not known a-priori. To this end, the robots are designed to learn and choose the most efficient ways of reorganizing into the desired configuration.

The paper is organized as follows. First, background information on previous research in modular robotics is discussed. Then, the proposed learning method is detailed. Last, as the high cost of launching equipment into space provides significant benefit to using equipment that is versatile and can accomplish many objectives, the paper concludes with an assessment in this context. The efficacy of modular robots as a solution to several real-world scenarios is assessed, with a focus on cost savings, the robots' ability to re-form to accomplish the objectives and how optimally the objectives are performed.

2. BACKGROUND

This section discusses research in the field of modular robotics. First, application areas are explored. Then, three topics in modular robot systems are analyzed in subsections. These topics include how the system determines configurations, forms into configurations, and coordinates/locomotes while in a configuration.

978-1-5386-6854-2/19/$31.00 ©2019 IEEE

Modular robot systems are highly versatile and have a wide range of potential application areas. Ryland and Cheng [3] developed iMobot with an intended goal of using it for search and rescue operations. Yim, et al. [4] provided an overview of many application areas for self-reconfigurable robot systems. An example being the "bucket of stuff" application where the robot system would reconfigure into shapes that allow them to perform chores such as changing the oil in a car. Another area of application discussed was using modular robots for space exploration due to their ability to change configuration and handle unforeseen situations.

Determining Configurations

Determining a configuration can be accomplished in many different ways. Gilpan and Rus [5] presented an algorithm for modular robots to duplicate/mimic the shape of presented 3D objects. I-ming and Burdick [6] used a genetic algorithm and kinematic classifications in order to find optimal configurations for a task. Stoy and Brandt [7] developed an algorithm for calculating the number of possible configurations a modular system can have, given the number of modules and module parameters. Icer, Giusti and Althoff [8] presented an algorithm for generating configurations that pruned possibilities over stages in order to increase the resulting configuration's mobility and obstacle avoidance capability.

Jin and Meng [9] proposed the term morphogenetic robotics, which is defined as a class of methodologies in robotics for designing self-organizing, self-reconfigurable, and self-repairable single- or multi-robot systems, using genetic and cellular mechanisms governing biological morphogenesis. Their approach for modular robot systems consisted of using robots that were able to attract and repel each other (by signaling), with a gene regulatory network (GRN) controlling the state/signal for each robot. In [10], they expanded on previous work with a two-layer hierarchical mechanochemical model for the self-reconfiguration of modular robots under changing environments. The first layer generated patterns using a virtual-cell based mechanochemical model, and the second layer coordinated the modular robots to form the pattern.

Forming into Configurations

Forming into a configuration is a form of self-assembly, which is the self-organized creation of structures composed of independent entities that are autonomous in their control [11]. If all the robots in the modular robot system are identical, computing the goal locations for each unit isn't necessary. This property would set self-reconfiguration apart from the intractable warehouse problem where modules are assigned unique IDs and must be placed at specific locations [12].

Inspired from social insect studies, Sahin, et al. [11] demonstrated a robotic system, called swarm-bot, that consisted of a swarm of mobile robots that had the ability to connect/disconnect from each other in order to form different kinds of structures. Dutta, et al. [13] proposed an algorithm based on graph isomorphism, where the modular robots currently in a configuration selected locations in the new configuration using a utility-based framework, while retaining portions of their original configuration (when possible) to save time.

Locomotion in Configurations

The ability to move and coordinate while in a configuration is an important aspect of modular robot systems. The motion model for modular robots is generally classified as being lattice-type or chain-type [14]. The lattice-type involves the modular robot continuously reconfiguring in order to move (modules attaching and detaching over a lattice of other modules). In contrast, chain-type involves the modular robot locomoting in a static configuration (i.e. without using reconfiguration) [14].

For this task, Butler, Murata and Rus [2] developed homogeneous distributed algorithms for the locomotion of a generic self-reconfiguring robot system. Sproewitz, et al. [14] proposed a framework for producing locomotion patterns that used a central pattern generator (CPG) and a gradient-free optimization algorithm. Kamimura, et al. [15] applied a neural oscillator network to generate stable locomotion patterns.

3. Proposed System Overview

In this section, we propose a learning technique that can be used to augment a self-reconfiguring modular robot system. Using our proposed technique, the robot system is intended to form itself into more efficient configurations, using knowledge learned through previous application trials.

There are two components to the proposed technique. The first component generates new configurations based on perceived needs and known modular robot capabilities. Second, the fitness of previous configurations is retained to inform future decision-making. The fitness scores are task dependent, as the utility of a configuration must be based on how useful it is to accomplish the specified task. Therefore, configurations may have several fitness scores (associated with different tasks) but only the most relevant score or scores are considered. A weighted average of multiple scores is used when a proposed application doesn't directly correspond with a previously performed task.

The choice of whether to use a new configuration or an established, previously tested, configuration has a number of factors. The primary factor is if the collection of tested configurations for this task all have scores that fall below a pre-determined threshold, such that they were considered relatively unsuccessful for this task. The threshold value is specified for each task, with tasks that require a higher standard of performance necessitating a corresponding higher threshold value.

Once the configuration is chosen, the resulting utility of the configuration is evaluated and used for future reference. The learning procedure is also used to update the known attributes

of the robot model. The evaluated capabilities of the robots are assessed and updated following deviations from expected results. For example, the mobility and capabilities of a chain of 'n' linearly connected robots would be initially mathematically calculated based on the properties of a smaller chain but updated if these initial estimates are proven incorrect.

4. Proposed System Detail

The input into the system (depicted in Figure 1) is a specified objective, which is parsed into requirements that a modular configuration must meet. From the requirements, the configuration layout is chosen. Then, the system checks whether it has a stored configuration that meets these requirements and configuration layout specifications. Although, the requirements are more heavily weighted than the configuration layout specifications for this case. From this, the system determines whether to use an existing configuration or to start the process of generating a new one.

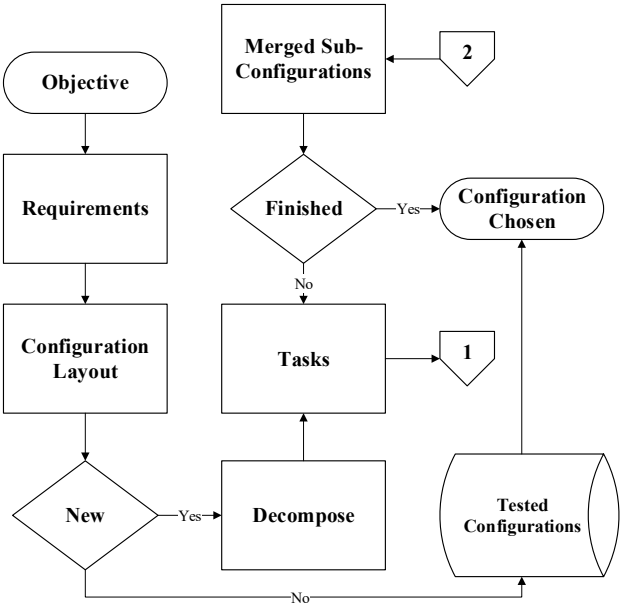


Figure 1. Flowchart showing the main proposed system.

In the case of choosing to use an existing configuration, the process is finished. Thus, base cases may be cached and used as a quick solution if circumstances do not permit generating a new configuration (due to time constraints or other risks). It also allows previously generated and tested configurations to be reused. Although, the goal of this process is the improve the utilized configurations; therefore, the results of the configuration performing the task is recorded. The fitness score is then updated based on these results, along with its designation being changed from estimated to observed. Thus, in subsequent selections, the observed fitness would be used to determine if it is a viable configuration for the requirements. The different labels for the fitness scores are as follows:

- Estimated: the fitness score was determined by the fitness estimation system.
- Observed: the fitness score is based on observed results from physically performing the requirements.
- Suggested: the fitness score was determined based on user input but can be overwritten based on observed results.
- Absolute: the fitness score was determined based on user input and will not be overwritten.

New Configurations

To generate a new configuration, the system first decomposes the configuration layout and requirements into subtasks. Each sub-task is then used as input for the generation loop, depicted in Figure 2. The generation loop first checks if it has time available to generate a new sub-configuration. This time constraint check is introduced to accommodate time constrained or emergency situations, where a suitable configuration is not currently in the database. In the case of time not being available, this stage goes straight to the choosing phase and only considers previously generated and/or tested sub-configurations.

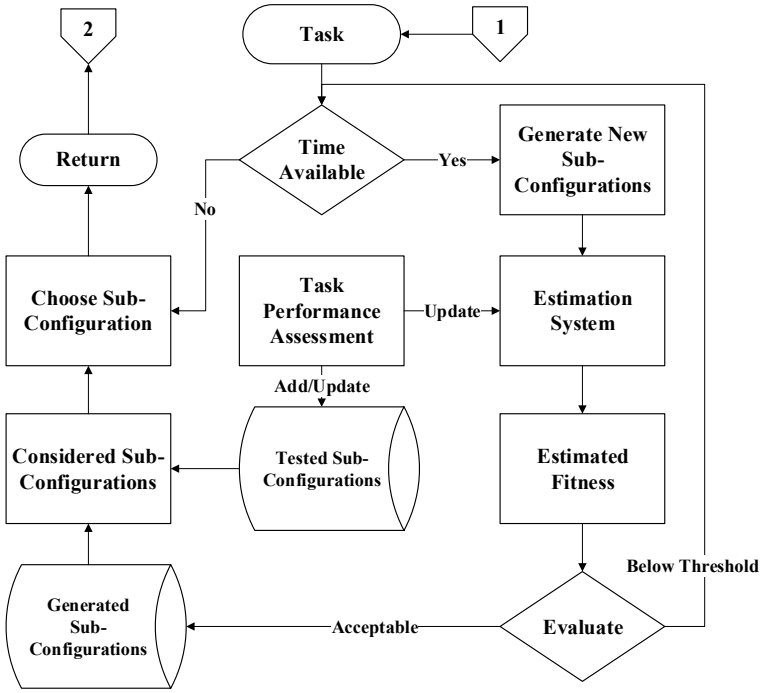


Figure 2. Flowchart showing the process of choosing a sub-configuration.

In the case that the system has time, it generates new sub-configurations. These resulting configurations are then run through the estimation system and given an estimated fitness score. The scores that are above a specified threshold are added to the set of generated sub-configurations. The threshold can be fixed or based on the top-ranked member of

the sub-configuration sets. If none of the scores are above the threshold then the process again checks if it has time to repeat the generation process Thus, the process repeats until an acceptable candidate sub-configuration is found or until it runs out of time. In the case of scores being above the specified threshold, then they are added to the set of generated sub-configurations and the process chooses which sub-configuration to return based on all of the considered sub-configurations.

Sub-Configuration Generation

Certain modular robot models and actuators may have difficulty when attempting to scale a configuration. For instance, consider the case of a cubic modular robot model in a homogeneous system (depicted in Figure 3). If an element of rotation is introduced in the form of elbow joints, individual robots may interfere with each-other without the plausibility of adequate connectivity to attachment points. In this case, modular robots could only be added unilaterally to maintain rotational mobility in all directions. If a module is added perpendicular to the specified unilateral direction, the module would impede further rotation about the mutually exclusive perpendicular axis.

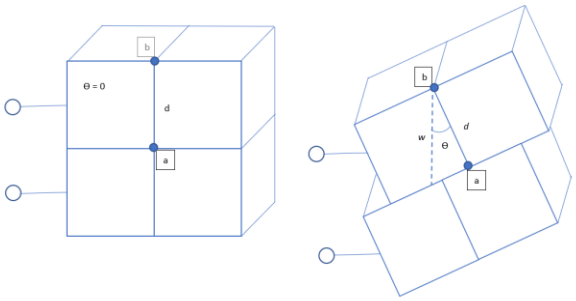


Figure 3. Diagram depicting how attaching more modules can impair certain motions.

In this way, certain capabilities of a configuration could be bound to calculated maximums. I.e. an arm with certain rotational capabilities could be constrained to a linear chain (with potential for a manipulator on the end). Thus, for active (mobile) sub-configurations, the base level sub-configurations would consist of these calculated motion models. For non-base level sub-configurations, it generates a new sub-configuration using a genetic algorithm. For the initial population, the genetic algorithm uses high scoring (lower level) configurations from both the tested and generated sub-configuration sets.

Configuration Layout

In the proposed approach, the configuration would be composed of two different categories: Active parts, and passive parts (depicted in Figure 4). Moreover, a configuration could be all active, all passive, or any combination of the two. The active parts are for mobility, object manipulation, and environment interaction through the use of actuators.

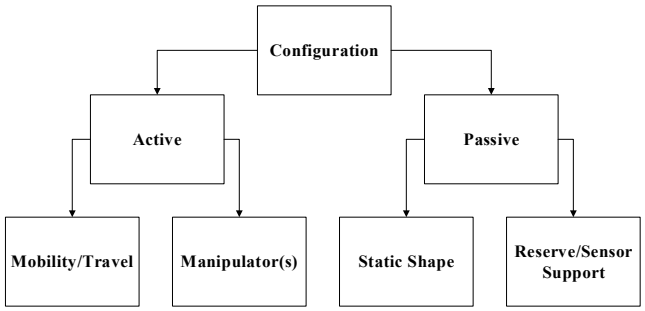


Figure 4. Parts of the configuration layout.

In contrast, the passive parts are for static shapes, sensor support placement, and reserve units. Here, a static shape refers to structural support or for shapes that may be useful for a task, such as a bin for holding objects. Sensor support refers to the presence of sensors in a modular robot unit that may benefit, or be less constricted by, being located in a certain position or orientation. For instance, sensors such as cameras may be impaired by line of sight if blocked. Modular units could also simply be put in reserve, such that they are not currently in use but still need to move with the other modular units in the configuration.

Depending on the task, the configuration could shift to accommodate the requirements. For instance, if a task only required the robot to move to another location, then the configuration could then focus on mobility aspects and sensor placement – with the rest going to reserve. In general, the mobility of a configuration is dependent on the number of modular robots in the configuration, as more robots would mean more mass.

Fitness Estimation System

The fitness estimation system, depicted in Figure 5, is centered around the use of a physics engine. The physics engine is updated based on the current or target environment, such that gravity, terrain and weather are accounted for (if information is available). To estimate the fitness, the physics engine takes a configuration, configuration attributes, and the task requirements as inputs. The physics engine then simulates the configuration's ability to perform the task requirements, and outputs a fitness score based on the estimated effectiveness. In general, the configurations could also be sub-configurations of a larger configuration, which would have its own fitness score for a corresponding sub-task. Thus, the methodology of decomposing the tasks directly influences the fitness scoring of the modular configurations.

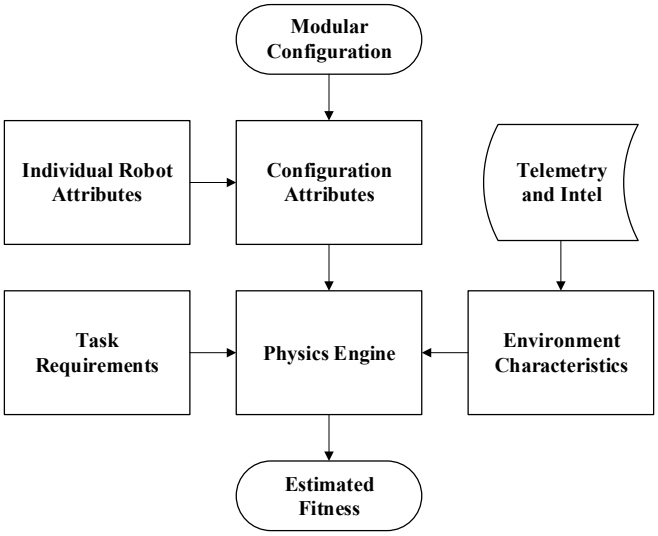


Figure 5. Flowchart showing the inputs to the physics engine.

The physics engine makes use of a helper metric, termed configuration attributes, which is used to aid the physics engine in estimating tolerances and other qualities. This is used because certain attributes and characteristics of a single modular robot change the resulting configuration when an attachment to another modular robot is made. The primary factors considered for this change are depicted in Figure 6. While the durability of a single modular unit is dependent on the make of the unit, the durability of a configuration also factors in the strength of the attachment(s) between units. Similarly, the mobility of a single unit can differ greatly from that of a configuration. The mobility involved with attachment points is the resulting capability of one robot to move (or affect the movement of) another robot when they are attached.

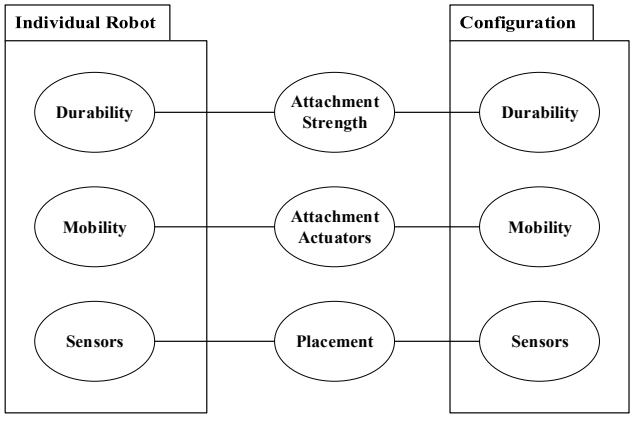


Figure 6. Diagram depicting how the modular robot model could affect the quality of a configuration.

5. Applicability to Space Missions

In this section, the proposed learning technique and modular robotics in general are analyzed in terms of applicability to missions in space. First, the necessary qualities for robotics that operate in space is discussed. Then, the potential utility of the proposed learning system in this application area is evaluated.

Yim, et al. [16] outlined three characteristics of equipment that are advantageous for missions in space:

1. *Compactness and Lightness*: the cost of sending equipment into space is directly correlated to its size and weight.
2. *Robustness*: missions often have only one attempt to succeed (usually at great cost).
3. *Versatility and adaptability*: in exploration where the environments are inherently unknown, adaptability increases the chance of success.

Modular robot systems conceptually have all three of these favorable characteristics. In terms of robustness, homogenous modular systems may be ideal so that malfunctioning units have less of an impact on the overall system. The versatility of modular systems stems from their ability to form into many different (task relevant) configurations [2]. In this regard, the usage of modular robots may provide a means to reduce the amount of equipment that would need to be sent into space. Furthermore, the proposed learning system may have advantageous characteristics for missions in space, particularly the ability to update the currently perceived capabilities of the robot model based on feedback. The reasoning behind this is that if the modular robot system is introduced into an environment with significantly different characteristics (i.e., varying gravity), it could potentially adjust to these conditions over time. This may, in effect, lower costs for missions due to the modular robot potentially being a substitute for equipment that would otherwise need to be launched into space.

6. Conclusion

In this paper, the use of machine learning for commanding a self-reconfiguring modular robot system was investigated. A method was proposed under which the robot system would be able to learn through a trial and error process to form itself into different configurations, more optimally.

Acknowledgements

Facilities and equipment used for this work were provided by the North Dakota State University Department of Computer Science.

References

[1] M. Yim, Ying Zhang, and D. Duff, "Modular robots," *IEEE Spectr.*, vol. 39, no. 2, pp. 30–34, 2002.

[2] Z. Butler, S. Murata, and D. Rus, "Distributed Replication Algorithms for Self-Reconfiguring Modular Robots," in *Distributed Autonomous Robotic Systems 5 (DARS'02)*, 2002, pp. 37–48.

[3] G. G. Ryland and H. H. Cheng, "Design of iMobot, an intelligent reconfigurable mobile robot with novel locomotion," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 60–65.

[4] M. Yim *et al.*, "Modular Self-Reconfigurable Robot Systems [Grand Challenges of Robotics]," *IEEE Robot. Autom. Mag.*, vol. 14, no. 1, pp. 43–52, Mar. 2007.

[5] K. Gilpin and D. Rus, "What's in the bag: A distributed approach to 3D shape duplication with modular robots," *Robot. Sci. Syst.*, 2012.

[6] I-Ming Chen and J. W. Burdick, "Determining task optimal modular robot assembly configurations," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, 1995, vol. 1, pp. 132–137.

[7] K. Stoy and D. Brandt, "Efficient enumeration of modular robot configurations and shapes," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 4296–4301.

[8] E. Icer, A. Giusti, and M. Althoff, "A Task-Driven Algorithm for Configuration Synthesis of Modular Robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5203–5209.

[9] Y. Jin and Y. Meng, "Morphogenetic Robotics: An Emerging New Field in Developmental Robotics," *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.*, vol. 41, no. 2, pp. 145–160, Mar. 2011.

[10] Y. Meng, Y. Zhang, and Y. Jin, "Autonomous Self-Reconfiguration of Modular Robots by Evolving a Hierarchical Mechanochemical Model," *IEEE Comput. Intell. Mag.*, vol. 6, no. 1, pp. 43–54, Feb. 2011.

[11] E. Sahin *et al.*, "SWARM-BOT: pattern formation in a swarm of self-assembling mobile robots," in *IEEE International Conference on Systems, Man and Cybernetics*, 2002, vol. 4, p. 6.

[12] D. Rus, Z. Butler, K. Kotay, and M. Vona, "Self-reconfiguring robots," *Commun. ACM*, vol. 45, no. 3, 2002.

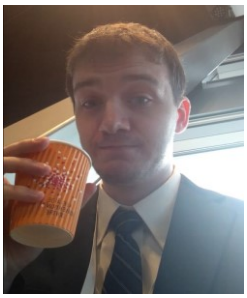
[13] A. Dutta, P. Dasgupta, and C. Nelson, "Distributed configuration formation with modular robots using (sub)graph isomorphism-based approach," *Auton. Robots*, Apr. 2018.

[14] A. Sproewitz, R. Moeckel, J. Maye, and a. J. Ijspeert, "Learning to Move in Modular Robots using Central Pattern Generators and Online Optimization," *Int. J. Rob. Res.*, vol. 27, no. 3–4, pp. 423–443, 2008.

[15] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji, "Automatic Locomotion Pattern Generation for Modular Robots," in *IEEE International Conference on Robotics & Automation*, 2003, vol. 70, no. 690, pp. 459–466.

[16] M. Yim, K. Roufas, D. Duff, Y. Zhang, C. Eldershaw, and S. Homans, "Modular reconfigurable robots in space applications," *Auton. Robots*, vol. 14, no. 2–3, pp. 225–237, 2003.

Biography



Andrew Jones *received a master's degree in Software Engineering from North Dakota State University (NDSU) in 2016. He is currently a Software Engineering Ph.D. student studying artificial intelligence and robotics. He has developed software for the OpenOrbiter Small Spacecraft Development Initiative. He has also served as the team lead for autonomous robot development competitions, such as the IGVC.*

Jeremy Straub *is an Assistant Professor in the Department of Computer Science at the North Dakota State University. He holds a Ph.D. in Scientific Computing, an M.S. and an M.B.A. and has published over 40 journal articles and over 120 full conference papers, in addition to making numerous other conference presentations. Straub's research spans the gauntlet between technology, commercialization and technology policy. In particular, his research has recently focused on robotic command and control, aerospace command and 3D printing quality assurance. Straub is a member of Sigma Xi, the AAAS, the AIAA and several other technical societies, he has also served as a track or session chair for numerous conferences.*

Thomas Cameron Jr*. is an undergraduate student studying mechanical engineering at North Dakota State University.*

Benjamin Eichholz *is an undergraduate student studying mechanical engineering at North Dakota State University.*

David Loegering *is an undergraduate student studying computer science at North Dakota State University.*

Taylor Kray *is an undergraduate student studying mechanical engineering at North Dakota State University.*