COLLECTIVE BEHAVIOR

An integrated system for perception-driven autonomy with modular robots

Jonathan Daudelin[1]*†, Gangyuan Jing[1]*†, Tarik Tosun[2]*†, Mark Yim[2], Hadas Kress-Gazit[1], Mark Campbell[1]

The theoretical ability of modular robots to reconfigure in response to complex tasks in a priori unknown environments has frequently been cited as an advantage and remains a major motivator for work in the field. We present a modular robot system capable of autonomously completing high-level tasks by reactively reconfiguring to meet the needs of a perceived, a priori unknown environment. The system integrates perception, high-level planning, and modular hardware and is validated in three hardware demonstrations. Given a high-level task specification, a modular robot autonomously explores an unknown environment, decides when and how to reconfigure, and manipulates objects to complete its task. The system architecture balances distributed mechanical elements with centralized perception, planning, and control. By providing an example of how a modular robot system can be designed to leverage reactive reconfigurability in unknown environments, we have begun to lay the groundwork for modular self-reconfigurable robots to address tasks in the real world.

INTRODUCTION

Modular self-reconfigurable robot (MSRR) systems are composed of repeated robot elements (called modules) that connect together to form larger robotic structures and can self-reconfigure, changing the connective arrangement of their own modules to form different structures with different capabilities. Since the field was in its nascence, researchers have presented a vision that promised flexible, reactive systems capable of operating in unknown environments. MSRRs would be able to enter unknown environments, assess their surroundings, and self-reconfigure to take on a form suitable to the task and environment at hand (*1*). Today, this vision remains a major motivator for work in the field (*2*).

Continued research in MSRR has resulted in substantial advancement. Existing research has demonstrated MSRR systems self-reconfiguring, assuming interesting morphologies, and exhibiting various forms of locomotion, as well as methods for programming, controlling, and simulating modular robots (*1*, *3*–*15*). However, achieving autonomous operation of a self-reconfigurable robot in unknown environments requires a system with the ability to explore, gather information about the environment, consider the requirements of a high-level task, select configurations with capabilities that match the requirements of task and environment, transform, and perform actions (such as manipulating objects) to complete tasks. Existing systems provide partial sets of these capabilities. Many systems have demonstrated limited autonomy, relying on beacons for mapping (*16*, *17*) and human input for high-level decision-making (*18*, *19*). Others have demonstrated swarm self-assembly to address basic tasks such as hill climbing and gap crossing (*20*, *21*). Although these existing systems all represent advancements, none has demonstrated fully autonomous, reactive self-reconfiguration to address high-level tasks.

This paper presents a system that allows modular robots to complete complex high-level tasks autonomously. The system automatically selects appropriate behaviors to meet the requirements of the task and constraints of the perceived environment. Whenever the task and environment require a particular capability, the robot autonomously self-reconfigures to a configuration that has that capability. The success of this system is a product of our choice of system architecture, which balances distributed and centralized elements. Distributed, homogeneous robot modules provide flexibility, reconfiguring between morphologies to access a range of functionality. Centralized sensing, perception, and high-level mission planning components provide autonomy and decision-making capabilities. Tight integration between the distributed low-level and centralized high-level elements allows us to leverage advantages of distributed and centralized architectures.

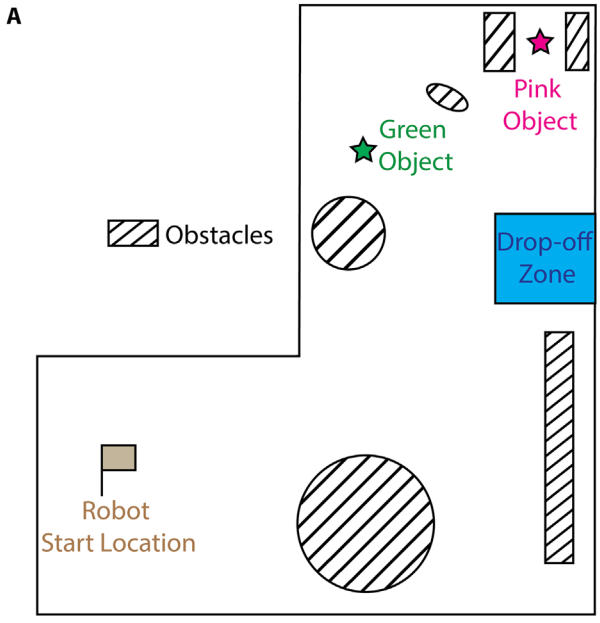
The system is validated in three hardware demonstrations, showing that, given a high-level task specification, the modular robot autonomously explores an unknown environment, decides whether, when, and how to reconfigure, and manipulates objects to complete its task. By providing a clear example of how a modular robot system can be designed to leverage reactive reconfigurability in unknown environments, we have begun to lay the groundwork for reconfigurable systems to address tasks in the real world.

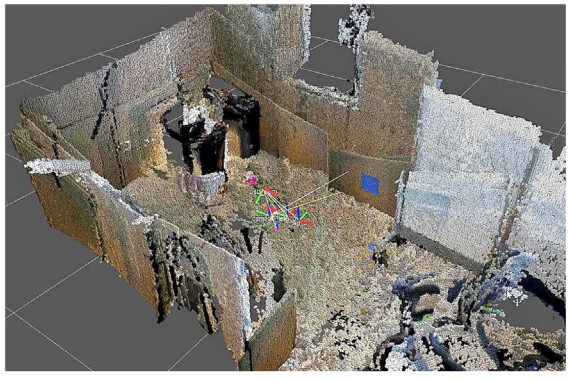
RESULTS

We demonstrate an autonomous, perception-informed, modular robot system that can reactively adapt to unknown environments via reconfiguration to perform complex tasks. The system hardware consists of a set of robot modules (that can move independently and dock with each other to form larger morphologies), a sensor module that contains multiple cameras, and a small computer for collecting and processing data from the environment. Software components consist of a high-level planner to direct robot actions and reconfiguration and perception algorithms to perform mapping, navigation, and classification of the environment. Our implementation is built around the SMORES-EP modular robot (*22*) but could be adapted to work with other modular robots.

Our system demonstrated high-level decision-making in conjunction with reconfiguration in an autonomous setting. In three hardware demonstrations, the robot explored an a priori unknown environment and acted autonomously to complete a complex task. Tasks

[1]Department of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY, USA. [2]Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA, USA.
*These authors contributed equally to this work.
†Corresponding author. Email: jd746@cornell.edu (J.D.); tarikt@seas.upenn.edu (T.T.); gj56@cornell.edu (G.J.)





C

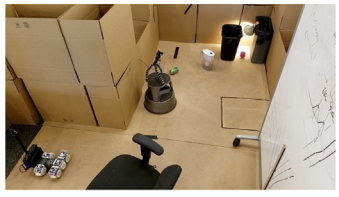
Environmental Setup	Task Description
	Demonstration I: Explore environment to find all pink or green objects and blue dropoff zone. Deliver all objects to dropoff zone.
	Demonstration II: Explore environment to find mailbox, then deliver a circuit to the box.
	Demonstration III: Explore environment to find package, then place a stamp on the package.

Fig. 1. Environments and tasks for demonstrations. (**A**) Diagram of demonstration I environment. (**B**) Map of environment 1 built by visual SLAM. (**C**) Setups and task descriptions.

were specified at a high level: users did not explicitly specify which configurations and behaviors the robot should use; rather, tasks were specified in terms of behavior properties, which described desired effects and outcomes (*23*). During task execution, the high-level planner gathered information about the environment and reactively selected appropriate behaviors from a design library, fulfilling the requirements of the task while respecting the constraints of the environment. Different configurations of the robot have different capabilities (sets of behaviors). Whenever the high-level planner recognized that task and environment required a behavior the current robot configuration could not execute, it directed the robot to reconfigure to a different configuration that could execute the behavior.

Figure 1 shows the environments used for each demonstration, and Fig. 2 shows snapshots during each of the demonstrations. A video of all three demonstrations is available as movie S1.

In demonstration I, the robot had to find, retrieve, and deliver all pink- and green-colored metal garbage to a designated drop-off zone for recycling, which was marked with a blue square on the wall. The demonstration environment contained two objects to be retrieved: a green soda can in an unobstructed area and a pink spool of wire in a narrow gap between two trash cans. Various obstacles were placed in the environment to restrict navigation. When performing the task, the robot first explored by using the "Car" configuration. Once it located the pink object, it recognized the surrounding environment as a "tunnel" type, and the high-level planner reactively directed the robot to reconfigure to the "Proboscis" configuration, which was then used to reach between the trash cans and pull the object out in the open. The robot then reconfigured to Car, retrieved the object, and delivered it to the drop-off zone that the system had previously seen and marked during exploration. Figure 1B shows the resulting three-dimensional (3D) map created from simultaneous localization and mapping (SLAM) during the demonstration.

For demonstrations II and III, the high-level task specification was the following: Start with an object, explore until finding a delivery location, and deliver the object there. Each demonstration used a different environment. For demonstration II, the robot had to place a circuit board in a mailbox (marked with pink-colored tape) at the top of a set of stairs with other obstacles in the environment. For demonstration III, the robot had to place a postage stamp high up on the box that was sitting in the open.

For demonstration II, the robot began exploring in the "Scorpion" configuration. Shortly, the robot observed and recognized the mailbox and characterized the surrounding environment as "stairs." On the basis of this characterization, the high-level planner directed the

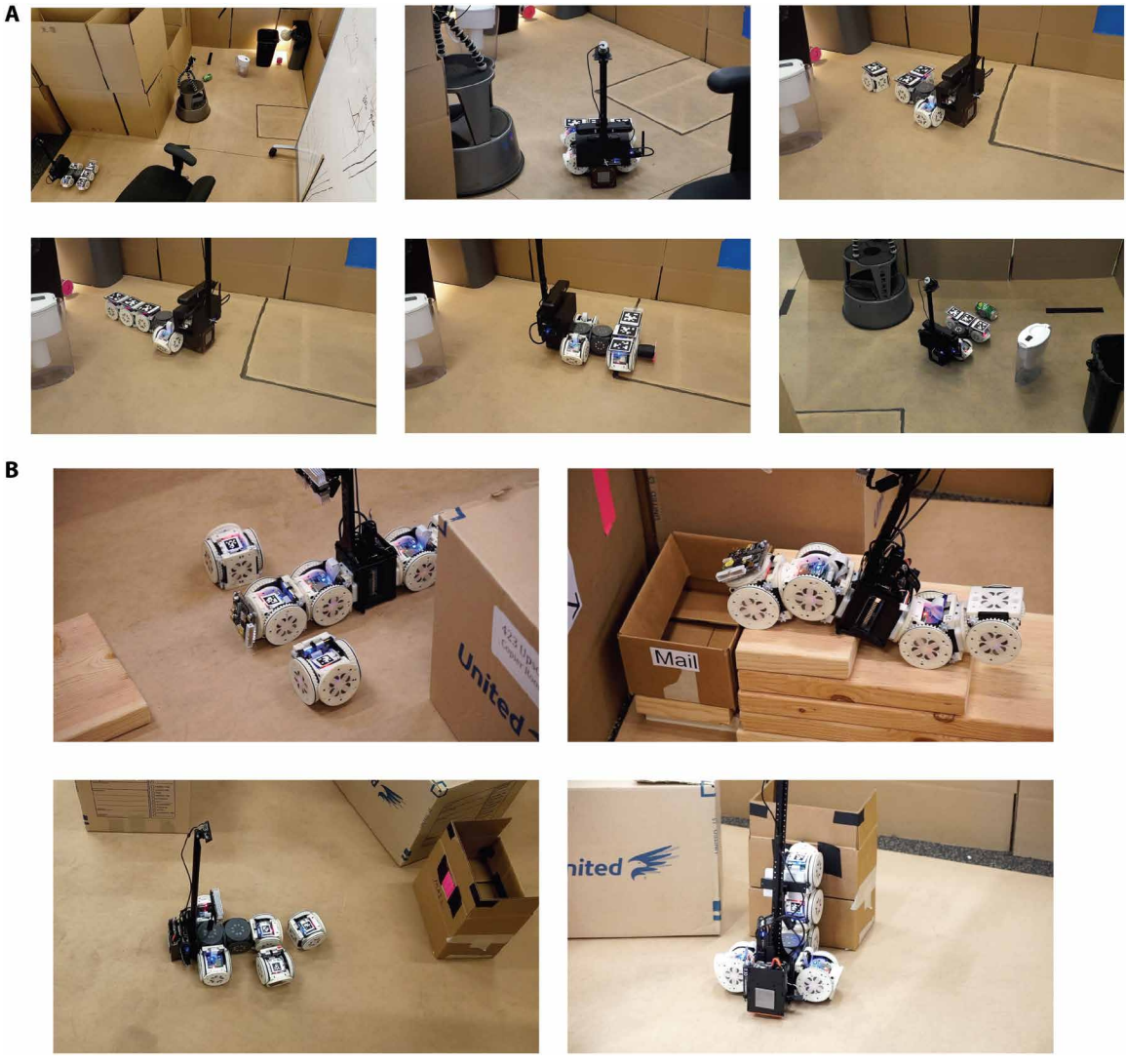


Fig. 2. Demonstrations I, II, and III. (**A**) Phases of demonstration I: environment (top left), exploration of environment (top middle), reconfiguration (top right), retrieving pink object (bottom left), delivering an object (bottom middle), and retrieving green object (bottom right). (**B**) (Top) Demonstration II: Reconfiguring to climb stairs (left) and successful circuit delivery (right). (Bottom) Demonstration III: Reconfiguring to place stamp (left) and successful stamp placement (right).

robot to use the "Snake" configuration to traverse the stairs. Using the 3D map and characterization of the environment surrounding the mail bin, the robot navigated to a point directly in front of the stairs, faced the bin, and reconfigured to the Snake configuration. The robot then executed the stair-climbing gait to reach the mail bin and dropped the circuit successfully. It then descended the stairs and reconfigured back to the Scorpion configuration to end the mission.

For demonstration III, the robot began in the Car configuration and could not see the package from its starting location. After a short period of exploration, the robot identified the pink-square marking the package. The pink square was unobstructed but was about 25 cm above the ground; the system correctly characterized this as the "high"-type environment and recognized that reconfiguration would be needed to reach up and place the stamp on the target. The robot navigated to a position directly in front of the package, reconfigured to the "Proboscis" configuration, and executed the "highReach" behavior to place the stamp on the target, completing its task.

All experiments were run with the same software architecture, same SMORES-EP modules, and same system described in this paper. The library of behaviors was extended with new entries as system abilities were added, and minor adjustments were made to motor speeds, SLAM parameters, and the low-level reconfiguration controller. In addition, demonstrations II and III used a newer, improved 3D sensor, and therefore a sensor driver different from that in demonstration I was used.

DISCUSSION

This paper presents a modular robot system that autonomously completed high-level tasks by reactively reconfiguring in response to its perceived environment and task requirements. Putting the entire system to the test in hardware demonstrations revealed several opportunities for future improvement. MSRRs are by their nature mechanically distributed and, as a result, lend themselves naturally to distributed planning, sensing, and control. Most past systems have

used entirely distributed frameworks (*3*–*5*, *17*, *18*, *21*). Our system was designed differently. It is distributed at the low level (hardware) but centralized at the high level (planning and perception), leveraging the advantages of both design paradigms.

The three scenarios in the demonstrations showcase a range of different ways SMORES-EP can interact with environments and objects: moving over flat ground, fitting into tight spaces, reaching up high, climbing over rough terrain, and manipulating objects. This broad range of functionality is only accessible to SMORES-EP by reconfiguring between different morphologies.

The high-level planner, environment characterization tools, and library worked together to allow tasks to be represented in a flexible and reactive manner. For example, at the high level, demonstrations II and III were the same task: deliver an object at a goal location. However, after characterizing the different environments (high in II, stairs in III), the system automatically determined that different configurations and behaviors were required to complete each task: the Proboscis to reach up high, and the Snake to climb the stairs. Similarly, in demonstration I, there was no high-level distinction between the green and pink objects—the robot was simply asked to retrieve all objects it found. The sensed environment once again dictated the choice of behavior: the simple problem (object in the open) was solved in a simple way (with the Car configuration), and the more difficult problem (object in tunnel) was solved in a more sophisticated way (by reconfiguring into the Proboscis). This level of sophistication in control and decision-making goes beyond the capabilities demonstrated by past systems with distributed architectures.

Centralized sensing and control during reconfiguration, provided by AprilTags and a centralized path planner, allowed our implementation to transform between configurations more rapidly than previous distributed systems. Each reconfiguration action (a module disconnecting, moving, and reattaching) takes about 1 min. In contrast, past systems that used distributed sensing and control required 5 to 15 min for single-reconfiguration actions (*3*–*5*), which would prohibit their use in the complex tasks and environments that our system demonstrated.

Through the hardware demonstrations performed with our system, we observed several challenges and opportunities for future improvement. All SMORES-EP body modules are identical and therefore interchangeable for the purposes of reconfiguration. However, the sensor module has a substantially different shape than a SMORES-EP body module, which introduces heterogeneity in a way that complicates motion planning and reconfiguration planning. Configurations and behaviors must be designed to provide the sensor module with an adequate view and to support its weight and elongated shape. Centralizing sensing also limits reconfiguration: modules can only drive independently in the vicinity of the sensor module, preventing the robot from operating as multiple disparate clusters.

Our high-level planner assumes that all underlying components are reliable and robust, so failure of a low-level component can cause the high-level planner to behave unexpectedly and result in failure of the entire task. Table 1 shows the causes of failure for 24 attempts of demonstration II (placing the stamp on the package). Nearly all failures were due to an error in one of the low-level components that the system relies on, with 42% of failure due to hardware errors and 38% due to failures in low-level software (object recognition, navigation, and environment characterization). This kind of cascading failure is a weakness of centralized, hierarchical systems: Distributed systems are often designed so that failure of a single unit can be compensated for by other units and does not result in global failure.

This lack of robustness presents a challenge, but steps can be taken to address it. Open-loop behaviors (such as stair climbing and reaching up to place the stamp) were vulnerable to small hardware errors and less robust against variations in the environment. For example, if the height of stairs in the actual environment is higher than the property value of the library entry, then the stair-climbing behavior is likely to fail. Closing the loop using sensing made exploration and reconfiguration significantly less vulnerable to error. Future systems could be made more robust by introducing more feedback from low-level components to high-level decision-making processes and by incorporating existing high-level failure-recovery frameworks (*24*). Distributed repair strategies could also be explored, to replace malfunctioning modules with nearby working ones on the fly (*25*).

To implement our perception characterization component, we assumed a limited set of environment types and implemented a simple characterization function to distinguish between them. This function does not generalize very well to completely unstructured environments and also is not very scalable. Thus, to expand the system to work well for more realistic environments and to distinguish between a large number of environment types, a more general characterization function should be implemented.

Table 1. Reasons for demonstration failure.

Failure reason	Number of times	Percentage (%)
Hardware issues	10	41.7
Navigation failure	3	12.5
Perception-related errors	6	25
Network issues	1	4.2
Human error	4	16.7

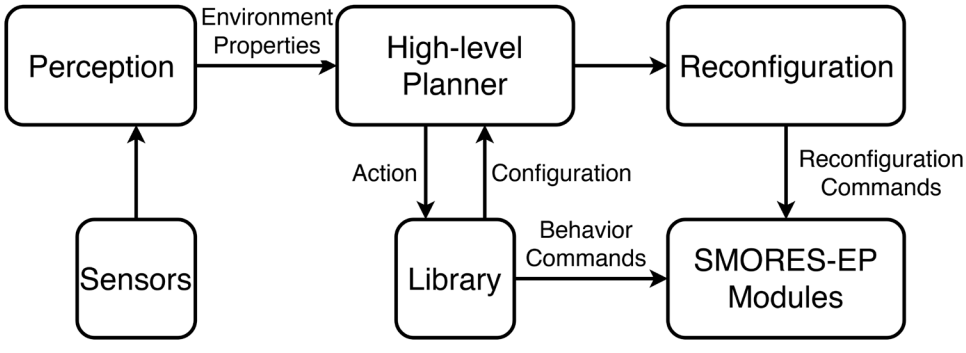


Fig. 3. System overview flowchart.

MATERIALS AND METHODS

The following sections discuss the role of each component within the general system architecture. Interprocess communication between the many software components in our implementation is provided by the Robot Operating System. Figure 3 gives a flowchart of the entire system. For more details of the implementation used in the demonstrations, see the Supplementary Materials.

Hardware

SMORES-EP modular robot

Each SMORES-EP module is the size of an 80-mm-wide cube and has four actuated joints, including two wheels that can be used for differential drive on flat ground (*22*, *26*). The modules are equipped with electropermanent (EP) magnets that allow any face of one module to connect to any face of another, allowing the robot to self-reconfigure. The magnetic faces can also be used to attach to objects made of ferromagnetic materials (e.g., steel). The EP magnets require very little energy to connect and disconnect and no energy to maintain their attachment force of 90 N (*22*).

Each module has an onboard battery, microcontroller, and WiFi chip to send and receive messages. In this work, clusters of SMORES-EP modules were controlled by a central computer running a Python program that sent WiFi commands to control the four DoF and magnets of each module. Wireless networking was provided by a standard off-the-shelf router, with a range of about 100 feet, and commands to a single module could be received at a rate of about 20 Hz. Battery life was about 1 hour (depending on motor, magnet, and radio usage).

Sensor module

SMORES-EP modules have no sensors that allow them to gather information about their environment. To enable autonomous operation, we introduced a sensor module that was designed to work with SMORES-EP (shown in Fig. 4B). The body of the sensor module is a 90 mm–by–70 mm–by–70 mm box with thin steel plates on its front and back that allow SMORES-EP modules to connect to it. Computation was provided by an UP computing board with an Intel Atom 1.92-GHz processor, 4-GB memory, and 64 GB of storage. A USB WiFi adapter provided network connectivity. A front-facing Orbecc Astra Mini camera provided RGB-D data, enabling the robot to explore and map its environment and to recognize objects of interest. A thin stem extended 40 cm above the body, supporting a downward-facing webcam. This camera provided a view of a 0.75 m–by–0.5 m area in front of the sensor module and was used to track AprilTag (*27*) fiducials for reconfiguration. A 7.4-V, 2200-mAh LiPo battery provided about 1 hour of running time.

A single sensor module carried by the cluster of SMORES-EP modules provided centralized sensing and computation. Centralizing sensing and computation has the advantage of facilitating control, task-related decision-making, and rapid reconfiguration but the disadvantage of introducing physical heterogeneity, making it more difficult to design configurations and behaviors. The shape of the sensor module could be altered by attaching lightweight cubes, which provided passive structure to which modules could connect. Cubes have the same 80-mm form factor as SMORES-EP modules, with magnets on all faces for attachment.

Perception and planning for information

Completing tasks in unknown environments requires the robot to explore, to gain information about its surroundings, and to use that information to inform actions and reconfiguration. Our system architecture included active perception components to perform SLAM, choose waypoints for exploration, and recognize objects and regions of interest. It also included a framework to characterize the environment in terms of robot capabilities, allowing the high-level planner to reactively reconfigure the robot to adapt to different environment types. Implementations of these tools should be selected to fit the MSRR system being used and types of environments expected to be encountered.

Environment characterization was done by using a discrete classifier (using the 3D occupancy grid of the environment as input) to distinguish between a discrete set of environment types corresponding to the library of robot configurations and gaits. To implement

Fig. 4. SMORES-EP module and sensor module. (**A**) SMORES-EP module. (**B**) Sensor module with labeled components. UP board and battery are inside the body.

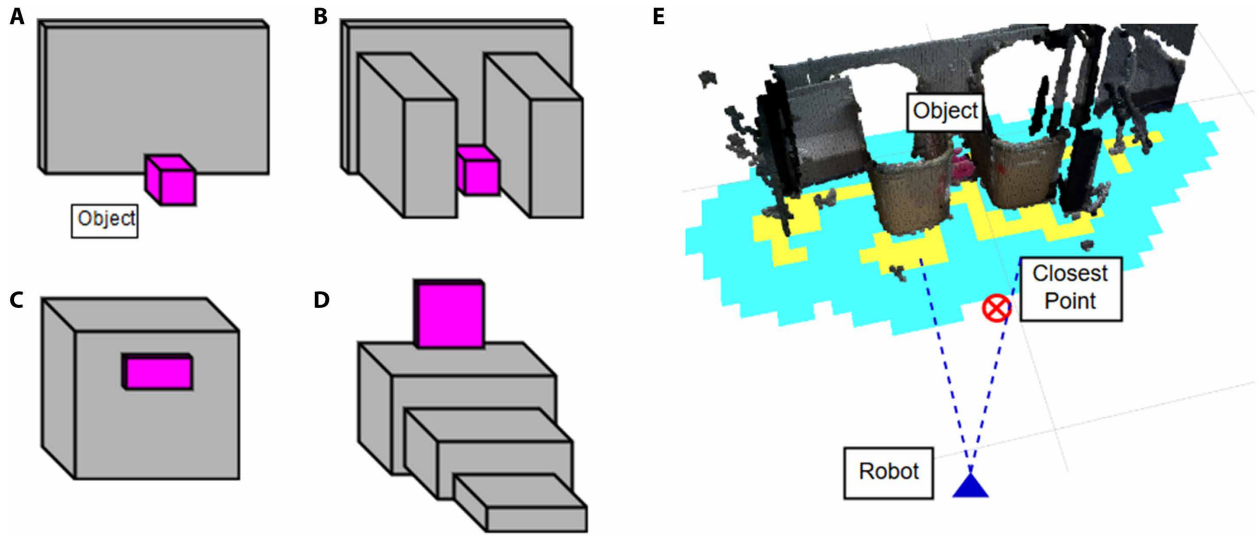


Fig. 5. Environment characterization. (**A**) Free. (**B**) Tunnel. (**C**) High. (**D**) Stairs. (**E**) An example of a tunnel environment characterization. Yellow grid cells are occupied; light blue cells are unreachable resulting from bloating obstacles.

Table 2. A library of robot behaviors.

Behavior properties	Environment types
Car	
PickUp	Free
Drop	Free
Drive	Free
Proboscis	
PickUp	Tunnel or free
Drop	Tunnel or free
HighReach	High
Scorpion	
Drive	Free
Snake	
ClimbUp	Stairs
ClimbDown	Stairs
Drop	Stairs or free

our system for a particular MSRR, the user must define the classification function to classify the desired types of environments. For our proof-of-concept hardware demonstrations, we assumed a simplified set of possible environment types around objects of interest. We assumed that the object of interest must be in one of four environment types shown in Fig. 5E: tunnel (the object is in a narrow corridor), stairs (the object is at the top of low stairs), high (the object is on a wall above the ground), and free (the object is on the ground with no obstacles around). Our implemented function performed characterization as follows: When the system recognized an object in the environment, the characterization function evaluated the 3D information in the object's surroundings. It created an occupancy grid around the object location and denoted all grid cells within a robot radius of obstacles as unreachable (illustrated in Fig. 5E). The algorithm then selected the closest reachable point to the object within 20° of the robot's line of sight to the object. If the distance from this point to the object was greater than a threshold value and the object was on the ground, then the function characterized the environment as a tunnel. If above the ground, then the function characterized the environment as a stairs environment. If the closest reachable point was under the threshold value, the system assigned a free or high environment characterization, depending on the height of the colored object.

On the basis of the environment characterization and target location, the function also returned a waypoint for the robot to position itself to perform its task (or to reconfigure, if necessary). In demonstration II, the environment characterization algorithm directed the robot to drive to a waypoint at the base of the stairs, which was the best place for the robot to reconfigure and begin climbing the stairs.

Our implementation for other components of the perception architecture used previous work and open-source algorithms. The RGB-D SLAM software package RTAB-MAP (*27*) provides mapping and robot pose. The system incrementally built a 3D map of the environment and stored the map in an efficient octree-based volumetric map using Octomap (*28*). The Next Best View algorithm by Daudelin *et al.* (*29*) enabled the system to explore unknown environments by using the current volumetric map of the environment to estimate the next reachable sensor viewpoint that will observe the largest volume of undiscovered portions of objects (the Next Best View). In the example object delivery task, the system began the task by iteratively navigating to these Next Best View waypoints to explore objects in the environment until discovering the drop-off zone.

To identify objects of interest in the task (such as the drop-off zone), we implemented our system by using color detection and tracking. The system recognized colored objects using opensource software called CMVision and tracked them in 3D using depth information from the onboard RGB-D sensor. Although we implement object recognition by color, more sophisticated methods could be used instead, under the same system architecture.

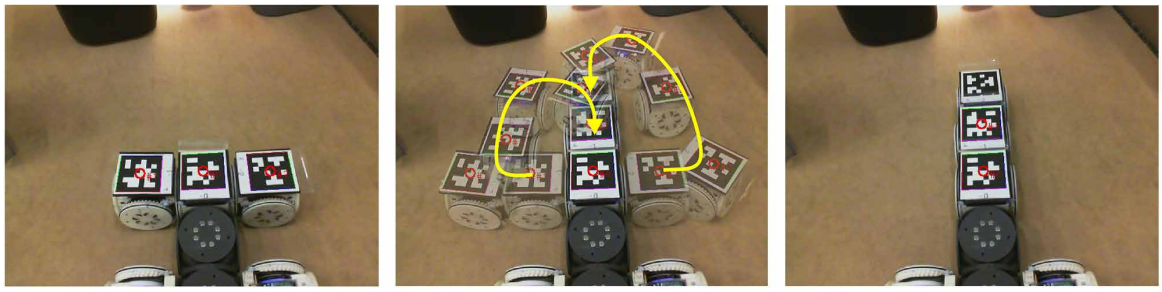


Fig. 6. Module movement during reconfiguration. (**Left**) Initial configuration (Car). (**Middle**) Module movement, using AprilTags for localization. (**Right**) Final configuration (Proboscis).

A

Specification 1 Drop an object in the mailbox

do **explore** if and only if the robot is not sensing **mailBox**

do **driveToMailBox** if and only if the robot is sensing (**mailBox** and not **arrived**)

do **drop** if and only if the robot is sensing (**arrived** and **mailBox**)

B

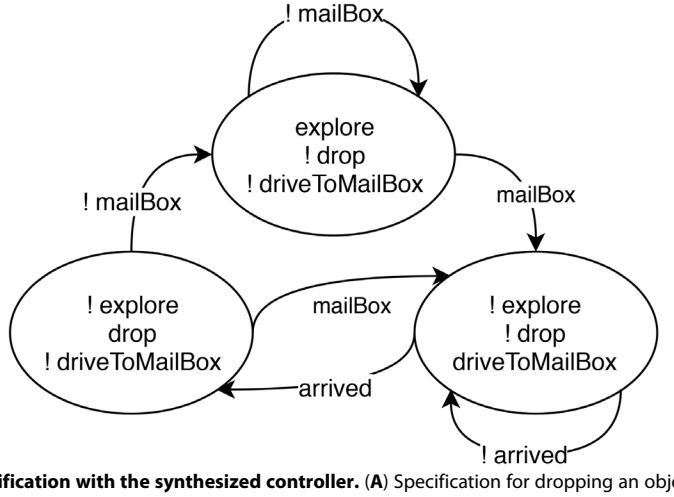


Fig. 7. A task specification with the synthesized controller. (**A**) Specification for dropping an object in the mailbox. (**B**) Synthesized controller. A proposition with an exclamation point has a value of false and true otherwise.

Library of configurations and behaviors

A library-based framework was used to organize user-designed configurations and behaviors for the SMORES-EP robot. Users can create designs for modular robot using our simulation tool and save designs to a library. Configurations and behaviors are labeled with properties, which are high-level descriptions of behaviors. Specifically, environment properties specify the appropriate environment that the behavior is designed for (e.g., a three-module-high ledge), and behavior properties specify the capabilities of the behavior (e.g., climb). Therefore, in this framework, a library entry is defined as $l = (C, B_c, P_b, P_e)$, where C is a robot configuration, B_c is the behavior of C, P_b is a set of behavior properties describing the capabilities of the behavior, and P_e is a set of environment properties. The high-level planner can then select appropriate configurations and behaviors based on given task specifications and environment information from the perception subsystem to accomplish tasks. In demonstration II, the task specifications required the robot to deliver an object to a mailbox, and the environment characterization algorithm reported that the mailbox was in a stairs-type environment. Then, the high-level planner searched the design library for a configuration and a behavior that were able to climb stairs with the object. Each entry is capable of controlling the robot to perform some actions in a specific environment. In demonstration II, we showed a library entry that controlled the robot to "climb" a stairs-type environment.

To aid users in designing configurations and behaviors, we created a design tool called VSPARC and made it available online (*23*). Users can use VSPARC to create, simulate, and test designs in various environment scenarios with an included physics engine. Moreover, users can save their designs of configurations (connectivity among modules) and behaviors (joint commands for each module) on our server and share them with other users. All behaviors designed in VSPARC can be used to directly control the SMORES-EP robot system to perform the same action. Table 2 lists 10 entries for four different configurations that are used in this work.

Reconfiguration

When the high-level planner decides to use a new configuration during a task, the robot must reconfigure. We have implemented tools for mobile reconfiguration with SMORES-EP, taking advantage of the fact that individual modules can drive on flat surfaces. As discussed in the "Hardware" section, a downward-facing camera on the sensor module provides a view of a 0.75 m–by–0.5 m area on the ground in front of the sensor module. Within this area, the localization system provides pose for any module equipped with an AprilTag marker to perform reconfiguration. Given an initial configuration and a goal configuration, the reconfiguration controller commands a set of modules to disconnect, move, and reconnect to form the new topology of the goal configuration. Currently, reconfiguration plans from one configuration to another are created manually and stored in the library. However, the framework can work with existing assembly planning algorithms (*30*, *31*) to generate reconfiguration plans automatically. Figure 6 shows reconfiguration from Car to Proboscis during demonstration I.

High-level planner

In our architecture, the high-level planner subsystem provides a framework for users to specify robot tasks using a formal language and generates a centralized controller that directs robot motion and actions based on environment information. Our implementation is based on the Linear Temporal Logic MissiOn Planning (LTLMoP) toolkit, which automatically generates robot controllers from user-specified high-level instructions using synthesis (*32*, *33*). In LTLMoP, users describe the desired robot tasks with high-level specifications over a set of Boolean variables and provide mapping from each variable to a robot sensing or action function. In our framework, users do not specify the exact configurations and behaviors used to complete tasks but, rather, specify constraints and desired outcomes for each Boolean variable using properties from the robot design library. LTLMoP automatically converts the specification to logic formulas, which are then used to synthesize a robot controller that satisfies the given tasks (if one exists). The high-level planner determines configurations and behaviors associated with each Boolean variable based on properties specified by users and continually executes the synthesized robot controller to react to the sensed environment.

Consider the robot task in demonstration II: The user indicates that the robot should explore until it locates the mailbox and then drop the object off. In addition, the user describes desired robot actions in terms of properties from the library. The high-level planner then generates a discrete robot controller that satisfies the given specifications, as shown in Fig. 7. If no controller can be found or no appropriate library entries can implement the controller, users are advised to change the task specifications or add more behaviors to the design library.

The high-level planner coordinates each component of the system to control our MSRR to achieve complex tasks. At the system level, the sensing components gather and process environment information for the high-level planner, which then takes actions based on the given robot tasks by invoking appropriate low-level behaviors. In demonstration II, when the robot is asked to deliver the object, the perception subsystem informs the robot that the mailbox is in a stairs-type environment. Therefore, the robot self-reconfigures to the Snake configuration to climb the stairs and deliver the object.

SUPPLEMENTARY MATERIALS

robotics.sciencemag.org/cgi/content/full/3/23/eaat4983/DC1
Text
Movie S1. Video of demonstrations I, II, and III.

REFERENCES AND NOTES

1. M. Yim, Locomotion with a unit-modular reconfigurable robot, thesis, Stanford University (1994).
2. M. Yim, W. M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, G. Chirikjian, Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robot. Automat. Mag.* **14**, 43–52 (2007).
3. M. Yim, B. Shirmohammadi, J. Sastra, M. Park, M. Dugan, C. J. Taylor, Towards robotic self-reassembly after explosion, in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2007), pp. 2767–2772.
4. M. Rubenstein, K. Payne, P. Will, W. M. Shen, Docking among independent and autonomous CONRO self-reconfigurable robots, in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04* (ICRA, 2004) p. 2877.
5. S. Murata, K. Kakomura, H. Kurokawa, Docking experiments of a modular robot by visual feedback, in *IEEE International Conference on Intelligent Robots and Systems* (IEEE, 2006), pp. 625–630.
6. J. Paulos, N. Eckenstein, T. Tosun, J. Seo, J. Davey, J. Greco, V. Kumar, M. Yim, Automated self-assembly of large maritime structures by a team of robotic boats, in *IEEE Transactions on Automation Science and Engineering* (IEEE, 2015), pp. 1–11.
7. G. Jing, T. Tosun, M. Yim, H. Kress-Gazit, An end-to-end system for accomplishing tasks with modular robots, in *Proceedings of Robotics: Science and Systems XII* (2016).
8. T. Fukuda, Y. Kawauchi, Cellular robotic system (CEBOT) as one of the realization of self-organizing intelligent universal manipulator, in *Proceedings in 1990 IEEE International Conference on Robotics and Automation* (IEEE, 1990), pp. 662–667.
9. S. Murata, H. Kurokawa, S. Kokaji, Self-assembling machine, in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation* (IEEE, 1994), pp. 441–448.
10. G. S. Chirikjian, Kinematics of a metamorphic robotic system, in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation* (IEEE, 1994), pp. 449–455.
11. A. Dutta, P. Dasgupta, C. Nelson, Distributed configuration formation with modular robots using (sub) graph isomorphism-based approach. *Autonom. Robot.* 1–21 (2018).
12. G. G. Ryland, H. H. Cheng, Design of imobot, an intelligent reconfigurable mobile robot with novel locomotion, in *2010 IEEE International Conference on Robotics and Automation* (IEEE, 2010), pp. 60–65.
13. K. C. Wolfe, M. S. Moses, M. D. Kutzer, G. S. Chirikjian, M3 express: A low-cost independently-mobile reconfigurable modular robot, in *2012 IEEE International Conference on Robotics and Automation* (IEEE, 2012), pp. 2704–2710.
14. J. W. Romanishin, K. Gilpin, S. Claici, D. Rus, 3d m-blocks: Self-reconfiguring robots capable of locomotion via pivoting in three dimensions, in *2015 IEEE International Conference on Robotics and Automation* (IEEE, 2015), pp. 1925–1932.
15. Y. Mantzouratos, T. Tosun, S. Khanna, M. Yim, On embeddability of modular robot designs, in *IEEE International Conference on Robotics and Automation* (IEEE, 2015), pp. 1911–1918.
16. R. Grabowski, L. E. Navarro-Serment, C. J. J. Paredis, P. K. Khosla, Heterogeneous teams of modular robots for mapping and exploration. *Autonom. Robot.* **8**, 293–308 (2000).
17. M. Dorigo, E. Tuci, R. Groß, V. Trianni, T. H. Labella, S. Nouyan, C. Ampatzis, J.-L. Deneubourg, G. Baldassarre, S. Nolfi, F. Mondada, D. Floreano, L. M. Gambardella, The SWARM-BOTS project, in *Lecture Notes in Computer Science*, E. Sahin, W. M. Spears, Eds. (Springer, 2005), pp. 31–44.
18. F. Mondada, L. M. Gambardella, D. Floreano, M. Dorigo, The cooperation of swarm-bots: Physical interactions in collective robotics. *IEEE Robot. Autom. Mag.* **12**, 21–28 (2005).
19. M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Ocampo, A. L. Christensen, A. Decugniere, G. Di Caro, F. Ducatelle, E. Ferrante, A. Foster, J. M. Gonzales, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. Montes de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Retomaz, J. Roberts, V. Sperati, Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robot. Autom. Mag.* **20**, 60–71 (2013).
20. R. Gross, M. Bonani, F. Mondada, M. Dorigo, Autonomous self-assembly in swarm-bots. *IEEE Transact. Robot.* **22**, 1115–1130 (2006).
21. R. O'Grady, R. Groß, A. L. Christensen, M. Dorigo, Self-assembly strategies in a group of autonomous mobile robots. *Autonom. Robot.* **28**, 439–455 (2010).
22. T. Tosun, J. Davey, C. Liu, M. Yim, Design and characterization of the EP-Face connector, in *2016 IEEE/RSJ Internatonal Conference on Intelligent Robots and Systems* (IROS, 2016), pp. 45–51.
23. G. Jing, T. Tosun, M. Yim, H. Kress-Gazit, Accomplishing high-level tasks with modular robots. *Autonom. Robot.* **42**, 1337–1354 (2018).
24. S. Maniatopoulos, P. Schillinger, V. Pong, D. C. Conner, H. Kress-Gazit, Reactive high-level behavior synthesis for an atlas humanoid robot, in *2016 IEEE International Conference on Robotics and Automation* (ICRA, 2016), pp. 4192–4199.
25. K. Tomita, S. Murata, H. Kurokawa, E. Yoshida, S. Kokaji, Self-assembly and self-repair method for a distributed mechanical system. *IEEE Transact. Robot. Autom.* **15**, 1035–1045 (1999).
26. T. Tosun, D. Edgar, C. Liu, T. Tsabedze, M. Yim, Paintpots: Low cost, accurate, highly customizable potentiometers for position sensing, in *2017 IEEE International Conference on Robotics and Automation* (IEEE, 2017).
27. E. Olson, Apriltag: A robust and flexible visual fiducial system, in *2011 IEEE International Conference on Robotics and Automation* (IEEE, 2011), pp. 3400–3407.
28. A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonom. Robot.* **34**, 189–206 (2013).
29. J. Daudelin, M. Campbell, An adaptable, probabilistic, next-best view algorithm for reconstruction of unknown 3-D objects. *IEEE Robot. Autom. Lett.* **2**, 1540–1547 (2017).
30. J. Werfel, D. Ingber, R. Nagpal, Collective construction of environmentally-adaptive structures, in *IEEE International Conference on Intelligent Robots and Systems* (IROS, 2007), pp. 2345–2352.

31. J. Seo, M. Yim, V. Kumar, Assembly planning for planar structures of a brick wall pattern with rectangular modular robots. in *2013 IEEE International Conference on Automation Science and Engineering* (CASE, 2013), pp. 1016–1021.
32. C. Finucane, G. Jing, H. Kress-Gazit, Ltlmop: Experimenting with language, temporal logic and robot control, in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18–22, 2010, Taipei, Taiwan* (2010), pp. 1988–1993.
33. H. Kress-Gazit, G. E. Fainekos, G. J. Pappas, Temporal-logic-based reactive mission and motion planning. *IEEE Trans. Robotics* **25**, 1370–1381 (2009).

Funding: This work was funded by NSF grant numbers CNS-1329620 and CNS-1329692.
Author contributions: All authors contributed to conceptualization of the study, writing, and reviewing the original manuscript, and preparing the figures. J.D., T.T., and G.J. developed the software and curated the data. G.J., T.T., H.K.-G., and M.Y. administered the project. **Competing interests:** Since the paper was submitted, J.D. has accepted a position at Toyota Research Institute, G.J. has been hired by Neocis Inc., and T.T. has been hired by Samsung Research America. The other authors declare that they have no competing financial interests. **Data and materials availability:** All data needed to support the conclusions of this manuscript are included in the main text or supplementary materials. See www.vsparc.org for software modules.

Submitted 14 March 2018
Accepted 2 October 2018
Published 31 October 2018
10.1126/scirobotics.aat4983

Citation: J. Daudelin, G. Jing, T. Tosun, M. Yim, H. Kress-Gazit, M. Campbell, An integrated system for perception-driven autonomy with modular robots. *Sci. Robot.* **3**, eaat4983 (2018).

Science Robotics

An integrated system for perception-driven autonomy with modular robots

Jonathan Daudelin, Gangyuan Jing, Tarik Tosun, Mark Yim, Hadas Kress-Gazit, and Mark Campbell

Sci. Robot. **3** (23), eaat4983. DOI: 10.1126/scirobotics.aat4983

Science Robotics (ISSN 2470-9476) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.