An Integrated System for Autonomously Performing Perception-Driven Tasks with Modular Robots

Mystery Authors

***Abstract*—We present a fully autonomous modular robot system that can perform complex, high-level tasks in an unknown environment without external sensing or control. The validity of the system is demonstrated in a real-world experiment. The physical robot is composed of modules that support multiple robot configurations. An onboard 3D sensor provides information about the environment, which is used to perform SLAM in the unknown environment and inform exploration and feedback control. No external sensors, pose providers, or beacons are used. A centralized planning algorithm uses the information from the environment and the desired high-level task description to synthesize low-level controllers to perform locomotion, reconfiguration, and special actions. A novel, centralized, self-reconfiguration method is used to change robot configurations when ~~desired.~~ ~~To the authors' knowledge,~~ the proposed work comprises the first modular robot system that uses perception-driven reconfiguration to intelligently adapt to an *a priori* unknown environment to perform complex tasks.**

I. Introduction

Modular self-reconfigurable robot (MSRR) systems are composed of a number of simple repeated robot elements (called *modules*) that connect together to form larger robotic structures. These systems can *self-reconfigure*, changing their shape (*i.e.* the connective structure of the modules) to meet the needs of the task at hand. In principal, these systems can address a wide variety of tasks by transforming into a wide variety of morphologies.

Over the past three decades, dozens of modular robot systems have been built [13]. Existing literature provides ample evidence of MSRR systems reconfiguring and assuming interesting morphologies, as well as methods for programming, controlling, and simulating modular robots [15, 5, 12].

These capabilities are impressive, and each represents a significant research accomplishment in its own right. However, in order to truly live up to their promise of flexible capability in the real world, MSRR systems must demonstrate autonomy: moving, navigating, interacting with objects, and self-reconfiguring, all in unknown environments and without external localization or control.

We provide a system capable of *autonomously* ~~solving~~ *high-level tasks* in *unknown environments* using *modular self-reconfigurable robots*. A *high-level task* is specified in terms of general objectives, and requires some decision-making regarding the specific way in which the task will be solved. The environment is *unknown*, meaning that the robot does not have a map of the environment or obstacles before the task begins. To our knowledge, this paper represents the first example of a truly autonomous MSRR system accomplishing these kinds of tasks.

The system we present provides four major ~~tools:~~

1) **Hardware:** The SMORES-EP Modular robot, and sensing hardware designed to work with SMORES-EP.
2) **High-Level Task Description:** A framework for specifying high-level tasks in terms of state machines, and for abstracting modular robots.
3) **Perception and Environment Characterization:** Tools for SLAM, navigation, and obstacle avoidance with modular robots, as well as tools that parse sensor information into actionable conclusions relevant to high-level tasks.
4) **Reconfiguration:** Software and hardware tools enabling robust autonomous reconfiguration with SMORES-EP.

Through hardware experiments, we demonstrate that this system is capable of autonomously completing high-level object-retrieval tasks in unknown environments.

A. System Overview

Here we provide a brief overview of the entire system. Figure 1 shows a flowchart which serves as a visual companion to this section.

Tasks are specified in a high-level mission planner using Linear Temporal Logic (LTL). The user does not specify a specific sequence of actions that should be completed, but rather expresses the task in the form of logical statements which map conditions to actions. Additionally, because a modular robot system is being used to complete the task, the task definitions do not directly specify which configuration (connected robot shape) should be used to execute each action. Instead, actions are specified in an abstract sense, for example "If you encounter a pink object, pick it up." When a pink object is encountered, the planner will evaluate the environment surrounding the pink object and decide whether the current configuration of the robot is capable of executing a "pick up" action. If it is not, the planner may decide reconfigure to another configuration which is capable of picking up the object under the current conditions. Section VII describes our high-level mission planner in detail.

Our system is built around the SMORES-EP modular robot system, described in detail in Section III. Because SMORES-EP is capable of changing its physical morphology through self-reconfiguration, we must provide appropriate high-level abstractions of the robot that allows our high-level planner to address tasks. These abstractions are described in Section V.

The modularity of our robot makes sensing challenging. We have extended the existing SMORES-EP robot system by developing a sensor module which can be carried as part

of a cluster of SMORES-EP modules to provide sensing and localization information, described in detail in Section III-B.

Our system is intended to address tasks where the robot is required to explore an *a priori* unknown environment and react to what it encounters. To that end, we provide a suite of software tools for sensing and perception. Our perception tools allow a SMORES-EP robot to perform SLAM, navigation, and obstacle avoidance using the sensor module. This software has been specifically optimized to work with data from the onbord XTion Pro depth camera, and to run efficiently using the limited computation power of the onboard UPboard computer. Additionally, a novel next-best-view algorithm automatically selects waypoints in the environment to allow the robot to maximize information gain as it explores. These tools are discussed in detail in Section IV-A

While exploring, the robot must recognize and react to objects and features of the environment. We provide environment characterization tools that parse raw sensor information into actionable conclusions relevant to high-level tasks. More specifically, we provide classifiers that can identify objects based on color, and classifiers that characterize different kinds of environments (such as ledges and tunnels) based on 3D depth information. These are described in Section IV-B.

Finally, we provide software and hardware tools enabling rapid, robust, autonomous reconfiguration with SMORES-EP. Our system is capable of completing reconfiguration tasks without any external localization, by using AprilTags for reconfiguration. The system features an overhead camera carried along with the robot as part of the sensor module, which is a novel localization modality for reconfiguration. Our reconfiguration system also relies on elements of our perception algorithms, which provide goal pose information for docking. Our reconfiguration tools are described in Section VI.

After presenting the complete system, we demonstrate how these individual components form a complete system capable of autonomously solving high-level tasks in unknown environments with modular robots. In Section VIII, we show how our system can be used to complete a realistic object retrieval task in an unknown environment with full autonomy. Finally, in Section IX, we evaluate the success of our system in achieving its goals, and conclude.

II. Related Work

A. Autonomous Self-Reconfiguration

Autonomous reconfiguration has been demonstrated with several modular robot systems. CKbot, Conro, and MTRAN have all demonstrated the ability to join disconnected clusters of modules together [15, 10, 8]. In order to align, Conro uses infra-red sensors on the docking faces of the modules, while CKBot and MTRAN use a separate sensor module on each cluster. In all cases, individual clusters locate and servo towards each other until they are close enough to dock.

While these proof-of-concept experiments demonstrate the ability to reconfigure, these capabilities have not been used as part of a larger system to complete tasks. The experiments do not include any planning or sequencing of multiple reconfiguration actions in order to create a goal structure appropriate for a task. Additionally, because these are all chain-type modular robots, individual modules are not able to locomote on their own, and mobile clusters of modules are limited to slow crawling gaits. Consequently, reconfiguration is very time consuming, with a single connection requiring 5-15 minutes.

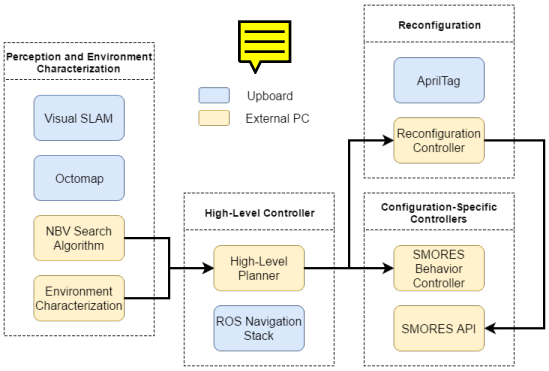


Fig. 1: System Overview Flowchart

Other work has focused on reconfiguration planning, but not autonomous reconfiguration. Paulos et al. present a system in which self-reconfigurable modular boats self-assemble into functional floating structures, such as a bridge [9]. Like the SMORES-EP modules used in this paper, individual boat modules are able to move about the pool, allowing for rapid reconfiguration. However, external localization is provided by an overhead AprilTag system.

To our knowledge, this paper represents the first examples of a MSRR system autonomously making the decision to reconfigure in response to its sensed environment, and then actually reconfiguring in order to complete its task. In [2], hand-bot and foot-bot elements of the swarmanoid system connect and disconnect in order to complete a book-retrieval task, but the decision to take this action is not made autonomously by the robot in response to sensed environment conditions.

B. Modular Robots Completing Tasks

Modular robots have long been regarded as having the potential to make impact in unknown environments (such as search and rescue scenarios), because self-reconfiguration theoretically gives them the flexibility to respond to whatever they encounter [13, 14]. However, examples of MSRR actually operating in unknown environments are very limited. To our knowledge, no MSRR system has been used for SLAM. We believe our system has more autonomy in an unknown environment than any existing modular robot system, and represents an important step toward the application of MSRR in the real world.

MSRR systems have demonstrated the ability to accomplish low-level tasks such as various modes of locomotion [12]. Recently work includes a system which integrate many low-

level capabilities of a MSRR system in a design library, and accomplishes high-level user-specified tasks by synthesizing elements of the library into a reactive state-machine [5]. While this system demonstrates autonomy with respect to task-related decision making, it is designed to operate in a fully known environment with external sensing.

There is work on mapping with swarm robot systems. The Millibot system has demonstrated the ability to map a partially unknown environment when operating as a swarm [3]. The autonomy of the Millibot swarm is limited: a human operator makes all high-level decisions, and is responsible for navigation using a GUI. Certain members of the swarm are designated as "beacons," and have known locations, making the environment only partially unknown.

The swarm-bots are a MSRR system that has been applied in exploration [1] and collective manipulation [7] scenarios. Like the Millibots, exploration is demonstrated in partially unknown environments, with some members of the swarm acting as "beacons" with known location. In a collective manipulation task, the swarm-bots have limited autonomy, with a human operator specifying the location of the manipulation target and the global sequence of manipulation actions. The swarmanoid project (successor to the swarm-bots), moves a step beyond this capability, using a heterogeneous swarm of ground and flying robots (called "hand-", "foot-", and "eye-" bots) to perform exploration and object retrieval tasks in unknown environments [2]. However, there is no reconfiguration or other transformation of the robots to adapt to unknown features of the environment. Although the location of the object to be retrieved is unknown, the method for retrieving the object is known and constant. We utilize the self-reconfiguration capability of MSRR systems to take autonomy a step further. The system uses perception of the environment to drive not only exploration but also the choice of robot configuration. This allows the robot to adapt its abilities to surmount challenges arising from *a priori* unknown features in the environment.

III. Hardware

A. SMORES-EP Modular Robot

Our system is built around the SMORES-EP robot, but could easily be adapted to work with other hardware platforms. In this section, we provide a brief introduction to the technical capabilities of SMORES-EP.

Each module is about the size of an *80mm cube*, and has four actuated DoF - three continuously rotating faces (left, right, and pan) and one central hinge (tilt) with a 180° range of motion (Fig. 2). The DoF marked left, right, and tilt have axes of rotation that are parallel and coincident. A single module can use its left and right wheels to drive around as a two-wheel differential drive robot. All four faces of the SMORES-EP module have electro-permanent (EP) magnets that serve as a high-strength, low-energy connector for self-reconfiguration [11]. Any face of one module can connect to any face of another.

The magnetic connectors can also attach to objects made of ferromagnetic materials (such as steel). By taking advantage of this capability, SMORES-EP modules can use their magnets to attract, lift, and carry metal objects. Provided the attachment surface is flat and smooth, the attachment force between a SMORES-EP face and a strongly ferromagnetic object can be as high as 90N [11].

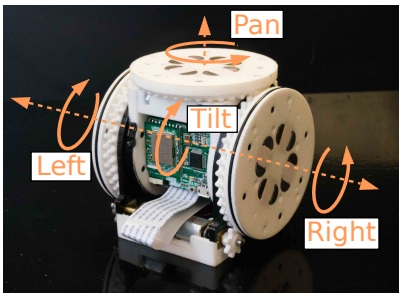


Fig. 2: SMORES-EP module

Each module has an onboard battery, microcontroller, and 802.11b wireless module to send and receive UDP packets. In this work, clusters of SMORES modules were controlled by a central computer running a Python program that sends wireless commands to control the four DoF and magnets of each module. Battery life is about one hour (depending on motor, magnet, and radio usage), and commands to a single module can be received at a rate of about 20hz. Wireless networking was provided by a standard off-the-shelf router, with a range of about 100 feet.

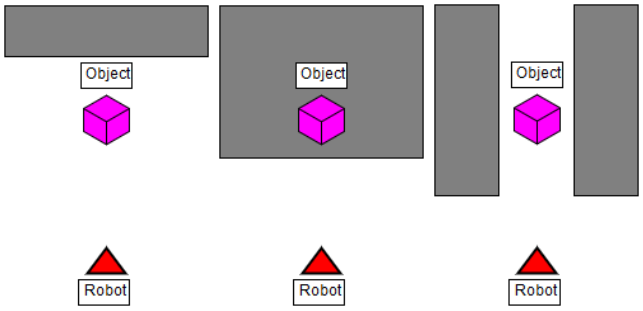
B. Sensor Module

In MSRR systems, sensing and processing capabilities of individual modules are severely limited by the size and weight constraints of the module form factor. Therefore, the proposed system uses a special sensor module dedicated to sensing and processing equipment, shown in Figure 3. The module has no actuation capabilities and is larger than SMORES-EP modules. It is equipped with a front-facing Xtion Pro Live RGB-D sensor which enables the robot to explore and map the environment and recognize objects of interest. A high, downward-facing HD webcam is included to provide a view of the robot itself, which is used for self-reconfiguration. Finally, an UP computing board provides high performance I/O and processing capability in a small form factor. The UP Board used in the proposed system has an Intel Atom 1.92 GHz processor, 4 GB memory, and a 64 GB hard drive. It is network connected via 802.11 wifi. A battery provides power to the Up Board with a lifetime of about 1.5 hours.

IV. Perception and Environment Characterization

Since the proposed system performs tasks in unknown environments and conditions, a robust suite of perception algorithms is required to inform control and decision-making. The robot must have the ability to explore and build a map of its environment while avoiding obstacles and tracking its pose. The system must be able to recognize objects and regions

Fig. 3: Sensor module.



(a) **"free"** environment (b) **"ledge"** environment (c) **"tunnel"** environment

Fig. 4: Environment characterization types for object retrieval.

of interest related to the desired task. Finally, the system must characterize the environment in terms of configuration capabilities. Features in the environment may restrict which robot configurations can viably perform parts of the high-level task, such as retrieving an object or navigating to a waypoint. The system needs to recognize these features to be able to intelligently choose the appropriate robot configuration for performing the task.

A. Perception

To facilitate the use of open source software and enable networking between components, the proposed system is built in a ROS framework.[1] Robot pose is provided by a RGB-D SLAM software package called RTAB-MAP[6]. A 3D map of the environment is incrementally built and stored in an efficient octree-based volumetric map using Octomap[4]. Task-related objects and regions of interest in the environment are given distinctive colors. These colors are recognized using color recognition software[2] and tracked in 3D using depth information from the Xtion sensor.[3]

The system must explore the *a priori* unknown environment to search for task-related objects and zones. The system performs this exploration in an intelligent manner using a new next best view planner for object exploration created by two of the authors.[4]. The exploration algorithm uses the current volumetric map of the environment and estimates the next reachable sensor viewpoint that will observe the largest volume of undiscovered portions of objects. It also estimates the amount of information (in an entropy-reduction sense) that will be gained from a sensor measurement taken at that viewpoint. To integrate into the proposed system, the exploration algorithm is offered as a service that can be queried by the high-level planner when desired. A volumetric map of the environment and the reachable space of sensor viewpoints in that map must be provided to the algorithm. Note that the reachable space is dependent on the locomotion capabilities of the specific robot configuration. The algorithm then computes and returns the next best view for the sensor, which the high-level planner can then set as a navigation waypoint.

B. Environment Characterization

In order to intelligently choose appropriate configurations when performing tasks, the robot must perceive and characterize its environment into a discrete set of environment types that correspond ~~with~~ configuration capabilities. The proposed system includes a perception component that discretely characterizes the environment for the purpose of retrieving an object. This characterization can then be used by the high-level planner to determine the appropriate configuration and gait for successful object retrieval. The 3 environment characterizations are shown in Figure 4. The environment surrounding objects to be retrieved is assumed to fall under one of these types. If the object's position is higher than a threshold height, the environment is characterized as the "ledge" environment. This requires a configuration/gait that can reach up to the top of the ledge to retrieve the object. If the object is on the ground, the remaining two characterizations are distinguished using the following method, illustrated in Figure 5. An occupancy grid is created around the object, and all grid cells within a robot radius of obstacles are denoted unreachable (light blue in Figure 5). The closest reachable point to the object within 20^o of the robot's line of sight to the object is selected. If the distance from this point to the object is greater than a threshold value, the environment is characterized as "tunnel". To retrieve the object in this type of environment, a configuration with a long, thin arm is required to reach between the surrounding obstacles to retrieve the object. If the distance to the object is under the threshold, the environment is characterized as "free". This means no special configurations

[1] http://www.ros.org
[2] CMVision: http://www.cs.cmu.edu/~jbruce/cmvision/
[3] Lucas Coelho Figueiredo: https://github.com/lucascoelho91/ballFollower
[4] This work has been newly accepted for journal publication in 2017

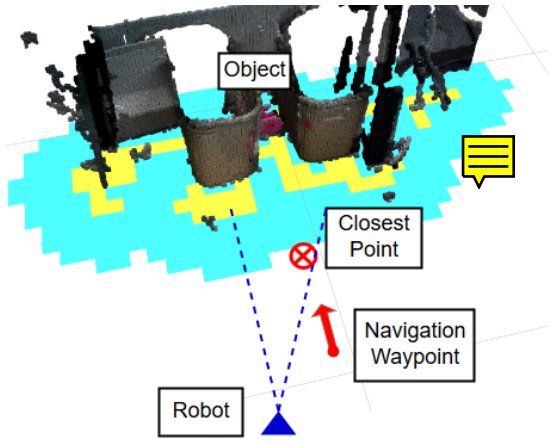


Fig. 5: **Tunnel** environment characterization method.

are required to reach the object. If the environment type is "tunnel", the algorithm also selects a navigation waypoint at which the robot can reconfigure before retrieving the object. This waypoint is calculated by extending the closest point farther away from the object to be retrieved (red arrow in Figure 5). This information is passed to the high-level planner for use in controller synthesis.

V. Configuration-Specific Controllers

Modular robots can change their configurations to gain new capabilities and accomplish various tasks. Since the configuration affects the dynamics of the modular robots, configuration-specific controllers are required for commanding modular robots. In this work, we utilize a library-driven system to create, organize and execute modular robot behaviors based on the robot configuration, the runtime environment and the robot behavior properties. A modular robot behavior library consists of a collection of different library entries. A library entry is defined as $l = (C, B_C, P_e, P_b)$ where:

- C specifies the configuration of the robot
- B is the controller that commands the robot to perform a specific behavior
- P_e holds information about the environment where this library entry is suitable
- P_b describes the properties of the robot behavior

In order to control the robot to perform desired behaviors in the *a priori* unknown environment, we employ a property-driven method to intelligently choose the library entry from the modular robot behavior library. First the desired behavior is specified as a set of behavior properties P'_b. We characterize the current environment as described in Section IV-B to a set of environment properties P'_e. Then we can find a set of library entries that match with P'_e and P'_b. If a library entry requires a different configuration from the current robot configuration, a reconfiguration process can be performed as described in Section VI. To minimize complexity in runtime, we bias towards the library entries that require no reconfiguration when choosing the appropriate library entry. Once the library entry is selected, the controller B is executed to produce the desired robot behavior.

Configuration	Behavior properties	Environment properties
Tank	Pick up	Item in "free" environment
Tank	Drop	Drop-off zone in "free" environment
Tank	Drive	Flat plain
Proboscis	Pick up	Item in "tunnel" or "free" environment
Proboscis	Drop	Drop-off zone in "tunnel" or "free" environment

TABLE I: A library of robot behaviors

In this work, we created five library entries for two different configurations as listed in Table. I. The "Tank" configuration shown in Fig. 9e is capable of picking up and dropping items in "free" environment while driving on a flat plain as a holonomic drive robot. The "Proboscis" configuration shown in Fig. 9d is setup to have a long arm for reaching between obstacles in a "tunnel" environment to grasp objects. However, the locomotion ability of this configuration is limited to forward/backward motion, making it unsuitable for general navigation.

VI. Reconfiguration

Reconfiguration refers to the process of changing the connective topology of an MSRR system's modules in order to change its morphology. SMORES-EP is capable of all three classes of modular self-reconfiguration (chain, lattice, and mobile reconfiguration). For the purposes of this work, we have developed tools for mobile reconfiguration with SMORES-EP, taking advantage of the modules' ability to drive on flat surfaces as described in section III.

Determining the relative position of modules during mobile self-reconfiguration is an important challenge. As discussed in Section II, past systems have relied on offboard global positioning systems [9] or distributed approaches, in which sensors are mounted on each disconnected set of modules [15]. Our localization method is centralized, using an RGB camera carried by the robot to track AprilTag fiducials mounted to individual modules. The camera is mounted to the sensor module at a height of 20cm above the xtion RGB-D camera, and faces downward at $40°$ to vertical, providing accurate tracking in a $1m$ by $1.5m$ rectangular area on the ground in front of the sensor module. Within this area (which we call the *reconfiguration zone*), any module equipped with an AprilTag marker can detach from the cluster, drive to another location, and reattach to the cluster, provided that both of its wheels were in contact with the ground in its starting position.

A. Reconfiguration Procedure

Given an initial configuration and a goal configuration, our reconfiguration controller commands a set of modules to disconnect, move and reconnect in order to form the new topology of the goal configuration. The process consists of three stages: i) Pre-reconfiguration, ii) Module Movement, iii) Post-reconfiguration.

a) Pre-reconfiguration: In the pre-reconfiguration stage, the robot takes actions to establish the conditions needed for reconfiguration. It begins by confirming that the reconfiguration zone is a flat surface free of obstacles (other than the modules themselves). If the robot is carrying an object, it drops the object outside of the reconfiguration zone. It then assumes a stance (by changing its joint angles) in which any modules that will need to detach have both of their wheels on the ground, ready to drive. Once these conditions are established, module movement begins.

b) Module Movement: During this stage, the topology of the cluster changes as a sequence of module movement operations are performed. Each operation involves detaching one module from the cluster, driving, and re-attaching at its new location in the goal configuration.

We denote a module movement operation as $MP = (m, m_d, m_a, f_m^d, f_m^a, f_{m_d}, f_{m_a}, \sigma)$ where:

- m is the module whose location will be changed in this module operation.
- m_d is the module that connects with m before the operation.
- m_a is the module that connects with m after the operation. Notice m_d and m_a can be the same module.
- The face f_m^d of module m connects with the face f_{m_d} of module m_d before the operation.
- The face f_m^a of module m connects with the face f_{m_a} of module m_a after the operation. Notice f_m^d and f_m^a can be the same face. In this work, we assume f_m^a can only be the front or the back face of the module m.
- σ is the path that module m will follow during the operation.

Fig. 6 demonstrates an example of a single module operation. As shown in Fig. 6a, module m detaches from module m_d. In this process, magnets on f_m^d and f_{m_d} are turned off to eliminate the bond between two faces. Then module m moves the tilt joint while rotating the left and right wheel forwards to break the connection with m_d and drives away from it. After m detaches from m_d, the controller drives m as a two-wheel differential drive robot to follow the path σ with localization provided with the sensor module using AprilTags system. This process is illustrated by Fig. 6b. The path σ is c on free with the rest of the robot and computed *a priori*. Once m reaches the end of σ, it adjusts its heading by spinning in place till face f_m^a aligning with f_{m_a}. The magnets on face f_m^a and f_{m_a} are turned on to help create connection later between two faces. As demonstrated in Fig. 6c, ~~as~~ last m drives towards m_a while perform minor adjustment in the tilt joint till two modules are fully connected.

c) Post-reconfiguration: Once all module movement operations have completed and the goal topology is formed, the robot changes its joint angles to assume a stance in which the goal configuration can begin performing desired behaviors, and if necessary, the robot picks up any objects it dropped. This completes the reconfiguration process.

(a) Module detaches from the rest of the robot (b) Module drives to the new location (c) Module attaches to the rest of the robot

Fig. 6: An example of a module operation for reconfiguration.

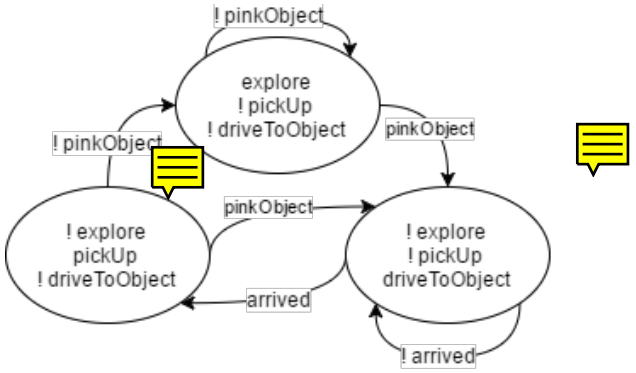


Fig. 7: Planner Automaton

B. Comparison with Existing Methods

Our method of reconfiguration has distinct advantages and disadvantages compared to existing methods. To the authors' knowledge, use of an onboard overhead camera to provide fiducial-based localization for reconfiguration is novel. This centralized strategy provides localization only near the sensor module, so the number of modules that can reconfigure simultaneously is limited by the size of the reconfiguration zone. This means it could not be directly scaled to large clusters of modules the way some existing distributed methods could be ([15],[8],[10]). However, it is not clear how these distributed methods would be employed in the context of a high-level task. Additionally, most require one sensor module per distinct cluster, which is cumbersome.

By exploiting high-fidelity centralized localization and the individual mobility of the SMORES-EP modules, we achieve reconfiguration that is rapid, robust, and ~~cleanly~~ deployable in the context of a high-level task. In our experiments (Section VIII), we observed that the pre-reconfiguration state required about 20 seconds, each module movement operation required about 45 seconds, and post-reconfiguration required about 10 seconds. Each reconfiguration action had a success rate over 90%. This compares very favorably with existing methods, in which each reconfiguration action can take on the order of ten minutes [15],[8],[10]

VII. High-Level Planner

To utilize the sensing and actuation capabilities of the robot as a complete system for accomplishing various tasks, we

employ an existing framework ~~called~~ LTLMoP for automatically generating robot controller from user specified high-level instructions using formal methods . LTLMoP allows us to use each component of our system as a low-level atomic controller and specify a wide-range of reactive robotic tasks as a set of high-level instructions using these controllers. In this section, we will mainly discuss how we model our system in order to use LTLMoP to automatically generate a high-level controller for our desired robot task.

A. ~~*Synthesize a controller*~~

We first abstract our system and the environment with a set of Boolean propositions. A system proposition "pickUp" is `True` means that the robot is picking up an item and `False` otherwise. An environment proposition "pinkObject" is `True` ~~means that~~ the robot is currently sensing a pink object and `False` otherwise. ~~Then we ca~~n specify the robot task in high-level instructions, such as Structured lish, supported by LTLMoP. Spec. 1 is an example of a robot task written in Structured English for searching a pink object and picking it up once the robot finds it.

Specification 1 Search and pick up a pink object

do **explore** if and only if you are not sensing **pinkObject** and you are not activating **pickUp**
do **driveToObject** if and only if you are not activating **pickUp** and you are sensing **pinkObject**
do **pickUp** if and only if you were activating **driveToObject** and you are sensing **arrived**

In additions to "pickUp" and "pinkObject", there are some more propositions. "explore" and "driveToObject" are system propositions that each represents a different robot action. "arrived" is an environment proposition that is `True` if the robot arrives at the pink object.

Using LTLMoP, we can synthesize a robot controller, if one exists, that satisfies the given robot task. The controller is in the form of a finite state automaton as shown in Fig. 7. Each state is labeled with the value of all system propositions. We denote a proposition with the value of `False` using a "!" prefix. Each transition between two states is labeled with the value of `some` environment propositions. Some states and transitions are omitted for clear presentation.

B. ~~*Execute a controller*~~

Since the synthesized high-level controller is a discrete finite state automaton, we need to implement it continuously in order to control the to satisfy the given task. Each system proposition is mapped to a low-level control program that commands the robot to perform some behaviors. For example, the proposition "pickUp" is mapped to a program that will be executed when "pickUp" is `True`. The program command the robot to perform a predefined behavior that will pick up a small magnetic object in front of the robot. Each environment propositions is mapped to a low-level sensing program that gathers and processes the information from sensors in order to characterize the environment state. For example, the propositions "pinkObject" mapped to a sensing program that returns `True` or `False` based on whether the robot can detect a predefined pink object with its camera.

To execute the synthesized high-level controller, we start with the predefined initial state in the finite state automaton. In each iteration, first we determine the value of each environment propositions by calling each corresponding sensing program. We then can find the next state in the finite state automaton by taking the transition that matches with the current values of all environment propositions. At last, we execute the corresponding control program based on the value of each system proposition specified in the next state and continue to the next iteration. For example, we start in the top state in Fig. 7 and execute the "explore" program. If the robot senses a pink object, the value of "pinkObject" is `True` and therefore the next state is the bottom right state. We then stop the "explore" program and execute the "driveToObject" program.

VIII. Experiment Results

To validate and demonstrate the capabilities of our system, an experiment w n in which a robot composed of SMORES-EP modules completed a complex, high-level task in an unknown environment. Figure 8 shows the environment layout, designed to represent a cluttered office. The high-level task consisted of 5 sub-tasks of 3 different types: explore the environment, find and retrieve 2 objects in the office, and deliver each to a drop-off zone. One object was in the open (a "free" environment type), and the other was positioned between two obstacles (a "tunnel" environment). Obstacles, environment types, and locations of the objects and drop-off zone were unknown to the ot *a priori*. Spec. 2 shows the high-level instructions from the user for synthesizing a robot controller for this experiment.

5 SMORES-EP modules, 1 Sensor Module, and 3 dummy modules were used to compose the robot for performing the task. Steel objects were attached to and carried using the electro-permanent magnets of the SMORES-EP modules. The two configurations and five behaviors described in Section V were available for selection by the high-level planner for this experiment.

Figure 9 shows snapshots from the experiment run. A video of the entire experiment is available as an attachment to this paper. The starting location prevented the robot from seeing the objects initially, forcing it to explore the environment to search for them. After a period of exploration the robot discovered the pink object first. The characterization algorithm correctly classified the surrounding environment as a "tunnel" type, and accordingly the robot navigated in front of the object and reconfigured to the "proboscis" configuration. Once this was done, the object was retrieved and pulled out into the open. The robot then dropped the object, reconfigured back to the "~~car~~" configuration for navigation to the drop-off zone, and again retrieved the object. The robot then navigated to and dropped off the object at the drop-off zone, which was

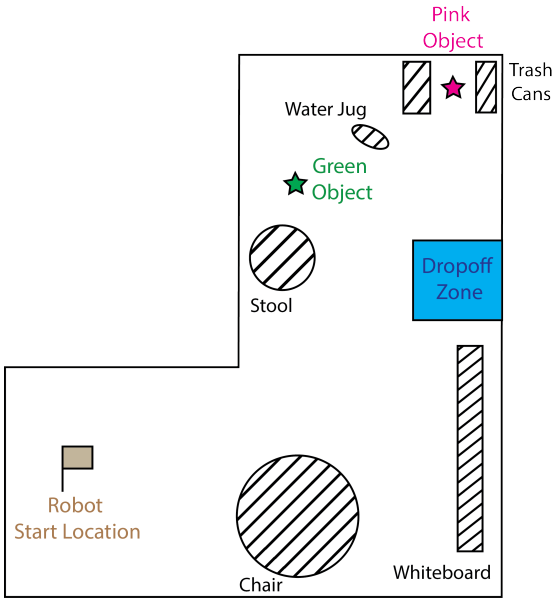


Fig. 8: Diagram of experiment environment

Specification 2 Search and move any pink or green object to the drop-off zone

carry is set on **pickup** and reset on **drop**
if you are not activating (**carry** or **pickup** or **drop** or **driveToItem** or **driveToDropoff**) then do **explore**
if you are activating **carry** and you are not sensing **dropoffzone** then do **explore**
do **driveToDropoff** if and only if you are activating **carry** and you are sensing **dropoffzone**
do **driveToItem** if and only if you are not activating **carry** and you are sensing **item**
do **drop** if and only if you were activating **driveToDropoff** and you are sensing **arrived**
do **pickup** if and only if you were activating **driveToItem** and you are sensing **arrived**
infinitely often not **carry**

seen during exploration and recorded in the global map built by the SLAM algorithm. Once done, the robot navigated to the green object which was correctly determined to not require any reconfiguration. No further exploration was required since the green object was discovered while the robot was dropping off the pink object. The robot finished the experiment by also delivering the green object to the drop-off zone. Figure 10 shows the final volumetric map of the environment explored by the robot as it explored the environment and delivered objects. The robot successfully completed all tasks in the experiment in about 26 minutes. Note that the video shows a human reaching into the field to touch the green object once the robot comes into contact with it. This was due to a error on the field resulting in the object becoming stuck between two floor boards, so a human dislodged it so the robot could grasp it ~~normally.~~

IX. Discussion

A. Challenges

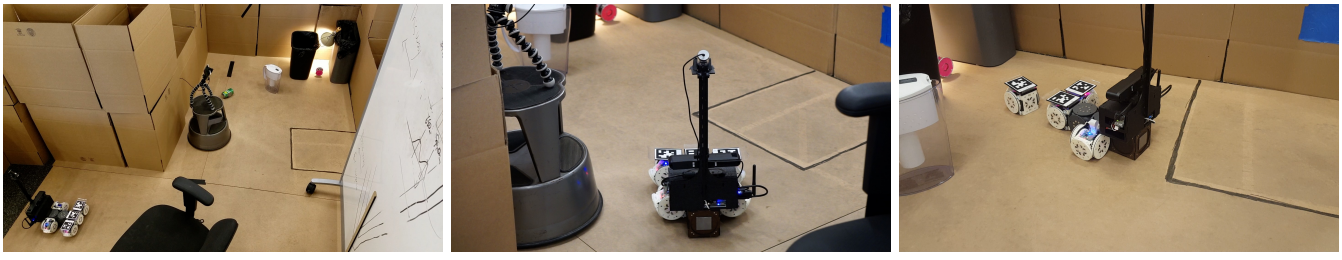
The experiment demonstrated that our system can autonomously perform complex high-level tasks in an unknown environment. It also revealed many issues that make autonomous reactive tasks difficult to achieve in modular systems. This section discusses issues that were observed and solutions that were implemented to overcome them.

Due to limitations of individual modules, robot speed was very low. In addition, having a robotic system composed of 6 individual robotic components (5 SMORES-EP modules and 1 sensor module) multiplies the chance of a component having a hardware failure. The long runtime combined with multiplied failure points resulted in a high risk of an error occurring during the experiment. Thus, each system component had to be designed with features to make it as robust as possible, and several error recovery features had to be implemented in the system.

The processing power required for perception and environment characterization presented issues with size and computation speed. As can be seen from Figure 3, the sensor module is significantly larger than a regular module in order to hold the required sensors and computer board. This imposes significant limitations on possible configurations and their locomotion abilities, due to the low connection strength and power of modules. At the same time, the small computer board used in the sensor module results in low computation power, which causes the sensor processing and perception algorithms to run slowly and have less accuracy. Simple configuration designs and efficient algorithms were used to address these issues. Future autonomous modular systems would benefit significantly from powerful but highly compact, lightweight sensing and processing components that can be carried onboard the robot while imposing minimal size and weight constraints.

As with many autonomous systems, uncertainties such as sensor noise and controller lag create issues with robust behavior. For example, the global map and robot pose determined by visual SLAM accumulate significant errors (due to sensor noise) as time progresses. To compensate, repetitive feedback controllers and checks were implemented during task performance. As the robot navigates toward and retrieves objects, it constantly updates its estimate of the object location as new measurements of the object are received. During reconfiguration, docking modules are checked and re-aligned several times during the docking process to ensure they are lined up precisely before connecting to each other. This helps the robot recover from controller lag or noisy AprilTag measurements.

Finally, some simple hardware failures were encountered during experiment runs. Since SMORES-EP modules are research prototypes built from scratch, they are more prone to failures such as microcontroller errors and position control errors from custom encoders. Future experiments can be improved in robustness by improving the hardware quality of modules to bring them closer to the level of a commercial product.



(a) Experiment setup and robot starting location

(b) Exploring environment while searching for objects

(c) Reconfiguring to retrieve pink object

(d) Retrieving pink object

(e) Dropping off an object in the delivery zone

(f) Retrieving green object

Fig. 9: Environment characterization types for object retrieval.

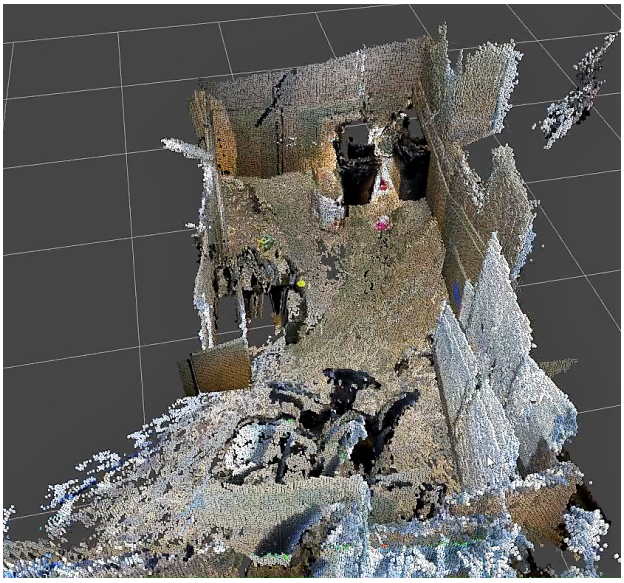


Fig. 10: Volumetric map of environment built by visual SLAM

B. Future Work and Conclusion

We presented a novel system for autonomously accomplishing high-level tasks in unknown environments with modular self-reconfigurable robots. This system is the first to use perception of an unknown environment to reactively perform complex high-level tasks using intelligent reconfiguration. Components of this system include novel controller synthesis, environment characterization, and self-reconfiguration methods. The system was validated using a physical experiment using a high-level task consisting of a heterogenous set of 5 sub-tasks in an *a priori* unknown environment.

The system is not without limitations, and could be significantly extended in future work. First and foremost, the experiment we present utilizes only two configurations, and our perception system demonstrates the ability to recognize only two special environments (ledge and tunnel). MSRR systems are supposed to provide wide flexibility at addressing tasks. To truly prove this promise, we need to demonstrate that MSRR systems can reconfigure between a much wider variety of morphologies, in response to a large number of environments. We believe that our system could be extended to such scenarios, but we do not minimize the difficulty of the problems one should expect to encounter in tackling these problems.

While the presented system successfully demonstrated that modular robots can autonomously achieve a realistic object retrieval task, limitations of the modular hardware make them unsuitable for application in harsh real-world scenarios such as search-and-rescue. One of the biggest limitations is simply the small size and small forces that can be applied by the modules. Given current actuator and connector technology, it is still very difficult to apply large enough forces to operate in a realistic search-and-rescue environment.

It is important to consider how the presented system could be adapted to work with other modular robot hardware. Certain aspects of the system are almost entirely hardware-independent, such as the sensing and navigation tools, the high-level planner, and the hardware abstractions. The reconfiguration method is in some ways specific to SMORES-EP, because it relies on the ability of each module to move by differential drive on the flat plane. The majority of MSRR hardware systems do not have this capability, and would not be able to reconfigure in this way. However, we believe the

success of this novel reconfiguration method could have some implications for MSRR hardware design: future designers might want to consider designing for individual motility and forgiving connectors, in order to take advantage of a reconfiguration method similar to the one present.

To conclude, this paper presents the first system that enables modular self-reconfigurable robots to autonomously achieve tasks requiring perception-informed reconfiguration in an unknown environment. This capability is crucial to the success of modular robots as a technology, and by demonstrating it for the first time, we believe we have taken a real step toward the application of modular robots to solve tasks in the real world.

ACKNOWLEDGMENTS

This work was funded by NSF grant numbers CNS-1329620 and CNS-1329692.

REFERENCES

[1] Marco Dorigo, Elio Tuci, Roderich Groß, Vito Trianni, Thomas Halva Labella, Shervin Nouyan, Christos Ampatzis, Jean-Louis Deneubourg, Gianluca Baldassarre, Stefano Nolfi, Francesco Mondada, Dario Floreano, and Luca Maria Gambardella. The SWARM-BOTS Project. *LNCS*, 3342:31–44, 2005.

[2] Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, Daniel Burnier, Alexandre Campo, Anders Lyhne Christensen, Antal Decugniere, Gianni Di Caro, Frederick Ducatelle, Eliseo Ferrante, Alexander Förster, Javier Martinez Gonzales, Jerome Guzzi, Valentin Longchamp, Stephane Magnenat, Nithin Mathews, Marco Montes De Oca, Rehan O'Grady, Carlo Pinciroli, Giovanni Pini, Philippe Rétornaz, James Roberts, Valerio Sperati, Timothy Stirling, Alessandro Stranieri, Thomas Stützle, Vito Trianni, Elio Tuci, Ali Emre Turgut, and Florian Vaussard. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics and Automation Magazine*, 20(4):60–71, 2013.

[3] Robert Grabowski, Luis E. Navarro-Serment, Christiaan J J Paredis, and Pradeep K. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8(3):293–308, 2000.

[4] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. doi: 10.1007/s10514-012-9321-0.

[5] Gangyuan Jing, Tarik Tosun, Mark Yim, and Hadas Kress-Gazit. An End-To-End System for Accomplishing Tasks with Modular Robots. *Robotics: Science and Systems XII*, 2016. URL http://www.roboticsproceedings.org/rss12/p25.pdf.

[6] M. Labbe and F. Michaud. Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2661–2666, Sept 2014.

[7] Francesco Mondada, Luca Maria Gambardella, Dario Floreano, Stefano Nolfi, Jean Louis Deneubourg, and Marco Dorigo. The cooperation of swarm-bots: Physical interactions in collective robotics. *IEEE Robotics and Automation Magazine*, 12(2):21–28, 2005.

[8] Satoshi Murata, Kiyoharu Kakomura, and Haruhisa Kurokawa. Docking experiments of a modular robot by visual feedback. *IEEE International Conference on Intelligent Robots and Systems*, pages 625–630, 2006.

[9] James Paulos, Nick Eckenstein, Tarik Tosun, Jungwon Seo, Jay Davey, Jonathan Greco, Vijay Kumar, and Mark Yim. Automated Self-Assembly of Large Maritime Structures by a Team of Robotic Boats. *IEEE Transactions on Automation Science and Engineering*, pages 1–11, 2015.

[10] M. Rubenstein, K. Payne, P. Will, and Wei-Min Shen Wei-Min Shen. Docking among independent and autonomous CONRO self-reconfigurable robots. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 3:2877–2882, 2004.

[11] Tarik Tosun, Jay Davey, Chao Liu, and Mark Yim. Design and characterization of the ep-face connector. In *IROS*. IEEE, 2016.

[12] M Yim. *Locomotion with a unit-modular reconfigurable robot*. PhD thesis, Stanford University, 1994.

[13] M Yim, WM Shen, and Behnam Salemi. Modular self-reconfigurable robot systems: Challenges and Opportunities for the Future. *IEEE Robotics & Automation Magazine*, (March), 2007. URL http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=4141032.

[14] Mark Yim, David G Duff, and Kimon Roufas. Modular Reconfigurable Robots, An Approach To Urban Search and Rescue. In *1st International Workshop on Human-friendly Welfare Robotics Systems*, pages 69–76, 2000. URL http://modlab.seas.upenn.edu/publications/HWRSpaper2.pdf.

[15] Mark Yim, Babak Shirmohammadi, Jimmy Sastra, Michael Park, Michael Dugan, and C J Taylor. Towards Robotic Self-reassembly After Explosion. In *IROS*, 2007.