Hierarchical Control for Self-assembling Mobile Trusses with Passive and Active Links

Carrick Detweiler*, Marsette Vona†, Keith Kotay and Daniela Rus
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
*carrick@mit.edu,†vona@mit.edu

***Abstract*— This paper explores the space of active modular trusses, ranging from a passive truss with one independent active climbing module to fully self-reconfiguring dynamically controllable trusses comprised of active modules and passive struts. We describe a hardware design for truss climbing and present hierarchical algorithms for controlling hyper-redundant modular trusses.**

I. Introduction

Self-reconfiguring robots are modular robot systems that are physically connected and capable of making structural geometric changes autonomously. Most current research in this field is focused on homogeneous systems in which all the modules are identical. In this paper we study a special class of heterogeneous self-reconfiguring robots we call active trusses. The robots in this class look like trusses and are comprised of two types of modules: passive structural modules which may either be fixed in the environment or free to move individually, and mobile active modules which may pick up or climb on the passive modules, organize and hold them in a desired shape, and actively move them for self-assembly, self-reconfiguration, or self-repair purposes. The passive modules can be passed around by the active modules and coordinated to form the skeleton of a large class of truss geometries. For example, figures 8 and 9 show a self-assembling active tower belonging to this class. Such active trusses have many potential applications, ranging from self-assembly of truss structures for space exploration to creating dynamic scaffolds and movable towers for construction tasks.

A long-term application of these systems is in-space structure construction. Simple self-reconfiguring robot modules will pack tightly in a spacecraft, yet they will be able to self-assemble, self-reconfigure, self-repair, and adapt their collective morphology, and function, to perform a variety of tasks—some known in advance (pre-launch) and some dynamic (post-launch). The modules can act both as effectors to assemble/repair/service other space structures and as active orbiting structures themselves. Other applications include terrestrial construction of increasingly more capable structures.

One specific application we are developing is customized window shading. We work in a lab with a large wall-window (about 4m tall and 8m wide) which has no shades to block sunlight. Instead of traditional shades which would block the whole window, we are using the fixed truss structure formed by the window's grid of aluminum supports as a passive element and one module, called *Shady*, as an active element. Shady will grip and locomote on the window frame grid to an optimal location where it will deploy a fan, thus creating active personal shading in the lab. Experiments with a preliminary version of the hardware have shown locomotion capability but require a slight modification to the window frame (section IV-A.2). We are currently developing revised hardware to solve this problem; initial results are promising, and we hope to report fully on this work in a latter publication.

The challenge in building such mobile trusses spans the entire spectrum from issues related to designing simple and robust active modules to problems of control and planning. Control for these systems is challenging because their c-space has dynamic topology, and in the general case they are under-constrained (hyper-redundant) systems with a continuum of solutions. Similarly, planning is challenging due to the large number of degrees of freedom that have to be coordinated for high-level tasks.

In this paper we propose the concept of building trusses with passive and active modules and examine some control and coordination issues that arise when using such systems. We describe a continuum of trusses that covers the spectrum from passive trusses with active modules that can traverse them all the way to active trusses that can self-assemble, self-inspect, self-repair, and move. We present a hardware instantiation of the truss-climbing robot *Shady*, and the concept of *Multi-Shady* which instantiates the idea of mobile active trusses in simulation. We present our hardware and control algorithms for Shady and our simulated control algorithms for Multi-Shady. Finally, we discuss the benefits of building self-reconfiguring robots with active and passive modules.

II. Related Work

Our proposed systems and algorithms are related to prior work in the fields of self-reconfiguring robots, hyper-redundant robots, and variable-geometry truss robots.

A. Self-Reconfiguring Robots

Of all the self-reconfiguring modular robots which have been previously reported, our current work seems most closely allied with systems based on rotary DOF and mechanical connection mechanisms, for example: Murata, Kurokawa, et al's "3D Fracta" [1]; Kotay and Rus' "Molecule" [2], [3], [4]; Unsal, Kiliccote, and Khosla's bi-partite "I-Cubes" [5] system;

Duff, Yim, et al's PolyBot [6]; and Lund, Beck, Dalgaard, Støy et al's ATRON [7], [8].

A major difference in our present work is that we are proposing *bi-partite* modular systems with only some modules containing active DOF—the rest serve as static structural elements. In contrast, all of the above referenced systems are either homogeneous (all modules identical and actuated) or are heterogeneous but still require actuation in all modules.

B. Hyper-Redundant Robots

Research in the field of "hyper-redundant" robots has mainly explored non-reconfiguring systems with high DOF and fixed kinematic topology, typically open chains. Both planar systems—e.g. Burdick and Chirikjian's "snakey" [9]; Greenfield, Rizzi, Choset et al's modular snake [10]—and full spatial mechanisms—e.g. Suthakorn and Chirikjian's binary-actuation manipulator [11]; Wolf, Choset, et al's "Schmoopie" [12]—have been explored. The planar systems typically have one kinematic DOF per link, and the spatial systems may have two or more. Sometimes the links are internally parallel mechanisms, an arrangement which has been called "hybrid serial-parallel" ([13], [14], [11]).

Our proposed two-leg tower construction (section V-B) is a hybrid serial-parallel mechanism; and our single-chain tower is kinematically equivalent to typical hyper-redundant snakes. Thus far we have applied classical pseudoinverse-derived inverse kinematics methods for these structures but we are also considering adaptation of methods developed specifically for hyper-redundant robots, for example Chirikjian's "backbone curve" method [15].

C. Variable Geometry Truss Robots

Variable geometry trusses (VGTs), can be viewed as a generalization of the serial-chain hyper-redundant systems to more general kinematic topologies. Both fixed-topology systems like the NASA/DOE "SERS DM" [16] and manually-reconfigurable systems—notably Hamlin, Sanderson, et al's TETROBOT [13]—have been considered. Also related are robotic systems which assemble static trusses, for example, Everest, Shen, et al's SOLAR [17]. Such self-assembling and self-reconfiguring truss systems can be a promising direction for robotic assembly of large structures in space—for example, see Doggett's overview of automatic structural assembly for NASA [18].

Truss *climbing* robots are also under active investigation, e.g. Amano et al's handrail-gripping robot for firefighting [19], Ripin et al's pole climbing robot [20], Nechba, Xu, Brown et al's "mobile space manipulator SM2" [21], [22], and Almonacid et al's parrallel mechanism for climbing on pipe-like structures [23]. Truss climbing also has been acknowledged to have clear applications in inspection and construction of in-space structures [24].

Our proposed systems can act as self-reconfiguring/self-assembling modular VGTs (section V), and our Shady robot (section IV) shows how the same module designs can also be applied to truss-climbing.

III. Abstract Truss Model and Software

We envision an abstract *continuum* of modular truss robots with varying functionality. The simple end of the continuum is a fixed truss with one active climbing unit (section IV), and the complex end is a self-assembling/self-reconfiguring variable-geometry truss composed of active and passive robot modules (section V). Examples of intermediate points along the continuum include a fixed truss with multiple independent climbing units and manually-assembled variable-geometry trusses.

By selecting a point along the continuum, a designer can match system function (and cost) to the requirements of a specific application. As a further aid to system design, we propose unified models for robot modules which allow re-use of basic electromechanical designs and kinematic control algorithms in implementations at all points on the continuum.

A. Generic Module Models

It is likely impractical to specify a single hardware design which applies to all modular truss applications, so instead we propose abstract module models which can be scaled, adapted, and specialized to yield hardware appropriate for classes of applications. This allows us to re-use not only the basic electromechanical layout, but also kinematic control algorithms (path planner, section IV-C; hierarchical control methods, section V), and user-interface visualization/commanding software (section III-B).

We propose two abstract module models: *passive* units, which are simple rigid bars; and *active* units which incorporate several actuated DOF and which grip one or more passive units.

Though most of our work to-date has been on 2D (planar) trusses, we also describe simple extensions to the module models for 3D trusses.

1) Passive Module Model: The passive modules are simply straight rigid bars. Their cross-section, length, and material properties are application-dependent, and in some applications these attributes may vary among modules. Since these passive modules are rigid they have zero intrinsic state. Their extrinsic state is the appropriate rigid body transformation: two translations and one rotation in 2D; three translations and three rotations (two if the module is a solid of revolution) in 3D. We consider passive modules to be oriented, so even when the bar geometry is symmetric we distinguish the two possible orientations the bar may take along a fixed axis. Individual passive modules may be marked *grounded* to represent static trusses attached to the environment.

We usually depict passive modules as light-colored line segments, as shown in figure 1.

2) Active Module Model: Active modules contain several actuated DOF and connection mechanisms for attaching passive modules. In 2D we have focused on a module with two rotating grippers, as shown in figure 1. Such a module can hold two passive modules in arbitrary relative orientations (figure 1, middle), and it can also locomote independently along a fixed truss by alternately gripping and swinging (figure 1, bottom). In 2D we model the active and passive modules to occupy

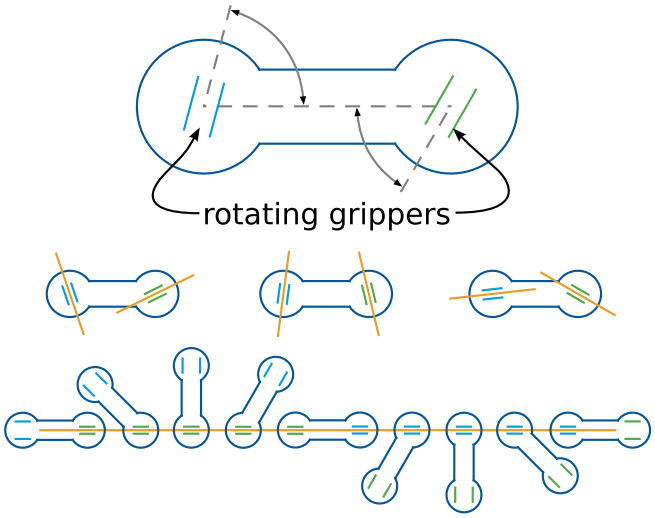

Fig. 1. A 2D active module concept (top) with two independently rotating grippers which can connect to passive modules (middle). Such a module can locomote independently along a fixed truss by alternately gripping and swinging (bottom).

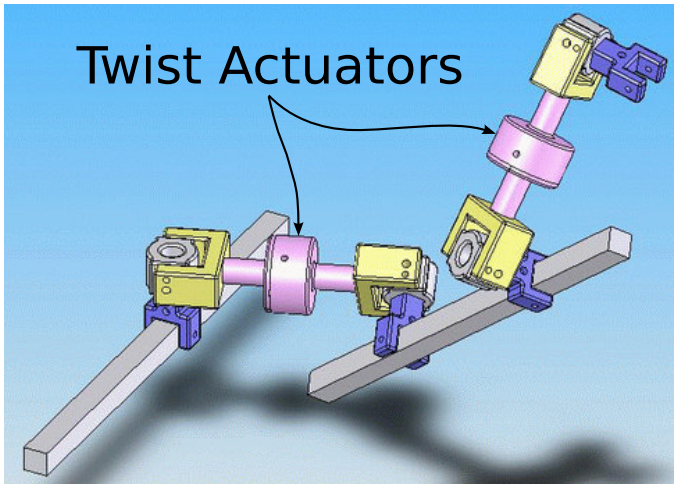

Fig. 2. A potential concept for extending the 2D active module concept shown in figure 1 to 3D by adding one new "twist" DOF. Two active and two passive modules are shown here in a chain topology. Figure contributed by Yeoreum Yoon.

separate parallel planes, and we assume that the grippers retract when open so that they may move over passive modules without collision.

The gripper rotations give these modules two intrinsic DOF, and their extrinsic state is again given by a 2D rigid-body transform. We consider the grippers to have distinct orientations, and we represent two properties when a gripper is attached to a passive module: *where* along the axis of the passive module the (center of) the gripper lies, and whether the gripper vector is aligned or anti-aligned with the passive module's orientation.

We are currently exploring the extension of this model to 3D by adding a third rotary "twist" DOF which can control the relative angle between the axes of the gripper rotations, as depicted in figure 2.

B. Software Architecture

We are developing a unified software package based on these generic abstract module models which can handle command and high-level control of hardware as well as full kinematic simulation. Each module type is represented by a software abstraction which encapsulates the details of particular implementations. Thus far, we have developed such implementations for:

- fixed and mobile simulated passive linear bar modules in 2D
- a simulated 2D active module which predicts resource consumption (battery usage, etc.) as well as kinematic state
- a hardware version of the 2D module adapted to truss climbing (section IV-A)
- a simulated 2D active module specifically adapted for large-scale simulations of many cooperating units

Hiding the details of particular module implementations below generic—but extensible—abstractions of the passive and active module types has allowed us to write generic high-level control and user-interface code that can be applied in all cases. These high-level components are always run on a workstation and present a real-time graphical display as well as a Scheme-language command interface.

For simulations, the active and passive module instance codes also run on the workstation, communicating with the higher-level generic code by direct procedure call. For hardware, each active module runs its own code in an on-board processor and interfaces with the generic high-level code running on the workstation by remote procedure call over the lab network. We have demonstrated both scenarios already, using the same workstation-based user-interface and high-level control code to run simulations and to operate the Shady hardware.

IV. *Shady*: Active Module on Passive Truss

The remainder of the paper is divided in two parts: here, we describe our initial hardware work at the simple end of the truss-robot spectrum—a single active module which locomotes on a fixed passive truss. Next, in section V, we describe simulations we have performed at the complex end of the spectrum on large-scale self-assembling/self-reconfiguring mobile trusses.

Shady (figure 3) is a four degree of freedom robot which can climb on the lattice truss structure of the large windows in our lab, and which carries a deployable fan for use as a personal sun-shade. The two connective and two revolute DOF are implemented as rotating grippers following the pattern of figure 1.

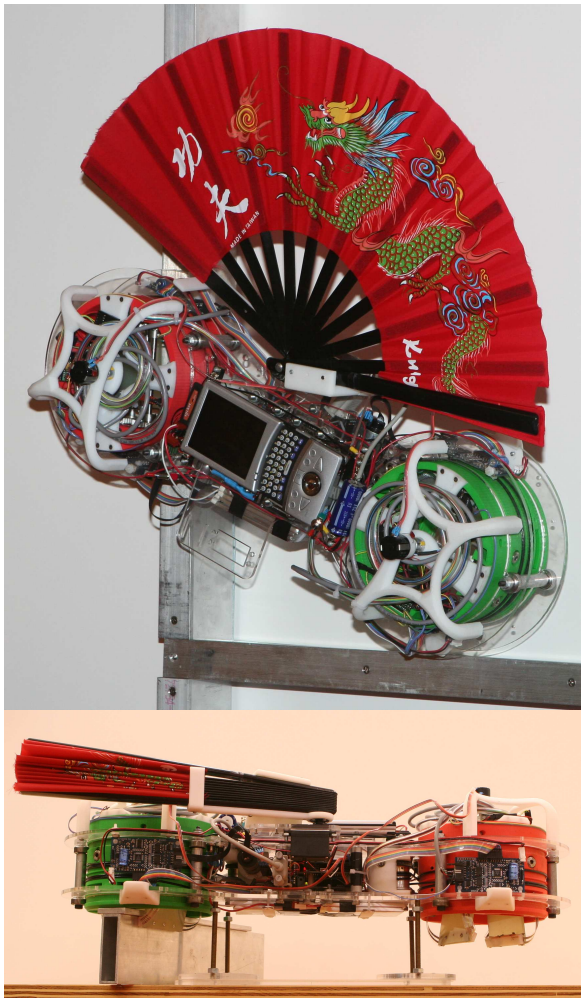

Fig. 3. *Shady*, a four degree of freedom robot which can climb on the lattice truss structure of the large windows in our lab (represented by the aluminum bars), and which carries a deployable fan for use as a personal sun-shade.

A. Hardware

The grippers are contained in rotating "barrels" which are connected via a "dog bone"-shaped body, as shown in the images. The body contains much of the controller electronics, batteries, and the barrel rotation actuators. In addition, there is an actuated fan which can be deployed as needed to provide shade. The body has a total length of 59cm with a barrel center-to-center distance of 39cm. The total weight of the robot is 5.5kg, including its Lithium-Polymer battery pack, and it is able to run un-tethered for over four hours of continuous motion, and about eight hours under more typical usage.

1) Barrel Design: Each barrel is a hollow tube containing a gripper mechanism. Barrel rotation is effected by a cable drive system which provides low-backlash $\pm 360^\circ$ rotation, and this drive is able to provide the necessary torque for rotating the robot body to any cantilevered orientation in the vertical plane. Rotation feedback is obtained from motor encoders and from potentiometers located above each barrel. Two infra-red proximity sensors are also included on each barrel, in-line with the gripper axis and facing toward the window, and are used to help align the gripper with the window frame. The barrel tubes were built on a rapid prototyper out of ABS plastic, and other standard and custom-machined parts complete the design.

2) Gripper Design: The current gripper design (figure 4) requires a slight modification to the window frame. Specifically, we added a small "T" cap to the window bars which allows the gripping mechanism to catch and positively hold the window frame. With this modification, the gripper is easily able to hold the robot in any orientation. We are currently testing a new gripper design which leverages linkage kinematic singularities to achieve much larger gripping pressure, and preliminary results indicate that this design will allow robust locomotion on the unmodified window frame. We hope to report more fully on this new work in a future publication.

An important feature of the gripper is its ability to fully retract, allowing disconnected barrels to pass over the window frame without collision.

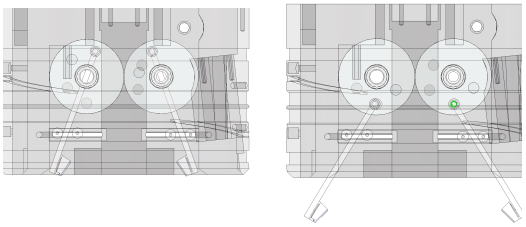

Fig. 4. A CAD image of the barrel with gripper open (left) and partially closed (right). Translucent rendering allows the internal mechanism of the gripper to be seen through the wall of the barrel. The gripper incorporates two "paddles" attached to counter-rotating gears which extend downward and then close onto a bar of the window frame.

The gripper is designed to be highly compliant. Its paddles open very wide—8.0cm at maximum—to grip onto the 2.5cm window bar, and they are able to grab the bar when it is up to 1.5cm away (a distance much greater than observed during experiments). These compliances allow the gripper to close even when not precisely aligned to the window frame. Additionally, there are force sensors located at the corners of the gripper paddles which can sense and compensate for angular misalignments of up to 25 degrees (measured experimentally).

3) Control System: Shady incorporates a network of five Acroname "Brainstem" motor control and sensor modules for low-level real-time control. These implement the basic hardware functions described below in section IV-B. Each barrel contains a sensor module to interface with all the sensors in the barrel and gripper. The robot's central body contains two motor control boards (two outputs each) and an additional sensor module which monitors battery levels and which networks the other modules together.

The Brainstem network communicates with an on-board Sharp Zaurus PDA (running Linux) which runs mid-level sensor-based motion control algorithms. The Zaurus shares much of its code base with the code we use for fully simulated

mobile modules, and implements the generic active module software interface described above, making algorithms developed in simulation easy to run on the hardware.

B. Experiments

We have performed a number of experiments to characterize the performance and reliability of the first version of the Shady hardware on a fixed truss. From this we have identified areas to improve in future revisions. We first introduce the primitive moves that are required for walking on a truss structure.

1) Primitive Moves: If the geometry of the environment and the pose of the robot are completely known then it is theoretically possible to explicitly compute the needed barrel angles to perform a step. However, uncertainty is inevitable, and to compensate for it we have developed a set of primitive moves which make use of the sensors we have included on Shady. The first move is called `find`. As the name implies, this move finds the nearest bar to the distal (i.e. disconnected) barrel in either a clockwise or counter-clockwise direction. The algorithm is straightforward: the connected barrel rotates, hence moving the distal barrel in a circular arc, until the distal barrel detects a bar with one of its proximity sensors.

After a `find` is successful, we perform an `align`. This move refines the alignment of the distal barrel to the bar by rotating it to find the edges of the bar and then moving to a position midway between these. Next, a `grip` move closes the gripper while monitoring the force sensors on the gripper paddles. Imbalance in these readings indicates that the barrel does not have correct angular alignment, so these sensors are used as input to a PD controller which further refines the alignment. The `grip` move was experimentally verified to handle up to 25 degrees of angular misalignment as well as 3.75cm of translational error.

The final move we perform is an `ungrip`, which opens the gripper. When this occurs in a horizontal configuration the `ungrip`ping barrel may drop somewhat due to gravity. This can cause problems as the retracting gripper paddle may catch on the window bar. To compensate for this during an `ungrip` we monitor the force sensors on the *other* (i.e. connected) barrel and actuate to correct for any undesired movement.

2) Experimental Setup: We performed 103 `find-align-grip-ungrip` sequences from various configurations. This was done on a lattice we built (aluminum bars in figure 3) which emulates the window frame but which is more easily accessed. The configurations we tested included horizontal, vertical, horizontal-to-vertical, and vertical-to-horizontal. We recorded the number of successes and failures of all the individual moves (a move was considered to have failed if it did not perform as described previously).

3) Results and Discussion: Below is the results of the 103 `find-align-grip-ungrip` sequences:

	`find`	`align`	`grip`	`ungrip`	Total
Success	90	88	88	88	84
Failure	13	11	4	2	19
% Success	87.4	88.9	95.7	97.8	81.6

The `find` move failed 13 times, however, a `find` failure is fairly easy to detect, and in this case we perform a binary search to attempt to find the bar, allowing continuation of the subsequent operations in 11 out of 13 trials. Most of the `find` failures related to an unresolved bug in the Brainstem software which caused the distal barrel to not rotate as commanded.

The remaining `find` failures, and many of the `align` failures, were due to a poorly initialized proximity sensor during one series of tests. These sensors are automatically calibrated each time the robot starts; however, sometimes this does not work as well as desired. Adding to the problem is the fact that the proximity sensors have a slightly larger field of detection than desirable, so the robot may think that it is aligned when, in fact, it is not.

While `find` and `align` fail gracefully, without potential damage to the robot, failures in `grip` and `ungrip` can damage the robot. The gripper element which holds the force sensors is particularly prone to damage. However, these types of failures occurred much less frequently than the other types.

`grip` failures occurred when the barrel was sensed to be aligned to the bar, but the bar was still misaligned too much for a successful grip. The two `ungrip` failures were caused by over-compensation for the "gravity" affect mentioned in section IV-B.1.

Overall, the experiments illustrated that the methods are sound, however, there is room for improvement–particularly on the hardware front. We have had several runs with over 6 consecutive steps, but larger motions appear to be less feasible with this initial hardware implementation.

C. Path Planning

The above-described low-level primitive moves can be combined sequentially to locomote on an arbitrary "environment" truss. Arbitrary-angle concave and convex transitions are possible, as are linear gaits along passive segments at any orientation. Such locomotion could be used for applications such as material and tool delivery across truss structures or inspection/repair on trusses. For the sun-shading application, for example, Shady could move to position its fan so that it intersects the ray from the sun to a particular researcher's computer screen. In this application the reachable shade locations are limited to an offset-band about the skeleton of the window frame, so our implementation of Shady and its fan has been scaled appropriately so that the reachable band covers a large fraction of the area of our window. The window and, in particular, its structural aluminum frame, are shown schematically in figure 5.

We have developed a path-planning algorithm to determine a short locomotion sequence to any target location in the reachable band, and we have implemented this algorithm in simulation. Since our Shady hardware presents the same generic control interface, it should be possible to run this same high-level planning code to control the hardware—an experiment we look forward to performing.

The path planning algorithm is illustrated in figure 5. The user interacts with the planner and can specify a target location

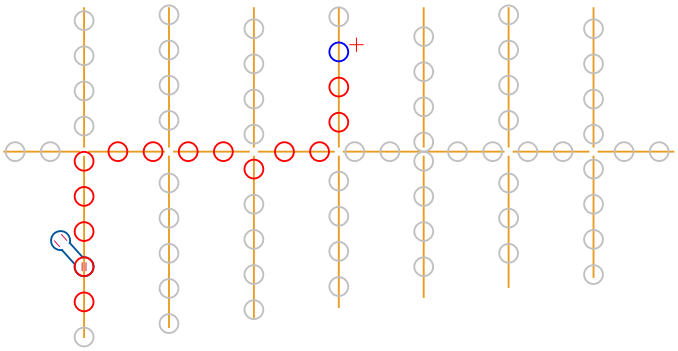

Fig. 5. All reachable Shady grip points (light blue) in the window frame truss environment (horizontal and vertical line segments) up to a certain granularity, and a short path through these grip points (red circles) to a user-specified target point (red cross). This simulator image depicts Shady beginning to traverse the indicated path, beginning from its start location in the lower-left corner of the window.

for Shady by clicking on the screen. The algorithm first performs a breadth-first search of reachable gripper locations on the truss based on the kinematic structure of the robot, the geometry of the truss, and the robot's starting point.

The set of reachable grip points is discrete, but may be infinite even in finite environments due to "spiral" motion sequences about truss joints which can return the robot to its original grip location plus an arbitrarily small delta. Thus, it is useful to put a maximum bound on the search—grip points closer than a specified distance to already-found points are pruned. The final set of discovered grip points is then used to create a graph on which Dijkstra's shortest-path algorithm can be run.

The final step, picking a particular grip point for any given target location, is non-trivial. Choosing the point closest to the target by the standard planar Euclidean distance metric is one strategy, although it is possible that this point will require a longer locomotion sequence than another grip point which is sufficiently close. It may be desirable to pick several nearby grip points and evaluate them based on their closeness to the target vs. their locomotion sequence length.

V. *MultiShady*: Mobile Active Trusses

Our current Shady hardware is scaled for climbing on a fixed truss, and is likely not appropriate for applications at the high end of the truss robot spectrum involving many cooperating active and passive modules. However, we expect that hardware with the same basic kinematic topology–that of the abstract mobile module shown in figure 1–could be designed and built in a way that is applicable to such cooperative applications. For a system operating under terrestrial gravity, necessary changes will likely include making the module significantly smaller, increasing its power-to-mass ratio, improving the gripper design, and possibly designing the system to distribute power through the truss. Other design considerations might become important in zero/low-gravity environments, or for particular applications.

We are beginning the design of such modules, shown conceptually in figure 2 (also depicted is a simple extension to three dimensions, which we are simultaneously designing), and we have begun to explore the possible capabilities and control of large-scale cooperating self-reconfiguring/self-assembling truss robots in our 2D simulation environment. We present some initial results from these simulations here.

A. Concept of Hierarchical Control

A key concept we have been developing is the *hierarchical control* of large-scale truss robots. We divide the total set of active and passive modules into disjoint groups, and we design particular controllers and planners for these smaller groups (there may be many instances of the same *type* of group, so the total number of distinct group control algorithms may be much lower than the total number of group instances). We also implement controllers and planners which operate at the highest level, and which consider the aforementioned groups to be monolithic meta-modules, thus forming a two-level hierarchy of control.

As the scale of the systems we explore increases, we predict that it will likely be useful to extend this hierarchy to additional levels; i.e. to form groups within groups, etc., at each level designing controllers and planners which operate on meta-modules of the lower-level.

Meta-modules in self-reconfiguring robots have previously been explored ([25], [26], [27], [28]), but mostly in the context of topological reconfiguration and structural shape-changing. We extend the concept to also include kinematic/geometric control (e.g. figures 9 through 11) and we also generalize it to a hierarchy of module-group controllers which may each have several different operational modes, as described next.

B. Tower Simulations

As an example of hierarchical control, figures 6 through 10 show the construction and operation of a reconfigurable mobile tower. The groups, in this case, are composed of five active modules (blue/dark segments) and four passive modules (orange/light segments). We have developed a set of seven separate controllers for such groups which can

- assemble the group from a starting "packed" configuration into a two-legged walking structure (figure 6)
- locomote the walking structure with a statically stable gait along a fixed truss segment from the site of the walker's creation to the base of a tower-in-progress (figure 7)
- make a concave transition from walking on the segment to walking up the side of the tower (figure 8)
- walk up the side of the tower (figure 8)
- make a convex transition from walking up the tower to standing on top of the tower (figure 8)
- reconfigure from the walker shape to an inverted-U shape trapezoidal tower structural block (figure 8)
- tilt, as a tower block, to the left or right (figure 9)

Using these group controllers, we can easily direct the simulated construction of an arbitrary-height tower. Figure 9 shows a 15-block tower containing 75 active modules and 60 passive modules.

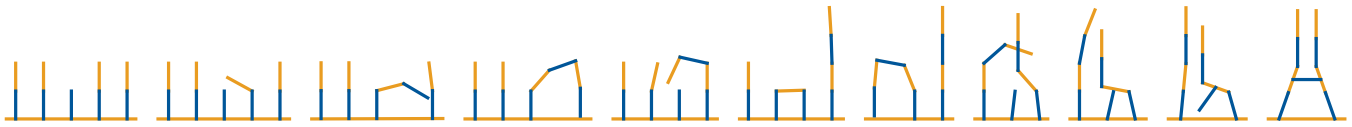

Fig. 6. Snapshots from a simulation showing the construction of a two-legged walking structure (rightmost) starting from a "packed" configuration of active and passive modules (leftmost).

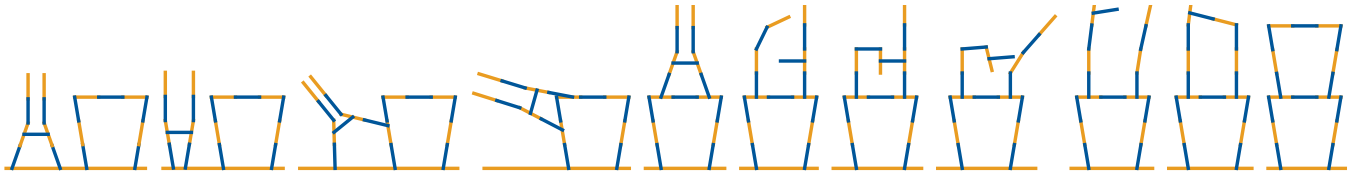

Fig. 8. Snapshots from a simulation showing a walker structure performing concave and convex transitions, walking up a tower, and reconfiguring into a new structural block of the tower.

Fig. 7. Snapshots from a simulation showing a walker structure locomoting on a truss segment.

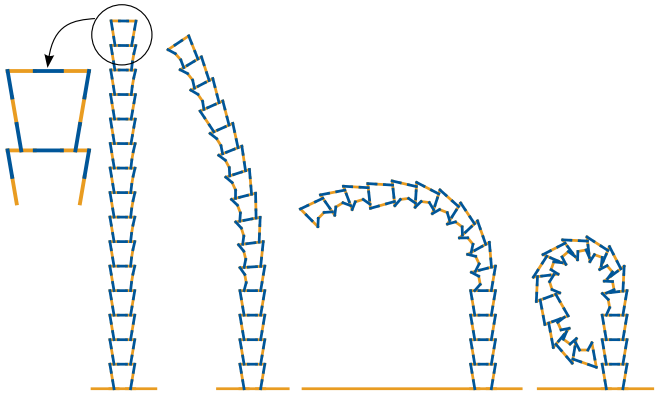

Fig. 9. Simulation of a 15-block tower acting as a hybrid serial-parallel hyper-redundant active structure, in this case under keyframe-type high-level control. We have also implemented damped-least squares inverse kinematics control (figure 10). This example shows a possible self-inspection capability—a camera mounted on the tower top could be aimed to inspect lower parts of the tower.

Once such a tower is assembled, we can apply a high-level controller to command the blocks to collectively perform a task. We have explored high-level controllers which utilize the block-tilting group controller to make the tower a hybrid serial-parallel ([13], [14], [11]) hyper-redundant ([9], [10], [12]) active structure, allowing it to bend and move like a tentacle. A possible application is tower-self inspection: a camera mounted on the tower top could be positioned to inspect lower sections (figure 9).

We have implemented a keyframe-type high-level controller which interpolates among manually specified vectors of block-tilt values (figure 9), and we have also implemented a damped-least-squares (DLS) inverse kinematics control, following [29], which allows the user to interactively drag the tower towards a goal configuration (figure 10).

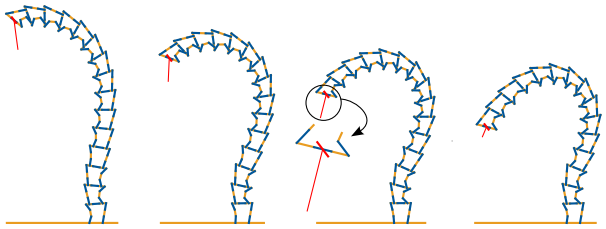

Fig. 10. An example of the simulated 15-block tower under damped-least-squares inverse kinematics control. The user's position and orientation command for the tower top is represented in red. The high-level controller automatically seeks a vector of block-tilts which approaches the command, and then the block controllers translate each such tilt into actual joint commands for the individual active modules.

The high-level controllers were implemented carefully to only rely on a generic abstraction of the block-tilt controller, so we can use the same high-level control code to actuate towers made of different types of blocks. Figure 11 shows an alternate single-chain tower in several poses under the DLS IK control.

VI. Conclusion

We have described a *continuum* of modular truss robots with varying functionality. The simple end of this continuum is a fixed truss with one active climbing unit, and here we have developed hardware called *Shady* for an indoor sun-shading application. We described high-level and low-level algorithms for planning and executing locomotion on trusses, and we presented results of experiments with an initial hardware implementation, indicating the potential of the approach.

The complex end of the continuum is a self-assembling/self-reconfiguring variable-geometry truss composed of active and passive robot modules. We hypothesize that such modules can re-use the same kinematic topology as our truss-climbing robot, and we presented initial results from simulations of

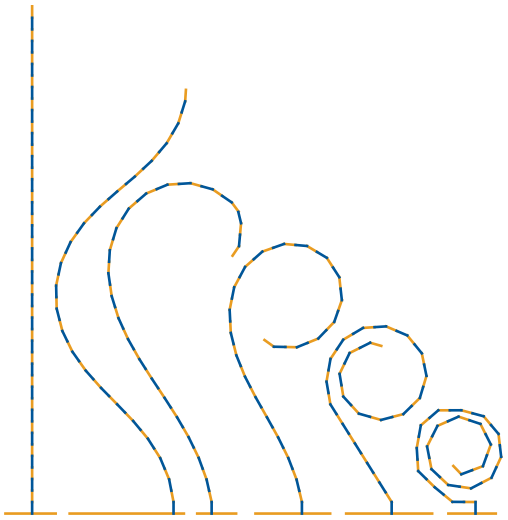

Fig. 11. Several poses of an alternate single-chain tower under damped-least-squares high-level control, showing that the same high-level controller can be used with different block instances, provided they support a common "tilt" abstraction.

systems containing up to 135 active and passive modules performing self-assembly and self-inspection tasks. We also presented our initial work in developing the concept of *hierarchical control* for such systems, where controllers are designed for particular groups of modules and then these groups are treated as monolithic elements by higher-level controllers.

ACKNOWLEDGMENT

The authors would like to thank Peter Osagie for implementing the path planning algorithm described in section IV-C. This work was supported by Intel, NSF IIS-0426838, and MURI ARO W911NF-0510219.

REFERENCES

[1] Satoshi Murata, Haruhisa Kurokawa, Eiichi Yoshida, Kohji Tomita, and Shigeru Kokaji, "A 3-d self-reconfigurable structure," in *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, Leeuven, Belgium, May 1998, pp. 432–439.

[2] Keith Kotay, Daniela Rus, Marsette Vona, and Craig McGray, "The self-reconfiguring robotic molecule: Design and control algorithms," in *Workshop on the Algorithmic Foundations of Robotics*, 1998.

[3] ——, "The self-reconfiguring robotic molecule," in *IEEE International Conference on Robotics and Automation*, 1998.

[4] Keith Kotay, "Self-reconfiguring robots: Designs, algorithms, and applications," Ph.D. dissertation, Dartmouth College, Dec. 2003.

[5] Cem Unsal, Han Kiliccote, and Pradeep Khosla, "A modular self-reconfigurable bipartite robotic system: Implementation and motion planning," *Autonomous Robots*, vol. 10, no. 1, pp. 23–40, Jan. 2001.

[6] David G. Duff, Mark Yim, and Kimon Roufas, "Evolution of polybot: A modular reconfigurable robot," in *Proceedings of the Harmonic Drive International Symposium*, Nagano, Japan, Nov. 2001.

[7] Henrik Hautop Lund, Richard Beck, and Lars Dalgaard, "ATRON hardware modules for self-reconfigurable robotics," in *Proceedings of 10th International Symposium on Artificial Life and Robotics (AROB'10), ISAROB*, Sugisaka and Takaga, Ed., Oita, 2005.

[8] Kasper Støy, "The ATRON self-reconfigurable robot: challenges and future directions," Presentation at the Workshop on Self-reconfigurable Robotics at the Robotics Science and Systems Conference, July 2005.

[9] Gregory S. Chirikjian and Joel W. Burdick, "A hyper-redundant manipulator," *IEEE Robotics & Automation Magazine*, pp. 22–29, Dec. 1994.

[10] Aaron Greenfield, Alfred A. Rizzi, and Howie Choset, "Dynamic ambiguities in frictional rigid-body systems with application to climbing via bracing," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, Apr. 2005, pp. 1959–1964.

[11] Jackrit Suthakorn and Gregory S. Chirikjian, "A new inverse kinematics algorithm for binary manipulators with many actuators," *Advanced Robotics*, vol. 15, no. 2, pp. 225–244, 2001.

[12] A. Wolf, H. B. Brown, R. Casciola, A. Costa, M. werin, E. Shamas, and H. Choset, "A mobile hyper redundant mechanism for search and rescue tasks," in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, Oct. 2003, pp. 2889–2895.

[13] Gregory J. Hamlin and Arthur C. Sanderson, "Tetrobot: A modular approach to parallel robotics," *IEEE Robotics & Automation Magazine*, pp. 42–49, Mar. 1997.

[14] Tanio K. Tanev, "Kinematics of a hybrid (parallel-serial) robot manipulator," *Mechanism and Machine Theory*, vol. 35, pp. 1183–1196, 2000.

[15] Gregory S. Chirikjian, "General methods for computing hyper-redundant manipulator inverse kinematics," in *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japan, 1993, pp. 1067–1073.

[16] Robert L. Williams II and James B. Mayhew IV, "Cartesian control of VGT manipulators applied to DOE hardware," in *Proceedings of the Fifth National Conference on Applied Mechanisms and Robotics*, Cincinnati, OH, Oct. 1997.

[17] Jacob Everist, Kasra Mogharei, Harshit Suri, Nadeesha Ranasinghe, Berok Khoshnevis, Peter Will, and Wei-Min Shen, "A system for in-space assembly," in *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004, pp. 2356–2361.

[18] William Doggett, "Robotic assembly of truss structures for space systems and future research plans," in *IEEE Aerospace Conference Proceedings*, Mar. 2002.

[19] Hisanori Amano, Koichi Osuka, and Tzyh-Jong Tarn, "Development of vertically moving robot with gripping handrails for fire fighting," in *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, HI, 2001, pp. 661–667.

[20] Zaidi Mohd Ripin, Tan Beng Soon, A.B. Abdullah, and Zahurin Samad, "Development of a low-cost modular pole climbing robot," in *TENCON*, vol. I, Kula Lumpur, Malaysia, 2000, pp. 196–200.

[21] Michael Nechyba and Yangsheng Xu, "SM2 for new space station structure: Autonomous locomotion and teleoperation control," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, May 1994, pp. 1765–1770.

[22] ——, "Human-robot cooperation in space: SM2 for new space station structure," *IEEE Robotics and Automation Magazine*, vol. 2, no. 4, pp. 4–11, Dec. 1995.

[23] M. Almonacid, R. J. Saltarén, R. Aracil, and O. Reinoso, "Motion planning of a climbing parallel robot," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 485–489, 2003.

[24] Ben Iannotta, "Creating robots for space repairs," *Aerospace America*, pp. 36–40, May 2005.

[25] Serguei Vassilvitskii, Jeremy Kubica, Elanor Rieffel, John Suh, and Mark Yim, "On the general reconfiguration problem for expanding cube style modular robots," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, Washington, DC, May 2002, pp. 801–808.

[26] An Nguyen, Leonidas J. Guibas, and Mark Yim, "Controlled module density helps reconfiguration planning," in *Proceedings of WAFR 2000: New Directions in Algorithmic and Computational Robotics*, 2001, pp. 23–36.

[27] Daniela Rus and Marsette Vona, "Crystalline robots: Self-reconfiguration with compressible unit modules," *Autonomous Robots*, vol. 10, no. 1, pp. 107–124, Jan. 2001.

[28] A. Pamecha, I. Ebert-Uphoff, and G.S. Chirikjian, "Useful metric for modular robot motion planning," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 4, pp. 531–545, 1997.

[29] Samuel R. Buss, "Introduction to inverse kinematics with jacobian transpose pseudoinverse and damped least squares methods," 17 Apr. 2004, available on the web at http://www.math.ucsd.edu/ sbuss/ResearchWeb/ikmethods/iksurvey.pdf.