Reliable External Actuation for Extending Reachable Robotic Modular Self-reconfiguration

Paul J. White and Mark Yim

GRASP Lab, University of Pennsylvania, Philadelphia, PA, USA
{whitepj,yim}@seas.upenn.edu

Summary. External actuation in self-reconfigurable modular robots promises to allow modules to shrink down in size. Synchronous external motions promise to allow fast convergence and assembly times. XBot is a modular system that uses synchronous external actuation, but has a limited range of reachable configurations stemming from a single motion primitive of a module rotating about another. This paper proposes to extend the motion primitives by using moves with two modules swinging in a dynamic chain. The feasibility of these motion primitives are proven experimentally. A parameterization of the external actuation motion profiles is explored to define a space of physically valid motion profiles. The larger the space, the more robust the motion primitives will be to inexact initial conditions and to imprecision in the external actuation mechanisms. Additionally, this paper proves a configuration of XBot meta-modules can reach any configuration using just these motion primitives.

1 Introduction

One of the grand challenges for modular self-reconfigurable (MSR) robotics is to develop systems with a large number of small modules. Smaller modules can take the shape of a given sized 3D object with higher resolution.

Typically the actuator that causes a module to move (e.g. main motor) consumes much of the modules size, weight and power. Utilizing energy input from external forces allows this actuator to be removed from the system and thus decrease the size. Stochastic MSR systems [1, 9], apply external energy to result in Brownian motion. These systems are sensitive to the motion distribution and can have very long convergence times.

The XBot system was introduced in [10] in which a simple shape memory alloy actuated magnetic latching module demonstrated simple reconfiguration. When viewing the planar XBot system from above, one XBot module appears to be an X with the magnet latches on the corners of the X. A set of XBots appear to be X's arranged on an imaginary lattice or grid. In [10] the XBots

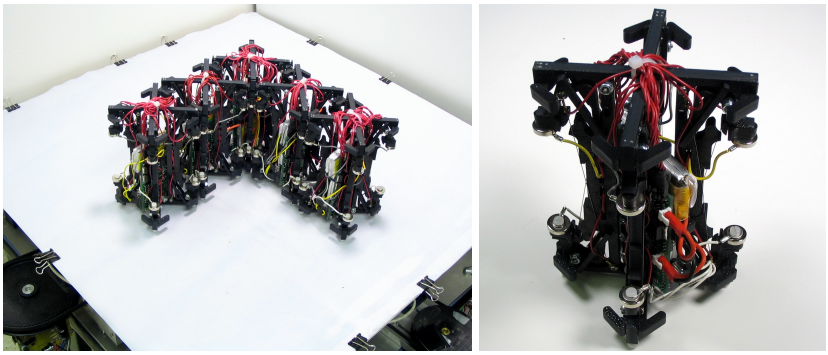


Fig. 1: Left: Five XBots on XY stage table. Right: An XBot.

sit on a moving table with one module fixed to the table and the others sliding on it. By accelerating the table in specific patterns, the inertia of the XBot modules cause them to rotate around the module fixed to the table. The direction of rotation is controlled by releasing one of the magnet latches and using the other as a hinge.

In contrast to stochastic MSR systems, the XBot system uses this applied motion so that it causes reconfigurations to occur deterministically. Thus, the system can reconfigure in a known time, orders of magnitude faster than stochastic methods. This determinism works assuming that the input energy pattern is reliable and consistent. In [10], the one degree of freedom (DOF) table was not consistent. There existed singular positions where the input acceleration was perpendicular to the required motion.

As this paper will show, several reliable and consistent motion patterns exist for simple motion primitives using a 2 DOF motion platform. However, to develop a large reachable space of configurations, a richer set of motion primitives are required.

In many self-reconfiguring lattice systems [11] the modules may combine in almost any fashion so long as the modules lie on a lattice and remain one connected component. In the case of a single cube shaped module moving around a corner as proposed in [10], the space of possible configurations is very limited. For example, a set of modules a single line n modules long has only 9 possible configurations no matter the size of n. This is because only the two end modules can move to one of three positions.

Many MSR systems [4, 7, 6, 8, 2] use meta-modules to relax such motion constraints and expand the reachable configuration space. By adding more motion primitives than the simple pendulum (as we call this one module rotation), the space of possible configurations grows to a more reasonable size. Adding meta-module moves where a chain of two modules moves at the same time does this. Explicitly enumerating the possible configurations is an interesting mathematical problem, it is beyond the scope of this paper but has been the subject of group theory problem for decades [3].

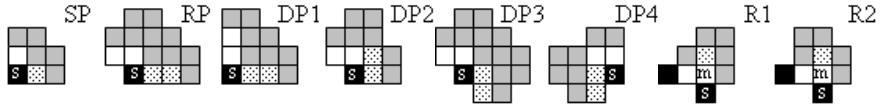


Fig. 2: The set of all motion primitives: simple pendulum (SP), rigid pendulum (RP), the four types of double pendulums (DP1-DP4), the two types of release double pendulums (R1-R2). White modules can reconfigure to hatched cells if gray cells are empty and vice versa. Dark cells are occupied; module pendulum rotates about support cell (dark 's').

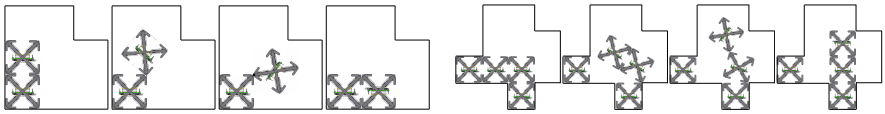


Fig. 3: Reconfiguration sequence for a simple pendulum (3a-3d) and release double pendulum (R2) (3e-3h).

By introducing a larger meta-module of 20 XBots, the Plus Meta-module (PM), the configuration space becomes fully reachable using the new motion primitives as will be shown in Section 3. It is then critical to demonstrate the feasibility of these motion primitives and develop the range motion platform patterns to generate the required motion primitives which is shown in Section 4. A demonstration using these new primitives is shown in Section 5.

2 XBot System and Constraints

An XBot structure is formed on a Teflon table attached to an XY stage. One module is fixed to the center of the table and other modules form a lattice structure stemming out from the fixed module. An XBot module bonds to another module using two pairs of neodymium magnets. To reconfigure, a module breaks one magnet pair bond with its neighbor by retracting the compliant magnet arms using shape memory alloy wires. As the XY stage moves through a carefully designed motion profile, inertial forces cause the module to pivot 180° about the remaining magnet pair which acts as a revolute joint.

As with other lattice style systems, an XBot reconfiguration must adhere to several constraints.

1. *Collision free*: The motion of the modules must not collide with other modules as it moves from one lattice position to a neighboring one
2. *One connected component*: Reconfiguration of the system must maintain one connected component.

[10] discusses the physical constraints on the XBot design including maximum joint force, inertia, and friction. Because a module rotates 180° during reconfiguration the lattice positions have a checkerboard parity. Modules start in one color on the checkerboard and can only move to lattice positions of the same color.

These constraints lead to several conditions that describe the ability of a module to reconfigure:

- *Landlocked*: A module is landlocked if there exists modules occupying lattice positions on opposing sides (e.g. to its north and south or to its east and west.) Note this means the module cannot move in any direction without colliding while it is landlocked.
- *Mobile*: A module is mobile if: (1) the target cell is empty, and (2) the module can move without collision to the target cell.
- *Non-critical*: A module is non-critical if it can be removed from the configuration and still maintain one connected component.
- *Free*: A module is free if it is *mobile* and *non-critical*.

Meta-module moves of two modules greatly expand the set of reachable configurations by adding to the set of mobile neighborhood types. In addition to the simple pendulum move (Figure 2a), modules can move together as a rigid pendulum (Figure 2b) or as a double pendulum (Figures 2c-2h). The white modules in a meta-module can relocate to the hatched cells if the gray cells are empty and vice versa. For a rigid pendulum, the *inner* module rotates about the support module (dark 's' cell in Figure 2) and the *outer* maintains its bonds to the inner module. For a double pendulum, the inner module rotates about the support module and the outer module rotates about one of its two bonds to the inner module. Figures 2g and 2h show two important motion primitives: the inner white module labeled 'm' facilitates the release of the other landlocked module and returns to its original position.

3 Proof of Reachability

Plus Meta-module (PM) consists of 5 groups of 4 XBots arranged in the shape of a plus symbol. It comprises a 3×3 grid of *subcells*: 5 center ones occupied by groups of 4 XBots and 4 unoccupied corner ones. The PM unit cell (Figure 4, left frame) is the smallest building block of a PM lattice configuration. A PM configuration is a connected structure of N PMs formed by placing each PM_i, $i \in \{1, .., N\}$ at a discrete position in the lattice grid (x_i, y_i). The configuration must include a fixed PM with its lattice point defined as the origin. In this section, we will show that any PM lattice configuration can be obtained from any other by applying a sequence of the motions using the primitives introduced above.

The PM unit cell is chosen because the structure of a PM configuration perimeter facilitates relocation and a PM's mobility can be determined from its four adjacent neighbor cells alone.

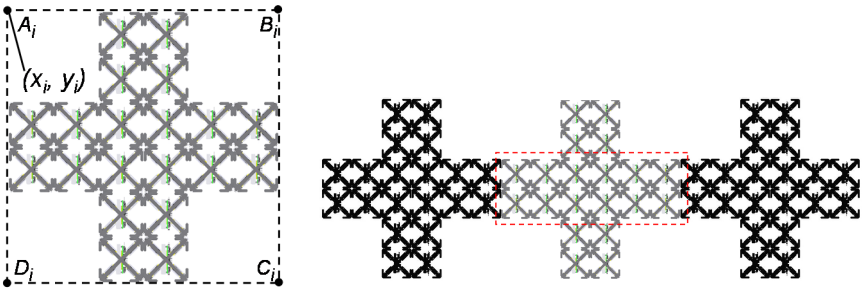


Fig. 4: Left: Plus Meta-module (PM) lattice unit cell with its lattice grid point (x_i, y_i) and lattice cell vertices $\{A_i, B_i, C_i, D_i\}$. Right: The gray PM is land-locked because XBots within the dashed rectangle are stuck between the adjacent (darker)PMs.

(a) (b) (c) (d) (e)

Fig. 5: The five cases for the four lattice cells that share the center vertex. 5a-5d are the four perimeter cases for a PM configuration (e.g. Figure 8)

3.1 Traversing the Perimeter

The perimeters which bound a PM configuration are formed along the edges of occupied lattice cells. In general, a PM configuration has one external perimeter and an arbitrary number of internal perimeters. The following operations illustrated in Figure 5 generate the *perimeter vertices* that define these perimeters. For any of the four lattice vertices ($\{A_i, B_i, C_i, D_i\}$ in Figure 4, left) of an occupied cell, there are 5 possible occupancy types for the surrounding 4 cells as shown in Figure 5. In Figures 5a and 5d, the vertex becomes a "corner" vertex of the perimeter, with the associated PM being a corner PM. The vertex in Figure 5e is not part of the perimeter because it is internal to the configuration. In Figure 5c a chamfer edge is added which is required to produce a valid perimeter curve: every perimeter vertex is the end point of two perimeter line segments. The perimeter vertices associated with a lattice cell include any vertices coincident with any of its lattice edges.

This procedure produces a planar shape with one external perimeter P_E and internal perimeters P_{I_i}. A perimeter is formed from a closed curve of line segments connecting the perimeter vertices, as in the example in Figure 8.

A group of four XBots can *traverse* the perimeter by reconfiguring from one subcell to a neighboring unoccupied cell adjacent to P_E using the motion primitives illustrated in Figure 2. Because of the repeated structure of a PM

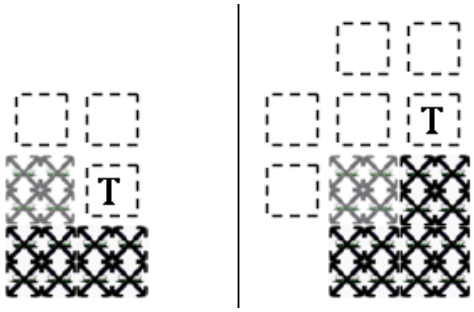


Fig. 6: Traversing primitives: "flat relocation" and "corner relocation". The group of 4 gray XBots can relocate to the target subcell T if there are no XBots in the dashed box subcells.

configuration, there are only two types of *traversing primitives*: "flat relocation" and "corner relocation". A group of four XBots traverse along the "flat" portion of the perimeter (Figure 6, left) and along a diagonal (Figure 6, right) provided the dashed box subcells are empty to avoid collision.

3.2 Mobility

The mobility and connectivity constraints for an individual XBot can be extended to the PM architecture. A PM is mobile if (1) the target PM cell is empty and (2) a sequence exists where all PM modules can move into the target PM cell. As shown in section 3.3, groups of 4 XBots that reach an "exit" subcell E can traverse the perimeter to a target PM cell. Figure 7 shows one sequence using motion primitives such that each group of 4 XBots in the PM can relocate to exit subcell E.

The connectivity constraint applies as follows: a PM is non-critical if all of the modules in it can be removed from the configuration while maintaining one connected component. Since PM cells are treated as a unit, the properties of non-criticality transfer.

The concept of landlocked for determining mobility of a PM also extends. If a set of modules are collinear, then all modules between the two ends are landlocked. All of the internal modules have the property that they cannot move. A PM cell is landlocked if it is between two opposing occupied PM cells as in Figure 4 as the 12 modules in the dashed rectangle are landlocked. Again since the PM cells are treated as a unit, if some of the modules remain landlocked then the PM is considered landlocked.

Theorems 1 and 2 show that a necessary and sufficient condition for a cell to be mobile is that it not be landlocked.

Theorem 1. *A mobile PM is not landlocked.*

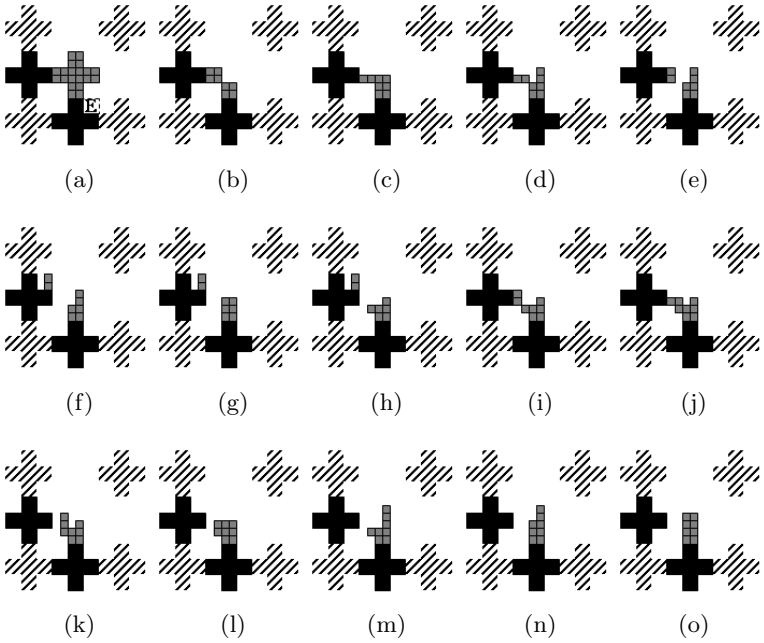


Fig. 7: Exit sequence for PM with two neighbors with exit subcell marked E. A group of four modules that reaches E can traverse the perimeter to the target cell and are removed from the figure for clarity.

Proof. A landlocked PM cannot move entirely as the 12 modules that are landlocked remain so as long as the PM is landlocked, thus it cannot be mobile. Proof by contradiction a mobile PM is not landlocked.

□

Theorem 2. *A PM that is not landlocked is mobile.*

Proof. If a PM is not landlocked, at least 2 of the 4 adjacent and opposing sides are unoccupied. Therefore, a non-landlocked PM either has one neighbor (e.g. to the south) or two non-landlocking neighbors (e.g. one to the south and one to the west). A PM with 3 neighbors must have one pair opposing and thus be landlocked. If the PM has only one neighbor, the traversing primitives can be applied to relocate each group of 4 XBots to the exit subcell. Otherwise, the PM has two neighbors as in Figure 7a. The four hatched PMs diagonally adjacent to the PM emphasize that it does not matter whether or not those cells are occupied to be able to exit the cell. In this case, the first three group of 4 XBots can exit using the traversing primitives to reach E but the final two require a special reconfiguration sequence. Figures 7b through 7o illustrate the motion primitive sequence required to get the final groups of 4 XBots to a state where the traversing cases can be used.

Because in either case all groups of 4 XBots of the PM can reach the exit subcell, the PM is mobile. □

Thus a necessary and sufficient condition for a PM to be mobile is that it be not landlocked. A *mobile cell* is an unoccupied cell that would contain a mobile PM if it were occupied. Since all module motions are physically reversible, a mobile cell is one in which a PM can exit it. By the same reversibility property, unoccupied cells that are between two opposing modules (e.g. cells where a landlock would form if it were filled) cannot be mobile cells.

Another property that is useful to analyze the mobility of PMs is the *external angle* θ_E which is defined to be the angle formed by the perimeter at a perimeter vertex. Figure 5 shows several examples: $\theta_E = +90°$ as in Figure 5a, and likewise, a pair of $+45°$ vertices of a chamfer edge are shown in Figure 5c.

Theorem 3. *If a lattice cell has at least one vertex with $\theta_E \geq +45°$, it is mobile.*

Proof. The external angles θ_E of a lattice cell's perimeter vertices determines its adjacent neighbor state. If a lattice cell's vertex has $\theta_E = +90°$, the two adjacent neighbor cells that share that vertex must be unoccupied. Likewise, a pair of $+45°$ vertices of a chamfer edge as in Figure 5c indicates each of the two adjacent cells that share a chamfer vertex is unoccupied. Thus any cell with at least one vertex with $\theta_E \geq +45°$ is not landlocked and is therefore mobile. □

3.3 Relocating

PM relocation requires three steps: (1) *exiting* its cell, (2) traversing the perimeter, and (3) *filling* the target cell. A PM exits a cell by relocating each group of 4 XBots to an exit subcell E of an adjacent PM. When a group of 4 XBots reaches E, they traverse the perimeter to the fill site F using the traversing primitives. Each group of 4 XBots then fills the *target* cell by forming the PM configuration in the cell.

In the previous section we have shown that a mobile PM can exit a mobile cell. By the reversibility property, a PM can also enter a mobile cell and reform a PM. This shows that the first and last step of relocation will work. The second step of traversing the perimeter can be shown to be valid by using the traversing primitives shown in Figure 6. The shape of the PM was chosen such that a group of 4 XBots can traverse through any unoccupied cell bordering a perimeter.

3.4 Reconfiguring to a line

The configuration space of a PM system is fully reachable if there exists a reconfiguration sequence from any arbitrary configuration C_1 to another

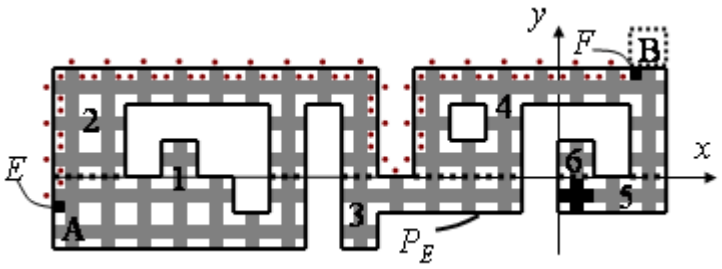


Fig. 8: Sample PM configuration with darkened fixed PM and external perimeter P_E. Each group of 4 XBots in PM A exits to exit subcell E, traverses the dotted path to fill subcell F, and fills B. The dashed segments along the x-axis divide the six nodes of the spanning tree with leaves 1 and 6.

configuration C_2. It is sufficient to show that all PM lattice configurations can reconfigure to a line to demonstrate reachability similar to the method of [5]. The general reconfiguration algorithm involves reconfiguring C_1 to a line and C_2 to a line then simply reversing the sequence for C_2 and concatenating the two.

A naive approach to forming a line might form a line extending in the $\pm y$ direction that includes the fixed PM (as origin). The problem is that since the configuration maybe concave or have holes in it, it is possible that as the line grows, a gap may appear in the line. This gap would include modules on opposing sides leaving an unoccupied landlocked cell - one which is not mobile and thus the line could not be completed. Instead, we introduce a two step process: relocate each PM_i that is above the x-axis ($y_i > 0$) to below the x-axis and then relocate each PM_i to the end of the line that extends in the $+y$ direction. The origin is coincident with the lattice point of the fixed PM which is excluded from the relocation processes. The first step is required to ensure there always exists a mobile target cell at the end of the line.

The x-axis can be considered to divide the perimeter P_E into an upper and lower portion. Figure 8 shows an example where the x-axis divides the single connected component configuration into 6 connected components. Note that the connected component labeled "1" has one PM that is above the x-axis. This is because no segment is added that creates a component with no PMs on the external perimeter. This segmentation results in a grouping of connected PMs that are separated by at least one PM from all other connected components on one side of the x-axis. This segmentation can be represented as a spanning tree T where the connected components are nodes and edges represent the connectivity between two components.

Theorem 4. *Every leaf L in the spanning tree T has at least one free PM on the external perimeter P_E.*

Proof. Let E_L be the set of PMs in L that border the external perimeter P_E. From Theorem 3, we know that PMs that are not between two PMs (i.e. on

Algorithm 1 Leaf Amalgamation

```
while T has more than one node do
  Select leaf L ∈ T
  if L = N_h then
    Move L to form line below c_l ∈ N_l
  else if L = N_l then
    Move L to form line above c_h ∈ N_h
  else
    Move L to form line below c_l ∈ N_l
  end if
end while
```

a corner) are mobile. Thus, if a corner PM is also non-critical it is free. A measure of non-criticality can be determined by examining the Manhattan walk distance $d_{MH}(a, b)$ from PM a to b which is the length of the shortest rectilinear path from a to b. We can choose a PM, e_a, to be the *anchor* PM in $E_L \subset L$ which is connected to the neighbor of L in T. If a PM a has all neighbors n_i where $d_{MH}(n_i, e_a) < d_{MH}(a, e_a)$ then a is non-critical as it cannot disconnect any n_i. Since the external perimeter of L forms a closed polygon whose external angles sum to 360°, there must be at least one PM with an external corner $\theta_E \geq +45°$. This leads to two cases for any perimeter set E_L:

(A) E_L contains a PM a with two external corners and thus only one neighbor b. It is trivially non-critical as the $d_{MH}(a, e_a) = d_{MH}(b, e_a) + 1$. OR:

(B) E_L contains at least two PMs with one external corner. Let $E_C \subset E_L$ be the set of all corner PMs. Either:

1. $\exists e \in E_C$ that has two neighbor PMs n_1 and n_2 such that $d_{MH}(n_1, e_a) < d_{MH}(e, e_a)$ and $d_{MH}(n_2, e_a) < d_{MH}(e, e_a)$. Because the neighbors have lower distance to the anchor PM, e is non-critical. OR:
2. $\exists e \in E_C$ which is part of a cycle and is non-critical because its neighbors both have paths to the anchor PM if it is removed.

In any of these cases, there exists a free (mobile and non-critical) PM. □

Algorithm 1 combines the leaves of T one by one until there exists one leaf below the x-axis guaranteeing a clear path for a line to form above the fixed PM. The node $N_h \in T$ contains the "highest rightmost" PM c_h satisfying $y_h \geq y_i \forall c_i \in C$ and $x_h > x_j \forall c_j$ such that $y_j = y_h$. Likewise, the node $N_l \in T$ contains the "lowest rightmost" PM c_l satisfying $y_l \leq y_i \forall c_i \in C$ and $x_l > x_j \forall c_j$ such that $y_j = y_l$. Because c_h and c_l are at the extremes of the configuration, there exists a mobile cell above c_h and below c_l.

By Theorem 4, every leaf has a free PM. A leaf *moves* above or below a node by relocating each PM to form a vertical line above or below the node respectively. When a leaf moves above node N_h, each PM in the leaf relocates to the target cell above c_h and becomes the new "highest rightmost" PM c_h. A

leaf moves below node N_l in a similar manner. When a PM in a leaf relocates, it may split the leaf into two leaves. A move is completed after all PMs in the original leaf have relocated. Algorithm 1 continues moving leaves until all PMs form one node below the x-axis.

Theorem 5. *Any PM configuration with all PMs below the x-axis can be reconfigured to a line.*

Proof. Given that all PMs are below the x-axis, T simply has one leaf which has a free PM by Theorem 4. Because each PM c_i in the leaf satisfies $y_i \leq 0$, the cell above the fixed PM is a mobile target cell. Therefore, each PM below the x-axis can relocate to form a line above the fixed PM.

□

Therefore, all PM lattice configurations of XBots are reachable by first reconfiguring C_1 to a line then to C_2. This second reconfiguration sequence is simply the reverse sequence that reconfigures C_2 to a line.

4 Reliability of External Actuation

4.1 Experiments

An initial analysis and manual exploration of the modules on the 2DOF movable table, showed that moving the table in a rapid circular path would result in modules reliably reconfiguring. The table and the fixed module move in a circle and the centripetal force causes the module pendulum to rotate 180 °. It is likely that there are very many other non-circular paths, however, circles are easy to implement and intuitive to understand.

In the case of the simple pendulum and the rigid pendulum, the table's motion profile is constrained to move in a circle of a constant radius and angular rate. For each trial, the radius and the angular rate are varied in order to determine the set of (radius, angular rate) pairs that successfully cause the module(s) to reconfigure.

It was found experimentally that it is difficult to reconfigure a double pendulum meta-module using only circular motion profiles. An elliptic motion profile (examples in Figure 10) is used to increase the parameter space to include the length of the major and minor axis of the ellipse, the angle of the major axis, and the period of the motion cycle.

4.2 Results

For the simple and rigid pendulum cases, for each motion profile radius, an initial angular rate is chosen that is insufficiently large to cause the module pendulum to reconfigure. For each successive trial, the angular rate is increased until the reconfiguration occurs reliably. For example, the left frame

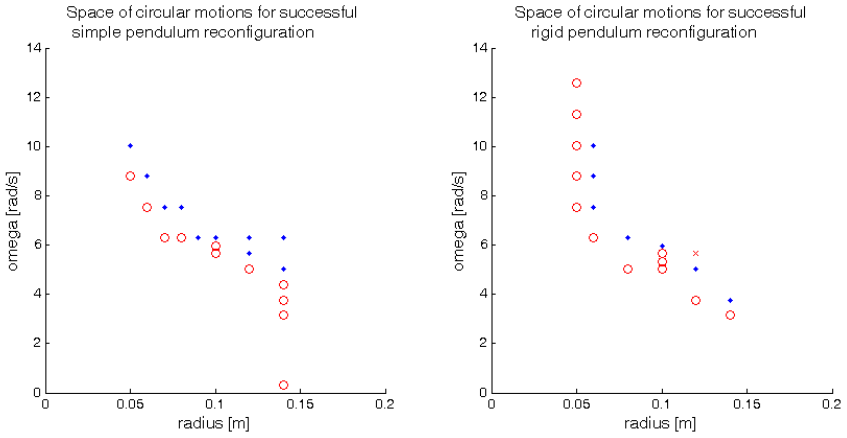


Fig. 9: Empirical partition of motion profile parameter space. The dots represent the (radius, angular rate) pairs that reliably cause the simple pendulum and rigid pendulum reconfiguration primitives to occur. Circles indicate the angular rate was too low for the given radius.

of Figure 9 presents the data for the simple pendulum motion profile. The first trial with a radius of 0.14m and 0.31 rad/s, has insufficient energy to cause the module to fully rotate 180°. Subsequent trials at that radius are performed until the simple pendulum successfully completes a reconfiguration (at 5.0 rad/s). This procedure is repeated for decreasing radius values in order to find the lower boundary in the (radius, angular rate) parameter space between successful and unsuccessful motion profiles. Note that in each case in Figure 9 as the radius R decreases the angular rate increases as $1/\sqrt{R}$. This relationship matches well with the theoretical expression for the centripetal force acting at the joint of the reconfiguring module given by: $F_{centripetal} \propto \omega^2 R$.

There exists an upper bound on the parameter space defined by the maximum bonding force of the magnet pair bond as shown in [10]. The current generation of XBot modules have a maximum bonding force of 9.5 N for one magnet pair. The x in the second frame of Figure 9 represents a failure due to inertial forces at the magnet pair joint exceeding the maximum bonding force limit causing the rigid pendulum to detach from the table fixed module.

Therefore, the external actuation of the system can be tuned such that each type of reconfiguration primitive can occur reliably and deterministically. Additionally, each type of motion primitive is guaranteed to occur within one cycle of the corresponding motion profile. A full motion profile period is defined by concatenating motion profile cycles in a continuous sequence that allows for all types of reconfiguration primitives and directions to occur.

Several double pendulum trials are run with varying ellipse properties and cycle frequencies. Because of the large parameter space and the added complexity of the double pendulum, the parameter space is not easily partitioned

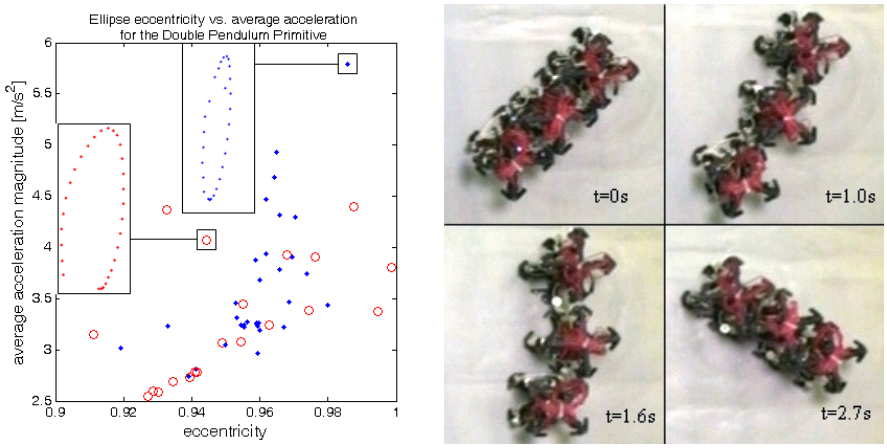


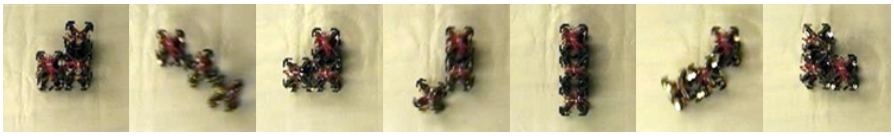
Fig. 10: Left: Plot of successful (dots) and failed (circles) double pendulum reconfigurations for varying ellipse eccentricities and average table accelerations. Two example elliptical motion profiles are shown as insets. Right: A double pendulum reconfiguration sequence.

into regions that guarantee reconfiguration. Figure 10 plots average acceleration of the table against the eccentricity of the ellipse. For eccentricity values between 0.95 and 0.98, the motion profile is sufficient to reconfigure the double pendulum with the exception of a few failure cases. However, by increasing the cycle frequency of the ellipse motion the average acceleration of the table increases and reduces the number of failed reconfigurations. The reliability of a valid motion profile is demonstrated by 10 consecutive successful reconfigurations indicated in Figure 10 by the cluster of points near eccentricity of .96 and average acceleration of $3.2m/s^2$.

5 Demonstration

Two example reconfigurations sequences demonstrate how external actuation reconfigures the XBot system. A demonstration consists of a sequence of motion primitives each paired with a XY stage motion profile. The ellipse properties for each motion profile are determined using a dynamics simulator written in MATLAB.

All modules run the same software and store a list of configurations which defines the reconfiguration sequence. The fixed module initiates each reconfiguration by passing a token with the desired state through the configuration. Each motion primitive has a leader module (the inner module for rigid and double pendulums) that coordinates the reconfiguration. When the leader receives the token, it determines the magnet bonds to break based on the motion primitive and, if necessary, tells neighboring modules which magnet bonds to break. Then, the XY stage executes the motion profile that recon-



(a) (b) (c) (d) (e) (f) (g)

Fig. 11: Demonstration reconfiguration sequence

figures the module(s). After it verifies the reconfiguration has completed, the leader passes the token back to the fixed module which initiates the next reconfiguration sequence.

The reconfiguration demonstrations run open loop: the modules do not coordinate with the XY stage motion controller. For each reconfiguration, the XY stage controller waits a predetermined amount of time before running the motion profile. The waiting period provides sufficient time for the token to reach the leader module and for the necessary magnet bonds to break. Future work includes closing the loop via communication between the fixed module and the motion controller.

The first reconfiguration example demonstrates the reliability of the 2 DOF stage over the 1 DOF stage used in [10]. One XBot performs four consecutive simple pendulum reconfigurations about the table fixed module. The XBot attached to the fixed module determines its position and breaks the magnet bond such that it rotates counter-clockwise. This repeats three more times. In contrast to 1 DOF stage used in [10], the motions of the XY stage are sufficient to reliably rotate the module in one table motion.

The second example shown in Figure 11 demonstrates the three general types of motion primitives. The leftmost module in 11a is fixed to the table. The reconfiguration sequences consists of a double pendulum (DP3) (11a-11c), a single pendulum (11c-11e), and a rigid pendulum (11e-11g).

6 Conclusion

It has been proposed that synchronous external actuation can shrink down module size and allow fast shape convergence. XBots has been shown as one example system that can use external actuation however, a single module rotating about another as a motion primitive has a small space of possible shapes into which a system can morph. Extending the motion primitives with a double module move greatly extends that shape. Moreover, a configuration of Plus Meta-modules is fully reachable because of its structure and the added motion primitives.

These motion primitives have been demonstrated to be feasible. The parameter space through which these motion primitives can be generated has been explored to find base motion patterns that will robustly achieve the desired motion primitive.

Future work includes allowing coordination between the fixed module and the XY Stage motion controller. Also the space of motion profiles will be explored to find the optimal motion pattern. The Plus Meta-module is relatively large (requiring 20 modules); it is likely that there are smaller meta-modules that will still result in a fully reachable space with the presented motion primitives.

7 Acknowledgment

The authors wish to thank Ying Zhang for her valuable comments on a draft of the paper.

References

1. J. Bishop, S. Burden, E. Klavins, R. Kreisberg, W. Malone, N. Napp, and T. Nguyen. Self-organizing programmable parts. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3684–3691, Edmonton, Canada, 2005.
2. D. J. Christensen, E. H. Ostergaard, and H. H. Lund. Metamodule control for the atron self-reconfigurable robotic system. *Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8)*, pages 685–692, 2004.
3. F. Harary and E. M. Palmer. *Graphical Enumeration*. Academic Press, 1973.
4. C. McGray and D. Rus. Self-reconfiguring molecules as 3d metamorphic robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 837–842, Victoria, Canada, October 1998.
5. D. Rus and M. Vona. Self-reconfiguration planning with compressible unit modules. In *Proceedings of IEEE/RSJ IEEE International Conference on Robotics and Automation*, volume 4, pages 2513–2520, Detroit, MI, USA, 1999.
6. D. Rus and M. Vona. Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1):107–124, 2001.
7. C. Unsal and P. K. Khosla. A multi-layered planner for self-reconfiguration of a uniform group of i-cube modules. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 598–605, Maui, HI, USA, 2001.
8. S. Vassilvitskii, M. Yim, and J. Suh. A complete, local and parallel reconfiguration algorithm for cube style modular robots. In *Proceedings of IEEE/RSJ IEEE International Conference on Robotics and Automation*, volume 1, pages 117–125, Washington, DC, United States, 2002.
9. P. White, V. Zykov, J. Bongard, and H. Lipson. Three dimensional stochastic reconfiguration of modular robots. In *Robotics: Science and Systems*, pages 161–168, Cambridge, MA, 2005.
10. P. J. White and M. Yim. Scalable modular self-reconfigurable robots using external actuation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2773–2778, San Diego, CA, USA, 2007.
11. M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics and Automation Magazine*, 14(1):43, 2007.