Learning Locomotion Gait of Lattice-Based Modular Robots by Demonstration

Seung-kook Yun

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
yunsk@mit.edu

***Abstract*— We explore supervised learning for a collective motion of lattice based modular robots that can access only local information. In our proposed learning algorithm, stable locomotion gaits of small size configurations are given to modules by human demonstration. Support vector machine and regularized least square are used to train the manually data given, and performances are compared. We show our method scales in various configurations.**

I. Introduction

The goal of this paper is to develop efficient algorithms for automatic design of a distributed controller for lattice-based self-reconfiguring modular robots. The self-reconfiguring robots are a group of identical and independent modules that may have actuators and sensors, and they have had attention because they can be versatile in any tasks. Over past 15 years many success for self-reconfiguring robots have been built such as [1]–[5] They can reconfigure themselves into a wanted structure for a given task such as manipulation [6] and locomotion [1], [3], [5], [7]. The lattice-based robots are one of the most general modular robots. They move by filling and emptying their space, and several control laws [7]–[10] have been proposed for a given task.

Parallelism in controlling modular robots is important, because the systems can be composed of a great number of modules and a central controller would not be efficient. The parallelism requires that controllers should be distributed so that each robot works with local information in parallel, and here arises a challenge of reconfiguring robots without any global knowledge. Most of the systems have their own reconfiguration algorithms with a central or a distributed controller, which are carefully designed and verified. A generic distributed algorithm for lattice-based robots was described in [7]. It uses a rule-based approach where the rules were manually developed in a task-divided way for locomotion, reconfiguration, self-repair and self-replication. In their algorithm, each module decides a direction of movement based on occupancy of the neighborhood: no cell, a cell or an obstacle (See Figure 1).

Developing these controllers manually is time consuming and proving that they are correct is very challenging. Some attempts have been made at automating controller design. MTRNAS-II incorporated a central pattern generator(CPG) [8]. Nagpal proposed a programmable self-assembly that translate a global plan into local rules [9]. Varshavskaya [10] proposed a policy search algorithm based on gradient ascent in policy space.

We propose an alternative method for constructing distributed controllers for locomotion of lattice-based modules that uses supervised learning. Inspired by the fact that global reconfiguration can look much simpler despite of complicated local decisions (note that locomotion of a cube - composed of lattice-based robot - can be done by simply moving the most outer robots over the inner ones [7],) The method requires a small amount of training data, it can be implemented easily, and it finds a solution fast.

This paper introduces a framework of demonstration-and-imitation. For rectangular configurations with a small number of robots, a human operator demonstrates how they can perform a given task such as locomotion. while robots collect local data and directions of their movement. After the demonstration, the robots infer a distributed controllers for the task by support vector machine and regularized least square. It is very important that the resultant controller scales, because configuration for demonstration should be small enough for a human to teach for time efficiency. We compare cases of designing a state, different learning methods and variation with parameters of the methods. The experiment shows that the proposed algorithm scales well for locomotion on the flat surface and over obstacles, even though the demonstration are performed with small number of configurations.

This paper is organized as follows. Section II presents a goal of this work and formulates the problem with a proposed solution. Section III shows how robots are trained, and the results are shown and discussed in Section IV. Locomotion over obstacles is learned in Section V. Section VI concludes this work.

II. Problem Formulation

The robot is a lattice system and each unit in the system follows the abstract cube model: it can translate and make convex and concave transition on a substrate and it cat see its neighbors when in close proximity.

A. System framework

We use the robot module in [7]. The robot can not move by itself but moves with a help from the adjacent robots as shown in Figure 2(a). The robot system lies to maintain connectivity. Each module has 8 directional movements

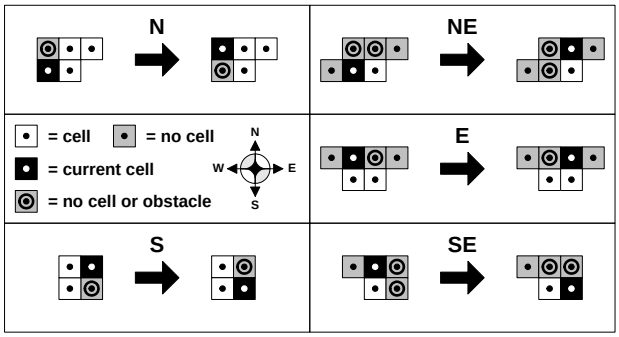


Fig. 1. Five rules for eastward locomotion of lattice-based self reconfigurable robots. Reprinted from [7].

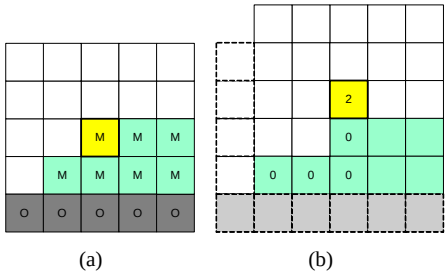


Fig. 3. A movement of demonstration: The yellow cube is the module we move. M denotes a robot and O is an obstacle. (a) The yellow module has a 5x5 view (b) After it moved to NE direction, it is given the label 2(NE) wheres the adjacent robots get the label 0(STAY). The dotted region is now out of sight.

labeled as Figure 2(b). Physical systems that fit this model include [4], [7], [11], [12].

We wish to develop a distributed decentralized controller for robustness and scalability. We assume that each module has limited range of sensing and communication, so that controller will use only local information such as the location of neighborhood modules. Figure 1 is an example of the rules for eastward locomotion in [7].

B. Learning by demonstration for locomotion

The following assumptions are used in our approach:

1) The robot moves in 2D.
2) A module can sense whether neighboring slot is empty or occupied.
3) Each module has a 5×5 view of the lattice neighborhood.
4) A module can sense a direction of its motion in demonstration.
5) Modules take turns

The second assumption required a sensor for detecting obstacles. The view is given in a synchronous way, because of the last assumption. The third assumption can be realized by local communication among the contacted robots. To fully access the 5x5 view, a robot sensor should be able to check distance from obstacles. The 5×5 window was selected as a sensing range since the hand-rules of [7] use the same size. Sensing motion of an actuator in the module enables the fourth assumption. The last assumption was introduced to manage the computation needs of learning.

Our algorithm starts with a set of motions for a stable gait successively given by an operator. An example of a motion is illustrated in Figure 3. The yellow robot learns a northeast(NE) motion with a given state of the 5×5 window in Figure 3(a). The modules have to move continuously with this method. However, learning to stop is also important. To learn stopping, we let robots learn not to move when the adjacent robot has disappeared. For example in Figure 3(b), the four adjacent robots of the yellow robot learn to stay(label 0) by checking that one of the adjacent robot moved away.

III. Learning Manually Given Sequences

In this section, we describe a method that train a modular robot system using a small number of manually developed motion sequences. We then use support vector machine(SVM) and joint boosting to learn the relation between the features and labels.

A. Training sequences

In order to develop a scalable approach to learning robot controllers we explore what we can do with a small amount of training data. The training data comes from manually-driven locomotion of 2×2, 3×3 and 4×4 modular lattice robot systems. Figure 4 shows the sequence of 4×4 modules. Only the outer robots move along the inner robots so that they maintain the original configuration. The sequences for 2×2 and 3×3 modules are generated in the same way. We generate 39 movements and there are 170 learned labels (including *stay*(0)).

B. Feature definition and selection

A training sample is a 5×5 window centered at a module that manually learns a label. For each grid in the window, there are three cases: empty, robot, or obstacle. To distinguish them, we use a boolean feature defined as:

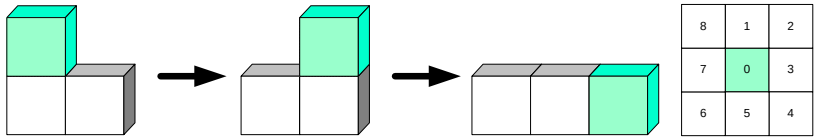
EMPTY=$[0 \quad 0]^T$; ROBOT=$[1 \quad 0]^T$; OBSTACLE=$[0 \quad 1]^T$

The number of training data is very small as compared to the size of the state. Therefore, a classifier should be able to learn from the sparse training set. To get a feature vector, we concatenate all the columns of the 5×5 window into a single column which has 50×1 dimenstion.

C. Tools for learning

In order to learn a classifier, we use SVM and regularized least square (RLS) with the RBF kernel.

1) Support vector machine: We implemented the SVM with the radial basis function(RBF) kernel. The RBF is a natural choice because our training data is highly non-linear. The OSUSVM toolbox [**?**] is used to classify multi-class labels. It implements one-to-one classification for multi-class so that it does total $k(k-1)/2$ times classifications where k is a number of the classes.



(a) (b)

Fig. 2. (a) A kinematic model of the lattice-based reconfigurable robots: a sliding and convex transition (b) The labels of the classes are defined by the moving direction

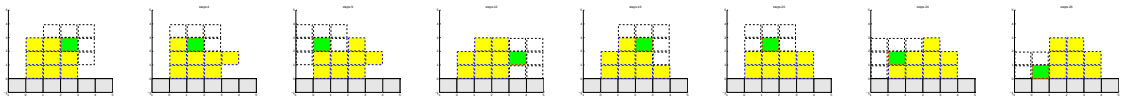


Fig. 4. Snapshots of a manually-driven sequence for 3×3 modules. Green modules are selected by an operator. Dotted regions denote the possible next locations.

2) Regularized least square:

D. Choice of the best parameters

To evaluate the best parameters from the SVM and the RLS, we implement leave-one-out-cross-validation(LOOCV). The LOOCV is calculated as:

$$LOOCV = \frac{1}{n}\sum_{i=1}^{n} Loss(y_i, f(x_i; w^{-i})), \tag{1}$$

where n is a number of the training data, $f(x_i; w^{-i})$ is a classifier trained without the i-th data, and $Loss(\cdot)$ is a zero-one loss function. Various combinations of (C, γ) are tested by the LOOCV, where C is a penalty for margin constraint and γ is a width constant for a RBK kernel.

$$RBF(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \tag{2}$$

E. Self-training with various configurations

Our global test metricis the stability of the learned gait. Failures include disconnecting the modules in the robot and stuck configurations.

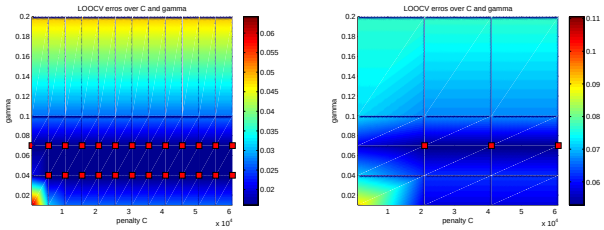
IV. Experiments

Based on the criteria in Section III, we choose the best parameters and test the classifiers in rectangular configurations. We exprimented with 25 test sets of rectangular configurations which are combinations of following rows and columns: $row = \{2,4,6,8\}$, $column = \{2,4,6,8\}$. The test stops when there is a failure or it makes successful movements so that the x-position of the mass center advances by 5 grids.

A. Iterations for SVM

Figure 5 depicts the iterated results of the LOOCV over following set of C and γ: $C \in [1000, 60000]$ $\gamma \in [0.01, 0.2]$.

Classes correspond to the directions in Each motion direction in Figure 2(b).



(a) (b)

Fig. 5. LOOCV graphs for the SVM classifier over iterations. Figures are interpolated to give a smoother view. The red squares direct the parameters for min-LOOCV over C and γ.

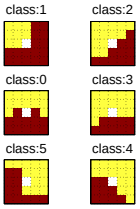
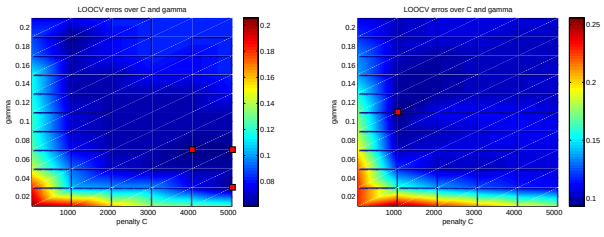


Fig. 6. Superposed support vectors. The yellow blocks denote empty and the brown is a robot or an obstacle. The white blocks are not included in the classifiers.



(a) (b)

Fig. 7. LOOCV graphs for the RLS classifier over iterations.

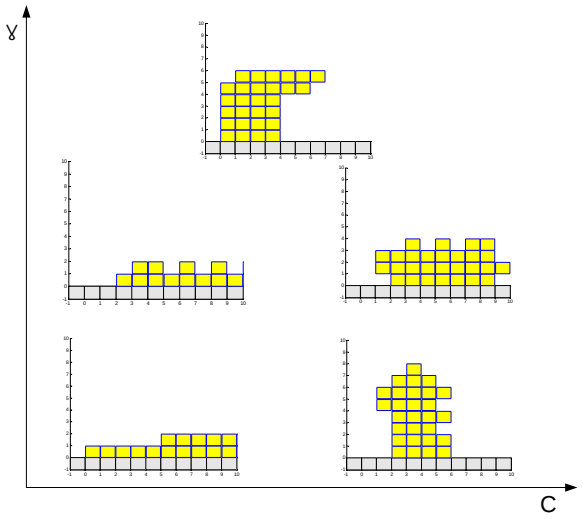


Fig. 10. The learned gaits over the parameter C and γ. The original configuration is 7×4.

B. Iterations for RLS

C. Learned gait

Figure 7 and Figure 8 show the learned gait of 10×10 modules using the best SVM and RLS classifiers. The RLS controllers lead to gaits of 4×4 modules. Thus the robot is morphed into a robot of 4 rows which can move with a stable gait. In contrast, boosting maintains the original 10×10 configuration and controls robot motion by moving the outer modules. If the initial robot has fewer than 4 rows, the robot follows a stable gait of 3×3 or 2×2 modules. Note that each module sees only within a window of 2-blocks centered on itself.

D. Discussion

E. Proof of convergence

The learned classifier succeeded in having stable gaits in all the 49 configurations used for testing locomotion without obstacles. The result imply that we have not explicitly but experimentally proved that the classifier works for any rectangular configurations.

1) How do parameters change gaits?: C and γ in the SVM, and number of the weak classifiers in the boosting affects the gaits. Figure 9 shows the gaits of 7×4 robots, over the parameters. A large γ leads robots to be stuck as in the upper of Figure 9, because the classifier would almost store the given training data and it may not make the right decision even with only a small deviation. In the middle range of the γ, the gaits form horizontally long rows. C appears to decide the number of rows in the final gaits. A large C tends to prevent modules from moving(non-zero labels) so that they have more rows than the case with a smaller C. With a low γ, modules can move more freely. However, modules have a high chance to have a non-connecting failure. In the lower-right figure, the robots maintain the original configuration while moving eastward.

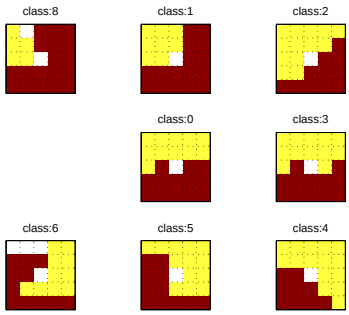


Fig. 12. Superposed support vectors with the linear and the boolean feature to walk over obstacles. The classifier has two new motions(NW, SW).

F. Comparison with other automated controller designs

There are only a few approaches to design a local controller in an automated way. The central pattern generator in [8] is an adaptive controller that depends on starting configurations and has a central brain to govern local pattern generators. Therefore, it may not scale well.

Reinforcement learning of decentralized controllers for lattice-based modules is proposed in [10]. The method is not task-specific and sometimes more efficient than hand-written rules in speed of gaits, and it also use only 3×3 window for training. However, it requires a great number of iteration even though incorporating global agreement among the modules, and scalability of the algorithm is not proven yet.

Our proposed algorithm has limitations in that it uses the 5×5 view and modules should move one by one, which requires more communications than [10] and slow down advance. The main advantage of our algorithm is that it scales very well. Training from only 3 configurations generates stable gaits for any rectangular configurations as we will see in Section V. Controllers are very quickly learned (less than a few seconds in MATLAB with Intel-core 6300 1.86GHz), since size of training data is small and we have used standard machine learning tools as the SVM and the boosting.

V. Extension: Locomotion over Obstacles

An extension of the proposed algorithm to learn a stable gait over obstacles. There also exist the hand-made rules for that [7]. Robots learn only a single series of motions in which 4x4 robots are moved over a single 2x1-sized obstacle as shown in Figure 10. 847 training data are added to the original data, and 200 of them are non-zero labels. The learned classifier with the least test error is shown in Figure 11 with two more labels - NW and SW. The learned gait of 7×7 modules over 4 obstacles is displayed in Figure 12. The modules move like flowing over the obstacles.

VI. Conclusion

We propose a new framework to learn a collective locomotion of lattice based self-reconfigurable robots by manually given stable gaits. The SVM and the RLS are implemented

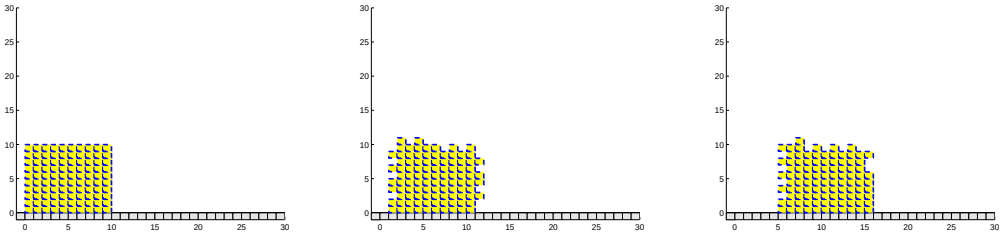


Fig. 8. Learned motion of 10×10 modules generated by the SVM classifier with the boolean feature. They maintain thier original configuration while moving. The robots made total 1000 movements.

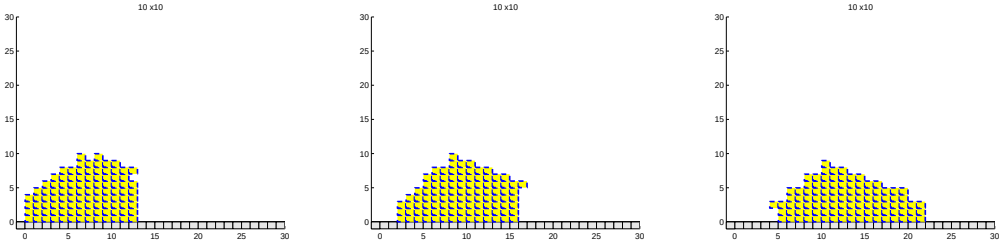


Fig. 9. Learned motion of 10×10 modules generated by the RLS classifier. The robots made total 1000 movements.

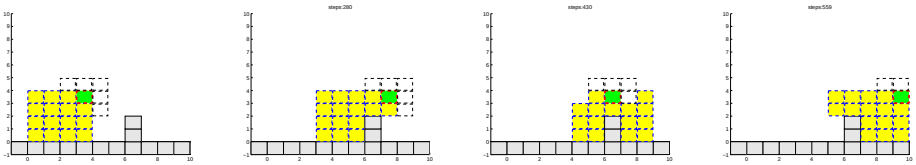


Fig. 11. snapshots of the learning sequence for 4×4 modules over a single obstacle.

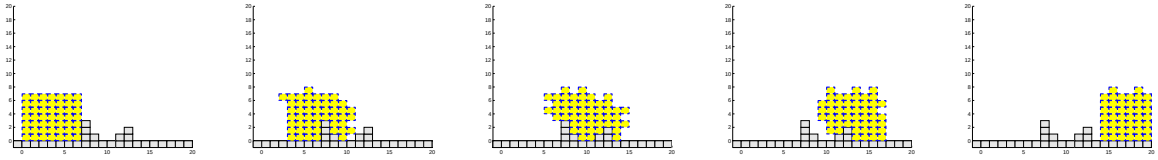


Fig. 13. Locomotion of 7×7 modules over four obstacles: the classifier was learned in 4×4 modules with a single obstacle. Total 1500 steps are made in the simulation. The boolean features are used.

to classify the training data with the linear and boolean features. The LOOCV is used to find the best parameters of the classifier.

The learned classifiers scale very well even though they are trained in small sized configurations. We can learn stable gaits in any rectangular forms of modules, and the SVM and the boosting show similar performances in our problem. We found that the LOOCV might not be so useful in choosing the best parameters, though we need to more think of what is a reasonable measure of a global goal with only local data.

Much work remains in future such as applying the algorithm for more situations as climbing and tunneling, and implmentation on hardwares.

REFERENCES

[1] Satoshi Murata, Haruhisa Kurokawa, and Shigeru Kokaji, "Self-assembling machine," in *IEEE Conference on Robotics and Automation*, 1994.

[2] Hosokawa, K, Tsujimori, T., Fujii, T., Kaetsu, H., Asama, H., Koruda, Y., and Endo, I., "elf-organizing collective robots with morphogenesis in a vertical plane." in *IEEE Conference on Robotics and Automation*, 1998.

[3] Amit Pamecha, Chih-Jung Chiang, David Stein, and Gregory Chirikjian, "Design and implementation of metamorphic robots," in *The 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, Irvine, CA, Aug. 1996.

[4] 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.

[5] Haruhisa Kurokawa, Akiya Kamimura, Eiichi Yoshida, Kohji Tomita, Satoshi Murata, and Shigeru Kokaji, "Self-reconfigurable modular robot (M-TRAN) and its motion design," in *Seventh International Con-*

ference on Control, Automation, Robotics And Vision (ICARCV'02), Singapore, Dec. 2002, pp. 51–56.

[6] Seung-kook Yun and Daniela Rus, "Self assembly of modular manipulators with active and passive modules," in *Proc. of IEEE/RSJ IEEE International Conference on Robotics and Automation (to appear)*, Pasadena, CA, May 2008.

[7] Z. Butler, K. Kotay, D. Rus, and K. Tomita, "Generic distributed control for locomotion with self-reconfiguring robots," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 919–938, 2004.

[8] Akiya Kamimura, Haruhisa Kurokawa, Eiichi Yoshida, Kohji Tomita, Satoshi Murata, and Shigeru Kokaji, "Automatic locomotion pattern generation for modular robots," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, Sept. 2003, pp. 714–720.

[9] Nagpal, R., "Programmable self-assembly using biologically-inspired multiagent control," in *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, 2002.

[10] Paulina Varshavskaya, "Distributed reinforcement learning for self-reconfiguring modular robots," Ph.D. dissertation, Massachusetts Institute of Technology, Sept. 2007.

[11] Akiya Kamimura, Haruhisa Kurokawa, Eiichi Yoshida, Satoshi Murata, Kohji Tomita, and Shigeru Kokaji, "Automatic locomotion design and experiments for a modular robotic system," *IEEE/ASME Transactions on Mechatronics*, vol. 10, no. 3, pp. 314–325, June 2005.

[12] W.-M. Shen, M. Krivokon, H. Chiu, J. Everist, M. Rubenstein, and J. Venkatesh, "Multimode locomotion via superbot reconfigurable robots," *Auton. Robots*, vol. 20, no. 2, pp. 165–177, 2006.