Isomorphic Gait Execution in Homogeneous Modular Robots

Michael Park Sachin Chitta Mark Yim
GRASP Laboratory, University of Pennsylvania, USA
E-mail: {parkmich,sachinc,yim}@grasp.upenn.edu

***Abstract*— In this workshop session, we present and demonstrate the implementation of a centralized algorithm that incorporates neighbor data to detect the configuration of any modular CKbot structure and execute a corresponding predetermined gait for the configuration. An interesting feature of this algorithm is that it permutes the module node labelings to account for arrangements isomorphic to canonical configurations that correspond to a gait. We introduce the Port Adjacency Matrix as a means of organizing configuration information as well as the Characteristic Polynomial of this matrix as a method for determining when two configurations are isomorphic. We also discuss determining isomorphism mappings off-board using the efficient Nauty algorithm for structures with greater numbers of modules. Lastly, we discuss how feedback can be incorporated into the control architecture to allow for multiple gaits for a single configuration.**

I. Introduction

The problem of controlling a modular robotic system has been an active area of research for over a decade. Centralized control [1], distributed control [2], [3], and genetic algorithm inspired control [4], have been studied. In this work, we consider a centralized controller whose primary function is to interpret structural information and map the corresponding actuator data to modules in the given configuration.

For a structure with unique module IDs, this problem naturally reduces to graph isomorphism problem. While Chen enumerated the unique shapes [5], we use the matrix characteristic polynomial as a filter and relabeling operations to find isomorphic mappings to a control for a known configuration. In [6] Castano and Will introduce this problem and a solution for the Conro modular robot using Heuristics. They also mention a more general solution using a graph isomorphism solver. However they did not implement it. This work uses the more generic solution. The platform for is CKbot (Figure 1), complete with IR TX/RX for inter-module communication. Local data is shared through the USART ports on a PIC microcontroller and data between modules and the controller is built upon the Robotics Bus [7].

II. Configuration Recognition for Gait Execution

The focus of this work is on the recognition of configurations using a Port Adjacency Matrix (PAM). The PAM is the same as an N x N Adjacency Matrix except that instead of only 1's and 0's signifying connectivity, numbers 1 - 7 signify which exact ports of the connected modules are shared (consequently the PAM is not generally symmetric). Hence, the PAM specifies the exact connection and inter-module orientation of a CKbot system. The controller organizes the neighbor data through local communications from neighbors into a PAM.

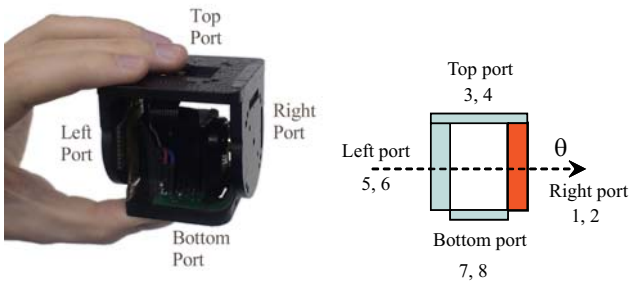

Fig. 1. One CKbot module with a schematic representation. The arrow indicates the rotational axis, the numbers are the port identification numbers assigned to each port

A. Configuration and ID Specific Algorithm

The precise configuration is then compared with a library of predetermined configurations for which a corresponding control scheme for each module is specified in the form of a gait table [8]. If the detected PAM matches a PAM in the library, then the corresponding gait is executed and the processing stage of the controller yields to the control stage that simply sends out angular control messages to each module over the Robotics Bus.

B. Configuration and ID Independent Algorithm

In theory, the above mentioned routine is all that is necessary to carry out the control for any CKbot system. However, in practice this is unsatisfactory in two ways:

1) There are on the order of N! ways of arranging N modules, so creating N! library entries for each permutation of the same shape is wasteful and impractical as N becomes large.
2) Each CKbot has a unique node ID, so two configurations in the same shape, but comprised of modules with different IDs would require different gaits.

The second point is easily solved by mapping *physical* module IDs to *logical* IDs. This entails simply ordering the physical IDs from lowest to highest and then mapping these IDs to logical IDs that count up incrementally regardless of the spacing between the physical IDs. For example, both sets of physical IDs 2A, 2B, 2D and 18, 21, 35 would be mapped to logical IDs 0, 1, 2. This provides the required generality for isomorphic permutation mappings needed to solve the first point. How we address the first point is best shown through pseudo-code (Figure 2).

The basic idea of this function is this: it checks to see if the current configuration is isomorphic to any configuration

```
for (  i=0 to maximum number of configurations
       in the library)
{
   if(characteristic polynomial[i] =
      characteristic polynomial[current
                                configuration])
   {
      for (j=0 to number of modules)
      {
         if (   permutation map[j] =
                library_configuration[i])
         {
            load corresponding gait;
            map permuted logical IDs;
         }
      }
   }
}
```

Fig. 2. pseudocode.

in the library. It does this by comparing the characteristic polynomials of the two different PAMs [9]. This works since swapping two columns and rows in the PAM corresponds to relabeling logical node ids (since we assume logical node ids increase in order from left to right, top to bottom along matrix columns and rows) and does not change its determinant. A similar line of reasoning can be applied when thinking about the characteristic polynomial since it is equivalent to $det(A - \lambda I)$. This is not always the case in graph theory (for example see [10]), but for our cases where all vertices (modules) have at least one neighbor, this check is valid. If a match is found (as determined by the characteristic polynomial), then the function proceeds to permute over all number of modules factorial permutations to find an exact match and determine the mapping.

III. Off-Board Computation

For configurations of more than 10 modules $(10! = 3,628,800)$, a more powerful, off-board isomorphism routine for mapping may be desired. *Nauty* [11] by McKay claims to handle on the order of 10^2 vertices within a reasonable amount of time. The implementation of Nauty with CKBot is planned (by the time of the RSS workshop) to be executed by wireless transmission (via Bluetooth) from the controller to a host PC where Nauty can carry out its routine and send back the permutation mapping (if a match is found). Converting the data from a PAM to a proper adjacency matrix can be done with Castano's directed graph method. This would correspond to an $7x7$ matrix (since there are 7 IR TX/RX ports on CKbot) for each entry in the PAM. The adjacency matrix would become a $(7xN)x(7xN)$ matrix. It remains to be seen how well Nauty performs under this situation.

IV. Conclusion

In this workshop we intend to present a centralized control scheme that incorporates local neighbor information as feedback parameters. The algorithm features a filter that check for any isomorphism between the given state and any known states and then proceeds to find the proper mapping. The next step for this work is to apply gait multiplicity for configurations. The algorithm in Section II restricts each configuration to one prescribed gait. For versatility, this is not ideal since different circumstances often require different means of locomotion or actuation. To add sensory feedback as another imaginary "element" in the PAM would allow for multiple gaits to be applied for a single configuration.

References

[1] G. S. Chirikjian, "Kinematics of a metamorphic robotic system," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 1996, pp. 1452–1457.

[2] Z. Butler, R. Fitch, D. Rus, and Y. Wang, "Distributed goal recognition algorithms for modular robots," in *Proc. of the 2002 IEEE Intl. Conf. on Robotics and Automation*, Washington, DC, May 2002, pp. 110–116.

[3] J. Walter, E. M. Tsai, and N. Amato, "Choosing good paths for fast distributed reconfiguration of hexagonal metamorphic robots," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2002.

[4] G. S. Hornby, H. Lipson, and J. B. Pollack, "Generative representations for the automated design of modular physical robots," *IEEE Transactions on Robotics and Automation*, vol. 19(4), pp. 703–719, August 2003.

[5] I. M. Chen and J. Burdick, "Enumerating the non-isomorphic assembly configurations of a modular robotic system," *Intl. Journal of Robotics Research*, vol. 17(7), pp. 702–719, 1996.

[6] A. Castano and P. Will, "Representing and discovering the configuration of conro robots," in *Proc. of the 2001 IEEE Intl. Conf. on Robotics and Automation*, Seoul, Korea, May 2001, pp. 3503–3509.

[7] D. Gomez-Ibanez, E. Stump, B. Grocholsky, V. Kumar, and C. J. Taylor, "The robotics bus: A local communications bus for robots," in *Proc. of the Society of Photo-Optical Instrumentation Engineers*, 2004.

[8] S. H. M. Yim and K. Roufas, "Climbing with snake-like robots," in *Proc. of the IFAC Workshopon Mobile Robot Technology*, Jejudo, Korea, May 2001.

[9] H. Wilf, Personal correspondence, May 2006.

[10] E. W. Weisstein, "Characteristic polynomial." http://mathworld.wolfram.com/CharacteristicPolynomial.html.

[11] B. D. McKay, "Practical graph isomorphism," *Congressus Numerantium 30*, pp. 45–87, 1981, available at http://cs.anu.edu.au/~bdm/nauty/PGI/.