International Research Journal in Global Engineering and Sciences. (IRJGES)
ISSN: 2456-172X | Vol. 3, No. 1, March - May, 2018 | Pages 201-205

MODULARITY FOR GEOMETRIC RECONFIGURATION OF ROBOT SYSTEMS

D. Pinky [1], S. Karthikeyan [2]

Department of Mechatronics Engineering, Agni College of Technology, Chennai.

ABSTRACT

This chapter is organized according to a number of broad classes of problem where modular robotic systems can prove beneficial. For each such problem, the benefits of modularity are described, along with the ways that particular systems or proposed systems have explored those benefits. The systems under consideration in general have some level of independent computation on each module, and this discussion will focus on systems in which modules maintain some sort of kinematic constraint between them during operation. Compared to the types of multi robot teams, the systems of interest here are generally much more tightly coupled, both physically and conceptually. That is, we are primarily concerned with systems which, though they have many processors and independent actuators, have a single goal or small set of goals which can only be achieved collectively, rather than a set of goals which can be apportioned to single (or a small number of) robots within the team.

MODULARITY FOR GEOMETRIC RECONFIGURATION OF ROBOT SYSTEMS

The use of modularity allows for systems of many independent components to be reconfigured quickly between task operations with a minimum of effort. This is useful in manufacturing systems, where the modules may be capable of stand-alone operation, as well as self - reconfiguring robots, where sensors can be repositioned as the robot performs its tasks. In both cases, there are several issues to consider: ease of reconfiguration, determination of the current configuration, choosing good configurations for a particular task, and the design of the underlying task-performance algorithms for use with arbitrary configurations.

MANUALLY RECONFIGURED SYSTEMS

Traditional manufacturing can be performed in work cells, in which usually a single robot has access to several stations and performs a series of assembly operations. These work cells are designed to be fairly modular from a component point of view, swapping different portions of the product in and out as the manufacturing task changes. Several work cells can then be laid out along a conveyor or other line mechanism to create an assembly line. While they may be very similar in construction, they tend to be fairly independent in operation. Scheduling in the traditional sense may be important, but real-time interactions are less likely to be. In the Mini factory project, on the other hand, the entire system including the conveyances is part of a single modular system. All modules conform to a common interface, both in hardware and software. This allows rapid reconfiguration, with each robot able to be placed in the system with clamps and simple connectors. The robots share a common high-level language and a centralized simulation environment, so that tasks can be programmed into the system ahead of time.

When the modules are assembled, the robots that ferry the products through the system can also be used to discover the exact locations of all of the components as placed. The task algorithms, written under a common framework, are then automatically updated with the exact locations to be visited.

SHAPE GENERATION VIA SELF-RECONFIGURING SYSTEMS

Self-reconfigurable robots, as the name implies, can autonomously change their overall shape to fit the task at hand. In addition to using this capability to perform a variety of locomotion tasks, lattice-based systems in particular have the capability to make nearly arbitrary three-dimensional shapes. This may be useful for generating different sizes and shapes of payload carriers, manipulators, or other devices. In more recent research, systems of heterogeneous modules, where the capabilities of each module are not necessarily the same, have again received more attention. These may be useful for positioning cameras or other specialized sensors at varying positions during the completion of a task even as the shape of the system remains the same.

To perform self-reconfiguration, the modules in a system need to cooperate to plan their motions so that they can form the desired configuration. In common practice, the modules run without a central controller, but the global desired shape in some specification is given to all modules, at which point each module makes local decisions about if and where to move. To perform the reconfiguration, primarily local algorithms as well as more global planners have been developed. Local shape-formation techniques tend to be connection based, that is, the desired shape is specified by listing the connections to be made between modules. Each module is then responsible for ensuring it has the appropriate set of modules connected to it. Tomita et al. first proposed a shape generation scheme of this nature for two-dimensional systems, in which a single seed module collected modules around in successive layers, with each layer collecting appropriate modules as specified by the global design. Along a similar vein, work by Lipson et al. constructs shapes out of modules that do not contain traditional actuation.

The modules can only control when and where to allow other modules to connect to them to build the desired shape, and so a connection-based scheme is natural. Recent work by Klavins avoids the need for a seed module, and uses reaction network theory to allow modules that move stochastically to intelligently detach and reattach and achieve many copies of the desired shape in an efficient and reliable way. Global techniques tend to look more like traditional path planners – the modules discover local mismatches between the current configuration and the desired configuration, and paths are planned in a distributed fashion for modules to move into empty locations. One interesting feature is that the available paths are over the surface of other modules, so there is computation available wherever the search needs to take place, and distributed planning is quite natural. In addition, the paths are naturally discrete, so no artificial constraints need to be applied to perform the planning. Here there are two main concerns: first, a matching problem must be solved, to ensure that each goal location is the destination of only one module and that each location is accounted for; and second, deadlock must be avoided and traversability maintained if and when multiple plans are being generated and executed within the system. The first can be solved by having each goal location spoken for by a particular module, and having that module initiate a path search for a module that can fill the location. In addition, these techniques also tend to build the goal shape one layer at a time, starting from the current shape and extending outward.

Each individual path search must be filled by only one module, either by executing a depth-first search that terminates when a capable module is found or having some other way to choose a module if multiple modules attempt to fill the requested location. Modules also must know to attempt motion only if they are not required at their current position and can move without disconnecting the overall structure. MeltGrow and its heterogeneous successor, MeltSortGrow, handle this latter problem by moving all the modules to an intermediate shape (a line) from which the end of the line can be locally detected and only this module will move to a goal location.

However, modules can detect whether they can will disconnect the structure through a depth-first search from their neighbors, and so this melting step is unnecessary. During motion, care must be taken to ensure that paths are still valid as other modules begin to move. This can be done through serialization or through a localized traffic control scheme, in which modules communicate to take turns passing through a particular area. Shape reconfiguration algorithms can be shown to require O(n2) module motions for n modules, but usually only O(n) time if the motions are executed in parallel. If the systems are heterogeneous, that is, the modules are of different types, modules must also determine which type is required at a given location, and onlymodules of that type can respond to the particular request. Fitch et al. propose a planning system for identically sized but arbitrarily typed modules in which the correct shape is first achieved regardless of type, after which modules swap location. This algorithm still only requires O(n2) motion under most circumstances and can be efficiently executed in a distributed way.

CONFIGURATION OPTIMIZATION

The determination of an appropriate configuration is another key issue in both manually reconfigured and self-reconfiguring systems, but has been little discussed in the literature. One common approach is an empirical one – to simulate a particular task or set of tasks (e.g., an assembly process) under different configurations and compare efficiency in each case. For example, in describing the RMMS system, mentions that a simulator is available for determining the best configuration for a particular manipulation task. Similar work was performed for the CEBOT system, in which the location of the manipulator as well as the required number of modules and their configuration was optimized. There is work with binary manipulators to find an optimal configuration for a given set of points to be reached, in which Lagrange multipliers are used to minimize the change from a base structure to a structure that can achieve the desired locations. In systems such as self-reconfiguring robots, where hundreds of modules may be placed in fairly arbitrary ways, there are vast numbers of configurations to be considered, and the tasks themselves can be more complex. An evolutionary approach to optimizing configurations of modular (though not self-reconfiguring) robots. More general configuration optimization approaches from multirobot teams may be applicable for tasks such as surveillance, though they would need to be modified to handle the added capabilities of self-reconfiguring systems.

SELF-REPLICATING SYSTEMS

One type of system that obviates the question of configuration determination is the self-replicating modular robot. In these systems, one group of modules builds a copy from a supply of modules. While the theory of self-replication was established by von Neumann, engineering such a robotic system is a significant challenge.

This capability has been demonstrated in restricted form by Suthakorn et al., in which the robots are constructed of four fairly complex modules, and one robot can construct a second, and by Zykov et al. in a system of more traditional lattice-based self-reconfiguring modules.

MODULARITY FOR ROBUSTNESS

Finally, the use of a large number of simple modules to complete a task allows for robustness in the event of failure. If the modules are completely homogeneous, then they can simply be swapped out and replaced. If the system is engineered with sufficient redundancy, modules may even be done without as they fail. This process of self-repair may be possible to execute in an autonomous fashion, as has been proposed for some self-reconfigurable systems. In these cases, the failed module is ejected from the system either by selective disassembly to get the failed module to the surface, or by a pushing gait that moves the module. The DRAGON connector can always be detached from either side in the event of module failure, so that the system can continue with minimal disruption. However, despite the presumed importance of this self-repair capability of modular systems, this has not been a major focus of research. On the other hand, systems which are manually reconfigured may not have the option to evict a nonfunctional module. Instead, they may be built with inherent redundancy so that nonfunctional modules can simply be handled, either actively or passively. For example, in the virtual vehicle, the failure of a single unit may not affect the overall performance if the objects in transport are large relative to the actuators. Likewise, redundant manipulators such as the RMMS can continue to function if a single joint seizes, though this must be actively detected so that their changed kinematics and dynamics can be taken into account.

CONCLUSION

The benefits of modularity can be seen in many different task domains, whether the systems change their configuration on their own or require assistance from humans. The modularity can be used to simply create larger machines, for example, for more capable locomotion, or more complex machines that still retain a notion of simplicity at the module level. At present, hardware systems have largely remained in the research realm, while algorithmic progress has been notably more advanced in several areas, such as self-reconfiguration. It is hoped that the costs associated with modular robot development can be reduced as a byproduct of mass production, enabling more deployments that take advantage of the robustness and versatility inherent to such systems.

REFERENCES

[1] M. Yim, D. Duff, K. Roufas: PolyBot: A modular reconfigurable robot, Proc. of IEEE ICRA (2000) pp. 514–520
[2] Z. Butler, D. Rus: Distributed motion planning for modular robots with unit-compressible modules, Int. J. Robot. Res. **22**(9), 699–716 (2003)
[3] S. Murata, E. Yoshida, K. Tomita, H. Kurokawa, A. Kamimura, S. Kokaji: Hardware design of modular robotic system, Proc. of IROS (2000) pp. 2210–2217
[4] A. Castano, W.-M. Shen, P. Will: CONRO: Towards deployable robots with inter-robots metamorphic capabilities, Auton. Robot. **8**(3), 309–324 (2000)
[5] Y. Zhang, M. Yim, C. Eldershaw, D. Duff, K. Roufas: Scalable and reconfigurable configurations and locomotion gaits for chain-type modular reconfigurable robots,

Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation (2003) pp. 893–899

[6] Z. Butler, K. Kotay, D. Rus, K. Tomita: Generic decentralized locomotion control for lattice-based self-reconfigurable robots, Int. J. Robot. Res. **23**(9), 919–938 (2004)

[7] M. Ohira, R. Chatterjee, T. Kamegawa, F. Matsuno: Development of three-legged modular robots and demonstration of collaborative task execution, Proc. of IEEE Int'l. Conf. on Robotics and Automation (2007) pp. 3895–3900

[8] V. Trianni, S. Nolfi, M. Dorigo: Cooperative hole avoidance in a swarm-bot, Robot. Auton. Syst. **54**(2), 97–103 (2006)

[9] S. Murata, H. Kurokawa, S. Kokaji: Self-assembling machine, Proc. of IEEE ICRA (1994) pp. 442–448

[10] D. Rus, M. Vona: A physical implementation of the crystalline robot, Proc. of IEEE ICRA (2000)

[11] C. Ünsal, P. Khosla: Mechatronic design of a modular self-reconfiguring robotic system, Proc. of IEEE ICRA (2000) pp. 1742–1747

[12] K. Kotay, D. Rus, M. Vona, C. McGray: The selfreconfiguring robotic molecule: design and control algorithms, Proc. of the Workshop on Algorithmic Foundations of Robotics (1998)

[13] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, S. Kokaji: M-TRAN: Self-reconfigurable modular robotic system, IEEE/ASME Trans. Mechatron. **7**(4), 431–441 (2002)

[14] W.-M. Shen, M. Krivokon, H. Chiu, J. Everist, M. Rubenstein, J. Venkatesh: Multimode locomotion for reconfigurable robots, Auton. Robot. **20**(2), 165–177 (2006)