Hormone-inspired Adaptive Distributed Synchronization of Reconfigurable Robots

Feili Hou, Wei-Min Shen
Information Sciences Institute, University of Southern California

Abstract In this paper, we present a hormone-inspired adaptive distributed synchronization of reconfiguration robots. The main approach is to use a combination of discrete event-driven hormone communication and continuous time-controlled motor motion. Without any prior knowledge, all the modules' speed can be self-adjusted to adapt to each other to achieve a collaborative motion. Simulation and experimental results show that our method is robust to accuracy of local timer, configuration change and unexpected disturbance.

Keywords Adaptive distributed synchronization, reconfigurable robot

1. Introduction

Made from a network of homogeneous reconfigurable modules, the self reconfigurable robots can change their shape and size by themselves to accomplish different tasks in dynamic, uncertain, and even unforeseen environments. Due to this versatility, the self-reconfigurable robots have a number of potential applications, such as fire fighting, battlefield reconnaissance, undersea mining.

One of the most fundamental and vital issues to realize these potentials is the synchronization. To make a large number of connected modules accomplish a robust, flexible and adaptive global behavior, the action of the different modules must be coordinated and synchronized for maximum effectiveness. Since the centralized synchronization will cause poor scalability and high vulnerability, a distributed communication by means of local interaction is preferable. The research challenges for the distributed synchronization are:

1. Timer constraint: In the absence of a uniform global clock, all the modules need act at the right time autonomously to make the global behavior smooth and efficient.

2. Communication Constraint: In distributed synchronization, there is no global broadcast or central controller. All the modules should detect other's state and coordinate with each other just by local interactions with direct neighbors.

3. Motor speed regulation: To get a collaborative global behavior, all the modules should have autonomous speed control to adapt to each other in the absence of a centralized controller. For example, for the creep motion (please refer to Section 5 on this gait in detail), the two legs should be controlled to have the same speed so that the robot can move forward straightly.

4. Robustness requirement: If some modules can not reach the desired angles as expected, or some modules fail in the halfway, or the robot configuration is changed, how can the robot adapt to these situations and keep moving without jamming?

Focusing on these problems, we proposed a hormone-inspired adaptive distributed synchronization protocol in this paper. The idea of virtual disconnection is proposed to prevent the message circulating for cyclic robot configuration. After that, a

combination of discrete event-driven hormone communication and continuous time-controlled motor motion is developed to achieve a scalable and robust behavior.

In the following text, the related work is discussed in section 2. Section 3 proposes the idea of virtual disconnection, while section 4 describes the adaptive distributed synchronization protocol. Section 5 shows the capacity of our method by implementing it both in a physics-based GALINA simulation environment and in our SUPERBOT self-reconfigurable robot. Finally, conclusion and future work are made in section 6.

2. Related Work

In the past, many researchers have been devoted to develop the synchronization control of locomotion in reconfigurable robots.

Yim[1-2] use a gait control table to synchronize the modules' motion transition from one gait to another. The need of central controller is an obvious performance bottleneck that causes poor scalability and high vulnerability.

As an alternative, many distributed approaches have been studied. K.Stoy[3-4]gave a distributed synchronization based on the role-based control, where all the servos' motions are confined to sinusoidal. This restriction limits the robot to certain repetitive locomotion. For some specific one, like rolling track etc, it is hard to implement.

Shen[5] proposed to use artificial hormones to synchronize global locomotion among modules. In the early version, the hormone is generated every predefined period to start the next motion step. This brings a question of what is the exact period. If it is too short, each module can not reach the triggered position in time. If too long, time is wasted on waiting between any two motion steps. To get the time period, a well prior understanding of all the module's motor attributes is required. Moreover, this open loop control scheme is vulnerable to non-predictable changes.

In [6], a synchronization method is proposed that every module waits for the completion of its local actions before it relays the hormone. This avoids the timer issue, but engenders some limitation to its scalability. For example, a malfunctioned module will block the message transmission and bottleneck the whole behavior.

As an integration and improvement of the previous synchronization methods, the adaptive distributed synchronization in this paper makes it possible for the self-reconfigurable robot to achieve versatile and robust movements autonomously.

3. Virtual Disconnection

In our distributed synchronization protocol, a module selects appropriate actions based on received "command" hormones, its local state, and its location in the current configuration. To be robust to configuration changes, the hormone is directed to the module doing a specific function rather than to a specific module. Hereby, there is no name or identifier for each module. A module can automatically switch behavior if its location is changed in the configuration. In this section, we will give the definition of local topology vector to represent a module's location in the robot, and propose the idea of virtual disconnection to prevent message from circulating for the robot that has loops in its configuration.

Each module has a local topology to denote how its connectors are connected to the connectors of its neighbor modules. Suppose that every module has n different connectors, and each connector can either have no connection or connect to any of the n connectors of another module. Since each of the n connectors can have n+1 possible connection choices, the total number of local topology for each module is $(n+1)^n$. As n

increases, $(n+1)^n$ will increase exponentially. Therefore, instead of representing the local topology as a number like [5], we express it as a vector for more general application.

To illustrate the idea of local topology vector, let us consider a module that has six connection side as front(F), left(L), right(R), up(U), down(D), back(B). The local topology is represented by a vector that specifies the connection direction of the six connectors in the order of "F L R U D B". If some connectors have no connection, its connection is N. For example, if a module's topology type is [N,U,D,N,N,F], it means that the module has a topology as link[F]=N, link[L]=U, link[R]=D, link[U]=N, link[D]=N, link[B]=F. Here, link[x]=y (x=F,B,U,D,L,R; y=F,B,U,D,L,R,N) means that the modules' x connector is connected to the y connector of another module.

If the robot configuration is acyclic, it can be viewed as a tree. Each node denotes a module and the link corresponds to a physical connection between two modules. A "command" hormone can propagate through the entire tree of modules, yet causes different modules react differently. After receiving a "command" hormone, the module will check its connection from the topology, and send the hormone to all active links except the link from which the hormone is received. The generator module can be viewed as the root of the tree, when each module will receive the hormone from its parent, and will send the hormone to all children. Since the tree includes every node, the hormone reaches every module.

For the robot with loops in its configuration, to ensure that each "command" hormone is received once and only once, a scheme is needed to break the transmission loops. Since there isn't any identifier for each module, it presents a challenge that where the hormone transmission should be stopped. In [5], the control mechanism of individual module is changed to break the loop of communication. Involving the communication with the low level control of modules makes it not general enough to be applied into all robot configurations with loops. Inspired by the spanning tree algorithm, here we propose the idea of virtual disconnection to break the loops. The benefit of virtual disconnection is that it prevents the hormone from propagating to the same module again and again, without interfering the low-level module control.

First, starting from an arbitrary module as the root, we use breadth-first-search (BFS) to explore all the modules and get a spanning tree, and then virtually cut all the connections that are not in the tree edges. Here, to virtual cut, we just change the corresponding elements in the modules' local topology vector from uppercase to lowercase, and prohibit a module from sending the hormone message through the link when its topology value is lowercase. For example, given the robot configuration as Fig. 1, we start from module 1 and traverse all the modules by BFS. Three links that are not in the traversal path (shown by the crosses) are virtually cut, and the corresponding topology values in module 2 3 4 and 5 are changed accordingly. For example, the local topology of module 2 is changed from [N,N,N,R,F,F] to [N,N,N,R,F,f]. So module 2 will know that its B

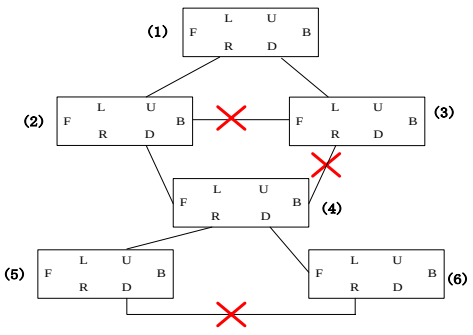

Fig. 1 Virtual disconnection for robot configuration with loops

connector is connected to another module's F connector, but it will not send any message out through it B connector. After the loops are broken, the robot can still be viewed as a tree, and the hormone message will not circulate any more.

4. Adaptive Distributed Synchronization Protocol

After virtual connection, the adaptive distributed synchronization protocol is used to collaborate the modules for an optimal, robust and scalable locomotion. The main approach is to use a combination of discrete event-driven hormone communication and continuous time-controlled motor motion.Here, each module works is deemed as an independent subsystem that works in a distributed way and runs the same program. Fig. 2 shows the main procedure of the program. Basically, it is a loop of receiving and sending hormones between neighbors, and executing local actions based on these messages. For each received hormone, the module will check its type, and then choose the time-controlled local actions or event-driven communication. The complex global behavior of the robot emerges from the interconnection and communication between individual modules. Here, a hormone is defined as [*type*, *data*, *src*], where *type* can be "command" or "feedback", *data* is the hormone data, and *src* is the receiving connector through which the message is received. The "command" hormone is used to trigger different actions on the modules, while "feedback" hormone is for coordinate all the modules action.

```
Loop do
    For each received hormone [type,data,src] do
            if (type==command)     then  Continuous time-controlled Local Actions.
            else          Event-driven hormone communication.
    ExecuteLocalAction()
    LocalTime++;
End Loop;
```

Fig. 2 The adaptive distributed synchronization protocol

4.1. Continuous Time-controlled Local Actions

```
function PropagateHormoneAndGetDesiredAngles (Hormone[command, data, src])
    NewHormone[type, newData, src]<- Generate Hormone data according the RULEBASE;
    Cp=src;
    Send out NewHormone to all active connectors except Cp.
    DesiredAngles <-Select Actions according to the RULEBASE;
    Determine the time DT for the last received hormone }
    Move servos from LastDesiredAngles to DesiredAngles according to equation (1)
```

Fig. 3 The continuous time-controlled local actions

Fig. 3 shows the pseudo code program of continuous time-controlled local actions. Originated from the head module, the "command" hormone is modified during its transmission, propagates through the entire tree, and stops at the leaf nodes. Different module reacts to the "command" hormone differently based on its own topology and local state information etc under a predefined RULEBASE. After a module receiving a "command" hormone[*command*, *data*, *src*], it will save *src* to be *Cp* (the connector that links to the parent module), rely the hormone to all of active connectors except *Cp*, set the *DesiredAngles* of its servo, and move the servo accordingly. Different locomotion has different RULEBASE, please refer to the hormone-inspired control [5] on how to construct it for detail.

As well as determine the *DesiredAngles*, the servo movement should have autonomous speed control so that the module can collaborate with each other. Here, we make the modules predict the time interval of each motion step by the history value, and self-regulate their own speed accordingly. For each received hormone [command, data, src], DT[data] represents the elapsed time before the module receives the next "command" hormone, and is used as the estimated time interval after the module receives the same hormone again. Suppose the module's servos have reached *LastDesiredAngles* at previous step, and is triggered to *DesiredAngles* now by the hormone [command, data, src]. The module will regulate the motor speed to move from *LastDesiredAngles* to *DesiredAngles* within time *DT*[*data*] under equation (1).

$$\text{ServoAngle[i](t)} = \begin{cases} \frac{\text{L[i]} - \text{D[i])}}{2} \sin\left(\frac{\pi}{\text{DT[data]}} t + \frac{\pi}{2}\right) + \frac{\text{L[i]} + \text{D[i]}}{2} & t < \text{DT[data]} \\ \text{D[i]} & t < \text{DT[data]} \end{cases} \quad (1)$$

where t is the module's local time, L[i] is the *LastDesiredAngles* of the i^{th} servo, and D[i] is the *DesiredAngles* of the i^{th} servo. Instead of moving in a steady speed straight line that may suffer sudden speed changes between any two steps, we make the servos move in a sinusoid pattern. The speed at the angle of *LastDesiredAngles* and *DesiredAngles* are all zero, so that the robot moves smoothly without speed discontinuity between any two motion steps. If the servos' speed can not achieve the action within DT[data], it will move directly towards *DesiredAngles* after that.

4.2. Event-driven Hormone Communication

Event-driven hormone communication is used to coordinate all the modules, including synchronizing local clock, giving feedback on action state etc. Here, the event is the state that a module and all its descendents in the tree have completed their actions in the current step. Once reaching this state, the non-head module will generate a "feedback" hormone to send its action state to its parent from Cp, while the head module will generate a new "command" hormone to start another motion step. Hereby, "feedback" hormone traces back from the leaves to the root module when all the actions are finished in that step. Fig.4 shows the routine of event-triggered hormone communication. When a module receives a "feedback" hormone through *src*, it is informed that all the modules in the sub-tree rooted by src have finished their actions, and thus set the *flag[src]* to be 1.If the module has reached the *DesiredAngles*, the *flag*[*self*] is set 1. When flag[self] and *flag[c]* both reaches 1 for all active connectors c except *Cp*, a new "command" hormone is engendered for the head module while a "feedback" hormone will be sent to its parent module for a non-head module.

```
function FeedbackHormone (Hormone[feedback, data, src])
    flag[src]=1;
    if (|DesiredAngles-CurrentAngles|< δ ) then flag[self]=1;
    if (flag[self]==1 & flag[c]==1 for all active connectors c except Cp) | (LocalTime>MaxClock) then
        clear all flag values to be 0
        if (Cp!=NIL) then Send out [feedback, 1, nil] to its parent module through Cp connector
        else Generate a new "command" hormone in sequence, and send it out to all connectors links
```

Fig.4 The Event-triggered hormone communication

To increase the robustness, we also set up a wake up mechanism. Due to the potential of communication errors, there may be situations where "feedback" hormone is lost. Or, the *DesiredAngles* is unreachable for the servos. To prevent the robot from

waiting forever in these situations, we set a threshold MaxClock. If the local timer reaches MaxClock, the module will wake up, and a new “command” or “feedback” hormone will be generated.

4.3. Discussion

The adaptive distributed synchronization protocol has the following properties that are essential for reconfigurable robots.

It is independent of the accuracy of the local timer rate of each module. Even though the clock rates are different among all the modules, the time interval in each motion step is well synchronized based on the feedback mechanism and DT prediction. DT is self-adjusted. If it is too short for a module, namely that some module can not complete the triggered action within that interval, the “feedback” message will be delayed by that module. So the next “command” hormone will be generated later than DT, and thus DT will be updated longer. In each motion step, DT is updated by the longest action time among all the modules in the robot, gradually increased to be long enough for all the modules to complete their actions, and becomes stable in the end. Based on the DT value, all the modules can regulate their speeds automatically to be consistent with each other without any prior knowledge about other’s motion attribute.

It can achieve collaborative behaviors just by local communication between neighbor modules. Once actions in that motion step are completed for all modules in the robot tree, the head module will be notified by the “feedback” hormone and generate a new “command” hormone to travel further right away. Therefore, the robot can move continuously without gap between any two steps.

It is resistant to disturbance. Due to our feedback mechanism, it is robust to the interference. If the speed of some modules is slowed unexpectedly, the delayed “feedback” hormone will postpone the generation of the next “forward” hormone, and thus increase the DT value for all modules. Hereby, all other modules will slow down their speed to achieve the collaborative behavior. Or, if some module can not reach the desired angles as expected, the wakeup mechanism will prevent it from stuck forever.

It is robust to shape changing as modules can be added deleted or rearranged. Not any specific module, but the one with specific topology, acts as the head module. So the robot tree is dynamic and any module can become a leader when its local topology is appropriate. All the modules work in a distributed way so that there is no single point of failure. For example, for the caterpillar robot, the leftmost module with topology [N,N,N,N,N,B] acts as head module, and generates “command” hormones to engender the movement. If the head module is tore apart, the new left-most module will quickly update its topology, become the head based on its new topology, and generate another sequence of “command” hormones to keep the remaining modules moving.

It is highly scalable. The robot is reduced to an interconnection of subsystems. Since all the modules work independently under the same rule, it is independent of the robot size. Moreover, it works for all robot configuration, because all configuration can be deemed as a graph, and evolves to be a tree under our virtual disconnection method.

5. Experimental results

The adaptive and robust advantage of our synchronization method have been implemented the both in a physics-based GALINA simulation environment and our self-reconfigurable robot called SUPERBOT.

5.1. SUPERBOT Module

Fig. 5 shows our design of SUPERBOT module. It has enough moving flexibility to combine M-TRAN[8], ATRON[9] and CONRO[5] into one. As shown in Fig.5, each module essentially has three parts, the two segments and central part. Each of the two segments has three universal connectors on three different sides so that a module can connect to any other module in any of the six directions in 3D (up, down, front, back, left, and right). Joined by a gimbal mechanism, the module has 3 rotational degrees of freedom and two pitch/yaw connected by a roll. This design will allow the module to pitch and yaw for up to 180 ^{o} and roll for 90^{o} in each direction. This design gives the SUPEROBT module the most flexible movements, and provides the needed flexibility for the different locomotion and reconfiguration of multi-modules. Every module is able to detect if a connector is docked or not, and talk with up to six neighboring modules with a high-speed and reliable communication bus.

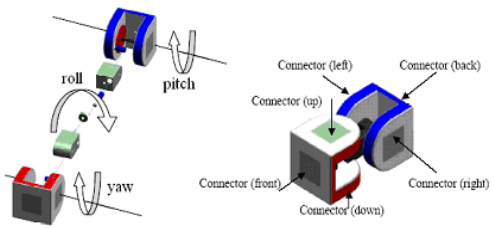

Fig. 5 Schema of SUPERBOT module

5.2. Simulation Result

We have developed a simulation software called GALINA to create superbot modules and the testing environments as realistic as possible. To emulate concurrent execution of control programs for different modules and resulting communication issues, we use the threads mechanism to emulate simultaneously running modules in GALINA. In the implementation of all these behaviors, all the modules are loaded with the same program.

The fist example is a creep motion. As shown in Fig. 6, the creep robot is composed of two modules that make the butterfly action. Intuitively, the two legs should be lifted from the ground when moving forward and touching the ground when moving backwards.

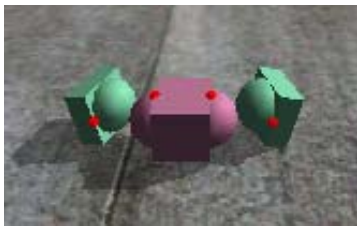
Fig. 6 The robot with creep motion

To show the independence of the internal clock and ability to self regulating the speed, we deliberately set the clock frequency of the two modules as 1000HZ and 1100KHZ, and the maximum motor speed of the left and right module to be 180^{o}/s and 220^{o}/s respectively. From creep.avi (http://www.isi.edu/robots/superbot/movies/sim/creep.avi), we can see that the two modules' timer and speed are well synchronized, and the robot moves in a straight way. Moreover, even though the motion of lifting leg has different amplitude from moving forward, the speeds are well regulated so that the legs do not touch the ground before reaching the destination when swing forward.

To demonstrate the adaptive ability to unexpected disturbance, we deliberately decrease the maximum speed of the left module in the halfway to make it unable to complete the desired motion in time. As we can see from creepRecover.avi (http://www.isi.edu/robots/superbot/movies/sim/creepRecover.avi), the robot makes a turn since the left leg is suddenly slowed down. After that, the feedback mechanism automatically increase the DT for each hormone, and the right module quickly adapts to the new speed of the left module. In a short while, the robot moves straightly again.

Multi-locomotion, including caterpillar, sidewinder, rolling track, and crawler gait etc, is also implemented. To test the ability of recovering from dynamic configuration change, we deliberately "cut" a 8-module caterpillar, a sidewinder, and an eight-legged crawler into halves in the halfway. Without any modification to the program, all these segments adapt to the new configuration immediately and move as two small caterpillars, sidewinders, or crawlers.

5.3. Experimental Result

We have also loaded our program into the real SUPERBOT modules for different locomotion. As of the writing of this paper, only two SUPERBOT modules are built, so the possible locomotion is very limited. After connecting the two modules in a line, we can make it have different motion, like caterpillar, creep etc. By changing the head module, the robot can move forward or backward. Both of the two modules are loaded with the same program and well synchronized.

For the video of different behaviors in simulation and in real robot, please visit http://www.isi.edu/robots/superbot/movies/

6. Conclusion

This paper presents a hormone-inspired adaptive distributed synchronization for reconfigurable robots. The communication protocol for both cyclic and acyclic configuration is improved with the idea of virtual disconnection. The "feedback" hormone is proposed to make the behavior robust to dynamic changes and unexpected disturbance. Collaborative locomotion is achieved by adjusting each module's motor speeds autonomously to make the modules adapt to each other. Effectiveness of our algorithm is demonstrated by the SUPERBOT modules both in physics-based simulation and in real robots. Multimode locomotion modes are implemented without increasing the hardware or software complexity. Our future work will be adding sensor information so that the robot can react to the environment.

References

[1] M. H. Yim, Y. Zhang, D. G. Duff, "Modular robots," IEEE Spectrum, 39(2), pp. 30-34, February 2002.

[2] M. Yim, Locomotion with a unit-modular reconfigurable robot (Ph.D. Thesis), in Department of Mechanical Engineering. 1994, Stanford University.

[3] Stoy, K, W.-M. Shen, P. Will, "Using Role-Based Control to Produce Locomotion in Chain-type Self-Reconfigurable Robots," IEEE Transactions on Mechatronics, 7(4), 410-417, Dec. 2002.

[4] K. Støy, W.-M. Shen, and P. Will, "How to Make a Self-Reconfigurable Robot Run", Proc. of the 1st international joint conference on autonomous agents and multiagent systems, July, 2002.

[5] W.-M. Shen, B. Salemi, and P. Will, "Hormone-Inspired Adaptive Communication and Distributed Control for CONRO Self-Reconfigurable Robots," IEEE Transactions on Robotics and Automation, 18(5), October 2002.

[6] W.-M. Shen, B. Salemi and P. Will, "Hormone for self-reconfigurable robots". Proc. Of Intl. Conf. Intelligent Autonomous Systems, IOS Press, pp. 918-925, 2000.

[7] W.-M. Shen, Y. Lu and P. Will, "Hormone-based control for self-reconfigurable robots." Proc. Intl. Conf. Autonomous Agents, 2000.

[8] S. Murata, E. Yoshida, etc, "Hardware Design of Modular Robotic System", Proc. of 2000 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, 2000

[9] .MWJorgensen, EHOstergaard, HHLund, "Modular ATRON: Modules for a self-reconfigurable robot", proceedings of IEEE/RSJ International Conference on Robots and Systems, pp. 2068-2073, 2004.