Cooperative Task Execution between Modular Robots Based on Tight-Loose Cooperation Strategies

José Baca and Claudio Rossi and Manuel Ferre and Rafael Aracil
Centre for Automation and Robotics
Universidad Politécnica de Madrid
j.baca@etsii.upm.es, claudio.rossi@upm.es, m.ferre@upm.es

Abstract—The complexity in the execution of cooperative tasks is high due to the fact that a robot team requires movement coordination at the beginning of the mission and continuous coordination during the execution of the task. A variety of techniques have been proposed to give a solution to this problem assuming standard mobile robots. This work focuses on presenting the execution of a cooperative task by a modular robot team. The complexity of the task execution increases due to the fact that each robot is composed of modules which have to be coordinated in a proper way to successfully work. A combined tight and loose cooperation strategy is presented and a bar-pushing example is used as a cooperative task to show the performance of this type of system.

I. INTRODUCTION

Nowadays, multirobot systems are an active field of research. A variety of techniques have been proposed in order to approach the problem of motion coordination in different kinds of applications [12]. Cooperation applications can be roughly divided in two classes: tight cooperation requires a continuous coordination between the robots; loose cooperation requires coordination at the beginning of the mission for planning a division of a task and at given moments of time, when re-planning may be needed. Behaviour-based [11] and schemas [1] are examples of techniques suitable for the first class of coordination problems, while market-based [5] and auction [6] techniques are commonly used in the second class of problems. The multirobot research community has focused primarily on implementing cooperative control strategies in highly efficient robots that were designed for a specific task such as wheel-based robots. However, cooperation applications with modular robot systems have been less explored. Over the last decade, research in this field has been demonstrating the potential advantages that this type of system offers such as versatility, fault-tolerance and cost-effective platforms. However, there are few studies about building multirobot teams made of modular robot configurations and evaluating the system during the execution of cooperative tasks [7] [15].

Modular robot systems are capable of forming different robot configurations made up of n-modules, which have to work in a coordinated fashion to show uniform behavior. Their ability to rearrange their modules and to adapt to different circumstances, allows them to cope with multiple tasks such as different types of locomotion and manipulation, a feature ensuring that their performance level excels in a variety of circumstances. When a robot is built by the union of several robots, such as modular robot systems, it is critical to have a tight cooperation to complete coordination of each robot configuration [17] [9], but also a loose cooperation to coordinate the colony during cooperative tasks [8] [10].

This work focuses on presenting the execution of a cooperative task by a multirobot team made of modular robot configurations. A bar-pushing example is used as a cooperative task to demonstrate that modular robots can successfully perform these types of tasks as standard robots can. A tight cooperation strategy is implemented to coordinate modules of each robot configuration and a loose cooperation strategy is used for task partition and assignment (Fig. 1). This paper is structured as follows: Section II briefly describes the modular robot system used in the experiments, named SMART. Section III points out a tight cooperation strategy implemented to coordinate modules of each robot configuration. Section IV describes a loose cooperation strategy used to divide a mission and assign to each robot its corresponding sub-task. The implementation of both strategies is detailed in section V. Section VI describes experimental results performed by modular robot configurations during an individual and cooperative task. Finally, concluding remarks are presented in section VII.

II. SMART DESCRIPTION

The SMART system [2] is a reconfigurable heterogeneous modular robot system composed of a set of interchangeable modules that form different robot or system configurations. Each module type aims to balance versatility, low-cost fabri-
A. Synchronizing M-Robot modules

the H4M-Robot modules. From the computational point of view, modular robots are movements at the same instant of time. With this type of modules have to work in perfect synchrony to behave as a communication has to be carried out by Bluetooth technology. P/C physical contact exist between bus technology. inter-robot communication is used when no modules. Since el-

A cooperative control strategy can be termed tight when

A closed-loop discrete time method [3] is implemented and consists of keeping all system clocks in the same phase and period, using a single short message in every cycle. The period of that cycle can be relatively longer (seconds) than the control cycle period (milliseconds, typically the period of timer interruption). The method needs a periodic signal acting as a trigger for the closed loop. This signal is generated by the master module for every N control cycles and consists of a high priority short CAN message. In theory, all modules receive the message at the same time, but this is not correct. Each time the message is received, a local timer (tick counter) is reset and its previous values are used to correct the local timer period. When a synchronizing message enters the process, the current counter value is used to recalculate the local timer period. Consequently the counter is reset.

B. Synchronizing a colony of M-Robots

The main idea of synchronizing a colony of modular robots is the creation of a colony behavior for execution of tasks such as simultaneously pushing or lifting an object with two or more robots. Assuming that each M-Robot achieves its correct robot behavior as explained in III-A, the next step is to synchronize master modules of each M-Robot of the colony. In order to achieve the desired synchronization, a control station (PC) is used as the main task coordinator. While the master module synchronizes the slave modules, it periodically sends a signal (clk a) to the control station via BT indicating its timer value. In order to synchronize a second M-Robot from the colony, it is required to acquire a second signal from the second master module. The second M-Robot periodically sends a signal (clk b) to the computer. At this time, the control station has two signals originating from each M-Robot. The control station calculates the offset between clk b with respect to clk a, represented by equation Offset = clk a(t) - clk b(t). Once the difference is calculated, the control station sends the correction to the corresponding master module. The master module receiving the modification adjusts its internal timer to the reference robot’s internal timer. Thus, synchronization is achieved for both robots. The motion control loops are executed at the same time, thereby enabling the execution of simultaneous movements in the robots as shown in Fig. 3. The timing calculations do account for communication delays which are expected with BT message transmission. The operator can set the period of transmission of the sync signal for the robots as well as the minimum allowable gap in synchronization signals [3].

IV. LOOSE COOPERATION STRATEGY

In loose cooperation problems, a given task has to be partitioned in sub-tasks, and sub-tasks have to be assigned to individual team-members for execution. The main difference between tight cooperation problems and loose cooperation problems is that in a loose cooperation, the action of a robot has an immediate impact on the global task to be executed which affects the team-mates’ goal; when a robot pushes the

A closed-loop discrete time method [3] is implemented and consists of keeping all system clocks in the same phase and period, using a single short message in every cycle. The period of that cycle can be relatively longer (seconds) than the control cycle period (milliseconds, typically the period of timer interruption). The method needs a periodic signal acting as a trigger for the closed loop. This signal is generated by the master module for every N control cycles and consists of a high priority short CAN message. In theory, all modules receive the message at the same time, but this is not correct. Each time the message is received, a local timer (tick counter) is reset and its previous values are used to correct the local timer period. When a synchronizing message enters the process, the current counter value is used to recalculate the local timer period. Consequently the counter is reset.

B. Synchronizing a colony of M-Robots

The main idea of synchronizing a colony of modular robots is the creation of a colony behavior for execution of tasks such as simultaneously pushing or lifting an object with two or more robots. Assuming that each M-Robot achieves its correct robot behavior as explained in III-A, the next step is to synchronize master modules of each M-Robot of the colony. In order to achieve the desired synchronization, a control station (PC) is used as the main task coordinator. While the master module synchronizes the slave modules, it periodically sends a signal (clk a) to the control station via BT indicating its timer value. In order to synchronize a second M-Robot from the colony, it is required to acquire a second signal from the second master module. The second M-Robot periodically sends a signal (clk b) to the computer. At this time, the control station has two signals originating from each M-Robot. The control station calculates the offset between clk b with respect to clk a, represented by equation Offset = clk a(t) - clk b(t). Once the difference is calculated, the control station sends the correction to the corresponding master module. The master module receiving the modification adjusts its internal timer to the reference robot’s internal timer. Thus, synchronization is achieved for both robots. The motion control loops are executed at the same time, thereby enabling the execution of simultaneous movements in the robots as shown in Fig. 3. The timing calculations do account for communication delays which are expected with BT message transmission. The operator can set the period of transmission of the sync signal for the robots as well as the minimum allowable gap in synchronization signals [3].

IV. LOOSE COOPERATION STRATEGY

In loose cooperation problems, a given task has to be partitioned in sub-tasks, and sub-tasks have to be assigned to individual team-members for execution. The main difference between tight cooperation problems and loose cooperation problems is that in a loose cooperation, the action of a robot has an immediate impact on the global task to be executed which affects the team-mates’ goal; when a robot pushes the
bar, the new bar position shall modify the way the other robot will push it. Our coordination algorithm is based on Rubinstein’s alternate-offers protocol [14] and performs a simultaneous task subdivision and allocation in a distributed way, taking into account robot characteristics at all times.

A. Task subdivision

The task objective is to push a bar to a desired location, or equivalently, to push the bar in a desired direction. This is an instance of a more general task partitioning problem that we have generalized as follows [13]. Given a global task $T_0$ and $r$ robots, the problem is how to partition $T_0$ into $r$ non-overlapping sub-tasks such as

$$T_i \cap T_j = \emptyset, \quad \forall i, j = 1 \ldots r \quad \text{and} \quad \bigcup_{i=1}^r T_i = T$$

and how to assign sub-tasks to the robots for execution. Our technique performs the two steps simultaneously and in a distributed way: each robot is aware of its own characteristics but does not know anything about its team-mates' characteristics. In our approach, the original task $T_0$, in this case the desired direction, is divided into two sub-tasks through a negotiation process, where each robot claims a desired push location and force such that the combination of the push actions of the two robots results in the desired direction.

A task $T_i$ is described by a set of $k$ parameters $P_i$. In this case, $k = 2$, and $P_i = \{L_i, F_i\}, i = 1, 2$. This means pushing the bar at location $L_i$ with force $F_i$ for $\Delta t$ time. Operators $\cap, \cup$ are to be defined according to the meaning of the task. In this case, considering the task parameters $P_i$ as vectors, and taking into account standard vector addition, the union of two tasks is the projection of the sum into the desired direction vector (“useful” component), and the intersection as the projection of the addition on the vector orthogonal to the desired direction (“wasted” component), as shown in figure 4 (a).

B. Task assignment

The task negotiation process is a bargaining loop where, in turns, each robot proposes its desired push location and force, and the other computes the best push location and force for itself, taking into account the combination of the two actions, the current bar orientation, and considering that a good combination is such that the resulting direction is the one defined in $T_0$.

We assume that robots are willing to perform as much as they can of the given task, the only limitations being their available resources (endurance, computation power, battery consumption etc.). Thus, in a negotiation, each agent will try to maximize its reward by (i) trying to obtain a possible sub-task as large as possible (the closer to $T_0$ the better) and (ii) minimizing overlap with other agents’ tasks (intersection, i.e. wasted effort). Each agent proposes the biggest possible share for itself, and reduces it until the counterpart finds it acceptable. Figure 4 (b) depicts the negotiation process.

---

\[\text{Fig. 3. Synchronizing a modular robot colony for parallel movement.}\]

\[\text{Fig. 4. (a) Union and intersection of bar pushing tasks. (b) Negotiation loop.}\]

At the end of each bargaining loop, the robots perform the agreed pushing action. Note that since the actions eventually cause a change in the orientation of the bar, the negotiation is performed again at each outer control loop, taking into account the actual bar position.
V. IMPLEMENTATION

The implementation consists of creating a control loop between the cooperative task and the modular robot colony. The main condition of this testbed is to have an autonomous team of modular robots performing a task after assigning the initial conditions. Task partition and allocation will be given by a loose cooperation strategy, and module coordination by a tight cooperation strategy.

A. Bar-pushing set-up

In a cooperative task, the main goal and the quantity of robots to use are settled in advance. In this case, a bar-pushing task has to be performed with two wheel-based robots. The scenario is composed of a bar and two W2M-Robots. Due to the fact that the robots do not contain any sensors, a video camera is placed on top of the scenario to capture the data required for the negotiation algorithm, as shown in Fig. 5.

![Fig. 5. Bar-pushing scenario.](image)

B. Visual tracking interface

In order to introduce the data into a loose cooperation strategy a visual tracking interface (VTI) based on video feedback was developed. At the beginning of the experiment, the bar and each M-Robot position and orientation is manually captured by the operator by clicking over each object on the VTI (Fig. 6). The desired bar orientation is also introduced into the algorithm and can be dynamically changed during the task execution.

![Fig. 6. Visual tracking interface.](image)

As mentioned in section IV, the strategy divides the global task and assigns subtasks to each W2M-Robot. With these values, and the following equations (in this case for robot\(_i\)), the interface displays on the screen the contact point over the longitudinal axis of the bar where each robot should apply the push force. (shown in Fig.7).

\[
\begin{align*}
O\hat{x} &= O\hat{A} + \hat{A}\hat{x} \\
\hat{A}x &= P_{r_i} \times (\hat{A}\hat{B}_{x}, \hat{A}\hat{B}_{y}) \\
\hat{A}\hat{B} &= \left(\frac{\hat{A}\hat{B}_{x} \times \hat{A}\hat{B}_{y}}{||\hat{A}\hat{B}||}\right) \\
||\hat{A}\hat{B}|| &= \sqrt{\hat{A}\hat{B}_{x}^2 + \hat{A}\hat{B}_{y}^2}
\end{align*}
\]

Where:

- \(O\hat{x}\) = Push force contact for robot\(_i\)
- \(P_{r_i}\) = Value generated by the negotiation algorithm.

![Fig. 7. The x and y represent the position where robots apply the force.](image)

C. Modular robot graphical user interface

The autonomy of the system is complete when the negotiation algorithm is linked with the VTI and the modular robot graphical user interface (MR-GUI). Normally, the MR-GUI is in charge of teleoperating and assigning behaviors to modular robot configurations. After linking the MR-GUI with the VTI, no operator is required for the cooperative task execution. The MR-GUI automatically controls and assigns the corresponding operating modes to the W2M-robots. The correct execution of the operating mode will depend on the tight cooperation strategy explained in section III.

![Fig. 8. Cooperative task execution is performed according the negotiation algorithm demands. Once the subtasks have been achieved, the new position and orientation parameters from the objects are uploaded to the negotiation algorithm and the cycle begins once again until task fulfillment.](image)
VI. EXPERIMENTAL RESULTS

This section describes the experiments that show the modular robots’ performance during the execution of a cooperative task. A bar-pushing task is used as an example to explore the cooperation between two wheel-based modular robot configurations.

A. Tight cooperation experiment

The first experiment attempts to demonstrate that the W2M-Robot configuration can satisfactorily perform a behavior that a standard robot of its kind can do. The robot configuration used in this experiment is composed by one P/C module, two J modules, two S modules, and two software modules (MsM and SsM). Each software module loaded on each side of the P/C module controls its corresponding J module, therefore a tight cooperation between the master and the slave modules has to be carried out. The assembled W2M-Robot configuration should behave as a differential drive mobile robot, that is, forward, backward and rotation movements have to be performed in a proper manner to satisfy the demonstration. Using the VTI described before, the robot position and orientation parameters are captured into the tracking system. Three movement types were executed by the W2M-Robot as shown in Fig.9.

B. Loose cooperation experiment

The intention of the second experiment is to demonstrate that a modular robot colony can satisfactorily perform a cooperative task, just as similarly specialized robots can. In other words, the modular robot colony will be controlled by a loose cooperation strategy which was designed for specialized robots. Two types of experiments were performed for this demonstration.

1) Pushing the bar to a desired orientation: The objective is to push a bar to a desired orientation. As it can be seen in Fig.11 the negotiation algorithm divides the mission in subtasks to each robot to complete the goal. Each robot configuration performs its task in an autonomous way until the bar orientation matches the desired orientation. It should be noted that the experiment did not require both robots to push the bar at the same time. The solution given by the loose cooperation strategy is that only W2M-Robot2 should apply the pushing-force for completing the mission. The tracking system displays bar and robots position path. It can be noticed that W2M-Robot1 keeps close to the bar but never pushes the bar. A total of 50 seconds were required for obtaining desired bar orientation.

2) Pushing the bar with desired orientation: This experiment emphasizes the next step after obtaining the desired bar orientation. The objective is to push the bar while maintaining the desired bar orientation, as shown in Fig. 12. The complexity of the task increases due to the fact that both robots should simultaneously move in a coordinated fashion to achieve the goal. It can be seen during the execution of the task a satisfactory performance of both W2M-Robots. During 25 seconds, an error of 4 degrees between the desired bar orientation and bar orientation is produced.
VII. CONCLUSIONS AND FUTURE WORK

The combination of both tight and loose cooperation strategies demonstrates that modular robot systems can successfully perform a cooperative task as standard robots can. An individual and cooperative experiments were executed to display each strategy’s performance. With a tight cooperation strategy, the M-Robot exhibits appropriate behavior with respect to its robot configuration. The modules achieve proper synchronization for the execution of coordinated movements. A bar-pushing experiment was used to demonstrate the execution of a cooperative task with a team of modular robot configurations. Our future work is to continue demonstrating that modular robots can be suitable systems for performing cooperative tasks. The implementation of new cooperative strategies and the execution of different cooperative tasks are the next goals.

VIII. ACKNOWLEDGMENTS

The authors would like to thank CICYT and CONACYT for financing this research project. Also special thanks to Erik Hernandez and Cecilia Garcia for their contribution during the execution of experiments.

REFERENCES


