LEGO MINDSTORMS CHEERLEADING ROBOTS

Naohiro Matsunami[1], Kumiko Tanaka-Ishii[2], Ian Frank[3], and Hitoshi Matsubara[3]

1 Chiba University, Japan

2 Tokyo University, Japan

3 Future University-Hakodate, Japan

nmatsuna@cogsci.l.chiba-u.ac.jp, kumiko@ipl.t.u-tokyo.ac.jp, ianf@fun.ac.jp, matsubar@fun.ac.jp

Abstract We present a novel multi-robot system capable of performing cheerleader dances. Our team of seven robots are constructed from very simple LEGO Mindstorms kits, yet still monitor their own synchronisation automatically in real time. We describe how we use a scripting language to abstract the overall control into a set of higher-order functions, and to produce fault tolerancy.

In addition to demonstrating the practical application of focusing on the fundamentals of multi-robot systems, we plan to use our cheerleaders to investigate the research issues that lie at the interface of robotics, external world events, human observers, and entertainment.

Keywords: Synchronisation, multi-robot systems, abstraction of control.

1. Introduction

Robots are now becoming more common in daily life [1, 6]. As the paths of humans and robots begin to cross each other more, we can expect that as well as one-on-one encounters, interactions between multiple robots and multiple people will become common. Further, just as Reeves and Nass have found [4] that humans react to computers in social ways, we can expect that robots carrying out "human-like" activities will also elicit social responses.

Our work addresses these aspects of robotics development by investigating how a team of robots can work together in an activity that is usually carried out by humans: cheerleading at sports games. We demonstrate the central nature of the abstraction of control for this task, and describe how a scripting language that describes the overall behaviour of a group of robots can be automatically compiled into

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35660-0_65

R. Nakatsu et al. (eds.), *Entertainment Computing*

programs for controlling the low-level actions of individuals. We also address the problem of error tolerance and describe our technique for maintaining coherency our among cheerleading team.

Robotics research to date has tended to focus on the task of constructing single moving robots. Research on collaboration between several robots is less common. However, in human activities, it is generally acknowledged that collaborative efforts can generate results that would not have been possible with the individuals working alone. We believe that the same "the whole is greater than the sum of the parts" message should apply to robots, and that therefore the study of collaboration in robots should be important.

One example of existing research in this area is the robotic leagues of the RoboCup soccer tournaments [5]. These leagues involve teams of up to five robots playing against each other in real time. Although the five years of RoboCup competitions have produced significant progress in the level of play, it is clear that the games are still a long way from beginning to approach the level of real soccer. We deliberately took a less ambitious goal for our project, choosing to concentrate instead on the lessons to be learned from implementing basic technologies such as coordinating a group of robots into a specific formation or advancing robots together in a line. Such fundamental motions performed by several independent robots have potential for application in many domains. Previous work on coordination among small numbers of robots has been carried out by Mataric [3]. Our work differs in that we produce group behaviour using LEGO Mindstorms kits [2] that are very basic, with very limited sensing abilities.

We demonstrate that even with limited hardware, augmenting the environment with extra information can make it possible to produce robust behaviour from a team of robots. We intend to use our cheerleader team at events such as the RoboCup competitions to investigate the nature of robotic entertainment.

2. System Overview

Figure 1 shows our cheerleader team in action. Broadly, the development work to produce the kind of performance shown here can be split into three areas:

- **Robotic hardware**. We use a total of seven robots, constructed from LEGO Mindstorms kits. Each cheerleader has a single RCX robot control brick (part of the LEGO Mindstorms package) that it uses to control infrared communication and the motors that move its wheels and arms.

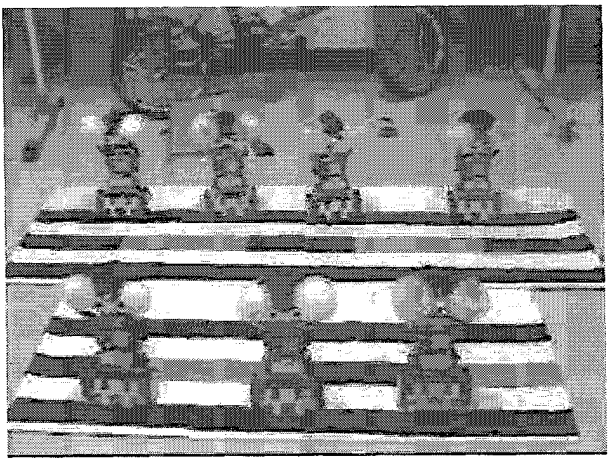

Figure 1. Seven cheerleader robots performing

- **Environmental hardware.** Since we only use very basic robotic hardware (e.g., with no high-level functions such as location sensing) we augment the environment with extra information. First, the field is marked with stripes that the robots can sense to adjust their positions. Second, we place two extra RCX blocks at the rear of the field to produce an infrared communication channel that covers the field, thus supporting individual robots with weak infrared emission or sensing.

- **Software Control.** A key feature in our approach to producing collaboration is abstraction of control. As the number of robots grows, maintaining separate programs for each individual can become increasingly difficult. Therefore, we abstract the essence of the robots' collaboration into a higher-order scripting language. From this higher-order language we can automatically produce programs that control the lower-level actions of the robots.

We describe our work on each of these three components in the following sections. We then give a brief qualitative evaluation of our current system, and discuss further work.

3. Robotic Hardware

Figure 2 shows the appearance of our robots. Each robot has a single RCX robot control brick. This brick is used to control the other parts of the robot, which for our purposes are: two motors, two light sensors,

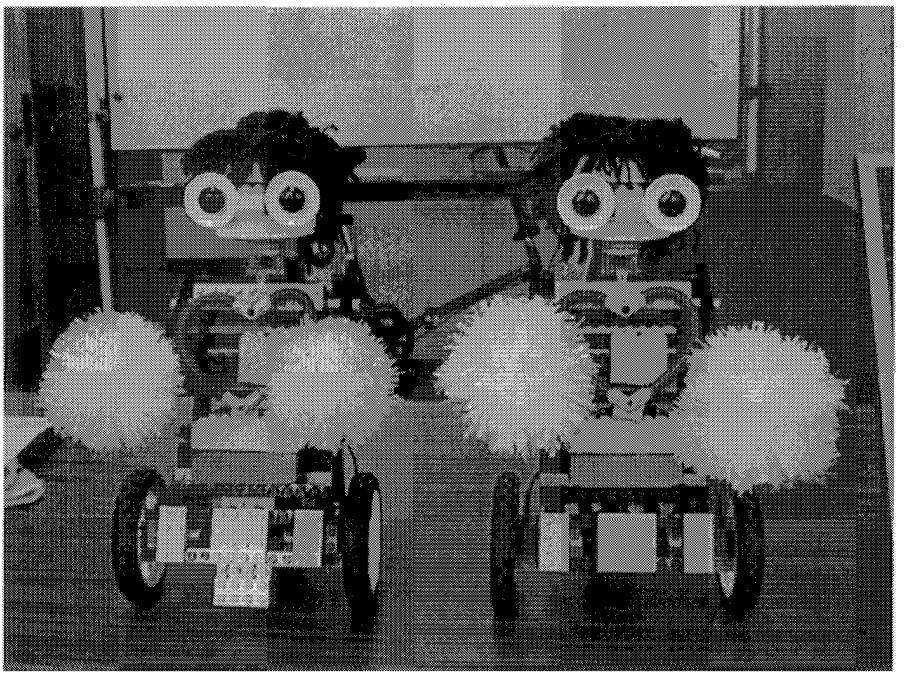

Figure 2. Two cheerleader robots

a touch sensor and an infrared communication sensor. The robots use one motor to power the two wheels on either side of their bodies and the other to raise and lower their arms. The light sensors help the robots with positioning by pointing downwards so that they can detect the stripes that form part of the environment.

The robots have eight basic actions: raise arms high, lower arms, vibrate arms, change positions, go back to the home position, turn around, turn right, and turn left. All dances are composed of combinations of these movements. There is one "leader" robot among the seven (in the middle of the front row), who commands the other six by monitoring their movements and sending commands over the infrared communication channel. All the robots follow these commands, and when they have completed them they report their success back to the leader (again, via the infrared channel). After each session of a dance, all the robots look for their home position by going back and forth. After all have found their home positions, they can restart the next session of the dance.

Although the cheerleaders are all built to exactly the same plan, each robot has its own idiosyncrasies. For example, one has a tendency to drift right when it moves, while another has poor infrared power. To cope with these differences we developed our control algorithms so that they could be easily parameterised. For each robot, we then created an initialisation file that could be incorporated at compile time to produce tailored control instructions.

4. Environmental Hardware

With just the basic hardware in a LEGO Mindstorms kit it is hard to produce robots that can carry out localisation, or reliably communicate in a domain that may contain obstructions that disrupt infrared communication. Thus, to produce a robust cheerleading team, we addressed these issues by augmenting the environment itself.

The task of localisation was the most difficult to solve, since the deviations in the movement of the LEGO robots is enough to quickly make teamwork impossible without error correction. We decided to tackle this problem by the simple expedient of placing thick lines on the floor of the performance area, which the robots could track with light sensors. Initially, we tried to create a pattern of lines that the robots would simply follow. However, we found that this made for jerky movements, and also restricted considerably the possible dances that could be performed. The current pattern of two sets of three thick horizontal stripes shown in Figure 1 allows for far more versatile positioning behaviour. Note that the middle stripe is interrupted by two silver bars. These bars are used to define the home positions for the robots. We tried vertical instead of horizontal stripes, but we found that the current configuration produced the best performance. With the current stripes, the robots can guide themselves to one of 5 different "vertical" locations.

We dealt with the problems of infrared communication by simply placing two additional RCX blocks at the rear of the performance area. These blocks enable two robots to communicate with each other even if the direct line-of-sight path between them is blocked.

5. Software Control

The software for producing collaborative behaviours in the robots has two stages. First there is a script language that describes the overall actions that the robots can take. A five-minute dance can be described in just a few lines of this script language. Second, there is a compiler that transforms scripts into programs in Dave Baum's NQC (" Not Quite

C") language and then attaches any required libraries and hardware difference initialisation files.

5.1. The Script Language

The script language defines actions in terms of *components*. One component of a dance is described on one line as follows:

```
robot_identifier actions
```

The `robot_identifier` may be a comma-separated list, in which case all the specified robots carry out the component simultaneously. For example,

```
1,2,3 turn arm-up
```

means that robots 1, 2 and 3 turn and then raise their arms together.

In order to make two distinct components occur in parallel, there is a *parallel block* command that requires all the components to happen at the same time. For example:

```
par
  1,2,3 turn arm-up
  6,7 arm-vibrate
parend
```

means that robots 1, 2, and 3 turn and then raise their arms, whilst at the same time robots 6 and 7 vibrate their arms up and down. Since robots 4 and 5 receive no commands, they will do nothing during this parallel block.

To maintain synchronisation, we designed our robots to report to leader robot after completing a given component. Only once all the robots report (or a pre-set timeout expires) does the action move on to the next line of the dance. In the case of a par block, this behaviour is extended to require reports from all the robots within the block (for instance, robots 1, 2, 3, 6 and 7 in the example above). However, synchronisation is costly in general. Therefore, in order to facilitate more complex sequences of actions without repeated synchronisation, we implemented a higher-order function *wave*. It is written as:

```
wave(robot_identifier ;  actions ;  time_offset)
```

The robots indicated by the `robot_identifier` perform the `actions` one after another at intervals of `time_offset`. For example,

```
wave(1,2,3,4,5,6,7; arm_up; 10)
```

means that first robot 1 raises its arms, then 10 msec later, robot 2 raises its arms, then after another 10 msec robot 3 raises its arms and so on. This means that the script lines `wave(1,2,3; arm_up; 0)` and `1,2,3 arm_up` are have the same effect, as does a parallel block containing the three components `1 arm_up`,`2 arm_up`, and `3 arm_up`.

In addition to the basic components described above, the script language also allows control structures such as iteration or the definition of functions.

5.2. Compilation

The compiler takes a script and outputs seven programs: one for each cheerleader. Since the current script language has a straightforward control structure with no variants, the actual translation is uncomplicated. The output programs produced by the compiler are initially in NQC. To these programs, the compiler then attaches the necessary libraries, which consist of two components; the definition of actions and the signalling and monitoring functions needed for synchronisation. These libraries are also written in NQC.

The compiler next adjusts any parameters according to each robot's initialisation file, and then gives a unique identifier number to each component of the dance. The identifier is what enables robots to rejoin the dance after losing their position. A failed robot is detected by the leader robot when it does not receive a component completion signal within a pre-set time. When a failed robot restarts again, the leader gives it the identifier of the current dance component so that the robot can take up the performance again.

The NQC programs resulting from the above steps are compiled into assembly language for the RCX blocks. The performance begins when a start message is sent by a control computer to the leader robot.

6. Qualitative Evaluations

We performed two simple qualitative evaluations of our cheerleaders. In the first, we tested the ability of the cheerleader team to maintain synchronisation over long periods. We prepared 2 dances, each of about 3 minutes in length, and performed them in a loop with the cheerleaders. In five trials of one hour each, the team was able to perform continuously with no stoppages and without losing synchronisation.

For the second evaluation we investigated the ability of individual robots to recover from failure. Since our robots do not often fail, we simulated failure by deliberately removing a robot from an active performance. We then replaced the robot after some time, and counted

the percentage of times that the robot could find its home position and rejoin the dance. We found that the robot restarted in about 70% of the trials.

7. Conclusion and Future Work

We have presented a robot cheerleading team made from LEGO Mindstorms kits. The seven independent robots perform a cheerleading dance while synchronising with each other automatically in real time. The robots are tolerant to faults, so that additional robots can join or leave the dance at any time without disrupting the remainder. Further, we described how control was abstracted to the level of a high-level script, which could then be compiled into a low-level language for controlling each individual in the team.

We regard our work as a first step towards understanding how people and robots can interact in larger numbers than just one-on-one. At the technical level, we showed that even simple hardware can be used to produce robust synchronisation among multiple robots. As part of our future work, we plan to investigate the interface between the cheerleading team and external factors. For example, we are planning to present our cheerleading team in the RoboCup 2002, which offers possibilities of linking the team's actions to a game, and also of interacting with an audience. Another possible direction is to include speech as part of the performance. We are now studying a script language to handle this natural language generation.

References

[1] Honda. Asimo home page, 2000, http://www.honda.co.jp/asimo.
[2] J. Knudsen. The Unofficial Guide to LEGO Mindstorms robots. O' Reilly Press, 1999
[3] Maja J Mataric, Behavior-based multi-robot coordination research, 2000. http://robotics.usc.edu/maja/group.html
[4] B. Reeves and C. Nass. The Media Equation, CSLI Publications, 1006
[5] RoboCup, Robocup home page, 2000. http://www.robocup.com
[6] Sony Aibo home page, 2000. http://sony.co.jp/aibo